FORMATUX

Ma formation Linux

Antoine Le Morvan, Xavier Sauvignon

Version 2.0.1 du 24 septembre 2019

Table des matières

Préface

GNU/Linux est un **système d'exploitation** libre fonctionnant sur la base d'un **noyau Linux**, également appelé **kernel Linux**.

Linux est une implémentation libre du système **UNIX** et respecte les spécifications **POSIX**.

GNU/Linux est généralement distribué dans un ensemble cohérent de logiciels, assemblés autour du noyau Linux et prêt à être installé. Cet ensemble porte le nom de "**Distribution**".

- La plus ancienne des distributions est la distribution **Slackware**.
- Les plus connues et utilisées sont les distributions **Debian**, **RedHat** et **Arch**, et servent de base pour d'autres distributions comme **Ubuntu**, **CentOS**, **Fedora**, **Mageia** ou **Manjaro**.

Chaque distribution présente des particularités et peut être développée pour répondre à des besoins très précis :

- services d'infrastructure ;
- pare-feu ;
- serveur multimédia ;
- serveur de stockage ;
- etc.

La distribution présentée dans ces pages est principalement la CentOS, qui est le pendant gratuit de la distribution RedHat, mais la plupart des supports s'appliquent généralement aussi à Ubuntu et Debian. La distribution CentOS est particulièrement adaptée pour un usage sur des serveurs d'entreprises.

Crédits

Ce support de cours a été rédigé sur plusieurs années par de nombreux formateurs.

Les formateurs à l'origine du projet et contributeurs principaux actuels :

- **Antoine Le Morvan** ;
- **Xavier Sauvignon** ;

Notre relecteur principal des premières versions :

- **Patrick Finet** ;

Il a rédigé la partie Git :

- **Carl Chenet**

Ils ont contribués à la rédaction :

- **Nicolas Kovacs** : Apache sous CentOS7 ;
- **Damien Dubédat** : Scripting shell, Fondamentaux ;
- …

Enfin, les illustrations de qualité sont dessinées pour formatux par **François Muller** (aka `Founet`).

L'histoire de Formatux

Nous étions (Xavier, Antoine et Patrick) tous les trois formateurs dans la même école de formation pour adulte.

L'idée de fournir aux stagiaires un support en PDF reprenant la totalité des cours dispensés pour leur permettre de réviser et approfondir les points vus en classe pendant la journée nous a rapidement paru être une priorité.

En décembre 2015, nous testions les solutions qui existaient pour rédiger un support d'une telle taille, et nous avons retenu dès le début du projet le format Asciidoc pour sa simplicité et le générateur Asciidoctor pour la qualité du support généré, la possibilité de personnaliser le rendu mais surtout pour l'étendue de ses fonctionnalités. Nous avons également testé le markdown, mais avons été plus rapidement limité.

5 ans après le début du projet et après avoir basculé notre site web sous Antora, nous ne regrettons absolument pas le choix technique d'Asciidoc.

La gestion des sources a été confiée dès l'année suivante à la forge Gitlab de Framagit, ce qui nous permettait de travailler à plusieurs en même temps sur le support, et de faciliter la relecture du support par Patrick. En découvrant la CI de gitlab-ci, nous avions enfin la stack qui nous permettrait d'automatiser totalement la génération du support. Il ne nous manquait plus que la génération d'un site web depuis ces mêmes sources.

Le travail de rédaction étant fortement personnel, sachant que nous risquions d'être muté rapidement dans les années à suivre, nous avons voulu ce support sous Licence Libre, afin qu'un maximum de personnes puissent à la fois contribuer et en profiter, et que notre beau support ne se perde pas.

Même après avoir tous quitté notre organisme de formation, nous avons continué à faire vivre le projet formatux, en faisant évoluer certaines parties, en nous appuyant sur d'autres pour nos formations et en intégrant de nouvelles parties, comme la partie Git de Carl Chenet.

En juillet 2019, nous (Xavier et Antoine, Patrick ayant pris sa retraite informatique) avons décidé de reprendre le développement de Formatux plus activement et d'en faire un peu plus sa promotion. L'organisation complète du support a été revue, en le scindant en 8 dépôts disctincts, correspondant à chacune des parties, au support global ainsi qu'au site web. Nous avons voulu notre organisation full devops, afin que la génération de chacune des parties soient totalement automatisées et inter-

dépendantes les unes des autres.

Il est difficile aujourd'hui d'évaluer la popularité de notre support. Ce dernier a longtemps été disponible en téléchargement par torrent sur freetorrent (malheureusement aujourd'hui disparu) et en direct depuis le site web. Nous n'avons pas de métriques et nous n'en voulons pas particulièrement. Nous retirons notre satisfaction dans les contacts que nous avons avec nos lecteurs.

Licence

Formatux propose des supports de cours Linux à destination des formateurs ou des personnes désireuses d'apprendre à administrer un système Linux en autodidacte.

Les supports de Formatux sont publiés sous licence Creative Commons-BY-SA et sous licence Art Libre. Vous êtes ainsi libre de copier, de diffuser et de transformer librement les œuvres dans le respect des droits de l'auteur.

BY : Paternité. Vous devez citer le nom de l'auteur original.

SA : Partage des Conditions Initiales à l'Identique.

- Licence Creative Commons-BY-SA : https://creativecommons.org/licenses/by-sa/3.0/fr/
- Licence Art Libre : http://artlibre.org/

Les documents de Formatux et leurs sources sont librement téléchargeables sur formatux.fr :

- https://www.formatux.fr/

Les sources de nos supports sont hébergés chez Framasoft sur leur forge Framagit. Vous y trouverez les dépôts des codes sources à l'origine de la version de ce document :

- https://framagit.org/formatux/

A partir des sources, vous pouvez générer votre support de formation personnalisé. Nous vous recommandons le logiciel AsciidocFX téléchargeable ici : http://asciidocfx.com/ ou l'éditeur Atom avec les plugins Asciidoc.

Comment contribuer au projet ?

Si vous voulez participer à la rédaction des supports formatux, **forkez-nous sur framagit.org**.

Vous pourrez ensuite apporter vos modifications, compiler votre support personnalisé et nous proposer vos modifications.

Vous êtes les bienvenus pour :

- Compléter le document avec un nouveau chapitre,

- Corriger ou compléter les chapitres existants,
- Relire le document et corriger l'orthographe, la mise en forme,
- Promouvoir le projet

De votre côté

1. Créer un compte sur https://framagit.org,
2. Créer un fork du projet que vous voulez modifier parmi la liste des projets du groupe : Créer le fork,
3. Créer une branche nommée develop/[Description],
    - Où [Description] est une description très courte de ce qui va être fait.
4. Faire des commits dans votre branche,
5. Pusher la branche sur votre fork,
6. Demander une merge request.

Si vous n'êtes pas un grand utilisateur de git, ce n'est pas grave. Vous pouvez toujours lire la partie Git de notre support puis nous contacter. Nous vous guiderons ensuite pour vos premiers pas.

Cette remarque est également vraie pour le format Asciidoc.

Essayer de conserver le même ton qui est déjà employé dans le reste du support (pas de 'je' ni de smileys par exemple).

La suite se passe de notre côté

1. Quelqu'un relira votre travail,
    - Essayez de rendre ce travail plus facile en organisant vos commits.
2. S'il y a des remarques sur le travail, le relecteur fera des commentaires sur la merge request,
3. Si la merge request lui semble correcte il peut merger votre travail avec la branche **develop**.

Corrections suite à une relecture

La relecture de la merge request peut vous amener à faire des corrections. Vous pouvez faire ces corrections dans votre branche, ce qui aura pour effet de les ajouter à la merge request.

Comment compiler mon support formatux ?

Après avoir forké notre projet ou l'avoir cloné (`git clone https://framagit.org/group/formatux-PARTIEXX.git`), déplacez-vous dans le dossier formatux-PARTIEXX nouvellement créé.

Vous avez ensuite plusieurs possibilités :

- Vous utilisez le logiciel AsciiDocFX ou Atom avec ses plugins Asciidoc (recommandé sous Windows) : lancez le logiciel, ouvrez le fichier `.adoc` désiré, et cliquez sur le bouton **PDF** (AsciidoctorFX) ou regardez la prévisualisation (Atom).
- Vous êtes sous Linux et vous avez déjà installé le paquet asciidoctor : exécutez la commande `asciidoctor-pdf -t -D public -o support.pdf support.adoc`.

Comment faire un don

Vous pouvez faire un don par paypal pour soutenir nos efforts. Ces dons serviront à payer notre nom de domaine et notre hébergement. Nous pourrons également reverser une partie de ces dons à Framasoft, qui héberge gracieusement nos repos et met à disposition un runner qui compile nos supports et notre site web. Enfin, une petite bière entre nous peut également être au programme.

Accès à la cagnotte paypal : https://www.paypal.com/pools/c/8hlM1Affp1.

Nous contacter

Nous sommes disponibles sur gitter pour échanger autour de formatux, de Linux et de la formation : https://gitter.im/formatux-fr/formatux.

Antora

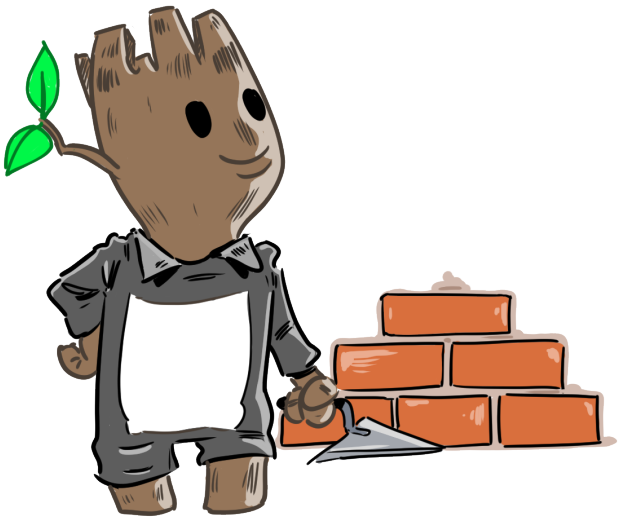
Pour la génération de notre site web, nous utilisons Antora. Antora nous permet, depuis les mêmes sources, de pouvoir générer le support en PDF et le site web statique en HTML. Le développement d'Antora est sponsorisé par OpenDevise Inc.

Gestion des versions

Table 1. Historique des versions du document

Version	Date	Observations
1.0	Avril 2017	Version initiale.
1.1	Juin 2017	Ajout des cours Nginx et Php-fpm.
1.2	Juin 2017	Ajout des cours MySQL et MySQL Master/Master.
1.3	Juillet 2017	Ajout de généralités devops.
1.4	Aout 2017	Ajout du cours Apache LAMP sous CentOS 7 (Nicolas Kovacs).
1.5	Aout 2017	Ajout du cours Jenkins et Rundeck.
1.6	Février 2019	Ajout des cours Ansible Niveau 2, Ansistrano, Asciidoc, Terraform, Varnish
2.0	Septembre 2019	Passage au format antora - Ajout des articles Git
2.0.1	Septembre 2019	Relecture du cours Bash

Partie 1 : Administration.



Dans cette première partie de notre support Formatux, nous vous proposons de découvrir, ou redécouvrir **les notions essentielles de l'administration Linux**.

Dans un premier temps, nous discuterons de **Linux**, des distributions, et de tout l'écosystème autour de notre système d'exploitation.

Nous verrons ensuite, les **commandes utilisateurs** indispensables à la prise en main de Linux. Les utilisateurs plus aguérris pourront également consulter le chapitre consacré aux commandes un peu plus "avancées".

Un chapitre qui nous semble toujours aussi important vient ensuite : la **prise en main de l'éditeur** `VI`, ou tout au moins sa démystification : savoir ouvrir un ficher, enregistrer, quitter ou quitter sans enregistrer. Pour les autres fonctionnalités, l'utilisateur sera plus à l'aise avec elles au fur et à mesure de son utilisation de cet éditeur pas comme les autres mais tellement puissant. Un mémo des commandes `VI` est disponible dans notre partie Annexes du support global.

Nous pourrons ensuite entrer dans le vif du sujet et dans le fonctionnement de Linux pour découvrir comment le système gère :

- les **utilisateurs**,
- les **systèmes de fichiers**,
- les **processus**.

Nous ferons un aparté sur les **sauvegardes**, sujet toujours si important qu'il est indispensable d'en parler : nous ne ferons jamais assez de sauvegardes. Nous découvrirons deux outils : `tar` mais également `cpio`, moins répandu, qui dans certains cas peut être très intéressant.

Nous reviendrons à la gestion du système avec le **démarrage**, qui a énormément évolué ces dernières années depuis l'arrivée de `systemd`. Le chapitre *Démarrage du système (init)* traitera des versions legacy toujours bien présentes dans les parcs informatiques (debian 7, RHEL 6) tandis que le chapitre *Démarrage du système sous CentOS 7 (systemd)* traitera des versions plus modernes (debian 8, RHEL 7) et permettra à l'administrateur système de se mettre à niveau sur ce sujet.

Nous conclurons cette partie en étudiant la **gestion des tâches**, la **mise en oeuvre du réseau** et l'**installation des logiciels**.

Bonne lecture.

Chapitre 1. Introduction au système d'exploitation Linux

1.1. Qu'est-ce qu'un système d'exploitation ?

Linux est un **système d'exploitation**.

Un système d'exploitation est un **ensemble de programmes permettant la gestion des ressources disponibles d'un ordinateur**.

Parmi cette gestion des ressources, le système d'exploitation est amené à :

- Gérer la mémoire physique ou virtuelle.
  - La **mémoire physique** est composée des barrettes de mémoires vives et de la mémoire cache du processeur, qui sert pour l'exécution des programmes.
  - La **mémoire virtuelle** est un emplacement sur le disque dur (la partition **swap**) qui permet de décharger la mémoire physique et de sauvegarder l'état en cours du système durant l'arrêt électrique de l'ordinateur (hibernation du système).
- Intercepter les **accès aux périphériques**. Les logiciels ne sont que très rarement autorisés à accéder directement au matériel (à l'exception des cartes graphiques pour des besoins très spécifiques).
- Offrir aux applications une **gestion correcte des tâches**. Le système d'exploitation est responsable de l'ordonnancement des processus pour l'occupation du processeur.
- **Protéger les fichiers** contre tout accès non autorisé.
- **Collecter les informations** sur les programmes utilisés ou en cours d'utilisation.

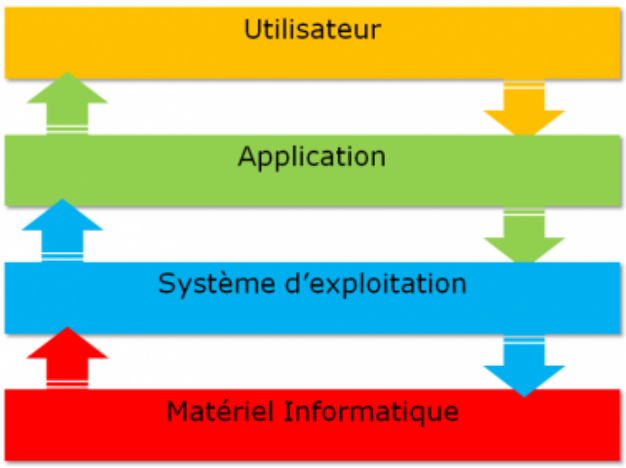


Figure 1. Fonctionnement d'un système d'exploitation

1.2. Généralités UNIX - GNU/Linux

Historique

UNIX

- De **1964 à 1968** : MULTICS (MULTiplexed Information and Computing Service) est développé pour le compte du MIT, des laboratoires Bell Labs (AT&T) et de General Electric.
- **1969** : Après le retrait de Bell (1969) puis de General Electric du projet, deux développeurs (Ken Thompson et Dennis Ritchie), rejoints plus tard par Brian Kernighan, jugeant MULTICS trop complexe, lancent le développement d'UNIX (UNiplexed Information and Computing Service). À l'origine développé en assembleur, les concepteurs d'UNIX ont développé le langage B puis le langage C (1971) et totalement réécrit UNIX. Ayant été développé en 1970, la date de référence des systèmes UNIX/Linux est toujours fixée au 01 janvier 1970.

Le langage C fait toujours partie des langages de programmation les plus populaires aujourd'hui ! Langage de bas niveau, proche du matériel, il permet l'adaptation du système d'exploitation à toute architecture machine disposant d'un compilateur C.

UNIX est un système d'exploitation ouvert et évolutif ayant joué un rôle primordial dans l'histoire de l'informatique. Il a servi de base pour de nombreux autres systèmes : Linux, BSD, Mac OSX, etc.

UNIX est toujours d'actualité (HP-UX, AIX, Solaris, etc.)

Minix

- **1987** : Minix. A.S. Tanenbaum développe MINIX, un UNIX simplifié, pour enseigner les systèmes d'exploitation de façon simple. M. Tanenbaum rend disponible les sources de son système d'exploitation.

Linux

Figure 2. Linus Torvald, créateur du noyau Linux

- **1991** : Linux. Un étudiant finlandais, Linus Torvalds, crée un système d'exploitation dédié à son ordinateur personnel et le nomme Linux. Il publie sa première version 0.02, sur le forum de discussion Usenet et d'autres développeurs viennent ainsi l'aider à améliorer son système. Le

terme Linux est un jeu de mot entre le prénom du fondateur, Linus, et UNIX.

- **1993** : La distribution Debian est créée. Debian est une distribution non commerciale a gestion associative. À l'origine développée pour une utilisation sur des serveurs, elle est particulièrement bien adaptée à ce rôle, mais elle se veut être un système universel et donc utilisable également sur un ordinateur personnel. Debian est utilisée comme base pour de nombreuses autres distributions, comme Mint ou Ubuntu.
- **1994** : La distribution commerciale RedHat est créée par la société RedHat, qui est aujourd'hui le premier distributeur du système d'exploitation GNU/Linux. RedHat soutient la version communautaire Fedora et depuis peu la distribution libre CentOS.
- **1997** : L'environnement de bureau KDE est créé. Il est basé sur la bibliothèque de composants Qt et sur le langage de développement C++.
- **1999** : L'environnement de bureau Gnome est créé. Il est quant à lui basé sur la bibliothèque de composants GTK+.
- **2002** : La distribution Arch est créée. Sa particularité est d'être diffusée en Rolling Release (mise à jour en continue).
- **2004** : Ubuntu est créée par la société Canonical (Mark Shuttleworth). Elle se base sur Debian, mais regroupe des logiciels libres et privateurs.

Parts de marché

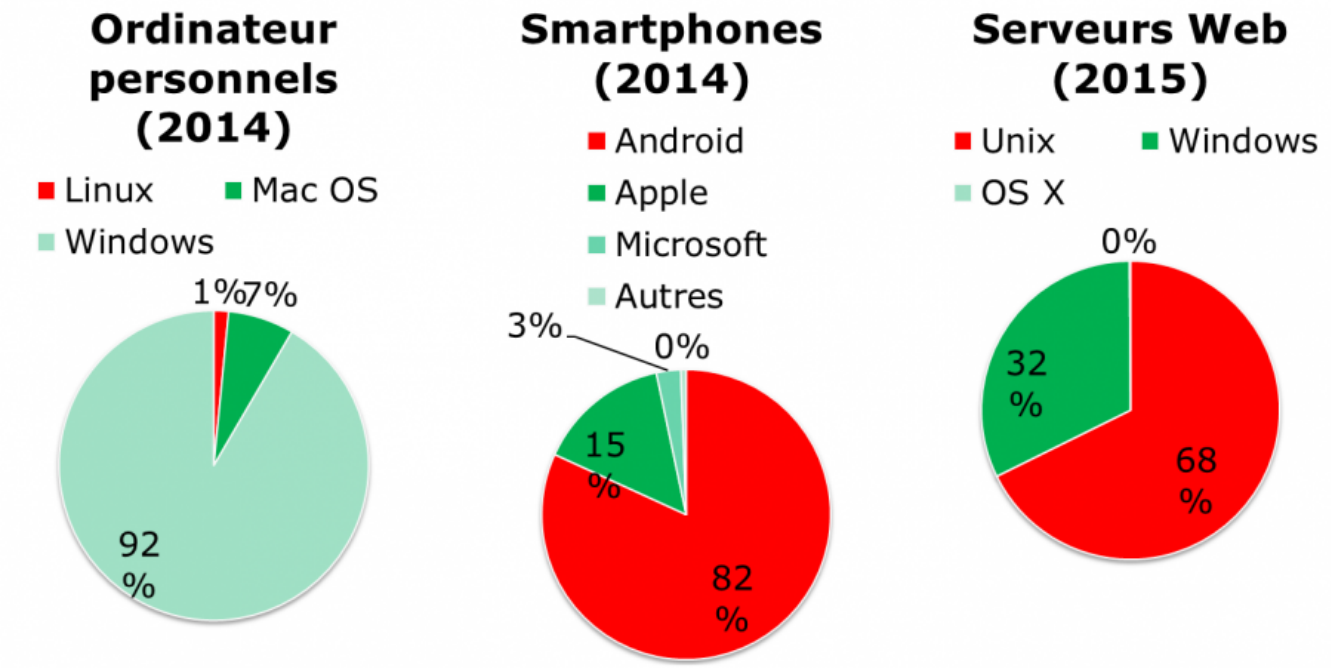


Figure 3. Les parts de marché de Linux

Linux est finalement encore peu connu du grand public, alors que ce dernier l'utilise régulièrement. En effet, Linux se cache dans les **smartphones**, les **téléviseurs**, les **box internet**, etc. Presque **70% des pages web** servies dans le monde le sont par un serveur Linux ou UNIX !

Linux équipe un peu plus d'**1,5% des ordinateurs personnels** mais plus de **82% des smartphones**. **Android** étant un système d'exploitation dont le kernel est un Linux.

Architecture

- Le noyau (ou kernel) est le premier composant logiciel.
  - Il est le cœur du système UNIX.
  - C'est lui qui gère les ressources matérielles du système.
  - Les autres composants logiciels passent obligatoirement par lui pour accéder au matériel.
- Le Shell est un utilitaire qui interprète les commandes de l'utilisateur et assure leur exécution.
  - Principaux shell : Bourne shell, C shell, Korn shell et Bourne Again shell (bash).
- Les applications regroupent les programmes utilisateurs comme :
  - le navigateur internet ;
  - le traitement de texte ;
  - ...

Multitâche

Linux fait partie de la famille des systèmes d'exploitation à temps partagé. Il partage le temps d'utilisation processus entre plusieurs programmes, passant de l'un à l'autre de façon transparente pour l'utilisateur. Cela implique :

- exécution simultanée de plusieurs programmes ;
- distribution du temps CPU par l'ordonnanceur ;
- réduction des problèmes dus à une application défaillante ;
- diminution des performances lorsqu'il y a trop de programmes lancés.

Multiutilisateurs

La finalité de Multics était de permettre à plusieurs utilisateurs de travailler à partir de plusieurs terminaux (écran et clavier) sur un seul ordinateur (très coûteux à l'époque). Linux étant un descendant de ce système d'exploitation, il a gardé cette capacité à pouvoir fonctionner avec plusieurs utilisateurs simultanément et en toute indépendance, chacun ayant son compte utilisateur, son espace de mémoire et ses droits d'accès aux fichiers et aux logiciels.

Multiprocesseur

Linux est capable de travailler avec des ordinateurs multiprocesseurs ou avec des processeurs multicœurs.

Multiplateforme

Linux est écrit en langage de haut niveau pouvant s'adapter à différents types de plate-formes lors de la compilation. Il fonctionne donc sur :

- les ordinateurs des particuliers (le PC ou l'ordinateur portable) ;

- les serveurs (données, applications,...) ;
- les ordinateurs portables (les smartphones ou les tablettes) ;
- les systèmes embarqués (ordinateur de voiture) ;
- les éléments actifs des réseaux (routeurs, commutateurs) ;
- les appareils ménagers (téléviseurs, réfrigérateurs,...).

Ouvert

Linux se base sur des standards reconnus (posix, TCP/IP, NFS, Samba ...) permettant de partager des données et des services avec d'autres systèmes d'applications.

La philosophie UNIX

- Tout est fichier.
- Portabilité.
- Ne faire qu'une seule chose et la faire bien.
- KISS : Keep It Simple and Stupid.
- "UNIX est simple, il faut juste être un génie pour comprendre sa simplicité" (*Dennis Ritchie*)
- "UNIX est convivial. Cependant UNIX ne précise pas vraiment avec qui." (*Steven King*)

1.3. Les distributions GNU/LINUX

Une distribution Linux est un **ensemble cohérent de logiciels** assemblés autour du noyau Linux et prêt à être installé. Il existe des distributions **associatives ou communautaires** (Debian, CentOS) ou **commerciales** (RedHat, Ubuntu).

Chaque distribution propose un ou plusieurs **environnements de bureau**, fournit un ensemble de logiciels pré-installés et une logithèque de logiciels supplémentaires. Des options de configuration (options du noyau ou des services par exemple) sont propres à chacune.

Ce principe permet d'avoir des distributions orientées **débutants** (Ubuntu, Linux Mint ...) ou d'une approche plus complexe (Gentoo, Arch), destinées à faire du **serveur** (Debian, RedHat, ...) ou dédiées à des **postes de travail**.

Les environnements de bureaux

Les environnements graphiques sont nombreux : **Gnome**, **KDE**, LXDE, XFCE, etc. Il y en a pour tous les goûts, et leurs **ergonomies** n'ont pas à rougir de ce que l'on peut retrouver sur les systèmes Microsoft ou Apple !

Alors pourquoi si peu d'engouement pour Linux, alors qu'il n'**existe pas (ou presque pas) de virus pour ce système** ? Parce que tous les éditeurs (Adobe) ou constructeur (NVidia) ne jouent pas le jeu du libre et ne fournissent pas de version de leurs logiciels ou de leurs drivers pour GNU/Linux. Trop

peu de jeux également sont (mais plus pour longtemps) distribués sous Linux.

La donne changera-t-elle avec l'arrivée de la steam-box qui fonctionne elle aussi sous Linux ?

Figure 4. L'environnement de bureau Gnome

L'environnement de bureau **Gnome 3** n'utilise plus le concept de Bureau mais celui de Gnome Shell (à ne pas confondre avec le shell de la ligne de commande). Il sert à la fois de bureau, de tableau de bord, de zone de notification et de sélecteur de fenêtre. L'environnement de bureau Gnome se base sur la bibliothèque de composants GTK+.

Figure 5. L'environnement de bureau KDE

L'environnement de bureau **KDE** se base sur la bibliothèque de composants **Qt**.

Il est traditionnellement plus conseillé aux utilisateurs venant d'un monde Windows.

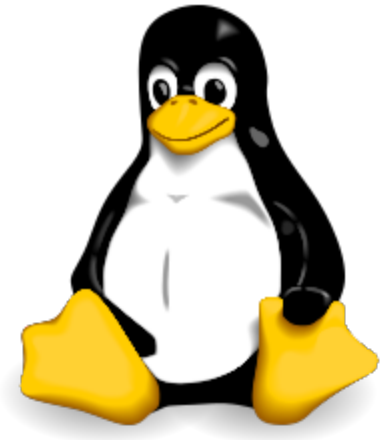


Figure 6. Tux, la mascotte Linux

Libre / Open source

Un utilisateur de système d'exploitation Microsoft ou Mac doit s'affranchir d'une licence d'utilisation du système d'exploitation. Cette licence a un coût, même s'il est généralement transparent (le prix de la licence étant inclus dans le prix de l'ordinateur).

Dans le monde GNU/Linux, le mouvement du Libre permet de fournir des distributions gratuites.

Libre ne veut pas dire gratuit !

Open source : les codes sources sont disponibles, il est donc possible de les consulter, de les modifier et de le diffuser.

Licence GPL (General Public License)

La Licence GPL garantit à l'auteur d'un logiciel sa propriété intellectuelle, mais autorise la modification, la rediffusion ou la revente de logiciels par des tiers, sous condition que les codes sources soient fournis avec le logiciel. La licence GPL est la licence issue du projet GNU (GNU is Not UNIX), projet déterminant dans la création de Linux.

Elle implique :

- la liberté d'exécuter le programme, pour tous les usages ;
- la liberté d'étudier le fonctionnement du programme et de l'adapter aux besoins ;
- la liberté de redistribuer des copies ;
- la liberté d'améliorer le programme et de publier vos améliorations, pour en faire profiter toute la communauté.

Par contre, même des produits sous licences GPL peuvent être payants. Ce n'est pas le produit en lui-même mais la garantie qu'une équipe de développeurs continue à travailler dessus pour le faire évoluer et dépanner les erreurs, voire fournir un soutien aux utilisateurs.

1.4. Les domaines d'emploi

Une distribution Linux excelle pour :

- **Un serveur** : HTTP, messagerie, groupware, partage de fichiers, etc.
- **La sécurité** : Passerelle, pare-feu, routeur, proxy, etc.
- **Ordinateur central** : Banques, assurances, industrie, etc.
- **Système embarqué** : Routeurs, Box Internet, SmartTV, etc.

Linux est un choix adapté pour l'hébergement de base de données ou de sites web, ou comme serveur de messagerie, DNS, pare-feu. Bref Linux peut à peu près tout faire, ce qui explique la quantité de distributions spécifiques.

1.5. Shell

Généralités

Le shell, interface de commandes en français, permet aux utilisateurs d'envoyer des ordres au système d'exploitation. Il est moins visible aujourd'hui, depuis la mise en place des interfaces graphiques, mais reste un moyen privilégié sur les systèmes Linux qui ne possèdent pas tous des

interfaces graphiques et dont les services ne possèdent pas toujours une interface de réglage.

Il offre un véritable langage de programmation comprenant les structures classiques : boucles, alternatives et les constituants courants : variables, passage de paramètres, sous-programmes. Il permet donc la création de scripts pour automatiser certaines actions (sauvegardes, création d'utilisateurs, surveillance du système,...).

Il existe plusieurs types de Shell disponibles et configurables sur une plate-forme ou selon le choix préférentiel de l'utilisateur :

- sh, le shell aux normes POSIX ;
- csh, shell orienté commandes en C ;
- bash, Bourne Again Shell, shell de Linux.
- etc, ...

1.6. Fonctionnalités

- Exécution de commandes (vérifie la commande passée et l'exécute) ;
- Redirections Entrées/Sorties (renvoi des données dans un fichier au lieu de l'inscrire sur l'écran) ;
- Processus de connexion (gère la connexion de l'utilisateur) ;
- Langage de programmation interprété (permettant la création de scripts) ;
- Variables d'environnement (accès aux informations propres au système en cours de fonctionnement).

Principe

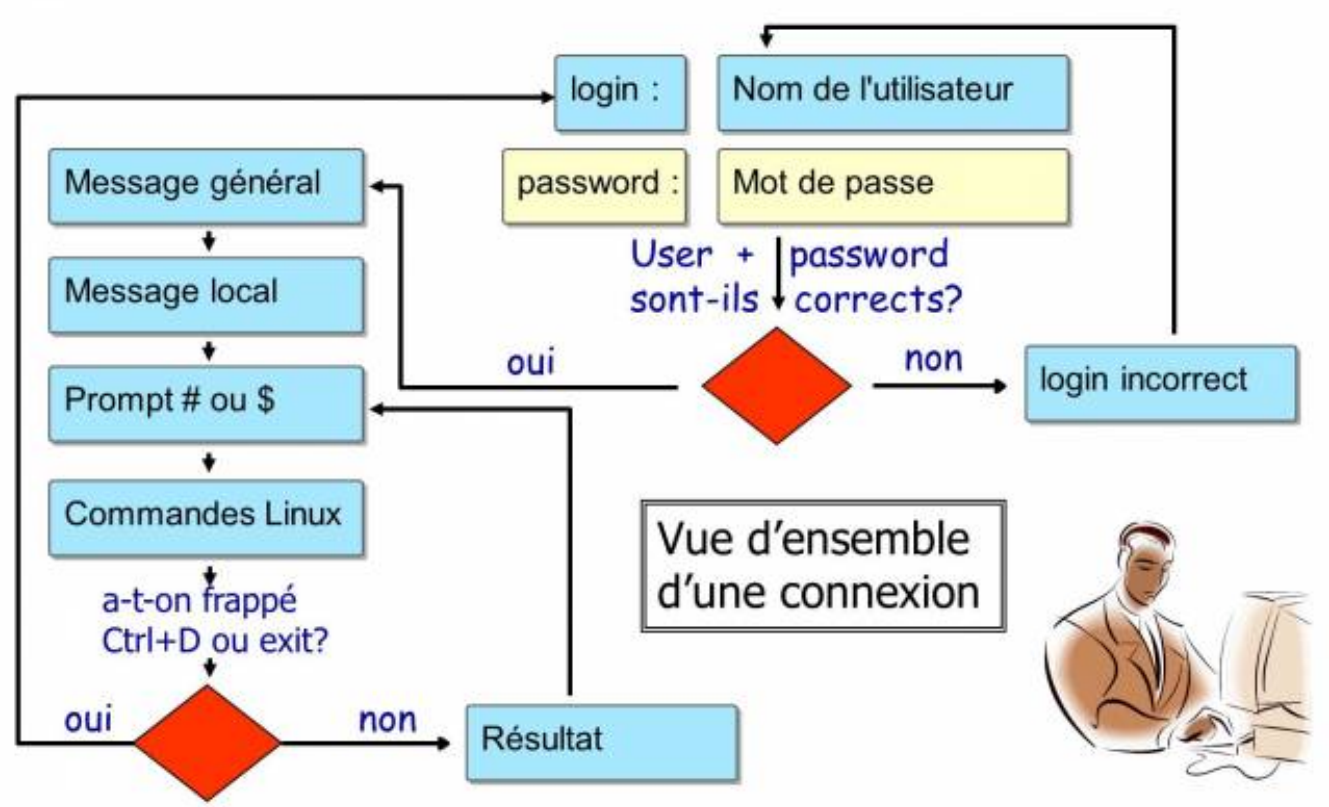


Figure 7. Principe de fonctionnement du SHELL

Chapitre 2. Commandes pour utilisateurs Linux

🎓 Objectifs

Apprendre aux futurs administrateurs à :

✔ se **déplacer** dans l'arborescence du système ;
✔ **créer** un fichier texte, **afficher** son contenu et le **modifier** ;
✔ utiliser les commandes les plus utiles de Linux.

2.1. Généralités

Les systèmes Linux actuels possèdent des utilitaires graphiques dédiés au travail d'un administrateur. Toutefois, il est important d'être capable d'utiliser l'interface en mode ligne de commandes et cela pour plusieurs raisons :

- La majorité des commandes du système sont communes à toutes les distributions Linux, ce qui n'est pas le cas des outils graphiques.
- Il peut arriver que le système ne démarre plus correctement mais qu'un interpréteur de commandes de secours reste accessible.
- L'administration à distance se fait en ligne de commandes avec un terminal SSH.
- Afin de préserver les ressources du serveur, l'interface graphique n'est soit pas installée, soit lancée à la demande.
- L'administration se fait par des scripts.

L'apprentissage de ces commandes permet à l'administrateur de se connecter à un terminal Linux, de gérer ses ressources, ses fichiers, d'identifier la station, le terminal et les utilisateurs connectés, etc.

Les utilisateurs

L'utilisateur du système Linux est défini (dans le fichier `/etc/passwd`) par :

- un **nom de connexion**, plus communément appelé « login », ne contenant pas d'espace ;
- un identifiant numérique : **UID** (User Identifier) ;
- un identifiant de groupe : **GID** (Group Identifier) ;
- un **mot de passe**, qui sera chiffré avant d'être stocké ;
- un **interpréteur de commandes**, un shell, qui peut être différent d'un utilisateur à l'autre ;
- un **répertoire de connexion**, le *home directory* ;

- une **invite de commande**, ou *prompt* de connexion, qui sera symbolisée par un `#` pour les administrateurs et un `$` pour les autres utilisateurs.

En fonction de la politique de sécurité mise en œuvre sur le système, le mot de passe devra comporter un certain nombre de caractères et respecter des exigences de complexité.

Parmi les interpréteurs de commandes existants, le **Bourne Again Shell** (`/bin/bash`) est celui qui est le plus fréquemment utilisé. Il est affecté par défaut aux nouveaux utilisateurs. Pour diverses raisons, des utilisateurs avancés de Linux choisiront des interpréteurs de commandes alternatifs parmi le Korn Shell (`ksh`), le C Shell (`csh`), etc.

Le répertoire de connexion de l'utilisateur est par convention stocké dans le répertoire `/home` du poste de travail. Il contiendra les données personnelles de l'utilisateur. Par défaut, à la connexion, le répertoire de connexion est sélectionné comme répertoire courant.

Une installation type poste de travail (avec interface graphique) démarre cette interface sur le terminal 1. Linux étant multi-utilisateurs, il est possible de connecter plusieurs utilisateurs plusieurs fois, sur des **terminaux physiques** (TTY) ou **virtuels** (PTS) différents. Les terminaux virtuels sont disponibles au sein d'un environnement graphique. Un utilisateur bascule d'un terminal physique à l'autre à l'aide des touches *Alt+Fx* depuis la ligne de commandes ou à l'aide des touches *Ctrl+Alt+Fx*.

Le shell

Une fois que l'utilisateur est connecté sur une console, le shell affiche l'invite de commandes (`prompt`). Il se comporte ensuite comme une boucle infinie, à chaque saisie d'instruction :

- affichage de l'invite de commande ;
- lecture de la commande ;
- analyse de la syntaxe ;
- substitution des caractères spéciaux ;
- exécution de la commande ;
- affichage de l'invite de commande ;
- etc.

La séquence de touche *Ctrl+C* permet d'interrompre une commande en cours d'exécution.

L'utilisation d'une commande respecte généralement cette séquence :

Séquence d'une commande

```
commande [option(s)] [arguments(s)]
```

Le nom de la commande est **toujours en minuscules**.

Un espace sépare chaque élément.

Les **options courtes** commencent par un tiret (`-l`), alors que les **options longues** commencent par deux tirets (`--list`). Un double tiret (`--`) indique la fin de la liste d'options. Il est possible de regrouper certaines options courtes :

Options courtes

```
$ ls -l -i -a
```

est équivalent à :

Regroupement d'options

```
$ ls -lia
```

Il peut bien entendu y avoir plusieurs arguments après une option :

Arguments d'une commande

```
$ ls -lia /etc /home /var
```

Dans la littérature, le terme « option » est équivalent au terme « paramètre », plus utilisé dans le domaine de la programmation. Le côté optionnel d'une option ou d'un argument est symbolisé en le mettant entre crochets `[` et `]`. Lorsque plusieurs options sont possibles, une barre verticale appelée « pipe » les sépare `[a|e|i]`.

2.2. Les commandes générales

Les commandes man et whatis

Il est impossible pour un administrateur, quel que soit son niveau, de connaître toutes les commandes et options dans les moindres détails. Une commande a été spécialement conçue pour accéder en ligne de commandes à un ensemble d'aides, sous forme d'un manuel : la commande `man` (« le man est ton ami »).

Ce manuel est divisé en 8 sections, regroupant les informations par thèmes, la section par défaut étant la section 1 :

1. Commandes utilisateurs ;
2. Appels système ;
3. Fonctions de bibliothèque C ;
4. Périphériques et fichiers spéciaux ;
5. Formats de fichiers ;

6. Jeux ;
7. Divers ;
8. Outils d'administration système et démons.

Des informations sur chaque section sont accessibles en saisissant `man x intro`, `x` indiquant le numéro de section.

La commande :

Syntaxe de la commande man

```
$ man passwd
```

informera l'administrateur sur la commande `passwd`, ses options, etc. Alors qu'un :

Syntaxe de la commande man avec section

```
$ man 5 passwd
```

l'informera sur les fichiers en relations avec la commande.

Toutes les pages du manuel ne sont pas traduites de l'anglais. Elles sont toutefois généralement très précises et fournissent toutes les informations utiles. La syntaxe utilisée et le découpage peuvent dérouter l'administrateur débutant, mais avec de la pratique, l'administrateur y retrouvera rapidement l'information qu'il recherche.

La navigation dans le manuel se fait avec les flèches ¬ et ¬. Le manuel se quitte en appuyant sur la touche *q*.

La commande `whatis` permet de faire une recherche par mot clef au sein des pages de manuel :

Syntaxe de la commande whatis

```
$ whatis clear
```

La commande shutdown

La commande `shutdown` permet de **stopper électriquement**, immédiatement ou après un certain laps de temps, un serveur Linux.

Syntaxe de la commande shutdown

```
[root]# shutdown [-h] [-r] heure [message]
```

L'heure d'arrêt est à indiquer au format `hh:mm` pour une heure précise, ou `+mm` pour un délai en minutes.

Pour forcer un arrêt immédiat, le mot `now` remplacera l'heure. Dans ce cas, le message optionnel n'est pas envoyé aux autres utilisateurs du système.

- Exemples :

Exemples de la commande shutdown

```
[root]# shutdown -h 0:30 "Arrêt du serveur à 0h30"
[root]# shutdown -r +5
```

- Options :

Table 2. Options de la commande shutdown

Options	Observations
`-h`	Arrête le système électriquement
`-r`	Redémarre le système

La commande history

La commande `history` permet d'afficher l'historique des commandes qui ont été saisies par l'utilisateur.

Les commandes sont mémorisées dans le fichier `.bash_history` du répertoire de connexion de l'utilisateur.

Exemple de commande history

```
$ history
  147 man ls
  148 man history
```

Table 3. Options de la commande history

Options	Commentaires
`-w`	L'option `-w` permet d'y copier l'historique de la session en cours.
`-c`	L'option `-c` effacera l'historique de la session en cours (mais pas le contenu du fichier `.bash_history`).

- Manipuler l'historique :

Pour manipuler l'historique, des commandes permettent depuis l'invite de commandes de :

Touches	Fonction
`!!`	Rappeler la dernière commande passée.

Touches	Fonction
`!n`	Rappeler la commande par son numéro dans la liste.
`!string`	Rappeler la commande la plus récente commençant par la chaîne de caractères.
`[↑]`	Remonter l'historique des commandes.
`[↓]`	Redescendre l'historique des commandes.

L'auto-complétion

L'auto-complétion est également d'une aide précieuse.

- Elle permet de compléter les commandes, les chemins saisis ou les noms de fichiers.
- Un appui sur la touche *TAB* complète la saisie dans le cas d'une seule solution.
- Sinon, il faudra faire un deuxième appui pour obtenir la liste des possibilités.

Si un double appui sur la touche *TAB* ne provoque aucune réaction de la part du système, c'est qu'il n'existe aucune solution à la complétion en cours.

2.3. Affichage et identification

La commande clear

La commande `clear` permet d'effacer le contenu de l'écran du terminal. En réalité, pour être plus précis, elle permet de décaler l'affichage de sorte que l'invite de commandes se retrouve en haut de l'écran sur la première ligne.

Dans un terminal, l'affichage sera définitivement masqué tandis que dans une interface graphique, un ascenseur permettra de remonter dans l'historique du terminal virtuel.

La commande echo

La commande `echo` permet d'afficher une chaîne de caractères.

Cette commande est plus particulièrement utilisée dans les scripts d'administration pour informer l'utilisateur pendant l'exécution.

L'option `-n` permet de ne pas revenir à la ligne après avoir affiché le texte (ce qui est le comportement par défaut de la commande).

Pour diverses raisons, le développeur du script peut être amené à utiliser des séquences spéciales (commençant par un caractère `\`). Dans ce cas, l'option `-e` sera stipulée, permettant l'interprétation des séquences.

Parmi les séquences fréquemment utilisées, nous citerons :

Table 4. Séquences spéciales de la commande echo

Séquence	Résultat
`\a`	Émet un bip sonore
`\b`	Retour en arrière
`\n`	Ajoute un saut de ligne
`\t`	Ajoute une tabulation horizontale
`\v`	Ajoute une tabulation verticale

La commande date

La commande `date` permet d'afficher la date et l'heure. La commande respecte la syntaxe suivante :

Syntaxe de la commande date

```
date [-d AAAAMMJJ] [format]
```

Exemples :

```
$ date
mer. Avril 17 16:46:53 CEST 2013
$ date -d 20150729 +%j
210
```

Dans ce dernier exemple, l'option `-d` affiche une date donnée. L'option `+%j` formate cette date pour n'afficher que le quantième.

Le format d'une date peut changer suivant la valeur de la langue définie dans la variable d'environnement `$LANG`.

L'affichage de la date peut suivre les formats suivants :

Table 5. Formats de la commande date

Option	Format
`+%A`	Nom complet du jour
`+%B`	Nom complet du mois
`+%c`	Affichage complet de la date
`+%d`	Numéro du jour
`+%F`	Date au format `AAAA-MM-JJ`
`+%G`	Année

Option	Format
`+%H`	Heure
`+%j`	Quantième du jour
`+%m`	Numéro du mois
`+%M`	Minute
`+%R`	Heure au format `hh:mm`
`+%s`	Secondes depuis le 1er janvier 1970
`+%T`	Heure au format `hh:mm:ss`
`+%u`	Jour de la semaine (`1` pour lundi)
`+%V`	Numéro de la semaine
`+%x`	Date au format `JJ/MM/AAAA`

La commande `date` permet également de modifier la date et l'heure système. Dans ce cas, l'option `-s` sera utilisée.

```
[root]# date -s "2013-04-17 10:19"
jeu. Avril 17 10:19:00 CEST 2013
```

Le format à respecter pour l'argument suivant l'option `-s` est celui-ci :

```
date -s "[AA]AA-MM-JJ hh:mm:[ss]"
```

Les commandes id, who et whoami

La commande `id` affiche le nom de l'utilisateur courant et ses groupes ou ceux d'un utilisateur, si le login de celui-ci est fourni comme argument.

```
$ id util1
uid=501(util1) gid=501(group1) groups=501(group1),502(group2)
```

Les options `-g`, `-G`, `-n` et `-u` affichent respectivement le GID du groupe principal, les GID des groupes secondaires, les noms au lieu des identifiants numériques et l'UID de l'utilisateur.

La commande `whoami` affiche le login de l'utilisateur courant.

La commande `who` seule affiche le nom des utilisateurs connectés :

```
$ who
stagiaire   tty1    2014-09-15 10:30
root        pts/0   2014-09-15 10:31
```

Linux étant multi-utilisateurs, il est probable que plusieurs sessions soient ouvertes sur la même station, que ce soit physiquement ou à travers le réseau. Il est intéressant de savoir quels utilisateurs sont connectés, ne serait-ce que pour communiquer avec eux par l'envoi de messages.

tty
: représente un terminal.

pts/
: représente une console virtuelle sous environnement graphique.

L'option « `-r` » affiche en plus le niveau d'exécution (voir chapitre « démarrage »).

2.4. Arborescence de fichiers

Sous Linux, l'arborescence des fichiers se présente sous la forme d'un arbre inversé, appelé **arborescence hiérarchique unique**, dont la racine est le répertoire « `/` ».

Le **répertoire courant** est le répertoire où se trouve l'utilisateur.

Le **répertoire de connexion** est le répertoire de travail associé à l'utilisateur. Les répertoires de connexion sont, en standard, stockés dans le répertoire `/home`.

À la connexion de l'utilisateur, le répertoire courant est le répertoire de connexion.

Un **chemin absolu** référence un fichier depuis la racine en parcourant l'arborescence complète jusqu'au niveau du fichier :

- `/home/groupeA/alice/monfichier`

Le **chemin relatif** référence ce même fichier en parcourant l'arborescence complète depuis le répertoire courant :

- `../alice/monfichier`

Dans l'exemple précèdent, les "`..`" font référence au répertoire parent du répertoire actuel.

Un répertoire, même s'il est vide, contiendra obligatoirement au minimum **deux références** :

« `.` »
: référence sur lui-même.

« `..` »
: référence le répertoire parent du répertoire actuel.

Un chemin relatif peut ainsi commencer par « `./` » ou par « `../` ». Lorsque le chemin relatif fait référence à un sous dossier ou à un fichier du répertoire courant, alors le « `./` » est souvent omis. Mentionner le premier « `./` » de l'arborescence ne sera réellement requis que pour lancer un fichier exécutable.

Les erreurs dans les chemins peuvent être la cause de nombreux problèmes : création de dossier ou de fichiers aux mauvais endroits, suppressions involontaires, etc. Il est donc fortement recommandé d'utiliser l'auto-complétion (cf. 2.2) lors des saisies de chemin.

Figure 8. Notre arborescence exemple

Dans l'exemple ci-dessus, nous cherchons à donner l'emplacement du fichier monfichier depuis le répertoire de bob.

- Par un **chemin absolu**, le répertoire courant importe peu. Nous commençons par la racine, pour descendre successivement dans les répertoires "`home`", "`groupeA`", "`alice`" et enfin le fichier "`monfichier`" : **`/home/groupeA/alice/monfichier`**.
- Par un **chemin relatif**, notre point de départ étant le répertoire courant "`bob`", nous remontons d'un niveau par "`..`" (soit dans le répertoire `groupeA`), puis nous descendons dans le répertoire "`alice`", et enfin le fichier "`monfichier`" : **`../alice/monfichier`**.

La commande pwd

La commande `pwd` (Print Working Directory) affiche le chemin absolu du répertoire courant.

```
$ pwd
/home/stagiaire
```

Pour se déplacer à l'aide d'un chemin relatif, il faut impérativement connaître son positionnement dans l'arborescence.

Selon l'interpréteur de commandes, l'invite de commandes peut également afficher le nom du répertoire courant.

La commande cd

La commande `cd` (Change Directory) permet de changer le répertoire courant, autrement dit, de se déplacer dans l'arborescence.

```
$ cd /tmp
$ pwd
/tmp
$ cd ../
$ pwd
/
$ cd
$ pwd
/home/stagiaire
```

Comme vous pouvez le constater dans le dernier exemple ci-dessus, la commande `cd` sans argument permet de repositionner le répertoire courant sur le répertoire de connexion (*home directory*).

La commande ls

La commande `ls` affiche le contenu d'un répertoire.

Syntaxe de la commande ls

```
ls [-a] [-i] [-l] [repertoire1] [repertoire2] [···]
```

Exemple :

```
$ ls /home
.   ..  stagiaire
```

Les options principales de la commande `ls` sont :

Table 6. Options principales de la commande ls

Option	Information
`-a`	Affiche tous les fichiers, même ceux cachés. Les fichiers cachés sous Linux sont ceux qui commencent par un "`.`".
`-i`	Affiche les numéros d'inode.
`-l`	Affiche sous forme de liste verticale la liste des fichiers avec des informations supplémentaires formatées par colonnes.

La commande ls offre toutefois de très nombreuses options (voir le man) :

Table 7. Options complémentaires de la commande ls

Option	Information
`-d`	Affiche les informations d'un répertoire au lieu de lister son contenu.
`-g`	Affiche les UID et GID plutôt que les noms des propriétaires.
`-h`	Affiche les tailles de fichiers dans le format le plus adapté (octet, kilo-octet, méga-octet, giga-octet, ...). `h` pour Human Readable.
`-s`	Affiche la taille en octets (sauf si option `k`).
`-A`	Affiche tous les fichiers du répertoire sauf "`.`" et "`..`".
`-R`	Affiche récursivement le contenu des sous répertoires.
`-F`	Affiche le type des fichiers. Imprime un `/` pour un répertoire, `*` pour les exécutables, `@` pour un lien symbolique, et rien pour un fichier texte.
`-X`	Trier les fichiers en fonction de leurs extensions.

- Description des colonnes :

```
$ ls -lia /home
78489 drwx------ 4 stagiaire users 4096 25 oct. 08:10 stagiaire
```

Table 8. Description des colonnes du résultat généré par la commande ls

Valeur	Information.
`78489`	Numéro d'inode.
`drwx------`	Type de fichier (`d`) et droits (`rwx------`).
`4`	Nombre de sous-répertoires ("`.`" et "`..`" inclus). Pour un fichier de type lien physique : nombre de liens physiques.
`stagiaire`	Utilisateur propriétaire.
`users`	Groupe propriétaire.
`4096`	Taille en octets.
`25 oct. 08:10`	Date de dernière modification.
`stagiaire`	Nom du fichier (ou du répertoire).

Des **alias** sont fréquemment positionnés au sein des distributions courantes.

C'est le cas de l'alias `ll` :

Alias de la commande ls -l

```
alias ll='ls -l --color=auto'
```

La commande `ls` dispose de nombreuses options dont voici quelques exemples avancés

d'utilisations :

- Lister les fichiers de `/etc` par ordre de dernière modification :

```
$ ls -ltr /etc
total 1332
-rw-r--r--.  1 root root    662 29 aout   2007 logrotate.conf
-rw-r--r--.  1 root root    272 17 nov.   2009 mailcap
-rw-------.  1 root root    122 12 janv.  2010 securetty
...
-rw-r--r--.  2 root root     85 18 nov.  17:04 resolv.conf
-rw-r--r--.  1 root root     44 18 nov.  17:04 adjtime
-rw-r--r--.  1 root root    283 18 nov.  17:05 mtab
```

- Lister les fichiers de `/var` plus gros qu'un méga-octet mais moins qu'un giga-octets :

```
[root]# ls -Rlh /var | grep [0-9]M
...
-rw-r--r--. 1 apache apache 1,2M 10 nov.  13:02 XB RiyazBdIt.ttf
-rw-r--r--. 1 apache apache 1,2M 10 nov.  13:02 XB RiyazBd.ttf
-rw-r--r--. 1 apache apache 1,1M 10 nov.  13:02 XB RiyazIt.ttf
...
```

- Afficher les droits sur un dossier :

Pour connaître les droits sur un dossier, dans notre exemple `/etc`, la commande suivante ne conviendrait pas :

```
$ ls -l /etc
total 1332
-rw-r--r--.  1 root root     44 18 nov.  17:04 adjtime
-rw-r--r--.  1 root root   1512 12 janv.  2010 aliases
-rw-r--r--.  1 root root  12288 17 nov.  17:41 aliases.db
drwxr-xr-x.  2 root root   4096 17 nov.  17:48 alternatives
...
```

puisque cette dernière liste par défaut le contenu du dossier et non le contenant.

Pour ce faire, il faut utiliser l'option `-d` :

```
$ ls -ld /etc
drwxr-xr-x. 69 root root 4096 18 nov.  17:05 /etc
```

- Lister les fichiers par taille :

```
$ ls -lhS
```

- Afficher la date de modification au format “timestamp” :

```
$ ls -l --time-style="+%Y-%m-%d $newline%m-%d %H:%M" /
total 12378
dr-xr-xr-x. 2 root root 4096 2014-11-23 11-23 03:13 bin
dr-xr-xr-x. 5 root root 1024 2014-11-23 11-23 05:29 boot
```

- Ajouter le “trailing slash” à la fin des dossiers :

Par défaut, la commande `ls` n’affiche pas le dernier slash d’un dossier.

Dans certains cas, comme pour des scripts par exemple, il est utile de les afficher :

```
$ ls -dF /etc
/etc/
```

- Masquer certaines extensions :

```
$ ls /etc --hide=*.conf
```

La commande mkdir

La commande `mkdir` crée un répertoire ou une arborescence de répertoire.

Syntaxe de la commande mkdir

```
mkdir [-p] repertoire [repertoire] [⋯]
```

Exemple :

```
$ mkdir /home/stagiaire/travail
```

Le répertoire « `stagiaire` » devra exister pour créer le répertoire « `travail` ».

Sinon, l’option « `-p` » devra être utilisée. L’option « `-p` » crée les répertoires parents s’ils n’existent pas.

Il est vivement déconseillé de donner des noms de commandes UNIX comme noms de répertoires ou fichiers.

La commande touch

La commande `touch` modifie l'horodatage d'un fichier ou crée un fichier vide si le fichier n'existe pas.

Syntaxe de la commande touch

```
touch [-t date] fichier
```

Exemple :

```
$ touch /home/stagiaire/fichier
```

Option	Information
`-t date`	Modifie la date de dernière modification du fichier avec la date précisée. Date au format : `[AAAA]MMJJhhmm[ss]`

La commande touch est utilisée en priorité pour créer un fichier vide, mais elle peut avoir un intérêt dans le cadre de sauvegarde incrémentale ou différentielle. En effet, le fait d'exécuter un touch sur un fichier aura pour seul effet de forcer sa sauvegarde lors de la sauvegarde suivante.

La commande rmdir

La commande `rmdir` supprime un répertoire vide.

Exemple :

```
$ rmdir /home/stagiaire/travail
```

Option	Information
`-p`	Supprime le ou les répertoire(s) parent(s) à la condition qu'ils soient vides.

Pour supprimer à la fois un répertoire non-vide et son contenu, il faudra utiliser la commande `rm`.

La commande rm

La commande `rm` supprime un fichier ou un répertoire.

Syntaxe de la commande rm

```
rm [-f] [-r] fichier [fichier] [···]
```

ATTENTION !!! Toute suppression de fichier ou de répertoire est définitive.

Table 9. Options de la commande rm

Options	Information
`-f`	Ne demande pas de confirmation de la suppression.
`-i`	Demande de confirmation de la suppression.
`-r`	Supprime récursivement les sous-répertoires.

La commande `rm` en elle-même ne demande pas de confirmation lors de la suppression de fichiers. Ce comportement est propre à la distribution RedHat/CentOS.

La commande `rm` est ici un alias de la commande `rm -i`. Ne soyez pas surpris sur une autre distribution, type Debian par exemple, de ne pas obtenir de demande de confirmation.

La suppression d'un dossier à l'aide de la commande `rm`, que ce dossier soit vide ou non, nécessitera l'ajout de l'option `-r`.

La fin des options est signalée au shell par un double tiret "`--`".

Dans l'exemple :

```
$ >-dur-dur # Creer un fichier vide appelé -dur-dur
$ rm -f -- -dur-dur
```

Le nom du fichier `-dur-dur` commence par un "`-`". Sans l'usage du "`--`" le shell aurait interprété le "`-d`" de "`-dur-dur`" comme une option.

La commande mv

La commande `mv` déplace et renomme un fichier.

Syntaxe de la commande mv

```
mv fichier [fichier ···] destination
```

Exemples :

```
$ mv /home/stagiaire/fic1 /home/stagiaire/fic2
$ mv /home/stagiaire/fic1 /home/stagiaire/fic2 /tmp
```

Table 10. Options de la commande mv

Options	Information
`-f`	Ne demande pas de confirmation si écrasement du fichier de destination.
`-i`	Demande de confirmation si écrasement du fichier de destination (par défaut).

Quelques cas concrets permettront de mieux saisir les difficultés qui peuvent se présenter :

```
$ mv /home/stagiaire/fic1 /home/stagiaire/fic2
```

Permet de renommer “`fic1`” en “`fic2`”, si “`fic2`” existe déjà, il sera remplacé par “`fic1`”.

```
$ mv /home/stagiaire/fic1 /home/stagiaire/fic2 /tmp
```

Permet de déplacer “`fic1`” et “`fic2`” dans le répertoire “`/tmp`”.

```
$ mv fic1 /repexiste/fic2
```

« `fic1` » est déplacé dans « `/repexiste` » et renommé « `fic2` ».

```
$ mv fic1 fic2
```

« `fic1` » est renommé « `fic2` ».

```
$ mv fic1 /repexiste
```

Si le répertoire de destination existe, « `fic1` » est déplacé dans « `/repexiste` ».

```
$ mv fic1 /repexistepas
```

Si le répertoire de destination n’existe pas, « `fic1` » est renommé « `repexistepas` » à la racine.

La commande cp

La commande `cp` copie un fichier.

Syntaxe de la commande cp

```
cp fichier [fichier ⋯] destination
```

Exemple :

```
$ cp -r /home/stagiaire /tmp
```

Table 11. Options de la commande cp

Options	Information
`-i`	Demande de confirmation si écrasement (par défaut).
`-f`	Ne demande pas de confirmation si écrasement du fichier de destination.
`-p`	Conserve le propriétaire, les permissions et l'horodatage du fichier copié.
`-r`	Copie un répertoire avec ses fichiers et sous-répertoires.

Quelques cas concrets permettront de mieux saisir les difficultés qui peuvent se présenter :

```
$ cp fic1 /repexiste/fic2
```

« `fic1` » est copié dans « `/repexiste` » sous le nom « `fic2` ».

```
$ cp fic1 fic2
```

« `fic1` » est copié sous le nom « `fic2` » dans ce répertoire.

```
$ cp fic1 /repexiste
```

Si le répertoire de destination existe, « `fic1` » est copié dans « `/repexiste` ».

```
$ cp fic1 /repexistepas
```

Si le répertoire de destination n'existe pas, « `fic1` » est copié sous le nom « `repexistepas` ».

2.5. Visualisation

La commande file

La commande `file` affiche le type d'un fichier.

Syntaxe de la commande file

```
file fichier [fichiers]
```

Exemple :

```
$ file /etc/passwd /etc
/etc/passwd:    ASCII text
/etc:       directory
```

La commande more

La commande `more` affiche le contenu d'un ou de plusieurs fichiers écran par écran.

Syntaxe de la commande more

```
more fichier [fichiers]
```

Exemple :

```
$ more /etc/passwd
root:x:0:0:root:/root:/bin/bash
...
```

En utilisant la touche *ENTREE*, le déplacement se fait ligne par ligne. En utilisant la touche *ESPACE*, le déplacement se fait page par page.

La commande less

La commande `less` affiche le contenu d'un ou de plusieurs fichiers. La commande `less` est interactive et possède des commandes d'utilisation qui lui sont propres.

Syntaxe de la commande less

```
less fichiers [fichiers]
```

Les commandes propres à `less` sont :

Table 12. Commandes internes à less

Commande	Action
`h`	Aide.
`Flèches`	Monter, descendre d'une ligne ou pour aller à droite ou à gauche.

Commande	Action
`Entrée`	Descendre d'une ligne.
`Espace`	Descendre d'une page.
`PgAR` ou `PgAV`	Monter ou descendre d'une page.
`Pos1` ou `Fin`	Se placer en début de fichier ou en fin de fichier.
`/texte`	Rechercher le texte.
`q`	Quitter la commande less.

La commande cat

La commande `cat` concatène (mettre bout à bout) le contenu de plusieurs fichiers et affiche le résultat sur la sortie standard.

Syntaxe de la commande cat

```
cat fichier [fichiers]
```

Exemple 1 - Afficher le contenu d'un fichier vers la sortie standard :

```
$ cat /etc/passwd
```

Exemple 2 - Afficher le contenu de plusieurs fichiers vers la sortie standard :

```
$ cat /etc/passwd /etc/group
```

Exemple 3 - Afficher le contenu de plusieurs fichiers et rediriger la sortie standard :

```
$ cat /etc/passwd /etc/group > utilisateursEtGroupes.txt
```

Exemple 4 - Afficher la numérotation des lignes :

```
$ cat -n /etc/passwd
 1 root:x:0:0:root:/root:/bin/bash
 2 bin:x:1:1:bin:/bin:/sbin/nologin
...
```

Exemple 5 - Affiche la numérotation des lignes non vides :

```
$ cat -b /etc/openldap/ldap.conf
1 #
2 # LDAP Defaults
3 #

4 # See ldap.conf(5) for details
5 # This file should be world readable but not world writable
```

La commande tac

La commande `tac` fait quasiment l'inverse de la commande `cat`. Elle affiche le contenu d'un fichier en commençant par la fin (ce qui est particulièrement intéressant pour la lecture des logs !).

Exemple : Afficher un fichier de logs en affichant en premier la dernière ligne :

```
[root]# tac /var/log/messages | less
```

La commande head

La commande `head` affiche le début d'un fichier.

Syntaxe de la commande head

```
head [-n x] fichier
```

Table 13. Options de la commande head

Option	Observation
`-n x`	Affiche les x premières lignes du fichier

Par défaut (sans l'option `-n`), la commande head affichera les 10 premières lignes du fichier.

La commande tail

La commande `tail` affiche la fin d'un fichier.

Syntaxe de la commande tail

```
tail [-f] [-n x] fichier
```

Table 14. Options de la commande tail

Option	Observation
`-n x`	Affiche les `x` dernières lignes du fichier

Option	Observation
`-f`	Affiche les modifications du fichier en temps réel

Exemple :

```
$ tail -n 3 /etc/passwd
sshd:x:74:74:Privilege-separeted sshd:/var/empty /sshd:/sbin/nologin
tcpdump::x:72:72::/:/sbin/nologin
user1:x:500:500:grp1:/home/user1:/bin/bash
```

Avec l'option `-f`, la commande tail ne rend pas la main et s'exécute tant que l'utilisateur ne l'interrompt pas par la séquence `[CTRL]` + `[C]`. Cette option est très fréquemment utilisée pour suivre les fichiers journaux (les logs) en temps réel.

Sans l'option `-n`, la commande tail affiche les 10 dernières lignes du fichier.

La commande sort

La commande `sort` trie les lignes d'un fichier.

Elle permet d'ordonner, ranger dans un ordre donné, le résultat d'une commande ou le contenu d'un fichier, selon un ordre numérique, alphabétique, par ordre de grandeur (Ko, Mo, Go) ou dans l'ordre inverse.

Syntaxe de la commande sort

```
sort [-kx] [-n] [-o fichier] [-ty] fichier
```

Exemple :

```
$ sort -k3 -t: -n /etc/passwd
root:x:0:0:root:/root:/bin/bash
adm:x:3:4:adm:/var/adm/:/sbin/nologin
```

Table 15. Options de la commande sort

Option	Observation
`-kx`	Précise la colonne `x` sur laquelle se fera le tri
`-n`	Demande un tri numérique
`-o fichier`	Enregistre le tri dans le fichier précisé
`-ty`	Précise le caractère séparateur de champs `y`
`-r`	Inverse l'ordre du résultat

La commande `sort` ne trie le fichier qu'à l'affichage écran. Le fichier n'est pas modifié par le tri. Pour enregistrer le tri, il faut utiliser l'option `-o` ou une redirection de sortie `>`.

Par défaut, le tri des nombres se fait selon leur caractère. Ainsi, "110" sera avant "20", qui sera lui-même avant "3". Il faut préciser l'option `-n` pour que les blocs caractères numériques soient bien triés par leur valeur.

La commande `sort` permet d'inverser l'ordre des résultats, avec l'option `-r` :

```
$ sort -k3 -t: -n -r /etc/passwd
root:x:0:0:root:/root:/bin/bash
adm:x:3:4:adm:/var/adm/:/sbin/nologin
```

Dans cet exemple, la commande `sort` rangera cette fois-ci le contenu du fichier `/etc/passwd` du plus grand uid au plus petit.

Quelques exemples avancés d'utilisation de la commande `sort` :

- Mélanger les valeurs

La commande `sort` permet également de mélanger les valeurs avec l'option `-R` :

```
$ sort -R /etc/passwd
```

- Trier des adresses IP

Un administrateur système est rapidement confronté au traitement des adresses IP issues des logs de ses services comme SMTP, VSFTP ou Apache. Ces adresses sont typiquement extraites avec la commande `cut`.

Voici un exemple avec le fichier `client-dns.txt` :

```
192.168.1.10
192.168.1.200
5.1.150.146
208.128.150.98
208.128.150.99
```

```
$ sort -nr client-dns.txt
208.128.150.99
208.128.150.98
192.168.1.200
192.168.1.10
5.1.150.146
```

- Trier des tailles de fichiers

La commande `sort` sait reconnaître les tailles de fichiers, issues de commande comme `ls` avec l'option `-h`.

Voici un exemple avec le fichier `taille.txt` :

```
1,7G
18M
69K
2,4M
1,2M
4,2G
6M
124M
12,4M
4G
```

```
[root]# sort -hr taille.txt
4,2G
4G
1,7G
124M
18M
12,4M
6M
2,4M
1,2M
69K
```

La commande wc

La commande `wc` compte le nombre de lignes, mots ou octets d'un fichier.

Syntaxe de la commande wc

```
wc [-l] [-m] [-w] fichier [fichiers]
```

Table 16. Options de la commande wc

Option	Observation
`-c`	Compte le nombre d'octets.
`-m`	Compte le nombre de caractères.
`-l`	Compte le nombre de lignes.

Option	Observation
`-w`	Compte le nombre de mots.

2.6. Recherche

La commande find

La commande `find` recherche l'emplacement d'un fichier.

Syntaxe de la commande find

```
find repertoire [-name nom] [-type type] [-user login] [-date date]
```

Les options de la commande `find` étant très nombreuses, il est préférable de se référer au `man`.

Si le répertoire de recherche n'est pas précisé, la commande `find` cherchera à partir du répertoire courant.

Table 17. Options de la commande find

Option	Observation
`-perm permissions`	Recherche des fichiers selon leurs permissions.
`-size taille`	Recherche des fichiers selon leur taille.

L'option -exec de la commande find

Il est possible d'utiliser l'option `-exec` de la commande `find` pour exécuter une commande à chaque ligne de résultat :

```
$ find /tmp -name *.txt -exec rm -f {} \;
```

La commande précédente recherche tous les fichiers du répertoire `/tmp` nommés `*.log` et les supprime.

Comprendre l'option -exec

Dans l'exemple ci-dessus, la commande find va construire une chaîne de caractères représentant la commande à exécuter.

Si la commande find trouve trois fichiers nommés `log1.txt`, `log2.txt` et `log3.txt`, alors la commande `find` va construire la chaîne en remplaçant dans la chaîne "`rm -f {} \;`" les accolades par un des résultats de la recherche, et cela autant de fois qu'il y a de résultats.

Ce qui nous donnera :

```
rm -f /tmp/log1.txt ; rm -f /tmp/log2.txt ; rm -f /tmp/log3.txt ;
```

Le caractère "`;`" est un caractère spécial du shell qui doit être protégé par un "`\`" pour éviter son interprétation trop tôt par la commande `find` (et non plus dans le `-exec`).

La commande whereis

La commande `whereis` recherche des fichiers liés à une commande.

Syntaxe de la commande whereis

```
whereis [-b] [-m] [-s] commande
```

Exemple :

```
$ whereis -b ls
ls: /bin/ls
```

Table 18. Options de la commande whereis

Option	Observation
`-b`	Ne recherche que le fichier binaire.
`-m`	Ne recherche que les pages de manuel.
`-s`	Ne recherche que les fichiers sources.

La commande grep

La commande `grep` recherche une chaîne de caractères dans un fichier.

Syntaxe de la commande grep

```
grep [-w] [-i] [-v] "chaîne" fichier
```

Exemple :

```
$ grep -w "root:" /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

Table 19. Options de la commande grep

Option	Observation
`-i`	Ignore la casse de la chaîne de caractères recherchée.
`-v`	Inverse le résultat de la recherche.
`-w`	Recherche exactement la chaîne de caractères précisée.

La commande `grep` retourne la ligne complète comprenant la chaîne de caractères recherchée.

- Le caractère spécial `^` permet de rechercher une chaîne de caractères placée en début de ligne.
- Le caractère spécial `$` permet de rechercher une chaîne de caractères placée en fin de ligne.

```
$ grep -w "^root" /etc/passwd
```

Cette commande est très puissante et il est fortement conseillé de consulter son manuel. Elle a de nombreux dérivés,

Il est possible de rechercher une chaîne de caractères dans une arborescence de fichiers avec l'option `-R`.

```
$ grep -R "Virtual" /etc/httpd
```

Les méta-caractères

Les méta-caractères se substituent à un ou plusieurs caractères (voire à une absence de caractère) lors d'une recherche.

Ils sont combinables.

Le caractère `*` remplace une chaîne composée de plusieurs caractères quelconques. Le caractère `*` peut également représenter une absence de caractère.

```
$ find /home -name test*
/home/stagiaire/test
/home/stagiaire/test1
/home/stagiaire/test11
/home/stagiaire/tests
/home/stagiaire/test362
```

Les méta-caractères permettent des recherches plus complexes en remplaçant tout ou partie d'un mot. Il suffit de remplacer les inconnues par ces caractères spéciaux.

Le caractère "?" remplace un unique caractère, quel qu'il soit.

```
$ find /home -name test?
/home/stagiaire/test1
/home/stagiaire/tests
```

Les crochets "[ ]" permettent de spécifier les valeurs que peut prendre un unique caractère.

```
$ find /home -name test[123]*
/home/stagiaire/test1
/home/stagiaire/test11
/home/stagiaire/test362
```

Il ne faut pas confondre les méta-caractères du shell et ceux des expressions régulières. La commande grep utilise les méta-caractères des expressions régulières.

2.7. Redirections et tubes

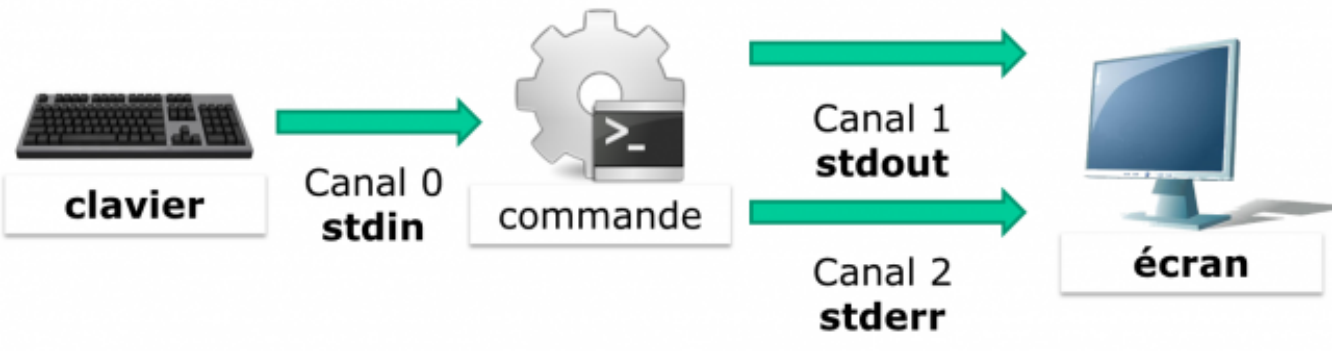
L'entrée et les sorties standards

Sur les systèmes UNIX et Linux, les flux standards sont aux nombres de trois. Ils permettent aux programmes, via la bibliothèque stdio.h de faire entrer ou sortir des informations.

Ces flux sont appelés canal X ou descripteur X de fichier.

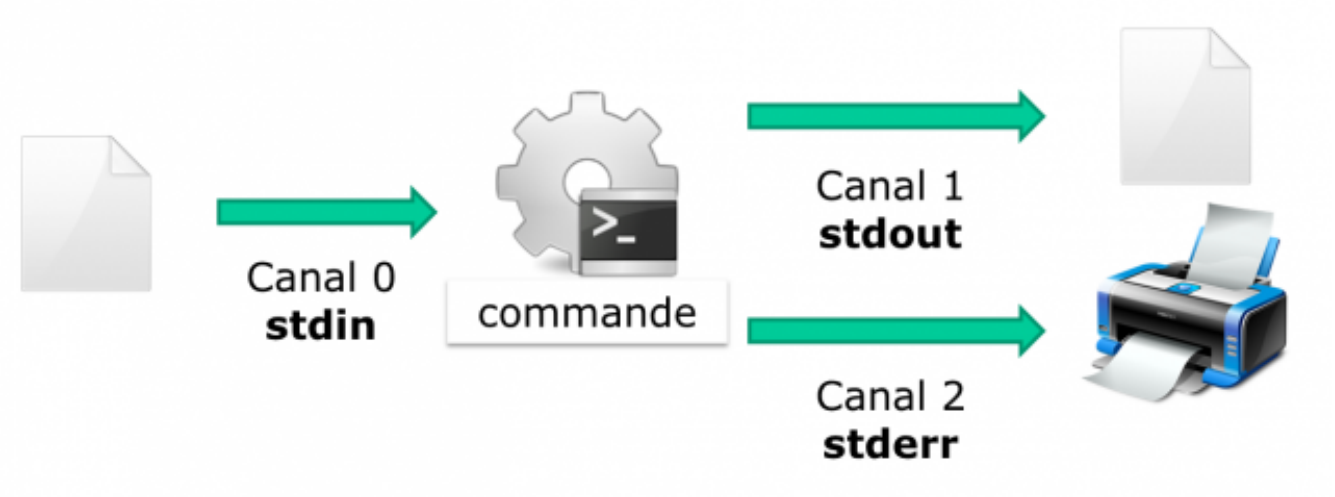
Par défaut :

- le clavier est le périphérique d'entrée pour le canal 0, appelé **stdin** ;
- l'écran est le périphérique de sortie pour les canaux 1 et 2, appelés **stdout** et **stderr**.



stderr reçoit les flux d'erreurs renvoyés par une commande. Les autres flux sont dirigés vers stdout.

Ces flux pointent vers des fichiers périphériques, mais comme tout est fichier sous UNIX, les flux d'entrées/sorties peuvent facilement être détournés vers d'autres fichiers. Ce principe fait toute la force du shell.



La redirection d'entrée

Il est possible de rediriger le flux d'entrée depuis un autre fichier avec le caractère inférieur < ou <<. La commande lira le fichier au lieu du clavier :

```
$ ftp -in serverftp << cdes-ftp.txt
```

Seules les commandes demandant une saisie au clavier pourront gérer la redirection d'entrée.

La redirection d'entrée peut également être utilisée pour simuler une interactivité avec l'utilisateur. La commande lira le flux d'entrée jusqu'à rencontrer le mot clef défini après la redirection d'entrée.

Cette fonctionnalité est utilisée pour scripter des commandes interactives :

```
$ ftp -in serverftp << FIN
user alice password
put fichier
bye
FIN
```

Le mot clef `FIN` peut être remplacé par n'importe quel mot.

```
$ ftp -in serverftp << STOP
user alice password
put fichier
bye
STOP
```

Le shell quitte la commande `ftp` lorsqu'il reçoit une ligne ne contenant que le mot clef.

La redirection de l'entrée standard est peu utilisée car la plupart des commandes acceptent un nom de fichier en argument.

La commande `wc` pourrait s'utiliser ainsi :

```
$ wc -l .bash_profile
27 .bash_profile # le nombre de lignes est suivi du nom du fichier
$ wc -l < .bash_profile
27 # le nombre de lignes est seul
```

Les redirections de sortie

Les sorties standards peuvent être redirigées vers d'autres fichiers grâce aux caractères `>` ou `>>`.

La redirection simple `>` écrase le contenu du fichier de sortie :

```
$ date +%F > fic_date
```

alors que la redirection double `>>` ajoute (concatène) au contenu du fichier de sortie.

```
$ date +%F >> fic_date
```

Dans les deux cas, le fichier est automatiquement créé lorsqu'il n'existe pas.

La sortie d'erreur standard peut être également redirigée vers un autre fichier. Cette fois-ci, il faudra préciser le numéro du canal (qui peut être omis pour les canaux 0 et 1) :

```
$ ls -R / 2> fic_erreurs
$ ls -R / 2>> fic_erreurs
```

Exemples de redirections

Redirection de 2 sorties vers 2 fichiers :

```
$ ls -R / >> fic_ok 2>> fic_nok
```

Redirection des 2 sorties vers un fichier unique :

```
$ ls -R / >> fic_log 2>&1
```

Redirection de **stderr** vers un "puits sans fond" (`/dev/null`) :

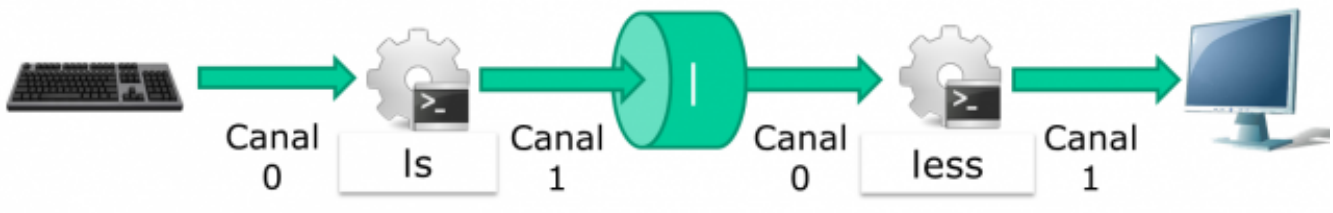
```
$ ls -R / 2>> /dev/null
```

Lorsque les 2 flux de sortie sont redirigés, aucune information n'est affichée à l'écran. Pour utiliser à la fois la redirection de sortie et conserver l'affichage, il faudra utiliser la commande `tee`.

Les tubes (pipe)

Un **tube** (**pipe** en anglais) est un mécanisme permettant de relier la sortie standard d'une première commande vers l'entrée standard d'une seconde.

Cette communication est monodirectionnelle et se fait grâce au symbole `|`. Le symbole pipe "|" est obtenu en appuyant simultanément sur les touches *`AltGR+6`*.



Toutes les données envoyées par la commande à gauche du tube à travers le canal de sortie standard sont envoyées au canal d'entrée standard de la commande placée à droite.

Les commandes particulièrement utilisées après un pipe sont des filtres.

- Exemples :

```
# N'afficher que le début :
$ ls -lia / | head

# N'afficher que la fin :
$ ls -lia / | tail

# Trier le résultat
$ ls -lia / | sort

# Compter le nombre de mots / caractères
$ ls -lia / | wc

# Chercher une chaîne de caractères dans le résultat :
$ ls -lia / | grep fichier
```

2.8. Points particuliers

La commande tee

La commande `tee` permet de rediriger la sortie standard d'une commande vers un fichier tout en maintenant l'affichage à l'écran.

Elle est combinée avec le pipe "|" pour recevoir en entrée la sortie de la commande à rediriger.

```
$ ls -lia / | tee fic
$ cat fic
```

L'option `-a` permet d'ajouter au fichier au lieu de l'écraser.

Les commandes alias et unalias

Utiliser les **alias** est un moyen pour demander au shell de se souvenir d'une commande particulière avec ses options et lui donner un nom.

Par exemple :

```
$ ll
```

remplacera la commande :

```
$ ls -l
```

La commande `alias` liste les alias de la session en cours. Des alias sont positionnés par défaut sur

les distributions Linux. Ici, les alias d'un serveur CentOS :

```
$ alias
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias vi='vim'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'
```

Les alias ne sont définis que de façon temporaire, le temps de la session utilisateur.

Pour une utilisation permanente, il faut les créer dans le fichier :

- `.bashrc` du répertoire de connexion de l'utilisateur ;
- `/etc/profile.d/alias.sh` pour tous les utilisateurs.

Une attention particulière doit être portée lors de l'usage d'alias qui peuvent potentiellement s'avérer dangereux ! Par exemple, un alias mis en place à l'insu de l'administrateur :

```
alias cd='rm -Rf'
```

La commande `unalias` permet de supprimer les alias.

```
$ unalias ll

# Pour supprimer tous les alias :
$ unalias -a
```

Alias et fonctions utiles

- Alias `grep`

Colorise le résultat de la commande `grep` :

```
alias grep='grep --color=auto'
```

- Fonction `mcd`

Il est fréquent de créer un dossier puis de se déplacer dedans :

```
mcd() { mkdir -p "$1"; cd "$1"; }
```

- Fonction `cls`

Se déplacer dans un dossier et lister son contenu :

```
cls() { cd "$1"; ls; }
```

- Fonction `backup`

Créer une copie de sauvegarde d'un fichier :

```
backup() { cp "$1"{,.bak}; }
```

- Fonction `extract`

Extrait tout type d'archive :

```
extract () {
  if [ -f $1 ] ; then
    case $1 in
      *.tar.bz2) tar xjf $1 ;;
      *.tar.gz) tar xzf $1 ;;
      *.bz2) bunzip2 $1 ;;
      *.rar) unrar e $1 ;;
      *.gz) gunzip $1 ;;
      *.tar) tar xf $1 ;;
      *.tbz2) tar xjf $1 ;;
      *.tgz) tar xzf $1 ;;
      *.zip) unzip $1 ;;
      *.Z) uncompress $1 ;;
      *.7z) 7z x $1 ;;
      *)
        echo "'$1' cannot be extracted via extract()" ;;
    esac
  else
    echo "'$1' is not a valid file"
  fi
}
```

- Alias `cmount`

```
alias cmount="mount | column -t"

[root]# cmount
/dev/simfs  on  /                                   type  simfs
(rw,relatime,usrquota,grpquota)
proc        on  /proc                               type  proc
(rw,relatime)
sysfs       on  /sys                                type  sysfs
(rw,relatime)
none        on  /dev                                type  devtmpfs
(rw,relatime,mode=755)
none        on  /dev/pts                            type  devpts
(rw,relatime,mode=600,ptmxmode=000)
none        on  /dev/shm                            type  tmpfs
(rw,relatime)
none        on  /proc/sys/fs/binfmt_misc            type  binfmt_misc
(rw,relatime)
```

Le caractère ;

Le caractère `;` chaîne les commandes.

Les commandes s'exécuteront toutes séquentiellement dans l'ordre de saisie une fois que l'utilisateur aura appuyé sur `[ENTREE]`.

```
$ ls /; cd /home; ls -lia; cd /
```

Chapitre 3. Commandes avancées pour utilisateurs Linux

3.1. La commande uniq

La commande **uniq** est une commande, utilisée avec la commande **sort** très puissante, notamment pour l'analyse de fichiers de logs. Elle permet de trier et d'afficher des entrées en supprimant les doublons.

Pour illustre le fonctionnement de la commande **uniq**, utilisons un fichier prenoms.txt contenant une liste de prénoms :

```
antoine
xavier
patrick
xavier
antoine
antoine
```

uniq réclame que le fichier d'entrée soit trié car il ne compare que les lignes consécutives.

Sans argument, la commande uniq ne va pas afficher les lignes identiques qui se suivent du fichier prenoms.txt :

```
$ sort prenoms.txt | uniq
antoine
patrick
xavier
```

Pour n'afficher que les lignes n'apparaissant qu'une seule fois, il faut utiliser l'option **-u** :

```
$ sort prenoms.txt | uniq -u
patrick
```

A l'inverse, pour n'afficher que les lignes apparaissant au moins deux fois dans le fichier, il faut utiliser l'option **-d** :

```
$ sort prenoms.txt | uniq -d
antoine
xavier
```

Pour simplement supprimer les lignes qui n'apparaissent qu'une seule fois, il faut utiliser l'option **-D** :

```
$ sort prenoms.txt | uniq -D
antoine
antoine
antoine
xavier
xavier
```

Enfin, pour compter le nombre d'occurrences de chaque ligne, il faut utiliser l'option **-c** :

```
$ sort prenoms.txt | uniq -c
      3 antoine
      1 patrick
      2 xavier
```

```
$ sort prenoms.txt | uniq -cd
      3 antoine
      2 xavier
```

3.2. La commande xargs

La commande **xargs** permet la construction et l'exécution de lignes de commandes à partir de l'entrée standard.

La commande xargs lit des arguments délimités par des blancs ou par des sauts de ligne depuis l'entrée standard, et exécute une ou plusieurs fois la commande (**/bin/echo** par défaut) en utilisant les arguments initiaux suivis des arguments lus depuis l'entrée standard.

Un premier exemple le plus simple possible serait le suivant :

```
$ xargs
utilisation
de
xargs
<CTRL+D>
utilisation de xargs
```

La commande xargs attend une saisie depuis l'entrée standard **stdin**. Trois lignes sont saisies. La fin de la saisie utilisateur est spécifiée à xargs par la séquence de touches <CTRL+D>. Xargs exécute alors la commande par défaut **echo** suivi des trois arguments correspondants à la saisie utilisateur, soit :

```
$ echo "utilisation" "de" "xargs"
utilisation de xargs
```

Il est possible de spécifier une commande à lancer par xargs.

Dans l'exemple qui suit, xargs va exécuter la commande *ls -ld* sur l'ensemble des dossiers qui seront spécifiés depuis l'entrée standard :

```
$ xargs ls -ld
/home
/tmp
/root
<CTRL+D>
drwxr-xr-x. 9 root root 4096  5 avril 11:10 /home
dr-xr-x---. 2 root root 4096  5 avril 15:52 /root
drwxrwxrwt. 3 root root 4096  6 avril 10:25 /tmp
```

En pratique, la commande xargs a exécuté la commande *ls -ld "/home" "/tmp" "/root"*.

Que se passe-t-il si la commande à exécuter n'accepte pas plusieurs arguments comme c'est le cas pour la commande find ?

```
$ xargs find /var/log -name
*.old
*.log
find: les chemins doivent précéder l'expression : *.log
```

La commande xargs a tenté d'exécuter la commande *find* avec plusieurs arguments derrière l'option *-name*, ce qui fait généré par find une erreur :

```
$ find /var/log -name "*.old" "*.log"
find: les chemins doivent précéder l'expression : *.log
```

Dans ce cas, il faut forcer la commande xargs à exécuter plusieurs fois (une fois par ligne saisie en entrée standard) la commande find. L'option **-L** suivie d'un **nombre entier** permet de spécifier le nombre maximal d'entrées à traiter avec la commande en une seule fois :

```
$ xargs -L 1 find /var/log -name
*.old
/var/log/dmesg.old
*.log
/var/log/boot.log
/var/log/anaconda.yum.log
/var/log/anaconda.storage.log
/var/log/anaconda.log
/var/log/yum.log
/var/log/audit/audit.log
/var/log/anaconda.ifcfg.log
/var/log/dracut.log
/var/log/anaconda.program.log
```

Si nous avions voulu pouvoir spécifier sur la même ligne les deux arguments, il aurait fallut utiliser l'option -n 1 :

```
$ xargs -n 1 find /var/log -name
*.old *.log
/var/log/dmesg.old
/var/log/boot.log
/var/log/anaconda.yum.log
/var/log/anaconda.storage.log
/var/log/anaconda.log
/var/log/yum.log
/var/log/audit/audit.log
/var/log/anaconda.ifcfg.log
/var/log/dracut.log
/var/log/anaconda.program.log
```

Cas concret d'une sauvegarde avec un tar en fonction d'une recherche :

```
$ find /var/log/ -name "*.log" -mtime -1 | xargs tar cvfP /root/log.tar
$ tar tvfP /root/log.tar
-rw-r--r-- root/root      1720 2017-04-05 15:43 /var/log/boot.log
-rw------- root/root    499270 2017-04-06 11:01 /var/log/audit/audit.log
```

La particularité de la commande `xargs` est quelle place l'argument en entrée à la fin de la commande appelée. Ceci fonctionne très bien avec l'exemple ci-dessus puisque les fichiers passés en entrée vont constituer la liste des fichiers à ajouter à l'archive.

Maintenant, si nous prenons l'exemple de la commande `cp` en voulant copier une liste de fichiers dans un répertoire, cette liste de fichiers sera ajoutée en fin de commande… or ce qui est attendu par la commande `cp` en fin de commande est plutôt la destination. Pour ce faire, nous utilisons l'option `-I` afin de placer les arguments en entrée ailleurs qu'en fin de ligne.

```
$ for i in $(cat /space/OTHERS/list) ; do find ./ -type f -name ""$i"*"
2>/space/OTHERS/errors | xargs -I % cp --parents % /space/restore ; done
```

L'option `-I` permet de spécifier un caractère (dans notre exemple le caractère `%`) où seront placés les fichiers en entrée de `xargs`.

L'exemple ci-dessus boucle sur une liste de fichiers à rechercher dans le répertoire courant (`./`) et envoie ces fichiers à l'emplacement du caractère `%` après `cp` (l'option `--parents` permet de conserver l'arborescence des fichiers trouvés).

3.3. Le paquet yum-utils

Le paquet **yum-utils** est une collection d'utilitaires de différents auteurs pour yum, qui le rendent plus simple et plus puissant à utiliser.

Voici quelques exemples d'utilisation :

- La commande package-cleanup :

Elle permet (entre autre) la suppression des anciens noyau.

Syntaxe de la commande package-cleanup

```
package-cleanup --oldkernels --count=1
```

Table 20. Options de la commande package-cleanup

Options	Commentaires
--oldkernels	Demande la suppression des anciens noyaux.
--count=X	Nombre de noyaux à conserver (par défaut = 2)

- La commande repoquery :

La commande **repoquery** interroge les dépots.

Exemples d'utilisation :

- Connaître les dépendances d'un paquet non-installé :

```
repoquery --requires <package>
```

- Connaître les fichiers fournis par un paquet non-installé :

```
$ repoquery -l yum-utils
/etc/bash_completion.d
/etc/bash_completion.d/yum-utils.bash
/usr/bin/debuginfo-install
/usr/bin/find-repos-of-install
/usr/bin/needs-restarting
/usr/bin/package-cleanup
/usr/bin/repo-graph
/usr/bin/repo-rss
/usr/bin/repoclosure
/usr/bin/repodiff
/usr/bin/repomanage
/usr/bin/repoquery
/usr/bin/reposync
/usr/bin/repotrack
/usr/bin/show-changed-rco
/usr/bin/show-installed
/usr/bin/verifytree
/usr/bin/yum-builddep
/usr/bin/yum-config-manager
/usr/bin/yum-debug-dump
/usr/bin/yum-debug-restore
/usr/bin/yum-groups-manager
/usr/bin/yumdownloader
...
```

- La commande yumdownloader :

La commande **yumdownloader** télécharge les paquets RPM depuis les dépôts.

Cette commande est très pratique pour construire un dépôt local de quelques rpm !

Exemple : yumdownloader va télécharger le paquet rpm de repoquery ainsi que toutes ses dépendances.

```
$ yumdownloader --destdir /var/tmp -- resolve repoquery
```

Table 21. Options de la commande yumdownloader

Options	Commentaires
--destdir	Les paquets téléchargés seront conservés dans le dossier spécifié.
--resolve	Télécharge également les dépendances du paquet.

3.4. Le paquet psmisc

Le paquet **psmisc** contient des utilitaires pour gérer les processus du système :

- **pstree** : la commande pstree affiche les processus en cours sur le système sous forme de structure en forme d'arbre.
- **killall** : la commande killall envoie un signal d'extinction à tous les procesuss identifiés par un nom.
- **fuser** : la commande fuser identifie les PIDs des processus qui utilisent les fichiers ou les systèmes de fichiers spécifiés.

Exemples :

```
$ pstree
systemd─┬─NetworkManager───2*[{NetworkManager}]
        ├─agetty
        ├─auditd───{auditd}
        ├─crond
        ├─dbus-daemon───{dbus-daemon}
        ├─firewalld───{firewalld}
        ├─lvmetad
        ├─master─┬─pickup
        │        └─qmgr
        ├─polkitd───5*[{polkitd}]
        ├─rsyslogd───2*[{rsyslogd}]
        ├─sshd───sshd───bash───pstree
        ├─systemd-journal
        ├─systemd-logind
        ├─systemd-udevd
        └─tuned───4*[{tuned}]
```

```
# killall httpd
```

Tue les processus (option -k) qui accèdent au fichier */etc/httpd/conf/httpd.conf* :

```
# fuser -k /etc/httpd/conf/httpd.conf
```

3.5. La commande watch

La commande **watch** exécute régulièrement une commande et affiche le résultat dans le terminal en plein écran.

L'option **-n** permet de spécifier le nombre de secondes entre chaque exécution de la commande.

Pour quitter la commande watch, il faut saisir les touches : <CTRL>+<C> pour tuer le processus.

Exemples :

- Afficher la fin du fichier /etc/passwd toutes les 5 secondes :

```
$ watch -n 5 tail -n 5 /etc/passwd
```

Résultat :

```
Toutes les 5,0s: tail -n 5 /etc/p...

lightdm:x:620:620:Light Display Manager:/var/lib/lightdm:/usr/bin/nologin
clamav:x:64:64:Clam AntiVirus:/dev/null:/bin/false
systemd-coredump:x:994:994:systemd Core Dumper:/:/sbin/nologin
ceph:x:993:993::/run/ceph:/sbin/nologin
dnsmasq:x:992:992:dnsmasq daemon:/:/sbin/nologin
```

- Surveillance du nombre de fichier dans un dossier :

```
$ watch -n 1 'ls -l | wc -l'
```

- Afficher une horloge :

```
$ watch -t -n 1 date
```

Chapitre 4. Éditeur de texte VI

🎓 Objectifs

✔ Utiliser les principales commandes de l'éditeur VI ;
✔ Modifier un texte grâce à l'éditeur VI.

4.1. Introduction

Visual (VI) est un éditeur de texte très populaire sous Linux malgré une ergonomie qui semble limitée. C'est en effet un éditeur entièrement en mode texte : chacune des actions se faisant avec une touche du clavier ou des commandes dédiées.

Très puissant, il est surtout très pratique puisqu'il est présent dans le noyau et donc accesssible en cas de défaillance du système. Son **universalité** (il est présent sur toutes les distributions Linux et sous Unix) en fait un outil **incontournable** de l'administrateur.

Ses fonctionnalités sont :

- Insertion, suppression, modification de texte ;
- Copie de mots, lignes ou blocs de texte ;
- Recherche et remplacement de caractères.

4.2. La commande vi

La commande `vi` ouvre l'éditeur de texte VI.

Syntaxe de la commande vi

```
vi [-c commande] [fichier]
```

Exemple :

```
$ vi /home/stagiaire/fichier
```

Table 22. Option de la commande vi

Option	Information
`-c commande`	Exécute VI en précisant une commande à l'ouverture

Si le fichier existe à l'endroit mentionné par le chemin, celui-ci est lu par VI qui se place en mode Commandes.

Si le fichier n'existe pas, VI ouvre un fichier vierge et une page vide est affichée à l'écran. A l'enregistrement du fichier, celui-ci prendra le nom précisé avec la commande.

Si la commande vi est exécutée sans préciser de nom de fichier, VI ouvre un fichier vierge et une page vide est affichée à l'écran. A l'enregistrement du fichier, VI demandera un nom de fichier.

L'éditeur `vim` reprend l'interface et les fonctions de VI avec de nombreuses améliorations.

Syntaxe de la commande vim

```
vim [-c commande] [fichier]
```

Parmi ces améliorations, l'utilisateur dispose de la coloration syntaxique, très utile pour éditer des scripts shell.

Pendant une session, VI utilise un fichier tampon dans lequel il inscrit toutes les modifications effectuées par l'utilisateur.

Tant que l'utilisateur n'a pas enregistré son travail, le fichier d'origine n'est pas modifié.

Au démarrage, VI est en mode **commandes**.

Une ligne de texte se termine en appuyant sur *ENTREE* mais si l'écran n'est pas assez large, VI effectue des retours à la ligne automatiques.

Pour sortir de VI, il faut, depuis le mode Commandes, taper sur *:* puis saisir :

- `q` pour sortir sans sauvegarder ;
- `w` pour enregistrer son travail ;
- `wq` ou `x` pour sortir et sauvegarder.

Pour forcer la sortie sans confirmation, il faut ajouter `!` aux commandes précédentes.

Il n'y a pas de sauvegarde automatique, il faut donc penser à sauvegarder son travail régulièrement.

4.3. Mode opératoires

Dans VI, il existe 3 modes de travail :

- Le mode **Commandes** ;
- Le mode **Insertion** ;
- Le mode **Ex**.

La philosophie de VI est d'alterner entre le mode Commandes et le mode Insertion.

Le troisième mode, Ex, est un mode de commandes de bas de page issu d'un ancien éditeur de texte.

Le mode Commandes

C'est le mode par défaut au démarrage de VI. Pour y accéder à partir d'un des autres modes, il suffit de taper sur la touche *ECHAP*.

Toutes les saisies sont interprétées comme des commandes et les actions correspondantes sont exécutées. Ce sont essentiellement des commandes permettant la modification de texte (copier, coller, ...).

Les commandes ne s'affichent pas à l'écran.

Le mode Insertion

C'est le mode de modification du texte. Pour y accéder à partir du mode Commandes, il faut taper sur des touches particulières qui effectueront une action en plus de changer de mode.

La saisie du texte ne s'effectue pas directement sur le fichier mais dans une zone tampon de la mémoire. Les modifications ne sont effectives que lors de l'enregistrement du fichier.

Le mode Ex

C'est le mode de modification du fichier. Pour y accéder, il faut d'abord passer en mode Commandes, puis saisir la commande Ex commençant fréquemment par le caractère **:**.

La commande est validée en appuyant sur la touche *ENTREE*.

4.4. Déplacer le curseur

En mode Commandes, il existe plusieurs façons de déplacer le curseur.

La souris n'étant pas active, il est possible de le déplacer caractère par caractère, mais des raccourcis existent pour aller plus vite.

VI reste en mode Commandes après le déplacement du curseur.

Le curseur est placé sous le caractère désiré.

À partir d'un caractère

- Déplacement d'un ou **n** caractères vers la gauche :

[←] ou [**n**][←]

- Déplacement d'un ou **n** caractères vers la droite :

[→] ou [n][→]

- Déplacement d'un ou n caractères vers le haut :

[↑] ou [n][↑]

- Déplacement d'un ou n caractères vers le bas :

[↓] ou [n][↓]

- Déplacement à la fin de la ligne :

[$] ou [FIN]

- Déplacement au début de la ligne :

[0] ou [POS1]

À partir du premier caractère d'un mot

Les mots sont constitués de lettres ou de chiffres. Les caractères de ponctuation et les apostrophes séparent les mots.

Si le curseur se trouve au milieu d'un mot [w] passe au mot suivant, [b] passe au début du mot.

Si la ligne est finie, VI passe automatiquement à la ligne suivante.

- Déplacement d'un ou n mots vers la droite :

[w] ou [n][w]

- Déplacement d'un ou n mots vers la gauche :

[b] ou [n][b]

À partir du premier caractère d'une ligne

- Déplacement à la dernière ligne du texte :

[G]

- Déplacement à la ligne n :

[n][G]

- Déplacement à la première ligne de l'écran :

[H]

- Déplacement à la ligne du milieu de l'écran :

[M]

- Déplacement à la dernière ligne de l'écran :

[L]

4.5. Insérer du texte

En mode Commandes, il existe plusieurs façons d'insérer du texte.

VI bascule en mode Insertion après la saisie d'une de ces touches.

VI bascule en mode Insertion. Il faudra donc appuyer sur la touche *ECHAP* pour revenir en mode Commandes.

Par rapport à un caractère

- Insertion de texte avant un caractère :

[i]

- Insertion de texte après un caractère :

[a]

Par rapport à une ligne

- Insertion de texte au début d'une ligne :

[I]

- Insertion de texte à la fin d'une ligne :

[A]

Par rapport au texte

- Insertion de texte avant une ligne :

[O]

- Insertion de texte après une ligne :

[o]

4.6. Caractères, mots et lignes

VI permet l'édition de texte en gérant :

- les caractères,
- les mots,
- les lignes.

Il est possible pour chaque cas de :

- supprimer,
- remplacer,
- copier,
- couper,
- coller.

Ces opérations se font en mode Commandes.

Caractères

- Supprimer un ou `n` caractères :

[x] ou [n][x]

- Remplacer un caractère par un autre :

[r][caractère]

- Remplacer plus d'un caractère par d'autres :

[R][caractères][ECHAP]

La commande [R] bascule en mode Remplacement, qui est une sorte de mode Insertion.

Mots

- Supprimer (couper) un ou `n` mots :

[d][w] ou [n][d][w]

- Copier un ou `n` mots :

[y][w] ou [n][y][w]

- Coller un mot une ou `n` fois après le curseur :

[p] ou [n][p]

- Coller un mot une ou `n` fois avant le curseur :

[P] ou [n][P]

- Remplacer un mot :

[c][w][mot][ECHAP]

Il faut positionner le curseur sous le premier caractère du mot à couper (ou copier) sinon VI coupera (ou copiera) seulement la partie du mot entre le curseur et la fin.

Supprimer un mot revient à la couper. S'il n'est pas collé ensuite, le tampon est vidé et le mot est supprimé.

Lignes

- Supprimer (couper) une ou n lignes :

[d][d] ou [n][d][d]

- Copier une ou n lignes :

[y][y] ou [n][y][y]

- Coller ce qui a été copié ou supprimé une ou n fois après la ligne courante :

[p] ou [n][p]

- Coller ce qui a été copié ou supprimé une ou n fois avant la ligne courante :

[P] ou [n][P]

- Supprimer (couper) du début de la ligne jusqu'au curseur :

[d][0]

- Supprimer (couper) du curseur jusqu'à la fin de la ligne :

[d][$]

- Copier du début de la ligne jusqu'au curseur :

[y][0]

- Copier du curseur jusqu'à la fin de la ligne :

[y][$]

- Supprimer (couper) le texte à partir de la ligne courante :

[d][L] ou [d][G]

- Copier le texte à partir de la ligne courante :

[y][L] ou [y][G]

Annuler une action

- Annuler la dernière action :

[u]

- Annuler les actions sur la ligne courante :

[U]

4.7. Commandes EX

Le mode Ex permet d'agir sur le fichier (enregistrement, mise en page, options, ...). C'est aussi en mode Ex que se saisissent les commandes de recherche et de remplacement. Les commandes sont affichées en bas de page et doivent être validées avec la touche *ENTREE*.

Pour passer en mode Ex, du mode Commandes, taper [:].

Numéroter les lignes

- Afficher/masquer la numérotation :

```
:set nu
```

```
:set nonu
```

Rechercher une chaîne de caractères

- Rechercher une chaîne de caractères à partir du curseur :

```
/chaîne
```

- Rechercher une chaîne de caractères avant le curseur :

```
?chaîne
```

- Aller à l'occurrence trouvée suivante :

[n]

- Aller à l'occurence trouvée précédente :

[N]

Il existe des caractères jokers permettant de faciliter la recherche sous VI.

- `[]` : Recherche d'un unique caractère dont les valeurs possibles sont précisées.

Exemple :

`/[Mm]ot`.

- `^` : Recherche d'une chaîne débutant la ligne.

Exemple :

`/Mot`.

- `,$` : Recherche d'une chaîne finissant la ligne.

Exemple :

`/Mot,$`

- `*` : Recherche d'un ou de plusieurs caractères, quels qu'ils soient.

Exemple :

`/M*t`

Remplacer une chaîne de caractères

De la 1ère à la dernière ligne du texte, remplacer la chaîne recherchée par la chaîne précisée :

```
:1,$s/recherche/remplace
```

De la ligne `n` à la ligne `m`, remplacer la chaîne recherchée par la chaîne précisée :

```
:n,ms/recherche/remplace
```

Par défaut, seule la première occurence trouvée de chaque ligne est remplacée. Pour forcer le remplacement de chaque occurence, il faut ajouter `/g` à la fin de la commande :

```
:n,ms/recherche/remplace/g
```

Opérations sur les fichiers

- Enregistrer le fichier :

```
:w
```

- Enregistrer sous un autre nom :

```
:w fichier
```

- Enregistrer de la ligne `n` à la ligne `m` dans un autre fichier :

```
:n,mw fichier
```

- Recharger le dernier enregistrement du fichier :

`e!`

- Coller le contenu d'un autre fichier après le curseur :

`:r fichier`

- Quitter le fichier sans enregistrer :

`:q`

- Quitter le fichier et enregistrer :

`:wq` ou `:x`

4.8. Autres fonctions

Il est possible d'exécuter VI en précisant les options à charger pour la session. Pour cela, il faut utiliser l'option `-c` :

```
$ vi -c "set nu" /home/stagiaire/fichier
```

Il est aussi possible de saisir les commandes Ex dans un fichier nommé `.exrc` mis dans le répertoire de connexion de l'utilisateur. À chaque démarrage de VI ou de VIM les commandes seront lues et appliquées.

La commande vimtutor

Il existe un tutoriel pour apprendre à utiliser VI. Il est accessible avec la commande `vimtutor`.

```
$ vimtutor
```

Chapitre 5. La gestion des utilisateurs

🎓 Objectifs

- ✔ ajouter, supprimer ou modifier un **groupe** ;
- ✔ ajouter, supprimer ou modifier un **utilisateur** ;
- ✔ connaître la syntaxe des fichiers associés à la gestion des groupes et des utilisateurs ;
- ✔ changer le **propriétaire** ou le **groupe propriétaire** d'un fichier ;
- ✔ **sécuriser** les comptes utilisateurs ;
- ✔ changer d'identité.

5.1. Généralités

Chaque utilisateur est membre d'au moins un groupe : **c'est son groupe principal**.

Plusieurs utilisateurs peuvent faire partie d'un même groupe.

Les utilisateurs peuvent appartenir à d'autres groupes. Ces utilisateurs sont **invités** dans ces **groupes secondaires**.

Chaque utilisateur possède un groupe principal et peut être invité dans un ou plusieurs groupes secondaires.

Les groupes et utilisateurs se gèrent par leur identifiant numérique unique `GID` et `UID`.

Les fichiers de configuration se trouvent dans `/etc`.

`UID`
: User IDentifier. Identifiant unique d'utilisateur.

`GID`
: Group IDentifier. Identifiant unique de groupe.

Il est recommandé d'utiliser les commandes d'administration au lieu de modifier manuellement les fichiers.

5.2. Gestion des groupes

Fichiers modifiés, ajout de lignes :

- `/etc/group`
- `/etc/gshadow`

Commande groupadd

La commande `groupadd` permet d'ajouter un groupe au système.

Syntaxe de la commande groupadd

```
groupadd [-f] [-g GID] groupe
```

Exemple :

```
[root]# groupadd -g 512 GroupeB
```

Table 23. Options de la commande groupadd

Option	Description
`-g GID`	`GID` du groupe à créer.
`-f`	Le système choisit un `GID` si celui précisé par l'option `-g` existe déjà.
`-r`	Crée un groupe système avec un `GID` compris entre `SYS_GID_MIN` et `SYS_GID_MAX`. Ces deux variables sont définies dans `/etc/login.defs`.

Règles de nommage des groupes :

- Pas d'accents, ni caractères spéciaux ;
- Différents du nom d'un utilisateur ou fichier système existant.

Sous **Debian**, l'administrateur devrait privilégier, sauf dans des scripts ayant la vocation d'être portables vers toutes les distributions Linux, les commandes addgroup/delgroup comme précisé dans le man :

```
[root] # man addgroup
DESCRIPTION
    adduser et addgroup ajoutent des utilisateurs ou des groupes au
système en fonction des options fournies en ligne de commande et des
informations contenues dans le fichier de configuration
/etc/adduser.conf. Ce sont des interfaces plus conviviales que les
programmes useradd et groupadd. Elles  permettent de choisir par défaut
des UID ou des GID conformes à la charte Debian, de créer un répertoire
personnel configuré suivant un modèle (squelette), d'utiliser un script
sur mesure, et d'autres fonctionnalités encore.
```

Commande groupmod

La commande `groupmod` permet de modifier un groupe existant sur le système.

Syntaxe de la commande groupmod

```
groupmod [-g GID] [-n nom] groupe
```

Exemple :

```
[root]# groupmod -g 516 GroupeP
[root]# groupmod -n GroupeC GroupeB
```

Table 24. Options de la commande groupmod

Option	Description
`-g GID`	Nouveau `GID` du groupe à modifier.
`-n nom`	Nouveau nom.

Il est possible de modifier le nom d'un groupe, son `GID` ou les deux simultanément.

Après modification, les fichiers appartenant au groupe ont un `GID` inconnu. Il faut leur réattribuer le nouveau `GID`.

```
[root]# find / -gid 502 -exec chgrp 516 {} \;
```

Commande groupdel

La commande `groupdel` permet de supprimer un groupe existant sur le système.

Syntaxe de la commande groupdel

```
groupdel groupe
```

Exemple :

```
[root]# groupdel GroupeC
```

Pour être supprimé, un groupe ne doit plus contenir d'utilisateurs.

La suppression du dernier utilisateur d'un groupe éponyme entraînera la suppression de ce groupe par le système.

Chaque groupe possède un `GID` unique. Un groupe peut être dupliqué. Par convention, les `GID` des groupes systèmes vont de 0 (`root`) à 499.

Un utilisateur faisant obligatoirement partie d'un groupe, il est nécessaire de créer les groupes avant d'ajouter les utilisateurs. Par conséquent, un groupe peut ne pas avoir de membres.

Fichier /etc/group

Ce fichier contient les informations de groupes (séparées par `:`).

```
[root]# tail -1 /etc/group
GroupeP:x:516:stagiaire
  (1)  (2)(3)   (4)
```

1

Nom du groupe.

2

Mot de passe (`x` si défini dans `/etc/gshadow`).

3

GID.

4

Membres invités (séparés par des virgules, ne contient pas les membres principaux).

Chaque ligne du fichier `/etc/group` correspond à un groupe. Les utilisateurs dont ce groupe est leur groupe principal ne sont pas listés à ce niveau.

Cette information d'appartenance est en fait déjà fournie par le fichier `/etc/passwd`...

Fichier /etc/gshadow

Ce fichier contient les informations de sécurité sur les groupes (séparées par `:`).

```
[root]# grep GroupeA /etc/gshadow
GroupeA:$6$2,9,v...SBn160:alain:stagiaire
   (1)      (2)           (3)      (4)
```

1

Nom du groupe.

2

Mot de passe chiffré.

3

Administrateur du groupe.

4

Membres invités (séparés par des virgules, ne contient pas les membres principaux).

Pour chaque ligne du fichier `/etc/group` doit correspondre une ligne du fichier `/etc/gshadow`.

Un `!` au niveau du mot de passe indique que celui-ci est bloqué. Ainsi aucun utilisateur ne peut utiliser le mot de passe pour accéder au groupe (sachant que les membres du groupe n'en ont pas besoin).

5.3. Gestion des utilisateurs

Définition

Un utilisateur se définit comme suit dans le fichier `/etc/passwd` :

1

Login ;

2

Mot de passe ;

3

UID ;

4

GID du groupe principal ;

5

Commentaire ;

6

Répertoire de connexion ;

7

Interpréteur de commandes (`/bin/bash`, `/bin/nologin`,...).

Il existe trois types d'utilisateurs :

- **root** : Administrateur du système ;
- **utilisateur système** : Utilisé par le système pour la gestion des droits d'accès des applications ;
- **utilisateur ordinaire** : Autre compte permettant de se connecter au système.

Fichiers modifiés, ajout de lignes :

- `/etc/passwd`
- `/etc/shadow`

Commande useradd

La commande `useradd` permet d'ajouter un utilisateur.

Syntaxe de la commande useradd

```
useradd [-u UID] [-g GID] [-d répertoire] [-s shell] login
```

Exemple :

```
[root]# useradd -u 1000 -g 513 -d /home/GroupeC/carine carine
```

Table 25. Options de la commande useradd

Option	Description
`-u UID`	`UID` de l'utilisateur à créer.
`-g GID`	`GID` du groupe principal.
`-d répertoire`	Répertoire de connexion.
`-s shell`	Interpréteur de commandes.
`-c`	Ajoute un commentaire.
`-U`	Ajoute l'utilisateur à un groupe portant le même nom créé simultanément.
`-M`	Ne crée pas le répertoire de connexion.

À la création, le compte ne possède pas de mot de passe et est verrouillé. Il faut assigner un mot de passe pour déverrouiller le compte.

Règles de nommage des comptes :

- Pas d'accents, de majuscules ni caractères spéciaux ;
- Différents du nom d'un groupe ou fichier système existant ;
- Définir les options `-u`, `-g`, `-d` et `-s` à la création.

L'arborescence du répertoire de connexion doit être créée à l'exception du dernier répertoire. Le dernier répertoire est créé par la commande `useradd` qui en profite pour y copier les fichiers du `skel`.

Un utilisateur peut faire partie de plusieurs groupes en plus de son groupe principal.

Pour les groupes secondaires, il faut utiliser l'option `-G`.

Exemple :

```
[root]# useradd -u 500 -g GroupeA -G GroupeP,GroupeC albert
```

Sous **Debian**, il faudra spécifier l'option `-m` pour forcer la création du répertoire de connexion ou positionner la variable `CREATE_HOME` du fichier `login.defs`. Dans tous les cas, l'administrateur devrait privilégier, sauf dans des scripts ayant la vocation d'être portables vers toutes les distributions Linux, les commandes `adduser`/`deluser` comme précisé dans le `man` :

```
[root] # man useradd
DESCRIPTION
    **useradd** is a low level utility for adding users. On Debian,
administrators should usually use **adduser(8)** instead.
```

Valeur par défaut de création d'utilisateur.

Modification du fichier `/etc/default/useradd`.

```
useradd -D [-b répertoire] [-g groupe] [-s shell]
```

Exemple :

```
[root]# useradd -D -g 500 -b /home -s /bin/bash
```

Table 26. Options de la commande useradd pour modifier les valeurs par défaut

Option	Description
`-D`	Définit les valeurs par défaut de création d'utilisateur.
`-b répertoire`	Définit le répertoire de connexion par défaut.
`-g groupe`	Définit le groupe par défaut.
`-s shell`	Définit le shell par défaut.
`-f`	Nombre de jours suivant l'expiration du mot de passe avant que le compte ne soit désactivé.
`-e`	Date à laquelle le compte sera désactivé.

Commande usermod

La commande `usermod` permet de modifier un utilisateur.

Syntaxe de la commande usermod

```
usermod [-u UID] [-g GID] [-d répertoire] [-m] login
```

Exemple :

```
[root]# usermod -u 544 carine
```

Options identiques à la commande `useradd`.

Table 27. Options de la commande usermod

Option	Description
`-m`	Associé à l'option `-d`, déplace le contenu de l'ancien répertoire de connexion vers le nouveau.
`-l login`	Nouveau nom.
`-e AAAA-MM-JJ`	Date d'expiration du compte.
`-L`	Verrouille le compte.
`-U`	Déverrouille le compte.
`-a`	Empêche la suppression de l'utilisateur d'un groupe secondaire lors de l'ajout dans un autre groupe secondaire.
`-G`	Précise plusieurs groupes secondaires lors de l'ajout.

Avec la commande `usermod`, le verrouillage d'un compte se traduit par l'ajout de `!` devant le mot de passe dans le fichier `/etc/shadow`.

Pour être modifié un utilisateur doit être déconnecté et ne pas avoir de processus en cours.

Après modification de l'identifiant, les fichiers appartenant à l'utilisateur ont un `UID` inconnu. Il faut leur réattribuer le nouvel `UID`.

```
[root]# find / -uid 1000 -exec chown 544: {} \;
```

Il est possible d'inviter un utilisateur dans un ou plusieurs groupes secondaires avec les options `-a` et `-G`.

Exemple :

```
[root]# usermod -aG GroupeP,GroupeC albert
```

La commande `usermod` agit en modification et non en ajout.

Pour un utilisateur invité dans un groupe par l'intermédiaire de cette commande et déjà positionné comme invité dans d'autres groupes secondaires, il faudra indiquer dans la commande de gestion de groupe tous les groupes dont il fait partie sinon il disparaîtra de ceux-ci.

L'option `-a` modifie ce comportement.

Exemples :

- Invite `albert` dans le groupe `GroupeP`

```
[root]# usermod -G GroupeP albert
```

- Invite `albert` dans le groupe `GroupeG`, mais le supprime de la liste des invités de `GroupeP`.

```
[root]# usermod -G GroupeG albert
```

- Donc soit :

```
[root]# usermod -G GroupeP,GroupeG albert
```

- Soit :

```
[root]# usermod -aG GroupeG albert
```

Commande userdel

La commande `userdel` permet de supprimer le compte d'un utilisateur.

Syntaxe de la commande userdel

```
[root]# userdel -r carine
```

Table 28. Options de la commande userdel

Option	Description
`-r`	Supprime le répertoire de connexion et les fichiers contenus.

Pour être supprimé, un utilisateur doit être déconnecté et ne pas avoir de processus en cours.

userdel supprime la ligne de l'utilisateur dans les fichiers `/etc/passwd` et `/etc/gshadow`.

Chaque utilisateur possède un `UID` unique. Par convention, les `UID` des utilisateurs 'système' vont de `0` (`root`) à `499`.

Un utilisateur est obligatoirement membre d'un groupe. Il est donc nécessaire de créer les groupes avant d'ajouter les utilisateurs.

Fichier /etc/passwd

Ce fichier contient les informations des utilisateurs (séparées par `:`).

```
[root]# head -1 /etc/passwd
root:x:0:0:root:/root:/bin/bash
(1)(2)(3)(4)(5)  (6)    (7)
```

1

Login.

2

Mot de passe (x si défini dans /etc/shadow).

3

UID.

4

GID du groupe principal.

5

Commentaire.

6

Répertoire de connexion.

7

Interpréteur de commandes.

Fichier /etc/shadow

Ce fichier contient les informations de sécurité des utilisateurs (séparées par `:`).

```
[root]# tail -1 /etc/shadow
root:$6$...:15399:0:99999:7:::
 (1)    (2)  (3) (4) (5) (6)(7,8,9)
```

1

Login.

2

Mot de passe chiffré.

3

Date du dernier changement.

4

Durée de vie minimale du mot de passe.

5

Durée de vie maximale du mot de passe.

6

Nombre de jours avant avertissement.

7

Délai avant désactivation du compte après expiration.

8

Délai d'expiration du compte.

9

Réservé pour une utilisation future.

Pour chaque ligne du fichier `/etc/passwd` doit correspondre une ligne du fichier `/etc/shadow`.

5.4. Propriétaires des fichiers

Tous les fichiers appartiennent forcément à un utilisateur et à un groupe.

Le groupe principal de l'utilisateur qui crée le fichier est, par défaut, le groupe propriétaire du fichier.

Commandes de modifications

Commande chown

La commande `chown` permet de modifier les propriétaires d'un fichier.

Syntaxe de la commande chown

```
chown [-R] [-v] login[:groupe] fichier
```

Exemples :

```
[root]# chown root fichier
[root]# chown albert:GroupeA fichier
```

Table 29. Options de la commande chown

Option	Description
`-R`	Modifie les propriétaires du répertoire et de son contenu.
`-v`	Affiche les modifications exécutées.

Pour ne modifier que l'utilisateur propriétaire :

```
[root]# chown albert fichier
```

Pour ne modifier que le groupe propriétaire :

```
[root]# chown :GroupeA fichier
```

Modification de l'utilisateur et du groupe propriétaire :

```
[root]# chown albert:GroupeA fichier
```

Dans l'exemple suivant le groupe attribué sera le groupe principal de l'utilisateur précisé.

```
[root]# chown albert: fichier
```

Commande chgrp

La commande `chgrp` permet de modifier le groupe propriétaire d'un fichier.

Syntaxe de la commande chgrp

```
chgrp [-R] [-v] groupe fichier
```

Exemple :

```
[root]# chgrp groupe1 fichier
```

Table 30. Options de la commande chgrp

Option	Description
-R	Modifie les groupes propriétaires du répertoire et de son contenu (récursivité).
-v	Affiche les modifications exécutées.

Il est possible d'appliquer à un fichier un propriétaire et un groupe propriétaire en prenant comme référence ceux d'un autre fichier :

```
chown [options] --reference=RRFILE FILE
```

Par exemple :

```
chown --reference=/etc/groups /etc/passwd
```

5.5. Gestion des invités

Commande gpasswd

La commande `gpasswd` permet de gérer un groupe.

Syntaxe de la commande gpasswd

```
gpasswd [-a login] [-A login] [-d login] [-M login] groupe
```

Exemples :

```
[root]# gpasswd -A alain GroupeA
[alain]$ gpasswd -a patrick GroupeA
```

Table 31. Options de la commande gpasswd

Option	Description
`-a login`	Ajoute l'utilisateur au groupe.
`-A login`	Définit l'administrateur du groupe.
`-d login`	Retire l'utilisateur du groupe.
`-M login`	Définit la liste exhaustive des invités.

La commande gpasswd `-M` agit en modification et non en ajout.

```
# gpasswd GroupeA
New Password :
Re-enter new password :
```

Commande id

La commande `id` affiche les noms des groupes d'un utilisateur.

Syntaxe de la commande id

```
id login
```

Exemple :

```
[root]# id alain
uid=500(alain) gid=500(GroupeA) groupes=500(GroupeA),516(GroupeP)
```

Commande newgrp

La commande `newgrp` permet d'utiliser temporairement un groupe secondaire pour la création de fichiers.

Syntaxe de la commande newgrp

```
newgrp [groupesecondaire]
```

Exemple :

```
[alain]$ newgrp GroupeB
```

Après utilisation de cette commande, les fichiers seront créés avec le `GID` de son groupe secondaire.

La commande `newgrp` sans paramètre réaffecte le groupe principal.

5.6. Sécurisation

Commande passwd

La commande `passwd` permet de gérer un mot de passe.

Syntaxe de la commande passwd

```
passwd [-d] [-l] [-S] [-u] [login]
```

Exemples :

```
[root]# passwd -l albert
[root]# passwd -n 60 -x 90 -w 80 -i 10 patrick
```

Table 32. Options de la commande passwd

Option	Description
`-d`	Supprime le mot de passe.
`-l`	Verrouille le compte.
`-S`	Affiche le statut du compte.
`-u`	Déverrouille le compte.
`-e`	Fait expirer le mot de passe.
`-n jours`	Durée de vie minimale du mot de passe.
`-x jours`	Durée de vie maximale du mot de passe.
`-w jours`	Délai d'avertissement avant expiration.
`-i jours`	Délai avant désactivation lorsque le mot de passe expire.

Avec la commande passwd, le verrouillage d'un compte se traduit par l'ajout de `!!` devant le mot de passe dans le fichier `/etc/shadow`.

L'utilisation de la commande usermod `-U` ne supprime qu'un seul des `!`. Le compte reste donc verrouillé.

Exemple :

- Alain change son mot de passe :

```
[alain]$ passwd
```

- root change le mot de passe d'Alain :

```
[root]# passwd alain
```

La commande `passwd` est accessible aux utilisateurs pour modifier leur mot de passe (l'ancien mot de passe est demandé). L'administrateur peut modifier les mots de passe de tous les utilisateurs sans restriction.

Ils devront se soumettre aux restrictions de sécurité.

Lors d'une gestion des comptes utilisateurs par script shell, il peut être utile de définir un mot de passe par défaut après avoir créé l'utilisateur.

Ceci peut se faire en passant le mot de passe à la commande `passwd`.

Exemple :

```
[root]# echo "azerty,1" | passwd --stdin philippe
```

Le mot de passe est saisi en clair, `passwd` se charge de le chiffrer.

Commande chage

La commande `chage` permet de gérer la stratégie de compte.

Syntaxe de la commande chage

```
chage [-d date] [-E date] [-I jours] [-l] [-m jours] [-M jours] [-W jours] [login]
```

Exemple :

```
[root]# chage -m 60 -M 90 -W 80 -I 10 alain
```

Table 33. Options de la commande chage

Option	Description
`-I jours`	Délai avant désactivation, mot de passe expiré (`i` majuscule).
`-l`	Affiche le détail de la stratégie (`l` minuscule).
`-m jours`	Durée de vie minimale du mot de passe.

Option	Description
`-M jours`	Durée de vie maximale du mot de passe.
`-d AAAA-MM-JJ`	Dernière modification du mot de passe.
`-E AAAA-MM-JJ`	Date d'expiration du compte.
`-W jours`	Délai d'avertissement avant expiration.

La commande `chage` propose également un mode interactif.

L'option `-d` force la modification du mot de passe à la connexion.

Exemples :

```
[root]# chage philippe
[root]# chage -d 0 philippe
```

En l'absence d'utilisateur précisé, la commande concernera l'utilisateur qui la saisit.

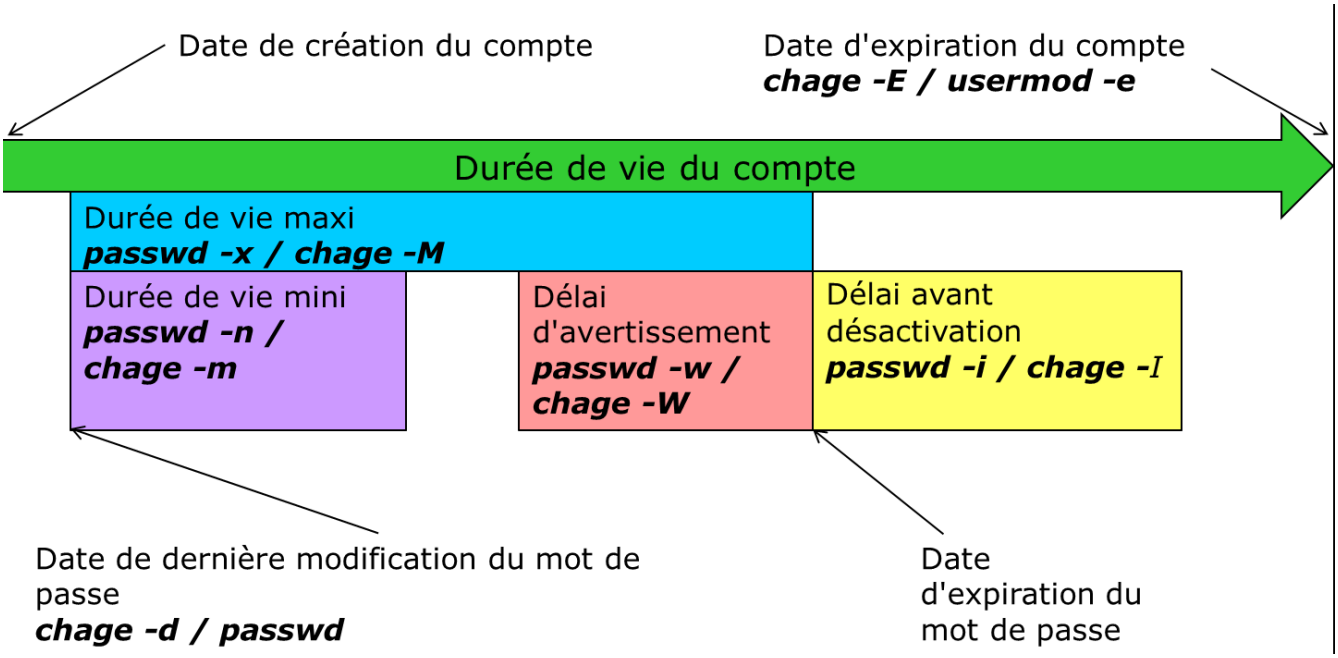


Figure 9. Gestion des comptes utilisateurs avec chage

5.7. Gestion avancée

Fichiers de configuration :

- `/etc/default/useradd`
- `/etc/login.defs`
- `/etc/skel`

L'édition du fichier `/etc/default/useradd` se fait grâce à la commande `useradd`.

Les autres fichiers sont à modifier avec un éditeur de texte.

Fichier /etc/default/useradd

Ce fichier contient le paramétrage des données par défaut.

Lors de la création d'un utilisateur, si les options ne sont pas précisées, le système utilise les valeurs par défaut définies dans `/etc/default/useradd`.

Ce fichier est modifié par la commande `useradd -D` (`useradd -D` saisie sans autre option affiche le contenu du fichier `/etc/default/useradd`).

Table 34. Contenu du fichier /etc/default/useradd

Valeur	Commentaire
`GROUP`	Groupe par défaut.
`HOME`	Chemin dans lequel le répertoire de connexion du nom de l'utilisateur sera créé.
`INACTIVE`	Nombre de jours suivant l'expiration du mot de passe avant que le compte ne soit désactivé.
`EXPIRE`	Date d'expiration du compte.
`SHELL`	Interpréteur de commandes.
`SKEL`	Répertoire squelette du répertoire de connexion.
`CREATE_MAIL_SPOOL`	Création de la boîte aux lettres dans `/var/spool/mail`.

Sans l'option `-g`, la commande `useradd` crée un groupe du nom de l'utilisateur et l'y place.

Pour que la commande `useradd` récupère la valeur du champ `GROUP` du fichier `/etc/default/useradd`, il faut préciser l'option `-N`.

Exemple :

```
[root]# useradd -u 501 -N GroupeA
```

Fichier /etc/login.defs

Ce fichier contient de nombreux paramètres par défaut utiles aux commandes de création ou de modification d'utilisateurs. Ces informations sont regroupées par paragraphe en fonction de leur utilisation :

- Boites aux lettres ;
- Mots de passe ;
- UID et GID ;
- Umask ;
- Connexions ;
- Terminaux.

Fichier /etc/skel

Lors de la création d'un utilisateur, son répertoire personnel et ses fichiers d'environnement sont créés.

Ces fichiers sont copiés automatiquement à partir du répertoire `/etc/skel`.

- `.bash_logout`
- `.bash_profile`
- `.bashrc`

Tous les fichiers et répertoires placés dans ce répertoire seront copiés dans l'arborescence des utilisateurs lors de leur création.

5.8. Changement d'identité

Commande su

La commande `su` permet de modifier l'identité de l'utilisateur connecté.

Syntaxe de la commande su

```
su [-] [-c commande] [login]
```

Exemples :

```
[root]# su - alain
[albert]$ su -c "passwd alain"
```

Table 35. Options de la commande su

Option	Description
`-`	Charge l'environnement complet de l'utilisateur.
`-c` commande	Exécute la commande sous l'identité de l'utilisateur.

Si le login n'est pas spécifié, ce sera `root`.

Les utilisateurs standards devront taper le mot de passe de la nouvelle identité.

Il y a création de 'couches' successives (un empilement d'environnement `bash`). Pour passer d'un utilisateur à un autre, il faut d'abord taper la commande `exit` pour reprendre son identité puis la commande `su` pour prendre une autre identité.

Chargement du profil

`root` endosse l'identité de l'utilisateur `alain` avec `su` :

```
...
/home/GroupeA/alain/bash_rc
/etc/bashrc
...
```

`root` endosse l'identité de l'utilisateur `alain` avec `su -` :

```
...
/home/GroupeA/alain/bash_profile
/home/GroupeA/alain/bash_rc
/etc/bashrc
...
```

Un utilisateur peut endosser temporairement (pour une autre commande ou une session entière) l'identité d'un autre compte.

Si aucun utilisateur n'est précisé, la commande concernera `root` (`su -`).

Il est nécessaire de connaître le mot de passe de l'utilisateur dont l'identité est endossé sauf si c'est `root` qui exécute la commande.

Un administrateur peut ainsi travailler sur un compte utilisateur standard et n'utiliser les droits du compte `root` que ponctuellement.

Chapitre 6. Système de fichiers

6.1. Partitionnement

Le partitionnement va permettre l'installation de plusieurs systèmes d'exploitation car il est impossible d'en faire cohabiter plusieurs sur un même lecteur logique. Le partitionnement permet également de cloisonner des données (sécurité, optimisation d'accès, ...).

Le découpage du disque physique en volumes partitionnés est inscrit dans la table des partitions stockée dans le premier secteur du disque (MBR : Master Boot Record).

Un même disque physique peut être découpé en 4 partitions maximum :

- **Primaire** (ou principale)
- **Étendue**

Il ne peut y avoir qu'une seule partition étendue par disque physique. Afin de bénéficier de lecteur supplémentaire, la partition étendue peut être découpée en partitions logiques

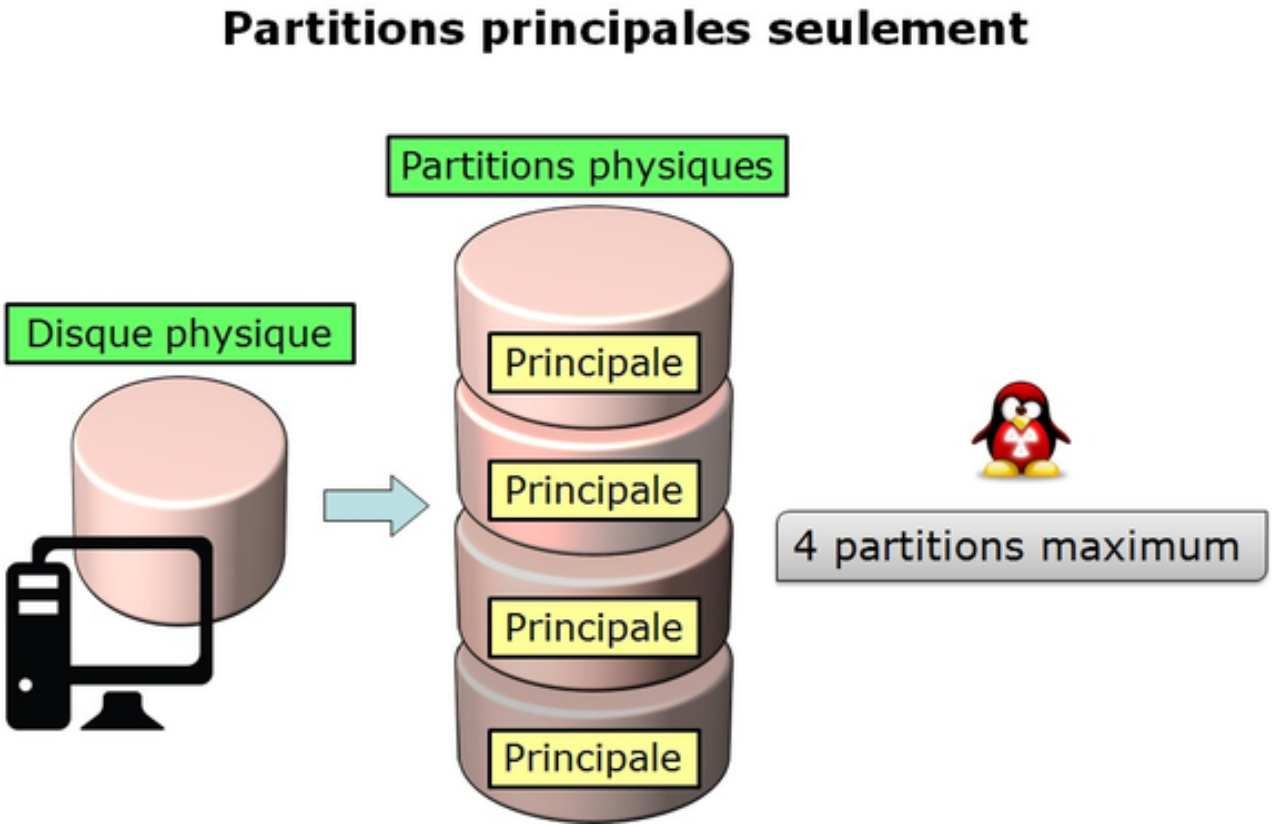


Figure 10. Découpage en quatre partitions principales seulement

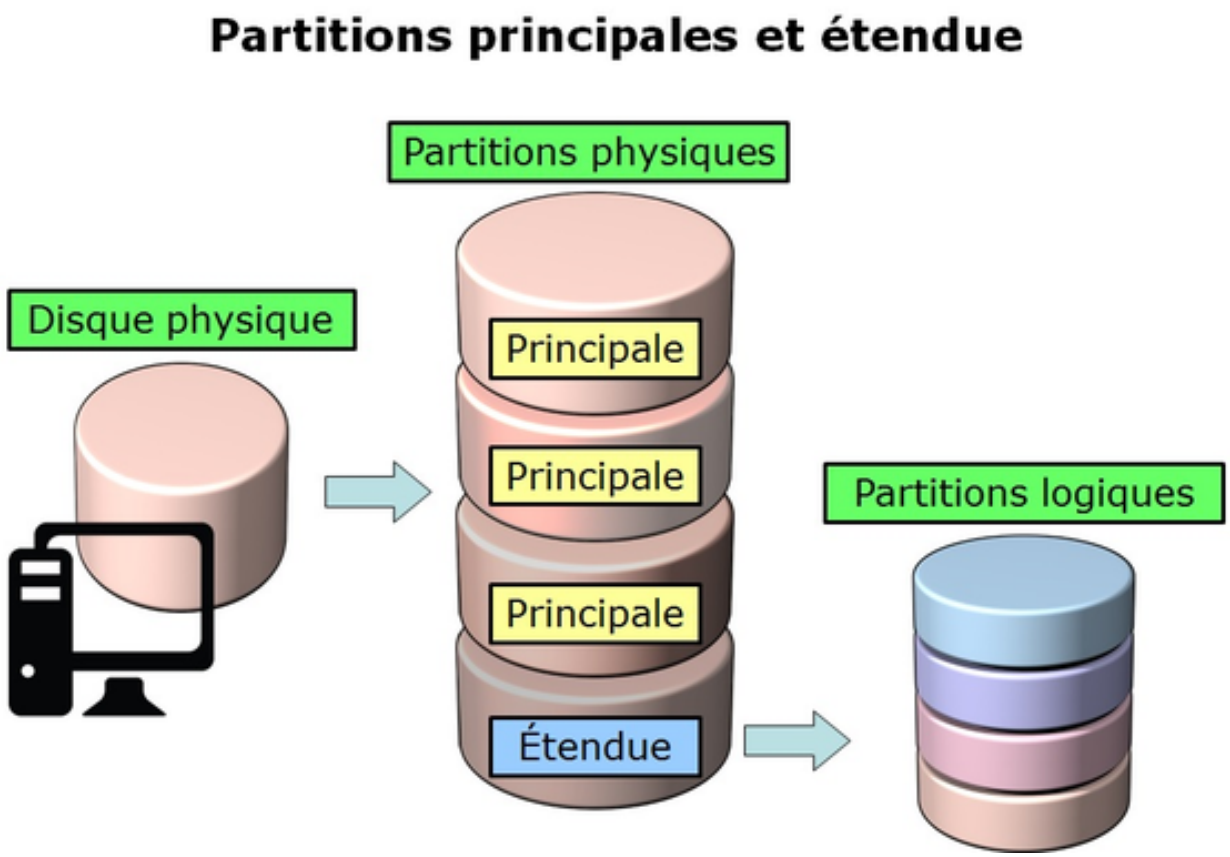


Figure 11. Découpage en trois partitions principales et une partition étendue

Les devices, ou périphériques, sont les fichiers identifiant les différents matériels détectés par la carte mère. Ces fichiers sont stockés sans **/dev**. Le service qui détecte les nouveaux périphériques et leur donne des noms s'appelle "udev".

Ils sont identifiés en fonction de leur type.

Les périphériques de stockage se nomment **hd** pour les disques durs IDE et **sd** pour les autres supports. Vient ensuite une lettre qui commence par **a** pour le premier périphérique, puis **b**, **c**, ...

Enfin nous allons trouver un chiffre qui définit le volume partitionné : **1** pour la première partition primaire, ...

Attention, la partition étendue, qui ne supporte pas de système de fichier, porte quand même un numéro.

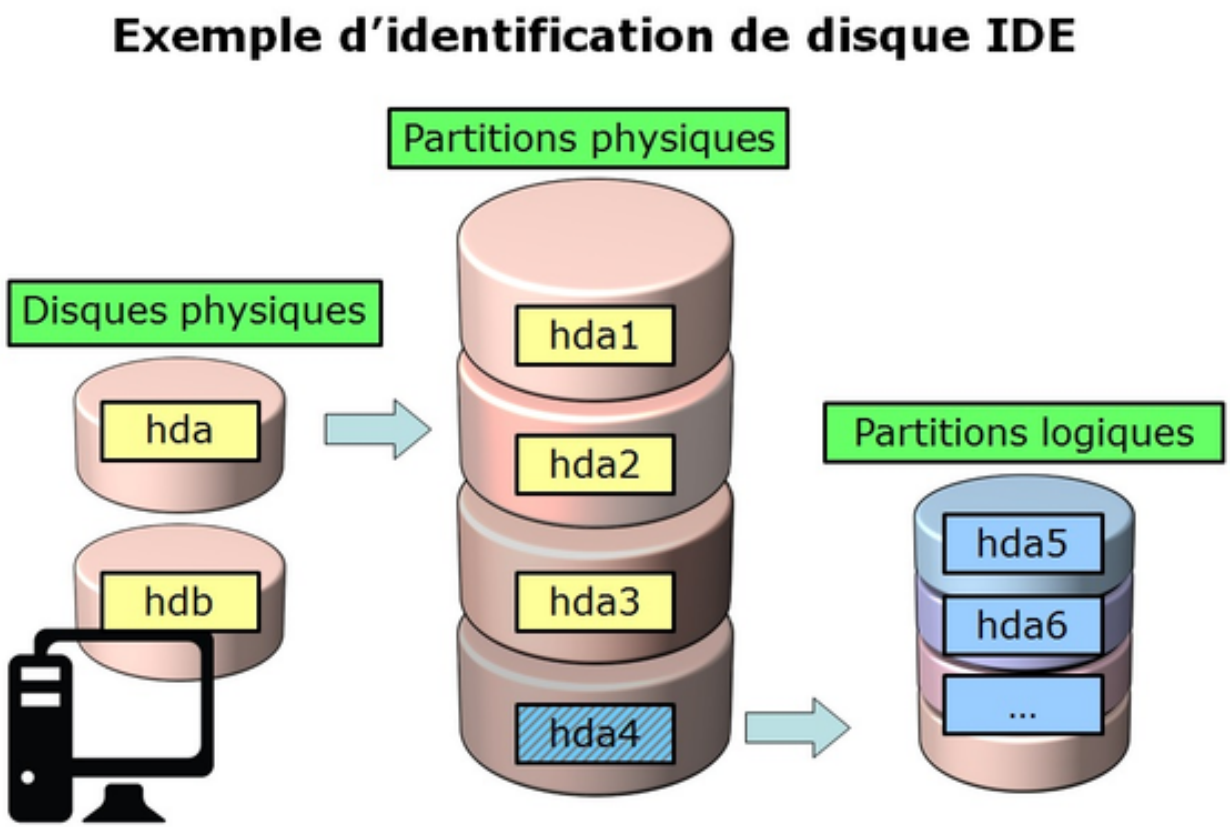


Figure 12. Identification des partitions

Il existe deux commandes permettant le partitionnement d'un disque : **fdisk** et **cfdisk**. Ces deux commandes possèdent un menu interactif. **cfdisk** étant plus fiable et mieux optimisée, il est préférable de l'utiliser.

La seule raison d'utiliser **fdisk** est lorsqu'on souhaite lister tous les périphériques logiques avec l'option **-l**.

```
[root]# fdisk -l
[root]# fdisk -l /dev/sdc
[root]# fdisk -l /dev/sdc2
```

Commande parted

La commande `parted` indexterm[parted] (**partition editor**) est capable de partitionner un disque.

Syntaxe de la commande parted

```
parted [-l] [device]
```

Elle dispose également d'une fonction de récupération capable de réécrire une table partition effacée.

La fonctionnalité de redimensionnement des partitions a été supprimé avec la version 2.4.

Sous interface graphique, il existe l'outil très complet `gparted` : *G*nome *PAR*tition *ED*itor. indexterm[gparted]

La commande `parted -l` permet de lister tous les périphériques logiques d'un ordinateur.

La commande `parted` exécutée seule renverra vers un mode interactif avec ses propres options internes :

- `help` ou une commande incorrecte affichera ces options.
- `print all` dans ce mode aura le même résultat que `parted –l` en ligne de commande.
- `quit` pour revenir sur le prompt

Commande cfdisk

La commande **cfdisk** permet de gérer les partitions

Syntaxe de la commande cfdisk

```
cfdisk device
```

Exemple :

```
[root]# cfdisk /dev/sda
             cfdisk (util-linux-ng 2.17.2)
               Unité disque : /dev/sda
         Taille: 10737418240 octets, 10.7 Go
   Têtes: 255 Secteurs par piste: 63 Cylindres: 1305
    Nom  Fanions  Part Type Sys.Fic  Étiq. Taille
    ------------------------------------------
                      ...
[aide] [nouvelle] [afficher] [quitter] [unités] [ecrire]
```

La préparation, sans LVM, du support physique passe par cinq étapes :

- Mise en place du disque physique ;
- Partitionnement des volumes (découpage géographique du disque, possibilité d'installer plusieurs systèmes, ...) ;
- Création des systèmes de fichiers (permet au système d'exploitation de gérer les fichiers, l'arborescence, les droits, ...) ;
- Montage des systèmes de fichiers (inscription du système de fichiers dans l'arborescence) ;
- Gérer l'accès aux utilisateurs.

6.2. Logical Volume Manager (LVM)

Logical **V**olume **M**anager (**LVM**)

La gestion de volume crée une couche abstraite sur un stockage physique offrant des avantages par rapport à l'utilisation directe du stockage physique :

- Capacité du disque plus flexible ;
- Déplacement des données en ligne ;
- Disques en mode "stripe" (découpage) ;
- Volumes miroirs (recopie) ;
- Instantanés de volumes (snapshot).

L'inconvénient est que si un des volumes physiques devient HS, alors c'est l'ensemble des volumes logiques qui utilisent ce volume physique qui sont perdus. Il faudra utiliser LVM sur des disques raid.

Le LVM est disponible sous Linux à partir de la version 2.4 du noyau.

LVM est uniquement géré par le système d'exploitation. Par conséquent le BIOS a besoin d'au moins une partition sans LVM pour démarrer.

Les groupes de volume

Les volumes physiques **PV** (issus des partitions) sont combinés en des groupes de volumes **VG**. Chaque **VG** représente un espace disque pouvant être découpé en volumes logiques **LV**. **L'extension** est la plus petite unité d'espace de taille fixe pouvant être allouée.

- **PE** : Physical Extension
- **LE** : Logical Extension

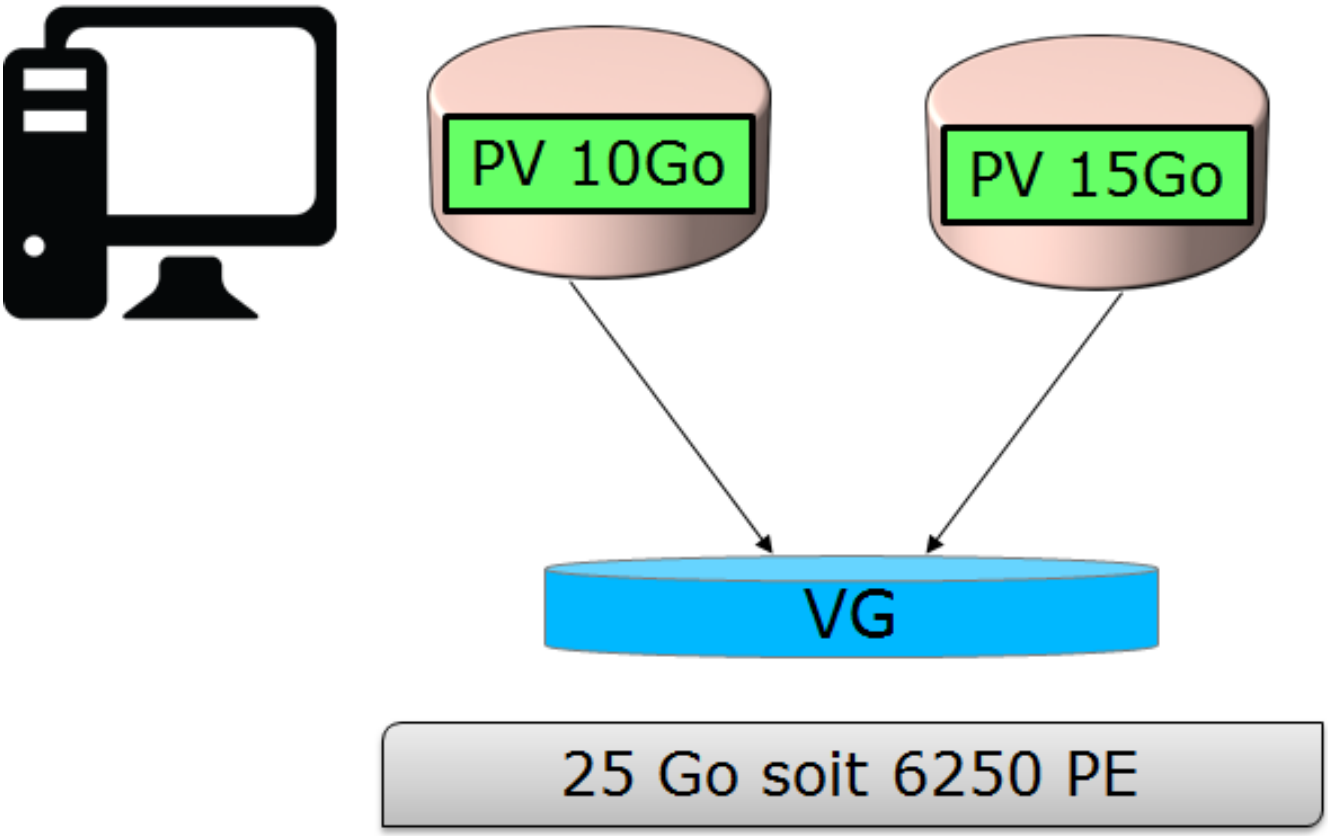


Figure 13. Groupe de volumes, taille de PE égale à 4Mo

Les volumes logiques

Un groupe de volume **VG** est divisé en volumes logiques **LV** offrant différents modes de fonctionnement :

- Volumes linéaires ;
- Volumes en mode stripe ;
- Volumes en miroirs.

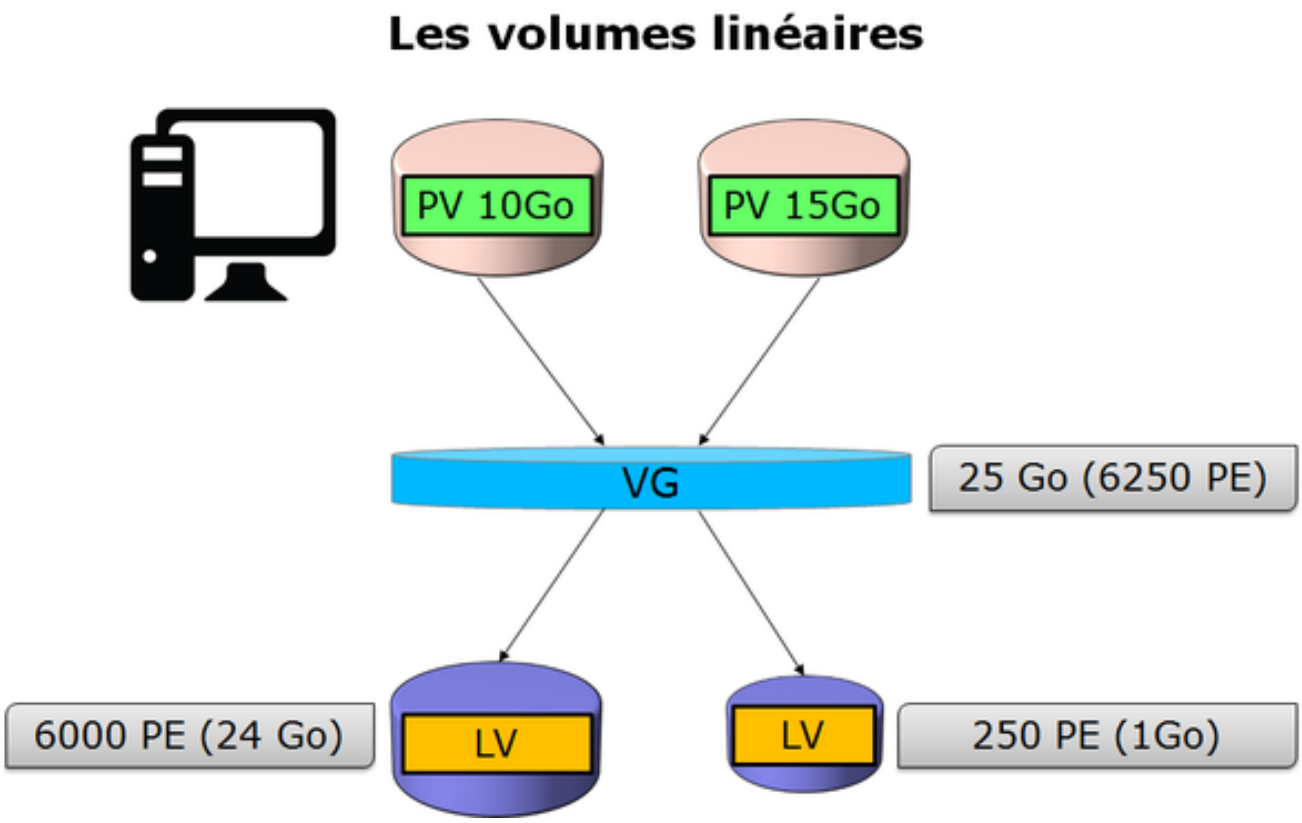


Figure 14. Volumes linéaires

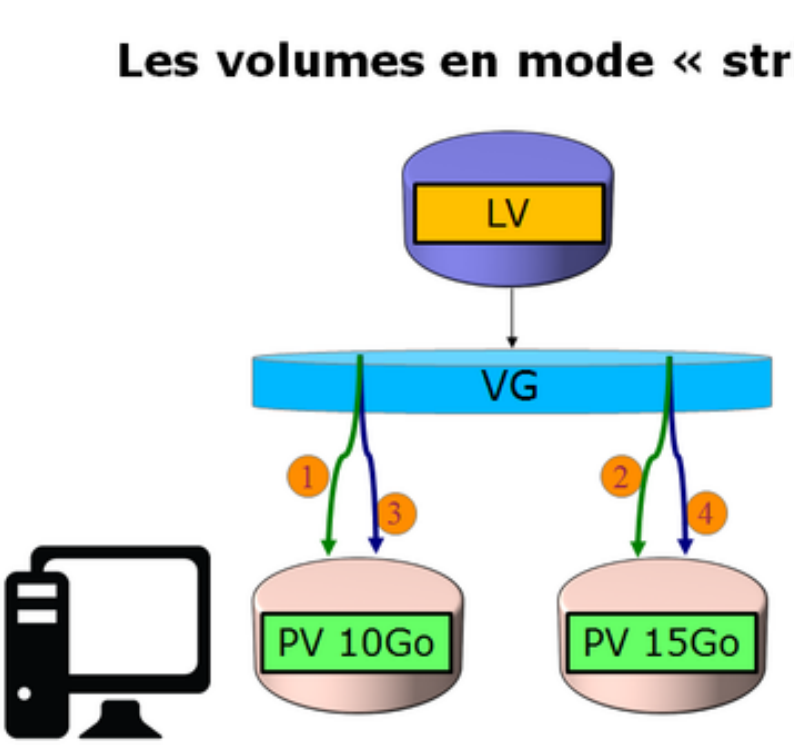


Figure 15. Volumes en mode stripe

Le “striping” améliore les performances en écrivant des données sur un nombre prédéterminé de volumes physiques avec une technique de round-robin.

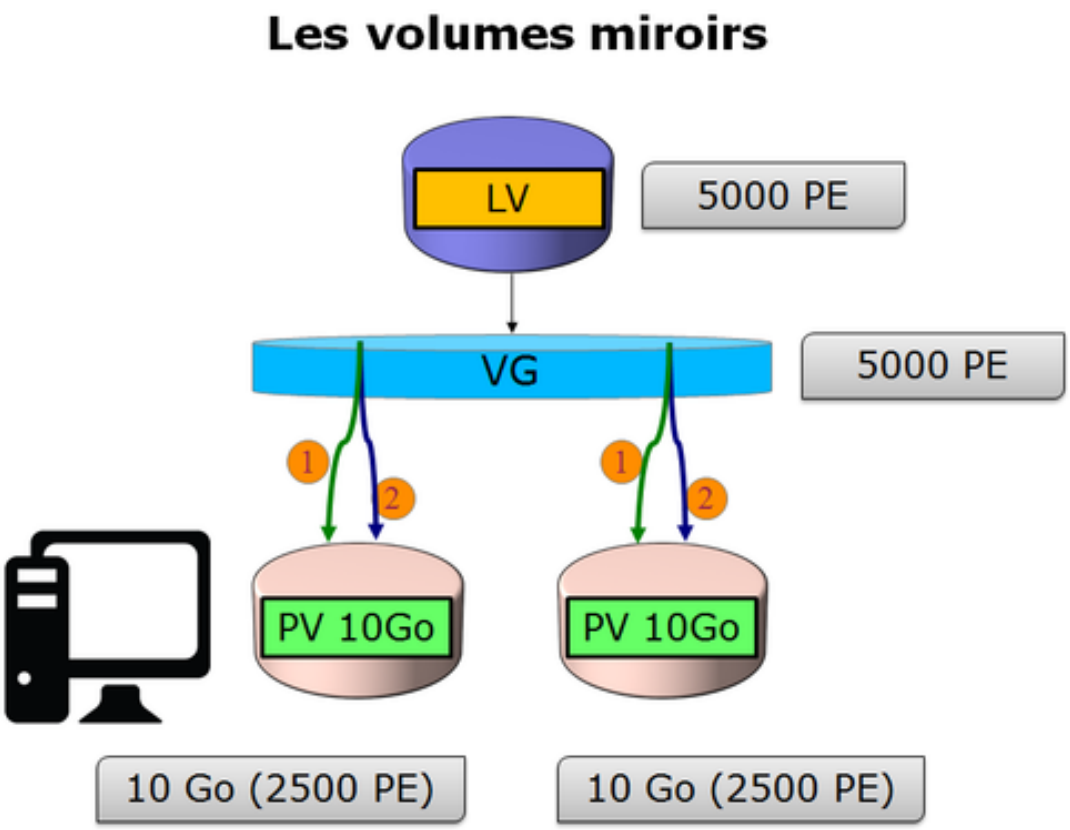


Figure 16. Volumes en miroirs

Commandes LVM pour la gestion des volumes

Commande pvcreate

La commande **pvcreate** permet de créer des volumes physiques. Elle transforme des partition Linux en volumes physiques.

Syntaxe de la commande pvcreate

```
pvcreate [-options] partition
```

Exemple :

```
[root]# pvcreate /dev/hdb1
pvcreate -- physical volume « /dev/hdb1 » successfuly created
```

Table 36. Option de la commande pvcreate

Option	Description
-f	Force la création du volume (disque déjà transformé en volume physique).

Commande vgcreate

La commande **vgcreate** permet de créer des groupes de volumes. Elle regroupe un ou plusieurs volumes physiques dans un groupe de volumes.

Syntaxe de la commande vgcreate

```
vgcreate volume physical_volume [PV...]
```

Exemple :

```
[root]# vgcreate volume1 /dev/hdb1
...
vgcreate – volume group « volume1 » successfuly created and activated
```

Commande lvcreate

La commande **lvcreate** permet de créer des volumes logiques. Le système de fichiers est ensuite créé sur ces volumes logiques.

Syntaxe de la commande lvcreate

```
lvcreate -L taille [-n nom] nom_VG
```

Exemple :

```
[root]# lvcreate –L 600M –n VolLog1 volume1
lvcreate -- logical volume « /dev/volume1/VolLog1 » successfuly created
```

Table 37. Options de la commande lvcreate

Option	Description
-L taile	Taille du volume logique en K, M ou G
-n nom	Nom du LV. Fichier spécial créé dans /**dev**/**nom_volume** portant ce nom

Commandes LVM pour visualiser les informations concernant les volumes

Commande pvdisplay

La commande **pvdisplay** permet de visualiser les informations concernant les volumes physiques.

Syntaxe de la commande pvdisplay

```
pvdisplay /dev/nom_PV
```

Exemple :

```
[root]# pvdisplay /dev/nom_PV
```

Commande vgdisplay

La commande **vgdisplay** permet de visualiser les informations concernant les groupes de volumes.

Syntaxe de la commande vgdisplay

```
vgdisplay nom_VG
```

Exemple :

```
[root]# vgdisplay volume1
```

Commande lvdisplay

La commande **lvdisplay** permet de visualiser les informations concernant les volumes logiques.

Syntaxe de la commande lvdisplay

```
lvdisplay /dev/nom_VG/nom_LV
```

Exemple :

```
[root]# lvdisplay /dev/volume1/VolLog1
```

Préparation du support physique

La préparation avec LVM du support physique se décompose comme suit :

- Mise en place du disque physique
- Partitionnement des volumes
- **Volume physique LVM**
- **Groupes de volumes LVM**
- **Volumes logiques LVM**
- Création des systèmes de fichiers
- Montage des systèmes de fichiers
- Gérer l'accès aux utilisateurs

6.3. Structure d'un système de fichiers

Un système de fichiers **SF** peut se nommer système de gestion de fichiers **SGF** mais également file system **FS**.

Un système de fichiers est en charge des actions suivantes :

- Sécuriser les droits d'accès et de modification des fichiers ;
- Manipuler des fichiers : créer, lire, modifier et supprimer ;
- Localiser les fichiers sur le disque ;
- Gérer l'espace mémoire.

Le système d'exploitation Linux est capable d'exploiter différents systèmes de fichiers (ext2, ext3, ext4, FAT16, FAT32, NTFS, HFS, BtrFS, JFS, XFS, ...).

Commande mkfs

La commande mkfs permet de créer un système de fichiers Linux.

Syntaxe de la commande mkfs

```
mkfs [-t fstype] filesys
```

Exemple :

```
[root]# mkfs -t ext4 /dev/sda1
```

Table 38. Option de la commande mkfs

Option	Description
-t	Indique le type de système de fichiers à utiliser

Sans système de fichiers il n'est pas possible d'utiliser l'espace disque.

Chaque système de fichiers possède une structure qui est identique sur chaque partition. Un Bloc de Boot et un Super Bloc initialisés par le système puis une Table des Inodes et une Zone de Données initialisées par l'administrateur.

La seule exception est concernant la partition **swap**.

Bloc de boot

Le bloc de boot occupe le premier bloc sur le disque et est présent sur toutes les partitions. Il contient le programme assurant le démarrage et l'initialisation du système et n'est donc renseigné

que pour la partition de démarrage.

Super bloc

La taille de la table du **super bloc** est définie à la création. Il est présent sur chaque partition et contient les éléments nécessaires à l'exploitation de celle-ci.

Il décrit le Système de Fichiers :

- Nom du Volume Logique ;
- Nom du Système de Fichiers ;
- Type du Système de Fichiers ;
- État du Système de Fichiers ;
- Taille du Système de Fichiers ;
- Nombre de blocs libres ;
- Pointeur sur le début de la liste des blocs libres ;
- Taille de la liste des inodes ;
- Nombre et la liste des inodes libres.

Une copie est chargée en mémoire centrale dès l'initialisation du système. Cette copie est mise à jour dès modification et le système la sauvegarde périodiquement (commande sync). Lorsque le système s'arrête, il recopie également cette table en mémoire vers son bloc.

Table des inodes

La taille de la table des inodes est définie à sa création et est stockée sur la partition. Elle se compose d'enregistrements, appelés inodes, correspondant aux fichiers créés. Chaque enregistrement contient les adresses des blocs de données constituant le fichier.

Un numéro d'inode est unique au sein d'un système de fichiers.

Une copie est chargée en mémoire centrale dès l'initialisation du système. Cette copie est mise à jour dès modification et le système la sauvegarde périodiquement (commande sync). Lorsque le système s'arrête, il recopie également cette table en mémoire vers son bloc. Un fichier est géré par son numéro d'inode.

La taille de la table des inodes détermine le nombre maximum de fichiers que peut contenir le SF.

Informations présentes dans la table des inodes :

- Numéro d'inode ;
- Type de fichier et permissions d'accès ;

- Numéro d'identification du propriétaire ;
- Numéro d'identification du groupe propriétaire ;
- Nombre de liens sur ce fichier ;
- Taille du fichier en octets ;
- Date du dernier accès au fichier ;
- Date de la dernière modification du fichier ;
- Date de la dernière modification de l'inode (= création) ;
- Tableau de plusieurs pointeurs (table de blocs) sur les blocs logiques contenant les morceaux du fichier.

Zone de données

Sa taille correspond au reste de l'espace disponible de la partition. Cette zone contient les catalogues correspondant à chaque répertoire ainsi que les blocs de données correspondant aux contenus des fichiers.

Afin de garantir la cohérence du système de fichiers, une image du super-bloc et de la table des inodes est chargée en mémoire (RAM) lors du chargement du système d'exploitation afin que toutes les opérations d'E/S se fassent à travers ces tables du système. Lorsque l'utilisateur crée ou modifie des fichiers, c'est en premier lieu cette image mémoire qui est actualisée. Le système d'exploitation doit donc régulièrement actualiser le super-bloc du disque logique (commande sync).

Ces tables sont inscrites sur le disque dur lors de l'arrêt du système.

En cas d'arrêt brutal, le système de fichiers peut perdre sa cohérence et provoquer des pertes de données.

Réparation du système de fichiers

Il est possible de vérifier la cohérence d'un système de fichiers à l'aide de la commande **fsck**.

En cas d'erreurs, des solutions sont proposées afin de réparer les incohérences. Après réparation, les fichiers restant sans entrées dans la table des inodes sont rattachés au dossier **/lost+found** du lecteur logique.

Commande fsck

La commande **fsck** est un outil en mode console de contrôle d'intégrité et de réparation pour les systèmes de fichiers Linux.

Syntaxe de la commande fsck

```
fsck [-sACVRTNP] [ -t fstype ] filesys
```

Exemple :

```
[root]# fsck /dev/sda1
```

Pour vérifier la partition racine, il est possible de créer un fichier forcefsck et de redémarrer ou de faire un shutdown avec l'option -F.

```
[root]# touch /forcefsck
[root]# reboot
ou
[root]# shutdown –r -F now
```

La partition devant être vérifiée doit impérativement être démontée.

6.4. Organisation d'un système de fichiers

Par définition, un Système de Fichiers est une structure arborescente de répertoires construite à partir d'un répertoire racine (un périphérique logique ne peut contenir qu'un seul système de fichiers).

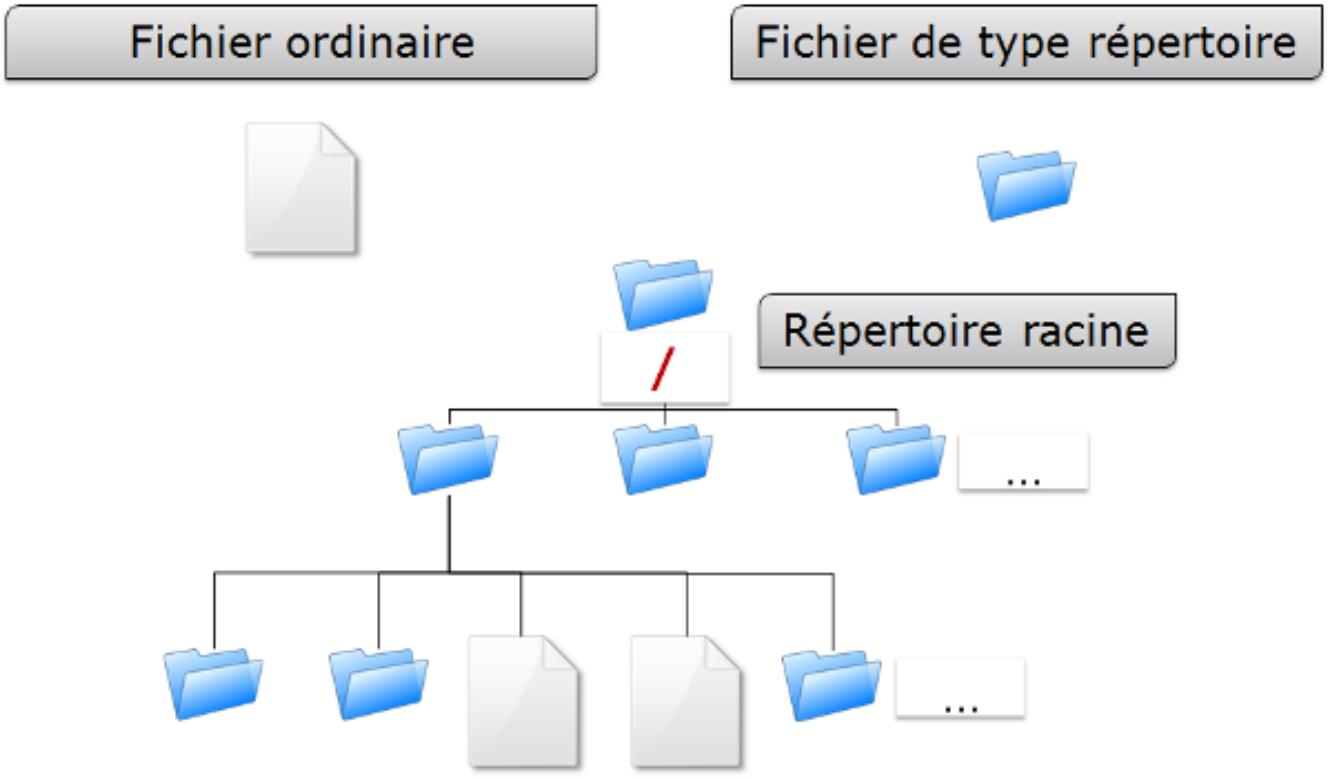


Figure 17. Organisation du système de fichiers

Sous Linux, tout est fichier.

Document texte, répertoire, binaire, partition, ressource réseau, écran, clavier, noyau Unix,

programme utilisateur, …

Linux répond à la norme FHS (Filesystems Hierarchy Standard) qui définit le nom des dossiers et leurs rôles.

Table 39. Organisation standard du système de fichiers

Répertoire	**Observation**	**Abréviation**
/	Contient les répertoires spéciaux	
/boot	Fichiers relatifs au démarrage du système	
/sbin	Commandes indispensables au démarrage système	system binaries
/bin	Exécutables des commandes de base du système	binaries
/usr/bin	Commandes d'administration système	
/lib	Librairies partagées et modules du noyau	libraries
/usr	Tout ce qui n'est pas nécessaire au fonctionnement minimal du système	UNIX System Resources
/mnt	Pour le montage de SF temporaires	mount
/media	Pour le montage de médias amovibles	
/root	Répertoire de connexion de l'administrateur	
/home	Données utilisateurs	
/tmp	Fichiers temporaires	temporary
/dev	Fichiers spéciaux des périphériques	device
/etc	Fichiers de configuration et de scripts	editable text configuration
/opt	Spécifiques aux applications installées	optional
/proc	Système de fichiers virtuel représentant les différents processus	processes
/var	Fichiers variables divers	variables

Montage, démontage…quelques affirmations :

- Pour effectuer un montage ou démontage, au niveau de l'arborescence, il ne faut pas se trouver sous le point de montage.
- Le montage sur un répertoire non vide n'efface pas le contenu. Il est seulement masqué.
- Seul l'administrateur peut effectuer des montages.
- Les points de montage devant être montés automatiquement au démarrage doivent être inscrit dans **/etc/fstab**.

Le fichier /etc/fstab

Ce fichier est lu au démarrage du système et contient les montages à effectuer. Chaque système de fichiers à monter est décrit sur une seule ligne, les champs étant séparés pas des espaces ou des tabulations.

Les lignes sont lues séquentiellement (fsck, mount, umount).

Structure du fichier **/etc/fstab**

```
/dev/mapper/VolGroup-lv_root   /         ext4    defaults        1   1
UUID=46⋯.92                     /boot     ext4    defaults        1   2
/dev/mapper/VolGroup-lv_swap   swap      swap    defaults        0   0
tmpfs                          /dev/shm  tmpfs   defaults        0   0
devpts                         /dev/pts  devpts  gid=5,mode=620  0   0
sysfs                          /sys      sysfs   defaults        0   0
proc                           /proc     proc    defaults        0   0

  1                              2         3        4             5   6
```

Champ	Description
1	Périphérique du système de fichiers (/dev/sda1, UUID=..., ...)
2	Nom du point de montage, **chemin absolu** (excepté **swap**)
3	Type de système de fichiers (ext4, swap, ...)
4	Options particulières pour le montage (defaults, ro, ...)
5	Active ou non la gestion des sauvegardes (0:non sauvegardé, 1:sauvegardé)
6	Ordre de vérification lors du contrôle du SF par la commande **fsck** (0:pas de contrôle, 1:prioritaire, 2:non prioritaire)

La commande **mount -a** permet de prendre en compte les nouveaux montages sans redémarrage. Ils sont ensuite inscrits dans le fichier **/etc/mtab** qui contient les montages actuels.

Seuls les points de montages inscrits dans **/etc/fstab** seront montés au redémarrage.

Il est possible de faire une copie du fichier **/etc/mtab** ou de copier son contenu vers **/etc/fstab**.

Commandes de gestion des montages

Commande mount

La commande **mount** permet de monter et de visualiser les lecteurs logiques dans l'arborescence.

Syntaxe de la commande mount

```
mount [-option] [device] [directory]
```

Exemple :

```
[root]# mount /dev/sda7 /home
```

Table 40. Options de la commande mount

Option	Description
-n	Monte sans écrire dans /etc/fstab
-t	Indique le type de système de fichiers à utiliser
-a	Monte tous les systèmes de fichiers mentionnés dans /etc/fstab
-r	Monte le système de fichiers en lecture seule (équivalent **-o ro**)
-w	Monte le système de fichiers en lecture/écriture, par défaut (équivalent **-o rw**)
-o	Argument suivi d'une liste d'option(s) séparée(s) par des virgules (remount, ro, ...)

La commande **mount** seule permet de visualiser tous les systèmes de fichiers montés.

Commande umount

La commande **umount** permet de démonter les lecteurs logiques.

Syntaxe de la commande umount

```
umount [-option] [device] [directory]
```

Exemple :

```
[root]# umount /home
[root]# umount /dev/sda7
```

Table 41. Options de la commande umount

Option	Description
-n	Démonte sans écrire dans /etc/fstab
-r	Si le démontage échoue, remonte en lecture seule
-f	Force le démontage

Option	Description
-a	Démonte tous les systèmes de fichiers mentionnés dans /etc/fstab

Pour le démontage, il ne faut pas rester en dessous du point de montage. Sinon, le message d'erreur suivant s'affiche : **"device is busy"**.

6.5. Types de fichiers

Comme dans tout système, afin de pouvoir se retrouver dans l'arborescence et la gestion des fichiers, il est important de respecter des règles de nommage des fichiers.

- Les fichiers sont codés sur 255 caractères ;
- Tous les caractères ASCII sont utilisables ;
- Les majuscules et minuscules sont différenciées ;
- Pas de notion d'extension.

Les groupes de mots séparés par des espaces doivent être encadrés par des guillemets :

```
[root]# mkdir "repertoire travail"
```

Le **.** sert seulement à cacher un fichier quand il débute le nom.

Sous Linux, la notion d'extension n'existe pas. Cependant, elle peut être utilisée mais fait alors partie intégrante du nom du fichier.

Exemples de conventions d'extension :

- .c : fichier source en langage C ;
- .h : fichier d'en-tête C et Fortran ;
- .o : fichier objet en langage C ;
- .tar : fichier de données archivées avec l'utilitaire tar ;
- .cpio : fichier de données archivées avec l'utilitaire cpio ;
- .gz : fichier de données compressées avec l'utilitaire gzip ;
- .html : page web.

Détails du nom d'un fichier

```
[root]# ls -liah /usr/bin/passwd
18 -rwxr-xr-x. 1 root root 26K 22 févr. 2012 /usr/bin/passwd
1  2    3      4  5    6    7       8              9
```

Champ	Description
1	Numéro d'inode
2	Type de fichiers
3	Droits d'accès
4	Nombre de liens (ordinaire) ou sous-répertoires (répertoires)
5	Nom du propriétaire
6	Nom du groupe
7	Taille (octet, kilo, méga)
8	Date de la dernière mise à jour
9	Nom du fichier

Différents types de fichiers

On retrouve sur un système les types de fichiers suivants :

- Ordinaires (textes, binaires, ...) ;
- Répertoires ;
- Spéciaux (imprimantes, écrans, ...) ;
- Liens ;
- Communications (tubes et socket).

Fichiers ordinaires

Ce sont des fichiers textes, programmes (sources), exécutables (après compilation) ou fichiers de données (binaires, ASCII) et multimédias.

```
[root]# ls -l fichier
-rwxr-xr-x   1   root  root  26  nov  31  15:21 fichier
```

Le tiret **-** au début du groupe de droits indique qu'il s'agit d'un fichier de type ordinaire.

Fichiers répertoires

Les fichiers de type répertoire contiennent des références à d'autres fichiers.

Par défaut dans chaque répertoire sont présents **.** et **..** .

Le **.** représente la position dans l'arborescence.

Le **..** représente le père de la position courante.

```
[root]# ls -l repertoire
drwxr-xr-x   1   root  root  26  nov  31  15:21 repertoire
```

La lettre **d** au début du groupe de droits indique qu'il s'agit d'un fichier de type répertoire.

Fichiers spéciaux

Afin de communiquer avec les périphériques (disques durs, imprimantes, ...), Linux utilise des fichiers d'interface appelés fichiers spéciaux (device file ou special file). Ils permettent donc d'identifier les périphériques.

Ces fichiers sont particuliers car ils ne contiennent pas de données mais spécifient le mode d'accès pour communiquer avec le périphérique.

Ils sont déclinés en deux modes :

- mode **bloc** ;
- mode **caractère**.

Le fichier spécial **mode bloc** permet en utilisant les buffers système de transférer des données vers le périphérique.

```
[root]# ls -l /dev/sda
brw-------   1   root  root  8, 0 jan 1 1970 /dev/sda
```

La lettre **b** au début du groupe de droits indique qu'il s'agit d'un fichier spécial bloc.

Le fichier spécial **mode caractère** est utilisé pour transférer des données vers le périphérique sous forme de flux un caractère à la fois sans utiliser de buffer. Ce sont les périphériques comme l'imprimante, l'écran ou les bandes DAT, ...

La sortie standard est l'écran.

```
[root]# ls -l /dev/tty0
crw-------   1   root  root  8, 0 jan 1 1970 /dev/tty0
```

La lettre **c** au début du groupe de droits indique qu'il s'agit d'un fichier spécial caractère.

Fichiers de communication

Il s'agit des fichiers tubes (pipes) et des fichiers sockets.

Les fichiers tubes passent les informations entre processus par FIFO (First In First Out). Un processus écrit de informations transitoires dans un fichier *pipe* et un autre les lit. Après lecture, les informations ne sont plus accessibles.

Les fichiers sockets permettent la communication bidirectionnelle inter-processus (sur système local ou distant). Ils utilisent un inode du système de fichiers.

Fichiers liens

Ces fichiers donnent la possibilité de donner plusieurs noms logiques à un même fichier physique. Un nouveau point d'accès au fichier est par conséquent créé.

On distingue deux types de fichiers lien :

- Les liens physiques ;
- Les liens symboliques.

Le lien physique

Le fichier lien et le fichier source ont le même numéro d'inode et le compteur de lien est incrémenté. Il est impossible de lier des répertoires et des fichiers de système de fichiers différents.

Si le fichier source est détruit, le compteur est décrémenté et le fichier lien accède toujours au fichier.

Commande ln

La commande **ln** permet de créer des liens

```
[root]# ls -li lettre
666 -rwxr--r-- 1 root root ··· lettre

[root]# ln /home/paul/lettre /home/jack/lire

[root]# ls -li /home/*/l*
666 -rwxr--r-- 2 root root ··· lettre
666 -rwxr--r-- 2 root root ··· lire
```

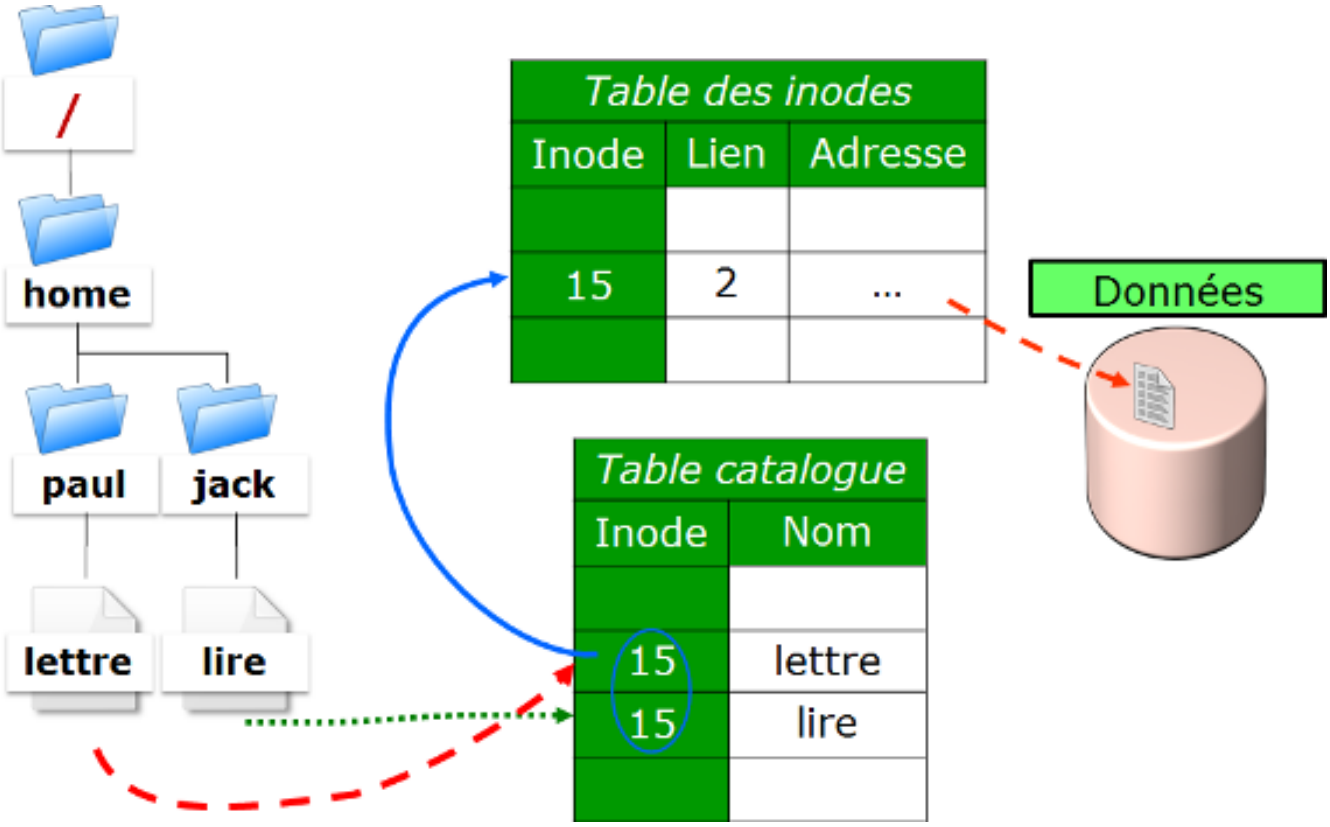


Figure 18. Représentation d'un lien physique

Le lien symbolique

Contrairement au lien physique, le lien symbolique implique la création d'un nouvel **inode**. Au niveau du lien symbolique, seul un chemin d'accès est stocké dans la table des inodes.

Le fichier créé ne contient qu'une indication sur le chemin permettant d'atteindre le fichier. Cette notion n'a plus les limitations des liens physiques et il est désormais possible de lier des répertoires et des fichiers appartenant à des systèmes de fichiers différents.

Si le fichier source est détruit, le fichier lien ne peut plus accéder au fichier.

```
[root]# ls -li lettre
666 -rwxr--r--- 1 root root ··· lettre

[root]# ln -s /home/paul/lettre /tmp/lire

[root]# ls -li /home/paul/lettre /tmp/lire
666 -rwxr--r--- 1 root root ··· lettre
678 lrwxrwxrwx 1 root root ··· lire -> lettre
```

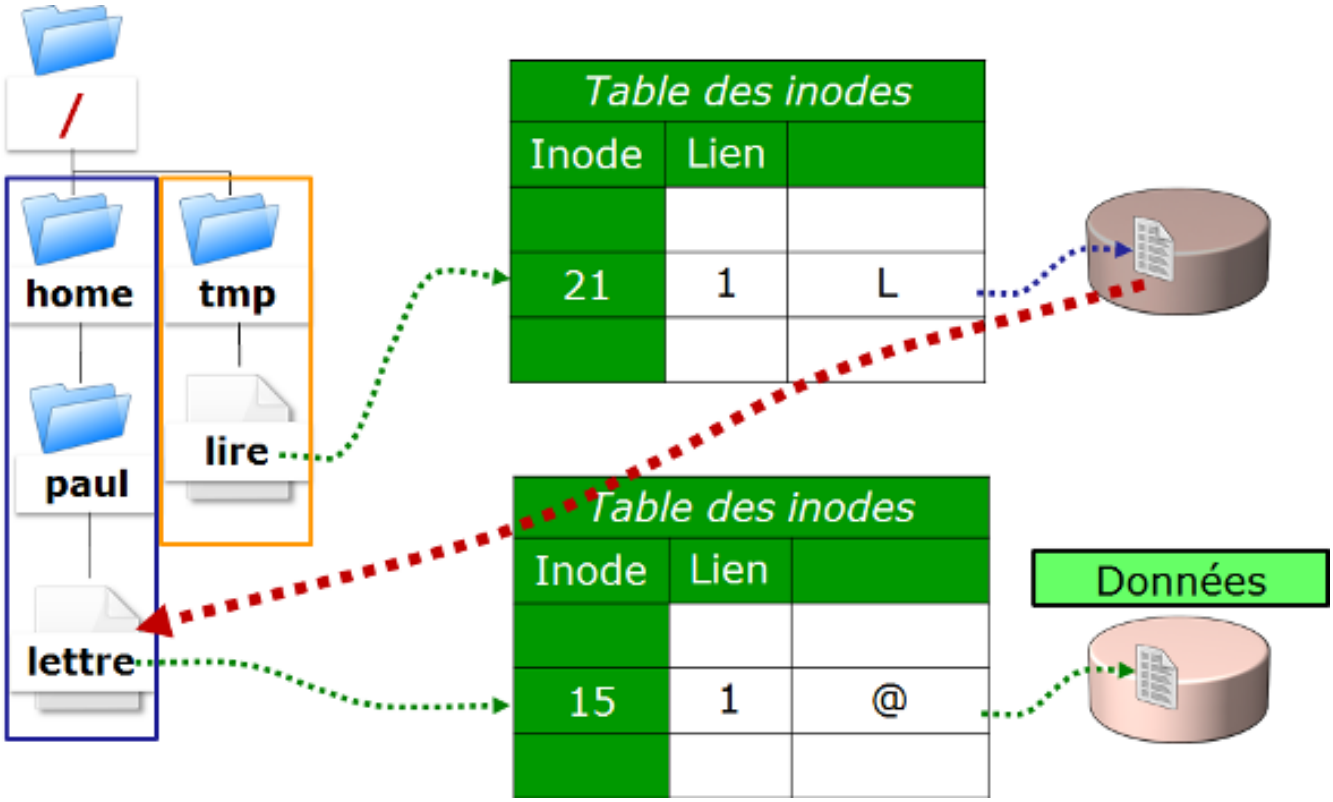


Figure 19. Représentation d'un lien symbolique

6.6. Attributs des fichiers

Linux est système d'exploitation multi-utilisateurs où l'accès aux fichiers est contrôlé.

Ces contrôles sont fonctions :

- des permissions d'accès au fichier ;
- des utilisateurs (ugo).

La commande **ls -l** permet afficher les attributs.

Il existe 4 droits d'accès aux fichiers :

- **r**ead (lecture) ;
- **w**rite (écriture) ;
- e**x**ecution (exécution) ;
- **-** aucun droit.

Les droits associés aux fichiers diffèrent de ceux associés aux répertoires (voir ci-dessous).

Les types d'utilisateurs associés aux droits d'accès des fichiers sont :

- **u**ser (propriétaire) ;

- **g**roup (groupe propriétaire) ;
- **o**thers (les autres) ;

Dans certaines commandes, il est possible de désigner tout le monde avec **a** (all).

a = **ugo**

Droits associés aux fichiers ordinaires

- **r**ead : Permet la lecture d'un fichier (cat, less, ...) et autorise la copie (cp, ...).
- **w**rite : Autorise la modification du contenu du fichier (cat, », vim, ...).
- e**x**ecute : Considère le fichier comme une commande (binaire, script).
- **-** : Aucune permission.

Déplacer ou renommer un fichier dépend des droits du répertoire cible. Supprimer un fichier dépend des droits du répertoire parent.

Droits associés aux répertoires

- **r**ead : Permet la lecture du contenu d'un répertoire (ls -R).
- **w**rite : Autorise la modification du contenu d'un répertoire (touch) et permet la **création et suppression de fichiers** si la permission **x** est activée.
- e**x**ecute : Permet de descendre dans le répertoire (cd).
- **-** : Aucun droit.

Gestion des attributs

L'affichage des droits se fait à l'aide de la commande **ls -l**

```
[root]# ls -l /tmp/fichier
-rwxrw-r-x  1  root  sys  ... /tmp/fichier
  1  2  3       4     5
```

Champ	Description
1	Permissions du propriétaire (**u**ser), ici **rwx**
2	Permissions du groupe propriétaire (**g**roup), ici **rw-**
3	Permissions des autres utilisateurs (**o**thers), ici **r-x**
4	Propriétaire du fichier
5	Groupe propriétaire du fichier

Les permissions s'appliquent sur **u**ser, **g**roup et **o**ther (**ugo**) en fonction du propriétaire et du groupe.

Par défaut, le propriétaire d'un fichier est celui qui le crée. Le groupe du fichier est le groupe du propriétaire qui a créé le fichier. Les autres sont ceux qui ne sont pas concernés par les cas précédents.

La modification des attributs s'effectue à l'aide de la commande **chmod**

Seuls l'administrateur et le propriétaire d'un fichier peuvent modifier les droits d'un fichier.

Commande chmod

La commande **chmod** permet de modifier les autorisations d'accès à un fichier.

```
chmod [option] mode fichier
```

L'indication de mode peut être une représentation octale (ex : 744) ou une représentation symbolique ([ugoa][+=-][rwxst]).

Plusieurs opérations symboliques peuvent être séparées par des virgules

Exemple :

```
[root]# chmod -R u+rwx,g+wx,o-r /tmp/fichier1
[root]# chmod g=x,o-r /tmp/fichier2
[root]# chmod -R o=r /tmp/fichier3

[root]# ls -l /tmp/fic*
-rwxrwx--- 1 root root ⋯ /tmp/fichier1
-rwx--x--- 1 root root ⋯ /tmp/fichier2
-rwx--xr-- 1 root root ⋯ /tmp/fichier3
```

```
[root]# chmod 741 /tmp/fichier1
[root]# chmod -R 744 /tmp/fichier2

[root]# ls -l /tmp/fic*
-rwxr----x 1 root root ⋯ /tmp/fichier1
-rwxr--r-- 1 root root ⋯ /tmp/fichier2
```

Option	Observation
-R	Modifier récursivement les autorisations des répertoires et de leurs contenus

Il existe deux méthodes pour effectuer les changements de droits :

- La méthode **octale** ;
- La méthode **symbolique**.

Les droits des fichiers et des répertoires ne sont pas dissociés. Pour certaines opérations, il faudra connaître les droits du répertoire contenant le fichier.

Un fichier protégé en écriture peut être supprimé par un autre utilisateur dans la mesure où les droits du répertoire qui le contient autorisent cet utilisateur à effectuer cette opération.

Principe de la méthode octale

Chaque droit possède une valeur.

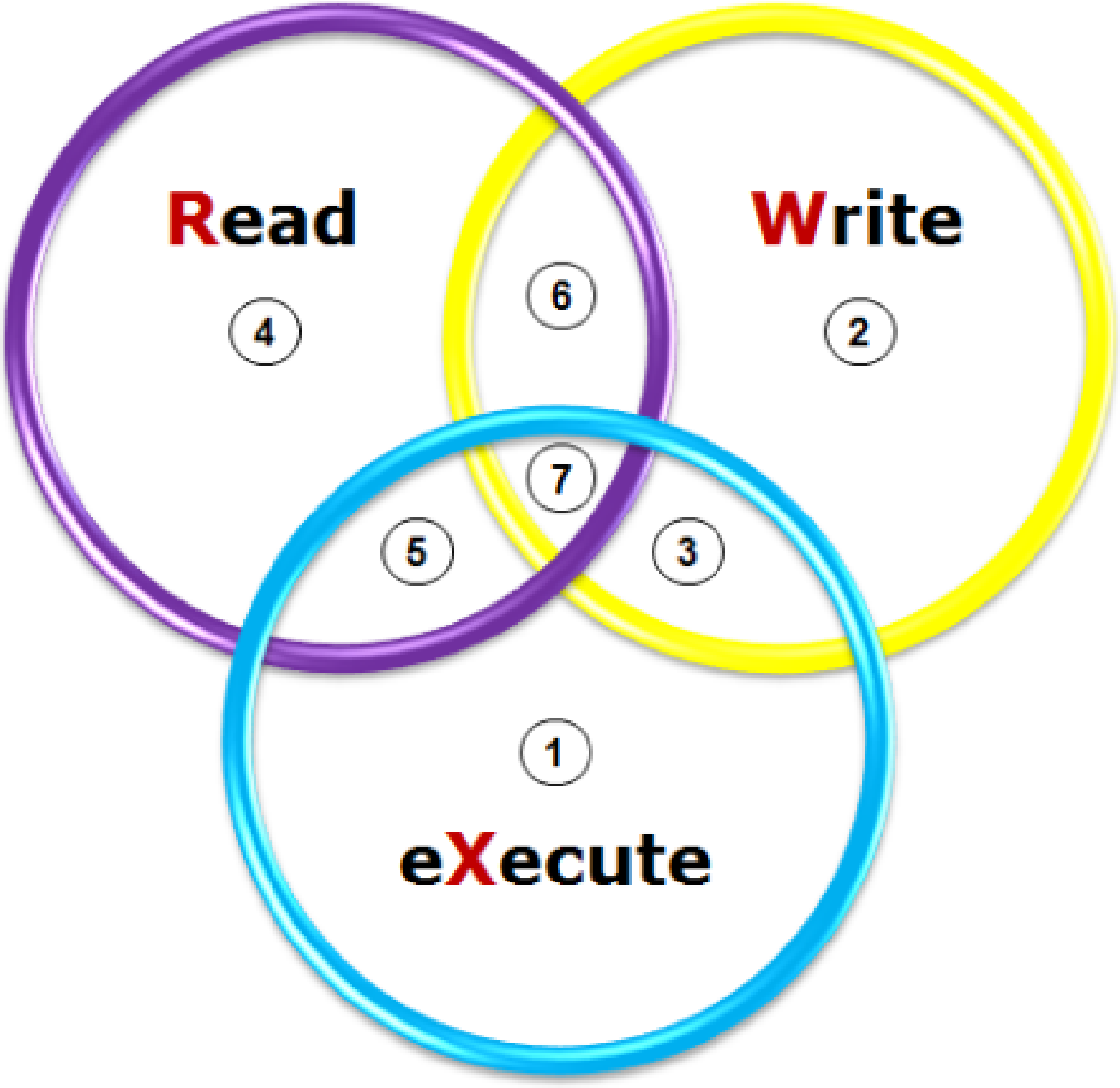


Figure 20. Méthode octale

```
[root]# ls -l /tmp/fichier
-rwxrwxrwx  1  root  root  ... /tmp/fichier
```

Figure 21. Droits 777

```
[root]# chmod 741 /tmp/fichier
-rwxr----x  1  root  root  ... /tmp/fichier
```

Figure 22. Droits 741

Principe de la méthode symbolique

Cette méthode peut être considérée comme une association “littérale” entre un type d’utilisateur, un opérateur et des droits.

Type utilisateur	
User	u
Group	g
Other	o
All	a

Opérateur	
Ajouter	+
Enlever	-
Remplacer	=

Droits	
Read	r
Write	w
eXecute	X

Figure 23. Méthode symbolique

```
[root]# chmod u+rwx,g+wx,o-r /tmp/fichier
[root]# chmod g=x,o-r /tmp/fichier
[root]# chmod o=r /tmp/fichier
```

```
[root]# ls -l /tmp/fichier
----r--r-- 1 root root ⋯ /tmp/fichier

[root]# chmod u+rwx,g+wx,o-r /tmp/fichier

[root]# ls -l /tmp/fichier
-rwxrwx--- 1 root root ⋯ /tmp/fichier
```

Les droits particuliers

En complément des droits fondamentaux (rwx), il existe les droits particuliers :

- set-user-ID (SUID)
- set-group-ID (SGID)
- sticky-bit

Comme pour les droits fondamentaux, les droits particuliers possèdent chacun une valeur. Celle-ci se place avant l'ensemble de droits **ugo**.

suid	sgid	sticky -bit	user			group			other		
s	s	t	r	w	x	r	w	x	r	w	x
4	2	1	4	2	1	4	2	1	4	2	1

Figure 24. Les droits particuliers

S, S et **T** en majuscules **si le droit n'existe pas**.

Le Sticky-bit

Une des particularités des droits sous Linux est que le droit d'écrire sur un répertoire permet également de supprimer **tous** les fichiers, propriétaire ou non.

Le sticky-bit positionné sur le répertoire ne permettra aux utilisateurs d'effacer que les fichiers dont ils sont propriétaires.

La mise en place du sticky-bit peut s'effectuer comme ci-dessous :

Méthode octale :

```
[root]# chmod 1777 repertoire
```

Méthode symbolique :

```
[root]# chmod o+t repertoire
```

```
[root]# ls -l
drwxrwxrwt ··· repertoire
```

SUID et SGID sur une commande

Ces droits permettent d'exécuter une commande suivant les droits positionnés sur la commande et non plus suivant les droits de l'utilisateur.

La commande s'exécute avec l'identité du propriétaire (**suid**) ou du groupe (**sgid**) de la commande.

L'identité de l'utilisateur demandant l'exécution de la commande n'est plus prise en compte.

Il s'agit d'une possibilité supplémentaire de droits d'accès attribués à un utilisateur lorsqu'il est nécessaire qu'il dispose des mêmes droits que ceux du propriétaire d'un fichier ou ceux du groupe concerné.

En effet, un utilisateur peut avoir à exécuter un programme (en général un utilitaire système) mais ne pas avoir les droits d'accès nécessaires. En positionnant les droits adéquats ("**s**" au niveau du propriétaire et/ou au niveau du groupe), l'utilisateur du programme possède, pour le temps d'exécution de celui-ci, l'identité du propriétaire (ou celle du groupe) du programme.

Exemple :

Le fichier **/usr/bin/passwd** est un fichier exécutable (une commande) qui porte un **SUID**.

Lorsque l'utilisateur bob va le lancer, ce dernier devra accéder au fichier **/etc/shadow**, or les droits sur ce fichier ne permettent pas à bob d'y accéder.

Ayant un **SUID** cette commande sera exécutée avec l'UID de root et le GID de root. Ce dernier étant le propriétaire du fichier **/etc/shadow**, il aura les droits en lecture.

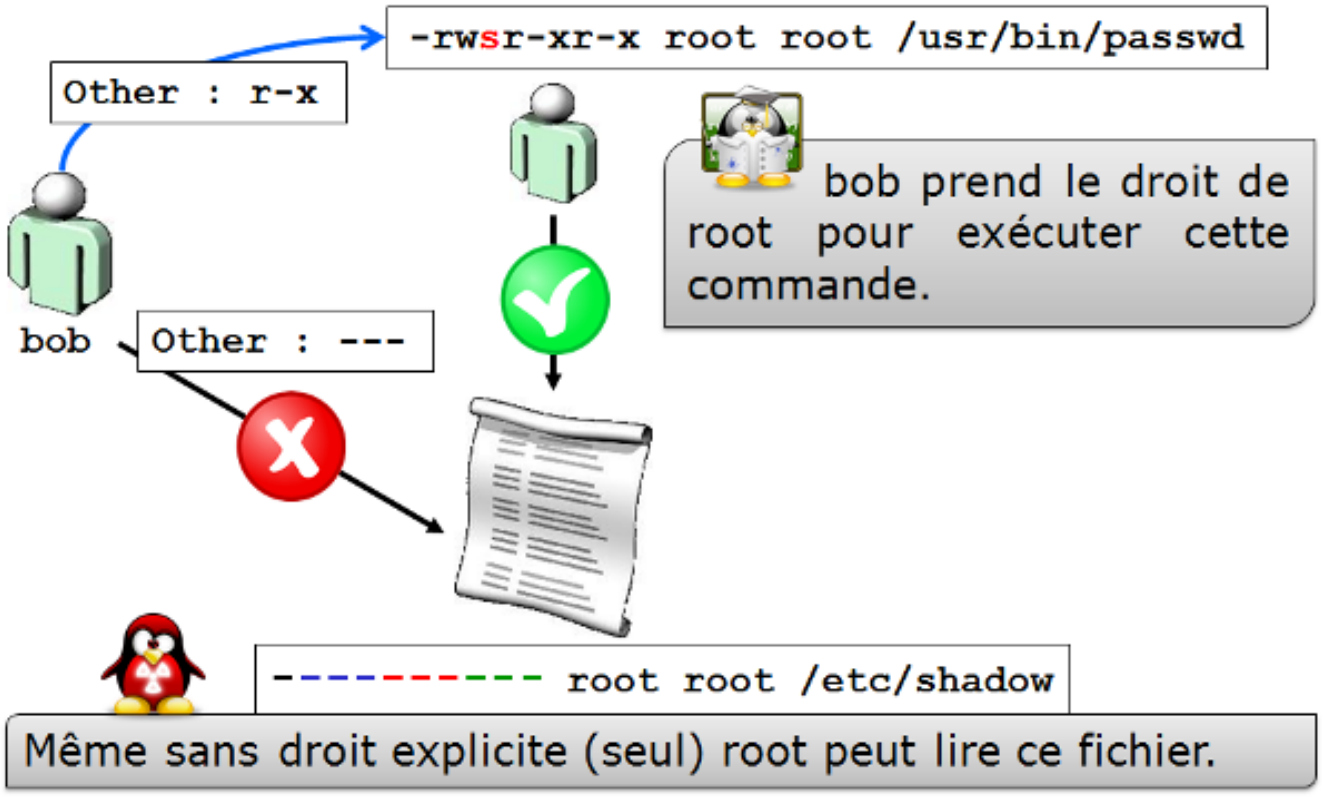


Figure 25. Fonctionnement du SUID

La mise en place du SUID et SGID peut s'effectuer comme ci-dessous :

Méthode octale :

```
[root]# chmod 4777 commande1
[root]# chmod 2777 commande2
```

Méthode symbolique :

```
[root]# chmod u+s commande1
[root]# chmod g+s commande2
```

```
[root]# ls -l
-rwsrwxrwx ⋯ commande1
-rwxrwsrwx ⋯ commande2
```

Il n'est pas possible de transmettre le SUID ou le SGID à un script shell. Le système ne le permet pas car trop dangereux pour la sécurité!

SGID sur un dossier

Dans un répertoire ayant le droit `SGID`, tout fichier créé héritera du groupe propriétaire du répertoire au lieu de celui de l'utilisateur créateur.

Exemple :

```
[stagiaire] $ ll -d /STAGE/
drwxrwsr-x 2 root users 4096 26 oct. 19:43 /STAGE
[stagiaire] $ touch /STAGE/test
[stagiaire] $ ll /STAGE
-rw-r--r--. 1 stagiaire users 0 26 oct. 19:43 test ①
```

① Le fichier `test` hérite du groupe propriétaire de son dossier `/STAGE` (ici `users`) quelque soit le groupe principal de l'utilisateur `stagiaire`.

6.7. Droits par défaut et masque

Lors de sa création, un fichier ou un répertoire possède déjà des permissions.

- Pour un répertoire : **rwxr-xr-x** soit **755**
- Pour un fichier : **rw-r–r–** soit **644**

Ce comportement est défini par le **masque par défaut**.

Le principe est d'enlever la valeur définit du masque aux droits maximums.

Pour un répertoire :

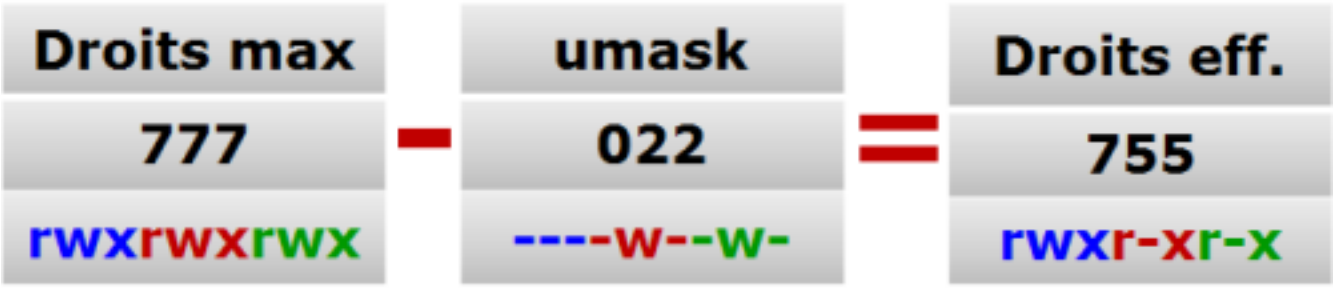


Figure 26. Droits par défaut d'un répertoire

Pour un fichier, les droits d'exécution sont retirés :

Figure 27. Droits par défaut d'un fichier

Commande umask

La commande **umask** permet d'afficher et de modifier le masque.

```
umask [option] [mode]
```

Exemple :

```
[root]# umask
0033
[root]# umask 025
[root]# umask
0025
```

Table 42. Options de la commande umask

Option	Description
-S	Affichage symbolique

umask n'affecte pas les fichiers existants.

umask modifie le masque jusqu'à la déconnexion. Pour garder la valeur, il faut modifier les fichiers de profile suivants :

Pour tous les utilisateurs :

- /etc/profile
- /etc/bashrc

Pour un utilisateur en particulier :

- ~/.bashrc

Chapitre 7. Gestion des processus

🎓 Objectifs

✔ Reconnaître le `PID` et le `PPID` d'un processus ;
✔ Afficher et rechercher des processus ;
✔ Gérer des processus.

7.1. Généralités

Un système d'exploitation se compose de processus. Ces derniers,sont exécutés dans un ordre bien précis et observent des liens de parenté entre eux. On distingue deux catégories de processus, ceux axés sur l'environnement utilisateur et ceux sur l'environnement matériel.

Lorsqu'un programme s'exécute, le système va créer un processus en plaçant les données et le code du programme en mémoire et en créant **une pile d'exécution**. Un processus est donc une instance d'un programme auquel est associé un environnement processeur (compteur ordinal, registres, etc.) et un environnement mémoire.

Chaque processus dispose :

- d'un `PID` : Process IDentifiant, identifiant unique de processus ;
- d'un `PPID` : Parent Process IDentifiant, identifiant unique de processus parent.

Par filiations successives, le processus `init` est le père de tous les processus.

- Un processus est toujours créé par un processus père ;
- Un processus père peut avoir plusieurs processus fils.

Il existe une relation père / fils entre les processus, un processus fils est le résultat de l'appel système de la primitive fork() par le processus père qui duplique son propre code pour créer un fils. Le `PID` du fils est renvoyé au processus père pour qu'il puisse dialoguer avec. Chaque fils possède l'identifiant de son père, le `PPID.

Le numéro PID représente le processus au moment de son exécution. À la fin de celui-ci, le numéro est de nouveau disponible pour un autre processus. Exécuter plusieurs fois la même commande produira à chaque fois un PID différent.

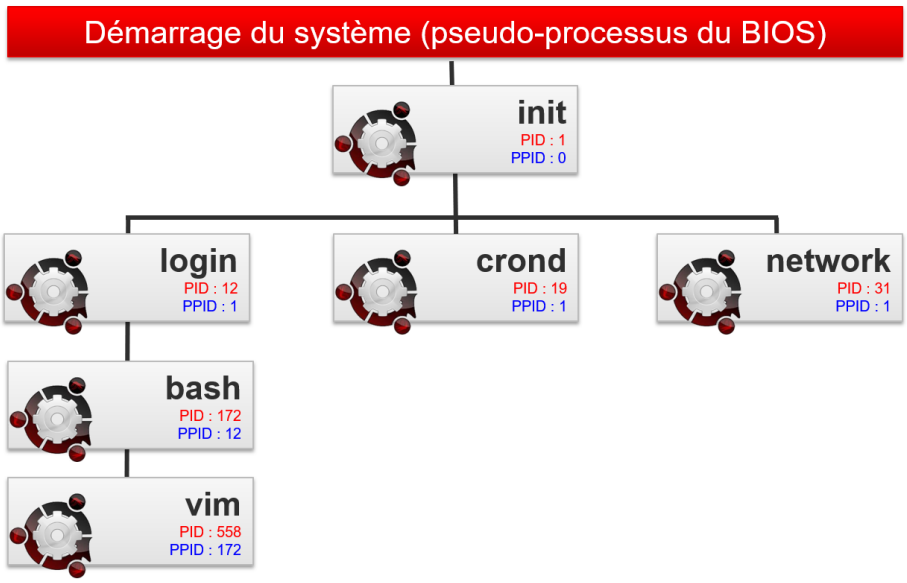


Figure 28. Filiation des processus

Les processus ne sont pas à confondre avec les threads. Chaque processus possède son propre contexte mémoire (ressources et espace d'adressage) alors que les threads issus d'un même processus partagent ce même contexte.

7.2. Visualisation des processus

La commande `ps` affiche l'état des processus en cours.

Syntaxe de la commande ps

```
ps [-e] [-f] [-u login]
```

Exemple :

```
# ps -fu root
```

Table 43. Options principales de la commande ps

Option	Description
`-e`	Affiche tous les processus.
`-f`	Affiche des informations supplémentaires.
`-u` login	Affiche les processus de l'utilisateur.

Quelques options supplémentaires :

Table 44. Options supplémentaires de la commande ps

Option	Description
`-g`	Affiche les processus du groupe.
`-t tty`	Affiche les processus exécutés à partir du terminal.
`-p PID`	Affiche les informations du processus.
`-H`	Affiche les informations sous forme d'arborescence.
`-I`	Affiche des informations supplémentaires.
`--sort COL	Trier le résultat selon une colonne.
`--headers	Affiche l'en-tête sur chaque page du terminal.
`--format "%a %b %c"`	Personnalise le format d'affichage de sortie.

Sans option précisée, la comme `ps` n'affiche que les processus exécutés à partir du terminal courant.

Le résultat est affiché par colonnes :

```
# ps -ef
UID  PID PPID C STIME  TTY TIME      CMD
root 1   0    0 Jan01  ?   00:00/03  /sbin/init
```

Table 45. Descriptions des colonnes du résultat de la commande ps -ef

Colonne	Description
`UID`	Utilisateur propriétaire.
`PID`	Identifiant du processus.
`PPID`	Identifiant du processus parent.
`C`	Priorité du processus.
`STIME`	Date et heure d'exécution.
`TTY`	Terminal d'exécution.
`TIME`	Durée de traitement.
`CMD`	Commande exécutée.

Le comportement de la commande peut être totalement personnalisé :

```
# ps -e --format "%P %p %c %n" --sort ppid --headers
 PPID   PID COMMAND          NI
    0     1 systemd           0
    0     2 kthreadd          0
    1   516 systemd-journal   0
    1   538 systemd-udevd     0
    1   598 lvmetad           0
    1   643 auditd           -4
    1   668 rtkit-daemon      1
    1   670 sssd              0
```

7.3. Types de processus

Le processus utilisateur :

- il est démarré depuis un terminal associé à un utilisateur ;
- il accède aux ressources via des requêtes ou des démons.

Le processus système (démon) :

- il est démarré par le système ;
- il n'est associé à aucun terminal et son propriétaire est un utilisateur système (souvent `root`) ;
- il est chargé lors du démarrage, il réside en mémoire et est en attente d'un appel ;
- il est généralement identifié par la lettre `d` associé au nom du processus.

Les processus système sont donc appelés démons de l'anglais `daemon` (Disk And Execution MONitor).

7.4. Permissions et droits

À l'exécution d'une commande, les identifiants de l'utilisateur sont transmis au processus créé.

Par défaut, les `UID` et `GID` effectifs (du processus) sont donc identiques aux `UID` et `GID` **réels** (les `UID` et `GID` de l'utilisateur qui a exécuté la commande).

Lorsqu'un `SUID` (et/ou un `SGID`) est positionné sur une commande, les `UID` (et/ou `GID`) effectifs deviennent ceux du propriétaire (et/ou du groupe propriétaire) de la commande et non plus celui de l'utilisateur ou du groupe utilisateur qui a lancé la commande. `UID` **effectifs et réels** sont donc **différents**.

À chaque accès à un fichier, le système vérifie les droits du processus en fonction de ses identifiants effectifs.

7.5. Gestion des processus

Un processus ne peut pas être exécuté indéfiniment, cela se ferait au détriment des autres processus en cours et empêcherait de faire du multitâche.

Le temps de traitement total disponible est donc divisé en petites plages et chaque processus (doté d'une priorité) accède au processeur de manière séquencée. Le processus prendra plusieurs états au cours de sa vie parmi les états :

- prêt : attend la disponibilité du processus ;
- en exécution : accède au processeur ;
- suspendu : en attente d'une E/S (entrée/sortie) ;
- arrêté : en attente d'un signal d'un autre processus ;
- zombie : demande de destruction ;
- mort : le père du processus tue son fils.

Le séquencement de fin de processus est le suivant :

1. Fermeture des fichiers ouverts ;
2. Libération de la mémoire utilisée ;
3. Envoi d'un signal au processus père et aux processus fils.

Lorsqu'un processus meurt, ses processus fils sont dits orphelins. Ils sont alors adoptés par le processus `init` qui se chargera de les détruire.

La priorité d'un processus

Le processeur travaille en temps partagé, chaque processus occupe un quantum de temps du processeur.

Les processus sont classés par priorité dont la valeur varie de `-20` (la priorité la plus élevée) à `+20` (la priorité la plus basse).

La priorité par défaut d'un processus est `0`.

Modes de fonctionnements

Les processus peuvent fonctionner de deux manières :

- `synchrone` : l'utilisateur perd l'accès au shell durant l'exécution de la commande. L'invite de commande réapparaît à la fin de l'exécution du processus.
- `asynchrone` : le traitement du processus se fait en arrière-plan, l'invite de commande est ré-affichée immédiatement.

Les contraintes du mode asynchrone :

- la commande ou le script ne doit pas attendre de saisie au clavier ;
- la commande ou le script ne doit pas retourner de résultat à l'écran ;
- quitter le shell termine le processus.

7.6. Les commandes de gestion des processus

La commande kill

La commande `kill` envoie un signal d'arrêt à un processus.

Syntaxe de la commande kill

```
kill [-signal] PID
```

Exemple :

```
$ kill -9 1664
```

Table 46. Codes numériques des principaux signaux d'arrêt des processus

Code	Signal	Description
2	SIGINT	Interruption du processus (*CTRL+D*)
9	SIGKILL	Terminaison immédiate du processus
15	SIGTERM	Terminaison propre du processus
18	SIGCONT	Reprise du processus
19	SIGSTOP	Suspension du processus

Les signaux sont les moyens de communication entre les processus. La commande `kill` permet d'envoyer un signal à un processus.

La liste complète des signaux pris en compte par la commande `kill` est disponible en tapant la commande :

```
$ man 7 signal
```

La commande nohup

La commande `nohup` permet de lancer un processus indépendant d'une connexion.

Syntaxe de la commande nohup

```
nohup commande
```

Exemple :

```
$ nohup MonProgramme.sh 0</dev/null &
```

`nohup` ignore le signal `SIGHUP` envoyé lors de la déconnexion d'un utilisateur.

`nohup` gère les sorties et erreur standards, mais pas l'entrée standard, d'où la redirection de cette entrée vers `/dev/null`.

[CTRL] + [Z]

En appuyant simultanément sur les touches *CTRL+Z*, le processus synchrone est temporairement suspendu. L'accès au prompt est rendu après affichage du numéro du processus venant d'être suspendu.

Instruction &

L'instruction `&` exécute la commande en mode asynchrone (la commande est alors appelée `job`) et affiche le numéro de `job`. L'accès au prompt est ensuite rendu.

Exemple :

```
$ time ls -lR / > list.ls 2> /dev/null &
[1] 15430
$
```

Le numéro de `job` est obtenu lors de la mise en tâche de fond et est affiché entre crochets, suivi du numéro de `PID`.

Les commandes fg et bg

La commande `fg` place le processus au premier plan :

```
$ time ls -lR / > list.ls 2>/dev/null &
$ fg 1
time ls -lR / > list.ls 2/dev/null
```

tandis que la commande `bg` le place à l'arrière plan :

```
[CTRL]+[Z]
^Z
[1]+ Stopped
$ bg 1
[1] 15430
$
```

Qu'il ait été mis à l'arrière plan lors de sa création grâce à l'argument `&` ou plus tard avec les touches *CTRL+Z*, un processus peut être ramené au premier plan grâce à la commande `fg` et à son numéro de job.

La commande jobs

La commande `jobs` affiche la liste des processus tournant en tâche de fond et précise leur numéro de job.

Exemple :

```
$ jobs
[1]- Running    sleep 1000
[2]+ Running    find / > arbo.txt
```

Les colonnes représentent :

1. le numéro de job ;
2. ordre de passage des processus
   - un `+` : le processus est le prochain processus à s'exécuter par défaut avec `fg` ou `bg` ;
   - un `-` : le processus est le prochain processus qui prendra le `+` ;
3. Running (en cours de traitement) ou Stopped (processus suspendu).
4. la commande

Les commandes nice/renice

La commande `nice` permet d'exécuter une commande en précisant sa priorité.

Syntaxe de la commande nice

```
nice priorité commande
```

Exemple :

```
$ nice -n+15 find / -name "fichier"
```

Contrairement à `root`, un utilisateur standard ne peut que réduire la priorité d'un processus. Seules les valeurs entre `+0` et `+19` seront acceptées.

Cette dernière limitation peut être levée par utilisateurs ou par groupes en modifiant le fichier `/etc/security/limits.conf`.

La commande `renice` permet de modifier la priorité d'un processus en cours d'exécution.

Syntaxe de la commande renice

```
renice priorité [-g GID] [-p PID] [-u UID]
```

Exemple :

```
$ renice +15 -p 1664
```

Table 47. Options principales de la commande renice

Option	Description
`-g`	`GID` du groupe propriétaire du processus.
`-p`	`PID du processus.
`-u`	`UID` du propriétaire du processus.

La commande `renice` agit sur des processus déjà en cours d'exécution. Il est donc possible de modifier la priorité d'un processus précis, mais aussi de plusieurs processus appartenant à un utilisateur ou à un groupe.

La commande `pidof`, couplée avec la commande `xargs` (voir le cours Commandes avancées), permet d'appliquer une nouvelle priorité en une seule commande :

```
$ pidof sleep | xargs renice 20
```

La commande top

La commande `top` affiche les processus et leur consommation en ressources.

```
$ top
PID  USER PR NI ... %CPU %MEM  TIME+    COMMAND
2514 root 20 0       15    5.5 0:01.14   top
```

Table 48. Descriptions des colonnes du résultat de la commande top

Colonne	Description
`PID`	Identifiant du processus.
`USER`	Utilisateur propriétaire.
`PR`	Priorité du processus.
`NI`	Valeur du nice.
`%CPU`	Charge du processeur.
`%MEM`	Charge de la mémoire.
`TIME+`	Temps d'utilisation processeur.
`COMMAND`	Commande exécutée.

La commande `top` permet de contrôler les processus en temps réel et en mode interactif.

Les commandes pgrep et pkill

La commande `pgrep` cherche parmi les processus en cours d'exécution un nom de processus et affiche sur la sortie standard les PID correspondant aux critères de sélection.

La commande `pkill` enverra le signal indiqué (par défaut `SIGTERM`) à chaque processus.

Syntaxe des commandes pgrep et pkill

```
pgrep processus
pkill [-signal] processus
```

Exemples :

- Récuperer le numéro de processus de `sshd` :

```
$ pgrep -u root sshd
```

- Tuer tous les processus `tomcat` :

```
$ pkill tomcat
```

Chapitre 8. Sauvegardes et restaurations

La sauvegarde va permettre de répondre à un besoin de conservation et de restauration des données de manière sûre et efficace.

La sauvegarde permet de se protéger des éléments suivants :

- **Destruction** : Volontaire ou involontaire. Humaine ou technique. Virus, …
- **Suppression** : Volontaire ou involontaire. Humaine ou technique. Virus, …
- **Intégrité** : Données devenues inutilisables.

Aucun système n'est infaillible, aucun humain n'est infaillible, aussi pour éviter de perdre des données, il faut les sauvegarder pour être en mesure de les restaurer suite à un problème.

Les supports de sauvegarde sont conservés dans une autre pièce (voire bâtiment) que le serveur afin qu'un sinistre ne vienne pas détruire le serveur et les sauvegardes.

De plus, l'administrateur devra régulièrement vérifier que les supports soient toujours lisibles.

8.1. Généralités

Il existe deux principes, la **sauvegarde** et l'**archive**.

- L'archive détruit la source d'information après l'opération.
- La sauvegarde conserve la source d'information après l'opération.

Ces opérations consistent à enregistrer des informations dans un fichier, sur un périphérique ou un support (bandes, disques, …).

La démarche

La sauvegarde nécessite de l'administrateur du système beaucoup de discipline et une grande rigueur. Il est nécessaire de se poser les questions suivantes :

- Quel est le support approprié ?
- Que faut-il sauvegarder ?
- Nombre d'exemplaires ?
- Durée de la sauvegarde ?
- Méthode ?
- Fréquence ?
- Automatique ou manuelle ?
- Où la stocker ?

- Délai de conservation ?

Méthodes de sauvegardes

- **Complète** : un ou plusieurs **systèmes de fichiers** sont sauvegardés (noyau, données, utilitaires, ...).
- **Partielle** : un ou plusieurs **fichiers** sont sauvegardés (configurations, répertoires, ...).
  - **Différentielle** : uniquement les fichiers modifiés depuis la dernière sauvegarde **complète** sont sauvegardés.
  - **Incrémentale** : uniquement les fichiers modifiés depuis la dernière sauvegarde sont sauvegardés.

Périodicité

- **Ponctuelle** : à un instant donné (avant une mise à jour du système, ...).
- **Périodique** : Journalière, hebdomadaire, mensuelle, ...

Avant une modification du système, il peut être utile de faire une sauvegarde. Cependant, il ne sert à rien de sauvegarder tous les jours des données qui ne sont modifiées que tous les mois.

Méthodes de restauration

En fonction des utilitaires disponibles, il sera possible d'effectuer plusieurs types de restaurations.

- **Restauration complète** : arborescences, ...
- **Restauration sélective** : partie d'arborescences, fichiers, ...

Il est possible de restaurer la totalité d'une sauvegarde mais il est également possible d'en restaurer uniquement une partie. Toutefois, lors de la restauration d'un répertoire, les fichiers créés après la sauvegarde ne sont pas supprimés.

Pour récupérer un répertoire tel qu'il était au moment de la sauvegarde il convient d'en supprimer complètement le contenu avant de lancer la restauration.

Les outils

Il existe de nombreux utilitaires pour réaliser les sauvegardes.

- **outils éditeurs** ;
- **outils graphiques** ;
- **outils mode de commande** : **tar**, **cpio**, pax, dd, dump, ...

Les commandes que nous verrons ici sont tar et cpio.

- tar :
  - simple d'utilisation ;
  - permet l'ajout de fichiers à une sauvegarde existante.
- cpio :
  - conserve les propriétaires ;
  - groupes, dates et droits ;
  - saute les fichiers endommagés ;
  - système de fichiers complet.

Ces commandes sauvegardent dans un format propriétaire et standardisé.

Convention de nommage

L'emploi d'une convention de nommage permet de cibler rapidement le contenu d'un fichier de sauvegarde et d'éviter ainsi des restaurations hasardeuses.

- nom du répertoire ;
- utilitaire employé ;
- options utilisées ;
- date.

Le nom de la sauvegarde doit être un nom explicite.

La notion d'extension sous Unix n'existe pas.

Contenu d'une sauvegarde

Une sauvegarde contient généralement les éléments suivants :

- le fichier ;
- le nom ;
- le propriétaire ;
- la taille ;
- les permissions ;
- date d'accès.

Le numéro d'inode est absent.

Modes de stockage

Deux modes de stockage se distinguent :

- fichier sur le disque ;
- périphérique.

8.2. Tape ArchiveR - tar

La commande tar permet la sauvegarde sur plusieurs supports successifs (options multi-volumes).

Il est possible d'extraire tout ou partie d'une sauvegarde.

Tar sauvegarde implicitement en mode relatif même si le chemin des informations à sauvegarder est mentionné en mode absolu.

Consignes de restauration

Il faut se poser les bonnes questions

- quoi : Partielle ou complète ;
- où : Lieu où les données seront restaurées ;
- comment : Absolu ou relatif.

Avant une restauration, il faut prendre le temps de la réflexion et déterminer la méthode la mieux adaptée afin d'éviter toutes erreurs.

Les restaurations s'effectuent généralement après un problème qui doit être résolu rapidement. Une mauvaise restauration peut dans certains cas aggraver la situation.

La sauvegarde avec tar

L'utilitaire par défaut pour créer des archives dans les systèmes UNIX est la commande tar. Ces archives peuvent être compressées avec une compression gzip ou bzip.

Tar permet d'extraire aussi bien un seul fichier ou un répertoire d'une archive, visualiser son contenu ou valider son intégrité, etc.

Créer une archive

Créer une archive non-compressée s'effectue avec les clefs cvf :

Syntaxe de la commande tar pour créer une archive

```
tar c[vf] [support] [fichiers(s)]
```

Exemple :

```
[root]# tar cvf /sauvegardes/home.133.tar /home/
```

Table 49. Clefs principales de la commande tar

Clef	Description
c	Crée une sauvegarde.
v	Affiche le nom des fichiers traités.
f	Permet d'indiquer le nom de la sauvegarde (support).

Il n'y a pas de tiret '-' devant les clefs de tar !

Créer une sauvegarde en mode absolu

Syntaxe de la commande tar pour créer une archive en mode absolu

```
tar c[vf]P [support] [fichiers(s)]
```

Exemple :

```
[root]# tar cvfP /sauvegardes/home.133.P.tar /home/
```

Clef	Description
P	Créer une sauvegarde en mode absolu.

Avec la clef **P**, le chemin des fichiers à sauvegarder doit être renseigné en **absolu**. Si les deux conditions (clef **P** et chemin **absolu**) ne sont pas indiquées, la sauvegarde est en mode relatif.

Créer une archive compressée avec gzip

Créer une archive compressée en gzip s'effectue avec les clefs cvzf :

```
[root]# tar cvzf archive.tar.gz dirname/
```

Clef	Description
z	Compresse l'archive en gzip.

L'extension .tgz est une extension équivalente à .tar.gz

Conserver les clefs 'cvf' ('tvf' ou 'xvf') inchangée pour toutes les manipulations d'archives et simplement ajouter à la fin des clefs celle de compression simplifie la compréhension de la commande (par exemple 'cvfz' ou 'cvfj', etc.).

Créer une archive compressée avec bzip

Créer une archive compressée en bzip s'effectue avec les clefs cvfj :

```
[root]# tar cvfj archive.tar.bz2 dirname/
```

Clef	Description
j	Compresse l'archive en bzip2.

Les extensions .tbz et .tb2 sont des extensions équivalentes à .tar.bz2

gzip vs bzip2

bzip2 nécessite plus de temps pour compresser ou décompresser que gzip mais offre des ratios de compression supérieurs.

Extraire (untar) une archive

Extraire (untar) une archive *.tar avec s'effectue avec les clefs xvf :

```
[root]# tar xvf /sauvegardes/etc.133.tar etc/exports
[root]# tar xvfj /sauvegardes/home.133.tar.bz2
[root]# tar xvfP /sauvegardes/etc.133.P.tar
```

Se placer au bon endroit.

Vérifier le contenu de la sauvegarde.

Clef	Description
x	Extrait des fichiers de l'archive, compressée ou non.

Extraire une archive tar-gzippée (*.tar.gz) s'effectue avec les clefs xvfz

```
[root]# tar xvfz archive.tar.gz
```

Extraire une archive tar-bzippée (*.tar.bz2) s'effectue avec les clefs xvfj

```
[root]# tar xvfj archive.tar.bz2
```

Lister le contenu d'une archive

Visualiser le contenu d'une archive sans l'extraire s'effectue avec les clefs tvf :

```
[root]# tar tvf archive.tar
[root]# tar tvfz archive.tar.gz
[root]# tar tvfj archive.tar.bz2
```

Lorsque le nombre de fichiers contenus dans une archive devient important, il est possible de passer à la commande less le résultat de la commande tar par un pipe ou en utilisant directement la commande less :

```
[root]# tar tvf archive.tar | less
[root]# less archive.tar
```

Extraire uniquement un fichier d'une archive .tar, tar.gz ou tar.bz2

Pour extraire un fichier spécifique d'une archive tar, spécifier le nom du fichier à la fin de la commande tar xvf.

```
[root]# tar xvf archive.tar /path/to/file
```

La commande précédente permet de n'extraire que le fichier file de l'archive archive.tar.

```
[root]# tar xvfz archive.tar.gz /path/to/file
[root]# tar xvfj archive.tar.bz2 /path/to/file
```

Extraire uniquement un dossier d'une archive tar, tar.gz, tar.bz2

Pour n'extraire qu'un seul répertoire (ses sous-répertoires et fichiers inclus) d'une archive, spécifier le nom du répertoire à la fin de la commande tar xvf.

```
[root] tar xvf archive.tar /path/to/dir/
```

Pour extraire plusieurs répertoires, spécifier chacun des noms les uns à la suite des autres :

```
[root] tar xvf archive_file.tar /path/to/dir1/ /path/to/dir2/
[root] tar xvfz archive_file.tar.gz /path/to/dir1/ /path/to/dir2/
[root] tar xvfj archive_file.tar.bz2 /path/to/dir1/ /path/to/dir2/
```

Extraire un groupe de fichiers d'une archive tar, tar.gz, tar.bz2 grâce à des expressions régulières (regex)

Spécifier une regex pour extraire les fichiers correspondants au pattern spécifié.

Par exemple, pour extraire tous les fichiers avec l'extension .conf :

```
[root] tar xvf archive_file.tar --wildcards '*.conf'
```

Clefs :

- --wildcards *.conf correspond aux fichiers avec l'extension .conf.

Ajouter un fichier ou un répertoire à une archive existante

Il est possible d'ajouter des fichiers à une archive existante avec la clef r.

Par exemple, pour ajouter un fichier :

```
[root]# tar rvf archive.tar filetoadd
```

Le fichier filetoadd sera ajouté à l'archive tar existante. Ajouter un répertoire est similaire :

```
[root]# tar rvf archive_name.tar dirtoadd
```

Il n'est pas possible d'ajouter des fichiers ou des dossiers à une archive compressée.

```
[root]# tar rvfz archive.tgz filetoadd
tar: Cannot update compressed archives
Try `tar --help' or `tar --usage' for more information.
```

Vérifier l'intégrité d'une archive

L'intégrité d'une archive peut être testée avec la clef W au moment de sa création :

```
[root]# tar cvfW file_name.tar dir/
```

La clef W permet également de comparer le contenu d'une archive par rapport au système de fichiers :

```
[root]# tar tvfW file_name.tar
Verify 1/file1
1/file1: Mod time differs
1/file1: Size differs
Verify 1/file2
Verify 1/file3
```

La vérification avec la clef W ne peut pas être effectuée avec une archive compressée. Il faut utiliser la clef d :

```
[root]# tar dfz file_name.tgz
[root]# tar dfj file_name.tar.bz2
```

Estimer la taille d'une archive

La commande suivante estime la taille d'un fichier tar en KB avant de la créer :

```
[root]# tar cf - /directory/to/archive/ | wc -c
20480
[root]# tar czf - /directory/to/archive/ | wc -c
508
[root]# tar cjf - /directory/to/archive/ | wc -c
428
```

Ajout d'éléments à une sauvegarde existante

Syntaxe de la commande tar pour ajouter un élément à une sauvegarde existante

```
tar {r|A}[clé(s)] [support] [fichiers(s)]
```

Exemple :

```
[root]# tar rvf /sauvegardes/home.133.tar /etc/passwd
```

Clef	Description
r	Ajoute un ou plusieurs fichiers à la fin d'une sauvegarde sur support à accès direct (disque dur).
A	Ajoute un ou plusieurs fichiers à la fin d'une sauvegarde sur support à accès séquentiel (bande).

Si la sauvegarde a été réalisée en mode relatif, ajoutez des fichiers en mode relatif. Si la sauvegarde a été réalisée en mode absolu, ajoutez des fichiers en mode absolu. En mélangeant les modes, vous risquez d'avoir des soucis au moment de la restauration.

Lire le contenu d'une sauvegarde

Syntaxe de la commande tar pour lire le contenu d'une sauvegarde

```
tar t[clé(s)] [support]
```

Exemple :

```
[root]# tar tvf /sauvegardes/home.133.tar
[root]# tar tvfj /sauvegardes/home.133.tar.bz2
```

Clef	Description
t	Affiche le contenu d'une sauvegarde (compressée ou non).

Toujours vérifier le contenu d'une sauvegarde.

Table 50. Convention d'écriture de la commande Tar

Clés	Fichiers	Suffixe
cvf	home	home.tar
cvfP	/etc	etc.P.tar
cvfz	usr	usr.tar.gz
cvfj	usr	usr.tar.bz2
cvfPz	/home	home.P.tar.gz
cvfPj	/home	home.P.tar.bz2

8.3. CoPy Input Output - cpio

La commande cpio permet la sauvegarde sur plusieurs supports successifs sans indiquer d'options.

Il est possible d'extraire tout ou partie d'une sauvegarde.

cpio ne permet pas de sauvegarder directement une arborescence. L'arborescence ou fichiers sont donc transmis sous forme de liste à cpio.

Il n'y a aucune option, comme pour la commande tar, permettant de sauvegarder et de compresser en même temps. Cela s'effectue donc en deux temps : la sauvegarde puis la compression.

Pour effectuer une sauvegarde avec cpio, il faut préciser une liste des fichiers à sauvegarder.

Cette liste est fourni avec les commandes find, ls ou cat.

- find : parcourt une arborescence, récursif ou non ;
- ls : liste un répertoire, récursif ou non ;
- cat : lit un fichier contenant les arborescences ou fichiers à sauvegarder.

ls ne peut pas être utilisé avec -l (détails) ou -R (récursif).

Il faut une liste simple de noms.

Créer une sauvegarde

Syntaxe de la commande cpio

```
[cde de fichiers |] cpio {-o| --create} [-options] [<fic-liste] [>support]
```

Exemple :

```
[root]# find /etc | cpio -ov > /sauvegardes/etc.cpio
```

Le résultat de la commande **find** est envoyé en entrée de la commande **cpio** par l'intermédiaire du signe "|" (*AltGr+6*). Ici, la commande find /etc renvoie une liste de fichiers correspondant au contenu du répertoire /etc (en récursif) à la commande cpio qui en effectue la sauvegarde. Ne surtout pas oublier le signe > lors de la sauvegarde.

Table 51. Options principales de la commande cpio

Options	Description
-o	Crée une sauvegarde (output).
-v	Affiche le nom des fichiers traités.

Options	Description
-F	Désigne la sauvegarde à modifier (support).

Sauvegarde vers un support :

```
[root]# find /etc | cpio -ov > /dev/rmt0
```

Le support peut être de plusieurs types :

- /dev/rmt0 : lecteur de bande ;
- /dev/sda5 : une partition.

Type de sauvegarde

- Sauvegarde avec chemin relatif

```
[root]# cd /
[root]# find etc | cpio -o > /sauvegardes/etc.cpio
```

- Sauvegarde avec chemin absolu

```
[root]# find /etc | cpio -o > /sauvegardes/etc.A.cpio
```

Si le chemin indiqué au niveau de la commande “find” est en **absolu** alors la sauvegarde sera réalisée en absolu.

Si le chemin indiqué au niveau de la commande “find” est en **relatif** alors la sauvegarde sera réalisée en relatif.

Ajouter à une sauvegarde

Syntaxe de la commande cpio pour ajouter un contenu

```
[cde de fichiers |] cpio {-o| --create} -A [-options] [<fic-liste] {F|>support}
```

Exemple :

```
[root]# find /etc/shadow | cpio -o -AF FicSyst.A.cpio
```

L’ajout de fichiers n’est possible que sur un support à accès direct.

Option	Description
-A	Ajoute un ou plusieurs fichiers à une sauvegarde sur disque.
-F	Désigne la sauvegarde à modifier.

Compresser une sauvegarde

- Sauvegarder **puis** compresser

```
[root]# find /etc | cpio  –o > etc.A.cpio
[root]# gzip /sauvegardes/etc.A.cpio
[root]# ls /sauvegardes/etc.A.cpio*
/sauvegardes/etc.A.cpio.gz
```

- Sauvegarder **et** compresser

```
[root]# find /etc | cpio –o | gzip > /sauvegardes/etc.A.cpio.gz
```

Il n'y a aucune option, comme pour la commande tar, permettant de sauvegarder et de compresser en même temps. Cela s'effectue donc en deux temps : la sauvegarde puis la compression.

La syntaxe de la première méthode est plus facile à comprendre et à retenir, car elle s'effectue en deux temps.

Pour la première méthode, le fichier de sauvegarde est automatiquement renommé par l'utilitaire gzip qui rajoute .gz à la fin du nom de ce fichier. De même l'utilitaire bzip2 rajoute automatiquement .bz2.

Lire le contenu d'une sauvegarde

Syntaxe de la commande cpio pour lire le contenu d'une sauvegarde cpio

```
cpio -t [-options] [<fic-liste]
```

Exemple :

```
[root]# cpio -tv </sauvegardes/etc.152.cpio | less
```

Options	Description
-t	Lit une sauvegarde.
-v	Affiche les attributs des fichiers.

Après avoir réalisé une sauvegarde, il faut lire son contenu pour être certain qu'il n'y a pas eu

d'erreur.

De la même façon, avant d'effectuer une restauration, il faut lire le contenu de la sauvegarde qui va être utilisée.

Restaurer une sauvegarde

Syntaxe de la commande cpio pour restaurer une sauvegarde

```
cpio {-i| --extract} [-E fichier] [-options] [<support]
```

Exemple :

```
[root]#cpio -iv </sauvegardes/etc.152.cpio | less
```

Options	Description
-i	Restauration complète d'une sauvegarde .
-E fichier	Restaure uniquement les fichiers dont le nom est contenu dans fichier.
-d	Reconstruit l'arborescence manquante.
-u	Remplace tous les fichiers même s'ils existent.
--no-absolute-filenames	Permet de restaurer une archive effectuée en mode absolu de manière relative.

Par défaut, au moment de la restauration, les fichiers sur le disque dont la date de dernière modification est plus récente ou égale à la date de la sauvegarde ne sont pas restaurés (afin d'éviter d'écraser des informations récentes par des informations plus anciennes).

L'option -u permet au contraire de restaurer d'anciennes versions des fichiers.

Exemples :

- Restauration en absolu d'une sauvegarde absolue :

```
[root]# cpio –iv <home.A.cpio
```

- Restauration en absolu sur une arborescence existante :

```
[root]# cpio –iuv <home.A.cpio
```

L'option "u" permet d'écraser des fichiers existants à l'endroit où s'effectue la restauration. * Restauration en relatif d'une sauvegarde absolue :

```
[root]# cpio –iv --no-absolute-filenames <home.A.cpio
```

L'option longue "--no-absolute-filenames" permet une restauration en mode relatif. En effet le "/" en début de chemin est enlevé.

- Restauration en relatif d'une sauvegarde relative :

```
[root]# cpio –iv <etc.cpio
```

- Restauration en absolu du fichier « passwd » :

```
echo "/etc/passwd" > tmp;cpio –iuE tmp <etc.A.cpio; rm -f tmp
```

8.4. Utilitaires de compression - décompression

Le fait d'utiliser la compression au moment d'une sauvegarde peut présenter un certain nombre d'inconvénients :

- Allonge le temps de la sauvegarde ainsi que celui de la restauration.
- Rend impossible l'ajout de fichiers à cette sauvegarde.

Il vaut donc mieux effectuer une sauvegarde et la compresser qu'effectuer la compression lors de la sauvegarde.

Compresser avec gzip

La commande gzip compresse les données.

Syntaxe de la commande gzip

```
gzip [options] [fichier ...]
```

Exemple :

```
[root]# gzip usr.tar
[root]# ls
usr.tar.gz
```

Le fichier reçoit l'extension .gz.

Il conserve les mêmes droits et les mêmes dates de dernier accès et de modification.

Compresser avec bunzip2

La commande bunzip2 compresse également les données.

Syntaxe de la commande bzip2

```
bzip2 [options] [fichier ...]
```

Exemple :

```
[root]# bzip2 usr.cpio
[root]# ls
usr.cpio.bz2
```

Le nom du fichier reçoit l'extension .bz2.

La compression par “bzip2” est meilleure que celle par “gzip” mais dure plus longtemps.

Décompresser avec gunzip

La commande gunzip décompresse les données compressées.

Syntaxe de la commande gunzip

```
gunzip [options] [fichier ...]
```

Exemple :

```
[root]# gunzip usr.tar.gz
[root]# ls
usr.tar
```

Le nom du fichier est tronqué par gunzip et se voit enlever l’extension .gz .

Gunzip décompresse également les fichiers portant les extensions suivantes :

- .z ;
- -z ;
- _z.

Décompresser avec bunzip2

La commande bunzip2 décompresse les données compressées.

Syntaxe de la commande bzip2

```
bzip2 [options] [fichier ...]
```

Exemple :

```
[root]# bunzip2 usr.cpio.bz2
[root]# ls
usr.cpio
```

Le nom du fichier est tronqué par bunzip2 et se voit enlever l’extension .bz2 .

bunzip2 décompresse également le fichier portant les extensions suivantes :

- -bz ;
- .tbz2 ;

- tbz.

Chapitre 9. Démarrage du système

9.1. Démarrage de l'ordinateur

Séquence de démarrage

La séquence de démarrage du système est variable en fonction du système mais peut d'une manière générale être découpée selon les étapes définies ci-dessous.

- démarrage de l'ordinateur ou amorçage ;
- exécution du chargeur de démarrage ;
- démarrage du noyau ;
- lancement du processus **init** ;
- lancement des scripts de démarrage.

Amorçage matériel

Après la mise sous tension, un programme stocké en mémoire morte sur la carte mère prend le contrôle. Sur les PC, ce programme est appelé le BIOS ou UEFI pour les dernières générations de carte mère. Pour la suite du cours nous ne verrons que le BIOS.

- **BIOS** : Basic Input Output System ;
- **UEFI** : Unified Extensible Firmware Interface ;
- **POST** : Power On Self Test.

Séquence d'amorçage matériel :

- mise sous tension de l'ordinateur ;
- lecture **BIOS/UEFI** stocké en mémoire morte ;
- BIOS effectue un autotest : **POST** ;
- lecture des paramètres (périphériques de boot, ...) ;
- lancement du chargeur de démarrage trouvé.

Le chargeur de démarrage

Le chargeur de démarrage **localise** le **noyau du système d'exploitation** sur le disque, le **charge** et l'**exécute**.

La majorité des chargeurs sont interactifs, ils permettent :

- de **spécifier un noyau** ;
- de positionner des **paramètres optionnels**.

Spécification d'un noyau alternatif : un noyau de sauvegarde dans le cas où la dernière version compilée ne fonctionne pas.

Types de chargeurs

On peut retrouver généralement sous Linux :

- **Grub** : GRand Unified Bootloader (le plus répandu) ;
- **Lilo** : LInux LOader (délaissé par les développeurs) ;
- **Elilo** : pour UEFI.

Pour la suite du cours nous utiliserons GRUB.

9.2. Le chargeur GRUB

Le chargeur GRUB est un programme de **multiboot** permettant de choisir entre plusieurs systèmes d'exploitation lors du démarrage.

La technique du **chain-loading** lui permet de démarrer une grande variété de systèmes d'exploitation. Il peut donc aussi bien charger des systèmes compatibles avec le multiboot que des systèmes non compatibles.

La configuration du GRUB est lue au démarrage du système.

Le **Chain-loading** est une technique qui permet à un chargeur de lancer un autre chargeur en l'encapsulant.

GRUB reconnaît nativement divers systèmes de fichiers.

Il fournit un interpréteur de commandes permettant le chargement manuel d'un système d'exploitation et le changement de la configuration au boot.

GRUB permet d'agir en interactif sur le démarrage.

GRUB peut être utilisé avec différentes interfaces. Beaucoup de distributions GNU/Linux utilisent le support graphique de GRUB pour afficher au démarrage de l'ordinateur un menu avec une image de fond et parfois un support de la souris.

GRUB peut télécharger des images de systèmes d'exploitations depuis un réseau et supporte donc les ordinateurs sans disques. Il peut donc décompresser ces images pour les charger ensuite.

Choix du système

GRUB fournit une interface avec menu à partir de laquelle vous pouvez choisir un système d'exploitation qui sera ensuite amorcé.

Ce menu est basé sur le fichier de configuration **grub.conf** qui se trouve dans le répertoire **/boot/grub/**.

En fonction des distributions, **/etc/grub.conf** peut être un lien symbolique vers ce fichier.

Lors des sélections dans le GRUB, le clavier est en **qwerty**.

Table 52. Les touches de sélection du menu de Grub

Touche	Action
[ENTREE]	Démarre le système sélectionné.
[e]	Édite la configuration du système.
[a]	Permet de modifier les arguments.
[c]	Permet d'utiliser l'interface Shell de GRUB.

La touche **[ENTREE]** permet de lancer l'initialisation du serveur.

Une fois le démarrage terminé, le système affiche les messages placés dans le fichier **/etc/issue** et propose de saisir un login puis un mot de passe pour se connecter.

La touche **[e]** permet d'éditer la configuration avant de démarrer. Il est alors possible de modifier les lignes en les éditant une à une avec la touche **[e]**.

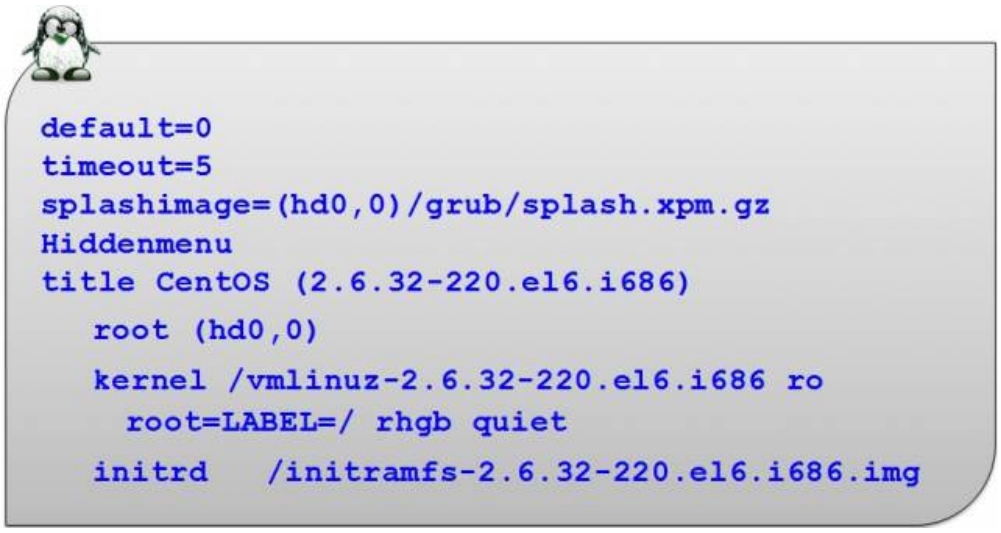
Après avoir édité la ligne sélectionnée, procéder à la modification puis valider la ligne modifiée en utilisant la touche **[ENTREE]**.

Pour initialiser le système en tenant compte de ces modifications, utiliser la touche **[b]** (comme 'boot').

La touche **[a]** / **[q]** permet de modifier les arguments du noyau.

La touche **[c]** permet d' obtenir l'interface Shell de Grub.

Fichier /boot/grub/grub.conf



La première partie est constituée de lignes de commentaires décrivant la structure du fichier.

Aucune compilation de ce fichier de configuration ne sera nécessaire.

Explications des options :

Table 53. Variables du fichier /boot/grub/grub.conf

Variable	**Observation**
default=0	Correspond au système d'exploitation lancé par défaut. La première rubrique **title** porte le numéro 0.
timeout=5	GRUB amorcera automatiquement le système par défaut au bout de 5 secondes, à moins d'être interrompu.
splashimage	Déclaration de l'image qui s'affiche avec le chargeur Grub. Il faut indiquer l'emplacement de cette image. Les systèmes de fichiers n'étant pas encore montés, indiquer le disque et la partition de /boot **(hd0,0)**, le chemin jusqu'à l'image **grub/** et enfin le nom de votre image **splash.xpm.gz**.
hiddenmenu	Sert à masquer le menu Grub et après le délai du timeout, le système se lancera automatiquement en fonction de l'option **default**.
title	Il s'agit en fait du nom qui apparaîtra dans le menu Grub (exemple "ma distrib Linux préférée"). En règle général, le nom du système est choisi : exemple Fedora, Suse, Ubuntu, Vista, Xp, Frugal, etc,... et éventuellement la version du noyau. Un seul nom par rubrique **title**, il faut donc déclarer autant de lignes 'title' qu'il y a d'options de démarrage ou de systèmes installés.
root	Indique le disque puis la partition (hdx,y) où se trouvent les fichiers permettant l'initialisation du système (exemple : (hd0,0), correspondant à la partition /boot) pour ce "title".

Variable	Observation
kernel	Indique le nom du noyau à charger, son emplacement et les options utiles à son démarrage pour ce "title".
initrd	initrd (INITial RamDisk) charge un ramdisk initial pour une image de démarrage au format Linux et définit les paramètres adéquats dans la zone de configuration de Linux en mémoire pour ce "title".

9.3. Sécuriser GRUB

GRUB est par défaut très permissif. Certaines opérations interactives ne nécessitent pas d'authentification. Ainsi, il est possible d'exécuter des commandes root sans s'être authentifié !

L'accès au menu interactif doit être protégé.

Définir un mot de passe

Il faut définir un mot de passe avec la directive **password** dans le fichier **/boot/grub/grub.conf**.

La directive **password** est ajoutée dans la partie globale avant la directive **title** .

Quand la directive **password** est saisie dans **/boot/grub/grub.conf**, GRUB interdit tout contrôle interactif, jusqu'à ce que la touche **[p]** soit pressée et qu'un mot de passe correct soit entré.

Chiffrer le mot de passe

Le mot de passe peut être chiffré.

À l'aide d'un shell traditionnel :

```
[root]# grub-crypt >> /boot/grub/grub.conf
```

Dans le fichier **grub.conf** l'option **[--encrypted]** devra être ajoutée entre la directive **password** et le mot de passe chiffré.

Vous pouvez chiffrer votre mot de passe avec la commande **grub-crypt**.

Copiez le mot de passe chiffré dans votre fichier de configuration et indiquez dans la rubrique qu'il est chiffré.

```
[root]#  grub-crypt
Password : mdpauser1
Retype password : mdpauser1
$6$uK6Bc/$90LR/0QW.14G4473EaENd
```

Le hash du mot de passe fourni par la commande grub-crypt commence par un 6 car l'algorithme utilisé est le SHA-512.

Table 54. Liste des algorithme de hachage des mots de passe

Préfixe	Algorithme utilisé
	DES
1	MD5
2, $2a$, $2x$, $2y$	bcrypt
3	NTHASH
5	SHA-256
6	SHA-512

Dans le but de récupérer ce mot de passe et de l'insérer directement dans le fichier **grub.conf**, utilisez la commande suivante :

```
[root]# grub-crypt >>/boot/grub/grub.conf
```

Menu interactif verrouillé

Pour agir sur l'interactivité du démarrage, utiliser la touche **[p]** pour saisir le mot de passe et ensuite disposer des options permettant d'agir sur le lancement du noyau.

Lancement verrouillé

Le lancement d'un système peut être verrouillé avec la directive **lock** positionnée en dessous de la ligne **title** à verrouiller.

- Le mot de passe sera systématiquement demandé.
- Pourquoi verrouiller le lancement d'un système ? L'administrateur peut prévoir une rubrique qui lancera son système en mode **single**. Aucun utilisateur ne devra donc utiliser cette rubrique.
- Il est possible d'affecter un mot de passe différent pour chaque menu. Pour cela il suffira de remettre une directive **password** avec un nouveau mot de passe dans chaque rubrique **title**.

9.4. Démarrage du noyau

Au démarrage du système, GRUB apparaît.

[ENTREE] active la configuration par défaut.

Une autre touche fait apparaître le menu du GRUB.

Niveaux de démarrage

Table 55. Les 6 niveaux de démarrage

s ou single	Le processus **init** démarre le système en mode mono-utilisateur. Par défaut l'utilisateur est connecté en tant que **root** sans fournir de mot de passe.
1 - 5	Le processus **init** démarre le système avec le niveau demandé.

Étapes du démarrage

Principales étapes du démarrage :

- chargement du noyau (processus 0) ;
- installation des périphériques via leur pilote ;
- démarrage du gestionnaire de swap ;
- montage du système de fichiers racine ;
- création par le noyau du premier processus qui porte le numéro 1 ;
- ce processus exécute le programme **/sbin/init** en lui passant les paramètres qui ne sont pas déjà

gérés par le noyau.

9.5. Le processus init (généralités)

Les différents niveaux d'exécution

Table 56. Les 8 niveaux d'exécution (détaillés)

0	Arrête le système.
1	Mode mono-utilisateur (console).
2	Mode multi-utilisateurs. Les systèmes de fichiers sont montés. Le service réseau est démarré.
3	Sur-ensemble du niveau 2. Il est associé au démarrage des services de partage à distance.
4	Mode multi-utilisateurs spécifique au site informatique.
5	Sur-ensemble du niveau 3. Interface X-Window (graphique).
6	Redémarre le système.
s, S, single	Mode mono-utilisateur (single). Les systèmes de fichiers sont montés. Seuls les processus fondamentaux pour le bon fonctionnement du système sont activés. Un shell en mode **root** est activé sur une console. Le répertoire **/etc** n'est pas indispensable.

Il n'y a qu'un niveau d'exécution actif à la fois.

La commande init

La commande init permet de changer le niveau d'exécution courant .

Syntaxe de la commande init

```
init [-options] [0123456Ss]
```

Exemple :

```
[root]# init 5
```

La commande runlevel

La commande runlevel permet de connaître le niveau d'exécution courant.

Syntaxe de la commande runlevel

```
runlevel
```

Exemple :

```
[root]# runlevel
N 3
```

Dans cet exemple, le système se trouve au niveau d'exécution 3 - Multiuser.

Le N indique que le niveau d'exécution précédent était le démarrage du système. Après un init 5, le résultat de la commande serait alors :

```
[root]# runlevel
3 5
```

Il n'y a qu'un niveau d'exécution actif à la fois.

Le fichier /etc/inittab

Lors du démarrage, à la création du processus **init**, le niveau est celui défini dans **GRUB** ou sinon celui dans **/etc/inittab**.

Le niveau défini dans **GRUB** est prioritaire à celui défini dans **inittab**.

Aperçu du fichier **/etc/inittab**

```
# For information on how to write upstart event handlers, or how
# upstart works, see init(5), init(8), and initctl(8).
# Default runlevel. The runlevels used are:
# 0 - halt (Do NOT set initdefault to this)
# 1 - Single user mode
# 2 - Multiuser, without NFS (The same as 3, if you do not have networking)
# 3 - Full multiuser mode
# 4 - unused
# 5 - X11
# 6 - reboot (Do NOT set initdefault to this)
id:5:initdefault:
```

Changement de niveau

Lorsqu'un changement de niveau est effectué, le processus **init** envoie un signal **SIGTERM (15)** à tous les processus non concernés par le nouveau niveau.

Un délai de 5 secondes est accordé afin que les processus se terminent correctement. Après ce délai, le processus **init** envoie un deuxième signal **SIGKILL (9)** à tous les processus non terminés.

init démarre ensuite les processus concernés par le nouveau niveau d'exécution.

Activer ou désactiver les terminaux

Pour activer ou désactiver les terminaux, il faut modifier la variable **ACTIVE_CONSOLES** dans le fichier **/etc/sysconfig/init**.

Exemples :

```
ACTIVE_CONSOLES="/dev/tty[1-6]" #Active les terminaux de 1 à 6

ACTIVE_CONSOLES="/dev/tty[1-6] /dev/tty8 /dev/tty9" #Active les terminaux de 1 à 6, le
8 et le 9
```

Ne pas oublier les " ".

Le système doit être redémarré pour la prise en compte.

Au niveau de démarrage 5, le système ne prend pas en compte le fichier /etc/sysconfig/init.

En cas d'erreur de manipulation dans ce fichier, un moyen de redémarrer un serveur est de force le redémarrage en init 5 !

Autoriser à root l'accès aux terminaux

Par défaut, une console déclarée dans **/etc/sysconfig/init** n'est accessible que par les utilisateurs.

Il faut renseigner le fichier **/etc/securetty** en ajoutant le nom de ce nouveau terminal pour autoriser root à s'y connecter.

Exemple suivant, à la dernière ligne (tty6) le # est retiré permettant l'accès à root sur ce terminal.

```
console
#vc/1
#vc/2
#vc/3
#vc/4
#vc/5
#vc/6
#tty1
#tty2
#tty3
#tty4
#tty5
tty6
```

9.6. Le processus init (démon)

Démarrage des démons

init lance le script **/etc/rc.d/rc.sysinit** quelque soit le niveau.

Init exécute ensuite le script **/etc/rc.d/rc** en lui passant en paramètre le niveau d'exécution demandé

Script de démarrage des services

Pour chaque service, il y a un script de démarrage stocké dans **/etc/rc.d/init.d**.

Chaque script accepte au minimum en argument :

- stop : pour arrêter le service ;
- start : pour démarrer le service ;
- restart : pour redémarrer le service ;
- status : pour connaître l'état du service.

Répertoires d'ordonnancement

Pour chaque niveau d'exécution, il existe un répertoire correspondant : **/etc/rc.d/rc[0-6].d/**.

Ces répertoires contiennent les liens symboliques vers les scripts placés dans **/etc/rc.d/init.d**.

L'avantage du lien : il n'existe qu'un seul exemplaire du script du service.

Nom des liens

Mise en route (**S**tart) : **S**XXnom

Arrêt (**K**ill) : **K**YYnom

XX et **YY** : nombre de 00 à 99 qui guide l'ordre d'exécution (Start ou Kill).

nom : nom exact du service à démarrer ou à arrêter tel qu'écrit dans /etc/rc.d/init.d/.

La somme des nombres est un complément à 100 : XX + YY = 100.

Cette méthode permet d'ordonnancer le démarrage et l'arrêt des services. Un service qui est démarré en premier doit être le dernier à s'arrêter. La liste étant lue dans l'ordre alphabétique.

Exemple :

Dans /etc/rc.d/rc3.d/, nous avons :

- K15httpd
- S10network
- S26acpid

Donc, pour le niveau d'exécution 3 (rc3.d) :

- le service httpd doit être arrêté (lettre K),
- les services network et acpid doivent être lancés (lettre S) dans cet ordre (numéro du service network 10 plus petit que celui de acpid 26)

Le programme /etc/rc.d/rc

Ce programme est lancé par **init** avec le niveau d'exécution en paramètre.

Il comporte deux boucles.

Init lance le script **rc** avec le niveau **X** en paramètre.

Première boucle : lecture des scripts d'arrêt **K...** présents dans **rcX.d**.

Deuxième boucle : lecture des scripts de démarrage **S...** présents dans **rcX.d**.

Architecture de démarrage

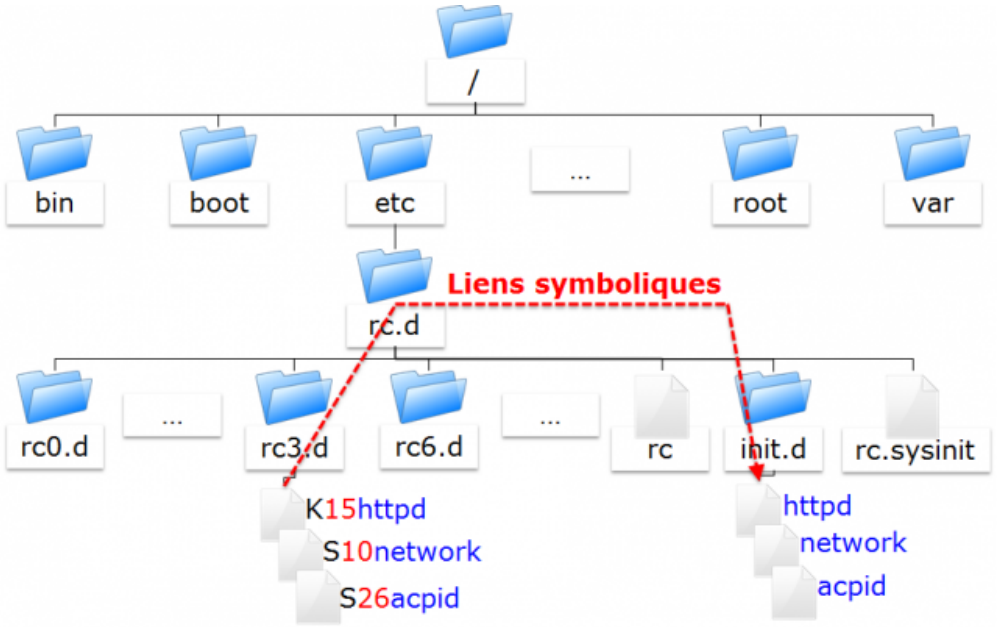


Figure 29. Synthèse de l'arborescence système liée au démarrage

9.7. La gestion des services

Comme il n'existe qu'un seul exemplaire du fichier script par service sous **/etc/rc.d/init.d**, leur gestion est facilitée. La gestion des liens s'effectue soit :

- manuellement avec la commande ln ;
- avec la commande de gestion des services chkconfig.

La commande ln

Créer un lien symbolique manuellement.

Syntaxe de la commande ln

```
ln -s source destination
```

Exemple :

```
[root]# cd /etc/rc.d/rc2.d
[root]# ln –s ../init.d/numlock S85numlock
```

Il faut alors créer tous les liens (**K** ou **S**) pour **chaque niveau de démarrage**.

La commande chkconfig

La commande chkconfig permet de gérer un service.

Il faut les deux lignes suivantes au début de chaque script.

```
# chkconfig: [niveau_exécution] [num_start] [num_kill]
# description: [descriptif du script]
```

Exemple 1 :

```
# chkconfig: 2345 10 90
# description: Commentaires libres
```

Exemple 2 :

```
# chkconfig: -
# description: Commentaires libres
```

Le - après **chkconfig:** signifie que le service ne doit jamais être démarré.

Gérer et visualiser l'état d'un service

Syntaxe de la commande chkconfig

```
chkconfig [--options] [service]
chkconfig --level service on|off|reset
```

Exemple :

```
[root]# chkconfig --list network
network 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt
```

Table 57. Options de la commande chkconfig

Option longue	Description
--list	Visualise l'état des services (--list seul liste tous les services créés).
--add	Crée des liens symboliques.
--del	Supprime des liens symboliques.
--level	Modifie les liens symboliques.

La commande **chkconfig --list** lit les niveaux définis dans l'en-tête du service et affiche la configuration de démarrage du service.

Cette commande ne donne pas un état actuel du service.

Arrêt ne signifie pas que le service est arrêté, mais qu'il ne sera pas démarré au niveau spécifié.

Un autre moyen de visualiser les liens symboliques en une seule commande :

```
[root]# ls -l /etc/rc.d/rc*.d/*
```

Créer les liens symboliques

```
chkconfig --add service
```

Exemple :

```
[root]# chkconfig --add network
[root]# chkconfig --list network
network 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt
```

chkconfig --add lit les niveaux définis dans l'en-tête du service et crée les liens correspondants.

Exemple :

Fichier /etc/rc.d/init.d/nomduservice

```
# chkconfig: 235 90 10
```

chkconfig crée les liens **S90...** dans les répertoires définis /etc/rc.d/rc2.d, rc3.d et rc5.d et les liens **K10...** dans les répertoires restants /etc/rc.d/rc0.d, rc1.d, rc4.d et rc6.d

Afin d'éviter les incohérences, faire un **chkconfig --del nomduservice** avant le **chkconfig --add**.

Supprimer des liens symboliques

```
chkconfig --del nomduservice
```

Exemple :

```
[root]# chkconfig --del network
[root]# chkconfig --list network
network 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt
```

Modifier des liens symboliques

Syntaxe de la commande chkconfig

```
chkconfig [--level niveaux] service <on | off>
```

Exemple :

```
[root]# chkconfig —-level 235 atd on
[root]# chkconfig —-level 0146 atd off
```

--level	Spécification du niveau auquel est créé le lien symbolique.
on	Le lien permettant de lancer le service est créé (Sxx...).
off	Le lien permettant d'arrêter le service est créé (Kxx...).

Démarrage d'un service

Manuellement :

- Avec un **script** de lancement :

```
/chemin/script  [status|start|stop|restart]
```

Exemple :

```
[root]#/etc/rc.d/init.d/crond start
Démarrage de crond :        [ OK ]
[root]#/etc/rc.d/init.d/crond status
crond (pid 4731) en cours d'exécution
```

- Avec la commande **service** :

Syntaxe de la commande service

```
service script  [status|start|stop|restart]
```

Exemple :

```
[root]# service  crond  start
Démarrage de crond :        [ OK ]
[root]# service  crond  status
crond (pid 4731) en cours d'exécution
```

La commande **service** prend en compte tous les scripts placés dans **/etc/rc.d/init.d**.

La commande **service** n'existe que dans le monde des distributions RHEL !

9.8. Arrêt du système

Les opérations de maintenance, diagnostics, modifications de logiciels, ajouts et retraits de matériel, taches administratives, coupures électriques, ... nécessitent parfois l'arrêt du système.

Cet arrêt peut être planifié, périodique ou impromptu et demander une réactivité immédiate.

Tous les systèmes Unix, y compris ceux fonctionnant sur PC doivent être mis hors service en utilisant les commandes décrites dans cette section.

Ceci garantit **l'intégrité** du disque et la **terminaison propre** des différents services du système.

Arrêt programmé du système :

- utilisateurs prévenus de l'arrêt ;
- applications arrêtées proprement ;
- intégrité des systèmes de fichiers assurée ;
- sessions utilisateurs stoppées.

En fonction des options, le système :

- passe en mode mono-utilisateur ;
- est arrêté ;
- est redémarré.

Commandes de mise hors service :

- init ;
- shutdown ;
- halt ;
- reboot.

Commande shutdown

La commande shutdown éteint le système.

Syntaxe de la commande shutdown

```
shutdown [-t sec] [options] heure [message-avertissement]
```

Exemples :

```
[root]# shutdown –r +2 "arrêt puis reboot dans 2 minutes"
[root]# shutdown –r 10:30 "arrêt puis reboot à 10h30"
[root]# shutdown –h now "arrêt électrique"
```

Table 58. Options de la commande shutdown

Options	Commentaires
-t sec	Attendre **sec** entre le message d'avertissement et le signal de fin aux processus
-r	Redémarrer la machine après l'arrêt du système
-h	Arrêter la machine après l'arrêt du système
-P	Éteindre l'alimentation
-f	Ne pas effectuer de fsck en cas de redémarrage
-F	Forcer l'utilisation de fsck en cas de redémarrage
-c	Annuler un redémarrage en cours

heure : Quand effectuer le shutdown (soit une heure fixe **hh:mm**, soit un délai d'attente en minute **+mm**).

message-avertissement : Message à envoyer à tous les utilisateurs.

Commande halt

La commande halt provoque un arrêt immédiat du système.

```
[root]# halt
```

Cette commande appelle le processus **init**

halt ⇒ init 0

Commande reboot

La commande reboot provoque un redémarrage immédiat du système.

```
[root]# reboot
```

Cette commande appelle le processus **init**

reboot ⇒ init 6

Chapitre 10. Démarrage du système sous CentOS 7

10.1. Le processus de démarrage

Il est important de comprendre le processus de démarrage de Linux pour pouvoir résoudre les problèmes qui peuvent y survenir.

Le processus de démarrage comprend :

Le démarrage du BIOS

Le **BIOS** (Basic Input/Output System) effectue le test **POST** (power on self test) pour détecter, tester et initialiser les composants matériels du système.

Il charge ensuite le **MBR** (Master Boot Record).

Le Master boot record (MBR)

Le Master Boot Record correspond aux 512 premiers bytes du disque de démarrage. Le MBR découvre le périphérique de démarrage et charge le chargeur de démarrage **GRUB2** en mémoire et lui transfert le contrôle.

Les 64 bytes suivants contiennent la table de partition du disque.

Le chargeur de démarrage GRUB2 (Bootloader)

Le chargeur de démarrage par défaut de la distribution CentOS 7 est **GRUB2** (GRand Unified Bootloader). GRUB2 remplace l'ancien chargeur de démarrage Grub (appelé également GRUB legacy).

Le fichier de configuration de GRUB 2 se situe sous **/boot/grub2/grub.cfg** mais ce fichier ne doit pas être directement édité.

Les paramètres de configuration du menu de GRUB2 se trouvent sous **/etc/default/grub** et servent à la génération du fichier grub.cfg.

Exemple de fichier /etc/default/grub

```
# cat /etc/default/grub
GRUB_TIMEOUT=5
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="rd.lvm.lv=rhel/swap crashkernel=auto rd.lvm.lv=rhel/root rhgb
quiet net.ifnames=0"
GRUB_DISABLE_RECOVERY="true"
```

Si des changements sont effectués à un ou plusieurs de ces paramètres, il faut lancer la commande **grub2-mkconfig** pour régénérer le fichier /boot/grub2/grub.cfg.

```
[root] # grub2-mkconfig –o /boot/grub2/grub.cfg
```

- GRUB2 cherche l'image du noyau compressé (le fichier vmlinuz) dans le répertoire /boot.
- GRUB2 charge l'image du noyau en mémoire et extrait le contenu du fichier image initramfs dans un dossier temporaire en mémoire en utilisant le système de fichier tmpfs.

Le noyau

Le noyau démarre le processus **systemd** avec le PID 1.

```
root          1      0  0 02:10 ?        00:00:02 /usr/lib/systemd/systemd --switched
-root --system --deserialize 23
```

systemd

Systemd est le père de tous les processus du système. Il lit la cible du lien **/etc/systemd/system/default.target** (par exemple /usr/lib/systemd/system/multi-user.target) pour déterminer la cible par défaut du système. Le fichier défini les services à démarrer.

Systemd positionne ensuite le système dans l'état défini par la cible en effectuant les tâches d'initialisations suivantes :

1. Paramétrer le nom de machine
2. Initialiser le réseau
3. Initialiser SELinux
4. Afficher la bannière de bienvenue
5. Initialiser le matériel en se basant sur les arguments fournis au kernel lors du démarrage
6. Monter les systèmes de fichiers, en incluant les systèmes de fichiers virtuels comme /proc

7. Nettoyer les répertoires dans /var
8. Démarrer la mémoire virtuelle (swap)

10.2. Protéger le chargeur de démarrage GRUB2

Pourquoi protéger le chargeur de démarrage avec un mot de passe ?

1. Prévenir les accès au mode utilisateur **Single** – Si un attaquant peut démarrer en mode single user, il devient l'utilisateur root.
2. Prévenir les accès à la console GRUB – Si un attaquant parvient à utiliser la console GRUB, il peut changer sa configuration ou collecter des informations sur le système en utilisant la commande cat.
3. Prévenir les accès à des systèmes d'exploitation non sécurisés. S'il y a un double boot sur le système, un attaquant peut sélectionner au démarrage un système d'exploitation comme DOS qui ignore les contrôles d'accès et les permissions des fichiers.

Pour protéger par mot de passe le GRUB2 :

- Retirer **–unrestricted** depuis la déclaration principale **CLASS=** dans le fichier **/etc/grub.d/10_linux**.
- Si un utilisateur n'a pas encore été configuré, utiliser la commande **grub2-setpassword** pour fournir un mot de passe à l'utilisateur root :

```
# grub2-setpassword
```

Un fichier **/boot/grub2/user.cfg** va être créé s'il n'était pas encore présent. Il contient le mot de passe hashé du GRUB.

Cet commande ne supporte que les configurations avec un seul utilisateur root.

Exemple de fichier /boot/grub2/user.cfg

```
[root]# cat /boot/grub2/user.cfg
GRUB2_PASSWORD=grub.pbkdf2.sha512.10000.CC6F56....A21
```

- Recréer le fichier de configuration avec la commande **grub2-mkconfig** :

```
[root]# grub2-mkconfig -o /boot/grub2/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-3.10.0-327.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-327.el7.x86_64.img
Found linux image: /boot/vmlinuz-0-rescue-f9725b0c842348ce9e0bc81968cf7181
Found initrd image: /boot/initramfs-0-rescue-f9725b0c842348ce9e0bc81968cf7181.img
done
```

- Redémarrer le serveur et vérifier.

Toutes les entrées définies dans le menu du GRUB vont maintenant nécessiter la saisie d'un utilisateur et d'un mot de passe à chaque démarrage. Le système ne démarrera pas de noyau sans intervention directe de l'utilisateur depuis la console.

- Lorsque l'utilisateur est demandé, saisir "root" ;
- Lorsqu'un mot de passe est demandé, saisir le mot de passe fourni à la commande grub2-setpassword.

Pour ne protéger que l'édition des entrées du menu de GRUB et l'accès à la console, l'exécution de la commande grub2-setpassword est suffisante.

10.3. Systemd

Systemd est un gestionnaire de service pour les systèmes d'exploitation Linux.

Il est développé pour :

- rester compatible avec les anciens scripts d'initialisation SysV,
- fournir de nombreuses fonctionnalités comme le démarrage en parallèle des services systèmes au démarrage du système, l'activation à la demande de démons, le support des instantanés ou la gestion des dépendances entre les services.

Systemd est le système d'initialisation par défaut depuis la RedHat/CentOS 7.

Systemd introduit le concept d'unités systemd.

Table 59. Principaux types d'unités systemd disponibles

Type	Extension du fichier	Observation
Unité de service	.service	Service système
Unité cible	.target	Un groupe d'unités systemd
Unité automount	.automount	Un point de montage automatique pour système de fichiers

Il existe de nombreux types d'unités : Device unit, Mount unit, Path unit, Scope unit, Slice unit, Snapshot unit, Socket unit, Swap unit, Timer unit.

- Systemd supporte les instantanés de l'état du système et leur restauration.
- Les points de montage peuvent être configurés comme des cibles systemd.
- Au démarrage, systemd créé des sockets en écoute pour tous les services systèmes qui supportent ce type d'activation et passe ces sockets à ces services aussitôt qu'elles sont démarrées. Cela rend possible la relance d'un service sans perdre un seul message qui lui est envoyé par le réseau durant son indisponibilité. La socket correspondante reste accessible et tous les messages sont mis en file d'attente.
- Les services systèmes qui utilisent D-BUS pour leurs communications inter-process peuvent être démarrés à la demande dès la première utilisation par un client.
- Systemd stoppe ou relance uniquement les services en cours de fonctionnement. Les versions précédentes de CentOS tentaient directement de stopper les services sans vérifier leur état en cours.
- Les services systèmes n'héritent d'aucun contexte (comme les variables d'environnements HOME et PATH). Chaque service fonctionne dans son propre contexte d'exécution.

Toutes les opérations des unités de service sont soumises à un timeout par défaut de 5 minutes pour empêcher un service mal-fonctionnant de geler le système.

Gérer les services systèmes

Les unités de service se terminent par l'extension de fichier .service et ont un but similaire à celui des scripts init. La commande systemctl est utilisée pour afficher, lancer, arrêter, redémarrer, activer ou désactiver des services système.

Les commandes service et chkconfig sont toujours disponibles dans le système et fonctionnent comme prévu, mais sont uniquement incluses pour des raisons de compatibilité et doivent être évitées.

Table 60. Comparaison des utilitaires service et systemctl

service	**systemctl**	**Description**
service *name* start	systemctl start *name*.service	Lancer un service
service *name* stop	systemctl stop *name*.service	Stoppe un service
service *name* restart	systemctl restart *name*.service	Relance un service
service *name* reload	systemctl reload *name*.service	Recharge une configuration
service *name* status	systemctl status *name*.service	Vérifie si un service fonctionne
service *name* condrestart	systemctl try-restart *name*.service	Relance un service seulement s'il fonctionne

service	systemctl	Description
service --status-all	systemctl list-units --type service --all	Affiche le status de tous les services

Table 61. Comparaison des utilitaires chkconfig et systemctl

chkconfig	systemctl	Description
chkconfig *name* on	systemctl enable name.service	Active un service
chkconfig *name* off	systemctl disable name.service	Désactive un service
chkconfig --list *name*	systemctl status name.service	Vérifie si un service fonctionne
chkconfig --list	systemctl list-unit-files --type service	Liste tous les services et vérifie s'ils fonctionnent
chkconfig --list	systemctl list-dependencies --after	Liste les services qui démarrent avant l'unité spécifiée
chkconfig --list	systemctl list-dependencies --before	Liste les services qui démarrent après l'unité spécifiée

Exemples :

```
systemctl stop nfs-server.service
# ou
systemctl stop nfs-server
```

Pour lister toutes les unités chargées actuellement :

```
systemctl list-units --type service
```

Pour lister toutes les unités pour vérifier si elles sont activées :

```
systemctl list-unit-files --type service
```

```
systemctl enable httpd.service
systemctl disable bluetooth.service
```

Exemple de fichier .service pour le service postfix

```
postfix.service Unit File
What follows is the content of the /usr/lib/systemd/system/postfix.service unit file
as currently provided by the postfix package:

[Unit]
Description=Postfix Mail Transport Agent
After=syslog.target network.target
Conflicts=sendmail.service exim.service

[Service]
Type=forking
PIDFile=/var/spool/postfix/pid/master.pid
EnvironmentFile=-/etc/sysconfig/network
ExecStartPre=-/usr/libexec/postfix/aliasesdb
ExecStartPre=-/usr/libexec/postfix/chroot-update
ExecStart=/usr/sbin/postfix start
ExecReload=/usr/sbin/postfix reload
ExecStop=/usr/sbin/postfix stop

[Install]
WantedBy=multi-user.target
```

Utiliser les targets systèmes

Sur CentOS7/RHEL7, le concept des niveaux d'exécution a été remplacé par les cibles Systemd.

Les cibles Systemd sont représentées par des unités de cible (target units). Les unités de cible se terminent par l'extension de fichier .target et leur unique but consiste à regrouper d'autres unités Systemd dans une chaîne de dépendances.

Par exemple, l'unité **graphical.target**, qui est utilisée pour lancer une session graphique, lance des services systèmes comme le gestionnaire d'affichage GNOME (gdm.service) ou le services des comptes (accounts-daemon.service) et active également l'unité multi-user.target.

De manière similaire, l'unité multi-user.target lance d'autres services système essentiels, tels que NetworkManager (NetworkManager.service) ou D-Bus (dbus.service) et active une autre unité cible nommée basic.target.

Table 62. Comparaison des targets systemctl et runlevel

Runlevel	Target Units	Description
0	poweroff.target	Arrête le système et l'éteint
1	rescue.target	Active un shell de secours
2	multi-user.target	Active un système multi-utilisateur sans interface graphique

Runlevel	Target Units	Description
3	multi-user.target	Active un système multi-utilisateur sans interface graphique
4	multi-user.target	Active un système multi-utilisateur sans interface graphique
5	graphical.target	Active un système multi-utilisateur avec interface graphique
6	reboot.target	Arrête puis redémarre le système

La cible par défaut

Pour déterminer quelle cible est utilisée par défaut :

```
systemctl get-default
```

Cette commande recherche la cible du lien symbolique située à /etc/systemd/system/default.target et affiche le résultat.

```
$ systemctl get-default
graphical.target
```

La commande systemctl peut également fournir la liste des cibles disponibles :

Lister toutes les cibles disponibles

```
sudo systemctl list-units --type target
UNIT                   LOAD   ACTIVE SUB    DESCRIPTION
basic.target           loaded active active Basic System
bluetooth.target       loaded active active Bluetooth
cryptsetup.target      loaded active active Encrypted Volumes
getty.target           loaded active active Login Prompts
graphical.target       loaded active active Graphical Interface
local-fs-pre.target    loaded active active Local File Systems (Pre)
local-fs.target        loaded active active Local File Systems
multi-user.target      loaded active active Multi-User System
network-online.target  loaded active active Network is Online
network.target         loaded active active Network
nss-user-lookup.target loaded active active User and Group Name Lookups
paths.target           loaded active active Paths
remote-fs.target       loaded active active Remote File Systems
slices.target          loaded active active Slices
sockets.target         loaded active active Sockets
sound.target           loaded active active Sound Card
swap.target            loaded active active Swap
sysinit.target         loaded active active System Initialization
timers.target          loaded active active Timers
```

Pour configurer le système afin d'utiliser une cible différente par défaut :

```
systemctl set-default name.target
```

Exemple :

```
[root]# systemctl set-default multi-user.target
rm '/etc/systemd/system/default.target'
ln -s '/usr/lib/systemd/system/multi-user.target' '/etc/systemd/system/default.target'
```

Pour passer à une unité de cible différente dans la session actuelle :

```
systemctl isolate name.target
```

Le **mode de secours** ("Rescue mode") fournit un environnement simple et permet de réparer votre système dans les cas où il est impossible d'effectuer un processus de démarrage normal.

En mode de secours, le système tente de monter tous les systèmes de fichiers locaux et de lancer plusieurs services système importants, mais n'active pas d'interface réseau ou ne permet pas à d'autres d'utilisateurs de se connecter au système au même moment.

Sur CentOS7/RHEL7, le mode de secours est équivalent au mode utilisateur seul (single user mode) et requiert le mot de passe root.

Pour modifier la cible actuelle et entrer en mode de secours dans la session actuelle :

```
systemctl rescue
```

Le **mode d'urgence** ("Emergency mode") fournit l'environnement le plus minimaliste possible et permet de réparer le système même dans des situations où le système est incapable d'entrer en mode de secours. Dans le mode d'urgence, le système monte le système de fichiers root uniquement en lecture, il ne tentera pas de monter d'autre système de fichiers locaux, n'activera pas d'interface réseau et lancera quelques services essentiels.

Pour modifier la cible actuelle et entrer en mode d'urgence dans la session actuelle :

```
systemctl emergency
```

Arrêt, suspension et hibernation

La commande systemctl remplace un certain nombre de commandes de gestion de l'alimentation utilisées dans des versions précédentes :

Table 63. Comparaison entre les commandes de gestion de l'alimentation et systemctl

Ancienne commande	Nouvelle commande	Description
halt	systemctl halt	Arrête le système.
poweroff	systemctl poweroff	Met le système hors-tension.
reboot	systemctl reboot	Redémarre le système.
pm-suspend	systemctl suspend	Suspend le système.
pm-hibernate	systemctl hibernate	Met le système en hibernation.
pm-suspend-hybrid	systemctl hybrid-sleep	Met en hibernation et suspend le système.

Le processus journald

Les fichiers journaux peuvent, en plus de rsyslogd, également être gérés par le démon **journald** qui est un composant de systemd.

Le démon journald capture les messages Syslog, les messages du journal du noyau, les messages du disque RAM initial et du début du démarrage, ainsi que les messages inscrits sur la sortie standard et la sortie d'erreur standard de tous les services, puis il les indexe et les rend disponibles à l'utilisateur.

Le format du fichier journal natif, qui est un fichier binaire structuré et indexé, améliore les

recherches et permet une opération plus rapide, celui-ci stocke également des informations de métadonnées, comme l'horodatage ou les ID d'utilisateurs.

La commande journalctl

La commande **journalctl** permet d'afficher les fichiers journaux.

```
journalctl
```

La commande liste tous les fichiers journaux générés sur le système. La structure de cette sortie est similaire à celle utilisée dans /var/log/messages/ mais elle offre quelques améliorations :

- la priorité des entrées est marquée visuellement ;
- les horodatages sont convertis au fuseau horaire local de votre système ;
- toutes les données journalisées sont affichées, y compris les journaux rotatifs ;
- le début d'un démarrage est marqué d'une ligne spéciale.

Utiliser l'affichage continu

Avec l'affichage continu, les messages journaux sont affichés en temps réel.

```
journalctl -f
```

Cette commande retourne une liste des dix lignes de journal les plus récentes. L'utilitaire journalctl continue ensuite de s'exécuter et attend que de nouveaux changements se produisent pour les afficher immédiatement.

Filtrer les messages

Il est possible d'utiliser différentes méthodes de filtrage pour extraire des informations qui correspondent aux différents besoins. Les messages journaux sont souvent utilisés pour suivre des comportements erronés sur le système. Pour afficher les entrées avec une priorité sélectionnée ou plus élevée :

```
journalctl -p priority
```

Il faut remplacer priority par l'un des mots-clés suivants (ou par un chiffre) :

- debug (7),
- info (6),
- notice (5),
- warning (4),

- err (3),
- crit (2),
- alert (1),
- et emerg (0).

Chapitre 11. Gestion des tâches

11.1. Généralités

La planification des tâches est gérée avec l'utilitaire **cron**. Il permet l'exécution périodique des tâches.

Il est réservé à l'administrateur et sous réserve aux utilisateurs et n'utilise qu'une commande : **crontab**.

Le service **cron** sert notamment pour :

- Les opérations d'administration répétitives ;
- Les sauvegardes ;
- La surveillance de l'activité du système ;
- L'exécution de programme.

crontab est le diminutif de **chrono table** : table de planification.

Pour mettre en place une planification, il faut que le système soit à l'heure.

11.2. Fonctionnement du service

Le fonctionnement du service **cron** est assuré par un démon **crond** présent en mémoire.

Pour vérifier son statut :

```
[root]# service crond status
```

Si le démon **crond** n'est pas en cours de fonctionnement, il faudra l'initialiser manuellement et/ou automatiquement au démarrage. En effet, même si des taches sont planifiées, elles ne seront pas lancées.

Initialisation du démon **crond** en manuel :

Depuis l'arborescence /etc/rc.d/init.d :

```
[root]# ./crond {status|start|restart|stop}
```

Avec la commande service :

```
[root]# service crond {status|start|restart|stop}
```

Initialisation du démon **crond** au démarrage :

Lors du chargement du système, il est lancé dans les niveaux d'exécution **2** à **5**.

```
[root]# chkconfig --list crond
crond 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt
```

11.3. La sécurité

Afin de mettre en oeuvre une planification, un utilisateur doit avoir la permission de se servir du service **cron**.

Cette permission varie suivant les informations contenues dans les fichiers ci-dessous :

- **/etc/cron.allow**
- **/etc/cron.deny**

Si aucun des deux fichiers n'est présent, tous les utilisateurs peuvent utiliser **cron**.

Autorisations

/etc/cron.allow

Seuls les utilisateurs contenus dans ce fichier sont autorisés à utiliser **cron**.

S'il est vide, aucun utilisateur ne peut utiliser **cron**.

Si cron.allow est présent, cron.deny est **ignoré**.

/etc/cron.deny

Les utilisateurs contenus dans ce fichier ne sont pas autorisés à utiliser **cron**.

S'il est vide, tous les utilisateurs peuvent utiliser **cron**.

Autoriser un utilisateur

Seul **user1** pourra utiliser **cron**

```
[root]# vi /etc/cron.allow
user1
```

Interdire un utilisateur

Seul **user2** ne pourra pas utiliser **cron**

```
[root]# vi /etc/cron.deny
user2
```

cron.allow ne doit pas être présent.

11.4. La planification des tâches

Lorsqu'un utilisateur planifie une tâche, un fichier portant son nom est créé sous **/var/spool/cron/**.

Ce fichier contient toutes les informations permettant au démon **crond** de savoir quelle commande ou quel programme lancer et à quel moment le faire (heure, minute, jour ...).

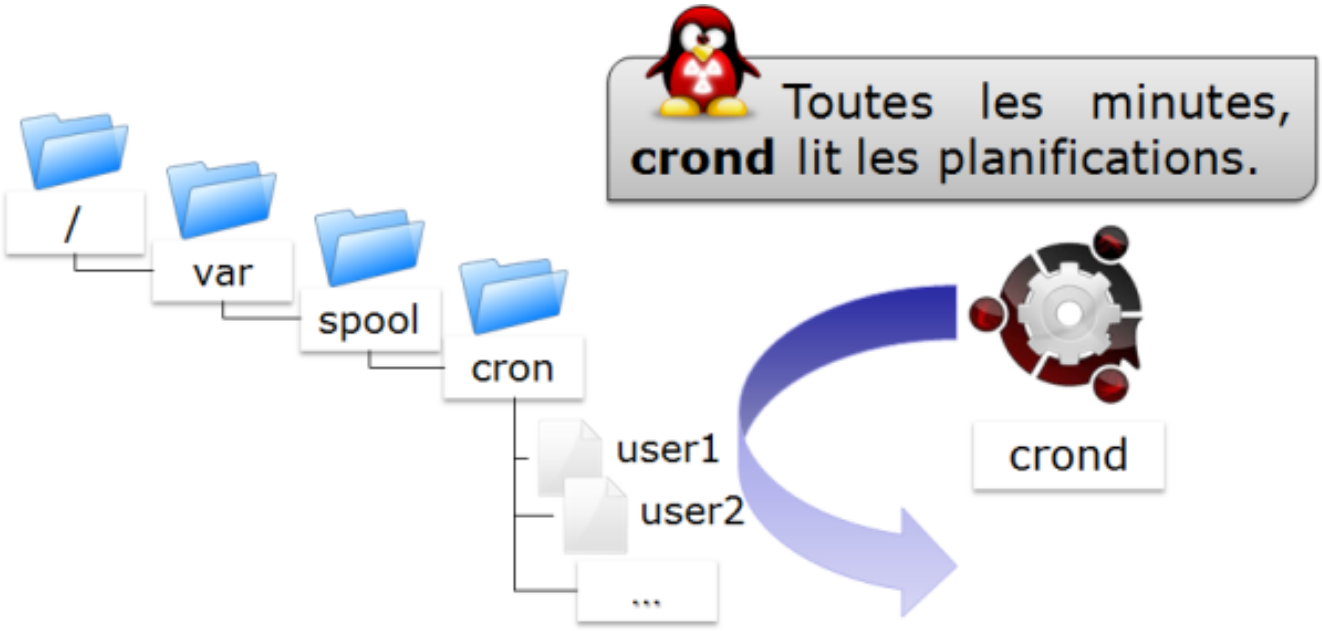


Figure 30. Arborescence de cron

Commande crontab

La commande **crontab** permet de gérer le fichier de planification.

```
crontab [-u utilisateur] [-e | -l | -r]
```

Exemple :

```
[root]# crontab –u user1 –e
```

Table 64. Options de la commande crontab

Option	Description
-e	Edite le fichier de planification avec vi
-l	Affiche le contenu du fichier de planification
-u	Nom de l'utilisateur dont le ficher de planification doit être manipulé
-r	Efface le fichier de planification

crontab sans option efface l'ancien fichier de planification et attend que l'utilisateur rentre de nouvelles lignes. Il faut taper [ctrl] + [d] pour quitter ce mode d'édition.

Seul **root** peut utiliser l'option **-u utilisateur** pour gérer le fichier de planification d'un autre utilisateur.

L'exemple proposé ci-dessus permet à root de planifier une tâche pour l'utilisateur user1.

Intérêts de la planification

Les intérêts de la planification sont multiples et notamment :

- Modifications des fichiers de planification prises en compte immédiatement ;
- Redémarrage inutile.

En revanche, il faut faire attention aux points suivants :

- Le programme doit être autonome ;
- Prévoir des redirections (stdin, stdout, stderr) ;
- Il n'est pas pertinent de lancer des commandes faisant appel à des demandes d'entrée/sortie sur un terminal.

Il faut bien comprendre que le but de la planification est d'effectuer des tâches de façon automatique, donc sans avoir besoin d'une intervention externe.

11.5. Le fichier de planification

Le fichier de planification est structuré est respecte les règles suivantes.

- Chaque ligne de ce fichier correspond à une planification ;
- Chaque ligne comporte six champs, 5 pour le temps et 1 pour la commande ;
- Chaque champs est séparé par un espace ou une tabulation ;
- Chaque ligne se termine par un retour chariot ;

- Un # en début de ligne commente celle-ci.

```
[root]# crontab –e
10 4 1 * * /root/scripts/backup.sh
1  2 3 4 5       6
```

Table 65. Champs du fichier de planification

Champ	Description	Détail
1	Minute(s)	De 0 à 59
2	Heure(s)	De 0 à 23
3	Jour(s) du mois	De 1 à 31
4	Mois de l'année	De 1 à 12
5	Jour(s) de la semaine	De 0 à 7 (0=7=dimanche)
6	Tâche à exécuter	Commande complète ou script

Les tâches à exécuter doivent utiliser des chemins absolus et si possible utiliser des redirections.

Afin de simplifier la notation pour la définition du temps, il est conseillé d'utiliser les symboles spéciaux.

Table 66. Métacaractères utilisables

Métacaractère	Description
*	Toutes les valeurs possibles du champs
-	Indique un intervalle de valeurs
,	Indique une liste de valeurs
/	Définit un pas

Exemples :

Script exécuté le 15 avril à 10h25 :

```
25 10 15 04 * /root/scripts/script > /log/…
```

Exécution à 11h puis à 16h tous les jours :

```
00 11,16 * * * /root/scripts/script > /log/…
```

Exécution toutes les heures de 11h à 16h tous les jours :

```
00 11-16 * * * /root/scripts/script > /log/…
```

Exécution toutes les 10 minutes aux heures de travail :

```
*/10 8-17 * * 1-5 /root/scripts/script > /log/…
```

Processus d'exécution d'une tâche

Un utilisateur, Patux, veut éditer son fichier de planification :

1) crond vérifie s'il est autorisé (/etc/cron.allow et /etc/cron.deny).

2) Si c'est le cas, il accède à son fichier de planification (/var/spool/cron/Pierre).

Toutes les minutes crond lit les fichiers de planification.

3) Il y exécute les tâches planifiées.

4) Il rend compte systématiquement dans un fichier journal (/var/log/cron).

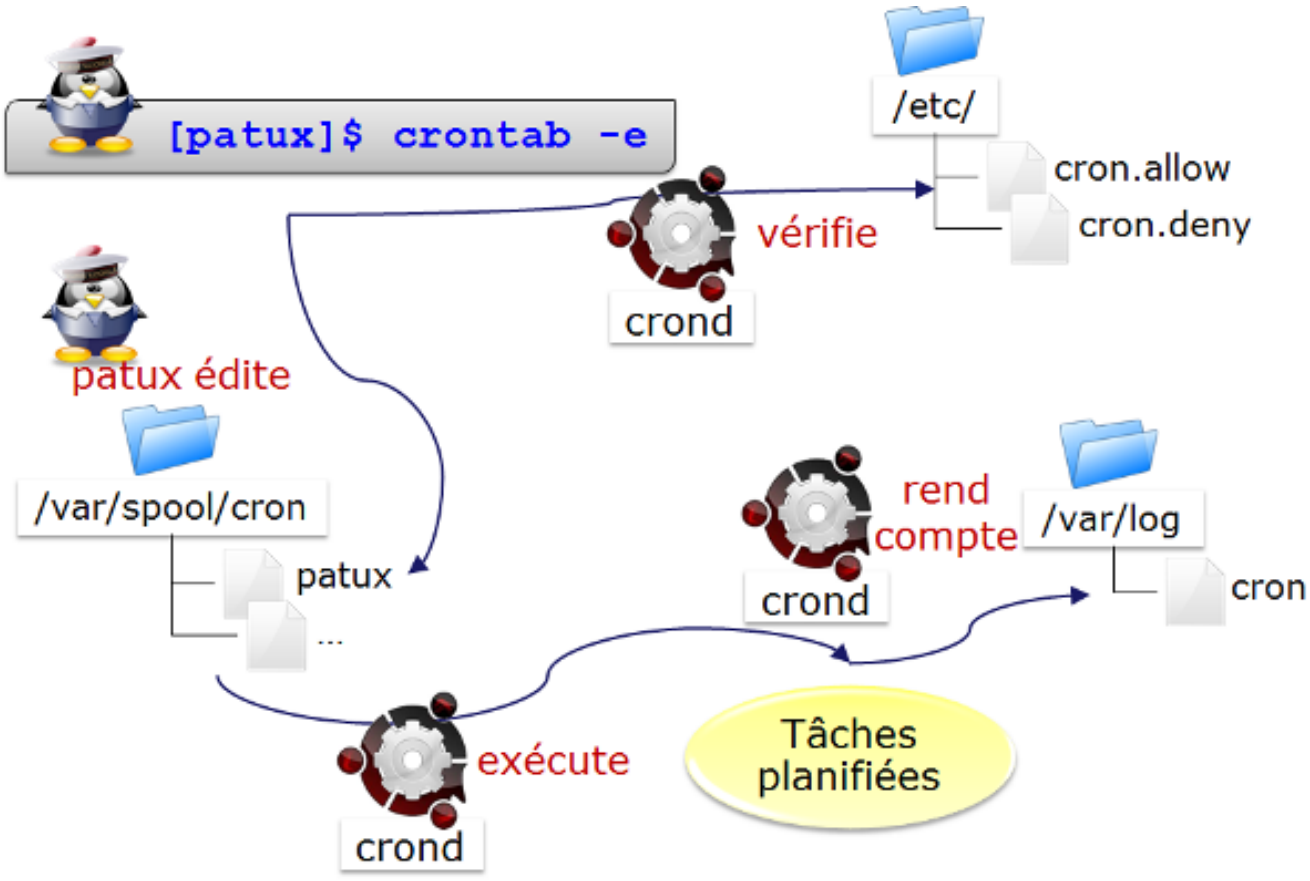


Figure 31. Processus d'exécution d'une tâche

Chapitre 12. Mise en oeuvre du réseau

12.1. Généralités

Pour illustrer ce cours, nous allons nous appuyer sur l'architecture suivante.

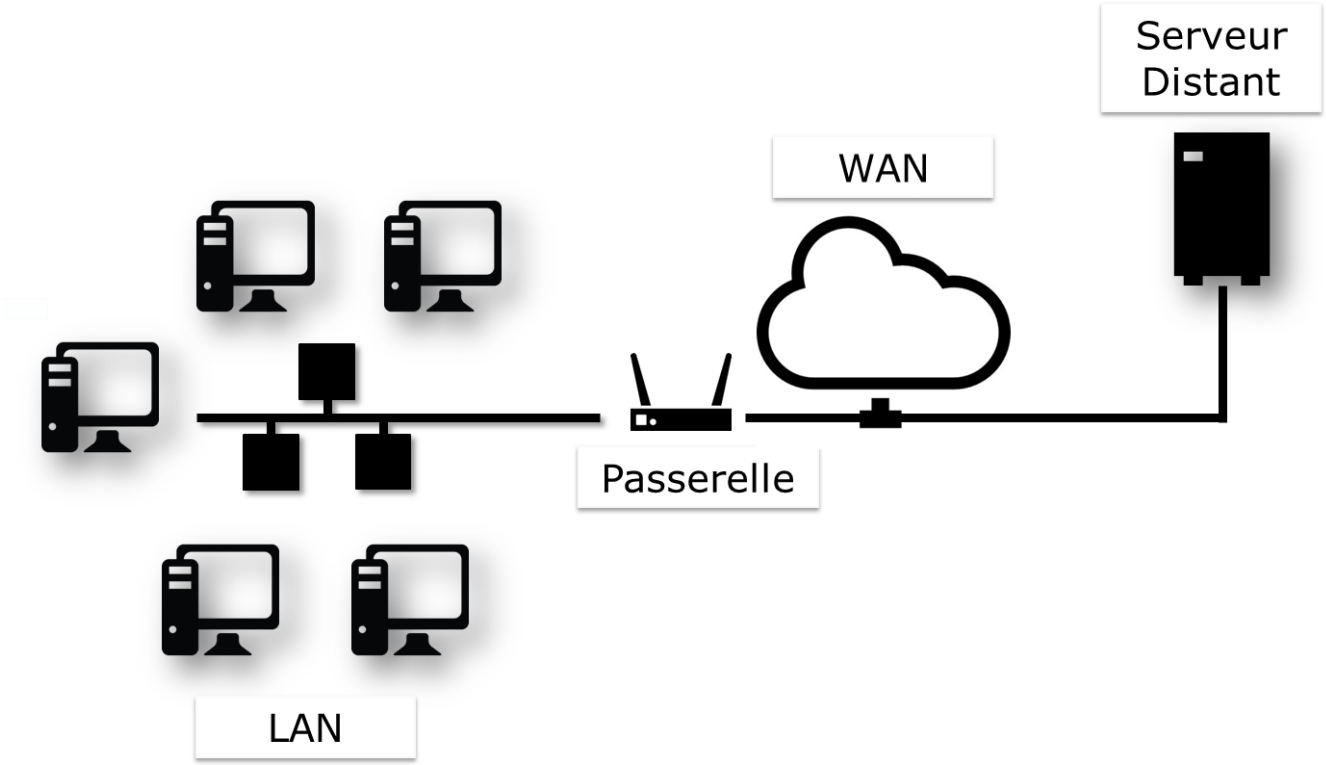


Figure 32. Illustration de notre architecture réseau

Elle nous permettra de considérer :

- l'intégration dans un LAN (local area network, ou réseau local) ;
- la configuration d'une passerelle pour joindre un serveur distant ;
- la configuration d'un serveur DNS puis mettre en œuvre la résolution de nom.

Les paramètres minimum à définir propres à la machine sont :

- le nom de la machine ;
- l'adresse IP ;
- le masque de sous-réseau.

Exemple :

- pc-tux ;
- 192.168.1.10 ;
- 255.255.255.0.

La notation appelée CIDR est de plus en plus fréquente : 192.168.1.10/24

Les adresses IP servent au bon acheminement des messages. Elles sont fractionnées en deux parties :

- la partie fixe, identifiant le réseau ;
- l'identifiant de l'hôte dans le réseau.

Le masque de sous-réseau est un ensemble de **4 octets** destiné à isoler :

- l'adresse de réseau (**NetID** ou **SubnetID**) en effectuant un ET logique bit à bit entre l'adresse IP et le masque ;
- l'adresse de l'hôte (**HostID**) en effectuant un ET logique bit à bit entre l'adresse IP et le complément du masque.

Il existe également des adresses spécifiques au sein d'un réseau, qu'il faut savoir identifier. La première adresse d'une plage ainsi que la dernière ont un rôle particulier :

- La première adresse d'une plage est l'**adresse du réseau**. Elle permet d'identifier les réseaux et de router les informations d'un réseau à un autre.
- La dernière adresse d'une plage est l'**adresse de broadcast**. Elle permet de faire de la diffusion à toutes les machines du réseau.

Adresse MAC / Adresse IP

Une **adresse MAC** est un identifiant physique inscrit en usine dans une mémoire. Elle est constituée de 6 octets souvent donnée sous forme hexadécimale (par exemple 5E:FF:56:A2:AF:15). Elle se compose de : 3 octets de l'identifiant constructeur et 3 octets du numéro de série.

Cette dernière affirmation est aujourd'hui un peu moins vraie avec la virtualisation. Il existe également des solutions pour changer logiciellement l'adresse MAC.

Une **adresse IP** (Internet Protocol) est un numéro d'identification attribuée de façon permanente ou provisoire à chaque appareil connecté à un réseau informatique utilisant l'Internet Protocol. Une partie définit l'adresse du réseau (NetID ou SubnetID suivant le cas), l'autre partie définit l'adresse de l'hôte dans le réseau (HostID). La taille relative de chaque partie varie suivant le masque de (sous) réseau.

Une adresse IPv4 définit une adresse sur 4 octets. Le nombre d'adresse disponible étant proche de la saturation un nouveau standard a été créé, l'IPv6 définie sur 16 octets. L'IPv6 est souvent représenté par 8 groupes de 2 octets séparés par un signe deux-points. Les zéro non significatifs peuvent être omis, un ou plusieurs groupes de 4 zéros consécutifs peuvent être remplacés par un double deux-points. Les masques de sous-réseaux ont de 0 à 128 bits. (par exemple 21ac:0000:0000:0611:21e0:00ba:321b:54da/64 ou 21ac::611:21e0:ba:321b:54da/64)

Dans une adresse web ou URL (Uniform Resource Locator), une adresse ip peut être suivi de deux-points, l'adresse de port (qui indique l'application à laquelle les données sont destinées). Aussi pour éviter toute confusion dans une URL, l'adresse IPv6 s'écrit entre crochets [ ], deux-points, adresse de port.

Les adresses IP et MAC doivent être uniques sur un réseau !

Sous VMWare, choisir l'option « I copied it » au lancement d'une VM génère une nouvelle adresse MAC.

Domaine DNS

Les postes clients peuvent faire partie d'un domaine DNS (**Domain Name System**, système de noms de domaine, par exemple mondomaine.lan).

Le nom de machine pleinement qualifié (FQDN) devient pc-tux.mondomaine.lan.

Un ensemble d'ordinateurs peut être regroupé dans un ensemble logique, permettant la résolution de nom, appelé domaine DNS. Un domaine DNS n'est pas, bien entendu, limité à un seul réseau physique.

Pour qu'un ordinateur intègre un domaine DNS, il faudra lui spécifier un suffixe DNS (ici mondomaine.lan) ainsi que des serveurs qu'il pourra interroger.

Rappel du modèle OSI

Aide mémoire : Pour se souvenir de l'ordre PLRTSPA, retenir la phrase suivante : *Pour Les Réseaux Tu Seras Pas Augmenté*.

Table 67. Les 7 couches du modèle OSI

Couche	Protocoles
7 - Application	POP, IMAP, SMTP, SSH, SNMP, HTTP, FTP, …
6 - Présentation	ASCII, MIME, …
5 - Session	TLS, SSL, NetBIOS, …
4 - Transport	TLS, SSL, TCP, UDP,…
3 - Réseau	IPv4, IPv6, ARP,…
2 - Liaison	Ethernet, WiFi, Token Ring,…
1 - Physique	Câbles, fibres optiques, ondes radio,…

La couche 1 (Physique) prend en charge la transmission sur un canal de communication (Wifi, Fibre optique, câble RJ, etc.). Unité : le bit.

La couche 2 (Liaison) prend en charge la topologie du réseau (Token-ring, étoile, bus, etc.), le fractionnement des données et les erreurs de transmissions. Unité : la trame.

La couche 3 (Réseau) prend en charge la transmission de bout en bout des données (routage IP = Passerelle). Unité : le paquet.

La couche 4 (Transport) prend en charge le type de service (connecté ou non connecté), le chiffrement et le contrôle de flux. Unité : le segment ou le datagramme.

La couche 7 (Application) représente le contact avec l'utilisateur. Elle apporte les services offerts par le réseau : http, dns, ftp, imap, pop, smtp, etc.

12.2. Le nommage des interfaces

lo est l'interface "**loopback**" qui permet à des programmes TCP/IP de communiquer entre eux sans sortir de la machine locale. Cela permet de tester si le **module « réseau » du système fonctionne bien** et aussi de faire un ping localhost. Tous les paquets qui entrent par localhost ressortent par localhost. Les paquets reçus sont les paquets envoyés.

Le noyau Linux attribue des noms d'interfaces composés d'un préfixe précis selon le type. Sur des distributions Linux RHEL 6, toutes les interfaces **Ethernet**, par exemple, commencent par **eth**. Le préfixe est suivi d'un chiffre, le premier étant 0 (eth0, eth1, eth2...). Les interfaces wifi se voient attribuées un préfixe wlan.

12.3. Utiliser la commande IP

Oubliez l'ancienne commande **ifconfig** ! Pensez **ip** !

Commentaire à destination des administrateurs d'anciens systèmes Linux :

La commande historique de gestion du réseau est **ifconfig**. Cette commande a tendance a être remplacée par la commande **ip**, déjà bien connue des administrateurs réseaux.

La commande **ip** est la commande unique pour gérer l'adresse **IP, ARP, le routage, etc.**

La commande ifconfig n'est plus installée par défaut sous RHEL 7. Il est important de prendre des bonnes habitudes dès maintenant.

12.4. Le nom de machine

La commande hostname affiche ou définit le nom d'hôte du système

Syntaxe de la commande hostname

```
hostname [-f] [hostname]
```

Table 68. Options principales de la commande hostname

Option	Description
-f	Affiche le FQDN
-i	Affiche les adresses IP du système

Cette commande est utilisée par différents programmes réseaux pour identifier la machine.

Pour affecter un nom d'hôte, il est possible d'utiliser la commande hostname, mais les changements ne seront pas conservés au prochain démarrage. La commande sans argument permet d'afficher le nom de l'hôte.

Pour fixer le nom d'hôte, il faut modifier le fichier ***/etc/sysconfig/network*** :

Le fichier /etc/sysconfig/network

```
NETWORKING=yes
HOSTNAME=stagiaire.mondomaine.lan
```

Le script de démarrage sous RedHat consulte également le fichier **/etc/hosts** pour résoudre le nom d'hôte du système.

Lors du démarrage du système, Linux vient évaluer la valeur **HOSTNAME** du fichier **/etc/sysconfig/network**.

Il utilise ensuite le fichier **/etc/hosts** pour évaluer l'adresse IP principale du serveur et son nom d'hôte. Il en déduit le nom de domaine DNS.

Il est donc primordiale de bien renseigner ces deux fichiers avant toute configuration de services réseaux.

Pour savoir si cette configuration est bien faîte, les commandes hostname et hostname –f doivent répondre les bonnes valeurs attendues.

12.5. Le fichier /etc/hosts

Le fichier **/etc/hosts** est une table de correspondance statique des noms d'hôtes, qui respecte le format suivant :

Syntaxe du fichier /etc/hosts

```
@IP <nom d'hôte>  [alias]  [# commentaire]
```

Exemple de fichier /etc/hosts :

Exemple de fichier /etc/hosts

```
127.0.0.1   localhost localhost.localdomain
::1         localhost localhost.localdomain
192.168.1.10    stagiaire.mondomaine.lan stagiaire
```

Le fichier **/etc/hosts** est encore employé par le système, notamment lors du démarrage durant lequel le FQDN du système est déterminé.

RedHat préconise qu'au moins une ligne contenant le nom du système soit renseignée.

Si le service DNS (Domain Name Service) n'est pas en place, vous devez renseigner tous les noms dans le fichier hosts de chacune de vos machines.

Le fichier /etc/hosts contient une ligne par entrée, comportant l'adresse IP, le FQDN, puis le nom d'hôte (dans cet ordre) et une suite d'alias (alias1 alias2 ...). L'alias est une option.

12.6. Le fichier /etc/nsswitch.conf

```
Le Name Service Switch (NSS) permet de substituer des fichiers de configuration (par
exemple /etc/passwd, /etc/group, /etc/hosts) par une ou plusieurs bases de données
centralisées
```

Le fichier /etc/nsswitch.conf permet de configurer les bases de données du service de noms.

Le fichier /etc/nsswitch.conf

```
passwd: files
shadow: files
group: files

hosts: files dns
```

Dans le cas présent, Linux cherchera en premier une correspondance de noms d'hôtes (ligne hosts:) dans le fichier /etc/hosts (valeur files) avant d'interroger le DNS (valeur dns)! Ce comportement peut simplement être changé en éditant le fichier /etc/nsswitch.conf.

Bien évidemment, il est possible d'imaginer interroger un serveur LDAP, MySQL ou autre en configurant le service de noms pour répondre aux requêtes du systèmes sur les hosts, les utilisateurs, les groupes, etc.

La résolution du service de noms peut être testée avec la commande getent que nous verrons plus loin dans ce cours.

12.7. Le fichier /etc/resolv.conf

Le fichier /etc/resolv.conf contient la configuration de la résolution de nom DNS.

/etc/resolv.conf

```
#Generated by NetworkManager
domain mondomaine.lan
search mondomaine.lan
nameserver 192.168.1.254
```

Ce fichier est historique. Il n'est plus renseigné directement !

Les nouvelles générations de distributions ont généralement intégré le service NetworkManager. Ce service permet de gérer plus efficacement la configuration, que ce soit en mode graphique ou console.

Il permet notamment de configurer les serveurs DNS depuis le fichier de configuration d'une interface réseau. Il se charge alors de renseigner dynamiquement le fichier /etc/resolv.conf qui ne devrait jamais être édité directement, sous peine de perdre les changements de configuration au prochain démarrage du service réseau.

12.8. La commande ip

La commande ip du paquet iproute2 permet de configurer une interface et sa table routage.

Afficher les interfaces :

```
[root]# ip link
```

Afficher les informations des interfaces :

```
[root]# ip addr show
```

Afficher les informations d'une interface :

```
[root]# ip addr show eth0
```

Afficher la table ARP:

```
[root]# ip neigh
```

Toutes les commandes historiques de gestion du réseau ont été regroupées sous la commande IP, bien connue des administrateurs réseaux.

12.9. Configuration DHCP

Le protocole DHCP (Dynamic Host Control Protocol) permet d'obtenir via le réseau une configuration IP complète. C'est le mode de configuration par défaut d'une interface réseau sous RedHat, ce qui explique qu'un système branché sur le réseau d'une box internet puisse fonctionner sans configuration supplémentaire.

La configuration des interfaces sous RHEL 6 se fait dans le dossier **/etc/sysconfig/network-scripts/**.

Pour chaque interface ethernet, un fichier **ifcfg-ethX** permet de configurer l'interface associée.

/etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
HWADDR=00:0c:29:96:32:e3
```

- Nom de l'interface : (doit être dans le nom du fichier)

```
DEVICE=eth0
```

- Démarrer automatiquement l'interface :

```
ONBOOT=yes
```

- Effectuer une requête DHCP au démarrage de l'interface :

```
BOOTPROTO=dhcp
```

- Spécifier l'adresse MAC (facultatif mais utile lorsqu'il y a plusieurs interfaces) :

```
HWADDR=00:0c:29:96:32:e3
```

Si NetworkManager est installé, les modifications sont prises en compte automatiquement. Sinon, il faut redémarrer le service réseau.

Redémarrer le service réseau :

```
[root]# service network restart
```

et sous RHEL 7 :

```
[root]# systemctl restart network
```

12.10. Configuration statique

La configuration statique nécessite à minima :

/etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=none
IPADDR=192.168.1.10
NETMASK=255.255.255.0
```

- Ne pas utiliser DHCP = configuration statique

```
BOOTPROTO=none
```

- Adresse IP :

```
IPADDR=192.168.1.10
```

- Masque de sous-réseau :

```
NETMASK=255.255.255.0
```

- Le masque peut être spécifié avec un préfixe :

```
PREFIX=24
```

Il faut utiliser NETMASK OU PREFIX - Pas les deux !

Pensez à redémarrer le service network !

12.11. Routage

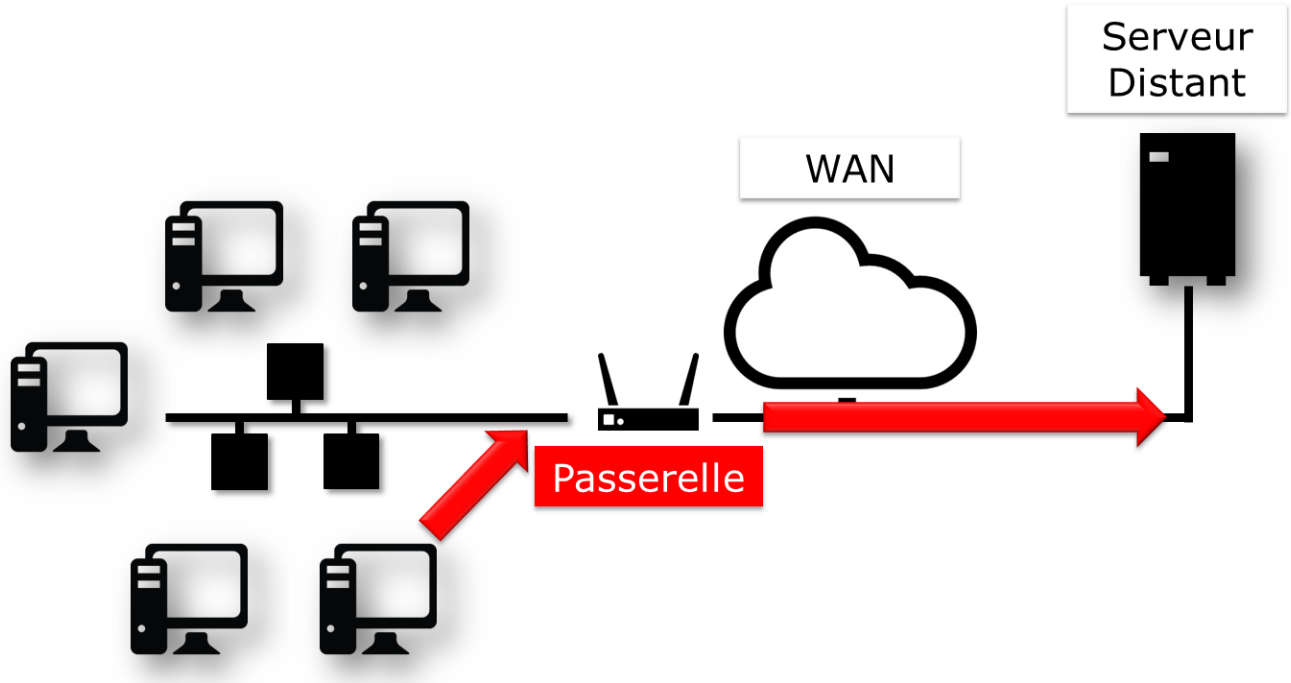


Figure 33. Architecture réseau avec une passerelle

/etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=none
HWADDR=00:0c:29:96:32:e3
IPADDR=192.168.1.10
NETMASK=255.255.255.0
GATEWAY=192.168.1.254
```

Pensez à redémarrer le service network !

La commande ip route

```
[root]# ip route show
192.168.1.0/24 dev eth0 [···] src 192.168.1.10 metric 1
default via 192.168.1.254 dev eth0 proto static
```

Il est judicieux de savoir lire une table de routage, surtout dans un environnement disposant de plusieurs interfaces réseaux.

- Dans l'exemple présenté, le réseau 192.168.1.0/24 est directement accessible depuis le périphérique eth0, il y a donc une métrique à 1 (ne traverse pas de routeur).

- Tous les autres réseaux que le réseau précédent seront joignables, toujours depuis le périphérique eth0, mais cette fois-ci les paquets seront adressés à une passerelle 192.168.1.254. Le protocole de routage est un protocole statique (bien qu'il soit possible d'ajouter un protocole de routage dynamique sous Linux).

12.12. Résolution de noms

Un système a besoin de résoudre :

- des FQDN en adresses IP

```
www.free.fr = 212.27.48.10
```

- des adresses IP en noms

```
212.27.48.10 = www.free.fr
```

- ou d'obtenir des informations sur une zone :

```
MX de free.fr = 10 mx1.free.fr + 20 mx2.free.fr
```

/etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=none
HWADDR=00:0c:29:96:32:e3
IPADDR=192.168.1.10
NETMASK=255.255.255.0
GATEWAY=192.168.1.254
DNS1=172.16.1.2
DNS2=172.16.1.3
DOMAIN=mondomaine.lan
```

Dans ce cas, pour joindre les DNS, il faut passer par la passerelle.

/etc/resolv.conf

```
#Generated by NetworkManager
domain mondomaine.lan
search mondomaine.lan
nameserver 172.16.1.2
nameserver 172.16.1.3
```

Le fichier a bien été mis à jour par NetworkManager.

12.13. Dépannage

La commande ping permet d'envoyer des datagrammes à une autre machine et attend une réponse.

C'est la commande de base pour tester le réseau car elle vérifie la connectivité entre votre interface réseau et une autre.

La syntaxe de la commande ping

```
ping [-c numérique] destination
```

L'option -c (count) permet de stopper la commande au bout du décompte exprimé en seconde.

Exemple :

```
[root]# ping –c 4 localhost
```

Validez la connectivité du plus proche au plus lointain

1) Valider la couche logicielle TCP/IP

```
[root]# ping localhost
```

« Pinguer » la boucle interne ne permet pas de détecter une panne matérielle sur l'interface réseau. Elle permet simplement de déterminer si la configuration logicielle IP est correcte.

2) Valider la carte réseau

```
[root]# ping 192.168.1.10
```

Pour déterminer que la carte réseau est fonctionnelle, il faut maintenant faire un « ping » de son adresse IP. La carte réseau, si le câble réseau n'est pas connecté, devrait être dans un état « down ».

Si le ping ne fonctionne pas, vérifier dans un premier temps le câble réseau vers votre le commutateur réseau et remonter l'interface (voir la commande if up), puis vérifiez l'interface elle-même.

3) Valider la connectivité de la passerelle

```
[root]# ping 192.168.1.254
```

4) Valider la connectivité d'un serveur distant

```
[root]# ping 172.16.1.2
```

5) Valider le service DNS

```
[root]# ping www.free.fr
```

La commande dig

La commande dig (dig : en français 'miner', chercher en profondeur) permet d'interroger le serveur DNS.

Syntaxe de la commande dig

```
dig [-t type] [+short] [name]
```

Exemples :

```
[root]# dig +short www.formatux.fr
46.19.120.31
[root]# dig –t MX +short formatux.fr
10 smtp.formatux.fr.
```

La commande **DIG** permet d'interroger les **serveurs DNS**. Elle est par défaut très verbeuse, mais ce comportement peut être changé grâce à l'option **+short**.

Il est également possible de spécifier un **type d'enregistrement** DNS à résoudre, comme par exemple un **type MX** pour obtenir des renseignements sur les serveurs de messagerie d'un domaine.

Pour plus d'informations à ce sujet, voir le cours « DNS avec Bind9 »

La commande getent

La commande **getent** (get entry) permet d'obtenir une entrée de NSSwitch (hosts + dns)

Syntaxe de la commande getent

```
getent hosts name
```

Exemple :

```
[root]# getent hosts www.formatux.fr
  46.19.120.31 www.formatux.fr
```

Interroger uniquement un serveur DNS peut renvoyer un résultat erroné qui ne prendrait pas en compte le contenu d'un fichier hosts, bien que ce cas de figure devrait être rare aujourd'hui.

Pour prendre en compte également le fichier **/etc/hosts**, il faut interroger le service de noms NSSwitch, qui se chargera d'une éventuelle résolution DNS.

La commande ipcalc

La commande ipcalc (ip calcul) permet de calculer l'adresse d'un réseau ou d'un broadcast depuis une adresse IP et un masque.

Syntaxe de la commande ipcalc

```
ipcalc  [options] IP <netmask>
```

Exemple :

```
[root]# ipcalc –b 172.16.66.203 255.255.240.0
BROADCAST=172.16.79.255
```

Cette commande est intéressante suivie d'une redirection pour renseigner automatiquement les fichiers de configuration de vos interfaces :

```
[root]# ipcalc –b 172.16.66.203 255.255.240.0 >>
/etc/sysconfig/network-scripts/ifcfg-eth0
```

Option	Description
-b	Affiche l'adresse de broadcast.
-n	Affiche l'adresse du réseau et du masque.

ipcalc permet de calculer simplement les informations IP d'un hôte. Les diverses options indiquent quelles informations ipcalc doit afficher sur la sortie standard. Des options multiples peuvent être indiquées. Une adresse IP sur laquelle opérer doit être spécifiée. La plupart des opérations nécessitent aussi un masque réseau ou un préfixe CIDR.

Table 69. Options principales de la commande ipcalc

Option courte	Option longue	Description
-b	--broadcast	Affiche l'adresse de diffusion de l'adresse IP donnée et du masque réseau.
-h	--hostname	Affiche le nom d'hôte de l'adresse IP donnée via le DNS.
-n	--netmask	Calcule le masque réseau de l'adresse IP donnée. Suppose que l'adresse IP fait partie d'un réseau de classe A, B, ou C complet. De nombreux réseaux n'utilisent pas les masques réseau par défaut, dans ce cas une valeur incorrecte sera retournée.
-p	--prefix	Indique le préfixe de l'adresse masque/IP.
-n	--network	Indique l'adresse réseau de l'adresse IP et du masque donné.
-s	--silent	N'affiche jamais aucun message d'erreur.

La commande ss

La commande ss (socket statistics) affiche les ports en écoute sur le réseau

Syntaxe de la commande ss

```
ss [-tuna]
```

Exemple :

```
[root]# ss –tuna
tcp   LISTEN   0   128   *:22   *:*
```

Les commande **ss** et **netstat** (à suivre) vont se révéler très importantes pour la suite de votre cursus Linux.

Lors de la mise en œuvre des services réseaux, il est très fréquent de vérifier avec l'une de ces deux commandes que le service est bien en écoute sur les ports attendus.

La commande netstat

La commande netstat (network statistics) affiche les ports en écoute sur le réseau

Syntaxe de la commande netstat

```
netstat -tapn
```

Exemple :

```
[root]# netstat –tapn
tcp  0  0  0.0.0.0:22  0.0.0.0:*  LISTEN 2161/sshd
```

Les conflits d'adresses IP ou d'adresses MAC

Un défaut de configuration peut amener plusieurs interfaces à utiliser la même adresse IP. Cette situation peut se produire lorsqu'un réseau dispose de plusieurs serveurs DHCP ou lorsque la même adresse IP est manuellement assignée plusieurs fois.

Lorsque le réseau fonctionne mal, que des dysfonctionnements ont lieu, et qu'un conflit d'adresses IP pourrait en être à l'origine, il est possible d'utiliser le logiciel arp-scan (nécessite le dépôt EPEL) :

```
$ yum install arp-scan
```

Exemple :

```
$ arp-scan -I eth0 -l

172.16.1.104  00:01:02:03:04:05       3COM CORPORATION
172.16.1.107  00:0c:29:1b:eb:97       VMware, Inc.
172.16.1.250  00:26:ab:b1:b7:f6       (Unknown)
172.16.1.252  00:50:56:a9:6a:ed       VMWare, Inc.
172.16.1.253  00:50:56:b6:78:ec       VMWare, Inc.
172.16.1.253  00:50:56:b6:78:ec       VMWare, Inc. (DUP: 2)
172.16.1.253  00:50:56:b6:78:ec       VMWare, Inc. (DUP: 3)
172.16.1.253  00:50:56:b6:78:ec       VMWare, Inc. (DUP: 4)
172.16.1.232   88:51:fb:5e:fa:b3       (Unknown) (DUP: 2)
```

Comme l'exemple ci-dessus le démontre, il est également possible d'avoir des conflits d'adresses MAC ! Ces problématiques sont apportées par les technologies de virtualisation et la recopie de machines virtuelles.

12.14. Configuration à chaud

La commande ip peut ajouter à chaud une adresse IP à une interface

```
ip addr add @IP dev DEVICE
```

Exemple :

```
[root]# ip addr add 192.168.2.10 dev eth1
```

La commande ip permet d'activer ou désactiver une interface :

```
ip link set DEVICE up
ip link set DEVICE down
```

Exemple :

```
[root]# ip link set eth1 up
[root]# ip link set eth1 down
```

La commande ip permet d'ajouter une route :

```
ip route add [default|netaddr] via @IP [dev device]
```

Exemple :

```
[root]# ip route add default via 192.168.1.254
[root]# ip route add 192.168.100.0/24 via 192.168.2.254 dev eth1
```

12.15. En résumé

Les fichiers mis en oeuvre durant ce chapitre sont :

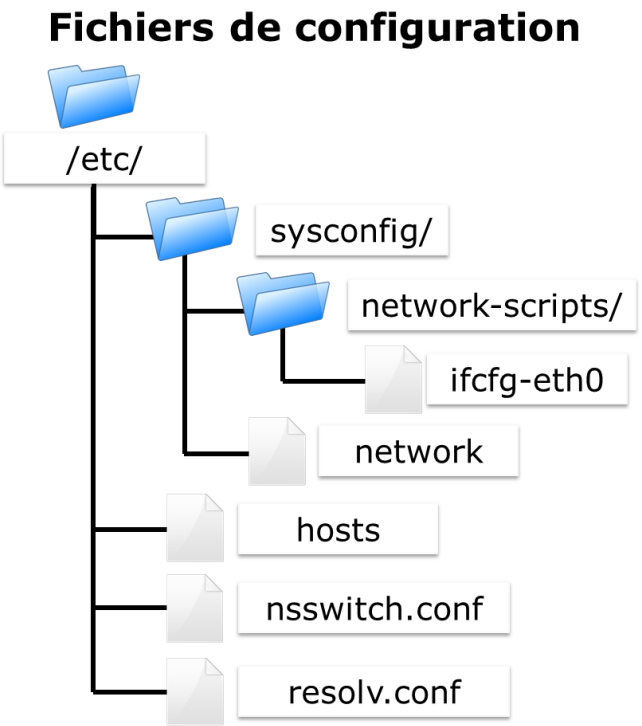


Figure 34. Synthèse des fichiers mis en oeuvre dans la partie réseau

Une configuration complète d'une interface pourrait être celle-ci :

/etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=none
HWADDR=00:0c:29:96:32:e3
IPADDR=192.168.1.10
NETMASK=255.255.255.0
GATEWAY=192.168.1.254
DNS1=172.16.1.1
DNS2=172.16.1.2
DOMAIN=formatux.fr
```

La méthode de dépannage doit aller du plus proche au plus lointain :

1. ping localhost (test logiciel)
2. ping adresse-IP (test matériel)
3. ping passerelle (test connectivité)
4. ping serveur-distant (test routage)
5. Interrogation DNS (dig ou ping)

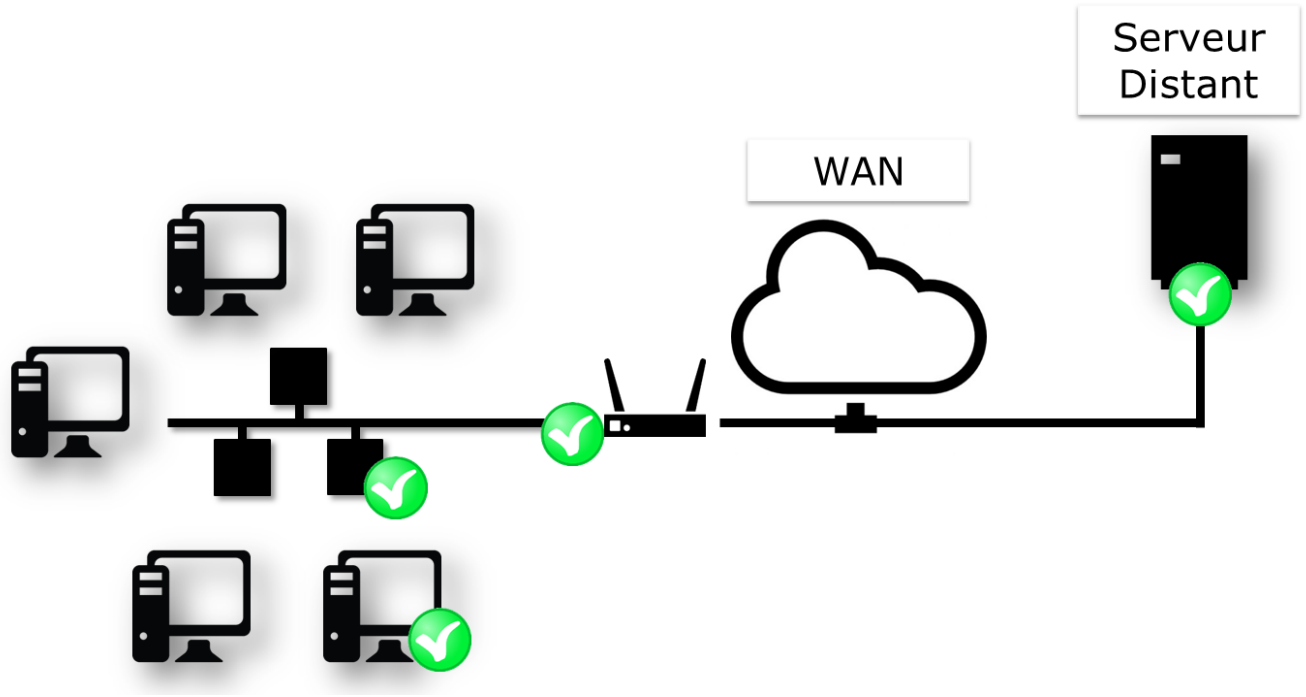


Figure 35. Méthode de dépannage ou de validation du réseau

Chapitre 13. Gestion des logiciels

13.1. Généralités

Sur un système Linux, il est possible d'installer un logiciel de deux façons :

- en utilisant un paquet d'installation ;
- en compilant les fichiers sources.

Le paquet : Il s'agit d'un unique fichier comprenant toutes les données utiles à l'installation du programme. Il peut être exécuté directement sur le système à partir d'un dépôt logiciel.

Les fichiers sources : Certains logiciels ne sont pas fournis dans des paquets prêts à être installés mais via une archive contenant les fichiers sources. Charge à l'administrateur de préparer ces fichiers et de les compiler pour installer le programme.

13.2. RPM : RedHat Package Manager

RPM (RedHat Package Manager) est un système de gestion des logiciels. Il est possible d'installer, de désinstaller, de mettre à jour ou de vérifier des logiciels contenus dans des paquets.

RPM est le format utilisé par toutes les distributions à base RedHat (Fedora, CentOS, SuSe, Mandriva, ...). Son équivalent dans le monde de Debian est DPKG (Debian Package).

Le nom d'un paquet RPM répond à une nomenclature précise :

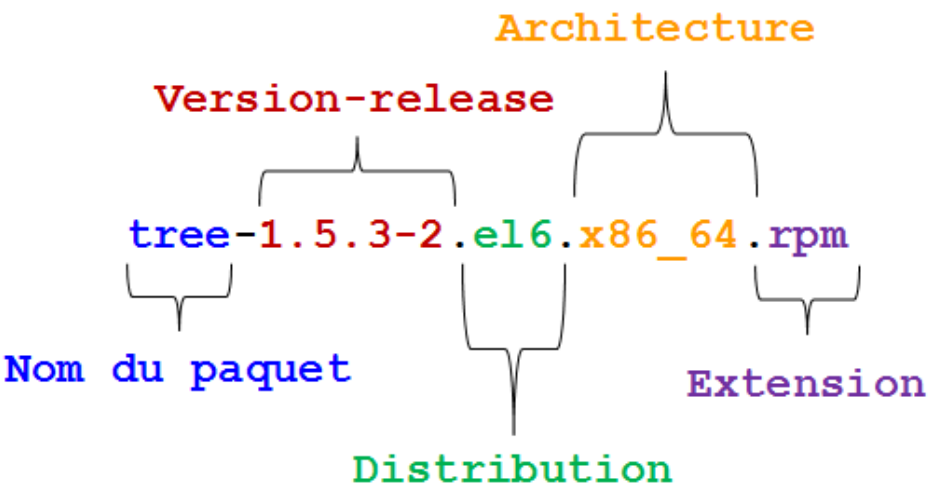


Figure 36. nom-version-edition.architecture.rpm

Commande rpm

La commande rpm permet d'installer un paquet.

Syntaxe de la commande rpm

```
rpm [-i][-U] paquet.rpm [-e] paquet
```

Exemple :

```
[root]# rpm -ivh paquet.rpm
```

Table 70. Options de la commande rpm

Option	Description
-i *paquet.rpm*	Installe le paquet.
-U *paquet.rpm*	Met à jour un paquet déjà installé.
-e *paquet.rpm*	Désinstalle le paquet.
-h	Affiche une barre de progression.
-v	Informe sur l'avancement de l'opération.
--test	Teste la commande sans l'exécuter.

La commande rpm permet aussi d'interroger la base de données des paquets du système en ajoutant l'option **-q**.

Il est possible d'exécuter plusieurs types de requêtes pour obtenir différentes informations sur les paquets installés. La base de donnée RPM se trouve dans le répertoire **/var/lib/rpm** .

Exemple :

```
[root]# rpm -qa
```

Cette commande interroge tous les paquets installés sur le système.

Syntaxe de la commande rpm pour interrogation

```
rpm -q [-a][-i][-l] paquet [-f] fichier
```

Exemple :

```
[root]# rpm -qil paquet
[root]# rpm -qf /chemin/fichier
```

Table 71. Options de la commande rpm pour interrogation

Option	Description
-a	Liste tous les paquets installés sur le système.
-i *paquet*	Affiche les informations du paquet.
-l *paquet*	Liste les fichiers contenus dans le paquet.
-f	Affiche le nom du paquet contenant le fichier précisé.
--last	La liste des paquets est donnée par date d'installation (les derniers paquetages installés apparaissent en premier).

Après l'option **-q**, le nom du paquet doit être exact. Les métacaractères ne sont pas gérés.

Il est cependant possible de lister tous les paquets installés et de filtrer avec la commande **grep**.

Exemple : lister les derniers paquets installés :

```
sudo rpm -qa --last | head
kernel-devel-3.10.0-514.21.2.el7.x86_64       mar. 27 juin 2017 14:52:21 CEST
webmin-1.850-1.noarch                         mar. 27 juin 2017 14:51:59 CEST
kernel-3.10.0-514.21.2.el7.x86_64             mar. 27 juin 2017 14:51:32 CEST
sudo-1.8.6p7-23.el7_3.x86_64                  mar. 27 juin 2017 14:51:23 CEST
python-perf-3.10.0-514.21.2.el7.x86_64        mar. 27 juin 2017 14:51:22 CEST
kernel-tools-3.10.0-514.21.2.el7.x86_64       mar. 27 juin 2017 14:51:22 CEST
glibc-headers-2.17-157.el7_3.4.x86_64         mar. 27 juin 2017 14:51:22 CEST
glibc-devel-2.17-157.el7_3.4.x86_64           mar. 27 juin 2017 14:51:22 CEST
kernel-headers-3.10.0-514.21.2.el7.x86_64     mar. 27 juin 2017 14:51:21 CEST
kernel-tools-libs-3.10.0-514.21.2.el7.x86_64  mar. 27 juin 2017 14:51:20 CEST
```

Exemple : lister l'historique d'installation du kernel :

```
sudo rpm -qa --last kernel
kernel-3.10.0-514.21.2.el7.x86_64             mar. 27 juin 2017 14:51:32 CEST
kernel-3.10.0-514.21.1.el7.x86_64             mar. 20 juin 2017 10:31:38 CEST
kernel-3.10.0-514.16.1.el7.x86_64             lun. 17 avril 2017 09:41:52 CEST
kernel-3.10.0-514.10.2.el7.x86_64             lun. 27 mars 2017 07:07:10 CEST
kernel-3.10.0-514.6.1.el7.x86_64              dim. 05 févr. 2017 18:40:50 CET
```

13.3. YUM : Yellow dog Updater Modified

YUM est un gestionnaire de paquets logiciels. Il fonctionne avec des paquets RPM regroupés dans un dépôt (un répertoire de stockage des paquets) local ou distant.

La commande **yum** permet la gestion des paquets en comparant ceux installés sur le système à ceux présents dans les dépôts définis sur le serveur. Elle permet aussi d'installer automatiquement les dépendances, si elles sont également présentes dans les dépôts.

YUM est le gestionnaire utilisé par de nombreuses distributions à base RedHat (Fedora, CentOS, ...). Son équivalent dans le monde Debian est APT (Advanced Packaging Tool).

Commande yum

La commande yum permet d'installer un paquet en ne spécifiant que le nom court.

Syntaxe de la commande yum

```
yum [install][remove][list all][search][info] paquet
```

Exemple :

```
[root]# yum install tree
```

Seul le nom court du paquet est nécessaire.

Table 72. Options de la commande yum

Option	Description
install	Installe le paquet.
remove	Désinstalle le paquet.
list all	Liste les paquets déjà présents dans le dépôt.
search	Recherche un paquet dans le dépôt.
*provides */nom_cmde*	Recherche une commande.
info	Affiche les informations du paquet.

La commande **yum list** liste tous les paquets installés sur le système et présents dans le dépôt. Elle accepte plusieurs paramètres :

Table 73. Paramètres de la commande yum list

Paramètre	Description
all	Liste les paquets installés puis ceux disponible sur les dépôts.
available	Liste uniquement les paquets disponible pour installation.
updates	Liste les paquets pouvant être mis à jour.
obsoletes	Liste les paquets rendus obsolètes par des versions supérieures disponibles.
recent	Liste les derniers paquets ajoutés au dépôt.

Exemple de recherche de la commande semanage :

```
[root]# yum provides */semanage
```

Fonctionnement de YUM

Sur un poste client, le gestionnaire YUM s'appuie sur un ou plusieurs fichiers de configuration afin de cibler les dépôts contenant les paquets RPM.

Ces fichiers sont situés dans **/etc/yum.repos.d/** et se terminent obligatoirement par **.repo** afin d'être exploités par YUM.

Exemple :

```
/etc/yum.repos.d/MonDepotLocal.repo
```

Chaque fichier **.repo** se constitue au minimum des informations suivantes, une directive par ligne. Exemple:

```
[DepotLocal]    #Nom court du dépôt
name=Mon dépôt local    #Nom détaillé
baseurl=http://..... ou file:///.....    #Adresse http ou local
enabled=1   #Activation =1, ou non activé =0"
gpgcheck=1  #Dépôt demandant une signature
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6 #Chemin de la clef publique GPG
```

Par défaut, la directive **enabled** est absente ce qui signifie que le dépôt est activé. Pour désactiver un dépôt, il faut spécifier la directive **enabled=0**.

13.4. Gérer son dépôt

La création d'un dépôt permet de disposer de sa propre banque de paquets. Celle-ci peut-être disponible par exemple par point de montage ou mise à disposition sur un serveur web.

Les étapes de la création d'un dépôt sur un serveur sont les suivantes :

- Créer un répertoire qui va accueillir tous les paquets rpm ;

```
[root]# mkdir /MonDepot
```

- Copier tous les paquets rpm nécessaires dans ce dossier ;

```
[root]# cp ../*.rpm /MonDepot/
```

- Créer la structure et générer le dépôt à l'aide de la commande **createrepo** ;

```
[root]# createrepo /MonDepot
```

- Configurer les fichiers **.repo** des clients afin qu'ils puissent installer les paquets depuis ce serveur (réinitialiser le cache des clients si besoin avec **yum clean all**).

13.5. Le dépôt EPEL

Le dépôt **EPEL** (**E**xtra **P**ackages for **E**nterprise **L**inux) est un dépôt contenant des paquets logiciels supplémentaires pour Entreprise Linux, ce qui inclut RedHat Entreprise Linux (RHEL), CentOS, etc.

Installation

Télécharger et installer le rpm du dépôt :

Si vous êtes derrière le proxy internet de l'école

```
[root]# export http_proxy=http://10.10.10.7:8080
```

- Pour une CentOS 6 :

```
[root]# rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-
8.noarch.rpm
```

Après avoir installé le paquet RPM du dépôt :

```
[root]# yum update
```

Partie 2 : Securité.

La sécurité sous LINUX.

Dans cette partie de notre support Formatux, nous ferons un tour d'horizon des aspects sécuritaire d'une distribution Linux :

- La gestion de l'endossement des droits su/sudo,
- Les modules d'authentification avec PAM,
- La sécurité avec SELinux,
- La gestion du firewall avec IPTables et Fail2Ban,
- La sécurisation du service d'administration à distance OpenSSH,
- La gestion d'une autorité de certification.

Bonne lecture.

Chapitre 1. Elévation des privilèges (su et sudo)

Le compte du super-utilisateur (root) possède de nombreuses contraintes car :

- il a tous les droits ;
- il peut causer des dommages irréparables ;
- il est la cible privilégiée des intrus.

L'administrateur qui travaille sur le serveur, se connecte avec son compte (qui dispose de droits restreints), puis au moment où il doit réaliser une tâche d'administration, endosse l'identité de **root** de façon temporaire, exécute ses actions, puis reprend son identité d'origine.

Il convient donc d'effectuer les opération suivantes :

- interdire le login root ;
- restreindre l'accès à la commande su ;
- privilégier la commande sudo ;
- robustesse du mot de passe :
  - il doit être complexe ;
  - non issu du dictionnaire ;
  - comporter des minuscules, majuscules, lettres, chiffres et caractères spéciaux ;
- changer régulièrement (ne pas être conservé identique indéfiniment) ;
- sécurisation (ne doit pas être inscrit à proximité des postes ou communiqué à des personnes qui n'ont pas à le connaître).

Ces principes sont les premières règles de sécurité.

1.1. Limiter le compte root

Le fichier **/etc/securetty** contient la liste des terminaux accessibles par le login **root**.

Pour interdire un terminal au login **root**, il faut :

- soit mettre la ligne en commentaire ;
- soit supprimer la ligne.

En fonction des distributions, la syntaxe du fichier /etc/securetty peut changer.

Pour une distribution **CentOS**, le fichier est constitué d'une ligne **tty** et d'une ligne **vc**, indispensable pour la connexion d'un utilisateur.

1.2. La commande su

su signifie **Substitute User** ou **Switch User**. Elle permet d'endosser l'identité d'un autre utilisateur sans se déconnecter. Cette commande utilisée sans login permet par défaut de prendre l'identité de **root**.

Syntaxe de la commande su

```
su [-] [-c commande] [login]
```

Exemple :

```
[root]# su - alain
[albert]$ su -c "passwd alain"
```

Option	Description
-	Charge l'environnement complet de l'utilisateur.
-c commande	Exécute la commande sous l'identité de l'utilisateur.

Pour la sécurité, les accès par la commande **su** sont répertoriés dans le fichier journal **/var/log/secure**.

Les utilisateurs non **root** devront taper le mot de passe de la nouvelle identité.

Il y a création de couches successives. Pour passer d'un utilisateur à un autre, il faut d'abord taper la commande **exit** pour reprendre son identité puis la commande **su** pour prendre une autre identité.

Il est conseillé de restreindre l'accès à la commande su à certaines personnes définies explicitement. Pour cela, il faut donner les droits à un groupe particulier pour accéder à la commande. Toutes les personnes appartenant à ce groupe obtiendront les droits réservés de celui-ci.

Le principe de restriction mis en place consiste à autoriser seulement un groupe d'utilisateurs particulier à exécuter la commande su.

1.3. La commande sudo

La commande **sudo** permet d'exécuter une ou plusieurs commandes avec les privilèges de root. Il n'est pas nécessaire de connaître son mot de passe.

Syntaxe de la commande sudo

```
sudo [-options] commande
```

Cet outil représente une évolution de la commande su.

Exemple :

```
[alain]$ sudo /sbin/route
Mot de passe : ********
Table de routage IP du noyau
Destination   Passerelle  Genmask    Indic Metric Ref Use Iface ·····
```

Option	Description
-E	Garde l'environnement du compte appelant.
-u	Désigne le compte qui exécutera la commande.

L'environnement est constitué des paramètres de l'utilisateur comme son répertoire de connexion, ses alias ...

Les accès possibles à la commande et les possibilités offertes aux utilisateurs sont configurables par l'administrateur qui leur affecte des droits explicites.

Vérification de l'existence du paquet :

```
[root] # rpm –qa "sudo*"
sudo-1.8.6p3-12.el6.i686
```

Avantages

Par rapport à l'utilisation des restrictions des groupes :

- Le contrôle d'accès aux commandes est centralisé dans le fichier **/etc/sudoers** ;
- Tous les accès et tentatives à partir de sudo sont journalisés dans le fichier **/var/log/secure**;
- L'utilisateur doit s'identifier à sudo en donnant son mot de passe.

Le fichier **/etc/sudoers** contient pour chacun des utilisateurs :

- son login ;
- sur quel ordinateur il est autorisé ;
- quelles commandes il peut exécuter.

1.4. Commande visudo

La commande visudo permet d'éditer le fichier /etc/sudoers (qui est en lecture seule) afin de configurer efficacement l'accès à la commande sudo pour les utilisateurs.

Syntaxe de la commande visudo

```
visudo [-c] [-f sudoers]
```

Exemple :

```
[root]# visudo
```

Option	Description
-c	Vérifie la syntaxe du fichier.
-f file	Désigne le fichier de configuration de sudo.

Le fichier /etc/sudoers

```
# This file must be edited with the
# 'visudo' command.
#
# Host alias specification
# User alias specification
# Cmnd alias specification
# User privilege specification
root ALL=(ALL) ALL
%wheel ALL=(ALL) ALL
```

Le groupe wheel

Le mot **wheel** fait référence à un groupe système qui dispose de privilèges lui permettant l'exécution de commandes à accès restreint.

Le groupe **wheel** est présent par défaut sur les distributions RHEL/CentOS.

Ne plus utiliser root

1 - Autoriser les utilisateurs du groupe **wheel** à lancer toutes les commandes (via visudo) en supprimant le commentaire de la ligne suivante :

```
%wheel    ALL=(ALL)    ALL
```

2 - Ajouter le compte utilisateur **bob** dans le groupe **wheel** :

```
[root]# usermod –aG wheel bob
```

3 - L'utilisateur **bob** dispose maintenant des pleins pouvoirs :

```
[bob]$ sudo chown root:root /tmp/test
[sudo] password for bob:
```

Il n'existe plus de raisons valables d'utiliser le compte root.

Le mot de passe **root** peut être séquestré (Keepass !) et son accès via ssh verrouillé.

Restreindre le sudo

Il est possible de ne pas donner tous les droits à un utilisateur mais de limiter ses accès et les commandes qu'il peut lancer. Par exemple, n'offrir à un compte de supervision que le droit de relancer le serveur (reboot) et uniquement celui-là.

Les accès à l'utilisation de sudo sont enregistrés dans les logs ce qui offre une traçabilité des actions entreprises.

Les restrictions se gèrent dans les différentes sections du fichier /etc/sudoers par l'intermédiaire de la commande visudo.

Section Host aliases

Définir des groupes de machines ou réseaux.

```
Host_Alias HOSTS-GROUP = host1 [,host2 ...]
```

Exemple :

```
Host_Alias FILESERVERS = 192.168.1.1, SF1, SF2
```

Cette section est utilisée pour créer des groupes de ressources réseaux autorisés à utiliser sudo.

Section User aliases

Définir des alias pour les utilisateurs.

```
User_Alias USER-GROUP = alain [,philippe, ...]
```

Exemple :

```
User_Alias ADMINS = root, AdminSF, adminPrinters
```

Cette section est utilisée essentiellement pour réunir sous un alias unique plusieurs utilisateurs ayant les mêmes besoins de la commande sudo.

Section Command aliases

Définir des alias pour les commandes.

```
Cmnd_Alias CDE-GROUP = cde1 [,cde2, ...]
```

Exemple :

```
Cmnd_Alias SOFTWARE = /bin/rpm, /usr/bin/yum
Cmnd_Alias SERVICES = /sbin/service, /sbin/chkconfig
```

Cette section est utilisée pour réunir plusieurs commandes Linux dans un groupe de commandes sudo.

Il faut alors créer des groupes cohérents.

Section User Specification

Lier les utilisateurs aux commandes.

```
USER-GROUP HOSTS-GROUP = [(cpte-cible)] CDE-GROUP
```

Exemple :

```
root       ALL=(ALL)          ALL
AdminSF    FILESERVERS=       SOFTWARE
```

Cette section définit qui a le droit d'utiliser des commandes particulières à partir de postes particuliers.

Il est possible de préciser qui exécute la commande (compte cible).

Exemple 1 :

Grâce au fichier **/etc/sudoers** ci-dessous, les utilisateurs alain, patrick et philippe peuvent désormais exécuter les commandes **ping** et **route** et effectuer des transferts FTP comme s'ils étaient

root.

```
# Host alias specification
  Host_Alias STA = 192.168.1.1, ma.machine
# User alias specification
  User_Alias CPTUSER = alain, patrick, philippe
# Cmnd alias specification
  Cmnd_Alias NET = /bin/ping, /sbin/route, /usr/bin/ftp
# User privilege specification
  root ALL=(ALL) ALL
  %wheel ALL=(ALL) ALL
  CPTUSER STA=(root) NET
```

Exemple 2 :

```
# Host alias specification
  Host_Alias MACHINE = station1
# User alias specification
  User_Alias ADMIN = adminunix
  User_Alias UTILISAT = alain, philippe
# Cmnd alias specification
  Cmnd_Alias SHUTDOWN = /sbin/shutdown
  Cmnd_Alias NET = /usr/bin/ftp
# User privilege specification
  root ALL=(ALL) ALL
  %wheel ALL=(ALL) ALL
  ADMIN MACHINE=NOPASSWD:ALL
  UTILISAT MACHINE=(root) SHUTDOWN, NET
```

Explications de l'exemple 2 :

- Création d'un alias pour la station :

```
Host_Alias MACHINE=station1
```

- Création de deux alias pour deux types d'utilisateurs (adminunix étant un équivalent « root »).

```
User_Alias ADMIN = adminunix
User_Alias UTILISAT = alain, philippe
```

- Création de deux alias de commandes qui regroupent les commandes exécutables.

```
Cmnd_Alias SHUTDOWN = /sbin/shutdown
Cmnd_Alias NET = /usr/bin/ftp
```

- L'utilisateur « root » peut exécuter toutes les commandes sur toutes les machines

```
root ALL=(ALL) ALL
```

Les utilisateurs qui font partie de l'alias ADMIN peuvent exécuter toutes les commandes, sur toutes les machines faisant partie de l'alias MACHINE et ce sans entrer de mot de passe (NOPASSWD:).

```
ADMIN MACHINE=NOPASSWD:ALL
```

Les utilisateurs qui font partie de l'alias UTILISAT peuvent exécuter la commande /sbin/shutdown sur toutes les machines faisant partie de l'alias MACHINE. Ils doivent entrer leur mot de passe.

```
UTILISAT MACHINE = SHUTDOWN, NET
```

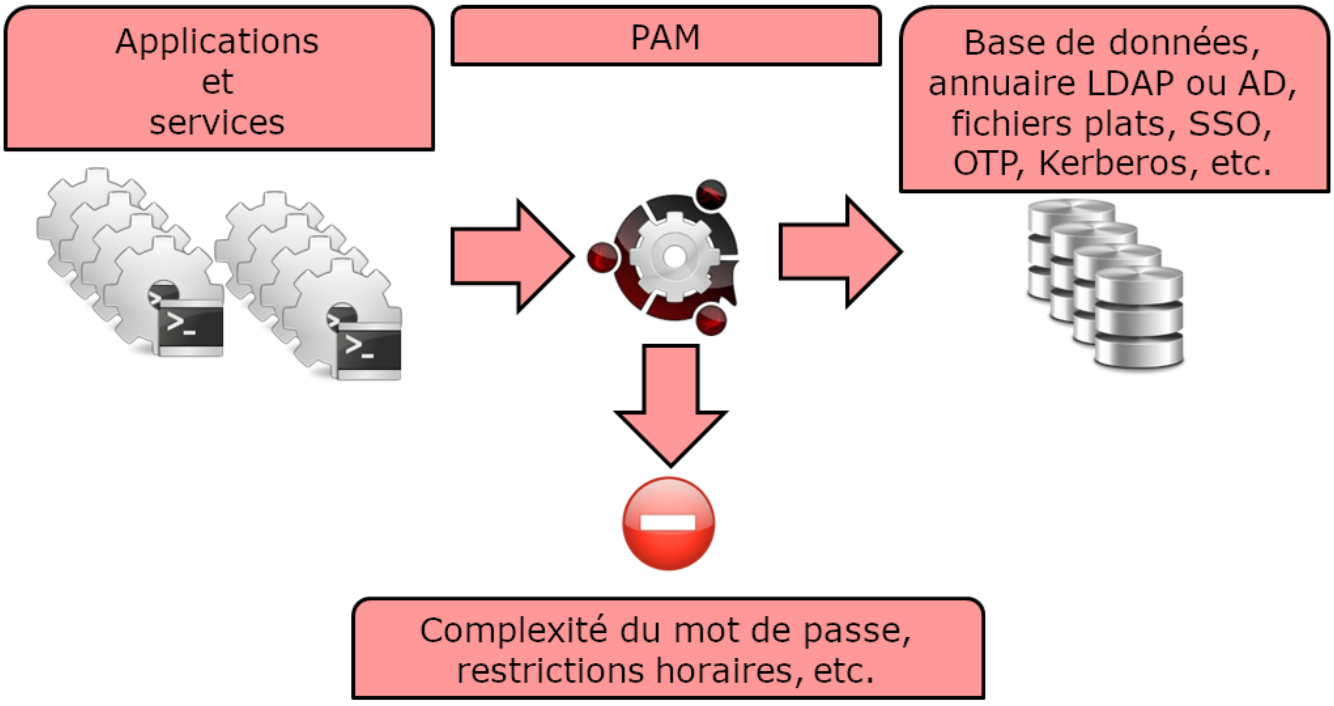
Chapitre 2. Les modules d'authentification PAM

PAM (**Pluggable Authentication Modules**) est le système sous GNU/Linux qui permet à de nombreuses applications ou services d'**authentifier** les utilisateurs de manière **centralisée**.

> PAM est un mécanisme permettant d'intégrer différents schémas d'authentification de bas niveau dans une API de haut niveau, permettant de ce fait de rendre indépendants du schéma les logiciels réclamant une authentification.
>
> — Définition de PAM par Wikipedia

2.1. Généralités

L'authentification est la phase durant laquelle il est vérifié que vous êtes bien la personne que vous prétendez être. Il existe d'ailleurs d'autres formes d'authentification que l'utilisation des mots de passe.



La mise en place d'une nouvelle méthode d'authentification ne doit pas nécessiter de modifications dans la configuration ou dans le code source d'un programme ou d'un service.

C'est pourquoi les applications s'appuient sur PAM, qui va leur fournir les primitives nécessaires à l'authentification de leurs utilisateurs.

L'ensemble des applications d'un système peuvent ainsi mettre en œuvre en toute transparence des

fonctionnalités complexes comme le **SSO** (Single Signe On), l'**OTP** (One Time Password) ou **Kerberos** de manière totalement transparente.

L'administrateur d'un système peut choisir exactement sa politique d'authentification pour une seule application (par exemple pour durcir le service SSH) indépendamment de celle-ci.

À chaque application ou service prenant en charge PAM correspondra un fichier de configuration dans le répertoire «/etc/pam.d». Par exemple, le processus «login» attribue le nom «/etc/pam.d/login» à son fichier de configuration.

Une mauvaise configuration de PAM peut compromettre toute la sécurité de votre système.

PAM est un système d'authentification (gestion des mots de passe). Si PAM est vulnérable alors l'ensemble du système est vulnérable.

Syntaxe d'une directive

Une directive permet de paramétrer une application pour PAM.

Syntaxe d'une directive

```
mécanisme [contrôle] chemin-module [argument]
```

Par exemple :

Le fichier /etc/pam.d/sudo

```
#%PAM-1.0
auth            include     system-auth
account         include     system-auth
password        include     system-auth
session         optional    pam_keyinit.so revoke
session         required    pam_limits.so
```

Une **directive** (une ligne complète) est composée d'un **mécanisme** (auth, account, password ou session), d'un **contrôle de réussite** (include, optional, required, ...), du chemin d'accès au module et éventuellement d'arguments (comme revoke par exemple).

L'ordre des modules est très important !

Chaque fichier de configuration PAM comprend un ensemble de directives. Les directives des interfaces de modules peuvent être empilées ou placées les unes sur les autres.

De fait, l'ordre dans lequel les modules sont répertoriés est très important au niveau du processus d'authentification.

2.2. Les mécanismes

Le mécanisme auth - Authentification

Concerne l'authentification du demandeur et établit les droits du compte :

- Authentifie généralement avec un mot de passe en le comparant à une valeur stockée en base de données ou en s'appuyant sur un serveur d'authentification,
- Établit les paramètres du comptes : uid, gid, groupes et limites de ressources.

Le mécanisme account - Gestion de compte

Vérifier que le compte demandé est disponible :

- Concerne la disponibilité du compte pour des raisons autres que l'authentification (par exemple pour des restrictions d'horaire).

Le mécanisme session - Gestion de session

Concerne la mise en place et la terminaison de la session :

- Accomplir les tâches associées à la mise en place d'une session (par exemple enregistrement dans les logs),
- Accomplir les tâches associées à la terminaison d'une session.

Le mécanisme password - Gestion des mots de passe

Utilisé pour modifier le jeton d'authentification associé à un compte (expiration ou changement) :

- Modifie le jeton d'authentification et vérifie éventuellement qu'il est assez robuste ou qu'il n'a pas déjà été utilisé.

2.3. Les indicateurs de contrôle

Les **mécanismes PAM (auth**, **account**, **session** et **password)** indiquent la réussite ou l'échec. Les **indicateurs de contrôle** (**required**, **requisite**, **sufficient**, **optional**) indiquent à PAM comment traiter ce résultat.

L'indicateur de contrôle required

La réussite de tous les modules **required** est nécessaire.

- Si le module réussit :

Le reste de la chaîne est exécuté. La requête est autorisée sauf si d'autres modules échouent.

- Si le module échoue :

Le reste de la chaîne est exécuté. Au final la requête est rejetée.

Le module doit être vérifié avec succès pour que l'authentification puisse se poursuivre. Si la vérification d'un module portant l'indication required échoue, l'utilisateur n'en est pas averti tant que tous les modules associés à cette interface n'ont pas été vérifiés.

L'indicateur de contrôle requisite

La réussite de tous les modules **requisite** est nécessaire.

- Si le module réussit :

Le reste de la chaîne est exécuté. La requête est autorisée sauf si d'autres modules échouent.

- Si le module échoue :

La requête est immédiatement rejetée.

Le module doit être vérifié avec succès pour que l'authentification puisse se poursuivre. Cependant, si la vérification d'un module requisite échoue, l'utilisateur en est averti immédiatement par le biais d'un message lui indiquant l'échec du premier module required ou requisite.

L'indicateur de contrôle sufficient

La réussite d'un seul module **sufficient** est suffisant.

- Si le module réussit :

La requête est immédiatement autorisée si aucun des précédents modules n'a échoué.

- Si le module échoue :

Le module est ignoré. Le reste de la chaîne est exécutée.

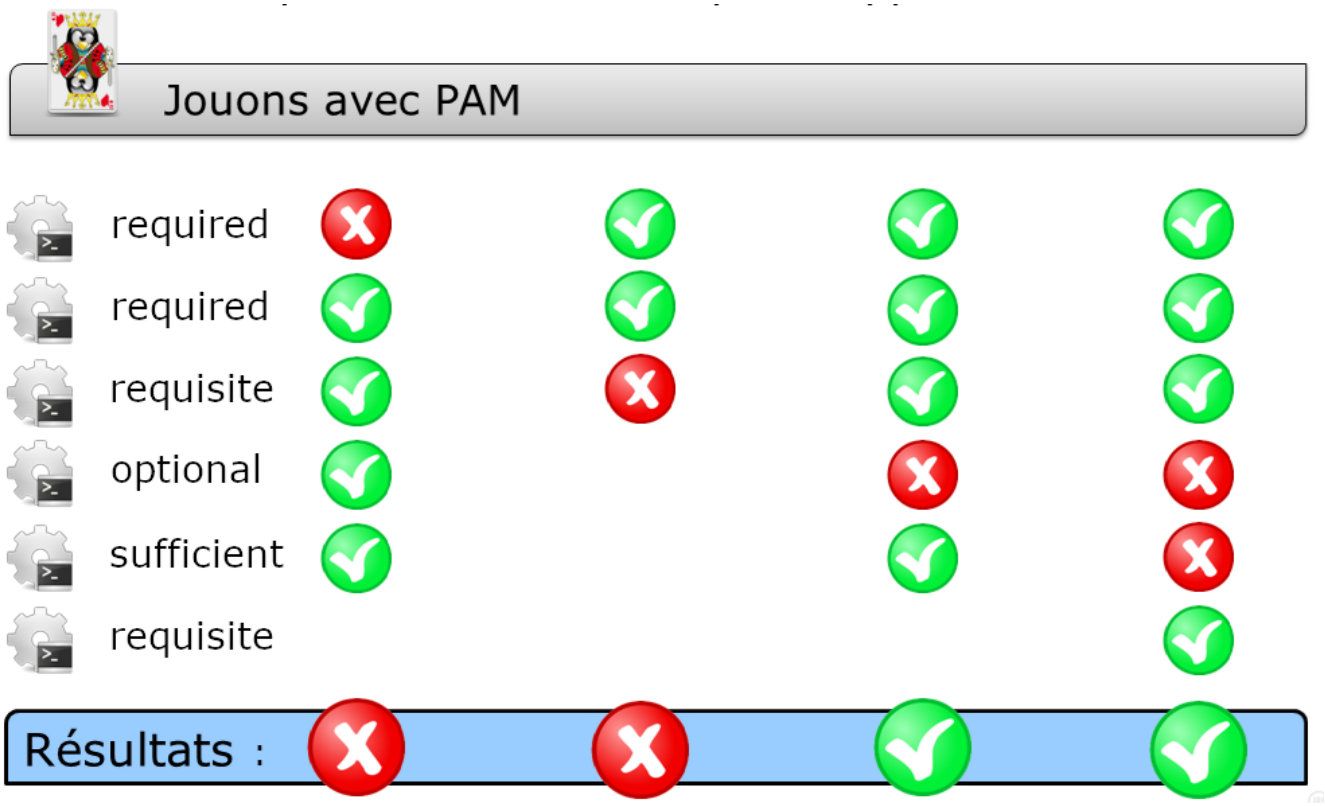
En cas d'échec, les vérifications de modules sont ignorées. Toutefois, si la vérification d'un module portant l'indication sufficient est réussie et qu'aucun module précédent portant l'indicateur required ou requisite n'a échoué, aucun autre module de ce type n'est nécessaire et l'utilisateur sera authentifié auprès du service.

L'indicateur de contrôle optional

Le module est exécuté mais le résultat de la requête est ignoré.

Si tous les modules de la chaînes étaient marqués optional, toutes les requêtes seraient toujours acceptées.

En conclusion



2.4. Les modules de PAM

Il existe de nombreux **modules** pour PAM. Voici les plus fréquents :

- pam_unix
- pam_ldap
- pam_wheel
- pam_cracklib
- pam_console
- pam_tally
- pam_securetty
- pam_nologin
- pam_limits
- pam_time
- pam_access

Le module pam_unix

Le module **pam_unix** permet de gérer la politique globale d'authentification.

Fichier /etc/pam.d/system-auth

```
password sufficient pam_unix.so sha512 nullok
```

Des arguments sont possibles pour ce module :

- **nullok** : dans le mécanisme auth autorise un mot de passe de connexion vide.
- **sha512** : dans le mécanisme password, définit l'algorithme de cryptage.
- **debug** : pour transmettre les informations à "syslog".
- **remember=n** : pour se souvenir des n derniers mots de passe utilisés (fonctionne conjointement avec le fichier "/etc/security/opasswd", qui est à créer par l'administrateur).

Le module pam_cracklib

Le module **pam_cracklib** permet de tester les mots de passe.

Fichier /etc/pam.d/password-auth

```
password sufficient pam_cracklib.so retry=2
```

Ce module utilise la bibliothèque **cracklib** pour vérifier la solidité d'un nouveau mot de passe. Il peut également vérifier que le nouveau mot de passe n'est pas construit à partir de l'ancien. Il ne concerne que le mécanisme password.

Par défaut ce module vérifie les aspects suivants et rejette si tel est le cas :

- le nouveau mot de passe est-il issu du dictionnaire ?
- le nouveau mot de passe est-il un palindrome de l'ancien (ex : azerty <> ytreza) ?
- seule la casse de(s) caractère(s) varie (ex : azerty <>AzErTy) ?

Des arguments possibles pour ce module :

- **retry=n** : impose n demandes (1 par défaut) du nouveau mot de passe.
- **difok=n** : impose au minimum n caractères (10 par défaut) différents de l'ancien mot de passe. De plus si la moitié des caractères du nouveau diffèrent de l'ancien, le nouveau mot de passe est validé.
- **minlen=n** : impose un mot de passe de n+1 caractères minimum non pris en compte en dessous de 6 caractères (module compilé comme tel !).

Autres arguments possibles :

- **dcredit=-n** : impose un mot de passe contenant au moins n chiffres,
- **ucredit=-n** : impose un mot de passe contenant au moins n majuscules,
- **credit=-n** : impose un mot de passe contenant au moins n minuscules,

- **ocredit=-n** : impose un mot de passe contenant au moins n caractères spéciaux.

Le module pam_tally

Le module **pam_tally** permet de verrouiller un compte en fonction d'un nombre de tentatives infructueuses de connexion.

Fichier /etc/pam.d/system-auth

```
auth required /lib/security/pam_tally.so onerr=fail no_magic_root
account required /lib/security/pam_tally.so deny=3 reset no_magic_root
```

- Le mécanisme **account** incrémente le compteur.
- Le mécanisme **auth** accepte ou refuse l'authentification et réinitialise le compteur.

Quelques arguments du module pam_tally sont intéressants à utiliser :

- **onerr=fail** : incrémentation du compteur,
- **deny=n** : une fois le nombre n d'essais infructueux dépassé, le compte est verrouillé,
- **no_magic_root** : inclus ou non les démons gérés par root (éviter le verrouillage de root),
- **reset** : remet le compteur à 0 si l'authentification est validée,
- **lock_time=nsec** : le compte est verrouillé pour n secondes.

Ce module fonctionne conjointement avec le fichier par défaut des essais infructueux /var/log/faillog (qui peut être remplacé par un autre fichier avec l'argument file=xxxx)et la commande associée faillog.

Syntaxe de la commande faillog

```
faillog[-m n] |-u login][-r]
```

Options :

- m : pour définir, dans l'affichage de la commande, le nombre maximum d'essais infructueux,
- u : pour spécifier un utilisateur,
- r : déverrouiller un utilisateur.

Le module pam_time

Le module **pam_time** permet de limiter les horaires d'accès à des services gérés par PAM.

Fichier /etc/pam.d/system-auth

```
account required /lib/security/pam_time.so
```

La configuration se fait via le fichier /etc/security/time.conf.

Fichier /etc/security/time.conf

```
login ; * ; users ;MoTuWeThFr0800-2000
http  ; * ; users ;Al0000-2400
```

La syntaxe d'une directive est la suivante :

```
services ; ttys ; users ; times
```

Dans les définitions suivantes, la liste logique utilise :

- **&** : et logique,
- | : ou logique,
- **!** : négation = « tous sauf »,
- ***** : caractère « joker ».

Les colonnes correspondent à :

- **services** : liste logique de services gérés par PAM qui sont concernés,
- **ttys** : liste logique de périphériques concernés,
- **users** : liste logique d'utilisateurs gérés par la règle,
- **times** : liste logique de détermination de l'horaire (jour/plage) autorisé.

Comment gérer les créneaux horaires :

- **les jours** : Mo Tu We Th Fr Sa Su Wk (du L au V) Wd (S et D) Al (du L au D),
- **la plage** : HHMM- HHMM,
- **une répétition annule l'effet** : WkMo = toutes les jours de la semaine (L-V) moins le lundi (répétition).

Exemples :

- Bob, peut se connecter via un terminal tous les jours entre 07h00 et 09h00, sauf le mercredi :

```
login ; tty* ; bob ; alth0700-0900
```

- Pas d'ouvertures de sessions, terminal ou distantes, sauf root, tous les jours de la semaine entre 17h30 et 7h45 le lendemain :

```
login ; tty* | pts/* ; !root ; !wk1730-0745
```

Le module pam_nologin

Le module **pam_nologin** permet de désactiver tous les comptes sauf root :

Fichier /etc/pam.d/login :

```
auth required   pam_nologin.so
```

Si le fichier /etc/nologin existe alors seul root pourra se connecter.

Le module pam_wheel

Le module **pam_wheel** permet de limiter l'accès à la commande su aux membres du groupes wheel.

Fichier /etc/pam.d/su

```
auth    required    pam_wheel.so
```

L'argument **group=mon_group** limite l'usage de la commande su aux membres du groupe mon_groupe

Si le groupe mon_groupe est vide, alors la commande su n'est plus disponible sur le système, ce qui force l'utilisation de la commande sudo.

Le module pam_mount

Le module **pam_mount** permet de monter un volume pour une session utilisateur.

Fichier /etc/pam.d/system-auth

```
auth        optional    pam_mount.so
password    optional    pam_mount.so
session optional    pam_mount.so
```

Les points de montage sont configurés dans le fichier /etc/security/pam_mount.conf :

Fichier /etc/security/pam_mount.conf

```
<volume fstype="nfs" server="srv" path="/home/%(USER)" mountpoint="~" />
<volume user="bob" fstype="smbfs" server="filesrv" path="public" mountpoint="/public"
/>
```

Chapitre 3. Sécurisation SELinux

Avec l'arrivée du noyau en version 2.6, un nouveau système de sécurité a été introduit pour fournir un mécanisme de sécurité supportant les stratégies de sécurité de contrôle d'accès.

Ce système s'appelle SELinux (Security Enhanced Linux) et a été créé par la NSA (National Security Administration) pour implémenter dans les sous-systèmes du noyau Linux une architecture robuste de type Mandatory Access Control (MAC).

Si, tout au long de votre carrière, vous avez soit désactivé ou ignoré SELinux, ce chapitre sera pour vous une bonne introduction à ce système qui travaille dans l'ombre de Linux pour limiter les privilèges ou supprimer les risques liés à la compromission d'un programme ou d'un démon.

Avant de débuter, sachez que SELinux est essentiellement à destination des distributions RHEL, bien qu'il soit possible de le mettre en œuvre sur d'autres distributions comme Debian (mais bon courage !). Les distributions de la famille Debian intègrent généralement le système AppArmor, qui pour sa part, fonctionne relativement différemment de SELinux.

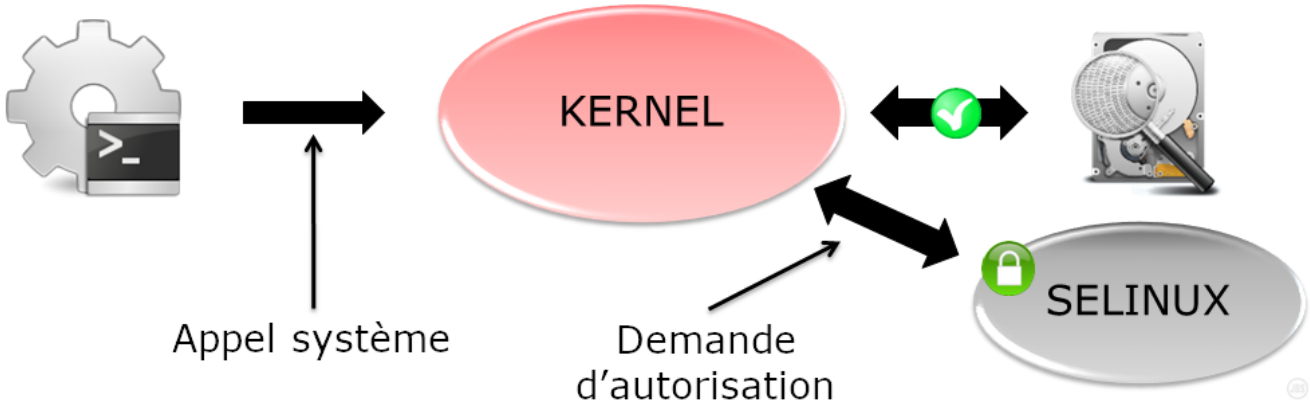
3.1. Généralités

SELinux (Security Enhanced Linux ou Amélioration de la Sécurité Linux) est un système de contrôle d'accès obligatoire (Mandatory Access Control).

Avant l'apparition des systèmes MAC, la sécurité standard de gestion d'accès reposait sur des systèmes DAC (Discretionary Access Control). Une application, ou un démon, fonctionnait avec des droits UID ou SUID (Set Owner User Id), ce qui permettait d'évaluer les permissions (sur les fichiers, les sockets, et autres processus...) en fonction de cet utilisateur. Ce fonctionnement ne permet pas de limiter suffisamment les droits d'un programme qui est corrompu, ce qui lui permet potentiellement d'accéder aux sous-systèmes du système d'exploitation.

Un système MAC renforce la séparation entre les informations sur la confidentialité et l'intégrité du système pour obtenir un système de confinement. Le système de confinement est indépendant du système de droits traditionnels et il n'existe pas de notion de superutilisateur.

À chaque appel système, le noyau interroge SELinux pour savoir s'il autorise l'action a être effectuée.



SELinux utilise pour cela un ensemble de règles (en anglais policy). Un ensemble de deux jeux de règles standards (targeted et strict) est fourni et chaque application fournit généralement ses propres règles.

Le contexte SELinux

Le fonctionnement de SELinux est totalement différent des droits traditionnels Unix.

Le contexte de sécurité SELinux est défini par le trio **identité+rôle+domaine**.

L'identité d'un utilisateur dépend directement de son compte linux. Une identité se voit attribué un ou plusieurs rôles, mais à chaque rôle correspond un domaine et un seul. C'est en fonction du domaine du contexte de sécurité (et donc du rôle...) que sont évalués les droits d'un utilisateur sur une ressource.

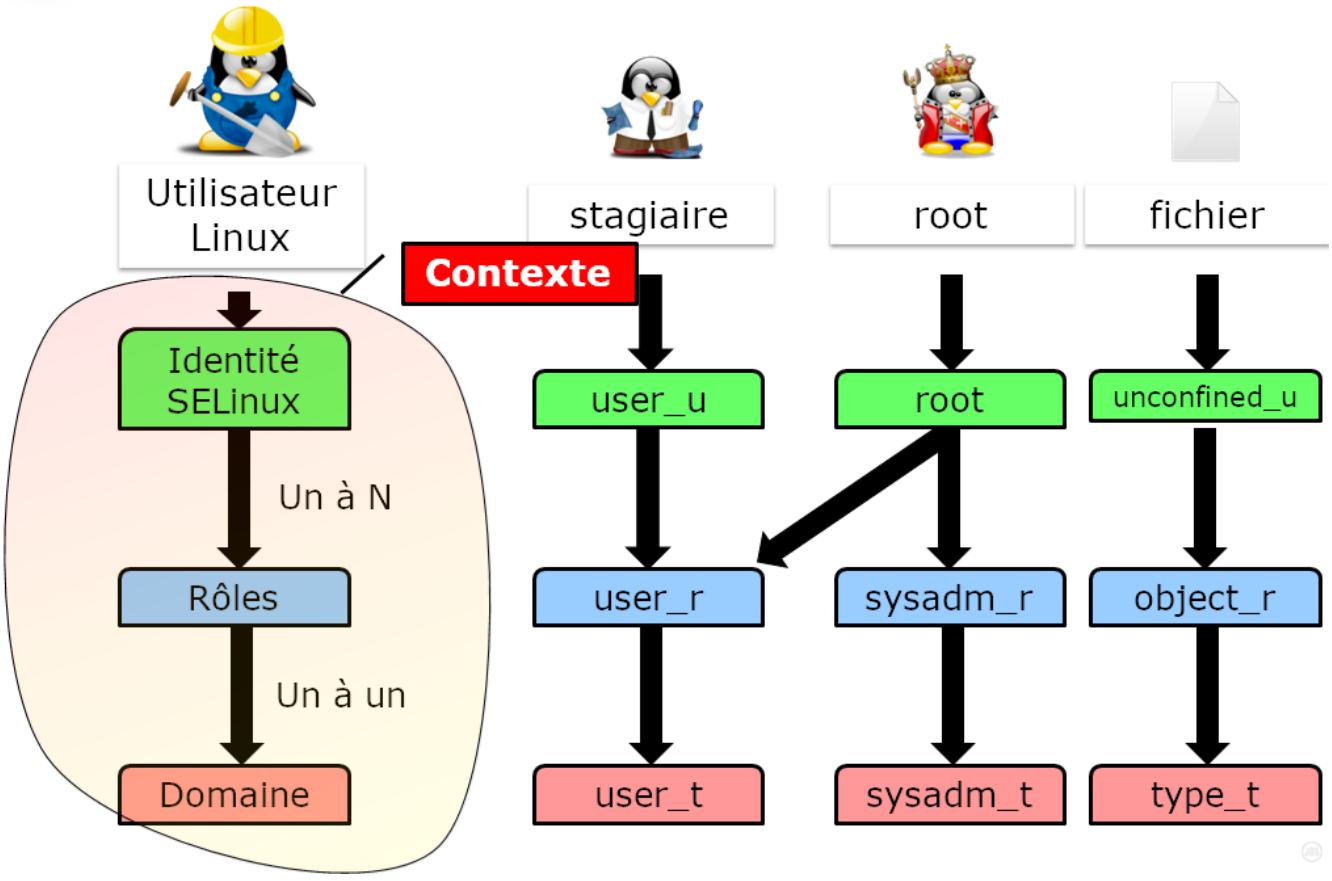


Figure 37. Le contexte SELinux

```
Les termes "domaine" et "type" sont similaires, typiquement "domaine" est utilisé
lorsque l'on se réfère à un processus tandis que "type" réfère à un objet.
La convention de nommage est :  _u user ; _r  rôle ; _t type.
```

Le contexte de sécurité est attribué à un utilisateur au moment de sa connexion, en fonction de ses rôles. Le contexte de sécurité d'un fichier est quant à lui défini par la commande **chcon** (change context) que nous verrons plus tard dans la suite de ce chapitre.

Considérez les pièces suivantes du puzzle SELinux :

- Les sujets
- Les objets
- Les stratégies
- Le mode

Quand un sujet (une application par exemple) tente d'accéder à un objet (un fichier par exemple), la partie SELinux du noyau Linux interroge sa base de données de stratégies. En fonction du mode de fonctionnement, SELinux autorise l'accès à l'objet en cas de succès, sinon il enregistre l'échec dans le fichier /var/log/messages.

Le contexte SELinux des processus standards

Les droits d'un processus dépendent de son contexte de sécurité.

Par défaut, le contexte de sécurité du processus est défini par le contexte de l'utilisateur (identité + rôle + domaine) qui le lance.

Un domaine étant un type (au sens SELinux) spécifique lié à un processus et hérité (normalement) de l'utilisateur qui l'a lancé, ses droits s'expriment en termes d'autorisation ou de refus sur des types (liés à des objets) :

Un processus dont le contexte a le domaine de sécurité D peut accéder aux objets de type T.

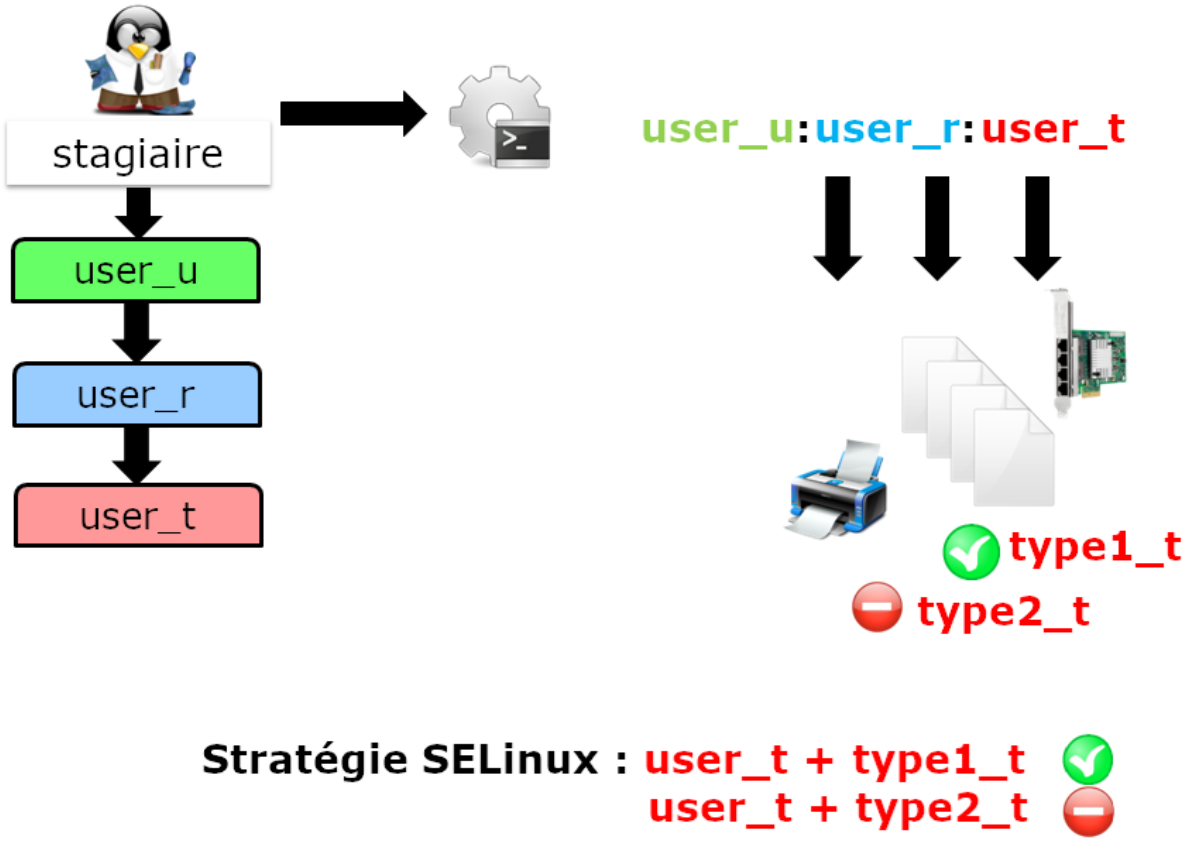


Figure 38. Le contexte SELinux d'un processus standard

Le contexte SELinux des processus importants

La plupart des programmes importants se voient attribuer un domaine dédié.

Chaque exécutable est étiqueté avec un type dédié (ici **sshd_exec_t**) qui fait basculer le processus associé automatiquement dans le contexte **sshd_t** (au lieu de **user_t**).

Ce mécanisme est essentiel puisqu'il permet de restreindre au plus juste les droits d'un processus.

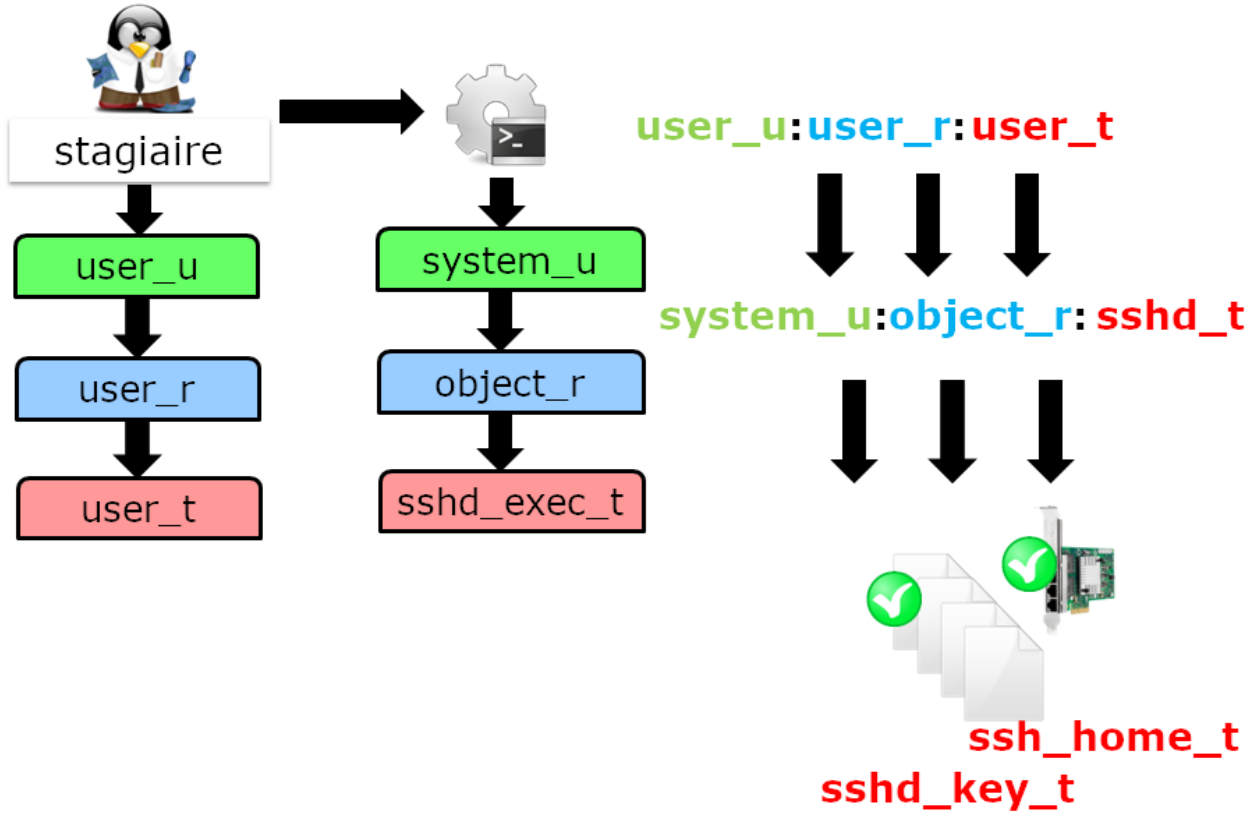


Figure 39. Le contexte SELinux d'un processus important - exemple de sshd

3.2. Gestion

La commande **semanage** (SE manage)permet d'administrer les règles SELinux.

Syntaxe de la commande semanage

```
semanage [type_d_objet] [options]
```

Exemple :

```
[root]# semanage boolean -l
```

Table 74. Options de la commande semanage

Options	Observations
-a	Ajoute un objet
-d	Supprime un objet
-m	Modifie un objet
-l	Liste les objets

La commande **semanage** n'est pas installée par défaut sous CentOS.

Sans connaître le paquet qui fournit cette commande, il convient de rechercher son nom avec la commande :

```
[root]# yum provides */semanage
```

puis l'installer :

```
[root]# yum install policycoreutils-python
```

Administrer les objets de type booléens

Les booléens permettent le confinement des processus.

Syntaxe de la commande semanage boolean

```
semanage boolean [options]
```

Pour lister les booléens disponibles :

```
[root]# semanage boolean –l
Booléen SELinux    State Default  Description
…
httpd_can_sendmail (fermé,fermé)  Allow http
  daemon to send mail
…
```

La commande **setsebool** permet de modifier l'état d'un objet de type booléen :

Syntaxe de la commande setsebool

```
setsebool [-PV] boolean on|off
```

Exemple :

```
[root]# setsebool -P httpd_can_sendmail on
```

Table 75. Options de la commande setsebool

Options	Observations
-P	Modifie la valeur par défaut au démarrage (sinon uniquement jusqu'au reboot)
-V	Supprime un objet

La commande semanage permet d'administrer les objets de type port :

Syntaxe de la commande semanage port

```
semanage port [options]
```

Exemple : autoriser le port 81 aux processus du domaine httpd

```
[root]# semanage port -a -t http_port_t -p tcp 81
```

3.3. Mode de fonctionnement

SELinux propose trois modes de fonctionnement :

- Enforcing (Appliqué)

Mode par défaut pour les Linux RedHat. Les accès seront restreints en fonction des règles en vigueur.

- Permissive (Permissif)

Les règles sont interrogées, les erreurs d'accès sont journalisées, mais l'accès ne sera pas bloqué.

- Disabled (Désactivé)

Rien ne sera restreint, rien ne sera journalisé.

Par défaut, la plupart des systèmes d'exploitation sont configurés avec SELinux en mode Enforcing.

La commande **getenforce** retourne le mode de fonctionnement en cours

Syntaxe de la commande getenforce

```
getenforce
```

Exemple :

```
[root]# getenforce
Enforcing
```

La commande **sestatus** retourne des informations sur SELinux

Syntaxe de la commande sestatus

```
sestatus
```

Exemple :

```
[root]# sestatus
SELinux status:         enabled
SELinuxfs mount:            /selinux
Current mode:           enforcing
Mode from config file : enforcing
Policy version:         24
Policy from config file:    targeted
```

La commande **setenforce** modifie le mode de fonctionnement en cours :

Syntaxe de la commande setenforce

```
setenforce 0|1
```

Passer SELinux en mode permissif :

```
[root]# setenforce 0
```

Le fichier /etc/sysconfig/selinux

Le fichier /etc/sysconfig/selinux permet de modifier le mode de fonctionnement de SELinux.

Désactiver SELinux se fait à vos risques et périls ! Il est préférable d'apprendre le fonctionnement de SELinux plutôt que de le désactiver systématiquement !

Modifier le fichier /etc/sysconfig/selinux

```
SELINUX=disabled
```

Redémarrer le système :

```
[root]# reboot
```

Attention au changement de mode SELinux !

En mode permissif ou désactivé, les nouveaux fichiers créés ne porteront aucune étiquette.

Pour réactiver SELinux, il faudra repositionner les étiquettes sur l'intégralité de votre système.

Labéliser le système entièrement :

```
[root]# touch /.autorelabel
[root]# reboot
```

3.4. Les jeux de règles (Policy Type)

SELinux fournit deux types de règles standards :

- Targeted : seuls les démons réseaux sont protégés (dhcpd, httpd, named, nscd, ntpd, portmap, snmpd, squid et syslogd)
- Strict : tous les démons sont protégés

3.5. Contexte

L'affichage des contextes de sécurité se fait avec l'option -Z. Elle est associée à de nombreuses commandes :

Exemples :

```
[root]# id -Z   # le contexte de l'utilisateur
[root]# ls -Z   # ceux des fichiers courants
[root]# ps -eZ  # ceux des processus
[root]# netstat –Z # ceux des connexions réseaux
[root]# lsof -Z # ceux des fichiers ouverts
```

La commande **matchpathcon** retourne le contexte d'un répertoire.

Syntaxe de la commande matchpathcon

```
matchpathcon répertoire
```

Exemple :

```
[root]# matchpathcon /root
 /root  system_u:object_r:admin_home_t:s0

[root]# matchpathcon /
 /      system_u:object_r:root_t:s0
```

La commande **chcon** modifie un contexte de sécurité :

Syntaxe de la commande chcon

```
chcon [-vR] [-u USER] [–r ROLE] [-t TYPE] fichier
```

Exemple :

```
[root]# chcon -vR -t httpd_sys_content_t /home/SiteWeb
```

Table 76. Options de la commande chcon

Options	Observations
-v	Passe en mode verbeux
-R	Applique la récursivité
-u,-r,-t	S'applique à un utilisateur, un rôle ou un type

La commande **restorecon** restaure le contexte de sécurité par défaut :

Syntaxe de la commande restorecon

```
restorecon [-vR] répertoire
```

Exemple :

```
[root]# restorecon -vR /home/SiteWeb
```

Table 77. Options de la commande restorecon

Options	Observations
-v	Passe en mode verbeux
-R	Applique la récursivité

La commande **audit2why** indique la cause d'un refus SELinux :

Syntaxe de la commande audit2why

```
audit2why [-vw]
```

Exemple :

```
[root]# less /var/log/audit/audit.log|grep AVC|grep denied|tail -1|audit2why
```

Table 78. Options de la commande audit2why

Options	Observations
-v	Passe en mode verbeux
-w	Traduit la cause d'un rejet par SELinux et propose une solution pour y remédier (option par défaut)

Aller plus loin avec SELinux

La commande **audit2allow** crée à partir d'une ligne d'un fichier "audit" un module pour autoriser une action SELinux :

Syntaxe de la commande audit2allow

```
audit2allow [-mM]
```

Exemple :

```
[root]# less /var/log/audit/audit.log|grep AVC|grep denied|tail -1|audit2allow -M
MonModule_Local
```

Table 79. Options de la commande audit2allow

Options	Observations
-m	Crée juste le module (*.te)
-M	Crée le module, le compile et le met en paquet (*.pp)

Exemple de configuration

Après l'exécution d'une commande, le système vous rend la main mais le résultat attendu n'est pas visible : aucun message d'erreur à l'écran.

- **Étape 1** : Lire le fichier journal sachant que le message qui nous intéresse est de type AVC (SELinux), refusé (denied) et le plus récent (donc le dernier).

```
[root]# less /var/log/audit/audit.log|grep AVC|grep denied|tail -1
```

Le message est correctement isolé mais ne nous est d'aucune aide.

- **Étape 2** : Lire le message isolé avec la commande audit2why pour obtenir un message plus explicite pouvant contenir la solution de notre problème (typiquement un booléen à positionner).

```
[root]# less /var/log/audit/audit.log|grep AVC|grep denied|tail -1|audit2why
```

Deux cas se présentent : soit nous pouvons placer un contexte ou renseigner un booléen, soit il faut passer à l'étape 3 pour créer notre propre contexte.

- **Étape 3** : Créer son propre module.

```
[root]# less /var/log/audit/audit.log|grep AVC|grep denied|tail -1|audit2allow -M
MonModule_Local
Generating type enforcement: MonModule_Local.te
Compiling policy: checkmodule -M -m -o MonModule_Local.mod MonModule_Local.te
Building package: semodule_package -o MonModule_Local.pp -m MonModule_Local.mod

[root]# semodule -i MonModule_Local.pp
```

Chapitre 4. IPtables, le pare-feu Linux

Gérer le trafic réseau est certainement la partie la plus difficile du métier d'administrateur système. Il faut impérativement configurer les pare-feu sur l'ensemble des éléments actifs du réseau, serveurs inclus, en prenant en compte les besoins des utilisateurs et des systèmes à la fois pour le trafic entrant et sortant, sans laisser des systèmes vulnérables à des attaques.

C'est le rôle du pare-feu **IPTables**, entièrement administrable en ligne de commandes.

IPtables utilise un jeu de tables qui contiennent des chaînes comprenant les règles du firewall.

Il existe 3 types de tables :

- La table **FILTER**, qui est la table par défaut. Elle est composée de 3 chaînes :
  - La chaîne **INPUT** : les paquets sont destinés à une socket locale.
  - La chaîne **FORWARD** : les paquets sont routés par le système.
  - La chaîne **OUTPUT** : les paquets sont générés en local.
- La table **NAT**, qui est consultée lorsque un paquet tente de créer une nouvelle connexion. Elle est composée de 3 chaînes :
  - **PREROUTING** : utilisée pour modifier des paquets dès leur réception par le système.
  - **OUTPUT** : utilisée pour modifier des paquets générés en local.
  - **POSTROUTING** : utilisée pour modifier des paquets au moment de leur sortie du système.
- La table **MANGLE** : cette table est utilisée pour modifier des paquets. Il y a 5 chaînes :
  - **PREROUTING** : pour modifier des connexions entrantes.
  - **OUTPUT** : pour modifier des paquets générés en local.
  - **INPUT** : pour modifier les paquets entrants.
  - **POSTROUTING** : pour modifier les paquets avant leur départ du système.
  - **FORWARD** : pour modifier les paquets qui sont routés par le système.

4.1. Gestion du pare-feu

Démarrer/arrêter ou redémarrer le parefeu

En fonction de la distribution, si elle utilise les scripts de démarrage SystemD (RHEL 7 et +) ou SysVinit, il faudra utiliser les lignes de commandes suivantes :

Distribution basée sur SystemD

```
[admin]$ sudo systemctl start iptables
[admin]$ sudo systemctl stop iptables
[admin]$ sudo systemctl restart iptables
```

Distribution basée sur SysVinit

```
[admin]$ sudo service iptables start
[admin]$ sudo service iptables stop
[admin]$ sudo service iptables restart
```

Afficher les règles

Pour afficher les règles IPTables :

```
[admin]$ sudo iptables -L -n -v
Chain INPUT (policy ACCEPT 927K packets, 219M bytes)
pkts bytes target prot opt in out source destination
...

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
...

Chain OUTPUT (policy ACCEPT 127 packets, 18M bytes)
pkts bytes target prot opt in out source destination
...
```

Il est possible de spécifier, avec l'option -t, quelle table afficher :

```
[admin]$ sudo iptables -t input -L -n -v
Chain INPUT (policy ACCEPT 927K packets, 219M bytes)
pkts bytes target prot opt in out source destination
...
```

4.2. Quelques exemples

Voici quelques règles qui mettent en oeuvre les fonctionnalités d'IPTables :

Bloquer des adresses IP spécifiques

Pour bloquer une adresse qui aurait un comportement abusif :

```
iptables -A INPUT -s xxx.xxx.xxx.xxx -j DROP
```

Changer xxx.xxx.xxx.xxx par l'adresse IP à bloquer.

pour la débloquer :

```
iptables -D INPUT -s xxx.xxx.xxx.xxx -j DROP
```

L'option **-D** ou **--delete** supprime la règle dans la chaîne spécifiée.

Bloquer/accepter des ports spécifiques

Pour bloquer un port spécifique en sortie :

```
iptables -A OUTPUT -p tcp --dport xxx -j DROP
```

alors que pour autoriser une connexion entrante :

```
iptables -A INPUT -p tcp --dport xxx -j ACCEPT
```

Pour bloquer le protocole udp, remplacer simplement tcp par udp.

Plusieurs ports peuvent être spécifiés en même temps :

```
iptables -A INPUT -p tcp -m multiport --dports 80,443 -j ACCEPT
```

Autoriser un réseau

Un réseau complet peut être spécifié, par exemple pour autoriser un accès SSH :

```
iptables -A OUTPUT -p tcp -d 10.10.10.0/24 --dport 22 -j ACCEPT
```

Limiter le nombre de connexion d'une adresse

Pour limiter le nombre de paquets par adresse (protection contre le flooding) :

```
iptables -A INPUT -p tcp --dport 80 -m limit --limit 50/minute --limit-burst 100 -j
ACCEPT
```

Pour limiter le nombre de connexions actives (ici 3 sur le port ssh) :

```
iptables -A INPUT -p tcp --syn --dport 22 -m connlimit --connlimit-above 3 -j REJECT
```

Bloquer le protocole ICMP (ping)

Bloquer le protocole ICMP peut être considéré comme une mesure de sécurité.

```
iptables -A INPUT -p icmp -i eth0 -j DROP
```

Accès à la loopback

L'accès à l'interface de loopback doit toujours être possible. Les lignes suivantes doivent impérativement être présentes :

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

Logger les paquets refusés

Iptables peut envoyer à syslog (/var/log/messages) les paquets qu'il refuse :

```
iptables -A INPUT -i eth0 -J LOG --log-prefix "REFUS IPTABLES : "
```

Cette règle est à mettre en toute dernière, juste avant la suppression des paquets.

Gérer les connexions établies

Il est nécessaire d'autoriser les paquets provenant de connexions déjà établies ou en relation avec d'autres connexions.

```
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

Supprimer les paquets invalides

Certains paquets sont marqués invalides à l'arrivée (duplication, dé-séquencement, etc.). Il est possible de les journaliser pour pouvoir éventuellement mener des actions correctrices, puis de les supprimer :

```
iptables -A INPUT -m conntrack --ctstate INVALID -J LOG --log-prefix "REFUS IPTABLES :
iptables -A INPUT -m conntrack --ctstate INVALID -j DROP
```

Bloquer le trafic SMTP sortant

Pour empêcher toute sortie de mails depuis un des ports correspondant aux ports SMTP :

```
iptables -A OUTPUT -p tcp --dports 25,465,587 -j REJECT
```

4.3. Conclusion

Ces exemples permettent de faire le tour des fonctionnalités offertes par le pare-feu iptables. Quelques règles permettent d'offrir un niveau de sécurité plus important.

La mise en place du pare-feu n'est donc pas un élément à négliger.

Chapitre 5. Fail2ban

Fail2ban permet le blocage des adresses IP tentant une intrusion sur votre serveur. Pour cela, il s'appuie sur le pare-feu Netfilter ou sur TCP Wrapper.

Il détecte les tentatives d'intrusion en parcourant les journaux du système.

5.1. Installation

Fail2ban est disponible dans le dépôt EPEL.p

```
yum install fail2ban
```

Le fichier **/etc/fail2ban/jail.conf** est fourni par le paquet **fail2ban**.

Il ne faut le modifier directement, sous peine de voir ses changements perdus lors de la prochaine mise à jour du paquet. Au lieu de cela, il faut créer un fichier **/etc/fail2ban/jail.local**, et y placer sa configuration personnalisée.

```
cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

5.2. Configuration

Options générales (section DEFAULT) :

- **bantime** : le temps en secondes durant lequel l'adresse IP ayant tentée une intrusion sera bannie ;
- **findtime** : la plage de temps en secondes utilisée pour analyser les logs. Plus cette plage est grande, plus l'analyse est longue ;
- **maxretry** : le nombre d'échecs de connexion tolérés avant le bannissement.

Il est primordiale d'ajuster au mieux ces trois valeurs. Un attaquant, avec la configuration par défaut, peut faire 5 tentatives toutes les 10 minutes, sans être banni. Cette valeur peut paraître ridiculement petite, mais elle représente par jour 5x6x24=720 tentatives, et 262 800 par an. Elle est encore à multiplier par le nombre de postes participant à l'attaque.

Même avec un système comme fail2ban, un poste n'est pas à l'abri des attaques. Fail2ban n'empêchera pas un attaquant de prendre possession de votre serveur, mais le retardera. Il est important de respecter les règles de bases de la sécurité informatique : changement fréquent de mot de passe, complexité, etc.

```
[root]# vim /etc/fail2ban/jail.local
[DEFAULT]
bantime  = 3600
findtime = 600
maxretry = 5

[ssh-iptables]
enabled  = true
filter   = sshd
action   = iptables[name=SSH, port=ssh, protocol=tcp] sendmail-whois[name=SSH, dest
=root, sender=fail2ban@formatux.fr]
logpath  = /var/log/secure
maxretry = 5
```

- section ssh-iptables :
  - enabled : active la règle
  - filter : fichier de log à analyser. Un chemin complet ou un raccourci (comme c'est le cas ici)
  - action : que dois faire fail2ban en cas d'échec de connexion ?
    - iptables : activer une règle dans le pare-feu,
    - sendmail-whois : envoyer un mail de rapport.

5.3. Lancement du service

- Fail2ban s'appuyant sur le firewall netfilter pour bannir les adresses IP tentant une intrusion, il faut s'assurer que celui-ci soit démarré :

```
service iptables status
service ip6tables status
```

- Si le pare-feu n'est pas actif sur le système :

```
chkconfig iptables on
chkconfig ip6tables on
service iptables start
service ip6tables start
```

- Démarrer le service fail2ban :

```
chkconfig fail2ban on
service fail2ban start
```

5.4. Vérification du service

La commande **fail2ban-client status** permet d'obtenir des informations sur les services surveillés ainsi que le nombre de règles iptables mises en place :

```
[root]# fail2ban-client status
Status
|- Number of jail:  2
`- Jail list:       mysqld-iptables, ssh-iptables
```

IPtables doit également renvoyer des informations concernant l'utilisation d'une chaîne fail2ban-SSH :

```
[root]# iptables --list

Chain INPUT (policy ACCEPT)
target     prot opt source               destination
fail2ban-SSH  tcp  --  anywhere             anywhere            tcp dpt:ssh
ACCEPT     all  --  anywhere             anywhere            state RELATED,ESTABLISHED
...

Chain fail2ban-SSH (1 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere
```

5.5. Interface graphique

Fail2Web est une interface graphique web pour Fail2Ban, qui communique via **Fail2Rest** (service REST).

Attention à la configuration du serveur Fail2Rest, qui par défaut ne propose pas d'authentification, il faudra le faire via l'authentification apache.

Chapitre 6. Sécuriser le serveur SSH

Le serveur openSSH permet l'administration d'un serveur à distance.

6.1. Configuration

La configuration du serveur SSH se fait dans le fichier **/etc/ssh/sshd_config**.

Á chaque modification, il faut relancer le service :

```
service sshd restart
```

6.2. Changer le port d'écoute et la version du protocole

Il est préférable de changer le port par défaut (22) par un port connu de vous seul et de n'utiliser que la dernière version du protocole :

```
Port XXXX
Protocol 2
```

6.3. Utilisation de clefs privées/publiques

Dans la mesure du possible, utilisez un couple de clef privée/publique pour l'accès au serveur, et désactivez les autres possibilités de connexion (authentification par utilisateur + mot de passe) :

```
PasswordAuthentication no
RSAAuthentication yes
PubkeyAuthentication yes
```

6.4. Limiter les accès

Il est possible de limiter les accès directement dans la configuration du service avec la directive AllowUsers :

```
AllowUsers antoine
```

Il est également possible de limiter les accès par adresse IP via TCP Wrapper. Par exemple, refusez tous les accès dans le fichier /etc/hosts.deny :

```
sshd: ALL
```

et n'acceptez dans le fichier /etc/hosts.allow que les connexions depuis des adresses IP validées :

```
sshd: 192.168.1. 221.10.140.10
```

Voir la configuration de TCP Wrapper pour plus d'informations.

6.5. Interdire l'accès à root !!!

La mesure essentielle à prendre est d'interdire l'accès direct à root au serveur ssh :

```
PermitRootLogin no
```

et d'utiliser les possibilités offertes par le fichier sudoers pour permettre aux utilisateurs administrateurs de lancer des commandes d'administration.

6.6. Sécurité par le parefeu

Il est également important de limiter les accès aux services grâce au pare-feu et de bannir les adresses IP tentant des attaques par dictionnaire.

Chapitre 7. Autorité de certification TLS avec easy-rsa

SSL (Secure Socket Layer) est le protocole historique développé par la société Netscape pour sécuriser les échanges entre les clients et les serveurs Web. SSL a été **standardisé par l'IETF** sous le nom de **TLS** (Transport Layer Security). TLS n'est rien d'autre que SSLv3 avec quelques corrections et améliorations.

Une autorité de certification (**CA, Certificate Authority**) agit comme une entité disposant d'une biclé (couple de clé privée/publique) de confiance représentant un "certificat racine". Ce certificat va seulement être employé pour signer d'autres certificats après avoir vérifié l'identité qui y est inscrite. Tout client voulant utiliser un service s'appuyant sur TLS devra disposer du certificat de sa CA pour valider l'authenticité du certificat TLS du serveur.

7.1. Installer easy-rsa

Easy-rsa est disponible dans le dépôt EPEL.

```
[root]# yum install easy-rsa
```

7.2. Configuration

- Se déplacer dans le répertoire easy-rsa :

```
[root]# cd /usr/share/easy-rsa/2.0/
```

- Configurer les réponses par défaut :

```
[root]# vim vars
...
export KEY_COUNTRY="FR"
export KEY_PROVINCE="fr"
export KEY_CITY="Rennes"
export KEY_ORG="Formatux"
export KEY_EMAIL="admin@formatux.fr"
export KEY_OU="formatux.fr"

# X509 Subject Field
export KEY_NAME="Formatux"
```

7.3. Créer une autorité de certification

- Se déplacer dans le répertoire easy-rsa :

```
[root]# cd /usr/share/easy-rsa/2.0/
[root]# source ./vars
[root]# ./clean-all
[root]# ./build-ca
Generating a 2048 bit RSA private key
..........................+++
.....................................................................+++
writing new private key to 'ca.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [FR]:
State or Province Name (full name) [fr]:
Locality Name (eg, city) [Rennes]:
Organization Name (eg, company) [Formatux]:
Organizational Unit Name (eg, section) [formatux.fr]:
Common Name (eg, your name or your server's hostname) [Formatux CA]:
Name [Formatux]:
Email Address [admin@formatux.fr]:

[root]# ./build-dh
```

7.4. Créer une biclé serveur

```
[root]# cd /usr/share/easy-rsa/2.0/
[root]# source ./vars
[root]# ./build-key-server servername
```

7.5. Installer le certificat de l'autorité de certification

- Installer le package ca-certificates

```
[root]# yum install ca-certificates
```

- Activer la configuration dynamique de l'autorité de certification :

```
[root]# update-ca-trust enable
```

- Ajouter le certificat de l'autorité :

```
[root]# cp foo.crt /etc/pki/ca-trust/source/anchors/
[root]# update-ca-trust extract
```

Partie 3 : Services.

Les services sous LINUX.

Dans cette partie de notre support Formatux, nous vous proposons des supports pour les services les plus courants que vous pourriez être amenés à mettre en oeuvre :

- Bind, le service DNS,
- Samba, le service de partage de fichier compatible Windows,
- Apache, le serveur Web,
- Postfix, le service de messagerie,
- OpenLDAP, le serveur d'annuaire,
- etc.

Bonne lecture.

Chapitre 1. Network File System

NFS (Network File System) est un système de partage de fichiers via montage réseau.

1.1. Généralités

NFS est basé sur un fonctionnement client-serveur : le serveur met à disposition des ressources pour tout ou partie du réseau (clients).

Le dialogue entre les clients et le serveur se fait grâce aux services RPC (Remote Procedure Call ou procédure d'appel à distance).

Les fichiers distants sont montés dans un répertoire et apparaissent comme un système de fichiers local. Les utilisateurs clients accèdent en toute transparence aux fichiers partagés par le serveur, en parcourant les répertoires comme s'ils étaient locaux.

1.2. Installation

2 services sont nécessaires au fonctionnement de NFS :

- Le service network ;
- Le service rpcbind.

L'état des services peut être visualisé par les commandes :

```
service rpcbind status
service network status
```

Si le paquet nfs-utils n'est pas installé :

```
yum install nfs-utils
```

Le paquet nfs-utils nécessite plusieurs dépendances dont nfs-utils-lib et rpcbind pour s'installer.

Le service NFS peut être démarré :

```
chkconfig nfs on
service nfs start
```

L'installation du service NFS crée deux utilisateurs :

- nfsnobody : utilisé lors des connexions anonymes ;
- rpcuser : pour le fonctionnement du protocole RPC.

1.3. Configuration du serveur

Les droits du répertoire et les droits NFS doivent être cohérents.

Le fichier /etc/exports

Le paramétrage des partages de ressources s'effectue dans le fichier /etc/exports. A une ligne de ce fichier correspond un partage NFS.

Syntaxe du fichier /etc/exports

```
/partage    client1(permissions) client2(permissions)
```

- **/partage** : Chemin **absolu** du répertoire partagé ;
- **clients** : Machines autorisées à accéder aux ressources ;
- **(permissions)** : Permissions sur les ressources.

Les machines autorisées à accéder aux ressources peuvent être déclarées par :

- Adresse IP : 192.168.1.2
- Adresse réseau : 192.168.1.0/255.255.255.0 ou au format CIDR 192.168.1.0/24
- FQDN : client_*.formatux.lan : autorise les FQDN commençant par client_ du domaine formatux.lan ;
- * pour tout le monde.

Plusieurs clients peuvent être spécifiés sur la même ligne en les séparant par un espace.

Permissions sur les ressources

Il existe deux types de permissions :

- ro : lecture seule ;
- rw : modification.

Si aucun droit n'est précisé, alors le droit appliqué sera lecture seule.

Par défaut les UID et GID des utilisateurs du client sont conservés (excepté pour root).

Pour forcer l'utilisation d'un UID ou d'un GID différent de l'utilisateur qui écrit la ressource, il faudra spécifier les options anonuid=UID et anongid=GID ou donner un accès anonyme aux données avec l'option **all squash** (squash dans sens d'écraser).

Il existe un paramètre, no_root_squash, qui permet d'identifier l'utilisateur root du client comme étant celui du serveur. Ce paramètre peut être dangereux pour la sécurité du système.

Par défaut, c'est le paramètre root_squash qui est activé (même si non précisé), identifiant root comme utilisateur anonyme.

Cas concrets

- /partage client(ro,all_squash)

Les utilisateurs du client n'ont accès qu'en lecture seule aux ressources et sont identifiés comme anonyme sur le serveur.

- /partage client(rw)

Les utilisateurs du client peuvent modifier les ressources et gardent leur UID sur le serveur. Seul root est identifié comme anonyme.

- /partage client1(rw) client2(ro)

Les utilisateurs du poste client1 peuvent modifier les ressources tandis que ceux du poste client2 n'ont qu'un accès en lecture seule.

Les UID sont gardés sur le serveur, seul root est identifié comme anonyme.

- /partage client(rw,all_squash,anonuid=1001,anongid=100)

Les utilisateurs du poste client1 peuvent modifier les ressources. Leurs UID sont transformés en 1001 et leur GID en 100 sur le serveur.

La commande exportfs

La commande exportfs (exported file systems) permet de gérer la table des fichiers locaux partagés avec les clients NFS.

Syntaxe de la commande exportfs

```
exportfs [-a] [-r] [-u partage] [-v]
```

Table 80. Options de la commande exportfs

Option	Description
-a	Active les partages NFS
-r	Prend en compte les partages du fichier /etc/exports
-u partage	Désactive un partage donné

Option	Description
-v	Affiche la liste des partages

La commande showmount

La commande showmount permet de surveiller les clients.

Syntaxe de la commande showmount

```
showmount [-a] [-e] [hôte]
```

Table 81. Options de la commande showmount

Option	Description
-e	Affiche les partages du serveur désigné
-a	Affiche tous les partages en cours sur le serveur

Cette commande permet aussi de savoir si le poste client a l'autorisation de monter les ressources partagées.

"showmount" trie et supprime les doublons dans les résultats (sort|uniq), il est donc impossible de déterminer si un client a fait plusieurs montages d'un même répertoire.

1.4. Configuration du client

L'accès aux ressources partagé d'un serveur NFS se fait par point de montage sur le client.

Si besoin, créer le dossier local pour le montage :

```
$ sudo mkdir /mnt/nfs
```

Lister les partages NFS disponibles du serveur :

```
$ showmount –e 172.16.69.237
/partage *
```

Monter le partage NFS du serveur :

```
$ mount –t nfs 172.16.69.237:/partage /mnt/nfs
```

Le montage peut aussi être automatisé au démarrage du système dans le fichier /etc/fstab :

```
$ sudo vim /etc/fstab
172.16.69.237:/partage /mnt/nfs nfs defaults 0 0
```

Chapitre 2. Serveur de noms DNS - Bind

Le DNS (Domain Name System) est un système de résolution de nom.

Il s'agit d'une base de données distribuée hiérarchique, dont la racine est symbolisée par un « . ».

L'interrogation de cette base se fait selon le principe de client-serveur.

L'URL www.etrs.lan est appelée FQDN (Fully Qualified Domain Name).

Une table de correspondances permet la traduction d'un FQDN en informations de plusieurs types qui y sont associées, comme par exemple son adresse IP.

2.1. Généralités

Il existe 13 serveurs racines répartis dans le monde, dont une majorité se situe en Amérique du Nord.

La traduction d'un FQDN en adresse IP est le cas que l'on rencontre le plus fréquemment mais DNS permet également de renseigner le système sur les serveurs de messagerie, sur les services disponibles, de faire des alias de noms de service, etc.

Par exemple, il est possible de proposer un FQDN smtp.domain.tld sans qu'il y ait réellement de serveur nommé SMTP.

La résolution est possible en IPv4 comme en IPv6.

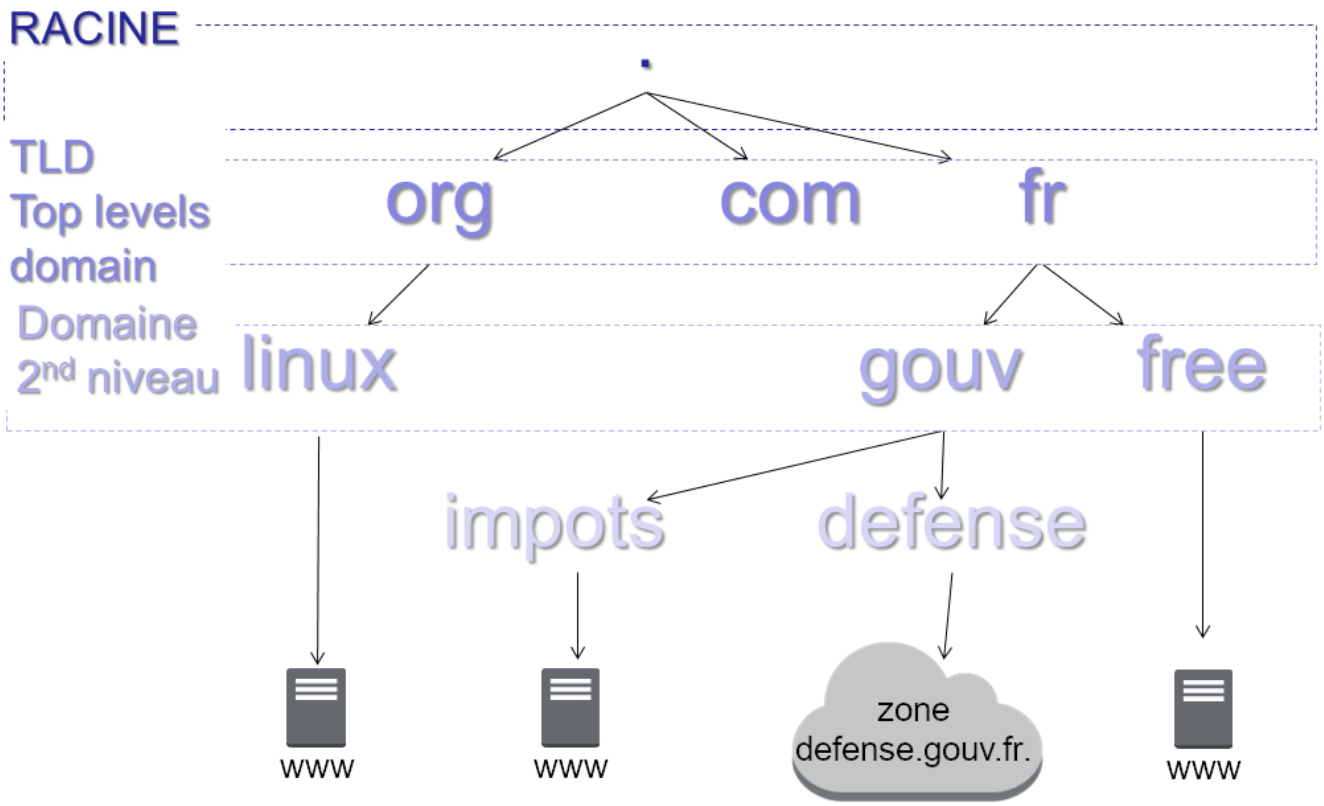


Figure 40. Principe de fonctionnement du DNS

Les Top Levels Domain TLD sont gérés par des organismes bien défini. Les Registars se chargent pour les clients d'enregistrer les noms de domaine auprès de ces organismes.

Le dialogue se fait en général via le protocole UDP (User Datagram Protocol) mais parfois aussi en TCP sur le port 53.

BIND (Berkeley Internet Name Daemon) est un serveur DNS populaire tournant sur système UNIX / Linux.

Bind est disponible sur les serveurs RedHat :

- RHEL 5 : Version 9.3
- RHEL 6 : Version 9.8
- RHEL 7 : Version 9.9

Le protocole peut aussi être utilisé en TCP (connecté) dans certains cas : transferts vers le ou les serveurs secondaires, requêtes dont la réponse est supérieure à la taille d'un paquet UDP, etc.

2.2. Installation du service

Le serveur DNS BIND s'articule autour du service named.

Installation à partir d'un dépôt YUM :

```
[root]# yum install bind
```

L'installation du service BIND ajoute un utilisateur named. Cet utilisateur sera le propriétaire des fichiers de configuration.

```
[root]# grep named /etc/passwd
named:x:25:25:Named:/var/named:/sbin/nologin
```

BIND étant un service réseau, il faut le paramétrer pour un démarrage dans les niveaux 3 et 5, puis le démarrer :

```
[root]# chkconfig --level 35 named on
[root]# service named start
```

Il peut être utile d'installer le paquet bind-utils pour disposer d'outils de test DNS.

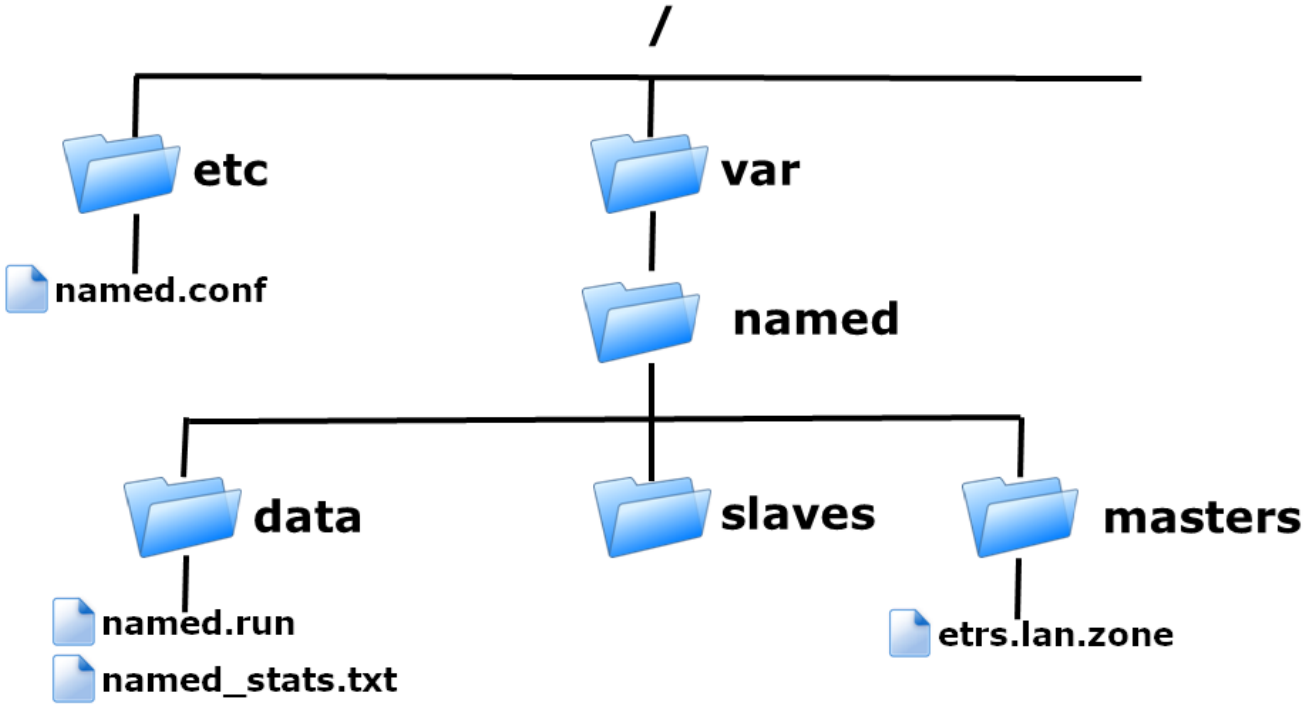


Figure 41. Arborescence du service Bind

Le dossier /var/named/masters n'est pas créé par défaut. Il faudra le créer et ne pas oublier d'y attribuer les droits à l'utilisateur named.

Le fichier de statistiques est généré par la commande rdnc (voir en fin de chapitre).

En environnement de production, les logs des requêtes clientes seront séparés des logs du service lui-même.

Cache DNS

Par défaut, Bind est configuré pour faire office de serveur de proxy-cache DNS (mandataire). Un serveur proxy, ou mandataire, effectue une requête réseau au nom d'un client.

Lorsqu'il résoud pour la première fois une requête DNS, la réponse est stockée dans son cache pour la durée de son TTL (Time To Live) restant. Si le même enregistrement est à nouveau demandé par un autre client DNS, le traitement de la requête sera plus rapide, puisque la réponse sera disponible depuis le cache du serveur.

Chaque enregistrement DNS dispose d'un TTL lorsqu'il est fourni par le serveur qui fait autorité pour la zone. Ce TTL est décrémenté jusqu'à la fin de sa validité. Il est ensuite supprimé des caches.

Lorsque l'enregistrement est mis en cache, le TTL qui lui est associé est le TTL restant.

Récursivité

Un serveur est configuré par défaut pour répondre de manière récursive aux requêtes de ses clients.

Il interroge tour à tour les serveurs DNS nécessaires à la résolution d'une requête jusqu'à obtenir la réponse.

Un serveur non-récursif déléguera la résolution du nom DNS à un autre serveur DNS.

Dans quel cadre utiliser un serveur non-récursif ?

Lorsque la latence entre le serveur DNS et le reste du réseau est trop forte, ou quand le débit est limité, il peut être intéressant de configurer un serveur DNS en non-récursif.

Ce sera par exemple le cas pour un théâtre d'opération ou un élément mobile d'une force.

Le serveur DNS local fera autorité sur la zone locale, mais déléguera la résolution des FQDN de l'intradef à un serveur en métropole, qui lui, prendra la requête en charge de manière récursive.

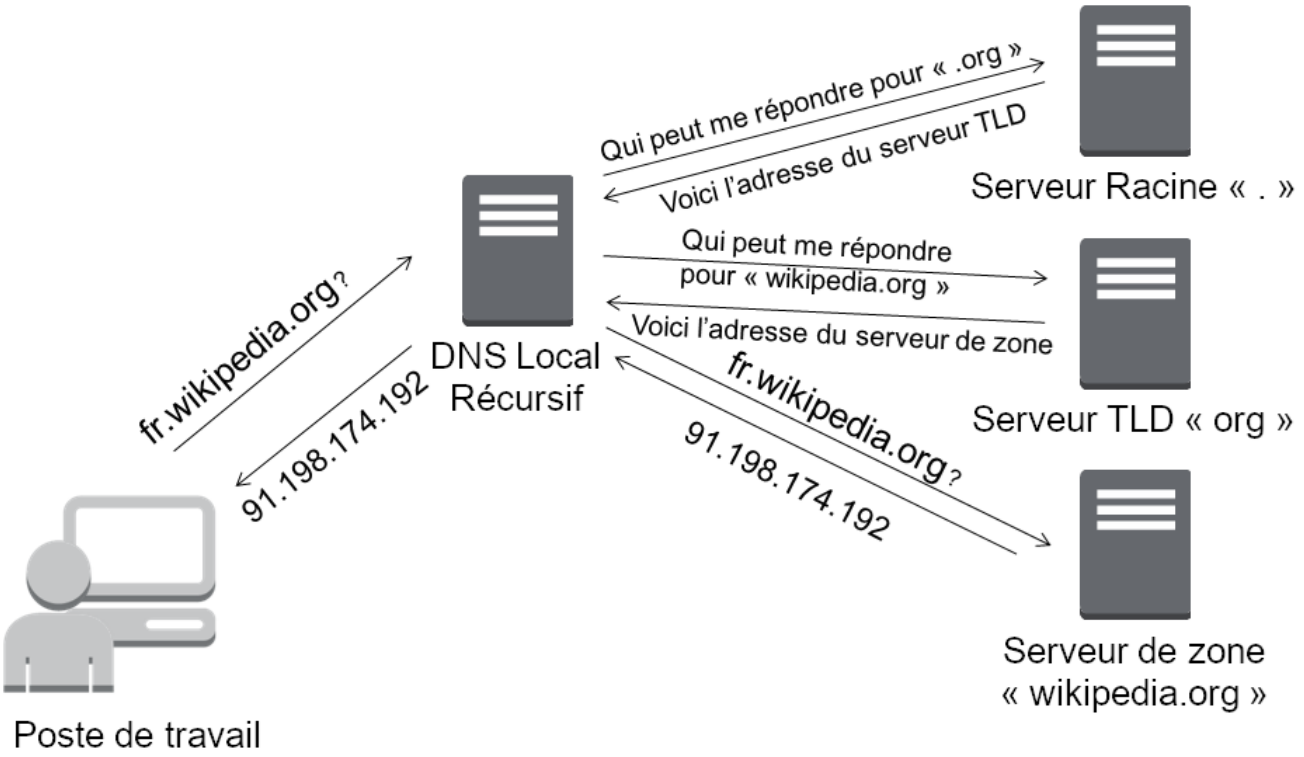


Figure 42. Le mode récursif

Architecture DNS

Un serveur DNS principal dispose d'une copie en écriture de la zone.

Il peut être intéressant de ne le rendre accessible que depuis le domaine local, et ne permettre l'interrogation des DNS depuis l'extérieur que vers les serveurs secondaires. D'un point de vue architectural cela lui évite ainsi les attaques ou les surcharges.

Le serveur est alors appelé serveur furtif.

Un serveur DNS n'est pas nécessairement le serveur maître de toutes les zones pour lesquels il fait autorité.

Un serveur DNS peut très bien être configurer pour être maître d'une zone et esclave d'une autre.

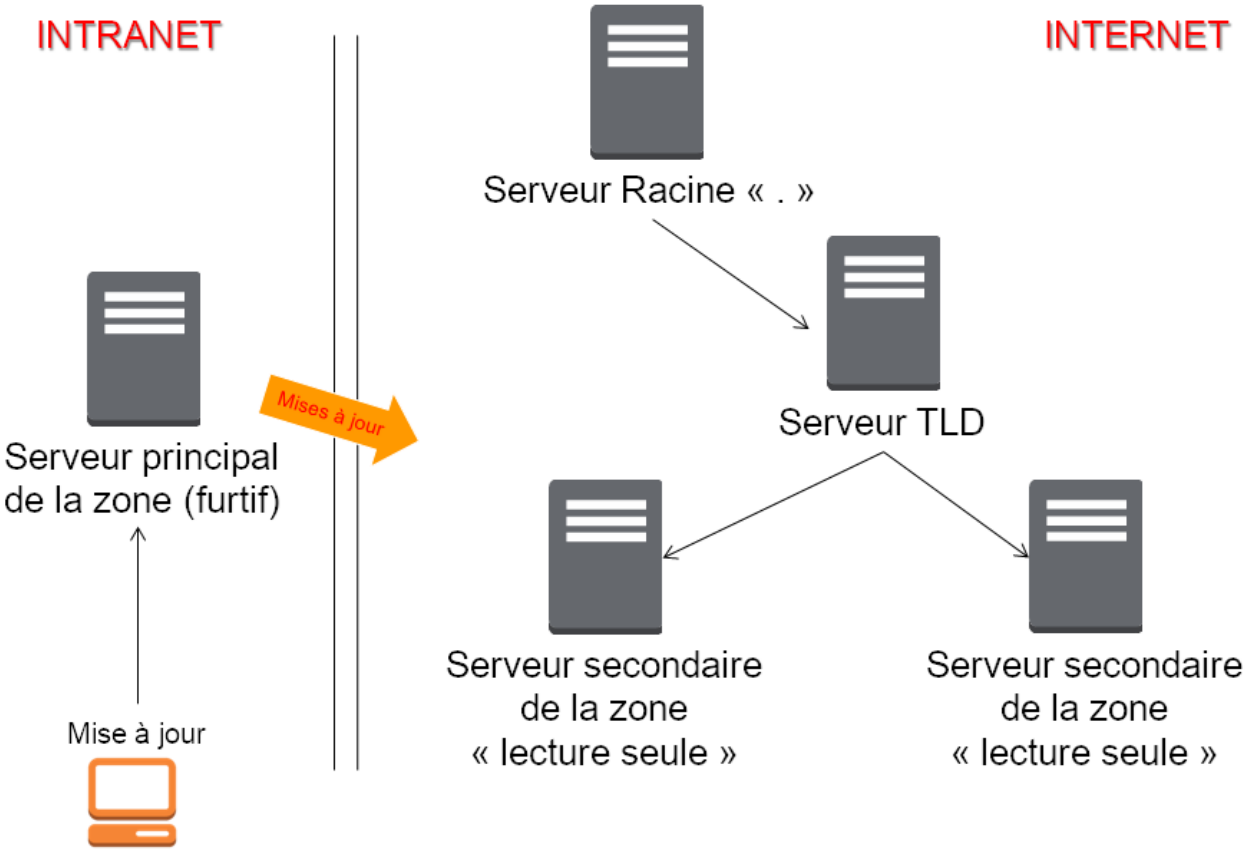


Figure 43. Architecture avec serveur furtif

La mise à jour du serveur se fait depuis le réseau local. Le serveur principal n'est pas accessible depuis l'extérieur.

La mise à jour se propage vers les serveurs secondaires.

L'interrogation par les clients internet ne se fait que sur les serveurs secondaires, ce qui protège le serveur principal des attaques par déni de service.

Pour un réseau complexe, il existe de nombreuses solutions architecturales de l'infrastructure DNS qu'il est important de bien étudier.

2.3. Configuration du serveur

Le fichier /etc/named.conf

Ce fichier contient les paramètres de configuration du service DNS.

```
[root]# less /etc/named.conf
options {
    listen-on port 53 { 192.168.1.200; };
    directory "/var/named";
    allow-query { 192.168.1.0/24; };
};
```

Chaque ligne du fichier /etc/named.conf (même à l'intérieur des accolades) se termine par un point-virgule.

L'oublie de ce ";" est l'erreur la plus fréquente dans la configuration d'un serveur Bind.

Les noms, sous la forme FQDN (Fully Qualified Domain Name) doivent se terminer par ".". En l'absence de ce ".", Bind suffixera automatiquement avec le nom de domaine l'enregistrement.

Eg : www.etrs.lan → www.etrs.lan.etrs.lan.

La rubrique options contient la configuration générale du serveur BIND via différentes directives :

- listen-on : Définit l'interface, l'adresse et le port du service sur le serveur.
- directory : Définit le répertoire de travail de BIND, contenant les fichiers de zone.
- allow-query : Définit les hôtes autorisés à faire des requêtes sur le serveur. Par adresse IP ou réseau.

Permettre qu'un serveur DNS résolve les requêtes de n'importe quel client est une très mauvaise idée. Il faut au moins restreindre les droits aux réseaux locaux.

Il est possible, pour cela, de créer des ACL pour simplifier l'administration du fichier de configuration.

Le fichier de configuration contient également les informations relatives aux fichiers de zone.

```
[root]# less /etc/named.conf
zone "etrs.lan" IN {
    type master;
    file "masters/etrs.lan.direct";
    allow-update { 192.168.1.0/24; };
};

zone "1.168.192.in-addr.arpa" IN {
    type master;
    file "masters/etrs.lan.inverse";
};
```

Les rubriques **zone** contiennent les configurations des zones de résolution de nom, inverses ou directes :

- type : Définit le type de serveur pour cette zone :
  - maître : possède la base de données en écriture
  - esclave : possède la base en lecture seule
  - forwarder : fait office de proxy-cache pour cette zone.
- file : Définit le chemin du fichier de zone.
- allow-update : Définit les hôtes ayant l'autorisation de mettre à jour les enregistrements DNS.

Les fichiers de zone inverse sont nommés en prenant l'adresse réseau de la zone (en inversant les octets) suivi du domaine in-addr.arpa.

Les rôles

Un serveur peut avoir le rôle de maître pour la zone, c'est-à-dire qu'il possède la zone en écriture.

```
[root]# less /etc/named.conf
zone "etrs.lan" IN {
    type master;
    file "masters/etrs.lan.direct";
    allow-update { 192.168.1.0/24; };
};
```

Seuls les clients figurant dans la variable allow-update pourront mettre à jour la base de données du DNS.

Un serveur peut également être un serveur secondaire (slave) pour la zone, c'est-à-dire qu'il possède la zone en lecture.

```
[root]# less /etc/named.conf
zone "etrs.lan" IN {
    type slave;
    file "slaves/etrs.lan.direct";
};
```

Un serveur peut enfin être expéditeur (forwarder) pour la zone, c'est-à-dire qu'il a connaissance de cette zone, et relaie les informations pour celle-ci.

```
[root]# less /etc/named.conf
zone "unautredomaine.fr" IN {
    type forwarder;
    forwarders {221.10.12.1};
};
```

Vous retrouverez cette notion sous Windows en tant que « redirecteur ».

2.4. Fichiers de zone

Les fichiers présents dans le répertoire /var/named doivent appartenir à l'utilisateur système named.

SELinux ne permettra pas l'enregistrement des fichiers de zone en dehors de ce répertoire.

Ces fichiers contiennent des enregistrements (RR : Resource Records) DNS de différents types. Ils permettent la résolution directe de noms (du nom vers l'adresse IP), ou la résolution inverse (de l'adresse IP vers le nom).

En plus de contenir les adresses IP et les noms des machines, les fichiers contiennent les paramètres de durée de vie des enregistrements (Time To Live, TTL).

Lorsqu'un enregistrement DNS est mis en cache, le temps restant sur son TTL est également conservé. À la fin du TTL, l'enregistrement est supprimé des caches.

- Un TTL plus long réduit les échanges DNS.
- Un TTL plus court permet une reconfiguration du réseau plus rapide.

Les types d'enregistrements

Table 82. Les types d'enregistrements

Type	Description
A	Nom attribué à une adresse de type IP V4

Type	Description
AAAA	Nom attribué à une adresse de type IP V6
CNAME	Alias d'un enregistrement A déjà défini **Éviter de faire un alias vers un alias**
MX	Serveur de messagerie destinataire pour la zone concernée
NS	Le ou les serveurs de noms de la zone (type A)
PTR	Enregistrement pointeur pour une zone inverse
SOA	Démarre la configuration (cf: diapos suivantes)
SVR	Service (protocole jabber,...)
TXT	Informations

- Champ MX : Le numéro précise la priorité, la plus faible étant la plus prioritaire. Ceci permet de définir un ou plusieurs serveurs de secours qui stockeront les mails en attendant le retour du serveur principal.
- Champ de type A : Enregistrement standard. Attribue un nom à une adresse IP.

Plusieurs enregistrements identiques de type A vers des adresses différentes permet de faire de l'équilibrage de charge par round-robin (RR).

Exemple :

```
mail    A   192.168.1.10
        A   192.168.1.11
```

- Champ AAAA : On utilise quatre A pour symboliser IPv6 car une adresse IPv6 est codée sur 16 octets, soit 4 fois plus qu'une adresse IPv4.
- CNAME : Permet d'attribuer un ou plusieurs alias à un enregistrement A déjà défini. Plusieurs enregistrements du même alias permettent également de faire de l'équilibrage de charge type RR.

On trouvera des enregistrements typiques, comme autoconfig, qui permet le mécanisme de configuration automatique d'un client de messagerie.

Fichier de zone directe

Ce fichier est nécessaire au fonctionnement du système DNS. C'est par lui que se fait la résolution d'un nom en adresse IP.

```
[root]# less /var/named/etrs.lan.direct
$ORIGIN .
$TTL 3600
etrs.lan.  SOA  npinf-mcsnlz01v.etrs.lan. contact.etrs.lan. (123; 14400; 3600; 604800;
3600; )

@               IN  NS      npinf-mcsnlz01v.etrs.lan.
poste1          IN  A       192.168.1.10
npinf-mcsnlz01v IN  A       192.168.1.200
etrs.lan.       MX  10      192.168.1.201
npinf-mcsnlw01v IN  A       192.168.1.202
www             IN  CNAME   npinf-mcsnlw01v.etrs.lan.
```

- $ORIGIN : Définit la valeur par défaut du domaine courant pour les renseignements du fichier. Un . signifie la racine.
- $TTL : Durée de vie par défaut des enregistrements de la zone dans le cache, exprimée en secondes. Le TTL peut également être précisé enregistrement par enregistrement.
- SOA : Start Of Authority. La ligne démarre la configuration d'une zone. Définit :
  - le nom du serveur maître principal,
  - l'email de l'administrateur de la zone (un . remplace le @ de l'adresse mail).
  - Entre parenthèses, le numéro de série du fichier (incrémenté à chaque mise à jour) et les délais de mise à jour ou de rafraîchissement, exprimés en secondes.
    - Numéro de zone : Numéro incrémental (voir le paragraphe suivant)
    - Rafraîchissement : Durée en secondes avant une tentative de synchronisation avec le serveur maître
    - Réitération : Intervalle de temps avant réitération si l'essai précédent n'a pas fonctionné
    - Expiration : Durée en secondes avant l'expiration car le serveur maître est injoignable
    - Cache négatif (TTL) : Durée de vie en secondes des enregistrements

Le @ a une signification particulière pour Bind. Il se représente lui même, raison pour laquelle le @ de l'adresse courriel d'administration est remplacée par un .

Le numéro de la zone sert à identifier la dernière modification du DNS maître. Tous les serveurs secondaires utilisent ce numéro pour savoir s'ils doivent se synchroniser.

Il existe deux méthodes d'incrémentation du numéro de zone :

- Incrémentale : 1, puis 2, puis 3 (pourquoi pas ?)
- Basée sur la date : AAAAMMJJXX, qui nous donne par exemple, pour la première modification du jour : 2017210101 (méthode à privilégier)

Le fichier de zone inverse

Bien que non obligatoire, ce fichier est fortement conseillé pour un fonctionnement optimal du système DNS. C'est par lui que se fait la résolution d'une adresse IP en nom.

Des services comme SSH s'appuie sur la résolution inverse.

```
[root]# more /var/named/etrs.lan.inverse
$ORIGIN 1.168.192.in-addr.arpa.
$TTL 259200
@               SOA npinf-mcsnlz01v.etrs.lan. contact.etrs.lan. ( 123; 14400; 3600;
604800; 3600; )
@               NS      npinf-mcsnlz01v.etrs.lan.
1   0           PTR     poste1.etrs.lan.
200     PTR     npinf-mcsnlz01v.etrs.lan.
```

La commande

L'usage de la commande nsupdate est exclusive. Il ne faut plus modifier les fichiers de zone manuellement, sous peine de pertes d'informations.

Syntaxe :

```
nsupdate
```

Exemple:

```
[root]# nsupdate
        > server 192.168.1.200
        > zone etrs.lan
        > update add poste2.etrs.lan 3600 A 192.168.1.11
        > update delete poste1
        > send
```

La commande nsupdate est interactive.

À la saisie, elle ouvre un prompt dans lequel il faut saisir les requêtes de mise à jour du fichier de zone.

Ces requêtes peuvent être :

- **server** : Précise le serveur BIND pour lequel les requêtes seront envoyées.
- **zone** : Précise la zone de résolution pour laquelle les requêtes seront envoyées.

- **prereq yxdomain nom** : L'existence de l'enregistrement nom est une condition de mise à jour.
- **update add nom TTL type @IP** : Ajoute l'enregistrement nom, en précisant son type, son adresse IP et son TTL.
- **update delete nom** : Supprime l'enregistrement nom.
- **send** : Valide et envoie les requêtes.

La commande

La commande **rndc** permet de manipuler le serveur DNS à chaud.

Syntaxe de la commande rndc

```
rndc reload
rndc querylog on|off
```

- **reload** : Prend en compte les modifications apportées sans devoir relancer le service
- **querylog** : Active ou non la journalisation

Après modification d'un fichier de zone, il est nécessaire de faire prendre en compte les modifications au service. Les fichiers étant lus au démarrage du service, cela permet de prendre en compte les modifications, mais résulte en la perte des statistiques. Il faut donc privilégier la méthode reload.

Le suivi des logs

La fonction d'enregistrement des fichiers journaux est activée ou désactivée par la commande rndc.

Le fichier de logs est par défaut /var/named/data/named.run

Exemple :

```
[root]# rndc querylog on
[root]# tail -f /var/named/data/named.run
```

Bind propose dans son fichier de configuration des options pour journaliser les informations :

- de transferts,
- de requêtes clients,
- d'erreurs,
- ...

2.5. Configuration du client

NetworkManager est un outil de gestion du réseau. Sur un serveur dont le réseau est défini par cet outil, la configuration cliente de Bind est décrite dans le fichier de l'interface.

```
[root]#less /etc/sysconfig/network-scripts/ifcfg-ethX
DOMAIN="etrs.lan"
DNS1=192.168.1.200
DNS2=192.168.1.201
```

NetworkManager modifiera lui-même le fichier /etc/resolv.conf à chaque relance du service réseau.

Avant de lancer une recherche DNS, le logiciel client vérifiera si la requête porte sur un FQDN ou non. Si le nom n'est pas pleinement qualifié, le client suffixera la requête avec le premier suffixe DNS fourni.

Si la résolution est impossible, le client émettra une nouvelle requête avec le suffixe suivant, ainsi de suite jusqu'à l'obtention d'une réponse.

Par exemple, il est possible de fournir deux suffixes :

```
dr-cpt.intradef.gouv.fr
intradef.gouv.fr
```

Lors d'une requête DNS portant sur, par exemple, portail-etrs, une première requête portail-etrs.dr-cpt.intradef.gouv.fr sera faite. En l'absence de réponse positive, une seconde requête sera effectuée portant sur portail-etrs.intradef.gouv.fr. Pour éviter ce phénomène d'interrogation multiple, il est préférable de fournir une adresse pleinement qualifiée.

Veiller à spécifier au moins deux serveurs DNS pour assurer une redondance en cas de panne du serveur principal.

Sans outil de gestion du réseau, la configuration cliente de Bind est décrite dans le fichier /etc/resolv.conf.

```
[root]# less /etc/resolv.conf
search "etrs.lan"
nameserver 192.168.1.200
nameserver 192.168.1.201
```

NetworkManager, si actif, écrasera les valeurs entrées manuellement dans ce fichier.

L'utilitaire **system-config-network-tui** (**nmtui** sous CentOS 7) permet une configuration graphique

correcte du réseau par une interface ncurse.

La commande

La commande dig (Domain Information Groper) permet d'interroger des serveurs DNS.

Dig doit être privilégié par rapport à NSLookup qui n'est plus maintenue.

Syntaxe de la commande dig

```
dig [name] [type] [options]
```

Exemple :

```
[root]# dig centos65.etrs.lan A
...
;; QUESTION SECTION:
; centos65.etrs.lan.    IN  A

;; ANSWER SECTION:
centos65.etrs.lan.  86400   IN  A   192.168.253.131

;; AUTHORITY SECTION:
etrs.lan.       86400 IN    NS  centos65.etrs.lan.
...
```

```
[root]# dig –t MX linux.fr
```

```
[root]# dig linux.fr MX +short
```

Mise en cache côté client

Le service NSCD est responsable de la mise en cache des requêtes réseaux type LDAP ou DNS.

Pour profiter de la mise en cache local, il faudra veiller à ce que le service NSCD soit démarré.

```
[root]# service nscd start
[root]# chkconfig nscd on
```

Nscd n'est pas installé par défaut sur les RHEL 6.

Mise à jour dynamique

Les clients peuvent s'enregistrer dynamiquement sur le serveur DNS, ce qui est intéressant dans le cadre d'une attribution de l'adressage IP dynamique avec DHCP.

2.6. Configuration du pare-feu serveur

Les règles iptables a configurer en tcp et udp sont les suivantes :

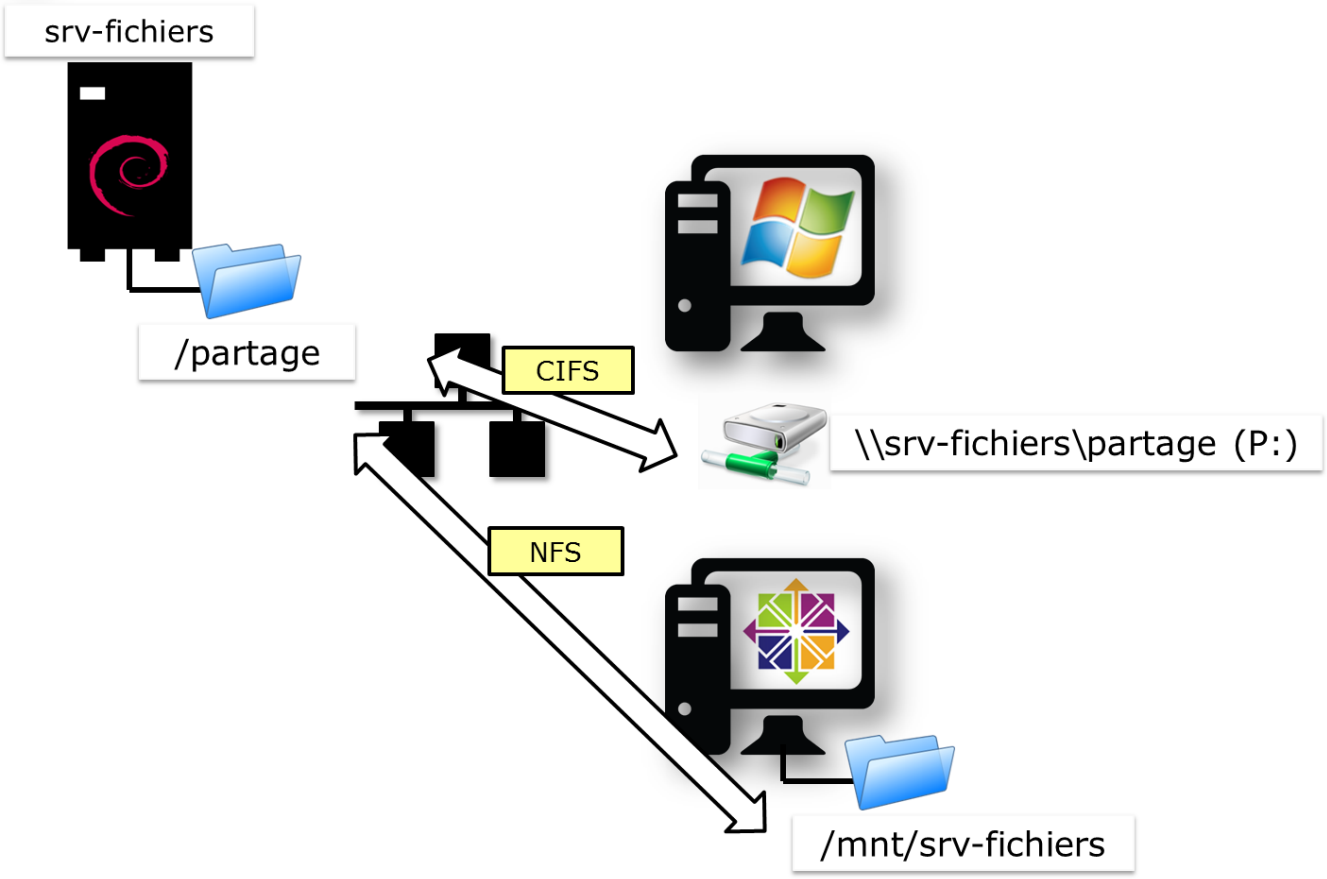
```
[root]# vi /etc/sysconfig/iptables
# Autoriser DNS
iptables –t filter –A INPUT –p tcp –dport 53 –j ACCEPT
iptables –t filter –A INPUT –p udp –dport 53 –j ACCEPT
```

system-config-firewall-tui est l'outil graphique permettant de configurer le pare-feu.

Chapitre 3. Serveur de fichiers Samba

Samba est un serveur de fichiers permettant l'interopérabilité entre divers systèmes, notamment les systèmes Linux et Microsoft. Il permet à des systèmes Linux de créer des partages utilisables par des machines Windows et vice-versa.

Le projet Samba a été initié dès 1992 sous licence GPL (et donc gratuit).



Un même dossier partagé par NFS et par Samba est ainsi accessible depuis toutes les plateformes clientes.

3.1. Le protocole SMB

Le protocole SMB (Server Message Block) était une extension de Microsoft pour permettre la redirection des entrées/sorties vers NetBIOS (Network Basic Input/Output System).

SMB permettait :

- le transfert de données entre machines Windows : partages de fichiers, impressions et messagerie électronique ;
- le parcours du voisinage réseau (browsing) ;
- la résolution de noms NetBIOS en adresses IP (Windows Internet Name Server) ;
- l'authentification centralisée (notion de domaine).

Le protocole NetBIOS était une interface permettant la mise en place de noms de machines, de groupes de travail, de domaines, etc. Il faisait fonctionner le voisinage réseau jusqu'à Windows 2000, mais son mode de fonctionnement induisait une charge réseau importante.

NetBIOS était le système de noms des réseaux SMB comme l'est aujourd'hui le service DNS.

Les diffusions NetBIOS ne passant pas les routeurs, la notion de voisinage réseau désigne l'ensemble des stations de travail utilisant le protocole NetBIOS sur un même segment de réseau IP. Le **maître explorateur** (master browser) est le poste client ou serveur tenant à jour la liste des ordinateurs utilisés par le service de voisinage réseau.

3.2. Le protocole

Les améliorations apportées au protocole SMB ont permis de faire aboutir la suite de protocoles clients/serveurs CIFS (Common Internet File System) que le service Samba implémente.

3.3. Installation de Samba

```
[root]# yum install samba smbclient
```

La partie serveur de Samba est basée essentiellement sur 2 démons :

- Le démon **smb** est le serveur SMB : il répond aux requêtes des clients lorsque ceux-ci accèdent aux partages définis et il est le seul à accéder aux systèmes de fichiers Linux.
- Le démon **nmb** est le serveur de noms NetBIOS essentiel au fonctionnement de SMB. Il peut être configuré comme serveur WINS.
- Le fichier de configuration principal est **smb.conf**. C'est dans ce fichier que sont définis les paramètres de fonctionnement des 2 démons, ainsi que la définition des exports de ressources.

La partie cliente de samba (samba-client) contient les outils qui permettent le montage et le parcours des ressources Samba.

Le paquet **samba-client** fournit les binaires :

- **findsmb** : afficher de nombreuses informations sur les stations d'un sous-réseau qui répondent aux requêtes de noms SMB.
- **nmblookup** : interroger le protocole NetBIOS et associe les noms Netbios à des adresses IP.
- **sharesec** : manipuler les droits de partages de fichiers
- **smbcacls** : manipuler les ACL NT sur les partages de fichiers
- **smbclient** : offrir une interface FTP Like pour accèder à des partages de fichiers
- **smbget** : télécharger des fichiers depuis un partage windows
- **smbspool** : envoyer une impression à un serveur d'impression

- **smbtree** : fournir un explorateur de fichier en mode texte similaire au “Voisinage réseau” des ordinateurs Windows. Il affiche un arbre des domaines connus, desserveurs et des partages accessibles depuis les serveurs.
- ...

Le paquet **samba-common** fournit les binaires :

- **net** : offrir les mêmes fonctionnalités que la commande net du monde Windows.
- **pdbedit** : gérer de la base SAM
- **smbcquotas** : manipuler les quotas NT d’un partage de fichiers
- **smbpasswd** : changer le mot de passe des utilisateurs
- **testparm** : tester la syntaxe d’un fichier de configuration smb.conf
- ...

```
[root]# chkconfig smb on
[root]# chkconfig nmb on
[root]# service smb start
[root]# service nmb start
```

3.4. Sécurité SELinux

Par défaut, la sécurité SELinux est active. Pour vérifier le contexte de sécurité en place sur des fichiers, il faut utiliser la commande :

```
[root]# ls -Zd /export/
```

Le contexte de sécurité *samba_share_t* doit être positionné sur les dossiers partagés :

```
[root]# chcon -R -t samba_share_t /export/
```

Plus d’information sur la sécurité SELinux et Samba sur le site de RedHat. Pensez à stopper le parefeu ou à le configurer dans le fichier **/etc/sysconfig/iptables** :

```
-A INPUT -m state --state NEW -m udp -p udp --dport 137 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 137 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 138 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 139 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 445 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 445 -j ACCEPT
```

Pour pouvoir utiliser Samba comme contrôleur de domaine et utiliser les commandes useradd et groupadd, il faudra mettre le booléen **samba_domain_controller** à on.

```
[root]# setsebool -P samba_domain_controller on
```

Pour rappel, il est possible d'obtenir tous les booléens SELinux concernant Samba avec la commande suivante :

```
[root]# getsebool -a | grep "samba"
samba_create_home_dirs --> off
samba_domain_controller --> off
samba_enable_home_dirs --> off
samba_export_all_ro --> off
samba_export_all_rw --> off
samba_portmapper --> off
samba_run_unconfined --> off
samba_share_fusefs --> off
samba_share_nfs --> off
sanlock_use_samba --> off
use_samba_home_dirs --> off
virt_use_samba --> off
```

Pour autoriser les partages via Samba des répertoires de connexions des utilisateurs, il faudra positionner le booléen **samba_enable_home_dirs** également à on.

```
[root]# setsebool -P samba_enable_home_dirs on
```

3.5. La configuration de SAMBA

La partie serveur de Samba est basée sur 1 seul fichier de configuration **/etc/samba/smb.conf** qui définit les paramètres de fonctionnement des 2 démons et des exports de ressources.

Chaque paramètre fait l'objet d'une ligne au format :

```
nom = valeur
```

Les commentaires commencent par un ';' ou un '#' et se termine à la fin de la ligne.

Le caractère '\' permet de scinder une ligne logique sur plusieurs lignes physiques.

Ce fichier est constitué de 3 sections spéciales :

- **[global]** : paramètres généraux ;

- **[homes]** : paramètres des répertoires utilisateurs ;
- **[printers]** : paramètres des imprimantes.

Les autres sections, déclarées entre '**[ ]**' sont des déclarations de partage.

Les niveaux de sécurité

Il existe cinq niveaux de sécurité (option security), mais un seul peut être appliqué par serveur :

- **share** : le niveau dit de partage, lié à une ressource. Un mot de passe est associé à chaque partage (Déprécié : ne plus utiliser) ;
- **user** : le niveau de sécurité de l'utilisateur, lié à son authentification. C'est le niveau recommandé et par défaut ;
- **server** : l'authentification est réalisée par un autre serveur (Déprécié : ne plus utiliser) ;
- **domain** : l'authentification est réalisée par un autre serveur, mais le serveur Samba doit être membre du domaine ;
- **ads** : l'authentification est réalisée par un serveur Active Directory.

Les variables internes à Samba

Table 83. Les variables internes à Samba

Variable	Observation
%a	Architecture du client
%I	Adresse IP du client
%M	Nom dns du client
%m	Nom NetBios du client
%u	Identité de l'utilisateur pour le partage concerné
%U	Identité souhaitée par l'utilisateur du partage
%H	Répertoire de connexion de l'utilisateur
%u%g ou %G	Groupe principal de l'utilisateur
%u ou %U%S	Nom du partage
%P	Répertoire racine du partage concerné
%d	PID du processus courant
%h	Nom DNS du serveur Samba
%L	Nom NetBIOS du serveur Samba
%v	Version de samba
%T	Date et heure système
%$var	valeur de la variable d'environnement var

La commande testparm

La commande **testparm** teste la validité du fichier /etc/samba/smb.conf.

L'option -v affiche tous les paramètres applicables.

```
[root]# testparm
Load smb config files from /etc/samba/smb.conf
...
Loaded services file OK.
Server role : ROLE_STANDALONE
...
```

Sans option, cette commande renvoie la configuration du serveur samba sans les commentaires et omet les paramètres positionnés à leur valeur par défaut, ce qui facilite sa lecture.

```
[root]# testparm
```

La section [global]

Table 84. Les directives de la section [global]

Directive	Exemple	Explication
workgroup	workgroup = ETRS	Définir le groupe de travail ou le nom de domaine NetBIOS. (À mettre en majuscule).
netbios name	netbios name = drinf-mcsnlw01v	Nom NetBIOS de la station Maximum 15 caractères Pas de rapport direct avec le nom de la machine Linux
server string	server string = Samba version %v	Description du serveur apparaissant dans l'explorateur Windows
hosts allow	hosts allow = 127. 172.16.76.	Permet de restreindre les clients du serveur aux seuls réseaux mentionnés. Notez la présence d'un point à la fin de l'adresse et l'absence du 0.
log file	log file = /var/log/samba/log.%m	Enregistrer les évènements dans un fichier %m représente le nom NetBIOS du client
security	security = user	Modèle de sécurité du serveur

Directive	Exemple	Explication
passdb backend	passdb backend = tdbsam	Stockage des utilisateurs et des mots de passe. Le format tdbsam (Trivial Database) est le format par défaut, limité en performance à 250 utilisateurs. Au delà, il faudra passer au format ldapsam et stocker les utilisateurs et les groupes dans une base LDAP.

La section [homes]

La section [homes] contient la configuration des partages utilisateurs.

C'est une section réservée par Samba, qui lui applique un fonctionnement très particulier. Ce nom de partage ne doit pas être utilisé pour un autre partage ni modifié.

```
[homes]
    comment = Home Directories
    browseable = no
    writable = yes
```

Tous les utilisateurs verront le même partage "homes" mais le contenu sera personnalisé pour chacun.

Attention à bien configurer les booléens SELinux pour autoriser le partage des dossiers personnels.

La section [printers]

La section [printers] contient la configuration du serveur d'impression.

```
[printers]
    comment = All Printers
    browseable = no
    writable = yes
    guest ok = no
    printable = yes
```

Samba peut ainsi faire office de serveur d'impressions, ce qui est une fonctionnalité intéressante (ne nécessite pas l'acquisition de licences clientes).

Partages personnalisés

Avant de paramétrer une nouvelle section du fichier smb.conf qui correspondra à un nouveau partage, il convient de se poser quelques questions :

- Quel est le chemin du partage ?
- Qui peut modifier le contenu ?
- Le partage doit-il être visible sur le réseau ou au contraire sera-t-il masqué ?
- Y aura-t-il un accès anonyme ?

Un nouveau partage est représenté par une section [nomdupartage] dans le fichier smb.conf. En voici un exemple :

```
[partage]
    comment = Partage
    browseable = yes
    writable = yes
    path = /export/data
    valid users = @users
    read list = georges
    write list = bob, alice
    invalid users = maurice
    create mask = 0664
    directory mask = 0775
    force group = users
```

De nombreuses directives sont disponibles pour configurer les partages :

Directive	Exemple	Explication
comment	comment = Exemple de partage	Affiche un commentaire dans l'explorateur de fichiers.
browseable	browseable = yes	Affiche le partage dans le voisinage réseau.
writeable	writeable = yes	Le partage est en lecture seule ou en écriture.
path	path = /export/data	Le chemin absolu à partager sur le réseau. Attention au contexte SELinux de ce dossier.
valid users	valid users = @users	Liste les utilisateurs ou les groupes autorisés à accéder au partage.

Directive	Exemple	Explication
invalid users	invalid users = alice	Liste les utilisateurs ou les groupes qui ne sont pas autorisés à accéder au partage.
read list	read list = bob	Liste les utilisateurs ou les groupes autorisés à accéder au partage en lecture.
write list	write list = patrick, alain	Liste les utilisateurs ou les groupes autorisés à accéder au partage en écriture.
create mask	create mask = 0664	Les fichiers créés prendront les droits spécifiés.
directory mask	directory mask = 0775	Les dossiers créés prendront les droits spécifiés.
force group	force group = users	Les nouveaux fichiers et dossiers appartiendront au groupe spécifié. Dans ce cas, il n'y a pas d'@ devant le groupe !

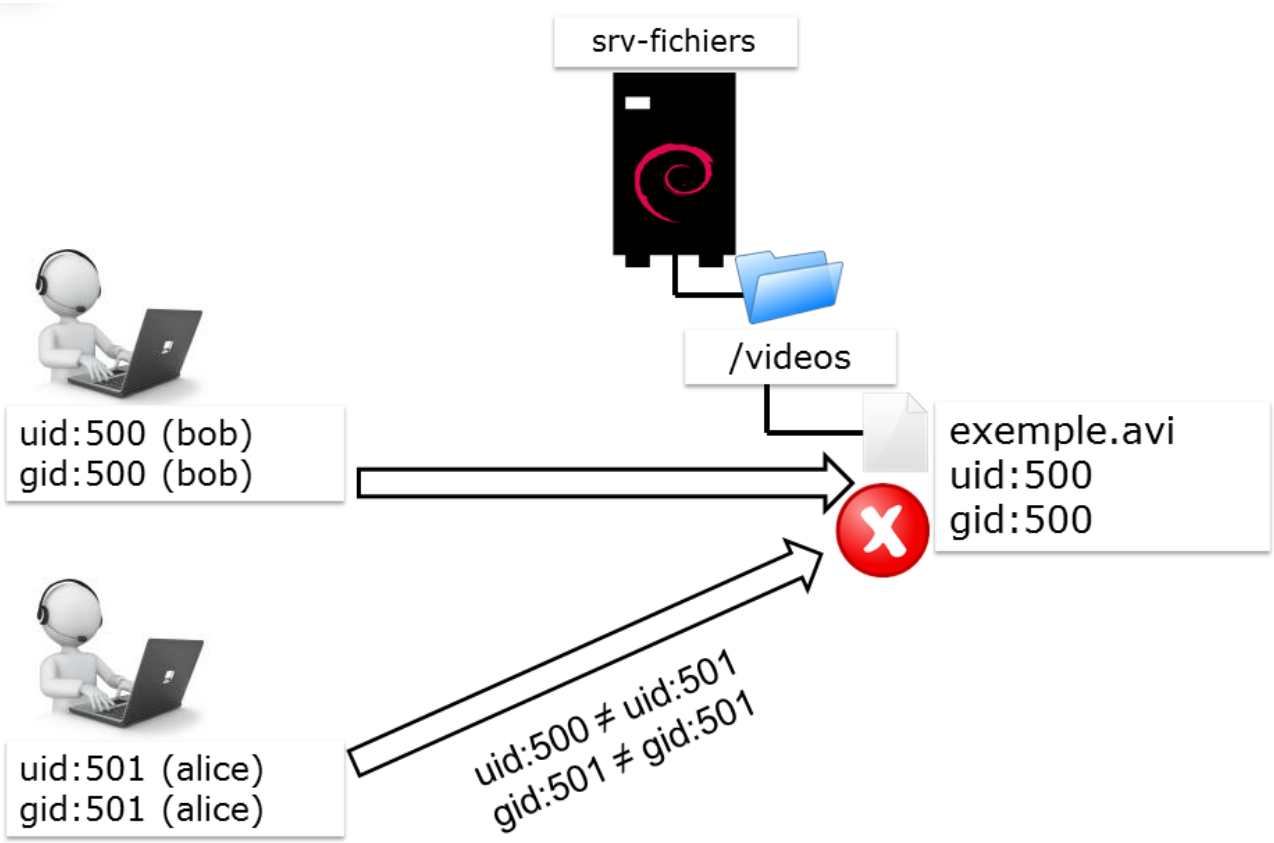
La directive force group

La directive **force group** permet de forcer l'appartenance d'un fichier créé à un groupe spécifique.

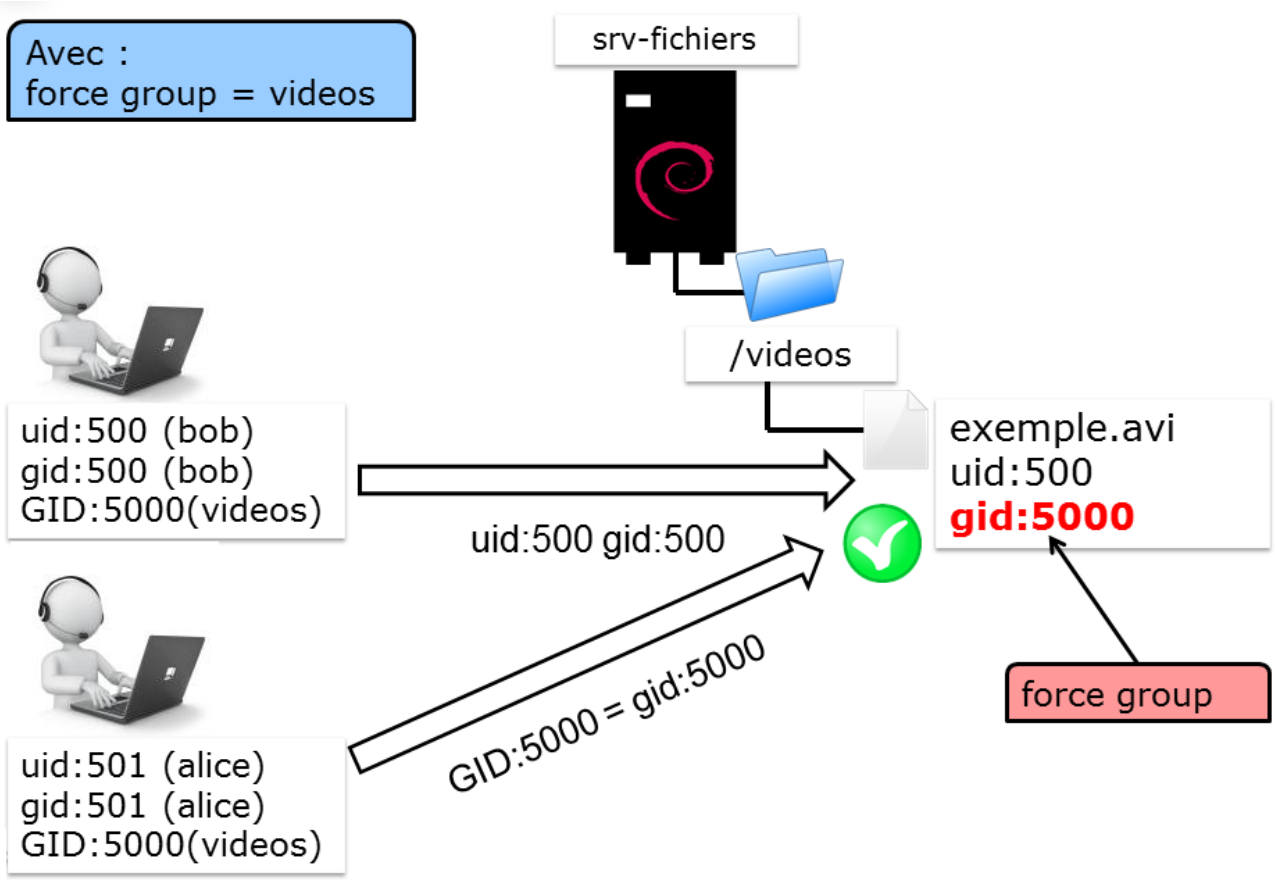
Cette directive est essentielle dans le fonctionnement de Samba, puisqu'elle assure que, quelque soit le groupe principal d'un utilisateur, celui-ci sera autorisé à accéder à un fichier sur le partage s'il fait parti, en tant qu'invité, du groupe spécifié dans la directive.

Les deux exemples ci-desous mettent en avant ce mécanisme :

Utilisation sans le mécanisme **force group** :



Utilisation avec le mécanisme **force group** :



3.6. Commandes d'administration

La commande pdbedit

La commande pdbedit permet de gérer la base SAM des utilisateurs Samba, que le backend soit au format tdbsam ou ldapsam, contrairement à la commande smbpasswd qui est limitée au format tdbsam.

Syntaxe de la commande pdbedit

```
pdbedit [-a|-r|-x|-L] [-u username] ...
```

Exemple :

```
[root]# pdbedit -L
stagiaire:1000:Stagiaire SYS
```

Table 85. Options principales de la commande pdbedit

Option	Observation
-a	Ajouter un utilisateur
-r	Modifier un utilisateur
-x	Supprimer un utilisateur
-L	Lister les utilisateurs
-u	Spécifier le nom de l'utilisateur pour les options -a, -r, et -x

Ajouter un utilisateur

Le format de cryptage du mot de passe entre le monde Microsoft et le monde Linux étant différent, Samba doit soit tenir à jour une base de données contenant les mots de passe au bon format ou déléguer cette gestion au serveur LDAP, ce qui explique l'usage de la commande smbpasswd.

```
pdbedit -a -u username [-f description]
```

Exemple :

```
[root]# pdbedit -a -u bob -f "Bob Leponge"
Unix username:     bob
User SID:          S-1-5-21-3024208064-2128810558-4043545969-1000
Full Name:         Bob Leponge
Home Directory:    \\srvfichiers\bob
Domain:            SRVFICHIERS
...
```

Option	Observation
-a	Ajouter un utilisateur. L'utilisateur doit exister dans le fichier /etc/passwd.
-u	Spécifier le nom de l'utilisateur à ajouter.

La commande smbpasswd

La commande smbpasswd permet de gérer les mots de passe des utilisateurs Samba.

Syntaxe de la commande smbpasswd

```
smbpasswd [-d|-e] username
```

Exemple :

```
[root]# smbpasswd bob
New SMB password:
Retype new SMB password:
```

Option	Observation
-e	Réactive un compte.
-d	Désactive un compte.

Il est possible de synchroniser les mots de passe Unix et Samba :

```
[global]
    unix password sync = yes
    obey pam restrictions = yes
```

Directive	Exemple	Explication
unix password sync	unix password sync = yes	Synchronise le mot de passe entre le compte unix et le compte samba. Fonctionne uniquement avec la commande smbpasswd. La directive n'est pas prise en compte par la commande tdbedit.
obey pam restrictions	obey pam restrictions = yes	Applique les restrictions PAM.

La commande smbclient

La commande **smbclient** permet d'accéder à des ressources Windows (ou Samba) depuis le monde Unix.

Syntaxe de la commande smbclient

```
smbclient '//serveur/partage' -U utilisateur
```

Exemple :

```
[root]# smbclient \\\\stat-wind\\partage-wind -U alain
smb: |> help
```

ou :

```
[root]# smbclient '//stat-wind/partage-wind' -U alain
smb: |> help
```

Le programme smbclient est couramment utilisé pour créer un interpréteur de type 'ftp' permettant ainsi d'accéder à des ressources SMB réseau.

Lister les partages :

```
[root]# smbclient -L drinf-mcsnlf01v
```

Se connecter à un partage data du serveur drinf-mcsnlf01v avec l'utilisateur bob :

```
[root]#  smbclient -L //drinf-mcsnlf01v/data -U bob
Enter bob's password:
```

Chapitre 4. Serveur web

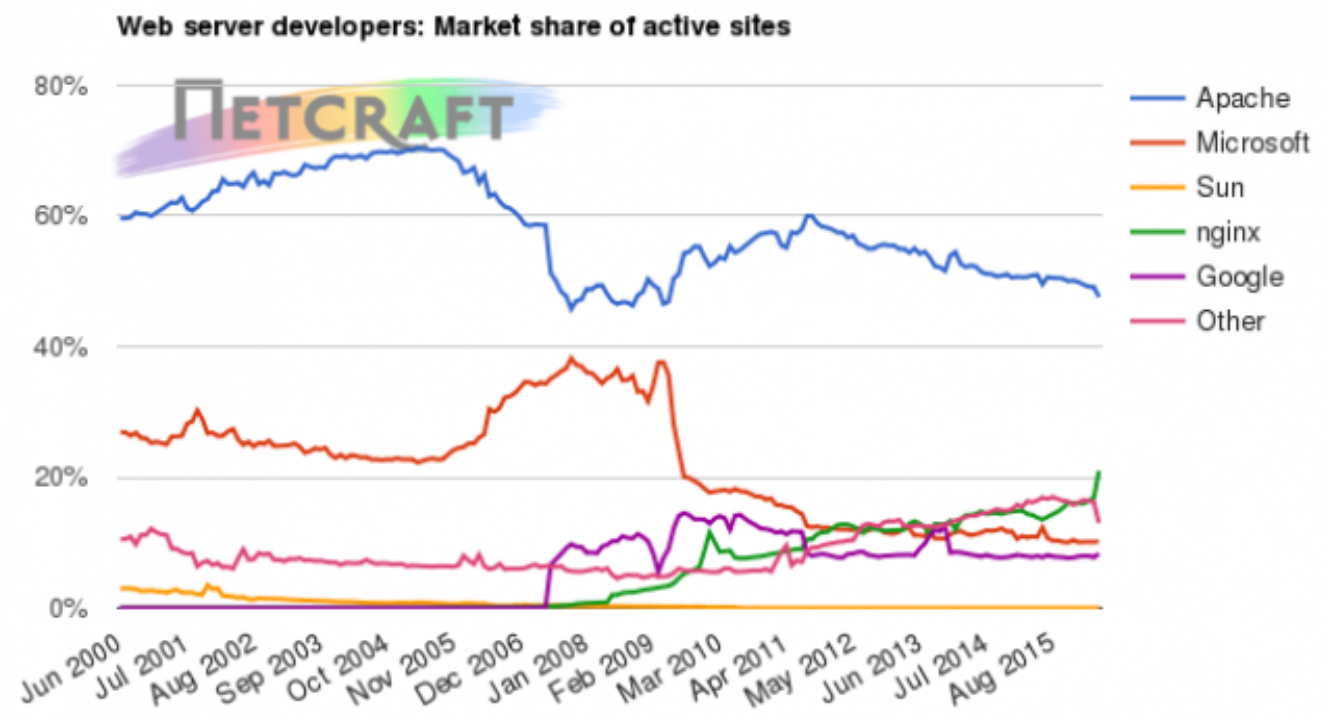
Le serveur **HTTP Apache** est le fruit du travail d'un groupe de volontaires : The Apache Group. Ce groupe a voulu réaliser un serveur Web du même niveau que les produits commerciaux mais sous forme de **logiciel libre** (son code source est disponible).

L'équipe d'origine a été rejointe par des centaines d'utilisateurs qui, par leurs idées, leurs tests et leurs lignes de code, ont contribué à faire d'Apache le plus utilisé des serveurs Web du monde.

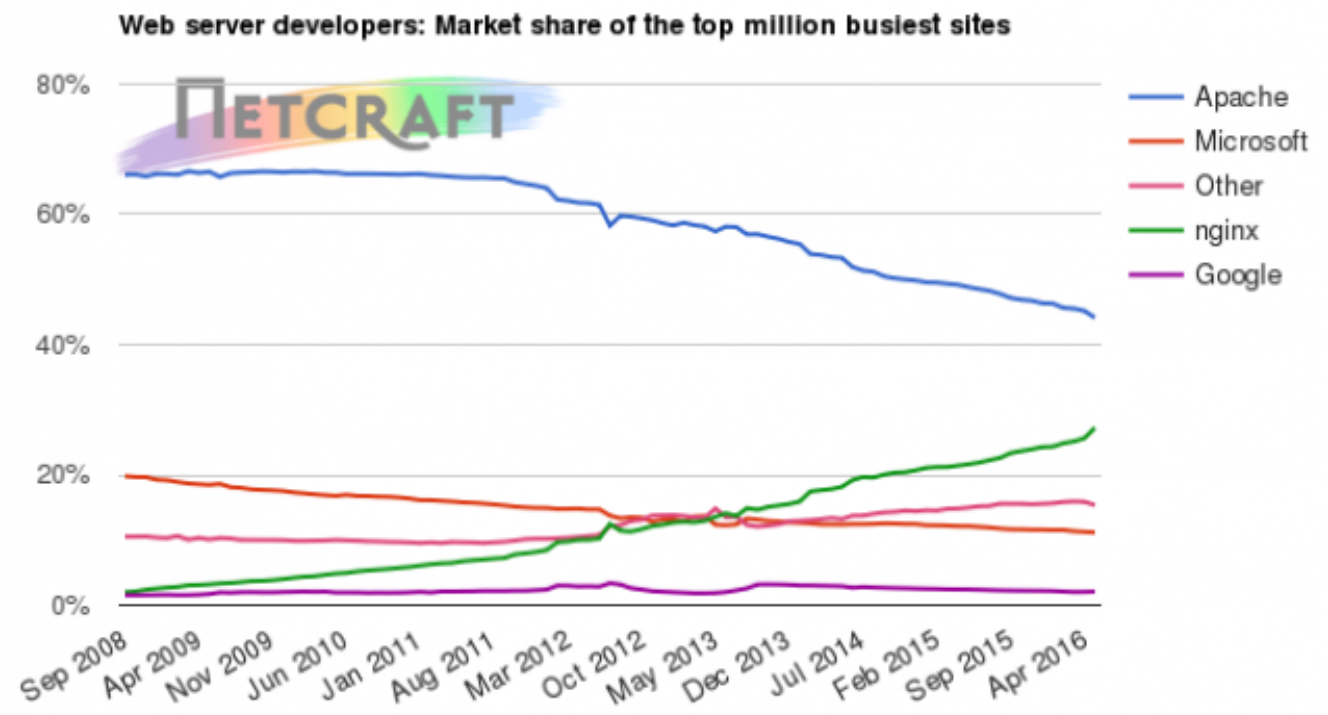
L'ancêtre d'Apache est le serveur libre développé par le National Center for Supercomputing Applications de l'université de l'Illinois. L'évolution de ce serveur s'est arrêtée lorsque le responsable a quitté le NCSA en 1994. Les utilisateurs ont continué à corriger les bugs et à créer des extensions qu'ils distribuaient sous forme de "patches" d'ou le nom "a patchee server".

La version 1.0 de Apache a été disponible le 1 décembre 1995 (il y a plus de 20 ans !).

L'équipe de développement se coordonne par l'intermédiaire d'une liste de diffusion dans laquelle sont proposées les modifications et discutées les évolutions à apporter au logiciel. Les changements sont soumis à un vote avant d'être intégrés au projet. Tout le monde peut rejoindre l'équipe de développement, il suffit de contribuer activement au projet pour pouvoir être nommé membre de The Apache Group.



Le serveur Apache est très présent sur l'Internet, puisqu'il représente encore environ 50% des parts de marché pour l'ensemble des sites actifs.



Les parts de marché perdues par Apache sont prises par son plus grand challenger : le serveur nginx. Ce dernier, plus rapide pour délivrer les pages web, et moins complets fonctionnellement parlant que le géant Apache.

4.1. Le protocole HTTP

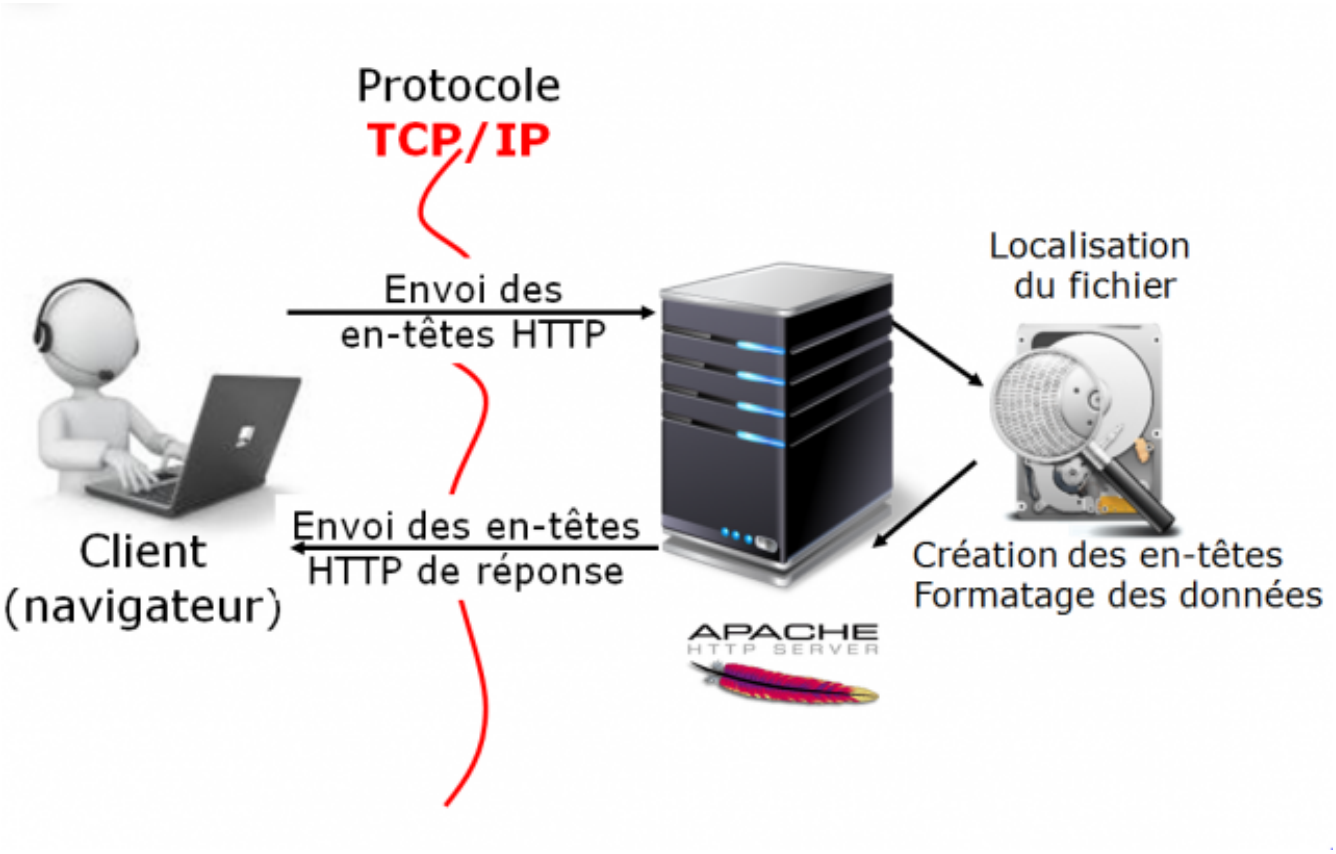
Le protocole **HTTP** (HyperText Transfer Protocol) est le protocole le plus utilisé sur Internet depuis 1990.

Ce protocole permet un transfert de fichiers (essentiellement au format HTML, mais aussi au format CSS, JS, AVI…) localisés grâce à une chaîne de caractères appelée URL entre un navigateur (le client) et un serveur Web (appelé d'ailleurs httpd sur les machines UNIX).

HTTP est un protocole "requête - réponse" opérant au dessus de TCP (Transmission Control Protocol).

1. Le client ouvre une connexion TCP vers le serveur et envoie une requête.
2. Le serveur analyse la requête et répond en fonction de sa configuration.

Le protocole HTTP est en lui même dit "**STATELESS**" : il ne conserve pas d'information sur l'état du client d'une requête à l'autre. Ce sont les langages dynamiques comme le php, le python ou le java qui vont permettre la conservation en mémoire des informations de session d'un client (comme dans le cadre d'un site de e-commerce par exemple).



Le protocole HTTP est en version 1.1. La version 2 est en cours de déploiement.

Une réponse HTTP est un ensemble de lignes envoyées au navigateur par le serveur. Elle comprend :

- **Une ligne de statut** : c'est une ligne précisant la version du protocole utilisé et l'état du traitement de la requête à l'aide d'un code et d'un texte explicatif. La ligne comprend trois éléments devant être séparés par un espace :
  - La version du protocole utilisé ;
  - Le code de statut ;
  - La signification du code .
- **Les champs d'en-tête de la réponse** : il s'agit d'un ensemble de lignes facultatives permettant de donner des informations supplémentaires sur la réponse et/ou le serveur. Chacune de ces lignes est composée d'un nom qualifiant le type d'en-tête, suivi de deux points (:) et de la valeur de l'en-tête.
- **Le corps de la réponse** : il contient le document demandé.

Voici un exemple de réponse HTTP :

Exemple de réponse HTTP

```
HTTP/1.1 200 OK
Date : Sat, 15 Jan 2016 14:37:12 GMT Server : Apache/2.17
Content-Type : text/HTML
Content-Length : 1245
Last-Modified : Fri, 14 Jan 2016 08:25:13 GMT
```

Le rôle du serveur web consiste à traduire une URL en ressource locale. Consulter la page http://www.free.fr/, revient à envoyez une requête HTTP à cette machine. Le service DNS joue donc un rôle essentiel.

Les

Une URL (**Uniform Ressource Locator** - littéralement “identifiant uniforme de ressources”) est une chaîne de caractères ASCII utilisée pour désigner les ressources sur Internet. Elle est informellement appelée adresse web.

Une URL est divisée en trois parties :

Composition d’une URL

```
<protocole>://<hôte>:<port>/<chemin>
```

- **Le nom du protocole** : il s’agit du langage utilisé pour communiquer sur le réseau. Le protocole le plus utilisé est le protocole HTTP (HyperText TransferProtocol), le protocole permettant d’échanger des pages Web au format HTML. De nombreux autres protocoles sont toutefois utilisables.
- **Identifiant et mot de passe** : permet de spécifier les paramètres d’accès à un serveur sécurisé. Cette option est déconseillée car le mot de passe est visible dans l’URL (dans le cadre de la sécurité).
- **L’hôte** : Il s’agit du nom de l’ordinateur hébergeant la ressource demandée. Notez qu’il est possible d’utiliser l’adresse IP du serveur, ce qui rend par contre l’URL moins lisible.
- **Le numéro de port** : il s’agit d’un numéro associé à un service permettant au serveur de savoir quel type de ressource est demandé. Le port associé par défaut au protocole est le port numéro 80. Ainsi, lorsque le service Web du serveur est associé au numéro de port 80, le numéro de port est facultatif.
- **Le chemin d’accès à la ressource** : Cette dernière partie permet au serveur de connaître l’emplacement auquel la ressource est située, c’est-à-dire de manière généralel’emplacement (répertoire) et le nom du fichier demandé. Si non renseignée, indique la première page de l’hôte. Sinon indique le chemin de la page à afficher.

Les ports

Une requête HTTP arrivera sur le port 80 (port par défaut pour http) du serveur fonctionnant sur l'hôte. L'administrateur peut toutefois choisir librement le port d'écoute du serveur.

Le protocole http se décline en une version sécurisée: le protocole https (port 443). Ce protocole chiffré s'implémente à partir du module mod-ssl.

D'autres ports peuvent être utilisés, comme le port 8080 (serveurs d'applications Java EE) ou le port 10 000 (Serveur webmin).

4.2. Installation du serveur

Apache est **multiplateforme**. Il peut être utilisé sur Linux, Windows, Mac…

L'administrateur devra choisir entre deux méthodes d'installation :

- **Installation par paquets** : l'éditeur de la distribution fourni des versions **stables et soutenues** (mais parfois anciennes) ;
- **Installation depuis les sources** : le logiciel apache est compilé, l'administrateur peut spécifier les options qui l'intéressent ce qui permet l'optimisation du service. Apache fournissant une architecture modulaire, la re-compilation du logiciel apache n'est généralement pas nécessaire pour ajouter ou supprimer des fonctionnalités complémentaires (ajout/suppression de modules).

Le choix de la méthode d'installation par paquets est fortement **conseillé**. Des dépots complémentaires permettent d'installer des versions plus récentes d'apache sur des versions de distributions anciennes mais, en cas de problème, RedHat n'apportera pas son soutien.

Exemples de modules et de leurs rôles respectifs :

- **mod_access** : filtre l'accès des clients par leur nom d'hôte, adresse IP ou autre caractéristique
- **mod_alias** : permet la création d'alias ou répertoires virtuels
- **mod_auth** : authentifie les clients
- **mod_cgi** : exécute les scripts CGI
- **mod_info** : fournit des informations sur l'état du serveur
- **mod_mime** : associe les types de fichiers avec l'action correspondante
- **mod_proxy** : propose un serveur proxy (serveur mandataire)
- **mod_rewrite** : réécrit les URL
- …

Installation par rpm

Interroger la base de données des “rpm”

```
[root]# rpm –qa "http*"
```

Installez à partir de paquets liés à la distribution

```
[root]# rpm –ivh httpd-xxx.rpm
```

Installer si nécessaire les dépendances demandées.

Installation par yum

Si vous avez à disposition un dépôt yum :

```
[root]# yum install httpd
```

Lancer Apache au démarrage du serveur :

```
[root]# chkconfig httpd on
```

Avant de se lancer dans une installation, il est important de savoir si une version du serveur Apache est installée :

```
rpm –qa "http*"
```

Lors de l’installation, un groupe apache et un utilisateur apache sont créés.

```
[root]# grep apache /etc/group
apache:x:48

[root]# grep apache /etc/passwd
apache:x:48:48:Apache:/var/www:/sbin/nologin

[root]# grep apache /etc/shadow
apache:!!:14411::::::
```

Le serveur Apache travaille sous l’identité d’un utilisateur “apache” appartenant à un groupe “apache”. Lors de l’installation, ce groupe et cet utilisateur sont créés. L’utilisateur apache étant un utilisateur système, aucun mot de passe ne lui est attribué, ce qui rend impossible la connexion au

système avec cet utilisateur.

Les fichiers de configuration

Le répertoire */etc/httpd/conf/* contient le fichier de configuration principal d'Apache httpd.conf. Ce fichier est très bien commenté. En général, ces commentaires suffisent pour éclairer l'administrateur sur les options dont il dispose.

Il peut être judicieux de créer un fichier de configuration sans commentaires :

```
[root]# egrep –v '^#|^$' /etc/httpd/conf/httpd.conf > httpd.conf.lite
```

Le répertoire /etc/httpd/conf.d/ contient les fichiers de configuration des sites virtuels ou des modules installés (ssl, php, welcome.conf, phpmyadmin, etc.)

Manuel d'utilisation

Il existe un package contenant un site faisant office de manuel d'utilisation d'apache. Il s'agit du package **httpd-manual-xxx.noarch.rpm**. Une fois ce package installé vous pourrez accéder au manuel simplement avec un navigateur web à cette adresse http://127.0.0.1/manual.

Lancement du serveur

- Par le script d'init :

```
[root]# /etc/rc.d/init.d/httpd {start|restart|status
```

- Avec la commande service :

```
[root]# service httpd {start|restart|status}
```

- Avec la commande apachectl fourni par apache :

```
[root]# apachectl {start|restart|stop}
```

Il est indispensable de lancer/relancer le serveur :

- après l'installation ;
- après toute modification de la configuration.

Parefeu

Le pare-feu “iptables” interdit l’accès aux ports 80 et 443. Pensez à le configurer (ou à désactiver ce service sur plateforme de test) !

```
[root]# system-config-firewall-tui
```

Ou :

```
[root]# service iptables stop
[root]# service ip6tables stop
```

La configuration d’un pare-feu fait l’objet d’un autre cours.

4.3. Arborescence

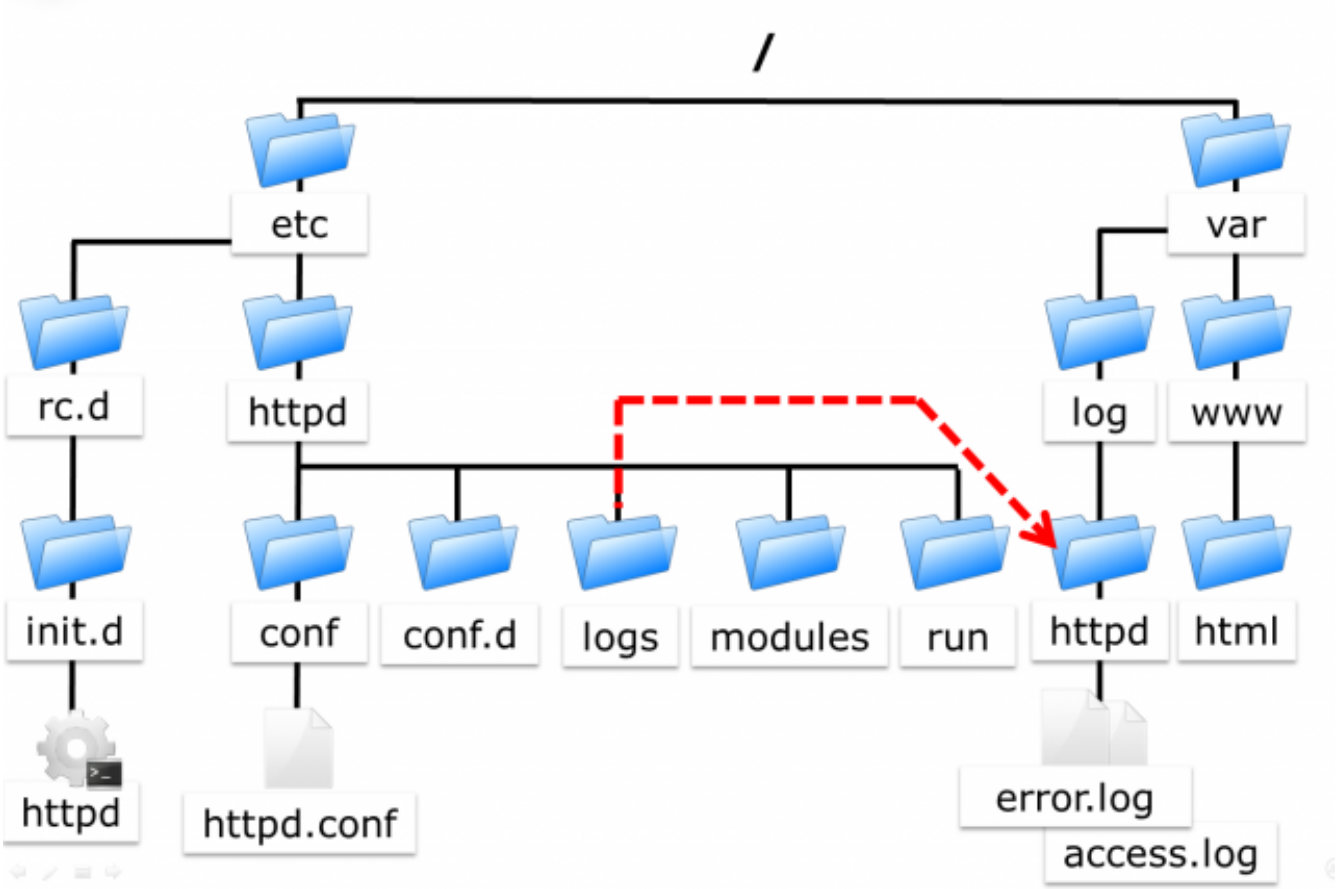


Figure 44. Arborescence d’Apache

L’arborescence peut varier en fonction des distributions.

- **/etc/httpd/** : Ce répertoire est la racine du serveur. Il contient l’ensemble des fichiers du serveur Apache.
- **/etc/httpd/conf/** : Ce répertoire contient l’ensemble des fichiers de configuration du serveur. Il

possède des sous-dossiers pour des éléments de configuration précis.

- **/var/www/html/** : Ce répertoire est le répertoire de publication par défaut. Il contient les fichiers nécessaires à l'affichage de la page web par défaut du serveur Apache. Quand l'administrateur veut publier un site, il peut déposer ses fichiers dans ce répertoire.

D'autres répertoires existent sous /var/www :

- **cgi-bin** : contient les scripts CGI ;
- **icons** : contient des icônes, notamment celles pour identifier le type de fichier ;
- **error** : contient les messages d'erreur d'Apache, ce sont ces fichiers qu'il faudra modifier pour personnaliser les mesages d'erreur.
- **/var/log/httpd/** : Ce répertoire contient les fichiers de logs du serveur Apache.
  - Le fichier access-log garde une trace des différents accès au serveur ;
  - Le fichier error-log contient la liste des erreurs rencontrées pendant l'exécution du service ;
  - Les fichiers logs sont personnalisables, l'administrateur peut aussi en créer de nouveaux.
- **/etc/httpd/modules** : Répertoire contenant les liens vers le répertoire "/usr/lib/httpd/modules" contenant les modules utilisables par Apache. Un module est une extension logicielle d'Apache, lui permettant par exemple d'interpréter le PHP (ex: mod-php5.so).
- **/etc/rc.d/init.d/httpd** : Script de démarrage du serveur httpd.

4.4. Configuration du serveur

La configuration globale du serveur se fait dans /etc/httpd/conf/httpd.conf.

Ce fichier est découpé en 3 sections qui permettent de configurer :

- en **section 1** l'environnement global ;
- en **section 2** le site par défaut et les paramètres par défaut des sites virtuels ;
- en **section 3** les hôtes virtuels.

L'**hébergement virtuel** permet de mettre en ligne **plusieurs sites virtuels** sur le même serveur. Les sites sont alors différenciés en fonction de leurs noms de domaines, de leurs adresses IP, etc.

La modification d'une valeur en section 1 ou 2 impacte l'ensemble des sites hébergés.

En environnement mutualisé, les modifications seront donc effectuées en section 3.

Pour faciliter les mises à jour futures, il est vivement recommandé de créer un fichier de configuration section 3 pour chaque site virtuel.

Section 1

Les différentes directives rencontrées en section 1 sont :

Table 86. Directives principales de la section 1

Directives	Observations
ServerTokens	Cette directive sera vue dans le cours Apache – sécurité.
ServertRoot	Indique le chemin du répertoire contenant l'ensemble des fichiers constituant le serveur Apache.
PidFile	Le fichier cible de la directive contient le numéro de PID du serveur à son démarrage.
Timeout	Le nombre de secondes avant le délai d'expiration d'une requête trop longue (entrante ou sortante).
KeepAlive	Connexion persistante (plusieurs requêtes par connexion TCP).
MaxKeepAliveRequests	Nombre maximum de connexions persistantes.
KeepAliveTimeout	Nombre de secondes à attendre la requête suivante du client avant fermeture de la connexion TCP.
Listen	Permettre à apache d'écouter sur des adresses ou des ports spécifiques.
LoadModule	Charger des modules complémentaires (moins de modules = plus de sécurité).
Include	Inclure d'autres fichiers de configuration au serveur.
ExtendedStatus	Afficher plus d'information sur le serveur dans le module server-status.
User et Group	Permet de lancer les processus apache avec différents utilisateurs. Apache se lance toujours en tant que root puis change son propriétaire et son groupe.

Le serveur Apache a été conçu comme un serveur puissant et flexible, pouvant fonctionner sur une grande variété de plateformes.

Plateformes différentes et environnements différents signifient souvent fonctionnalités différentes, ou utilisation des méthodes méthodes pour implémenter la même fonctionnalité le plus efficacement possible.

La conception modulaire d'apache autorise l'administrateur à choisir quelles fonctionnalités seront incluses dans le serveur en choisissant les modules à charger soit à la compilation, soit à l'exécution.

Cette modularité comprend également les fonctions les plus élémentaires du serveur web.

Certains modules, les Modules Multi-Processus (MPM) sont responsables de l'association aux ports réseau de la machine, acceptent les requêtes, et se chargent de les répartir entre les différents processus enfants.

Pour la version d'Apache de Windows, le MPM utilisé sera mpm-winnt.

Sous Linux, les sites très sollicités utiliseront un MPM threadé comme worker ou event, tandis que les sites privilégiant la stabilité utiliseront prefork.

Voir la page http://httpd.apache.org/docs/2.2/fr/mpm.html

Configuration par défaut des modules prefork et worker :

Configuration des modules dans /etc/http/conf/httpd.conf

```
<IfModule prefork.c>
StartServers        8
MinSpareServers     5
MaxSpareServers     20
ServerLimit     256
MaxClients      256
MaxRequestPerChild  4000
</IfModule>

<IfModule worker.c>
StartServers        4
MaxClients      300
MinSpareThreads     25
MaxSpareThreads     75
ThreadsPerChild     25
MaxRequestsPerChild 0
</IfModule>
```

Le module Prefork, activé par défaut, s'appuie sur des processus. Il est donc plus stable mais nécessite plus de mémoire pour fonctionner. Le module Worker, quand à lui, s'appuie sur des threads, ce qui le rend plus performant, mais des modules comme le Php ne sont pas compatible.

Choisir un module plutôt qu'un autre est donc une tâche complexe, tout autant que l'optimisation du module MPM retenu (nombre de client, de requêtes, etc.).

Par défaut Apache est configuré pour un service moyennement sollicité (256 clients max).

La configuration minimal d'un serveur apache ressemble à ceci :

Configuration d'apache minimal dans /etc/httpd/conf/httpd.conf

```
ServerRoot /etc/httpd
KeepAlive On
Listen 80
Listen 160.210.150.50:1981
LoadModule ...
Include conf.d/*.conf
User apache
Group apache
```

Attention par défaut la sécurité via SELinux est active. Elle empêche la lecture d'un site sur un

autre répertoire que “/var/www/”.

Le répertoire contenant le site doit posséder le contexte de sécurité httpd_sys_content_t.

Le contexte actuel se vérifie par la commande :

```
[root]# ls –Z /rep
```

Rajouter le contexte via la commande :

```
[root]# chcon -vR --type=httpd_sys_content_t /rep
```

Elle empêche également l’ouverture d’un port non standard. Il faut ouvrir manuellement le port désiré à l’aide de la commande semanage (non installée par défaut).

```
[root]# semanage port -a -t http_port_t -p tcp 1664
```

Directives User et Group

Définir un compte et un groupe de gestion d’Apache

Historiquement, apache était lancé par root, ce qui posait des problèmes de sécurité. Apache est toujours lancé par root mais change ensuite son identité. Généralement User Apache et Group Apache.

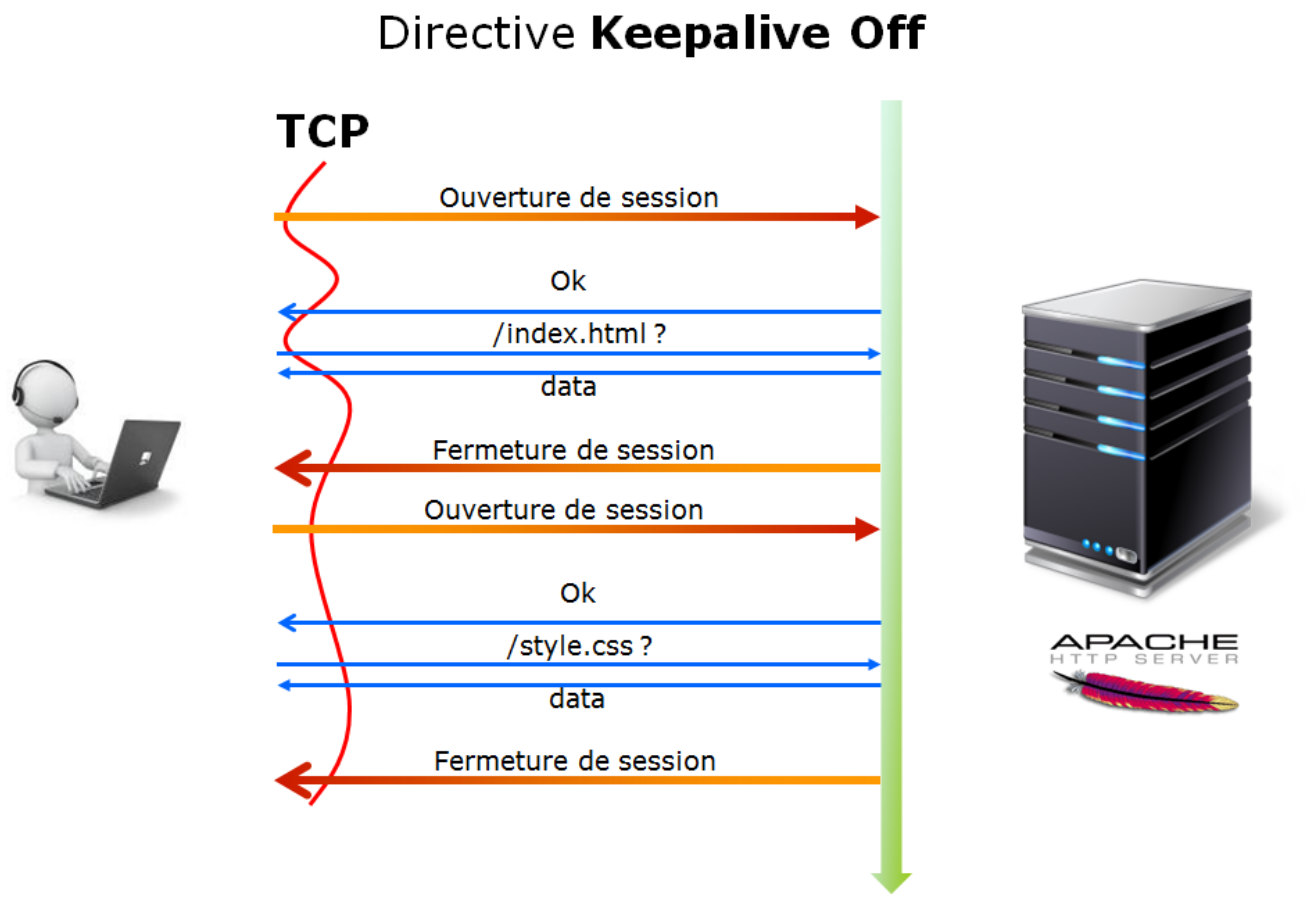
Attention jamais ROOT !!!

Le serveur Apache (processus httpd) est lancé par le compte de super-utilisateur root. Chaque requête d’un client déclenche la création d’un processus “fils”. Pour limiter les risques, il faut lancer ces processus enfants avec un compte moins privilégié.

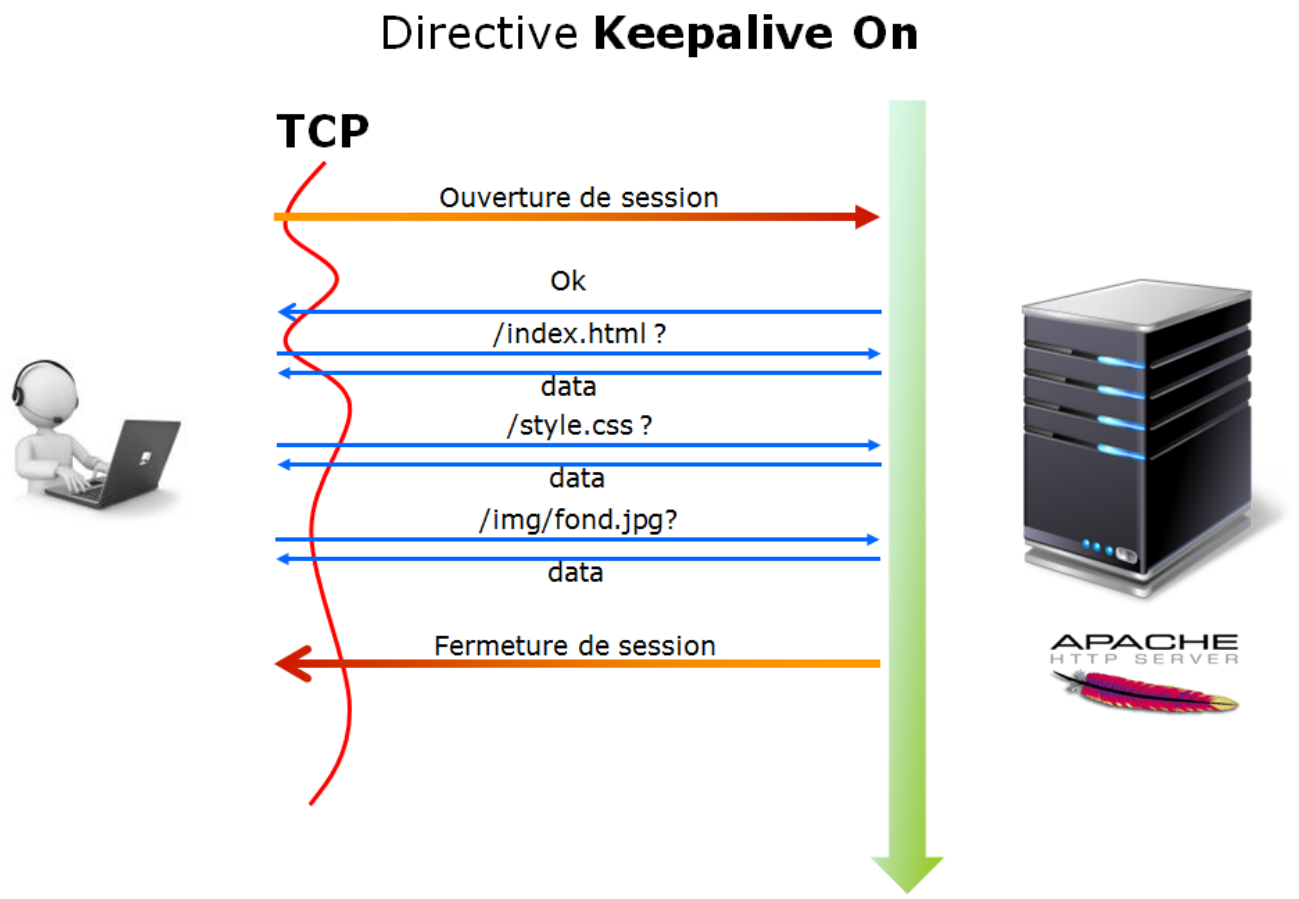
Les directives User et Group servent à déclarer le compte et le groupe utilisés pour la création des processus enfants.

```
User apache
Group apache
```

Ce compte et ce groupe doivent avoir été créés dans le système (par défaut cela est fait à l’installation). Par mesure de précaution supplémentaire, s’assurer que le compte n’est pas interactif (ne peut pas ouvrir de session).



Avec la directive KeepAlive désactivée, chaque demande de ressource sur le serveur nécessite une ouverture de connexion TCP, ce qui est long à effectuer d'un point de vue réseau et gourmand en ressource système.



Avec la directive KeepAlive à On, le serveur conserve la connexion ouverte avec le client le temps du Keepalive.

Sachant qu'une page web est constituée de plusieurs fichiers (images, feuilles de styles, javascripts, etc.), cette stratégie est rapidement gagnante.

Il est toutefois nécessaire de bien paramétrer cette valeur au plus juste :

- Une valeur trop courte pénalise le client,
- Une valeur trop longue pénalise les ressources du serveur.

Des demandes de configuration spécifiques peuvent être faîtes par le client en hébergement mutualisé. Auquel cas, les valeurs de KeepAlive seront paramétrées directement dans le VirtualHost du client ou au niveau du mandataire (ProxyKeepalive et ProxyKeepaliveTimeout).

L'affichage de cette page prouve que le serveur est fonctionnel. Mais le serveur ne dispose pas encore de site à publier. Paramétrons la section 2.

Section 2

La section 2 paramètre les valeurs utilisées par le serveur principal. Le serveur principal répond à toutes les requêtes qui ne sont pas prises en charge par un des Virtualhosts de la sections 3.

Les valeurs sont également utilisées comme valeur par défaut pour les sites virtuels.

- **ServerAdmin** : spécifie une adresse de messagerie qui apparaîtra dans certaines pages auto-générées, comme dans les pages d'erreurs.
- **ServerName** : spécifie le nom qui servira d'identification pour le serveur. Peut être déterminé automatiquement, mais il est recommandé de le spécifier explicitement (adresse IP ou nom DNS).
- **DocumentRoot** : spécifie le répertoire contenant les fichiers à servir aux clients. Par défaut **/var/www/html/**.
- **ErrorLog** : spécifie le chemin vers le fichier d'erreurs.
- **LogLevel** : debug, info, notice, warn, error, crit, alert, emerg.

- **LogFormat** : définir un format spécifique de log à utiliser avec la directive CustomLog.
- **CustomLog** : spécifie le chemin vers le fichier d'accès.
- **ServerSignature** : vue dans le cours sécurité.
- **Alias** : spécifie un répertoire extérieur à l'arborescence et le rend accessible par un contexte. La présence ou l'absence du dernier slash dans le contexte à son importance.
- **ScriptAlias** : spécifie le dossier contenant les scripts serveurs (idem alias) et les rend exécutables.
- **Directory** : spécifie des comportements et des droits d'accès par répertoire.
- **AddDefaultCharset** : spécifie le format d'encodage des pages envoyés (les caractères accentués peuvent être remplacés par des ?...).
- **ErrorDocument** : personnaliser les pages d'erreurs.
- **server-status** : rapport sur l'état du serveur.
- **server-info** : rapport sur la configuration du serveur.

La directive ErrorLog

La directive ErrorLog permet de définir le journal des erreurs.

Cette directive définit le nom du fichier dans lequel le serveur enregistre toutes les erreurs qu'il rencontre. Si le file-path n'est pas absolu, il est supposé être relatif à ServerRoot.

Syntaxe :

```
ErrorLog file-path
```

Exemple :

```
ErrorLog logs/error-log
```

La directive DirectoryIndex

La directive DirectoryIndex permet de définir la page d'accueil du site.

Cette directive indique le nom du fichier qui sera chargé en premier, qui fera office d'index du site ou de page d'accueil.

Syntaxe :

```
DirectoryIndex page-à-afficher
```

Le chemin complet n'est pas précisé car le fichier est recherché dans le répertoire spécifié par

DocumentRoot

Exemple :

```
DocumentRoot    /var/www/html
DirectoryIndex  index.php, index.htm
```

Cette directive indique le nom du fichier index du site web. L'index est la page par défaut qui s'ouvre quand le client tape l'URL du site (sans avoir à taper le nom de cet index). Ce fichier doit se trouver dans le répertoire indiqué par la directive DocumentRoot.

La directive DirectoryIndex peut spécifier plusieurs noms de fichiers index séparés par des espaces. Par exemple, une page d'index par défaut au contenu dynamique et en deuxième choix une page statique.

La directive ServerAdmin

La directive ServerAdmin permet d'indiquer le mail de l'administrateur.

Syntaxe :

```
ServerAdmin email-adresse
```

Exemple :

```
ServerAdmin root@etrs.terre.defense.gouv.fr
```

La balise Directory

La balise Directory permet de définir des directives propre à un répertoire.

Cette balise permet d'appliquer des droits à un ou plusieurs répertoires. Le chemin du répertoire sera saisi en absolu.

Syntaxe :

```
<Directory directory-path>
Définition des droits des utilisateurs
</Directory>
```

Exemple :

```
<Directory /home/SitesWeb/SiteTest>
Allow from all   # nous autorisons tout le monde
</Directory>
```

La section Directory sert à définir un bloc de consignes s'appliquant à une partie du système de fichiers du serveur. Les directives contenues dans la section ne s'appliqueront qu'au répertoire spécifié (et ses sous-répertoires).

La syntaxe de ce bloc accepte les caractères génériques mais il faudra préfèrer alors utiliser le bloc DirectoryMatch.

Dans l'exemple suivant, nous allons refuser l'accès au disque dur local du serveur quelque soit le client. Le répertoire « / » représente la racine du disque dur.

```
<Directory />
    Order deny, allow
    Deny from all
</Directory>
```

Dans l'exemple suivant, nous allons autoriser l'accès au répertoire de publication /var/www/html pour tous les clients.

```
<Directory /var/www/html>
    Order allow, deny
    Allow from all
</Directory>
```

Prise en compte des modifications

La commande apachectl permet de tester la syntaxe du fichier de conf :

```
[root]# service httpd configtest
```

ou :

```
[root]# apachectl -t
```

puis :

```
[root]# /etc/rc.d/init.d/httpd {start|restart|status}
```

ou :

```
[root]# service httpd {start|restart|status}
```

ou :

```
[root]# apachectl {start|restart|stop}
```

Les commandes précédentes ont pour effet de couper les connexions en cours. Apache propose une solution plus élégante, qui lance de nouveaux serveurs et attends la fin du timeout pour détruire les anciens processus :

Ne pas couper les connexions TCP actives :

```
[root]# service httpd graceful
```

4.5. Configuration avancée du serveur

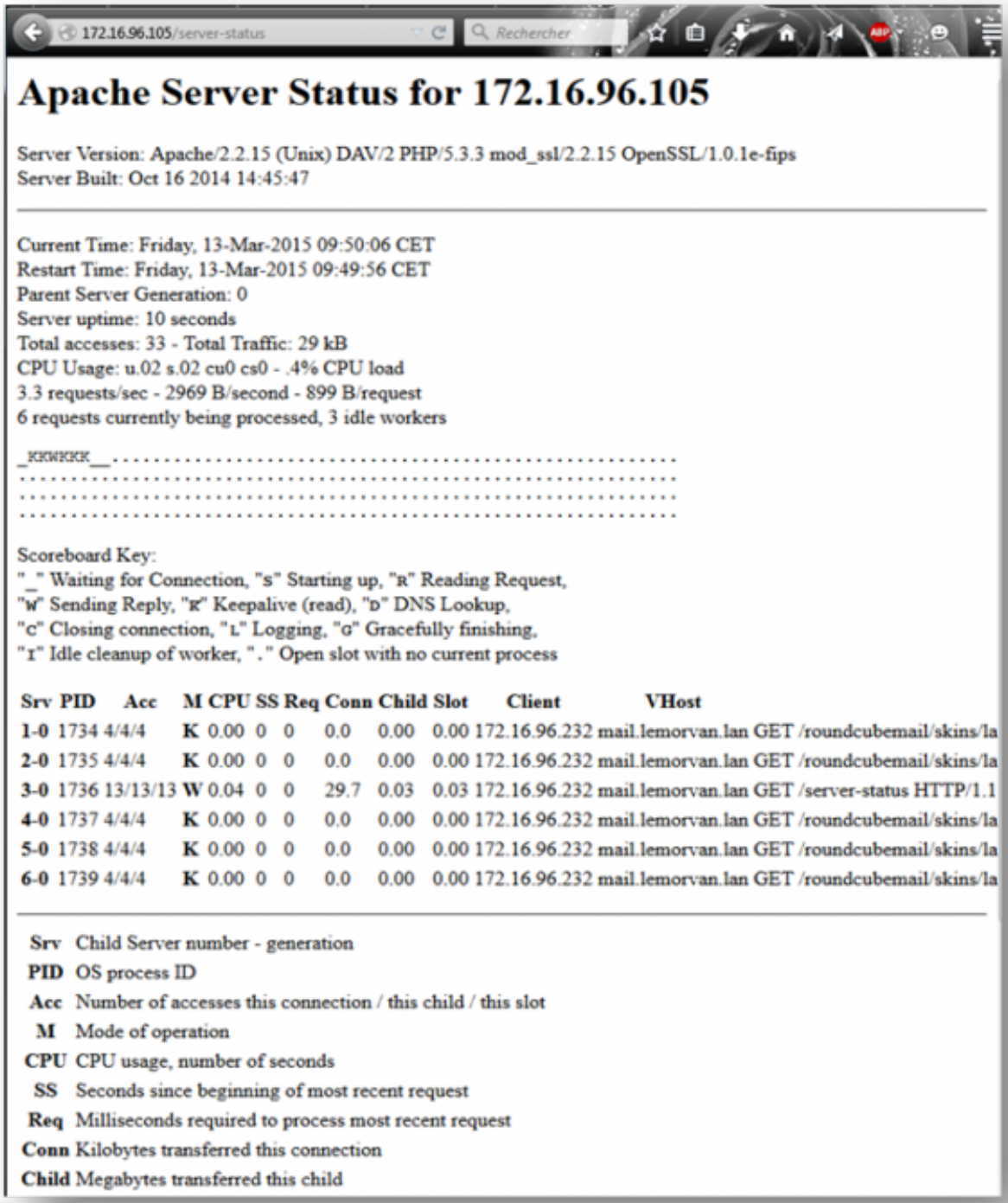
Le mod_status

Le mod_status permet d'afficher une page /server-status ou /server-info récapitulant l'état du serveur :

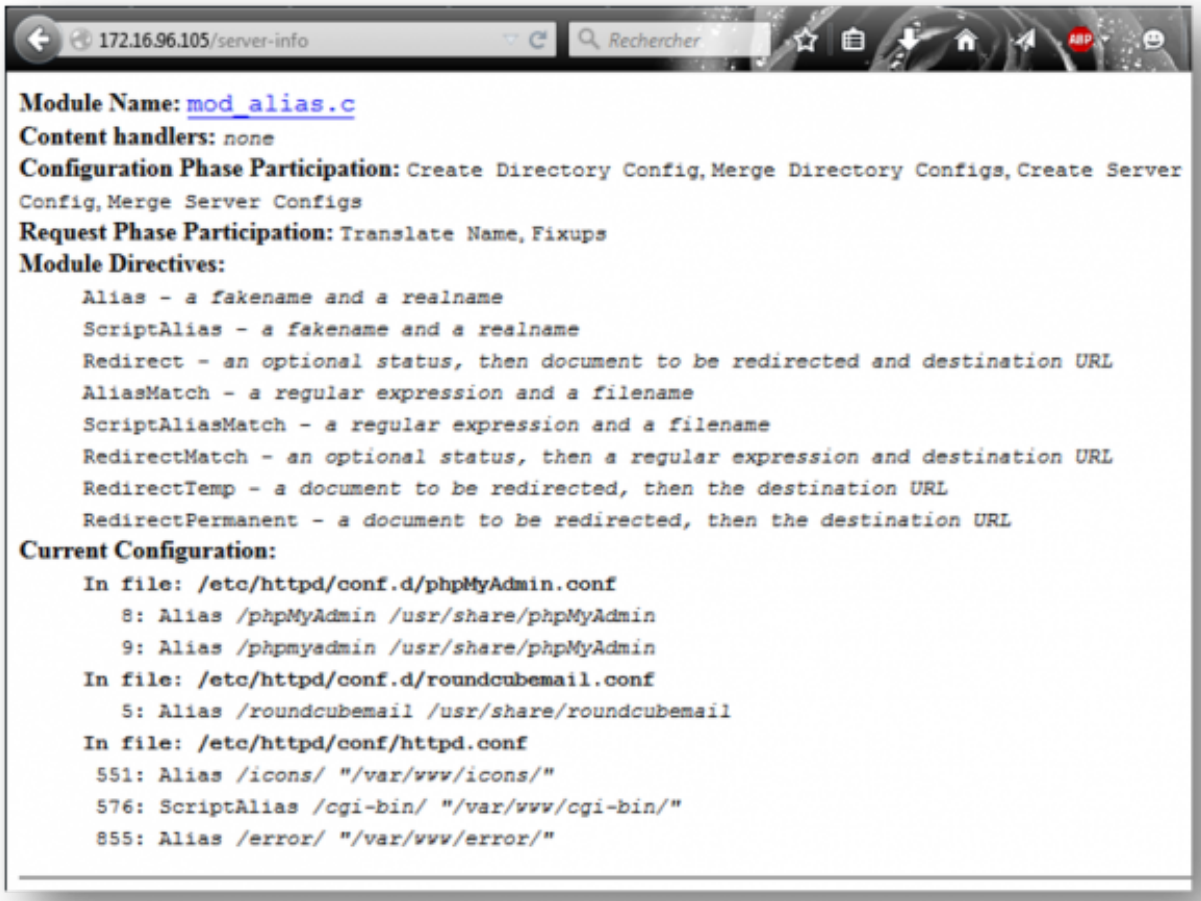
Configuration des directives server-status et server-info

```
<Location /server-status>
    SetHandler server-status
    Order allow,deny
    Allow from 127.0.0.1
</Location>

<Location /server-info>
    SetHandler server-info
    Order allow,deny
    Allow from 127.0.0.1
    Allow from 172.16.96.105
</Location>
```

La page /server-status :



La page /server-info :



Hébergement mutualisé (section 3)

Dans le cas d'un hébergement mutualisé, le client pense visiter plusieurs serveurs. En réalité, il n'existe qu'un seul serveur et plusieurs sites virtuels.

Pour mettre en place un hébergement mutualisé, il faut mettre en place des hôtes virtuels :

- en déclarant plusieurs ports d'écoute ;
- en déclarant plusieurs adresses IP d'écoute (hébergement virtuel par **IP**) ;
- en déclarant plusieurs noms de serveur (hébergement virtuel par **nom**);

Chaque site virtuel correspond à une arborescence différente.

La section 3 du fichier httpd.conf permet de déclarer ces hôtes virtuels.

Pour faciliter les mises à jour futures, il est vivement recommandé de créer un fichier de configuration section 3 pour chaque site virtuel.

Choisissez un hébergement virtuel "par IP" ou "par nom". En production, il est déconseillé de mixer les deux solutions.

- Chaque site virtuel peut être configuré dans un fichier indépendant ;
- Les VirtualHosts sont stockés dans /etc/httpd/conf.d/ ;
- L'extension du fichier est .conf.

La balise VirtualHost

La balise VirtualHost permet de définir des hôtes virtuels.

Syntaxe :

Syntaxe d'un fichier virtualhostXXX.conf

```
<VirtualHost adresse-IP[:port]>
    # si la directive "NameVirtualHost" est présente
    # alors "adresse-IP" doit correspondre à celle saisie
    # sous "NameVirtualHost" ainsi que pour le "port".
 ...
 </VirtualHost>
```

Si nous configurons le serveur Apache avec les directives de base vues précédemment, nous ne pourrons publier qu'un seul site. En effet, nous ne pouvons pas publier plusieurs sites avec les paramètres par défaut : même adresse IP, même port TCP et absence de nom d'hôte ou nom d'hôte unique.

L'usage des sites virtuels va nous permettre de publier plusieurs sites web sur un même serveur Apache. Nous allons définir des blocs qui décriront chacun un site web. Ainsi chaque site aura sa propre configuration.

Pour des facilités de compréhension, nous associons souvent un site web à une machine unique. Les sites virtuels ou hôtes virtuels (virtual hosts) sont appelés ainsi parce qu'ils dématérialisent le lien entre machine et site web.

Exemple 1 :

```
Listen 192.168.0.10:8080
<VirtualHost 192.168.0.10:8080>
  DocumentRoot /var/www/site1/
  ErrorLog /var/log/httpd/site1-error.log
</VirtualHost>
```

```
Listen 192.168.0.11:9090
<VirtualHost 192.168.0.11:9090>
  DocumentRoot /var/www/site2/
  ErrorLog /var/log/httpd/site2-error.log
</VirtualHost>
```

L'hébergement virtuel basé sur IP est une méthode permettant d'appliquer certaines directives en fonction de l'adresse IP et du port sur lesquels la requête est reçue. En général, il s'agit de servir différents sites web sur des ports ou des interfaces différents.

La directive NameVirtualHost

La directive NameVirtualHost permet de définir des hôtes virtuels à base de nom.

Cette directive est obligatoire pour configurer des hôtes virtuels à base de nom. Nous spécifions avec cette directive l'adresse IP sur laquelle le serveur recevra des demandes des hôtes virtuels à base de nom.

Syntaxe :

```
NameVirtualHost adresse-IP[:port]
```

Exemple :

```
NameVirtualHost 160.210.169.6:80
```

Il faut placer la directive avant les blocs descriptifs de sites virtuels. Elle désigne les adresses IP utilisées pour écouter les requêtes des clients vers les sites virtuels. La syntaxe est la suivante :

Pour écouter les requêtes sur toutes les adresses IP du serveur il faut utiliser le caractère *.

4.6. Exemple de publication de sites

Fichier /etc/httpd/conf.d/80-site1.conf :

```
<VirtualHost 160.210.69.6:80>
  # déclaration de l'arborescence du site
  DocumentRoot "/var/sitesweb/site1"
  # déclaration des index du site
  DirectoryIndex "Index1.htm"
  # déclaration des droits sur le site
  <Directory "/var/sitesweb/site1">
    Allow from all
  </Directory>
</VirtualHost>
```

Fichier /etc/httpd/conf.d/1664-site2.conf :

```
Listen 1664
<VirtualHost 160.210.69.6:1664>
  # déclaration de l'arborescence du site
  DocumentRoot "/var/sitesweb/site2"
  # déclaration des index du site
  DirectoryIndex "Index2.htm"
  # déclaration des droits sur le site
  <Directory "/var/sitesweb/site2">
    Allow from all
  </Directory>
</VirtualHost>
```

Chapitre 5. Serveur web Apache - LAMP - sous CentOS 7

Apache est le principal serveur web du monde de l'Open Source. À l'origine, c'était la continuation du serveur libre développé par le NCSA (National Center for Supercomputing Applications) à l'Université de l'Illinois. Lorsque le projet officiel a été abandonné en 1994, une équipe de développeurs volontaires a continué à fournir du code sous forme de nombreux correctifs, ce qui explique la genèse du nom a patchy server, c'est-à-dire "serveur rafistolé".

D'après les statistiques de Netcraft, un peu moins de la moitié des sites Web du monde tournent sur un serveur Apache. Ces dernières années, les parts de marché perdues par Apache sont reprises par Nginx, son principal concurrent, qui est orienté vers la performance tout en offrant moins de fonctionnalités.

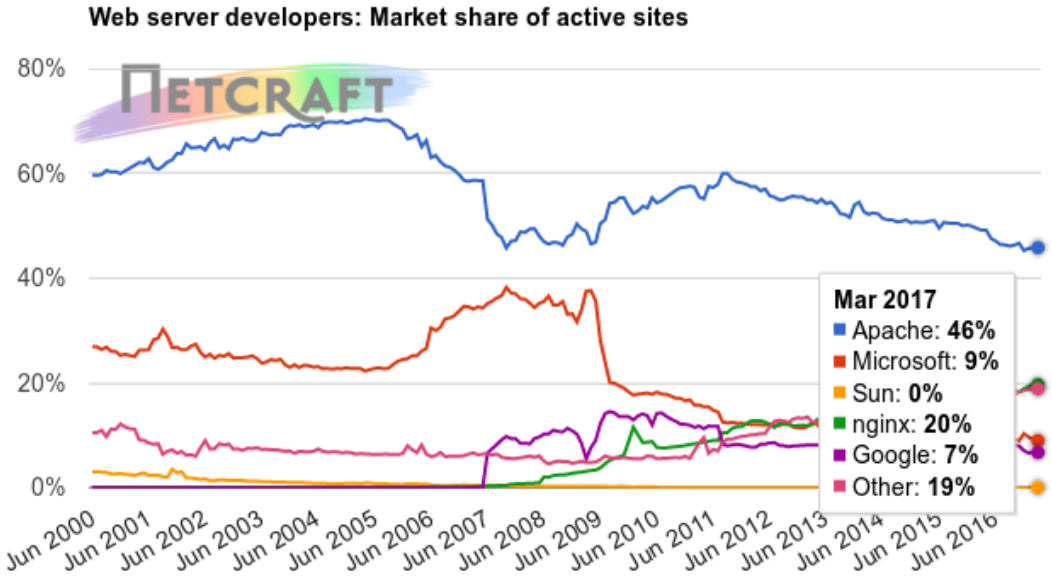


Figure 45. Statistiques NetCraft : sites actifs

Une installation typique d'Apache est généralement constituée d'un assemblage cohérent de paquets.

- le serveur Apache à proprement parler ;
- des bibliothèques diverses et variées ;
- des plug-ins ;
- des langages de programmation ;
- etc.

Ce cours décrit la configuration d'un serveur Web de type LAMP (***L**inux* + ***A**pache* + ***M**ySQL/**M**ariaDB* + ***P**HP*) sur CentOS 7.

5.1. Le protocole HTTP et les URL

Le protocole HTTP (***Hyper**text **T**ransfer **P**rotocol*, c'est-à-dire "protocole de transfert hypertexte") est un protocole de communication client-serveur développé pour le World Wide Web. Il permet un transfert de fichiers (html, css, js, mp4) localisés grâce à une chaîne de caractères appelée URL entre un navigateur (le client) et un serveur web.

HTTP est un protocole "requête-réponse" de la couche application qui utilise le protocole TCP comme couche de transport.

1. Le client ouvre une connexion TCP vers le serveur et envoie une requête.
2. Le serveur analyse la requête et répond en fonction de sa configuration.

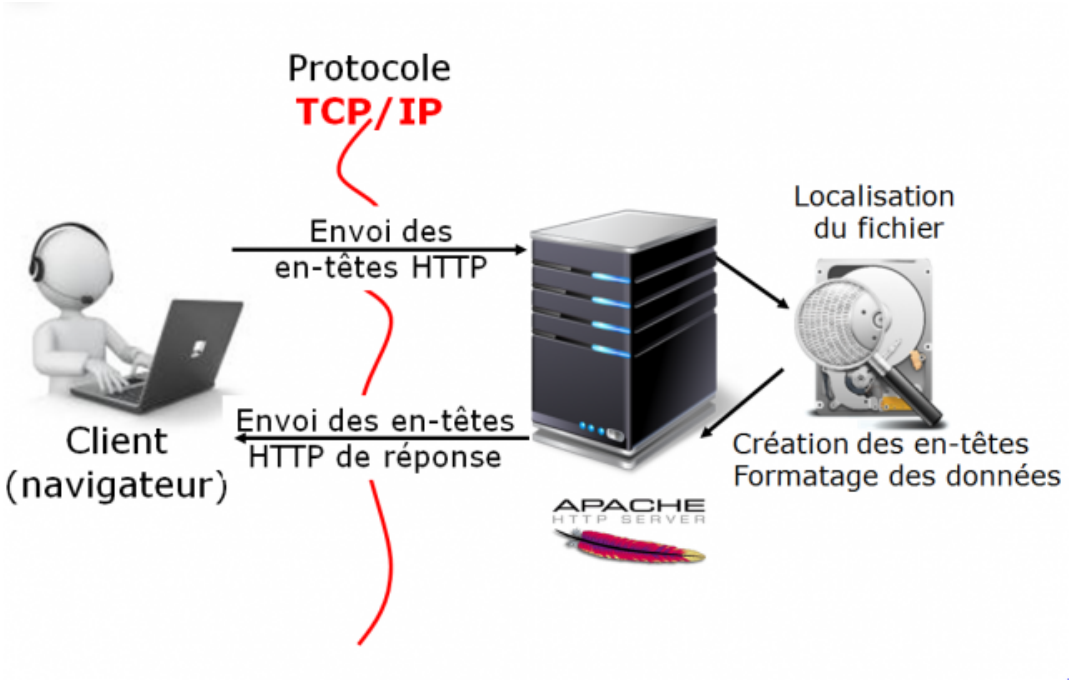


Figure 46. HTTP est un protocole client/serveur

Une réponse HTTP est un ensemble de lignes envoyées au client par le serveur. Elle comprend une ligne de statut, les champs d'en-tête et le corps de la réponse.

Voici un exemple de réponse HTTP :

```
$ curl --head --location https://www.formatux.fr
HTTP/1.1 200 OK
Date: Fri, 18 Aug 2017 18:44:18 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips mod_fcgid/2.3.9 PHP/5.4.16
mod_python/3.5.0- Python/2.7.5
X-Powered-By: PHP/7.0.21
Cache-Control: max-age=604800
Expires: Thu, 19 Nov 1981 08:52:00 GMT
domain=www.formatux.fr
Last-Modified: Sun, 23 Jul 2017 18:11:50 GMT
ETag: "faba58243b57482f98e0fa49387257e9"
Content-Type: text/html; charset=UTF-8
```

Le rôle du serveur web consiste à traduire une URL (comme par exemple `http://www.formatux.fr`) en ressource locale. Consulter la page `http://www.formatux.fr` revient à envoyer une requête HTTP à cette machine.

Une URL (***U**niform **R**esource **L**ocator*, autrement dit “identifiant uniforme de ressources”) est une chaîne de caractères ASCII utilisée pour désigner les ressources sur Internet. Elle est informellement appelée “adresse web” et elle est divisée en plusieurs parties, comme ceci.

```
<protocole>://<hôte>:<port>/<chemin>
```

Le protocole

Langage utilisé pour communiquer sur le réseau, comme par exemple `http`, `https`, `ftp`.

L’hôte

Ordinateur qui héberge la ressource demandée. Il est possible d’utiliser l’adresse IP (mais cela rend l’URL moins lisible).

Le numéro de port

Numéro associé à un service permettant de savoir quel type de ressource est demandé. Le port 80 est associé par défaut au protocole HTTP. Si l’on utilise ce port, ce n’est pas la peine de le spécifier explicitement.

Le chemin d’accès

Le chemin d’accès à la ressource permet au serveur de connaître l’emplacement du fichier demandé.

5.2. Ports et pare-feu

Apache utilise le port 80 en TCP pour le protocole HTTP. Il faudra donc songer à ouvrir ce port dans le pare-feu.

Notons que théoriquement, l'administrateur peut choisir librement le port d'écoute du serveur.

5.3. Installation

Le serveur Apache est fourni par le paquet httpd.

```
# yum install httpd
```

L'installation du paquet crée un utilisateur système apache et un groupe système apache correspondant.

```
# grep apache /etc/passwd
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
# grep apache /etc/group
apache:x:48:
# grep apache /etc/shadow
apache:!!:17352::::::
```

5.4. Premier lancement du serveur

Sous Red Hat et CentOS, Apache est préconfiguré pour afficher une page statique par défaut. Il suffit d'activer et de lancer le service.

```
# systemctl enable httpd
# systemctl start httpd
```

Tester le bon fonctionnement du serveur.

```
$ links http://localhost
```

On doit voir quelque chose de ce genre.

```
                    Testing 123..
This page is used to test the proper operation of the Apache HTTP
server after it has been installed. If you can read this page it
means that this site is working properly. This server is powered
by CentOS.
```

Dans le réseau local, ouvrir l'adresse IP du serveur avec un navigateur.

`http://192.168.1.10`

On peut également invoquer le nom d'hôte.

`http://stagiaire.formatux.fr`

Sur un serveur dédié, on essaiera successivement l'adresse IP, le nom de domaine et l'alias associé.

`http://172.16.100.1` `http://stagiaire1.formatux.fr` `http://www.stagiaire1.lan`

Voici à quoi ressemble la page par défaut dans un navigateur graphique.

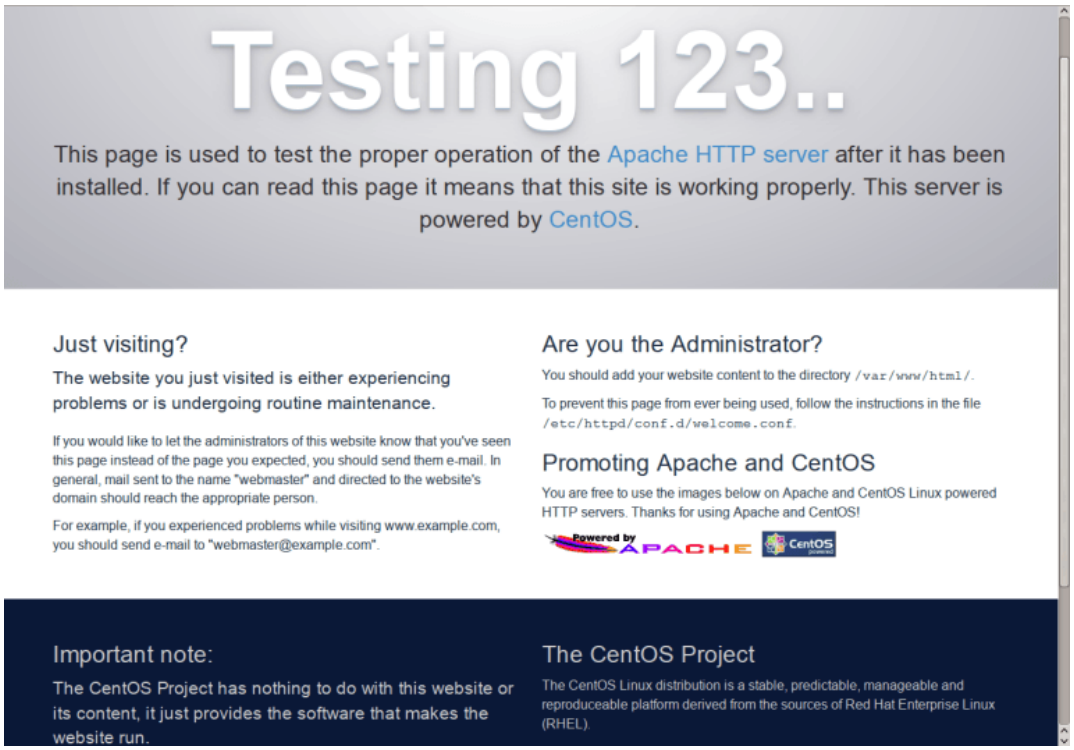


Figure 47. La page de test Apache

5.5. Les fichiers de configuration

Initialement, la configuration du serveur Apache s'effectuait dans un seul fichier `/etc/httpd/conf/httpd.conf`. Avec le temps, ce fichier est devenu de plus en plus volumineux et de moins en moins lisible.

Les distributions modernes ont donc tendance à répartir la configuration d'Apache sur une série de fichiers `*.conf` répartis dans les répertoires `/etc/httpd/conf.d` et `/etc/httpd/conf.modules.d`, rattachés au fichier principal `/etc/httpd/conf/httpd.conf` par la directive `Include`.

Le fichier `/etc/httpd/conf/httpd.conf` est amplement documenté. Pour commencer, nous allons sauvegarder ce fichier par défaut et créer une version dépourvue de commentaires et plus lisible. Ici, la commande `egrep` filtre les lignes qui commencent soit par un commentaire (`^#`), soit par un espace (`^$`), soit par quatre espaces suivis d'un commentaire.

```
# cd /etc/httpd/conf
# cp httpd.conf httpd.conf.orig
# grep -Ev '^#|^$|^    #' httpd.conf.orig > httpd.conf
```

5.6. La configuration par défaut

Jetons un oeil sur le fichier `httpd.conf` par défaut et regardons de plus près les principales directives qui le constituent.

La directive `ServerRoot` permet de définir le répertoire dans lequel le serveur est installé.

```
ServerRoot "/etc/httpd"
```

La directive `Listen` permet à Apache d'écouter sur des adresses ou des ports spécifiques. Notons que cette directive est requise. Si elle est absente du fichier de configuration, Apache refuse de démarrer.

```
Listen 80
```

Comme nous l'avons vu un peu plus haut, la directive `Include` permet l'inclusion d'autres fichiers de configuration dans le fichier de configuration principal du serveur. `IncludeOptional` fonctionne comme `Include`, au détail près que la directive ne produira pas une erreur au cas où le métacaractère `*` ne correspond à aucun fichier. Le chemin est relatif par rapport à l'emplacement spécifié dans la directive `ServerRoot`.

```
Include conf.modules.d/*.conf
...
IncludeOptional conf.d/*.conf
```

Apache ne doit pas être lancé en tant que `root`, mais en tant qu'utilisateur spécial défini par les directives `User` et `Group` dans `/etc/httpd/conf/httpd.conf`. Plus précisément, le processus principal est lancé par `root` qui lance ensuite des processus enfant avec l'utilisateur et le groupe configurés.

```
User apache
Group apache
```

L'adresse mail de l'administrateur, définie par la directive `ServerAdmin`, apparaîtra sur certaines pages générées par le serveur, notamment les pages d'erreur.

```
ServerAdmin root@localhost
```

Les balises `<Directory>` et `</Directory>` permettent de regrouper un ensemble de directives qui ne s'appliquent qu'au répertoire précisé, à ses sous-répertoires et aux fichiers situés dans ces sous-répertoires.

```
<Directory />
  AllowOverride none
  Require all denied
</Directory>
...
<Directory "/var/www">
  AllowOverride None
  Require all granted
</Directory>
<Directory "/var/www/html">
  Options Indexes FollowSymLinks
  AllowOverride None
  Require all granted
</Directory>
```

La directive `DocumentRoot` permet de définir le répertoire à partir duquel Apache va servir des fichiers.

```
DocumentRoot "/var/www/html"
```

Lorsque le serveur trouve un fichier `.htaccess`, il doit savoir quelles directives placées dans ce fichier sont autorisées à modifier la configuration préexistante. Le traitement des fichiers `.htaccess` est contrôlé par la directive `AllowOverride`, qui ne peut être utilisée que dans les sections `<Directory>`. À partir du moment où elle est définie à `none`, les fichiers `.htaccess` sont totalement ignorés.

```
<Directory />
  AllowOverride none
  Require all denied
</Directory>
```

La directive `Require` permet de contrôler l'accès au système de fichiers du serveur. `Require all denied` bloque l'accès pour tous les utilisateurs, `Require all granted` autorise tout le monde.

```
<Directory />
  AllowOverride none
  Require all denied
</Directory>
...
<Directory "/var/www">
  AllowOverride None
  Require all granted
</Directory>
```

Comme son nom l'indique, la directive `Options` permet d'activer ou de désactiver une série d'options (ou de comportements) pour un répertoire donné. Ici par exemple, l'option `Indexes` affiche la liste des fichiers d'un répertoire en cas d'absence de fichier `index.html`. `FollowSymlinks` permet de suivre les liens symboliques.

```
<Directory "/var/www/html">
  Options Indexes FollowSymLinks
  AllowOverride None
  Require all granted
</Directory>
```

Les balises `<IfModule>` et `</IfModule>` contiennent des directives qui ne s'appliquent qu'en fonction de la présence ou de l'absence d'un module spécifique. La directive `DirectoryIndex` spécifie le fichier à envoyer par Apache lorsqu'une URL se termine par `/` et concerne un répertoire entier.

```
<IfModule dir_module>
  DirectoryIndex index.html
</IfModule>
```

Les balises `<Files>` et `</Files>` contiennent des directives qui s'appliquent aux fichiers précisés. Ici par exemple, on interdit l'accès à tous les fichiers dont le nom commence par `.ht`, notamment `.htaccess`.

```
<Files ".ht*">
  Require all denied
</Files>
```

La directive `ErrorLog` définit le chemin vers le journal des erreurs. La verbosité de ce journal est contrôlée par la directive LogLevel.

```
ErrorLog "logs/error_log"
LogLevel warn
```

La directive `CustomLog` permet de contrôler la journalisation des requêtes destinées au serveur. Elle définit le nom et le format du fichier journal.

```
CustomLog "logs/access_log" combined
```

La directive `AddDefaultCharset` paramètre le jeu de caractères par défaut pour les pages de texte. Lorsqu'elle est désactivée (`AddDefaultCharset off`), Apache prend en compte l'encodage spécifié dans la balise `<meta>` des fichiers HTML à envoyer au navigateur.

```
meta http-equiv="Content-Type" content="text/html; charset=utf-8"
```

Ici en revanche, Apache utilise d'emblée le jeu de caractères spécifié en ignorant la balise `<meta>`.

```
AddDefaultCharset UTF-8
```

5.7. Configuration de base

Apache est immédiatement utilisable dans sa configuration par défaut. Avant d'héberger notre premier site, nous allons procéder à quelques ajustements.

Pour commencer, renseigner l'adresse mail de l'administrateur du serveur.

```
ServerAdmin contact@formatux.fr
```

Le nom du serveur peut être déterminé automatiquement, mais il vaut mieux le spécifier explicitement grâce à la directive `ServerName`.

```
ServerName stagiaire.formatux.fr
```

Sur un serveur dédié, on aura ceci.

```
ServerName sd-100246.dedibox.fr
```

Pour la journalisation, on choisira un format un peu moins bavard.

```
CustomLog "logs/access_log" common
```

Enfin, on permettra aux pages hébergées de spécifier leur propre encodage.

```
AddDefaultCharset off
```

Tester la nouvelle configuration :

```
# apachectl configtest
Syntax OK
```

Prendre en compte les modifications :

```
# systemctl reload httpd
```

5.8. Héberger un site statique

Dans la configuration par défaut, Apache est censé servir le contenu de `/var/www/html`. En l'absence de contenu, la page de test s'affiche, en fonction de la configuration prédéfinie dans `/etc/httpd/conf.d/welcome.conf`.

Pour nous épargner la corvée de créer du contenu fictif, nous pouvons récupérer un site web existant. On choisira la documentation de Slackware, qui vient sous forme d'une série de pages HTML statiques.

```
# cd /var/www/html/
# wget -r -np -nH --cut-dirs=1 http://www.slackbook.org/html/
```

Ouvrir le site dans un navigateur (Firefox, Links, Lynx) et apprécier le résultat.

5.9. Apache et les permissions de fichiers

Apache ne doit pas être lancé en tant que root, mais en tant qu'utilisateur spécial défini par les directives `User` et `Group` dans `/etc/httpd/conf/httpd.conf`.

```
User apache
Group apache
```

La règle de sécurité générale veut que les contenus du serveur web ne doivent pas appartenir au processus qui fait tourner le serveur.

Ici, nous attribuons les contenus à l'utilisateur non privilégié `stagiaire1` et au groupe associé `stagiaires`. En passant, nous en profitons pour restreindre les droits d'accès du groupe.

```
# cd /var/www/html
# chown -R stagiaire1:stagiaire ./*
# find ./ -type d -exec chmod 0755 \{} \;
# find ./ -type f -exec chmod 0644 \{} \;
```

5.10. Héberger plusieurs sites sur le même serveur

Le principe des hôtes virtuels (Virtual Hosts) consiste à faire fonctionner un ou plusieurs sites Web sur une même machine. L'utilisateur final ne perçoit pas qu'en fait il s'agit d'un même serveur physique.

Nous allons héberger trois sites :

- http://slackware.formatux.fr hébergera la documentation de Slackware.
- http://freebsd.formatux.fr affichera la documentation de FreeBSD.
- http://formatux.fr pointera vers la page par défaut du serveur.

Pour commencer, on va déplacer le site existant dans un nouveau répertoire slackware/html.

```
# cd /var/www/
# mkdir -pv ./slackware/html
mkdir: création du répertoire « ../slackware »
mkdir: création du répertoire « ../slackware/html »
# mv ./html/* ./slackware/html/
```

Puis, on va créer un autre répertoire freebsd/html, dans lequel on va télécharger un autre site, en l'occurrence la documentation de FreeBSD.

```
# mkdir -pv freebsd/html
mkdir: création du répertoire « freebsd »
mkdir: création du répertoire « freebsd/html »
# cd freebsd/html
# wget -r -p -np -nH --cut-dirs=4 \
  http://www.freebsd.org/doc/fr_FR.ISO8859-1/books/handbook/
```

Enfin, on va mettre en place une page par défaut dans le répertoire default/html. Pour ce faire, on va utiliser la page qui s'affiche lorsqu'il n'y a pas de contenu.

```
# cd /var/www/
# mkdir -pv default/html
mkdir: création du répertoire « default »
mkdir: création du répertoire « default/html »
# cp -R /usr/share/httpd/noindex/* default/html/
```

Au total, nous avons donc :

```
# ls -l
total 12
drwxr-xr-x 3 root root 4096 23 févr. 06:36 default
drwxr-xr-x 3 root root 4096 23 févr. 06:21 freebsd
drwxr-xr-x 3 root root 4096 23 févr. 06:19 slackware
```

Nous allons redéfinir les permissions :

```
# chown -R stagiaire:stagiaire ./*
# find . -type d -exec chmod 0755 \{} \;
# find . -type f -exec chmod 0644 \{} \;
```

Créer un fichier `/etc/httpd/conf.d/00-formatux.fr.conf`. Ce fichier définira le site affiché par défaut, c'est-à-dire lorsqu'on invoque l'adresse IP ou le nom d'hôte de la machine :

```
# Page par défaut
<VirtualHost *:80>
  ServerAdmin info@formatux.fr
  DocumentRoot "/var/www/default/html"
  ServerName www.formatux.fr
  ServerAlias formatux.fr
  ErrorLog logs/formatux.fr-error_log
  CustomLog logs/formatux.fr-access_log common
</VirtualHost>
```

Prendre en compte les modifications :

```
# systemctl reload httpd
```

Vérifier si la page par défaut du serveur s'affiche comme prévu :

```
$ links http://www.formatux.fr
```

À présent, nous pouvons ajouter les deux autres sites. Le site http://slackware.formatux.fr sera configuré comme ceci :

```
# http://slackware.formatux.fr
<VirtualHost *:80>
  ServerAdmin info@formatux.fr
  DocumentRoot "/var/www/slackware/html"
  ServerName slackware.formatux.fr
  ErrorLog logs/slackware.formatux.fr-error_log
  CustomLog logs/slackware.formatux.fr-access_log common
</VirtualHost>
```

La configuration de http://freebsd.formatux.fr suivra la même logique :

```
# http://freebsd.formatux.fr
<VirtualHost *:80>
  ServerAdmin info@formatux.fr
  DocumentRoot "/var/www/freebsd/html"
  ServerName freebsd.formatux.fr
  ErrorLog logs/freebsd.formatux.fr-error_log
  CustomLog logs/freebsd.formatux.fr-access_log common
</VirtualHost>
```

Pour l'instant, les noms d'hôtes `slackware.formatux.fr` et `freebsd.formatux.fr` ne correspondent à rien dans notre réseau local. Nous devons les ajouter à `/etc/hosts` sur le serveur.

```
192.168.2.5 www.formatux.fr formatux.fr  freebsd.formatux.fr slackware.formatux.fr
```

Redémarrer Dnsmasq pour propager l'info DNS.

```
# systemctl restart dnsmasq
```

Prendre en compte la nouvelle configuration d'Apache.

```
# systemctl reload httpd
```

Tester les deux sites en local avec Links ou Firefox sur une machine du réseau local.

`http://slackware.formatux.fr`
`http://freebsd.formatux.fr`

5.11. Héberger des sites dynamiques avec PHP

Installer PHP :

```
# yum install php
```

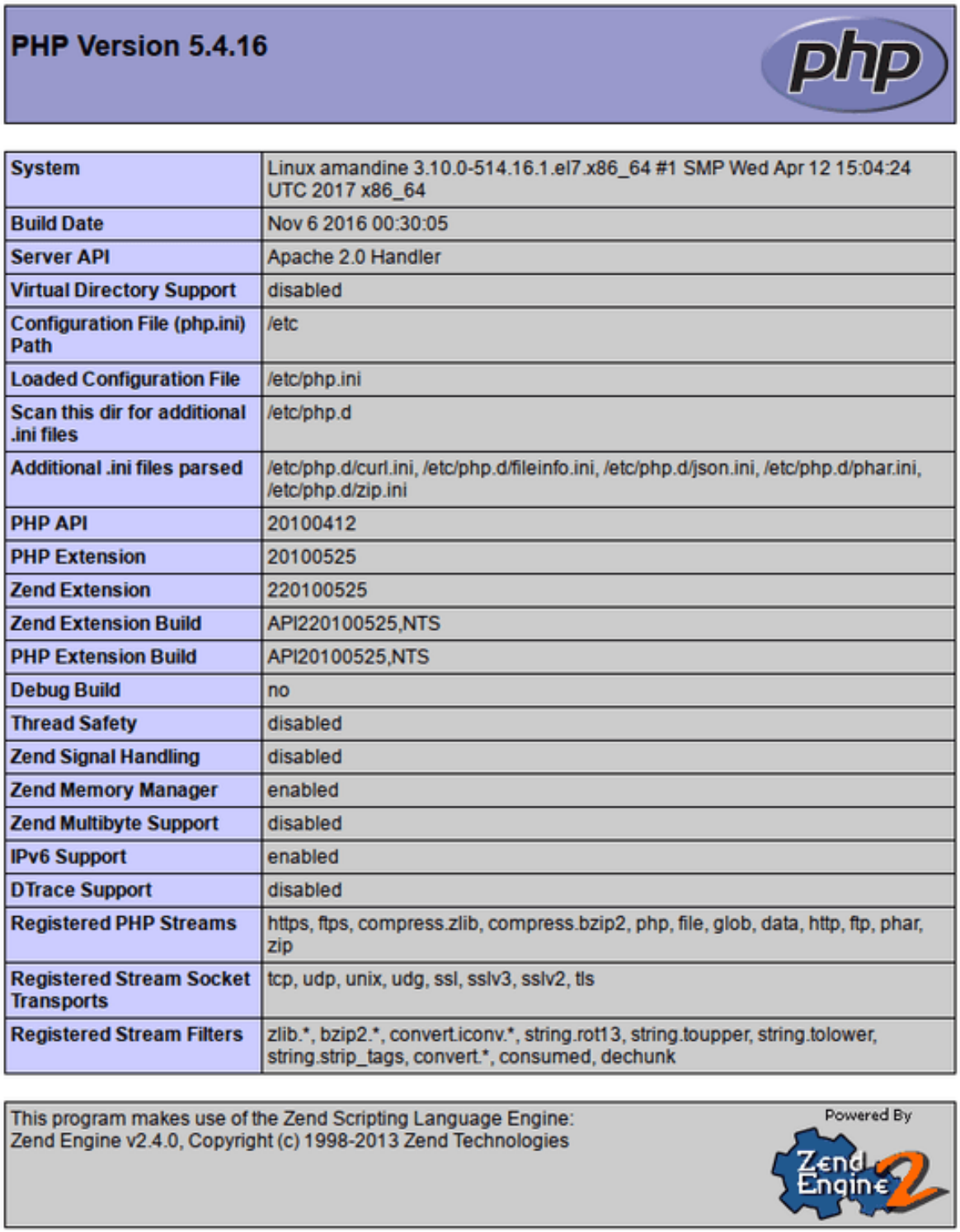
Mettre en place un hôte virtuel http://phpinfo.formatux.fr et éditer la configuration correspondante. L'hôte virtuel contiendra une seule page index.php que l'on éditera comme ceci :

```
<?php
 echo phpinfo();
?>
```

Ajouter une entrée correspondante dans la configuration DNS ou dans le fichier `/etc/hosts` et redémarrer Apache :

```
# systemctl restart httpd
```

Afficher la page `http://phpinfo.formatux.fr` dans un navigateur. On doit obtenir quelque chose qui ressemble grosso modo à ceci :



PHP Version 5.4.16

System	Linux amandine 3.10.0-514.16.1.el7.x86_64 #1 SMP Wed Apr 12 15:04:24 UTC 2017 x86_64
Build Date	Nov 6 2016 00:30:05
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/curl.ini, /etc/php.d/fileinfo.ini, /etc/php.d/json.ini, /etc/php.d/phar.ini, /etc/php.d/zip.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension Build	API220100525,NTS
PHP Extension Build	API20100525,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	https, ftps, compress.zlib, compress.bzip2, php, file, glob, data, http, ftp, phar, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, sslv2, tls
Registered Stream Filters	zlib.*, bzip2.*, convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk

This program makes use of the Zend Scripting Language Engine:
Zend Engine v2.4.0, Copyright (c) 1998-2013 Zend Technologies

Powered By Zend Engine 2

Figure 48. La page phpinfo() de PHP

Le fichier `/etc/php.ini` contient la configuration de PHP. On peut commencer par définir le fuseau horaire du serveur, nécessaire pour le bon fonctionnement de certaines applications :

```
[Date]
; Defines the default timezone used by the date functions
; http://php.net/date.timezone
date.timezone = Europe/Paris
```

Redémarrer Apache et vérifier les données correspondantes dans la page qui affiche les infos PHP.

Vous pouvez en apprendre plus sur PHP en suivant notre cours SVR-121-Php-fpm.

5.12. Utiliser MySQL/MariaDB à partir de PHP

Pour utiliser MySQL/MariaDB à partir de PHP, il suffit d'installer le module correspondant et de redémarrer Apache.

```
# yum install php-mysql
# systemctl restart httpd
```

La gestion d'une base de données MariaDB/MySQL est abordée dans le cours SVR-131-MySQL.

5.13. Documentation

- La documentation officielle du projet
- Blog Micro Linux
- Michael Kofler – Apache, Linux 2017

Chapitre 6. Sécurisation du serveur web Apache

6.1. Introduction

La sécurisation d'un site internet nécessite la mise en place de trois mesures :

- La sécurité du service en lui-même.

Par défaut un serveur Apache va envoyer des informations avec ses pages web. Ces informations contiennent des détails sur sa version ou ses modules. Pour un pirate, ces informations vont lui permettre de cibler les attaques en fonction des failles connues. Il est donc impératif de masquer ces informations sur un serveur de production.

- La sécurité des accès aux données.

Pour empêcher l'accès aux données, il est nécessaire de mettre en place une authentification. Cette authentification peut être d'ordre applicative, et s'appuyer par exemple sur une base de données, ou être gérée par le service Apache.

- La sécurité des échanges (protocole https).

La mise en place d'un processus d'authentification sur le serveur pour protéger l'accès aux données n'empêche pas l'attaquant d'intercepter les requêtes sur le réseau, et d'ainsi obtenir les documents désirés voir les informations d'identification (utilisateur et mot de passe). L'authentification va donc de pair avec le chiffrement des échanges.

Moins d'informations = intrusion plus difficile = Masquer l'identité du serveur Apache.

6.2. Masquage de l'identité d'Apache

Directive ServerSignature

Afficher une ligne de bas de page pour les documents générés par le serveur (messages d'erreur)

Syntaxe de la directive ServerSignature

```
ServerSignature  On | Off | EMAil
```

Exemple :

```
ServerSignature  Off
```

La directive "ServerSignature" ajoute un pied de page aux documents générés par le serveur (messages d'erreur, liste des répertoires, ftp, etc.).

L'utilité d'ajouter ces informations apparaît par exemple lorsqu'une réponse à une requête traverse plusieurs proxys. Dans ces conditions, sans ces informations, il devient difficile pour l'utilisateur de déterminer quel élément de la chaîne de proxy a produit un message d'erreur.

Sur un serveur de production, pour des raisons de sécurité, il est préférable de positionner cette option à Off.

ServerSignature On :

ServerSignature Off :

L'information fournie par cette page semble anodine mais pourtant est très précieuse pour un attaquant.

En effet, un éventuel attaquant apprend que le serveur apache, disponible à l'adresse IP 160.210.125.126 est un serveur Apache 2.2.6. La version actuelle d'apache étant la version 2.2.29 (en décembre 2014).

L'attaquant peut donc utiliser le document situé à cette adresse : http://www.apache.org/dist/httpd/

CHANGES_2.2 pour cibler ses attaques.

Directive ServerTokens

Renseigner le champ "server" de l'entête http.

Syntaxe de la directive ServerTokens

```
ServerTokens  Prod[uctOnly] | Min[imal] | OS | Full
```

Exemple :

```
ServerTokens Prod
```

Dans un navigateur web, la page internet que nous consultons n'est que la partie visible par l'utilisateur du contenu d'une requête HTTP. Les en-têtes HTTP fournissent de nombreuses informations, que ce soit du client vers le serveur ou du serveur vers le client.

Ces en-têtes peuvent être visualisée par exemple avec :

- la commande wget, qui permet le téléchargement d'une URL en ligne de commande,
- avec le module "Développement Web" fourni en standard avec le navigateur Firefox.

En-têtes sans ServerTokens (full)

```
[root]# wget –S  http://160.210.125.126

--14:30:07--  http://160.210.125.126/
=> `index.html'
Connexion vers 160.210.125.126:80...connecté.
requête HTTP transmise, en attente de la réponse...
  HTTP/1.1 200 OK
  Date: Mon, 01 Mar 2010 13:30:07 GMT
  Server: Apache/2.2.6 (Fedora) DAV/2
  Last-Modified: Mon, 01 Mar 2010 08:26:10 GMT
  ETag: "16e9e-3c55-594c80"
  Accept-Ranges: bytes
  Content-Length: 15445
  Connection: close
  Content-Type: text/html; charset=UTF-8
  Longueur: 15445 (15K) [text/html]

  100%[======================================>] 15445         --.--K/s

  14:30:07 (86.86 MB/s) - « index.html » sauvegardé [15445/15445]
```

Pour les mêmes raisons que pour la directive ServerSignature vue précédemment, il est impératif de limiter les informations transmises par un serveur de production.

Table 87. La directive ServerTokens

Valeur	Information
Prod[uctOnly]	Server : Apache
Min[imal]	Server : Apache/1.3.0
OS	Server : Apache/1.3.0 (Unix)
Full	Server : Apache/1.3.0 (Unix) PHP/3.0 MyMod/1.2

En-têtes avec ServerTokens Prod

```
[root]# wget –S  http://160.210.125.126

--14:30:07--  http://160.210.125.126/
=> `index.html'
Connexion vers 160.210.125.126:80...connecté.
requête HTTP transmise, en attente de la réponse...
  HTTP/1.1 200 OK
  Date: Mon, 01 Mar 2010 13:30:07 GMT
  Server: Apache
  Last-Modified: Mon, 01 Mar 2010 08:26:10 GMT
  ETag: "16e9e-3c55-594c80"
  Accept-Ranges: bytes
  Content-Length: 15445
  Connection: close
  Content-Type: text/html; charset=UTF-8
  Longueur: 15445 (15K) [text/html]

  100%[====================================>] 15445         --.--K/s

  14:30:07 (86.86 MB/s) - « index.html » sauvegardé [15445/15445]
```

Le choix le plus judicieux est généralement de modifier le fichier /etc/httpd/conf/httpd.conf pour mettre la directive ServerTokens à Prod

6.3. Gestion des authentifications

L'authentification est la procédure qui consiste à vérifier l'identité d'une personne ou d'un ordinateur afin d'autoriser l'accès de cette entité à des ressources (systèmes, réseaux, applications,...).

Il existe de nombreuses méthodes d'authentifications :

- Authentification classique (simple) ;

- Authentification LDAP ;
- Authentification via serveur de base de données ;
- Authentification via PAM ;
- Authentification via SSO.

Apache permet par défaut d'assurer ce processus, avec une authentification classique ou simple, qui s'appuie sur des fichiers de textes contenant les informations de connexion des utilisateurs.

Cette fonctionnalité basique peut être enrichie par des modules, et permettre à apache de s'appuyer sur :

- Une authentification via un serveur LDAP. Ce procédé permet de déléguer l'authentification des utilisateurs (et leur appartenance à des groupes) à un serveur LDAP, dont c'est la fonctionnalité première. Apache utilise alors le module mod_authnz_ldap, qui dépend du module Apache d'accès à LDAP, mod_ldap, qu'il faut aussi installer.
- Une authentification via un serveur de base de données. Ce procédé s'appuiera sur le langage SQL (Structured Query Language) pour la gestion des utilisateurs, de leurs mots de passe et leur appartenance à des groupes.
- Le module PAM (Pluggable Authentification Modules). Ce procédé permet de déléguer l'authentification au système d'exploitation. Pour mettre en place cette procédure, il faut installer deux modules : mod_auth_pam et pam_auth_external.

Ces modules ne sont pas actifs par défaut mais sont présents dans les dépôts.

Dans ce chapitre nous ne verrons que l'authentification classique (dite aussi « simple »).

Authentification classique

Authentification classique : protéger l'accès à un site ou à un dossier d'un site par la mise en place d'une authentification par mot de passe.

L'activation du module d'authentification pour un site peut se faire :

- Soit dans la configuration globale du site, si l'administrateur du site est également administrateur du serveur, ou que l'administrateur lui en a laissé l'accès.

Les directives de configuration se positionne entre les balises <Directory> ou <Location>.

C'est la méthode à privilégier.

- Si la modification du fichier de configuration globale n'est pas possible, ce qui est souvent le cas d'un serveur mutualisé, la protection se fera alors dans un fichier .htaccess directement dans le dossier à protéger.

Dans ce cas, l'administrateur du serveur aura explicitement configuré cette possibilité dans la configuration globale avec la directive AllowOverride à All ou à AuthConfig.

Ce fichier étant évalué à chaque accès, cela peut induire une petite perte au niveau des performances.

L'authentification sécurise l'accès aux données, mais la donnée transite toujours en clair durant la transmission au client.

Directive AuthType

Renseigner le type de contrôle des autorisations

Directives associées : AuthName, AuthUserFile

Syntaxe de la directive AuthType

```
AuthType Basic | Digest
```

Exemple :

```
AuthType Basic
```

AuthType indique à Apache d'utiliser le protocole Basic ou Digest pour authentifier l'utilisateur :

- Authenfication Basic : Transmission du mot de passe client en clair

Pour mettre en place cette méthode, il faut utiliser la commande htpasswd qui permet de créer un fichier qui va contenir les logins (ou les groupes) et les mots de passe des utilisateurs (ou les groupes) habilités à accéder au dossier Web sécurisé (la commande étudiée plus loin).

- Authentification Digest : Hachage MD5 128 bits du mot de passe avant transmission

Ce module implémente l'authentification HTTP basée sur les condensés MD5, et fournit une alternative à mod_auth_basic en ne transmettant plus le mot de passe en clair.

Cependant, cela ne suffit pas pour améliorer la sécurité de manière significative par rapport à l'authentification basique. En outre, le stockage du mot de passe sur le serveur est encore moins sûr dans le cas d'une authentification à base de condensés que dans le cas d'une authentification basique.

C'est pourquoi l'utilisation de l'authentification basique associée à un chiffrement de la connexion via mod_ssl constitue une bien meilleure alternative.

Plus d'informations : http://httpd.apache.org/docs/2.2/fr/mod/mod_auth_digest.html.

Pour des raisons de sécurité, seul le mécanisme Basic sera utilisé par la suite.

Configuration du fichier /etc/httpd/conf/httpd.conf :

Dans les balises <Directory> ou <Location>, ajouter les directives d'authentification :

Syntaxe Apache pour protéger l'accès à un dossier

```
<Directory directory-path >
    AuthType Basic | Digest
    AuthName "text"
    AuthUserFile directory-path/.PrivPasswd
    Require user | group | valid-user
</Directory>
```

- AuthName est le message qui est affiché dans la boite de dialogue de saisie du login et du mot de passe.
- AuthUserFile indique le chemin et le nom du fichier contenant le nom des utilisateurs et leur mot de passe. Pour une identification sur un groupe il faut travailler avec la directive AuthGroupFile. Ces deux directives font parties du module mod_auth .
- Require est la règle d'authentification à proprement parler. Elle indique à Apache qu'un utilisateur ayant réussi à s'authentifier à partir du fichier des mots de passe (spécifié dans la directive AuthUserFile) peut accéder au site Web sécurisé.

Exemple :

```
<VirtualHost www.monsite.com >
......
  <Directory /var/www/html/sitetest>
   # Type d'authentification
   AuthType Basic
   # texte affiché dans la boite de dialogue
   AuthName " Acces Securise "
   # fichier contenant les logins et mdp
   AuthUserFile /var/www/private/sitetest/.PrivPasswd
   # Accès par vérification du mot de passe
   require valid-user
  </Directory>
</VirtualHost>
```

Le fichier .PrivPasswd est créé avec la commande htpasswd, que nous verrons plus loin dans ce chapitre.

Le fichier de gestion des logins et des mots de passe .PrivPasswd est un fichier sensible. Il ne faut pas le placer dans le répertoire de publication de votre site, mais plutôt dans un répertoire extérieur à l'arborescence de votre site suivi du nom du site.

Exemple

```
[root]# mkdir -p /var/www/private/SiteTest
```

Le fichier .PrivPasswd" sera ensuite créé dans ce répertoire à l'aide de la commande htpasswd.

Le fichier .htaccess utilise les mêmes directives que précédemment, mais non encadrés par les balises Directory ou Location.

Syntaxe:

```
AuthType Basic | Digest
AuthName "text"
AuthUserFile directory-path/.PrivPasswd
Require user | group | valid-user
```

Avec dans le fichier « /etc/httpd/conf/httpd.conf » <Directory directory-path > AllowOverride AuthConfig </Directory>

Un serveur web répond généralement pour plusieurs sites. Dans ce cas, il est plus simple de configurer ce type de spécificité de configuration directement dans le dossier en question.

- Avantages : l'usage d'un fichier .htaccess a le mérite d'être simple et permet notamment de protéger un dossier Web racine ainsi que les sous-dossiers (sauf si un autre fichier .htaccess y contrevient), il est possible de déléguer la configuration ou la personnalisation du service à un administrateur du site.
- Inconvénients : il n'est pas simple de maintenir un nombre élevé de fichiers .htaccess. L'évaluation du fichier .htaccess par le serveur peut induire sur les performances.

Il ne faut pas oublier d'autoriser la configuration du module d'identification par fichier .htaccess à l'aide de la directive : AllowOverride AuthConfig.

Sans cette directive, apache ne permettra pas au fichier .htaccess d'écraser la configuration définie dans la configuration globale.

```
<VirtualHost www.monsite.com >
    <Directory /home/SitesWeb/SiteTest>
       # déclaration de l'utilisation de .htaccess
      AllowOverride  AuthConfig
    </Directory>
</VirtualHost>
```

Exemple de fichier .htaccess

```
# Type d'authentification
AuthType Basic
# texte affiché dans la boite de dialogue
AuthName " Acces Securise "
# fichier contenant les logins et mdp
AuthUserFile /var/www/private/sitetest/.PrivPasswd
# Accès par vérification du mot de passe
require valid-user
```

Lors du traitement d'une requête, Apache cherche la présence du fichier .htaccess dans tous les répertoires du chemin menant au document depuis la directive DocumentRoot.

Exemple, avec une directive DocumentRoot à /home/sitesweb/ avant de retourner /home/sitesWeb/sitetest/indextest.htm Apache examine les fichiers :

- /home/sitesWeb/.htaccess
- /home/sitesWeb/sitetest/.htaccess

Mieux vaut désactiver cette fonctionnalité sur / (activé par défaut) : <Directory /> AllowOverride None </Directory>

Commande htpasswd

La commande htpasswd permet de gérer les utilisateurs du site.

Syntaxe de la commande htpasswd

```
htpasswd [-options] passwordfile username [password]
```

Exemples :

```
[root]# cd /var/www/private/sitetest
[root]# htpasswd -cb .PrivPasswd user1 mdpuser1
[root]# htpasswd –D .PrivPasswd  user
```

Table 88. La directive ServerTokens

Option	Observation
-c	Créer un nouveau fichier
-b	Indiquer le mot de passe sur la ligne de commande
-m	Chiffrer le mot de passe en md5
-D	Supprimer un utilisateur de la liste d'accès

La commande htpasswd met en place la liste des utilisateurs habilités à accéder au site sécurisé, dans le cas de l'utilisation de la directive « AuthType » avec la valeur « Basic » et vérifie les droits sur le répertoire, afin qu'au moins le groupe « apache » puisse y accéder.

Pour créer le fichier et définir le premier utilisateur :

```
[root]# cd /var/www/private/siteTest
[root]# htpasswd –c .PrivPasswd user1
```

Puis saisir le mot de passe

Pour ajouter les autres utilisateurs dans le fichier :

```
[root]# htpasswd  .PrivPasswd user2
```

Puis saisir le mot de passe

Pour ajouter un utilisateur et définir son mot de passe à la suite de la commande

```
[root]# htpasswd  -b .PrivPasswd user3 mdpuser3
```

Ajouter un utilisateur avec un mot de passe chiffré en md5 :

```
[root]# htpasswd  -bm .PrivPasswd user4 mdpuser4
```

Le résultat donne un fichier /var/www/private/sitetest/.PrivPasswd :

```
user1:$1Pd$EsBY75M
user2:Mku4G$j4p£kl1
user3:Ng7Rd$5F$68f
...
```

6.4. Utilisation du module SSL

Le protocole TLS (Transport Layer Security), successeur du protocole SSL (Secure Socket Layer) est un protocole de sécurisation des échanges sur Internet.

Le protocole SSL a été créé à l'origine par Netscape. Le principe de fonctionnement du TLS repose sur le recours à un tiers, l'Autorité de Certification (CA/Certificate Authority).

C'est un protocole de niveau 4 sur lequel les protocoles des couches OSI supérieures s'appuient. Ils est donc commun pour les protocoles imaps, pops, ldaps, etc.

Le fichier openssl.cnf (/etc/pki/tls/openssl.cnf) peut être configuré de façon à minimiser les données à renseigner à chaque invocation des utilitaires openssl lors de la création des clefs de chiffrement.

Etant donné son caractère hautement critique, l'administrateur veillera à utiliser une version d'openssl à jour de correctifs.

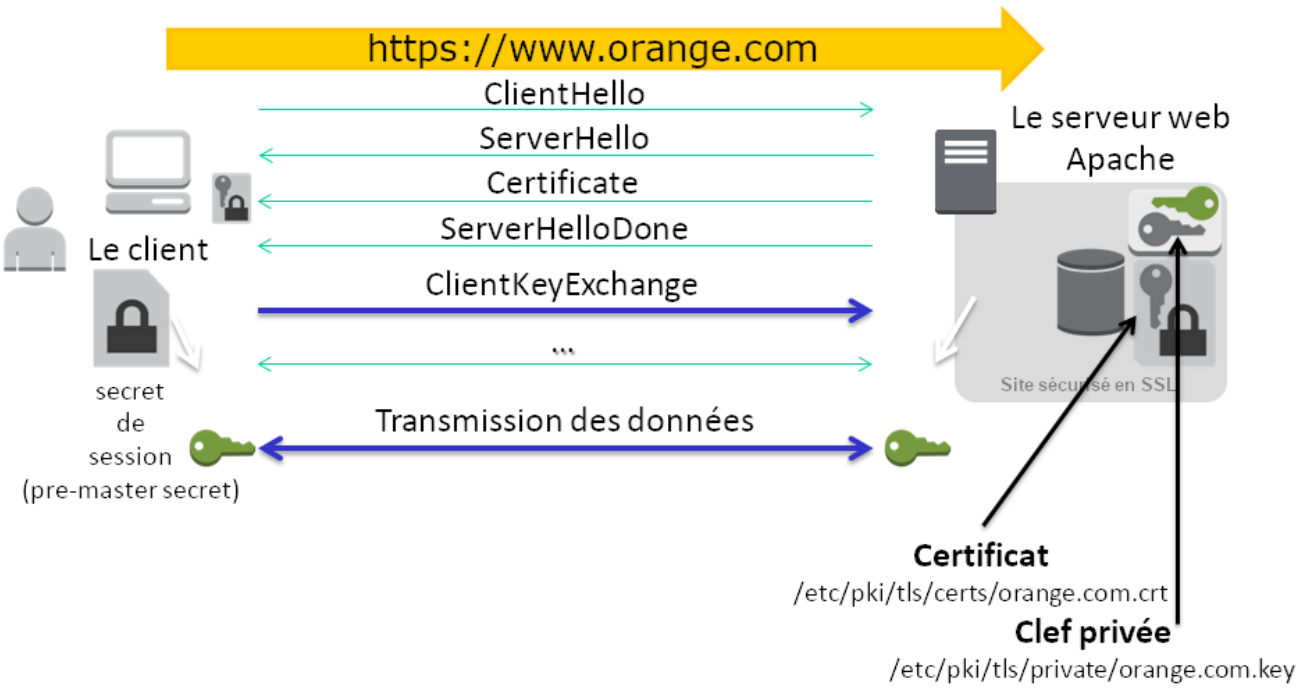
Le protocole SSL/TLS existe en 5 versions :

- SSLv2
- SSLv3
- TLSv1
- TLSv1.1
- TLSv1.2

Prérequis

- Le port 443 (https) est-il ouvert sur les pare-feu ?
- Le logiciel OpenSSL est-il installé et à jour sur le serveur ?
- Le module mod_ssl est-il installé sur le serveur Apache et activé ?

Établissement d'une session TCP https (port 443)



Lors du ClientHello, le client se présente au serveur. Il lui soumet les méthodes de cryptologie, de compression, et les standards SSL qu'il connait.

Le serveur répond au client avec un message ServerHello. Les 2 parties se sont mises en accord sur le CypherSuite qu'ils allaient utiliser, par exemple avec un CypherSuite TLS_RSA_WITH_AES_128_MD5.

Ils doivent maintenant s'échanger un secret de session : le pre-master secret. Son chiffrement s'effectue avec le certificat publique du serveur transmis pendant le message Certificate.

Le pre-master secret est dérivé en clefs symétriques sur le client et le serveur. Ces clefs symétriques serviront à la transmission des données.

La chaîne (CypherSuite) est composée de 5 éléments distincts :

- Le protocole SSL/TLS
- L'algorithme asymétrique utilisé, principalement RSA et ECDSA
- L'algorithme symétrique utilisé, comme AES, Camelia, SEED, 3DES ou RC4
- Le mécanisme de protection des données pour éviter qu'un assaillant puisse modifier les messages chiffrés (HMAC ou AEAD). Dans le cas du HMAC on choisira une fonction de hachage (MD5, SHA-1, SHA-2)
- La présence ou non de confidentialité persistante ou PFS (Perfect Forward Secrecy)

Mise en place d'un site TLS

La mise en place d'un site TLS respecte les étapes suivantes :

1. Installation du module mod_ssl

2. Configuration du module mod_ssl
3. Création de la clé privée du serveur
4. Création du certificat du serveur depuis sa clé privée
5. Création d'un Certificat Signing Request (CSR) depuis le certificat et transmission à la CA pour signature
6. Installation du certificat
7. L'hôte virtuel peut être configuré en TLS

Le logiciel OpenSSL

Le logiciel OpenSSL, utilitaire cryptographique, implémente les protocoles réseaux :

- Secure Sockets Layer (SSL V2/V3, couche de sockets sécurisés) ;
- Transport Layer Security (TLS v1, sécurité pour la couche de transport).

OpenSSL permet :

- Création de paramètres des clefs RSA, DH et DSA
- Création de certificats X.509, CSRs et CRLs
- Calcul de signature de messages
- Chiffrement et Déchiffrement
- Tests SSL/TLS client et server
- Gestion de mail S/MIME signé ou chiffrés

Création des clés et certificats

Les clés et certificats sont stockés dans le répertoire /etc/pki/tls/.

Un dossier private accueille les clés privées. Un dossier certs accueille les certificats publiques.

Les droits doivent être à 440.

Pour installer le module mod_ssl :

```
[root]# yum install mod_ssl
```

Pour vérifier la présence du module dans le serveur Apache :

```
[root]# httpd –t –D DUMP_MODULES | grep ssl_module
ssl_module (shared)
```

La commande nous indique que le module est disponible mais pas qu'il est activé.

Après cette installation, vous devez trouver le module "mod_ssl.so" dans le répertoire /etc/httpd/modules" qui est en fait un lien symbolique sur /usr/lib/httpd/modules.

Pour activer le mod_ssl dans /etc/httpd/conf/httpd.conf, la ligne suivante doit être présente :

```
LoadModule ssl_module modules/mod_ssl.so
```

Le module mod_ssl peut être configuré dans le fichier /etc/httpd/conf.d/ssl.conf

N'oubliez pas de redémarrer le service Apache :

```
[root]# service httpd restart
```

Pour accepter les requêtes TLS, le serveur Apache a besoin de deux fichiers :

- une clé privée : NomDeFichier.key;
- un certificat signé : NomDeFichier.crt

La signature de ce certificat est réalisée par un organisme de certification tiers (tel que Verisign ou Thawte). Cependant vous pouvez signer vous même votre certificat, la seule différence sera un avertissement par le navigateur lors de l'accès à une ressource TLS de votre serveur. La sécurité est la même, que le certificat soit signé par vous même ou pas un organisme.

Si vous décidez d'utiliser une autorité de certification auto-signée, son certificat devra être installée sur l'ensemble de vos postes.

- 1ère étape : Générer la clef privée

```
openssl genrsa \    –out /etc/pki/tls/private/orange.com.key \    2048
Generating RSA private key, 2048 bit long modulus
- - - - - - - - - - - - - - - - +++
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +++
e is 65537 (0x10001)

chmod 400 /etc/pki/tls/private/orange.com.key
```

- 2ème étape : Générer une demande de certificat

```
openssl req    –new    –key /etc/pki/tls/private/orange.com.key   –out
/etc/pki/tls/certs/orange.com.csr
```

- 3ème étape : Envoi de la demande de certificat à l'autorité de certification (CA)
- 4ème étape : L'autorité de certification (CA) renvoie un certificat signé
- 5ème étape : Sécuriser et sauvegarder les certificats.

```
chmod 400 /etc/pki/tls/private/orange.com.key
chmod 400 /etc/pki/tls/certs/orange.com.crt
```

- 6ème étape : Déployer les certificats sur le serveur
- 7ème étape : Configurer le vhost

```
NameVirtualHost 192.168.75.50:443
<VirtualHost 192.168.75.50:443>
  ServerName www.orange.com

  SSLEngine on
  SSLCertificateFile /etc/pki/tls/certs/orange.com.crt
  SSLCertificateKeyFile /etc/pki/tls/private/orange.com.key

  DocumentRoot /home/SitesWeb/SiteOrange
  DirectoryIndex IndexOrange.htm

  <Directory /home/SitesWeb/SiteOrange>
    allow from all
  </Directory>

</VirtualHost>
```

Certificats Auto-Signés

Auto-signer ses certificats revient à créer sa propre autorité de certification.

- 1ère Etape : Générer la paire de clefs de l'autorité de certification

```
openssl genrsa –out /etc/pki/CA/private/cakey.pem
openssl req \    –new \    –x509 \    –key /etc/pki/CA/private/cakey.pem \    -out
/etc/pki/CA/certs/cacert.pem \    -days 365
```

- 2ème Etape : Configurer openssl

/etc/pki/tls/openssl.cnf

```
[ ca ]default_ca     =   CA_default

[ CA_default ]

dir = /etc/pki/CA
certificate = $dir/certs/cacert.pem
private_key = $dir/private/cakey.pem
```

Vérifier la présence du fichier /etc/pki/CA/index.txt.

Si celui-ci s'avère absent, il faut le créer :

```
touch /etc/pki/CA/index.txt
echo '1000' >  /etc/pki/CA/serial
```

Puis faire :

```
echo '1000' > /etc/pki/CA/serial
```

- 3ème Etape : Signer une demande de certificat

```
openssl ca \   -in /etc/pki/tls/certs/orange.com.csr \   -out
/etc/pki/tls/certs/orange.com.crt
```

Pour que ce certificat soit reconnu par les clients, le certificat cacert.pem devrait également être installé sur chaque navigateur.

Le certificat orange.com.crt est à envoyer au client

Chapitre 7. Apache - haute disponibilité

Objectifs de ce cours :

- paramétrer Apache en serveur frontal
- répartir la charge entre plusieurs serveurs applicatifs
- paramétrer Apache pour la tolérance de panne

7.1. Introduction

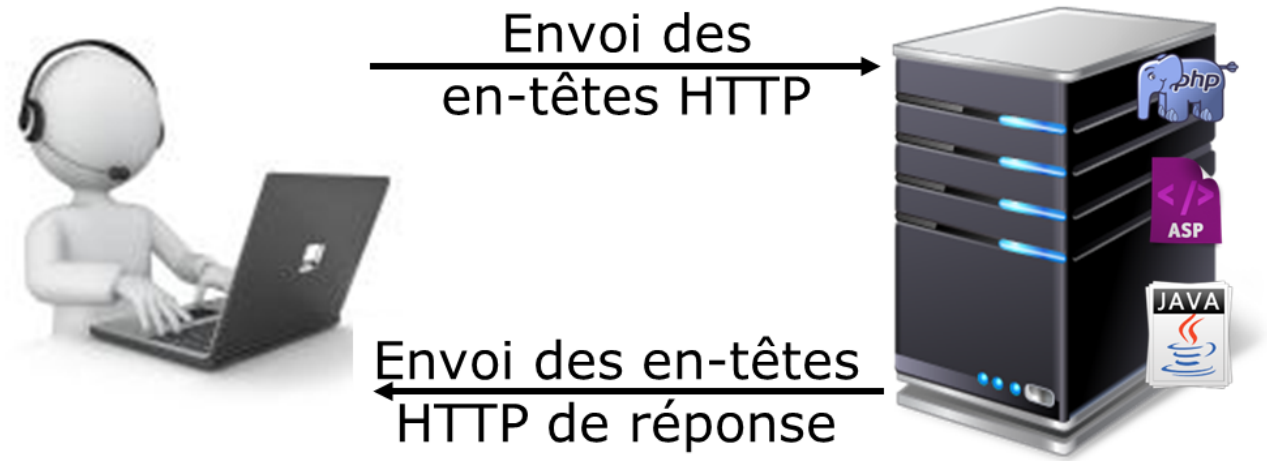
Apache est un serveur **modulaire** web (protocole HTTP).

Il peut :

- Gérer un cache de données ;
- Faire office de serveur mandataire ;
- Compresser les données envoyées aux clients ;
- et plein d'autres choses encore…

7.2. Serveur Applicatif

Un serveur applicatif héberge des applications dynamiques développées dans un langage de programmation web : Php / Asp / Java / Python …



7.3. Reverse proxy

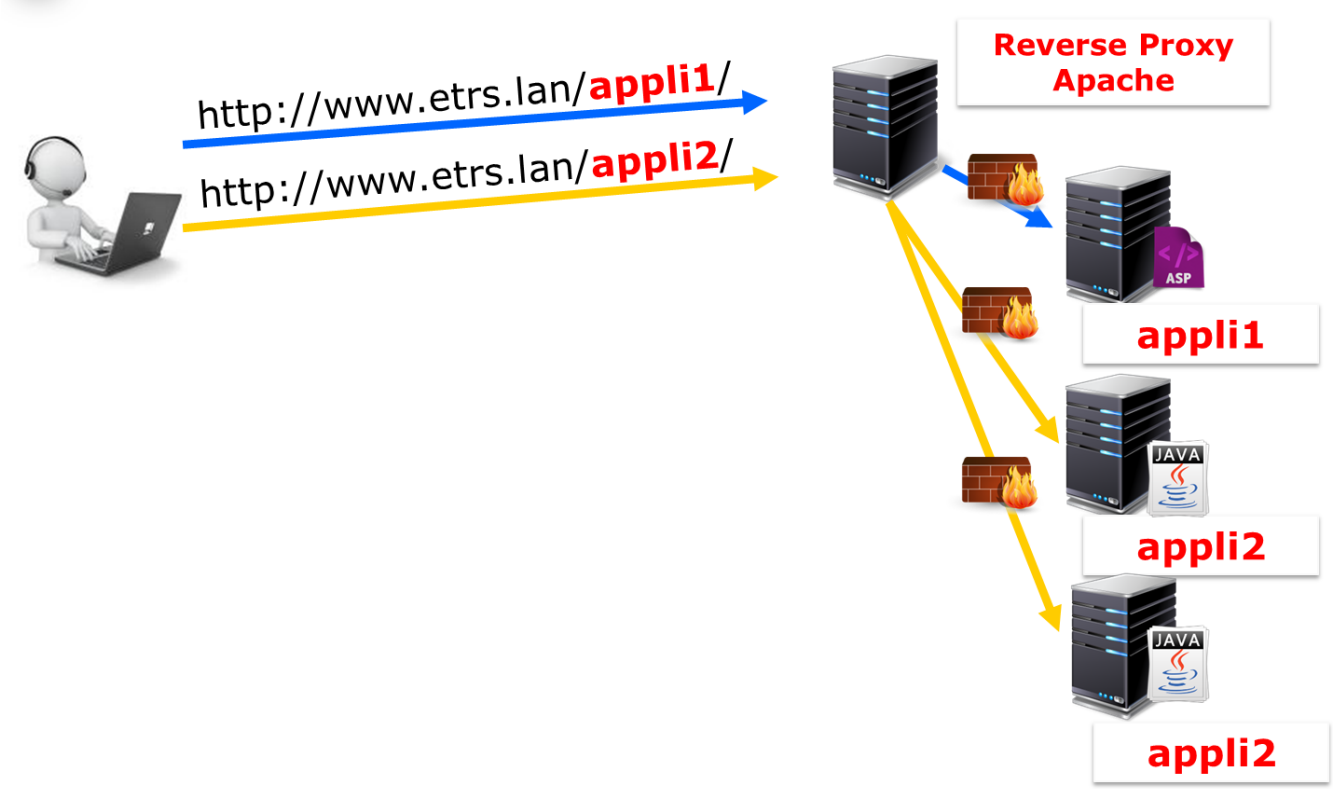
Un serveur reverse proxy (ou serveur mandataire) est mandaté par des clients pour interroger les serveurs applicatifs.

Aux yeux du client, un seul serveur est en ligne pour lui répondre. La complexité du SI lui est

masqué.

Les serveurs applicatifs sont isolés derrière un (ou plusieurs) pare-feu. Leurs ressources sont dédiées aux processus métiers (SQL, Web-services, etc.).

Le reverse proxy se charge de la compression, de la mise en cache, du chiffrement, et de la répartition de charge et/ou tolérance de panne.



Le module mod_proxy

Le module **mod_proxy** est une extension d'apache pour faire office de serveur mandataire / Passerelle HTTP.

- La directive **ProxyPreserveHost On** utilise l'en-tête de requête entrante Host pour la requête du mandataire
- La directive **ProxyPass chemin url** référence des serveurs distants depuis l'espace d'URLs du serveur local
- La directive **ProxyPassReverse chemin url** ajuste l'URL dans les réponses HTTP envoyées par le mandataire en inverse

Les différentes fonctionnalités de mandataire d'Apache sont réparties entre plusieurs modules complémentaires :

- mod_proxy_http,
- mod_proxy_ftp,
- mod_proxy_ajp,

- mod_proxy_balancer
- et mod_proxy_connect.

Apache peut être configuré comme mandataire directe (proxy) ou comme mandataire inverse (reverse proxy ou mode passerelle).

Pour la configuration d'un mandataire direct, reportez vous à la documentation du serveur SQUID.

Un mandataire inverse apparaît au client comme un serveur web standard. Aucune configuration du client n'est nécessaire. Le client adresse ses demandes de contenus ordinaires dans l'espace de nommage du mandataire inverse. Ce dernier décide alors où envoyer ces requêtes et renvoie le contenu au client comme s'il l'hébergeait lui-même.

L'accès des utilisateurs à un serveur situé derrière un pare-feu est une utilisation typique du mandataire inverse.

Configuration exemple d'un VirtualHost avec reverse-proxy

```
<VirtualHost 127.0.0.1:8080>
  ProxyPreserveHost On
  ProxyVia On
  <Proxy *>
    Order deny,allow
    Allow from all
  </Proxy>

  ProxyPass /pagebleue http://127.0.0.1:8080/pagebleue
  ProxyPassReverse /pagebleue http://127.0.0.1:8080/pagebleue
</VirtualHost>
```

- La directive **ProxyPreserveHost on**, lorsqu'elle est activé, va transmettre l'en-tête Host: de la requête entrante vers le serveur mandaté au lieu du nom d'hôte spécifié par la directive ProxyPass.
- La directive **ProxyVia On** permet de contrôler l'utilisation de l'en-tête http **Via:**. Définie à On, chaque requête ou réponse se verra ajouter une ligne d'en-tête Via: pour le serveur courant.

Des solutions spécialisées, comme **HaProxy** ou des équipements physiques comme les **Citrix NetScaler**, existent sur le marché, et sont plus puissantes qu'Apache. Au moment de choisir la solution de mandataire inverse, il faudra prendre en compte les éléments du tableau suivant :

	Points positifs	**Points négatifs**
HaProxy	Plus rapide	Spécifique au reverse proxy (pas de cache, nécessite Varnish ou Memcached)

	Points positifs	Points négatifs
	Développement actif	La syntaxe est différente des autres logiciels utilisés (NginX, Varnish, etc.)
Apache	Grand nombre de fonctionnalités, il peut tout faire	Performances moindres
	Solution unique pour tout le SI	
	Facilité de maintenance	

7.4. Simuler la charge

La commande **ab** (Apache Benchmark) permet d'envoyer des requêtes http à un client

Syntaxe de la commande ab

```
ab –n X –c Y url
```

Exemple :

Exemple d'utilisation de la commande ab

```
ab –n 5000 –c 2 http://127.0.0.1:8080/page1
```

Table 89. Options de la commande ab

Options	Commentaires
-n	Nombre de requêtes à envoyer
-c	Nombre de requêtes à envoyer par paquets

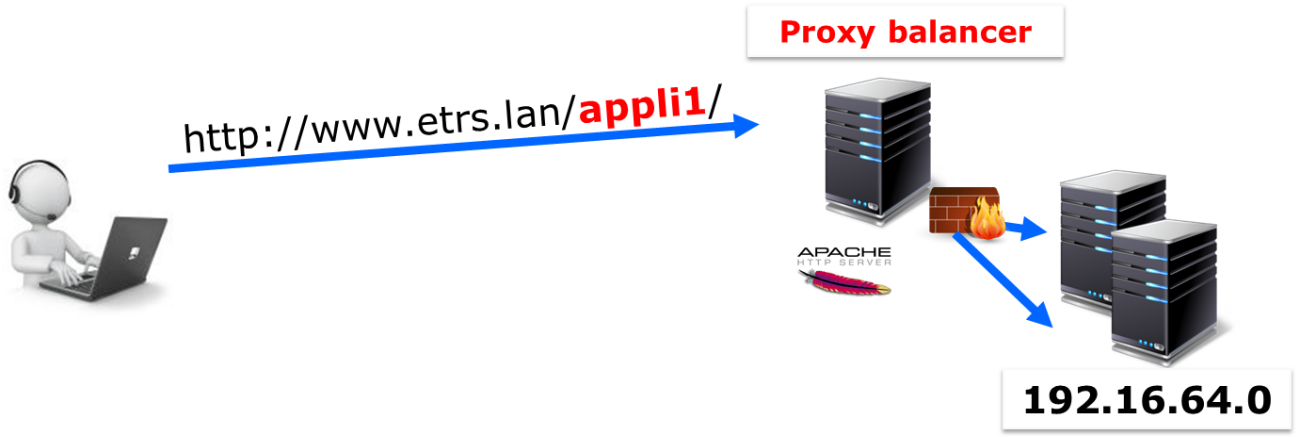
7.5. Répartir la charge

Pourquoi répartir la charge ?

- Améliorer les performances
- Accueillir plus de clients
- Gérer la maintenance
- Faire de la tolérance de panne

Plusieurs petits serveurs seront plus puissants qu'un gros pour un coût moindre.

Pour répartir la charge, Apache utilise un proxy-balancer.



Le module mod_proxy_balancer

Le module **mod_proxy_balancer** est une extension pour la répartition de charge.

- La directive **BalancerMember url [clé=valeur]** ajoute un membre à un groupe de répartition de charge
- La directive **loadfactor=X** est un facteur de charge entre 1 et 100
- La directive **ProxySet url [clé=valeur]** défini les différents paramètres relatifs à la répartition de charge
- La directive **lbmethod=byrequests|bytraffic|bybusyness** spécifie la méthode de répartition de charge (par défaut à byrequests). Les 3 méthodes de répartition de charge sont :
  - byrequests : réparti la charge entre les membres du cluster en fonction du nombre de requêtes traitées
  - bytraffic : réparti la charge entre les membres du cluster en fonction du trafic traitées
  - bybusyness : réparti la charge entre les membres du cluster en fonction de la charge actuelle
- La directive **stickysession=JSESSIONID|PHPSESSIONID** spécifie un nom de session persistant du répartiteur et dépend du serveur d'applications d'arrière plan.

```
<Proxy balancer://mycluster>
  BalancerMember http://192.16.164.1:8080/pages loadfactor=50
  BalancerMember http://192.16.164.2:8080/pages loadfactor=50

  ProxySet lbmethod=bytraffic
</Proxy>

ProxyPass /monsite balancer://mycluster
ProxyPassReverse /monsite balancer://mycluster
```

Le module mod_status

Le module **mod_status** permet de suivre l'état du load-balancer.

- La directive **ProxyStatus On** affiche l'état du répartiteur de charge du mandataire
- La directive **SetHandler balancer-manager** force le traitement par le gestionnaire balancer-manager

```
ProxyStatus On

<Location /balancer-manager>
  SetHandler balancer-manager
  Allow from all
</Location>
```

Vous pouvez maintenant visiter la page /balancer-manager

7.6. Tolérance aux pannes

Apache peut réagir en cas de panne du serveur métier et exclure du cluster le service qui ne répond plus et qui renvoie un code erreur `http`.

```
<Proxy balancer://mycluster>
  BalancerMember http://127.0.0.1:8080/pagebleue loadfactor=50 retry=30
  BalancerMember http://127.0.0.1:8080/pageverte loadfactor=50 retry=30

  ProxySet lbmethod=byrequests failonstatus=500,502,503,504

</Proxy>
```

Chapitre 8. Serveur de messagerie

8.1. Généralités

Le courrier électronique est apparu sur le réseau **ARPANET** dans les années 1970, ce qui fait du protocole **SMTP** un des plus anciens protocoles de **TCP/IP**.

Avec Internet, les systèmes de messagerie ont pris de l'importance et en 1980, **Sendmail** a été développé. Sendmail est devenu le premier serveur de messagerie important et utilisait déjà le protocole SMTP.

Postfix, dont le développement a été aidé par IBM, est apparu dès **1998**, afin de résoudre les problèmes de sécurité de Sendmail, tout en offrant une **administration beaucoup plus souple et modulaire**.

Le **MTA (Mail Transfer Agent)** par défaut de la distribution RedHat (Sendmail) est remplacé par Postfix sur la RedHat 6 (Novembre 2010).

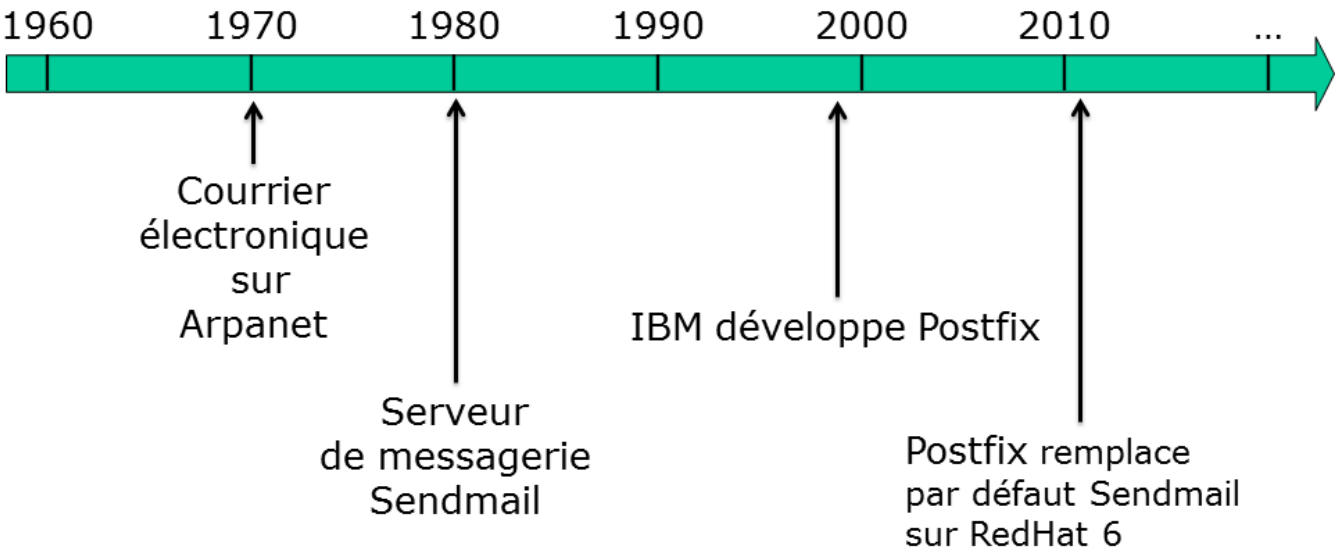


Figure 49. Historique des serveurs de messagerie sous Linux

Les systèmes de messagerie électronique reposent sur plusieurs standards et protocoles, définissant la façon dont sont composés les messages et leur acheminement, du rédacteur au destinataire.

L'**envoi de message** est assuré par le protocole **Simple Mail Transfer Protocol** (SMTP). SMTP est un protocole de communication de type texte.

Le fait que le protocole SMTP soit de type texte est intéressant. En effet, cela permet de tester son bon fonctionnement avec la commande **telnet**.

Les ports utilisés sont :

- **25** (sans authentification) ;
- **587** (avec authentification) ;
- **465** (SSL).

Local Mail Transfer Protocol (**LMTP**) est une variante de **ESMTP**, l'extension de SMTP. LMTP est défini dans la RFC 2033. LMTP a été conçu comme alternative aux échanges SMTP normaux dans les situations où la partie réceptrice ne possède pas de file d'attente des messages reçus. C'est le cas par exemple d'un agent de transfert de courrier agissant en tant qu'agent de distribution du courrier.

La **réception de message** peut se faire à l'aide de deux protocoles :

- **Internet Message Access Protocol** (**IMAP**) ;
- **Post Office Protocol** (**POP**).

IMAP est un protocole permettant de récupérer les courriers électroniques déposés sur des serveurs de messagerie. Les messages sont conservés sur le serveur, ses fonctionnalités avancées en font le protocole de préférence (accès **multipostes**, accès via **webmail**, etc.).

IMAP utilise le port **143** en clair ou via **STARTTLS** et le port **993** via **IMAPS** (déprécié) en SSL.

POP est également un protocole permettant de récupérer les courriers électroniques de l'utilisateur. En règle générale, POP se connecte sur le serveur, récupère le courrier, efface le courrier sur le serveur et se déconnecte. Ce fonctionnement par défaut ne permet pas l'accès aux mails depuis plusieurs postes, ni l'itinérance.

POP utilise le port **110** en clair ou via **STARTTLS** et le port **995** (déprécié) en SSL.

Le langage Sieve a été conçu pour permettre de filtrer des messages directement sur les serveurs.

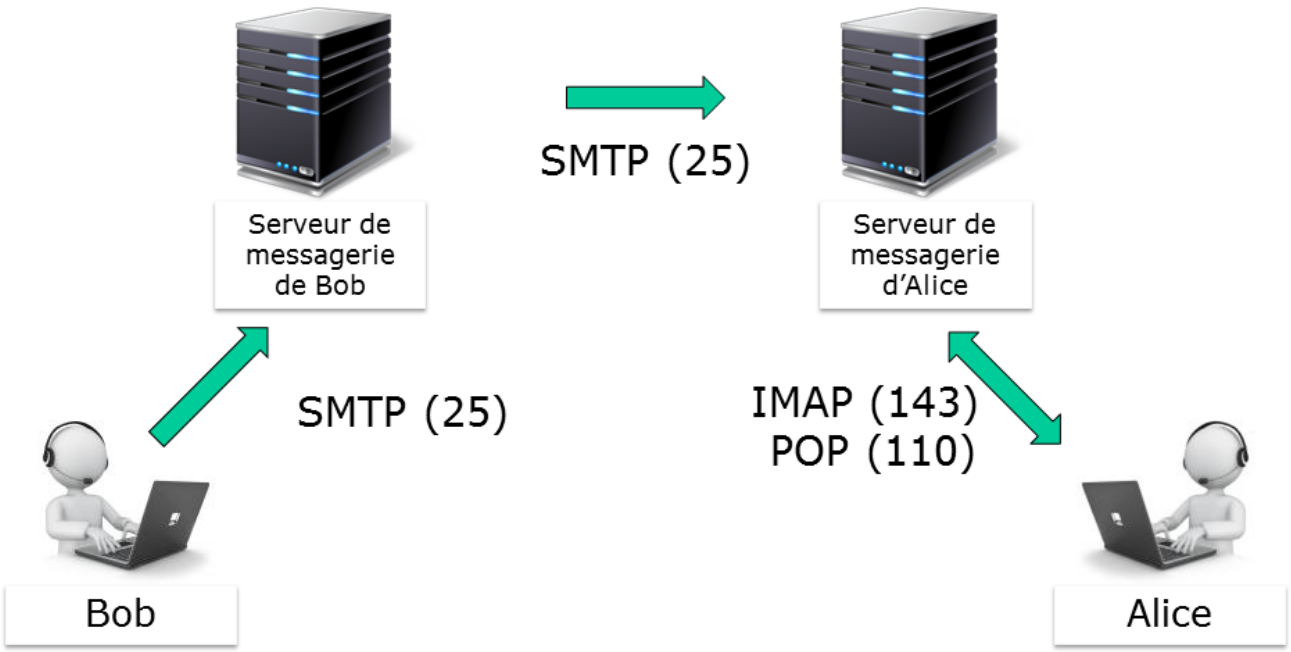


Figure 50. Les protocoles de messagerie mis en oeuvre

Les agents de messagerie

Mail User Agent

Un **Mail User Agent** (**MUA**), ou client de messagerie, est un logiciel servant à **lire** et à **envoyer** des courriers électroniques. Ce sont en général des clients lourds, mais il existe aussi des applications Web : les **webmails**. Les webmails tentent d'offrir les mêmes fonctionnalités qu'un client lourd.

Ces logiciels prennent en charge le protocole **SMTP** pour l'envoi de messages et les protocoles **POP** et **IMAP** pour leur réception.

Parmi les MUA les plus connus, citons : Thunderbird (Mozilla), Evolution (Novell), Outlook et Windows Mail (Microsoft), Kmail, Lotus Notes, Apple Mail, Opéra Mail, etc.

Mail Transfer Agent

Lorsqu'un message est envoyé depuis un client de messagerie **MUA**, il est transféré au serveur de courrier, le **Mail Transfer Agent** (**MTA**). Un MTA implémente à la fois le **client** (tranfert de mail) et le **serveur** (réception) du protocole SMTP.

Les termes Mail Server, Mail Exchanger, Mail Relay et MX Hosts peuvent aussi faire référence à un serveur MTA.

Le système de nom de domaine **DNS** associe à un domaine un ou plusieurs serveur de messagerie (Mail Exchanger - MX) par ordre de priorité (la plus haute priorité étant la plus faible valeur). Un MTA peut donc transférer un mail pour lequel il n'est pas destinataire soit à un serveur relais (si configuré), soit au serveur désigné par l'enregistrement MX du système DNS (le saut suivant - Next Hop).

Les MTA font donc en sorte qu'un message soit délivré d'un système à un autre. Si le MTA ne peut ni accepter, ni relayer un message, il le renvoie à son expéditeur.

Le message arrive au MTA responsable du domaine qui le stocke dans la boîte aux lettres de l'utilisateur.

Parmi les MTA les plus connus, citons : Postfix, Sendmail, Exim, Exchange, Qmail, Lotus Notes.

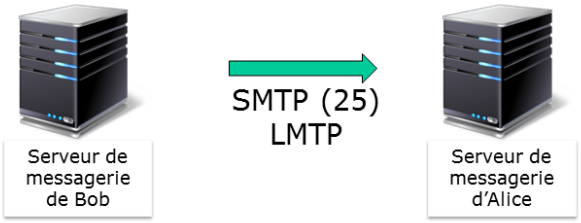


Figure 51. Le transfert de message entre MTA

Mail Delivery Agent

Le **Mail Delivery Agent** (**MDA**), agent de **distribution du courriel**, est le logiciel qui intervient dans la dernière étape du processus de distribution d'un courrier électronique. Il est responsable de la **disposition du message dans la boîte aux lettres** de l'utilisateur. Il peut également être appelé **Local Delivery Agent** (**LDA**).

C'est le MDA qui est chargé de gérer les problèmes comme un disque plein, une corruption de la boîte aux lettres, etc. et de signaler au MTA toute erreur de distribution.

Le MTA communique avec le MDA par l'intermédiaire des canaux d'entrées-sorties standards ou par un protocole spécialisé comme LMTP ou UUCP.

Parmi les MDA les plus connus, citons : Dovecot, Cyrus, Procmail, Local.

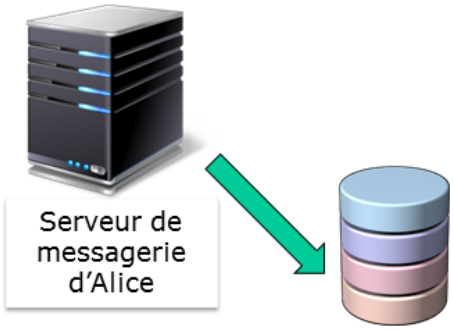


Figure 52. Le rôle de MDA

Mail Access Agent

Le **Mail Access Agent** (**MAA**), permet à l'utilisateur final, après **authentification**, de récupérer le message. Le MUA de l'utilisateur communique avec le MAA par le protocole IMAP ou POP.

Parmi les MAA les plus connus, citons : Dovecot, Cyrus, Courier, Exchange.

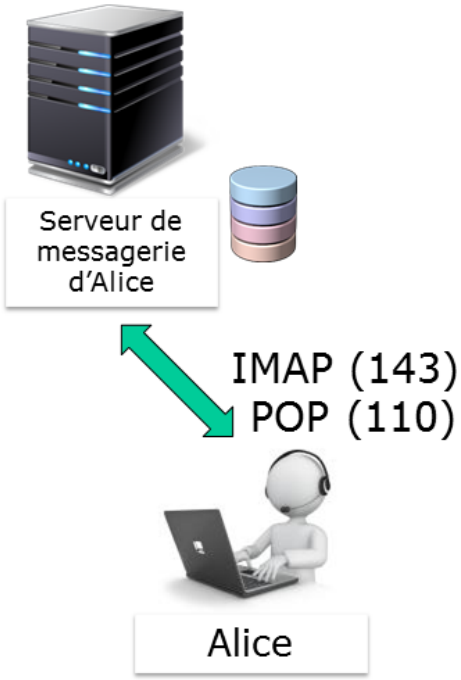


Figure 53. Le rôle de MAA

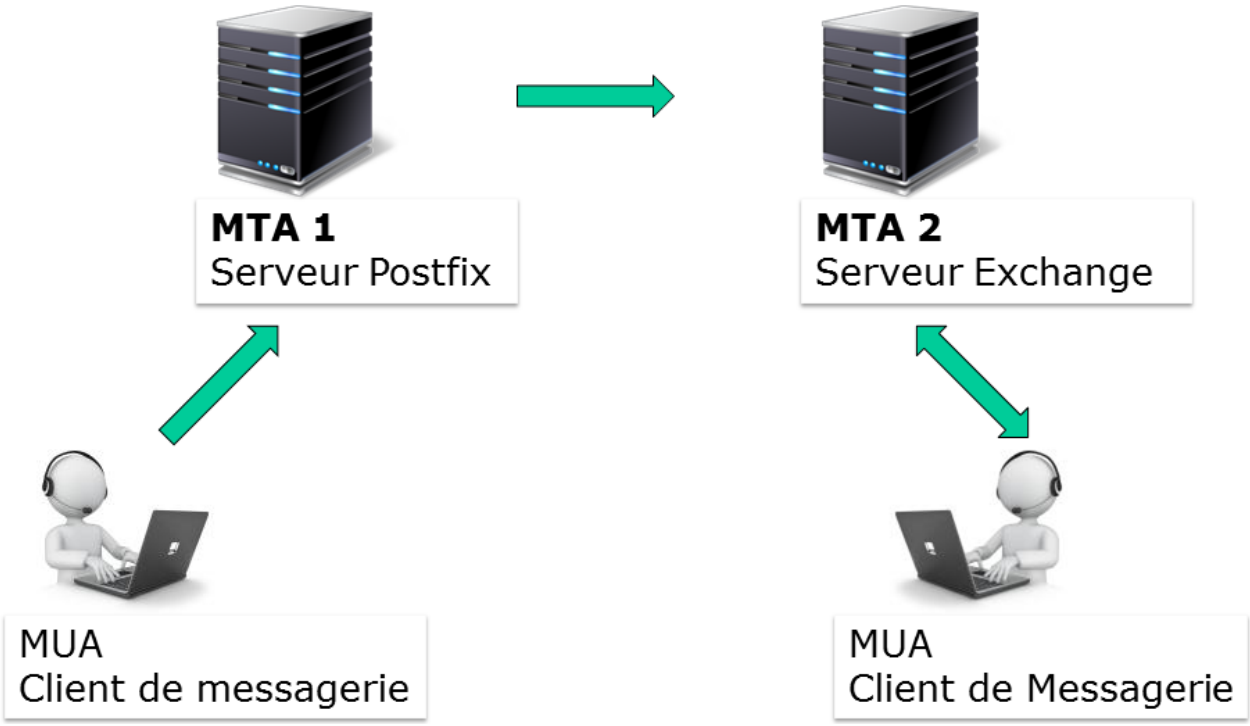


Figure 54. Les agents de transmission de message

Les relais SMTP

Le terme de "**relais de messagerie**" est couramment utilisé, dans le cas d'un MTA qui n'assurerait pas lui même la livraison du message au MTA final, mais qui se contenterait de servir d'intermédiaire entre le MUA du client et le MTA qui prend réellement l'acheminement du message en charge.

Cette fonctionnalité de relais est courante. Elle est par exemple présente dans nos box internet, qui

acheminent l'ensemble des messages émis par un domicile vers un des serveurs MTA centraux. Les messages peuvent ensuite être filtrés (lutte anti-spam ou surveillance ?). Les fournisseurs d'accès à Internet évitent, en bloquant le port 25 en sortie des box, que des serveurs SMTP soient directement sollicités.

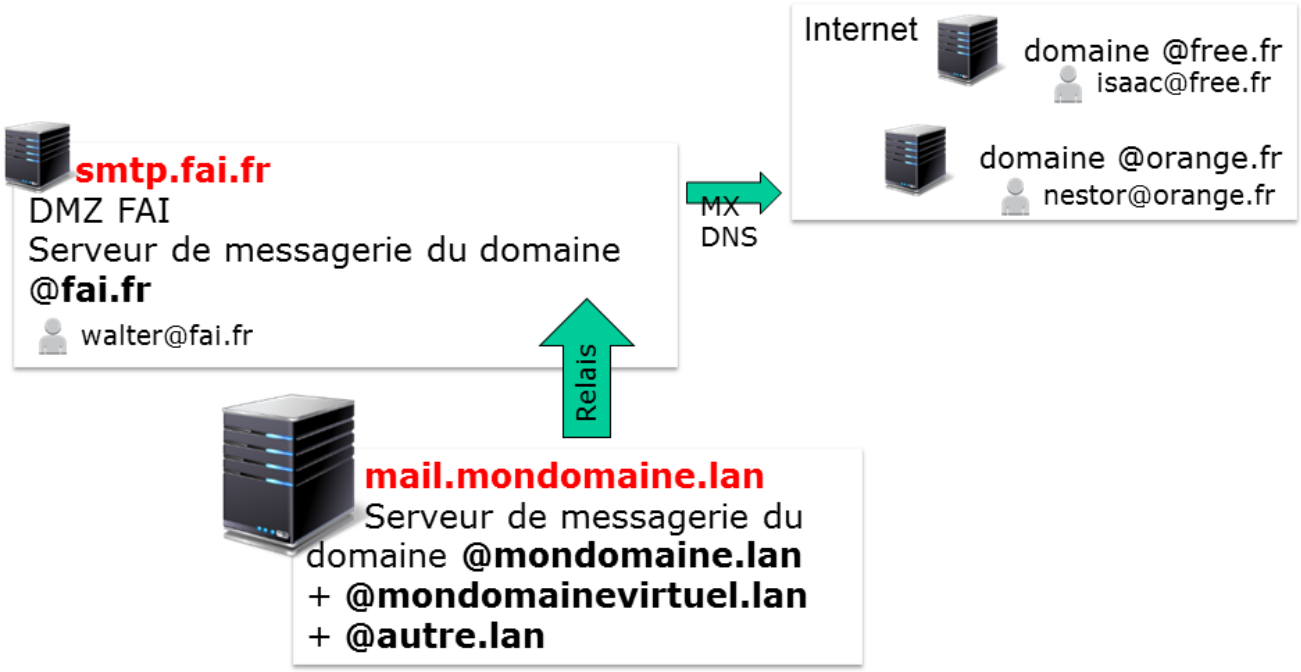


Figure 55. Système de messagerie avec relais

Les formats de stockage

Le format **mbox** est un format ouvert de stockage de courrier. Il repose sur le principe d'attribuer un fichier à chaque dossier (au lieu d'un fichier par message ou d'un répertoire par dossier).

Le format mbox permet un **affichage rapide d'une liste de mail**, puisqu'il ne nécessite l'ouverture que d'un seul fichier. La suppression ou l'ajout de mail est plus long et plus complexe à réaliser d'un point de vue système. L'accès concurrent à la même boîte aux lettres n'est pas possible car un verrou est positionné sur le fichier lors d'une action d'ajout ou de suppression.

Le format mbox est le format par défaut de postfix

Le format **Maildir** est une **structure de répertoires particulière**. Les courriels sont sauvegardés dans un fichier séparé, au sein d'une arborescence spécifique, ce qui lève le problème de verrou du format mbox.

De par son architecture, le format **Maildir est performant et fiable**. Il est plus adapté au protocole IMAP. À noter toutefois que l'affichage d'une liste de mails sera moins rapide qu'avec le format mbox.

Chaque répertoire Maildir contient au moins 3 sous-répertoires : **tmp**, **new**, **cur**. Les mails sont placés dans le répertoire tmp, puis déplacés par le MTA dans le répertoire new, pour enfin être déplacés après accès par un MUA dans le répertoire cur.

Synoptique

Un **MTA** pourra assurer les fonctionnalités suivantes :

- **Serveur de messagerie local** pour les comptes systèmes locaux comme root, bob, alice, etc.
- **Serveur d'un ou de plusieurs domaines de messagerie**, pour des comptes root@mondomain.lan, bob@mondomaine.lan, etc.
- **Serveur relais pour les domaines extérieurs** à son périmètre de gestion.

Un MTA peut également posséder des **routes spéciales**, contenues dans une **table de routage**, pour délivrer des messages à des serveurs de messagerie sans prendre en compte le chemin standard (par le relais ou par le serveur MX désigné lors d'une requête DNS).

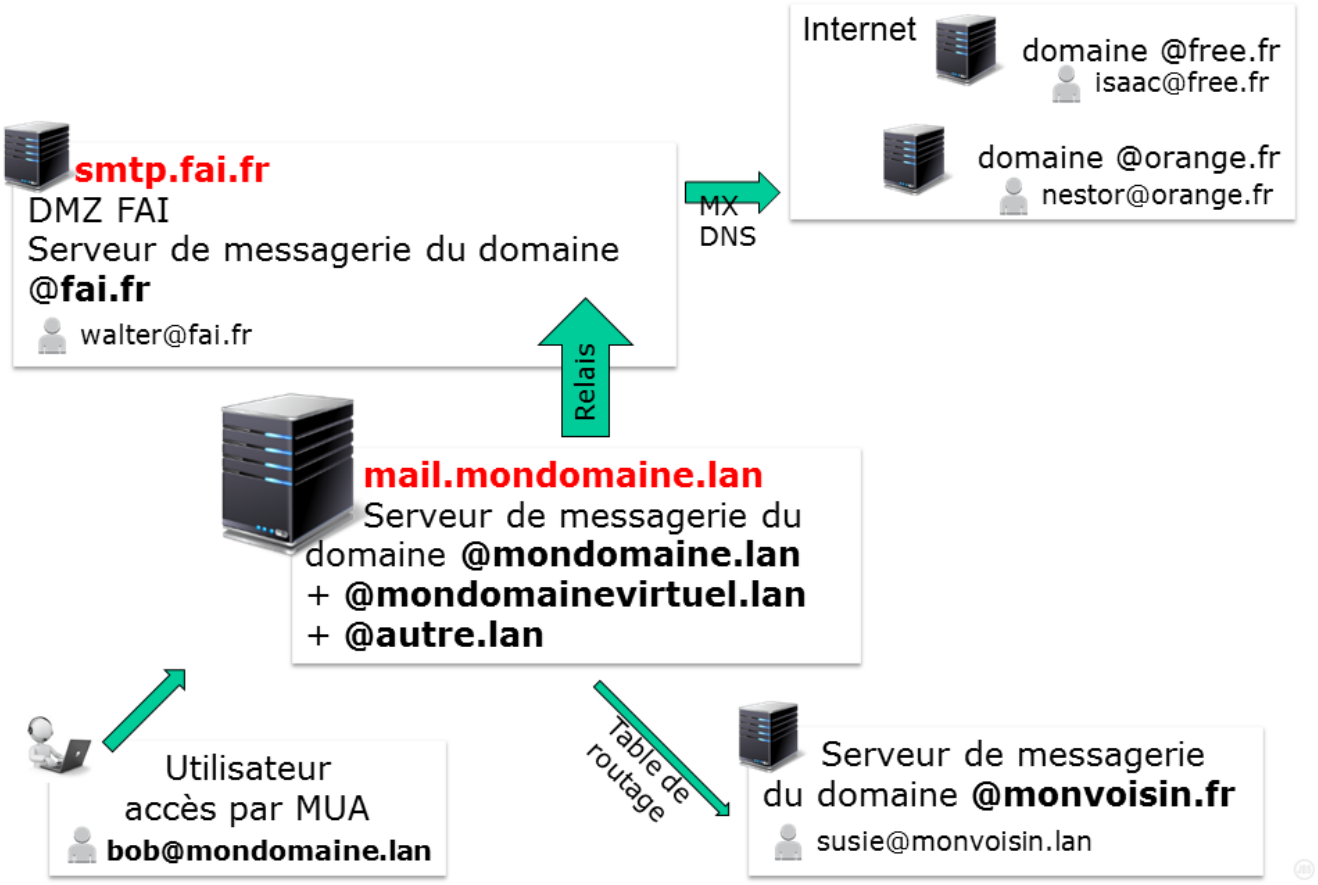


Figure 56. Synoptique global du système de messagerie

8.2. Installation du service

Postfix devrait normalement être installé par défaut sur une RedHat/CentOS 6 ou 7.

```
[root]# yum install postfix
```

Postfix nécessite bien évidemment que le service network soit correctement configuré.

L'installation de postfix ajoute un utilisateur postfix, membre du groupe postfix.

```
[root]# grep postfix /etc/passwd
postfix:x:89:89::/var/spool/postfix:/sbin/nologin

[root]# grep postfix /etc/group
postfix:x:89:x
```

Postfix étant un service réseau, il faut le paramétrer pour un démarrage à minima aux niveaux 3 et 5, puis le démarrer.

L'installation d'un nouveau service sur les distributions RedHat/CentOS n'implique pas leur démarrage automatique.

Après toute installation d'un service, il ne faut pas oublier de le démarrer avec la commande service, et d'automatiser le démarrage au reboot avec la commande chkconfig.

Tout service dépendant du réseau devrait être démarré comme le service network (chkconfig --list network).

```
[root]# chkconfig postfix on
[root]# service postfix start
```

8.3. Arborescence et fichiers

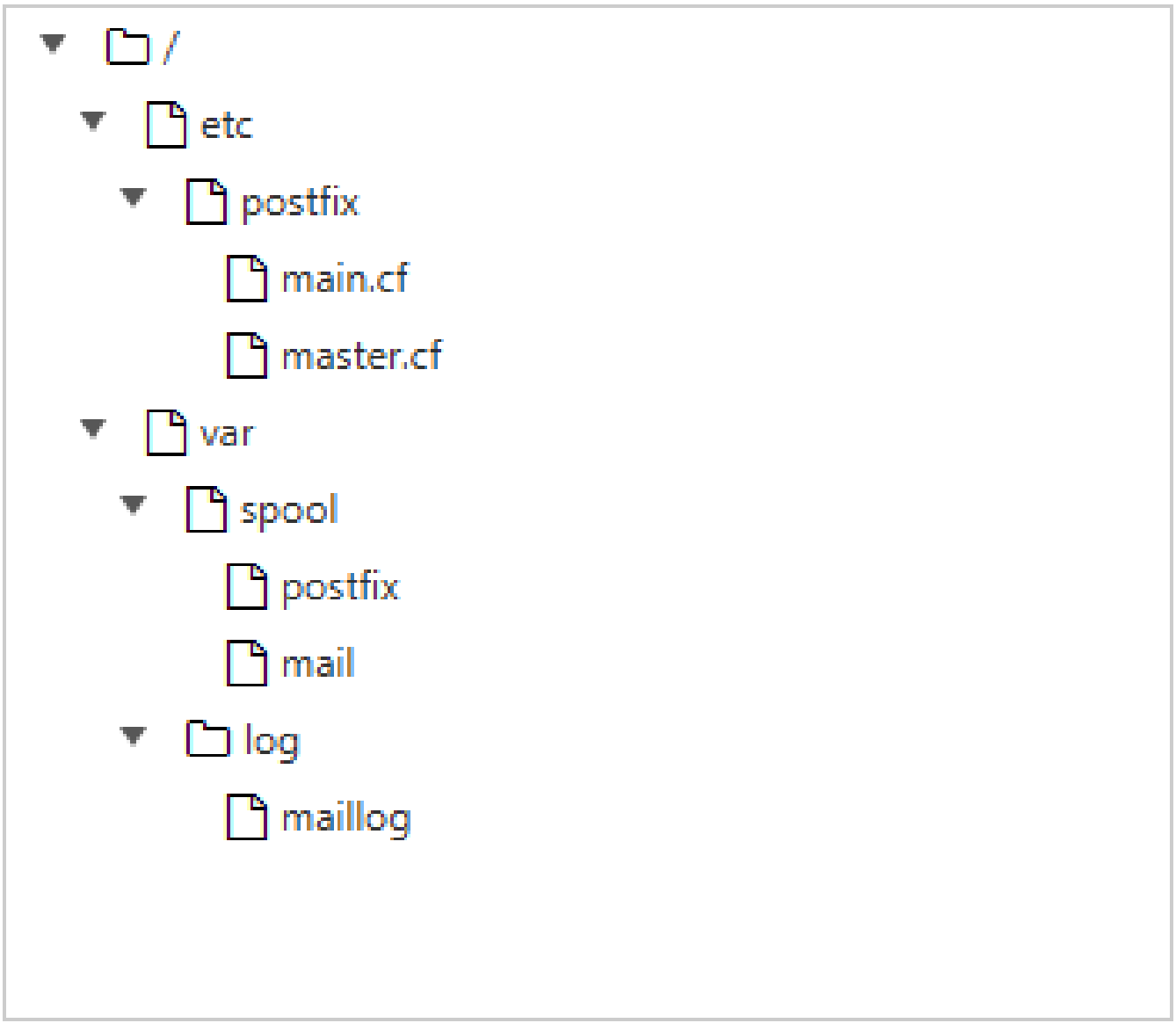


Figure 57. Arborescence et fichiers du serveur postfix

- Le fichier de configuration du serveur Postfix : /etc/postfix/main.cf ;
- Les files d'attente sont groupées dans le répertoire : /var/spool/postfix/ ;
- Les boîtes de messagerie mbox sont stockées dans /var/spool/mail/ ;
- Les logs sont dans le fichier : /var/log/maillog.

Le fichier /etc/postfix/main.cf contient les paramètres de configuration de Postfix. Les paramètres qui n'y sont pas explicitement renseignés sont initialisés avec leur valeur par défaut. Seule la dernière occurence du paramètre compte lorsque ce paramètre est défini plusieurs fois.

En cas de besoin, un fichier commenté de main.cf existe sous /usr/share/postfix/main.cf.

Les directives sont modifiables avec un éditeur de texte, mais la commande postconf permet l'édition en limitant le risque d'erreur.

Dans le fichier /etc/postfix/main.cf :

- Chaque instruction doit être en début de ligne (pas d'espace avant).
- Les espaces autour du signe "=" sont ignorés, comme ceux situés à la fin de la ligne logique.
- Une ligne démarrant avec un espace continue la ligne logique précédente.

8.4. Mise en oeuvre

La commande **telnet** est particulièrement adaptée pour tester des protocoles en mode texte tel SMTP.

Sa syntaxe est la suivante :

```
[root]# telnet localhost 25
```

Pour communiquer avec le serveur, il faut respecter les étapes attendues par le service.

Déroulé d'une session telnet sur le port 25 :

1. HELO
2. MAIL FROM:
3. RCPT TO:
4. DATA

Pour terminer la rédaction du mail, il conviendra de saisir un . sur une ligne seule.

Déroulé d'une session SMTP

Lancer dans un terminal la commande telnet localhost 25. Voici le déroulé de la session :

```
[root]# telnet localhost 25
Trying ::1...
Connected to localhost.
Escape character is '^]'.
220 mail.mondomaine.lan ESMTP Postfix
HELO mail.mondomaine.lan
250 mail.mondomaine.lan
MAIL FROM: bob
250 2.1.0 Ok
RCPT TO: alice
250 2.1.5 Ok
DATA
354 End data with<CR><LF>.<CR><LF>
From: test
To: monalice
Subject: Test de message

Ceci est un test.
Merci de votre coopération

.
250 2.0.0 Ok: queued as 642A59F6E8
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

Les champs From: et To: du contenu du mail (après DATA) ne sont pas vérifiés par le serveur SMTP. Ils peuvent différer des valeurs fournies au service SMTP, un peu comme l'adresse du destinataire d'un courrier postal peut différer de l'adresse affichée dans le courrier.

Le message se termine lorsque le serveur reçoit une ligne ne contenant qu'un point.

Suivi du mail

Le traitement du mail par les différents agent peut être suivi dans le fichier **/var/log/maillog**. Dans ce cas, l'utilisation de la commande *tail -f nomdufichier* est particulièrement bien adaptée. L'utilisation de la commande **ccze**, qui permet la coloration syntaxique des fichiers de log, peut également être utilisée via un **grep** : *tail -f /var/log/maillog | grep ccze*.

Voici un extrait du fichier de log, généré par la session telnet précédente :

```
[root]# tail -f /var/log/maillog  | grep ccze
postfix/smtpd[19747]: connect from localhost[::1]
postfix/smtpd[19747]: 642A59F6E8: client=localhost[::1]
postfix/cleanup[19828]: 642A59F6E8: message-id=<...>
postfix/qmgr[19742]: 642A59F6E8: from=<bob@mail.mondomaine.lan>, size=462, nrcpt=1
(queue active)
postfix/local[20100]: 642A59F6E8: to=<alice@mail.mondomaine.lan>, orig_to=<alice>,
   relay=local, ..., status=sent (delivered to mailbox)
postfix/qmgr[19742]: 642A59F6E8: removed
postfix/smtpd[19747]: disconnect from localhost[::1]
```

L'agent **smtpd** a pris en charge la connexion du client par le réseau en passant par **telnet** sur le port **25**. Si la connexion avait été faîte via un outil local, c'est l'agent **pickup** qui aurait alors pris en charge le message, comme nous le verrons à la section suivante.

L'agent **cleanup** a ensuite pris en charge le message. Le destinataire d'origine, bob, n'étant pas **pleinement qualifié** et non conforme à la norme **RFC 822**, **cleanup** l'a fourni au démon **trivial-rewrite** (évènement qui est non journalisé) pour permettre la réécriture de l'adresse de messagerie d'origine.

L'agent **cleanup** a ensuite déposé le message dans la file d'attente **incoming**, en attendant que l'agent **qmgr** le déplace dans la file **active** (queue active).

Le message ayant une portée locale (**orig_to=<alice>**), l'agent **trivial-rewrite** est de nouveau appelé pour rendre conforme cette adresse (**to=alice@mail.mondomaine.lan**) par l'agent **qmgr**, qui le délivre à l'agent **local** pour stockage dans la boîte aux lettres d'**Alice**.

L'agent qmgr supprime alors le message de la file active.

La commande

La commande mailx est une commande de traitement du courrier (un MUA) dont nous n'étudierons que la partie envoi du courrier.

```
mailx [-iInv] [-s sujet] [-a en-tete] [-c adresses cc] [-b adresses bcc] adresse[s]
```

Le tableau suivant récapitule les principales options :

Table 90. Options principales de la commande mailx

Options	Information
-v	Affiche les détails de la livraison sur le terminal
-s	Spécifie le sujet en ligne de commande (seul le premier argument après le flag -s est utilisé en tant que sujet ; pensez à mettre des guillemets autour des sujets contenant des espaces).

Options	Information
-c liste	Liste les destinataires en copie carbone. 'liste' doit être une liste de noms séparés par des virgules.
-b	Liste les destinataires en copie cachée (Blind Carbon Copy).

Quelques options supplémentaires :

Table 91. Options supplémentaires de la commande mailx

Options	Information
-i	Ignore les signaux d'interruption du terminal. Cela est particulièrement utile lors de l'utilisation de mail sur des lignes téléphoniques à bruit.
-l	Force mailx à se lancer en mode interactif même losque l'entrée n'est pas un terminal. En particulier, le caractère de commande spécial ~, utilisé lors de l'envoi d'un courrier, est seulement disponible interactivement.
-N	Désactive l'affichage initial des en-têtes du message lors de la lecture d'un courrier ou de l'édition d'un dossier de courriers.
-a	Spécifie des champs d'en-tête additionnels dans la ligne de commande comme "X-Loop: foo@bar", etc. Vous devez utiliser des guillemets si la chaîne contient des espaces. Cet argument peut être spécifié plus d'une fois, les en-têtes étant dans ce cas concaténés.
-e	N'envoie pas de courrier vide. Si le corps est vide, le message est sauté.
-f nom	Procède à la lecture du contenu de votre boîte aux lettres (ou le fichier spécifié nom) ; lorsque vous quittez, mail réécrit les messages non supprimés dans ce fichier.
-u utilisateur	Est équivalent à "mail -f /var/mail/utilisateur" sauf qu'il y a verrouillage.

Exemples :

```
[stagiaire]$ less corps
    Bonjour
    Au revoir
[stagiaire]$ mailx -s "coucou" "alice,bob" < corps
```

```
[stagiaire]$ mailx -s "coucou" alice@mail.etrs.lan
    ->Bonjour
    ->Au revoir
    ->. (ou ctrl+d)
```

Suivi du mail

Voici un extrait du fichier de log, généré par la session mailx précédente :

```
[root]# tail -f /var/log/maillog
postfix/pickup[19741]: 5707A9F8A: uid=1000 from=<stagiaire>
postfix/cleanup[22647]: 5707A9F8A: message-id=<...>
postfix/qmgr[19742]: 5707A9F8A: from=<stagiaire@mail.mondomaine.lan>, size=549, nrcpt
=2 (queue active)
postfix/local[22649]: 5707A9F8A: to=<alice@mail.mondomaine.lan>, orig_to=<alice>,
   relay=local, ..., status=sent (delivered to mailbox)
postfix/local[22650]: 5707A9F8A: to=<bob@mail.mondomaine.lan>, orig_to=<bob>,
   relay=local, ..., status=sent (delivered to mailbox)
postfix/qmgr[19742]: 5707A9F8A: removed
```

Les messages générés **localement** (mailx, php, scripts, etc.) sont placés par **sendmail** dans la file d'attente **maildrop**.

Le message ayant été émis par la commande local mailx, c'est l'agent **pickup** qui a pris en charge le message placé dans maildrop.

L'agent **cleanup** a ensuite pris en charge le message, notamment en le fournissant au démon **trivial-rewrite** (non journalisé par défaut) pour permettre la réécriture des adresses de messagerie locale (FROM: root et TO: alice et TO: bob) en adresses conformes à la norme RFC 822 (FROM: root@mail.etrs.lan, TO: alice@mail.etrs.lan et TO: bob@mail.etrs.lan).

Cleanup a ensuite déposé le message dans la file d'attente **incoming**. De la file d'attente **incoming**, le message est passé dans la file active (**queue active**) par l'agent **qmgr**.

Le message ayant une portée local, il est transmis à l'agent **local** pour stockage dans la boîte aux lettres d'Alice puis de nouveau transmis à l'agent **local** pour stockage dans la boîte aux lettres de Bob.

qmgr supprime alors le message de la file **active**.

Utilisation interactive de mailx

```
[alice]$ mailx
Heirloom Mail version 12.4 7/29/08. Type ? for help.
"/var/spool/mail/alice": 2 messages 2 new
>N 1 test@mail.etrs.lan ...    ... "Test de mail"
 N 2 stagiaire                    "coucou"
& _
```

La première ligne identifie la version de mail utilisée.

La deuxième désigne la boite aux lettres.

Le N (new) placé au début de la ligne indique qu'il s'agit d'un nouveau message, tandis que la lettre U (unread) indique qu'elle n'a pas encore été lue lors de la session précédente du programme mailx.

Pour lire le mail, il faut saisir après l'esperluette (et commercial) le numéro du mail à lire.

Pour quitter, simplement saisir la lettre q.

Lorsque la touche q est saisie, mailx sauvegarde le contenu de la boîte aux lettres dans le fichier mbox du répertoire personnel de l'utilisateur, ainsi que les éventuelles modifications ou suppressions effectuées.

Le fichier /home/alice/mbox doit maintenant contenir les messages qui ont été lus.

Table 92. Gestion des mails avec mailx

Options	Information
num	Affiche le mail n° num
d num	Supprime le mail num
h	Affiche la liste des mails
q	Quitte mailx

La commande

La commande swaks (SWiss Army Knife for Smtp) est un outil de test orienté transaction pour SMTP.

Syntaxe de la commande swaks

```
swaks --to user@etrs.lan --server smtp.etrs.lan
```

8.5. Configuration du serveur

La commande **postconf** permet la configuration de Postfix.

Syntaxe de la commande postconf

```
postconf [-d] [-e] [-n] ['directive']
```

Exemple :

```
postconf -e 'myhostname = mail.etrs.fr'
```

Table 93. Options principales de la commande postconf

Options	Information
-d	Affiche les valeurs par défaut des paramètres
-e	Modifie le fichier main.cf avec le paramètre précisé.
-n	Affiche seulement les valeurs qui ne sont pas celles par défaut.

Utilisé sans option ni argument, la commande postconf affiche la configuration courante de Postfix.

La commande postfix check vérifie la configuration du fichier **main.cf** :

```
[root]# postfix check
```

Lorsque la configuration est correcte, la commande ne retourne aucune information.

Les alias

Comment rediriger une adresse vers une autre ? Comment créer une liste de diffusion ? Comment créer une adresse en prenom.nom ? En utilisant les **alias** !

Les alias sont contenus dans le fichier /etc/aliases :

/etc/aliases

```
...
postmaster:    root
...
# Person who shold get root's mail
root:          bob
# Alias locaux
bob.leponge:   bob
# Liste de diffusion
admins:        bob,alice
```

Les modifications sont prises en compte avec la commande **newaliases** :

```
[root]# newaliases
```

Suivi des mails

Suivi d'un mail généré avec mailx à root :

```
postfix/local[25354]: 12C969F84F: to=<bob@mail.etrs.lan>, orig_to=<root>, relay=local,
..., status=sent (delivered to mailbox)
```

Suivi d'un mail généré avec mailx à admins :

```
postfix/local[25639]: 8DD1A9F84F: to=<bob@mail.etrs.lan>, orig_to=<admins>, relay
=local, ..., status=sent (delivered to mailbox)
postfix/local[25639]: 8DD1A9F84F: to=<alice@mail.etrs.lan>, orig_to=<admins>, relay
=local, ..., status=sent (delivered to mailbox)
```

Suivi d'un mail généré avec mailx à bob.leponge :

```
postfix/local[25920]: 0CD8B9F84F: to=<bob@mail.etrs.lan>, orig_to=<bob.leponge>,
relay=local, ..., status=sent (delivered to mailbox)
```

Configurer un serveur relais

Mon serveur est protégé par une DMZ ! Mon fournisseur d'accès bloque le protocole SMTP ! Comment faire ?

Lorsque le serveur n'est pas directement connecté à Internet, il faut utiliser un serveur relais !

Les messages à destination d'utilisateurs non locaux sont relayés par le serveur MTA de la DMZ ou le MTA du fournisseur d'accès.

Configuration du relais

```
[root]# postconf -e 'relayhost = [svrmail.etrs.terre.defense.gouv.fr]'
[root]# service postfix reload
[root]# mailx bob.leponge@free.fr
```

Suivi du mail :

/var/log/maillog

```
postifx/smtp[26595]: 0D7809F84F: to=<bob.leponge@free.fr>,relay
=svrmail.etrs.terre.defense.gouv.fr[XXX.XXX.XXX.XXX]:25, ⋯, status=sent (250 2.0.0
Ok: queued as 2997B686008)
```

Lorsqu'un serveur n'est pas **directement** connecté à l'Internet, les mails qu'il émet devront être envoyés à un serveur intermédiaire : un **MTA relais**.

Il faut alors renseigner la directive **relayhost**.

Pour ne pas utiliser de résolution DNS sur le champ MX du domaine, il convient de mettre le FQDN ou l'adresse IP du serveur relais entre crochets.

Dans l'exemple au dessus, le serveur 'svrmail.etrs.terre.defense.gouv.fr' étant le serveur de messagerie pointé par l'enregistrement MX du DNS pour le domaine 'etrs.terre.defense.gouv.fr', les deux configurations suivantes sont identiques, mais la deuxième affranchit le serveur d'une requête DNS :

```
[root]# postconf -e 'relayhost = etrs.terre.defense.gouv.fr'
```

est identique à :

```
[root]# postconf -e 'relayhost = [svrmail.etrs.terre.defense.gouv.fr]'
```

Le message est cette fois-ci transmis par l'agent **mgr** au démon **smtp**, chargé de transférer le message via le protocole **SMTP**.

Prendre en compte un domaine

Je veux transformer mon serveur de messagerie, je voudrais centraliser les messages pour tout le domaine etrs.lan !

Il faut configurer les directives mydomain et mydestination !

Avant de modifier la configuration du serveur, envoyer un mail à bob@etrs.lan :

```
[root]# mailx bob@etrs.lan
```

Suivi du message :

```
postifx/smtp[27446]: B522C9F84F: to=<bob@etrs.lan>,relay
=svrmail.etrs.terre.defense.gouv.fr[172.16.160.7]:25, ⋯, status=sent (250 2.0.0 Ok:
queued as CB201686008)
```

Le serveur ne sait pas pour l'instant qu'il doit délivrer à l'agent local ce message.

Les directives mydomain et mydestination

La directive **mydomain** contient le nom de domain internet du système de messagerie. Par défaut, $mydomain vaut $myhostname oté de son premier composant.

Si $myhostname vaut serveur.etrs.lan alors $mydomain vaut etrs.lan.

La directive **mydestination** liste les domaines livrés par l'agent local.

Pour visualiser la valeur par défaut de la directive mydestination :

```
[root]# postconf 'mydestination'
mydestination = $myhostname, localhost.$mydomain, localhost
```

Le serveur transmettra donc à l'agent local tous les mails correspondant à @serveur.etrs.lan, @localhost.etrs.lan et @localhost.

Pour ajouter le domaine etrs.lan à la liste des domaines gérés localement :

```
[root]# postconf -e 'mydestination = $myhostname, localhost.$mydomain, localhost,
$mydomain'
[root]# postconf -e 'mydomain = etrs.lan'
[root]# service postfix reload
```

Le même test que précédement peut être rejoué :

```
mailx bob@etrs.lan
```

Suivi du mail :

/var/log/maillog

```
postfix/local[28603]: AE6D99F84F: to=<bob@etrs.lan>, relay=local, ⋯, status=sent
delivered to mailbox)
```

Le message est cette fois-ci pris en compte par le serveur est délivré à une boîte aux lettres locale via le démon **local**.

La directive mynetworks

Par défaut, le serveur postfix refuse de prendre en compte des messages provenant du réseau (excepté depuis sa loopback localhost), ce qui aurait pour effet de devenir un serveur OpenRelay à la merci des spammers.

Maintenant que les clients du réseau local disposent d'une boîte mails, ils vont devoir envoyer leur messages au serveur. Ils doivent donc avoir accès à postfix via le réseau.

Il faut configurer postfix pour qu'il accepte les connexions réseaux, tout en prenant soin de le limiter aux connexions du réseau local.

Dans un premier temps, il convient d'autoriser postfix à écouter sur toutes les interfaces réseaux :

```
[root]# postconf -e 'inet_interfaces = all'
```

Dans un second temps, il faut vérifier que le firewall autorise les connexions (au moins sur le port 25).

La directive **mynetworks** précise les réseaux autorisés à envoyer des messages sur le serveur.

```
mynetworks = 192.168.96.0/19
```

La directive **mynetworks_style** est ignorée si **mynetworks** est définie. Sinon elle précise le type d'hôtes autorisés à envoyer des messages au serveur (host, subnet ou class).

```
# autoriser tous les hôtes de mon sous-réseau
mynetworks_style = subnet
```

Attention à ne pas devenir un serveur "open-relay", et ainsi servir de serveur relais pour les spammeurs.

Le format de stockage

Par défaut, postfix stocke les mails au format mbox.

Pour passer du format mbox au format maildir :

```
[root]# postconf -e 'home_mailbox = Maildir/'
[root]# service postfix reload
```

Le changement pourra être vérifié :

```
[root]# mailx bob@etrs.lan
[root]# ls /home/bob/Maildir/new/
```

La table de routage

J'aimerais que les messages vers monvoisin.fr ne passe pas par le serveur relais. Il faudrait définir une route spécifique !

La table de routage va permettre de définir un serveur relais pour un domaine donné. Le fichier par défaut à renseigner est /etc/postfix/transport. Plusieurs enregistrements différents peuvent être présent.

L'exemple ci-dessous permet de définir que tous les messages à destination du domaine **monvoisin.fr** seront directement envoyés en smtp au serveur d'adresse IP spécifiée sans tenir compte d'un relais. Les crochets permettent de s'affranchir de la requête DNS de type MX.

/etc/postfix/transport

```
...
monvoisin.fr        smtp:[172.16.96.100]:25
...
```

Postfix doit également connaitre dans son fichier configuration l'emplacement du fichier /etc/postfix/transport, dans notre cas :

```
[root]# postconf -e 'transport_maps = hash:/etc/postfix/transport'
```

La prise en compte de la nouvelle route se fait avec la commande postmap :

```
[root]# postmap /etc/postfix/transport
[root]# service postfix reload
```

Après avoir modifié la table des transports, il est nécessaire de lancer la commande **postmap**.

Cette commande permet de transformer le fichier en base de données type clef:valeur interprétable par postfix.

N'oubliez pas de relancer le service postfix.

8.6. Protocoles POP/IMAP

Le serveur Dovecot est un serveur POP3/IMAP orienté vers la sécurité.

Installer le serveur dovecot :

```
[root]# yum install dovecot
[root]# chkconfig dovecot on
```

La configuration de dovecot est répartie entre de nombreux fichiers de configuration.

- Activer le protocole IMAP.
- Le serveur dovecot écoutera sur toutes les interfaces réseaux.

/etc/dovecot/dovecot.conf

```
protocols = imap
listen= *
```

- Spécifier à dovecot que le serveur Postfix stocke les mails au format Maildir (maildir:) dans le

répertoire Maildir du répertoire de connexion de l'utilisateur (~/).

/etc/dovecot/conf.d/10-mail.conf

```
mail_location = maildir:~/Maildir
```

- L'authentification plaintext permet de gérer l'authentification des clients par login et mot de passe qui transitent en clair sur le réseau. Cette option n'est pas sécurisée si elle n'est pas couplée avec un moyen de chiffrement du flux imap (IMAPs) ce qui explique sa désactivation par défaut.

/etc/dovecot/conf.d/10-auth.conf

```
disable_plaintext_auth = no # signifie authentification plaintext enable
```

Redémarrer le service

```
[root]# service dovecot restart
```

8.7. Architecture de postfix

Postfix est un serveur modulaire basé sur des boîtes aux lettres et régi par le démon principal **master**.

Chaque démon assume une fonction, chaque fonction correspondant à une tâche distincte. Les démons sont gérés par le démon master, qui est le premier a être lancé.

Les messages sont stockés dans des files d'attente, où ils sont récupérés par les démons.

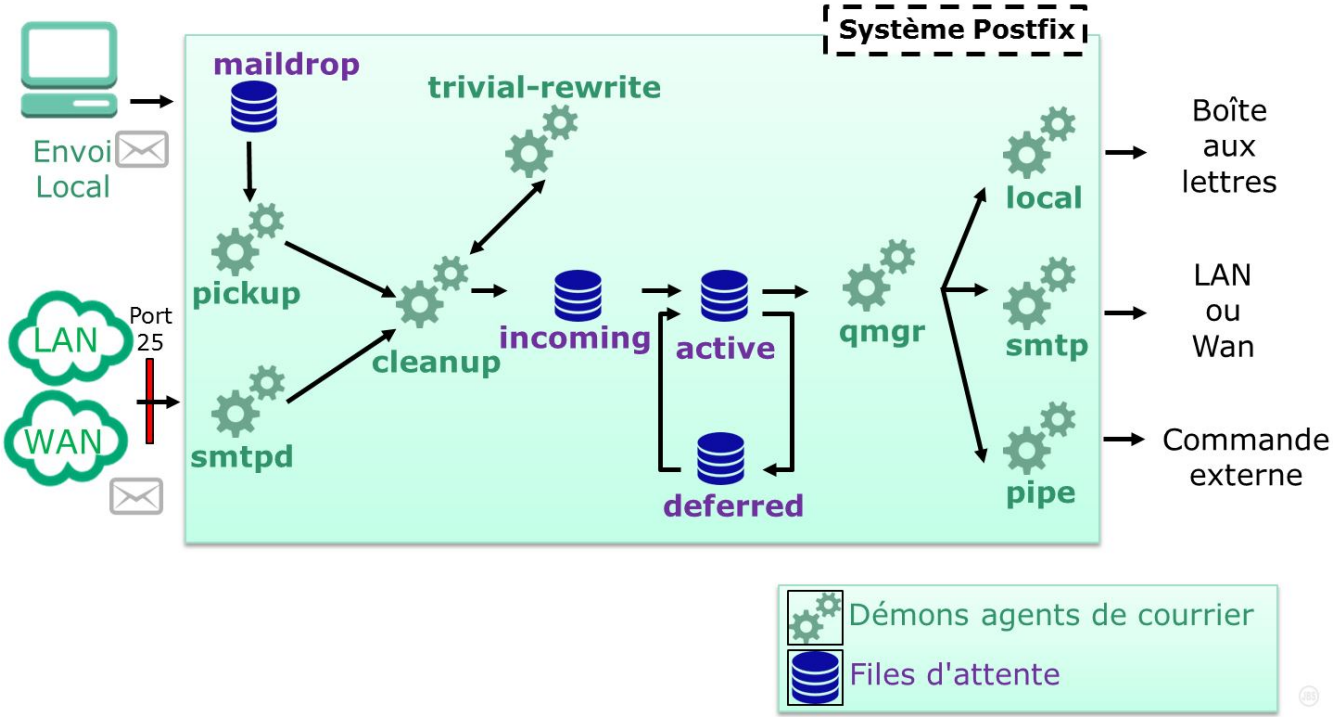


Figure 58. Synoptique global de postfix

Postfix accepte des messages provenant de plusieurs sources :

- Source locale : envoyé par un utilisateur du serveur via un logiciel local (mailx, php, etc.) ;
- En provenance du réseau connecté au serveur ;
- Produit par Postfix lui-même ;
- Un message ressoumis pour être transféré à une autre adresse.

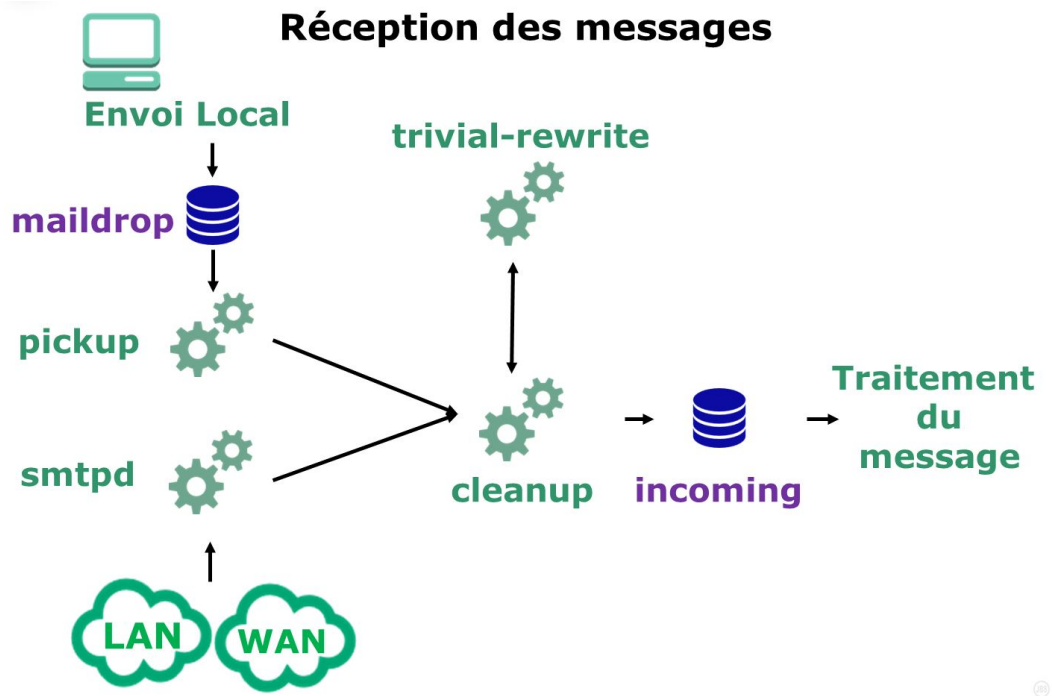


Figure 59. Synoptique de postfix partie réception des messages

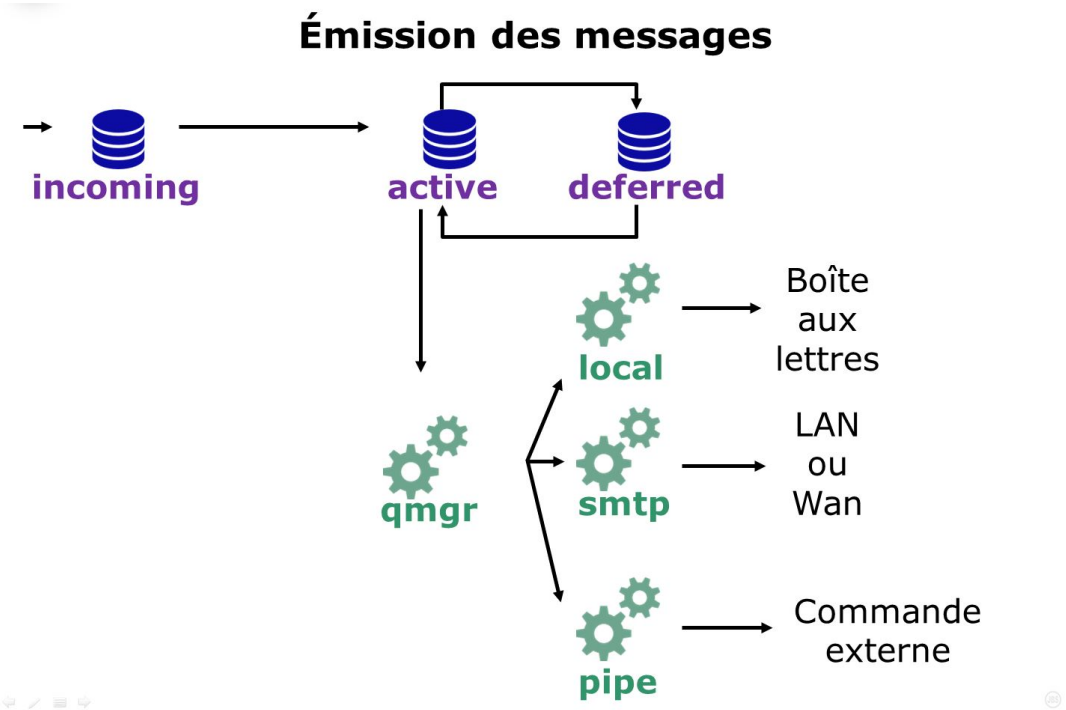


Figure 60. Synoptique de postfix partie émission des messages

Postfix produit lui-même les messages de service pour indiquer la non-réception d'un message ou son report. Ces messages suivent le même cheminement que les messages locaux.

Les messages locaux sont déposés dans la file d'attente maildrop.

Le démon pickup prend les messages dans la file d'attente et les passe à cleanup.

Les messages provenant du réseau sont pris en charge par le démon smtpd. Ce dernier vérifie qu'ils vont pouvoir être traités par le serveur et ensuite il les transfère à cleanup.

Un message se doit de respecter certaines normes de formatage. Les adresses de provenance ou de destination doivent être pleinement qualifiées, il ne doit pas manquer d'en-têtes, etc. Un message ne respectant pas ces règles sera reformaté (remis en forme) par trivial-rewrite. Une fois corrigé, il sera de nouveau pris en charge par cleanup qui le placera dans la file incoming et préviendra le gestionnaire qmgr.

Le gestionnaire de file d'attente qmgr effectue l'essentiel du traitement du courrier. Il gère les files d'attente incoming, active et deferred.

Après traitement par cleanup, les messages sont placés dans la file incoming. Si les ressources systèmes sont disponibles, qmgr déplace alors le message dans la file active et appelle l'un des agents de distribution pour le délivrer.

Les messages qui ne peuvent pas être distribués sont mis dans la file deferred où ils attendent que qmgr tente de nouveau de les distribuer.

Si le message n'est pas à destination d'un utilisateur géré par le serveur, qmgr le transfère au démon smtp qui l'expédie au MTA concerné.

Si le destinataire fait bien partie du domaine géré par le serveur Postfix, le démon qmgr achemine le message vers local.

Le démon local dépose les messages dans l'espace de stockage local des messages. Il contrôle également les alias et vérifie si les messages doivent être délivrés ailleurs.

Le message peut être délivré à un autre processus, comme un gestionnaire de liste de diffusion, ou tout autre processus.

Démons agents de courrier

- **pickup** : collecteur de messages locaux acheminés par *maildrop*. Il fournit à l'agent cleanup les messages déposés dans la file maildrop.
- **smtpd** : collecteur de messages reçus du réseau. L'agent smtpd accepte les connexions réseaux et effectue les transactions SMTP pour fournir les messages à l'agent cleanup.
- **trivial-rewrite** : agent de résolution et de réécriture des adresses (en lien avec le domaine). L'agent trivial-rewrite offre 3 types de services aux requêtes des clients :
  - Réécriture du contexte d'adresse vers une forme standard : ajoute le nom de domaine spécifié par $myorigin ou par $mydomain aux adresses incomplètes des messages postés localement,
  - Résolution de l'adresse en un quadruple (transport, saut suivant, récipient, drapeaux) :
    - Le transport correspond à l'agent de délivrance à utiliser,
    - Le MTA suivant à qui délivrer le mail,
    - L'adresse du destinataire à fournir au prochain MTA,
    - Les drapeaux : la classe d'adresse, si l'adresse nécessite un relais, si l'adresse a un problème ou si la requête a échoué.
  - Résolution de l'adresse de l'envoyeur (pour des besoins de vérifications).
- **cleanup** : agent de formatage des messages selon la norme RFC822. Il traite les messages entrant, les insère dans la file d'attente incoming puis informe qmgr de leur arrivée.
  - L'agent cleanup opère toujours ces transformations :
    - Ajout des en-têtes manquantes : From:, To:, Message-Id: et Date:.
    - Transforme au besoin les adresses de l'enveloppe et des en-têtes au standard utilisateur@fqdn qui est attendu par les autres agents postfix. Cette tâche est déléguée à l'agent trivial-rewrite.
    - Supprime les adresses dupliquées de l'enveloppe,
    - Supprime les en-têtes : Bcc:, Content-Length:, Resent-Bcc, Return-Path:.
  - Optionnellement, les adresses peuvent être transformées en fonction de la table canonical ou virtual et du paramètre masquerade_domains,
- **qmgr** : agent de gestion des files d'attente *active* et *deferred*. L'agent qmgr attend l'arrivée de message entrant et s'assure de leur livraison via un des agents de livraison. La stratégie de

routage des messages est délégué au démon trivial-rewrite.

  - qmgr maintient les files d'attente incoming, active, deferred, corrupt et hold.
  - qmgr surveille les rapports de livraison par message dans les répertoires suivant et demande aux agent concernés d'envoyer les rapports :
    - bounce : rapport des messages refusés. Cette file est maintenue par l'agent bounce.
    - defer : rapport des messages retardés. Cette file est maintenue par l'agent bounce.
    - trace : rapport des messages suivis. Cette file est maintenue par l'agent trace.

- **local** : agent de livraison des messages locaux, MDA. L'agent local met à jour les files d'attente des messages et marque les messages si finis ou informe qmgr si les messages doivent être retraités plus tard. Les messages de livraison sont transmis à l'agent approprié (bounce, defer ou trace).
- **smtp** : agent de livraison des messages vers le réseau. Il implémente les protocoles SMTP et LMTP. Il procède à la livraison des messages à la demande de qmgr. L'agent met à jour les files d'attente des messages et marque les messages si finis ou informe qmgr si les messages doivent être retraités plus tard. Les messages de livraison sont transmis à l'agent approprié (bounce, defer ou trace).
  - L'agent smtp interroge le service DNS pour obtenir une liste d'adresses de serveurs de messagerie MX du domaine du destinataire de message, les trie par ordre de préférence et tente une connection vers chacun jusqu'à ce qu'il en trouve un qui réponde.
- **pipe** : agent de livraison des messages vers une commande externe.
- **bounce** : agent de suivi des messages (informations sur la délivrance des messages). Ce démon procède à deux types de requêtes :
  - Ajoute un enregistrement de (non-)délivrance à un fichier de suivi de message (un fichier par message).
  - Génère un message de notification de délivrance, avec une copie du fichier de suivi du message correspondant. Lorsque le message est correctement généré, le fichier de suivi est supprimé.

Files d'attente

- **maildrop** : messages locaux postés par *sendmail*.
- **incoming** : messages après formatage en attente de traitement. Tous les messages entrant dans le système Postfix sont écrits par l'agent cleanup dans la file incoming. Les nouveaux fichiers sont créés avec comme propriétaire l'utilisateur "postfix" avec des droits à 0600. Une fois que le fichier est prêt a être traité, l'agent cleanup change les droits à 0700 et notifie qmgr d'une nouvelle arrivée. Les messages qui n'ont pas les droits à 0700 sont tout simplement ignorés car considérés comme en cours d'écriture par cleanup.
- **active** : messages prêts a être acheminés. La file d'attente active n'est pas uniquement un ensemble de fichiers sur le disque. La file d'attente "réelle" active comprend également un ensemble de structures de données dans la mémoire de l'agent gmgr, ce qui explique que la

quantité de message traités dans la file active soit limitée, pour éviter un dépassement de mémoire libre.

- **deferred** : messages n'ayant pas pu être livrés et pour lesquels un envoi ultérieur pourrait réussir.

8.8. Boites aux lettres virtuelles

Est-il vraiment utile de créer un compte système Linux pour chaque adresse de messagerie ?

Il est possible d'héberger la messagerie de différents domaines sans associer les boîtes aux lettres à des comptes système.

Les boîtes seront stockées (par exemple) sous /var/mail/vmail et gérées par l'utilisateur vmail (uid=5000, gid=5000).

Créer l'utilisateur virtual mailbox:

```
[root]# groupadd -g 5000 vmail
[root]# useradd vmail -u 5000 -g 5000 -s /sbin/nologin -d /var/mail/vmail
```

/etc/postfix/main.cf

```
virtual_mailbox_domains = mondomaine.com, autre.com
virtual_mailbox_base = /var/mail/vmail
virtual_mailbox_maps = hash:/etc/postfix/vmailbox
virtual_minimum_id = 100
virtual_uid_maps = static:5000
virtual_gid_maps = static:5000
virtual_alias_maps = hash:/etc/postfix/virtual
```

Ne jamais lister ici un domaine renseigné dans la directive mydestination ou virtual_alias_domain.

/etc/postfix/vmailbox

```
bob@mondomaine.com      mondomaine.com/bob/
alice@mondomaine.com        mondomaine.com/alice/
bob@autre.com           autre.com/bob/
```

Le / à la fin des chemins vers les boîtes aux lettres précise qu'ici le format de stockage est Maildir.

/etc/postfix/vmailbox

```
postmaster@mondomaine.com   postmaster
```

Postfix ne peut pas traiter directement les fichiers vmailbox et virtual dans leur format humain. Il a besoin de générer une base de données au format clé-valeur, plus couramment appelée hash-table.

Générer les tables clés/valeurs (hachées) :

```
[root]# postmap /etc/postfix/vmailbox
[root]# postmap /etc/postfix/virtual
```

Ce rôle est rempli par la commande postmap, qui dans notre exemple, générera deux fichiers : vmailbox.db et virtual.db.

Créer les répertoires de stockage :

```
[root]# su – vmail –s /bin/bash
[vmail]$ mkdir /var/mail/vmail/mondomaine.com/
[vmail]$ mkdir /var/mail/vmail/autre.com/
```

Authentification des utilisateurs :

/etc/dovecot/users

```
bob@mondomaine.com:{PLAIN}password:5000:5000::/var/mail/vmail/mondomaine.com/bob/
alice@mondomaine.com:{PLAIN}password:5000:5000::/var/mail/vmail/mondomaine.com/alice/
```

Authentification par fichier plat :

/etc/dovecot/conf.d/10-auth.conf

```
disable_plaintext_auth = no
!include auth-passwdfile.conf.ext
```

Nouvel emplacement de stockage :

/etc/dovecot/conf.d/10-mail.conf

```
mail_location = maildir:/var/mail/vmail/%d/%n
```

Les macros dovecot :

Dovecot est capable de remplacer dynamiquement des valeurs renseignées dans ses fichiers de configuration.

Dans notre exemple, le %d sera remplacé par le domaine de messagerie et le %n par le nom de la boîte aux lettres. Par exemple, un mail destiné à bob@mondomaine.lan sera stocké dans le sous-dossier mondomaine.lan/bob/.

Les comptes de messagerie ne nécessitent plus de compte système, ce qui facilite l'administration et améliore la sécurité.

Les fichiers plats utilisés dans nos exemples peuvent facilement être remplacés par une table mysql ou un annuaire LDAP.

Pourquoi ne pas utiliser uniquement des utilisateurs virtuels dans ce cas ?

8.9. Suivi des messages à des fins légales

Il peut être demandé de conserver le sujet, le rédacteur et le destinataire d'un message.

Pour cela, le processus cleanup doit vérifier les entêtes des messages et générer un log lorsqu'il rencontre la valeur attendue. Ces valeurs sont stockées dans le fichier /etc/postfix/header_checks sous forme de regex :

/etc/postfix/header_checks

```
/^subject:/     WARN
/^Subject:/     WARN
/^to:/          WARN
/^To:/          WARN
/^from:/        WARN
/^From:/        WARN
```

Postfix utilise la directive **header_checks** :

```
postconf -e 'header_checks = regexp:/etc/postfix/header_checks'
service postfix restart
```

Un message généré avec la commande swaks :

```
swaks --to alice@etrs.lan --from bob@etrs.lan --header "Subject: test test test"
--server 127.0.0.1
```

Générera les logs suivants :

```
tail -f /var/log/maillog | grep warning
postfix/cleanup[13423]: 125D162F88: warning: header To: alice@etrs.lan [...]
postfix/cleanup[13423]: 125D162F88: warning: header From: bob@etrs.lan [...]
postfix/cleanup[13423]: 125D162F88: warning: header Subject: test test test [...]
```

Chapitre 9. Serveur d'annuaire OpenLDAP

9.1. Généralités

Le protocole **LDAP (LightWeight Directory Access Protocol)** est une suite **standardisée** de protocoles permettant l'accès à un annuaire centralisé. Cet annuaire centralisé stocke des informations diverses comme :

- des noms ;
- des adresses ;
- des numéros de téléphone ;
- des utilisateurs ;
- des groupes ;
- etc.

La version actuelle de ce protocole est la version 3.

Le protocole LDAP s'inspire de la spécification **X.500** mais de manière moins complexe. La norme X.500 désigne l'ensemble des normes informatiques sur les services d'annuaires définies par l'IUT (International Union Telecommunication). Il s'agit donc d'un annuaire version électronique dont l'organisation est hiérarchique, à la manière du DNS, qui correspond généralement à l'organisation de l'entreprise. Le terme Lightweight ne doit pas être compris comme "allégé", ce qui signifierait qu'il serait amputé de certaines fonctionnalités, mais bien dans ce cas de "simplifié", car la spécification X.500 est très lourde.

Le système LDAP est né en 1993 à l'université du Michigan. Le dérivé libre OpenLDAP est lui apparu en 1998.

Une base de données LDAP, de part sa nature, est optimisée pour la lecture d'informations :

- authentification ;
- recherche dans l'annuaire.

Les ports utilisés par le protocole LDAP sont les ports **389** (en clair comme en chiffré par **startTLS**) et **636** pour une connexion en TLS (solution dépréciée).

L'implémentation la plus célèbre du protocole LDAP sous Windows est l'Active Directory.

Sous Linux, les choix sont nombreux :

- OpenLDAP ;
- RedHat Directory Studio ;
- Samba4 qui intègre un serveur LDAP ;

- LinID de Linagora ;
- 389 DS ;
- IBM Tivoli ;
- ...

À noter que la suite OpenLDAP au sein de RedHat 6 (et versions supérieures) n'utilise plus OpenSSL. Elle utilise à la place l'implémentation de Mozilla de NSS (Network Security Services).

La

Un annuaire LDAP est **un arbre de données**. Cette **structure hiérarchique** est appelée **DIT (Directory Information Tree)**.

Une entrée de l'arbre est un ensemble d'**attributs**.

Chaque entrée dispose d'un identifiant unique : son **DN (Distinguished Name)**.

L'OLC

OpenLDAP est le serveur d'annuaire de référence pour les distributions Linux.

Dans les versions récentes d'OpenLDAP (>2.4), sa configuration n'est plus stockée dans un fichier de configuration.

La configuration réside directement dans la base de données elle-même, au sein d'une **DIT** spécifique : c'est la fonctionnalité **OLC (On-Line Configuration)**, aussi connue sous le nom **cn=config**.

L'approche consistant à stocker 'en ligne' la configuration LDAP peut paraître complexe, mais est justifiée par la criticité du service LDAP. Stocker la configuration dans des fichiers plats imposait un redémarrage du service à chaque modification, ce qui représentait beaucoup de temps d'arrêt pour de grosses bases de données.

Il n'y a plus de fichiers .conf à modifier dans les versions récentes d'OpenLDAP.

Le schéma

Le contenu des entrées d'un annuaire est régi par des schémas. Les schémas définissent les types d'attributs d'une entrée regroupés par classe d'objets.

- schéma : ensemble des classes et des attributs disponibles.
- objectClass : une classe objet rassemble un ensemble d'attributs obligatoires ou facultatifs (par exemple la classe inetOrgPerson).
- attribut : exemple

- mail: john.doe@etrs.lan ;
- preferredLanguage: french.

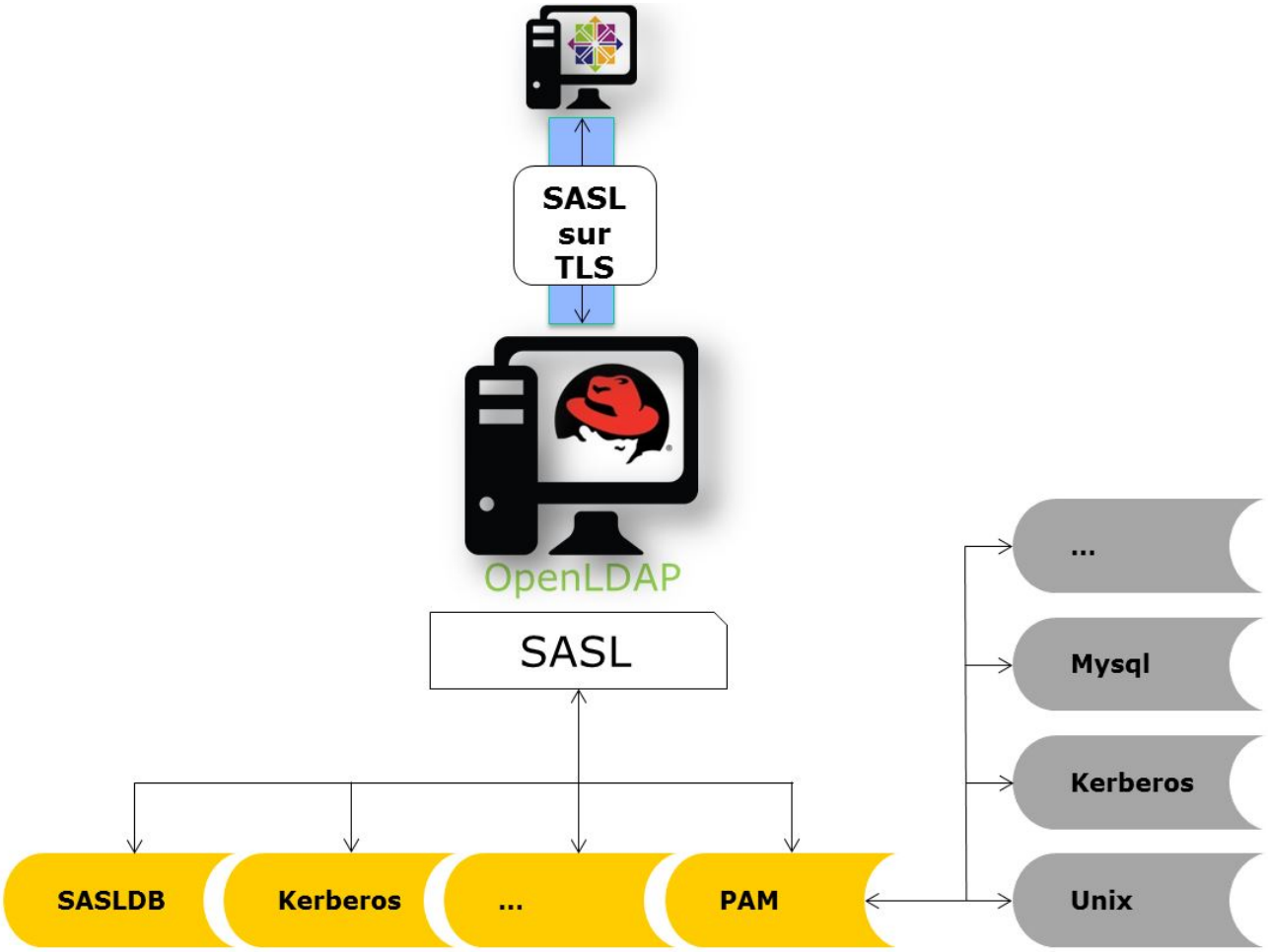
SASL (Couche d'Authenfication et de Sécurité Simple) est une méthode pour ajouter le support d'**authentification** aux **protocoles basés sur la connexion** (LDAP, SMTP, IMAP, XMPP, IRC, etc.). Pour utiliser SASL, un protocole inclut une **commande d'identification et d'authentification** d'un utilisateur sur un serveur et la **négociation éventuelle de la protection** des interactions consécutives du protocole.

Si son utilisation est négociée, **une couche de sécurité est insérée entre le protocole et la connexion**.

Séparer ainsi la couche d'authentification de la couche applicative permet en théorie à n'importe quel mécanisme d'authentification pris en charge par SASL d'être employé à partir de n'importe quel protocole d'application capable d'utiliser SASL.

Les mécanismes principaux SASL sont :

- **EXTERNAL** : l'authentification est dérivée du contexte (authentification système);
- **ANONYMOUS** : accès anonyme sans authentification ;
- **PLAIN** : mot de passe en clair ;
- **OTP** : mot de passe unique (One Time Password) ;
- **CRAM-MD5** et **DIGEST-MD5** : basés sur MD5 ;
- **NTLM** : authentification pour réseau local NT ;
- **GSSAPI** : authentification Kerberos via GSSAPI.



Le format

Le format LDIF (LDAP Data Interchange Format) est un format de fichier texte utilisé lors des échanges d'informations en client/serveur ou entre serveurs.

Exemple de fichiers LDIF :

```
dn: cn=John Doe,dc=example,dc=com
  objectClass: top
  objectClass: person
  objectClass: inetOrgPerson
  objectClass: organizationalPerson
  cn: John Doe
  givenName: John
  sn: Doe
  mail: john.doe@example.com
```

Dans cette entrée, nous retrouvons, outre le Distinguished Name (DN) de l'objet :

- 4 objectClass : inetOrgPerson, organizationPerson, person et top ;
- 4 attributs : cn, givenName, sn et mail.

Les objectClass permettent d'inclure des attributs obligatoires ou optionnels dans une entrée. Toujours dans notre exemple, c'est l'ajout de l'objectClass inetOrgPerson qui va permettre d'ajouter un attribut mail.

L'ensemble des objectClass et des attributs sont définis dans des schémas, qu'il conviendra d'ajouter en fonction du rôle du serveur. Par exemple, pour permettre l'authentification Samba depuis le serveur LDAP, il faudra ajouter le schéma samba.schema à la configuration du serveur.

Le format LDIF a des caractéristiques très importantes :

- les séries d'entrées sont séparées par des lignes vides ;
- la dernière ligne doit être vide, sinon la dernière entrée pourrait ne pas être prise en compte.

Les outils clients LDAP

Des outils en ligne de commande permettent l'administration du serveur en utilisant en entrée des fichiers LDIF ou la console.

- ldapadd : ajouter des entrées ;
- ldapdelete : supprimer des entrées ;
- ldapmodify : modifier des entrées ;
- ldappasswd : modifier un mot de passe ;
- ldapsearch : rechercher dans l'annuaire.

9.2. Installation du serveur

Prérequis à l'installation :

- disposer des droits root ou sudo ;
- disposer d'un dépot yum configuré ;
- avoir ouvert les ports 389 et 636 sur le parefeu local et sur les éléments actifs

Installation :

```
[root]# yum install openldap-servers openldap-clients
[root]# cp /usr/share/openldap-servers/DB_CONFIG.example /var/lib/ldap/DB_CONFIG
[root]# chown ldap:ldap /var/lib/ldap/DB_CONFIG
[root]# chkconfig slapd on
[root]# service slapd start
```

LDAP a besoin d'un fichier de configuration (/var/lib/ldap/DB_CONFIG) pour sa base de données. Le fichier fourni en exemple lors de l'installation convient parfaitement.

Le fichier /etc/openldap/ldap.conf contient la configuration pour les clients openldap comme ldapsearch, ldapadd, etc. Toutes les informations inscrites dans ce fichier allègeront d'autant les lignes de commande interactives, puisqu'il ne sera plus nécessaire de préciser les options positionnées ici.

Le fichier /etc/openldap/ldap.conf

```
#
# LDAP Defaults
#

#BASE dc=example,dc=com
#URI ldap://ldap.example.com ldaps://ldap.example.com:666

#SIZELIMIT 12
#TIMELIMIT 15
#DEREF     never

TLS_CACERTDIR /etc/openldap/certs
```

Le répertoire /etc/openldap/slapd.d/ contient les bases de données et le schéma :

Arborescence du service OpenLDAP

```
# /etc/openldap/slapd.d/
|-- cn=config
|   |-- cn=schema # les schémas disponibles
|   |   |-- cn={10}ppolicy.ldif
|   |   |-- cn={1}core.ldif
|   |   |-- cn={2}cosine.ldif
|   |   |-- cn={5}inetorgperson.ldif
|   |   |-- cn={8}nis.ldif
|   |   |-- cn={9}openldap.ldif
|   |-- cn=schema.ldif # le schéma du serveur
|   |-- olcDatabase={0}config.ldif
|   |-- olcDatabase={-1}frontend.ldif
|   |-- olcDatabase={1}monitor.ldif
|   |-- olcDatabase={2}bdb.ldif # la DIT principale au format BDB
|-- cn=config.ldif # configuration globale du serveur
```

Les bases de données du serveur LDAP ne doivent jamais être modifiées manuellement !!!

9.3. Configuration du serveur

Avant de pouvoir utiliser les outils en ligne de commande, il convient de configurer les options par

défaut :

```
BASE dc=etrs,dc=lan
URI ldap://localhost
```

En version TLS sécurisée, l'URI doit impérativement correspondre au FQDN renseigné dans le certificat !

Pour la suite du cours, nous retiendrons que :

- le dn de base est : dc=etrs,dc=lan ;
- l'administrateur LDAP est cn=admin,dc=etrs,dc=lan ;
- les utilisateurs sont stockés dans l'unité d'organisation : ou=users,dc=etrs,dc=lan.

Il est intéressant à ce stade de visualiser la configuration par défaut avec la commande slapcat :

```
[root]# slapcat -b cn=config | less
...
dn: olcDatabase={2}bdb,cn=config # base de données de l'annuaire
...
olcSuffix: dc=my-domain,dc=com # suffix par défaut
olcRootDN: cn=Manager,dc=my-domain,dc=com # administrateur par défaut
...
```

À noter que l'administrateur n'a pas de mot de passe (olcRootPW)

Le suffixe

Le suffixe représente la racine de l'organisation. C'est l'identité même de l'entreprise. Elle correspond habituellement au suffixe DNS.

Nous allons le changer avec la commande ldapmodify :

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:///
dn: olcDatabase={2}bdb,cn=config
changetype: modify
replace: olcSuffix
olcSuffix: dc=etrs,dc=lan
```

Le suffixe étant défini à l'installation, il faut le modifier !

Le et son mot de passe

L'entrée RootDN contient le DN de l'utilisateur autorisé à faire des modifications de l'annuaire.

Son mot de passe est défini par RootPW.

Nous allons le configurer avec la commande ldapmodify :

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:///
dn: olcDatabase={2}bdb,cn=config
changetype: modify
replace: olcRootDN
olcRootDN: cn=admin,dc=etrs,dc=lan
```

Le suffixe étant défini à l'installation, il faut le modifier !

Pour définir un mot de passe utilisable par openldap, il faut utiliser la commande slappasswd.

Ajouter le RootPW :

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:///
dn: olcDatabase={2}bdb,cn=config
changetype: modify
add: olcRootPW
olcRootPW: {SSHA}Eke0fnWgD90xZWPT/UivZEBjzBgC/Z+
```

Cette fois-ci, le RootPW n'ayant pas été défini à l'installation, il faudra l'ajouter!

Les trois commandes auraient pu être regroupées en une seule. Dans ce cas, il faut séparer chaque modification de l'objet par une ligne contenant un "-".

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:///
dn: olcDatabase={2}bdb,cn=config
changetype: modify
replace: olcSuffix
olcSuffix: dc=etrs.lan
-
replace: olcRootDN
olcRootDN: cn=admin,dc=etrs,dc=lan
-
add: olcRootPW
olcRootPW: {SSHA}Eke0fnWgD90xZWPT/UivZEBjzBgC/Z+
```

Connexion avec le RootDN

Un RootDN et son mot de passe ayant maintenant été définis dans la DIT dc=etrs,dc=lan, il est possible de les utiliser pour se connecter :

```
[root]# ldapmodify -x -D cn=admin,dc=etrs,dc=lan -W
```

Il n'est pas nécessaire de préciser ici le serveur à contacter (options -H ou -h), la commande ldapmodify utilisera les informations du fichier /etc/openldap/ldap.conf qui a été renseigné précédemment.

La commande

Exporter le contenu de l'annuaire au format LDIF.

Syntaxe de la commande slapcat

```
slapcat -b suffix
```

Exemple :

```
[root]# slapcat -b cn=config | less
...
dn: olcDatabase={2}bdb,cn=config
...
olcSuffix: dc=my-domain,dc=com
olcRootDN: cn=Manager,dc=my-domain,dc=com
...
```

Option	Description
-b	Détermine quelle base de données est exportée.

La commande

La commande ldapmodify permet de modifier le contenu de l'annuaire.

Authentification binding par SASL

Syntaxe de la commande ldapmodify avec authentification SASL

```
ldapmodify [-y SASLMecanisme] [-H host] [-v] [-f fichier.ldif]
```

Exemple :

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:/// -v -f modldap.ldif
```

Option	Description
-Y	Mécanisme SASL à utiliser pour l'authentification.
-v	Mode verbeux pour diagnostique.
-H	Spécifier un serveur. Le protocole **ldapi** permet une communication sécurisée via une socket UNIX (nécessaire pour utiliser SASL).

Authentification binding simple

Syntaxe de la commande ldapmodify avec authentification simple

```
ldapmodify [-x] [-D RootDN] [-W|-w pwd] [-H host] [-f fichier.ldif]
```

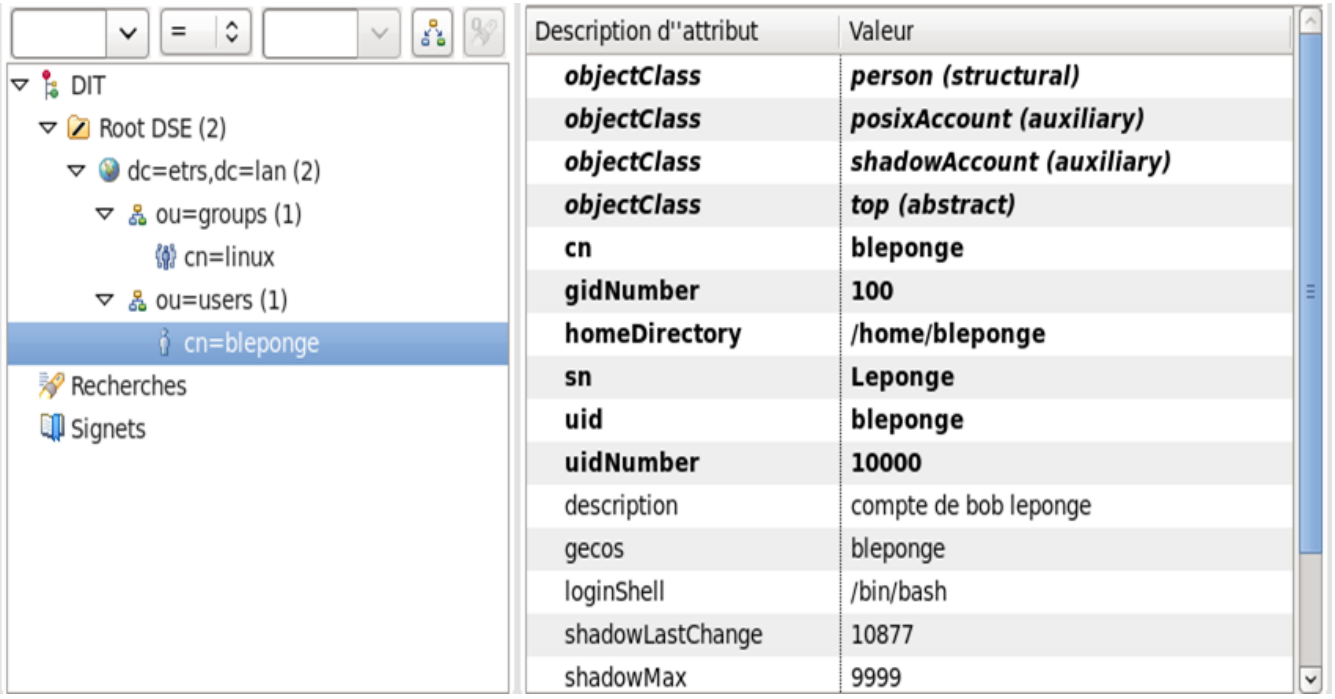
Exemple :

```
[root]# ldapmodify -x -D cn=admin,dc=etrs,dc=lan -W
```

Option	Description
-x	Utiliser l'authentification simple au lieu de SASL
-D	BindDN à utiliser pour la connexion à la base.
-W ou -w	Demander le mot de passe (interactif ou non).
-f	Lire les modifications à effectuer depuis un fichier

Il n'est pas nécessaire de préciser le serveur à contacter (options -H ou -h) si celui-ci est renseigné dans le fichier /etc/openldap/ldap.conf

Exemples



Il faut séparer les cas suivants :

- Ajouter/Supprimer un objet. Supprimer bleponge ou ajouter asaglisse.
- Modifier un objet en lui ajoutant, supprimant ou modifiant un attribut.

Ajouter un objet

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:///
dn:dndelobjetaajouter
changetype: add
...
```

Supprimer un objet

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:///
dn:dndelobjetasupprimer
changetype: delete
```

Modifier un objet

- Ajouter un attribut :

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:///
dn: dndelobjetamodifier
changetype: modify
add: nomdelatrribut
nomdelattribut: nouvellevaleur
```

- Supprimer un attribut

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:///
dn: dndelobjetamodifier
changetype: modify
delete: nomdelattribut
```

- Modifier un attribut

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:///
dn: dndelobjetamodifier
changetype: modify
replace: nomdelattribut
nomdelattribut: nouvellevaleur
```

La structure de la DIT

Les données de l'arbre de l'annuaire doivent être rangées dans des unités d'organisation (OU).

Les OU **users** et **groups** sont généralement utilisées.

Commencer par ajouter une entrée dans l'annuaire correspondant à l'organisation de l'entité :

```
[root]# ldapmodify -x -D cn=admin,dc=etrs,dc=lan -W
Enter LDAP Password:
dn: dc=etrs,dc=lan
changetype: add
objectClass: dcObject
objectClass: organization
dc: etrs
o: etrs
description: Serveur ETRS
```

Puis les deux OU concernées :

Le fichier /root/structure.ldif

```
dn: ou=users,dc=etrs,dc=lan
changetype: add
objectClass: top
objectClass: organizationalUnit
ou: users
description: Utilisateurs de l'ETRS

dn: ou=groups,dc=etrs,dc=lan
changetype: add
objectClass: top
objectClass: organizationalUnit
ou: groups
description: Groupes d'utilisateurs de l'ETRS
```

```
[root]# ldapmodify -x -D cn=admin,dc=etrs,dc=lan -W -f /root/structure.ldif
```

Activation des logs

Dans certains cas, il sera intéressant d'activer la journalisation dans la base cn=config.

Celle-ci étant très verbeuse, elle sera activée ou désactivée au besoin.

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:///
dn: cn=config
changeType: modify
replace: olcLogLevel
olcLogLevel: stats
```

Il faut paramétrer le service syslog pour qu'il accepte les logs (niveau local4).

Le fichier /etc/rsyslog.conf

```
local4.* /var/log/slapd.log
```

sans oublier de redémarrer le démon syslog :

```
[root]# service rsyslogd restart
```

Le logLevel 4 permet de journaliser les requêtes effectuées sur la base.

Activation du

Avant de pouvoir configurer le TLS sous OpenLDAP, il convient de disposer du certificat et de la clef pour le serveur ainsi que le certificat de l'autorité de certification, qui est indispensable au bon fonctionnement d'OpenLDAP.

Pour créer ces certificats, il est possible d'utiliser easy-rsa, qui sera abordé dans la quatrième partie du document.

Une autre méthode est d'utiliser l'outil certtool du paquet gnutls-utils.

Si l'accès au serveur LDAP se fait via le FQDN ldap.etrs.lan, il faudra impérativement créer le certificat qui répondra à ce nom. Il ne sera plus possible par la suite de se connecter en LDAPS ou en starttls via l'adresse de loopback localhost.

Création des certificats avec certtools

Installer le paquet gnutls-utils :

```
[root]# yum install gnutls-utils
```

Dans le cas d'un certificat auto-signé, il faut dans un premier temps créer une clef privée pour l'autorité de certification :

```
[root]# certtool --generate-privkey --outfile /etc/pki/CA/private/ca-key.key
```

et décliner cette clef privée en certificat public.

```
[root]# certtool --generate-self-signed --load-privkey /etc/pki/CA/private/ca-key.key
--outfile /etc/pki/CA/certs/ca-cert.pem
```

Il faut ensuite générer un certificat privé pour le serveur (ldap.etrs.lan par exemple)

```
[root]# certtool --generate-privkey --outfile /etc/pki/tls/private/ldap.key
```

Puis son certificat public signé par la clef privée de l'autorité de certification créée ci-dessus :

```
[root]# certtool --generate-certificate --load-privkey /etc/pki/tls/private/ldap.key
--outfile /etc/pki/tls/certs/ldap.pem --load-ca-certificate /etc/pki/CA/certs/ca-
cert.pem --load-ca-privkey /etc/pki/CA/private/ca-key.key
```

Prise en compte des certificats

Le fichier /root/tls.ldif

```
dn: cn=config
changetype: modify
replace: olcTLSCertificateFile
olcTLSCertificateFile: /etc/pki/certs/ldap.pem
-
replace: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/pki/private/ldap.key
-
replace: olcTLSCACertificateFile
olcTLSCACertificateFile: /etc/pki/CA/certs/ca-cert.pem
```

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:/// -f /root/tls.ldif
```

La chaîne de connexion du fichier /etc/openldap/ldap.conf doit également être mise à jour :

Le fichier /etc/openldap/ldap.conf

```
BASE dc=etrs,dc=lan
URI ldaps://ldap.etrs.lan

TLS_CACERTDIR /etc/openldap/certs
TLS_REQCERT try
```

La commande **cacertdir_rehash** permet de créer un lien symbolique vers le certificat de la CA dont le nom correspond au hash de ce certificat. Ceci est nécessaire au fonctionnement d'openLDAP en TLS !

```
[root]# cacertdir_rehash /etc/pki/CA/certs
[root]# ls -l /etc/pki/CA/certs
-rw-r--r--. 1 root root 1281 4 déc. 10:52 ca-cert.pem
lrwxrwxrwx. 1 root root 11   4 déc. 10:54 ce6a8cab.0 -> ca-cert.pem
```

Par défaut, le service slapd n'écoute pas sur le port 636 (ldaps) et il faut privilégier le startTLS sur le port 389. Pour activer le ldaps :

Le fichier /etc/sysconfig/ldap

```
SLAPD_LDAPS=yes
```

sans oublier de relancer le serveur :

```
[root]# service slapd restart
```

Tester la connexion

La commande openssl permet de tester la connexion uniquement sur le port 636 :

```
[root]# openssl s_client -connect ldap.etrs.lan:636 -showcerts
```

Le certificat de la

De nombreuses applications auront besoin du certificat de la CA.

Il est recommandé de le mettre à disposition des utilisateurs sur le serveur web du serveur LDAP :

```
[root]# cp /etc/pki/CA/certs/ca-cert.pem /var/www/html/
```

Le certificat est ainsi accessible via l'adresse http://ldap.etrs.lan/cacert.pem.

Configuration du PAM

PAM peut être configuré pour utiliser le service openldap avec la commande authconfig :

```
[root]# yum install nss-pam-ldapd
```

```
[root]# authconfig --enableldap --enableldapauth --ldapserver=ldap://ldap.etrs.lan
--ldapbasedn="ou=users,dc=etrs,dc=lan" --enableldaptls --ldaploadcacert
=http://ldap.etrs.lan/ca-cert.pem --enablemkhomedir --update
```

Création des utilisateurs

Création du fichier pour l'utilisateur :

```
vim /root/antoine.ldif

dn: cn=alemorvan,ou=users,dc=etrs,dc=lan
objectClass: top
objectClass: person
objectClass: posixAccount
objectClass: shadowAccount
cn: alemorvan
sn: Le Morvan
uid: alemorvan
uidNumber: 10000
gidNumber: 500
homeDirectory: /home/alemorvan
loginShell: /bin/bash
userPassword: {crypt}password
gecos: alemorvan
shadowWarning: 7
```

```
ldapadd -x -D cn=admin,dc=etrs,dc=lan -W -f /root/antoine.ldif
```

Chapitre 10. Serveur proxy SQUID

10.1. Principes de fonctionnement

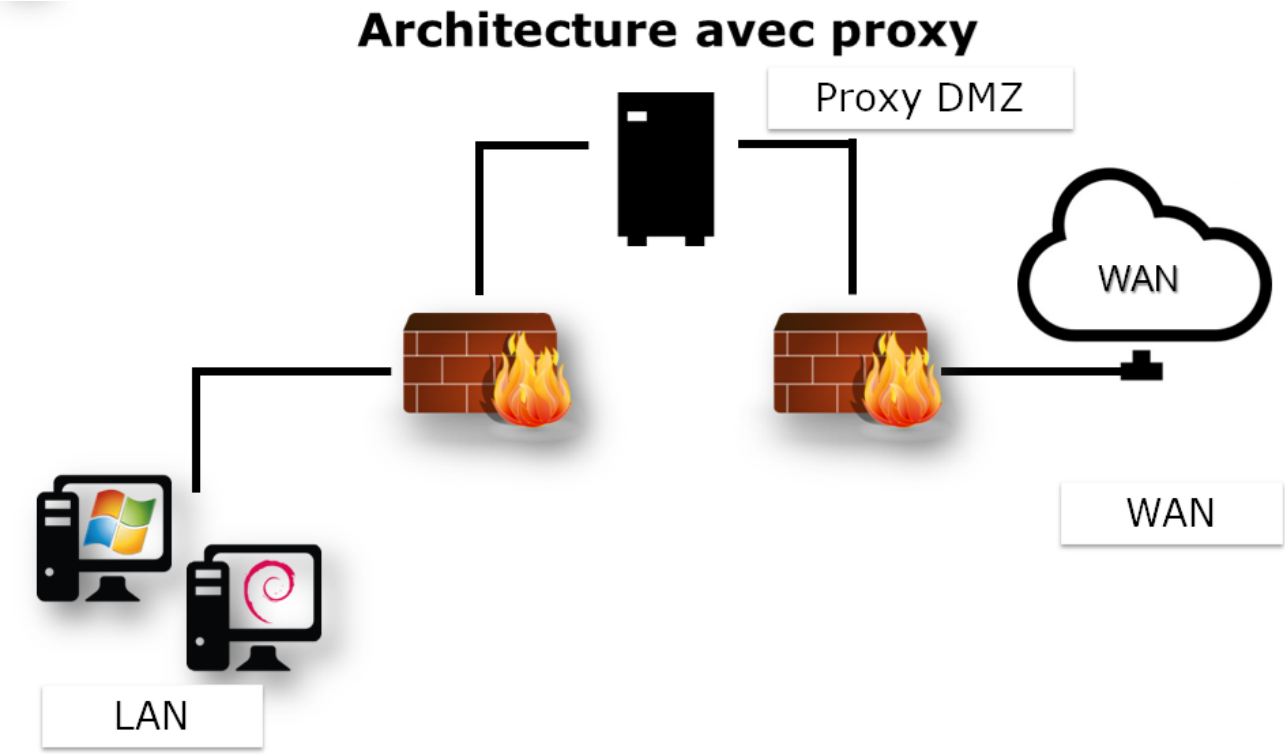
La mise en place d'un serveur proxy nécessite de choisir entre deux types d'architectures :

- Une architecture proxy standard, qui nécessitera une configuration spécifique de chaque client et de leurs navigateurs internet,
- Une architecture dite proxy captif, qui nécessitera l'interception des trames émises par le client et de les réécrire vers le serveur proxy.

Dans un cas comme dans l'autre, il y a rupture au niveau du réseau.

Un client ne peut physiquement plus s'adresser directement à un serveur distant, sans passer par un mandataire, appelé plus couramment serveur proxy.

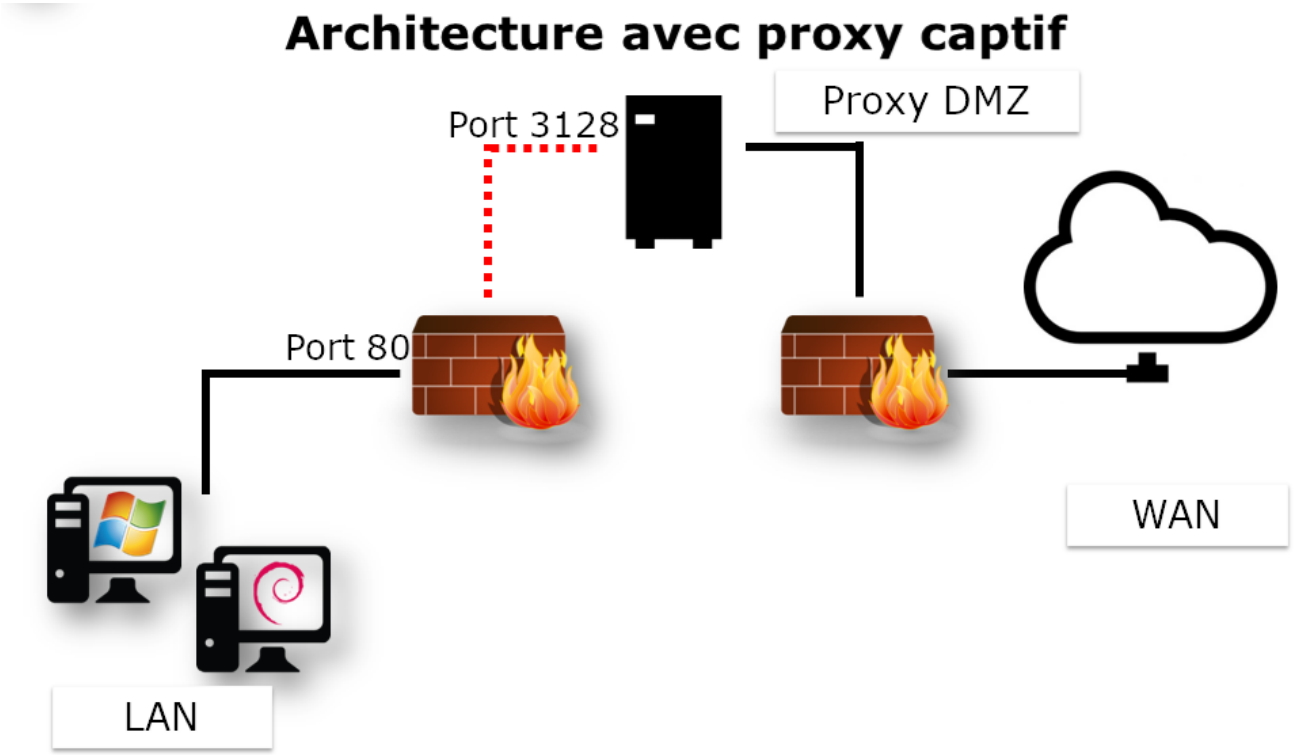
Le poste client est protégé par deux pare-feux et ne communique jamais directement vers le réseau extérieur.



Cette architecture nécessite la configuration du navigateur sur le poste client.

Dans le cas du proxy captif, il n'y a pas de nécessité de configurer l'ensemble des postes clients.

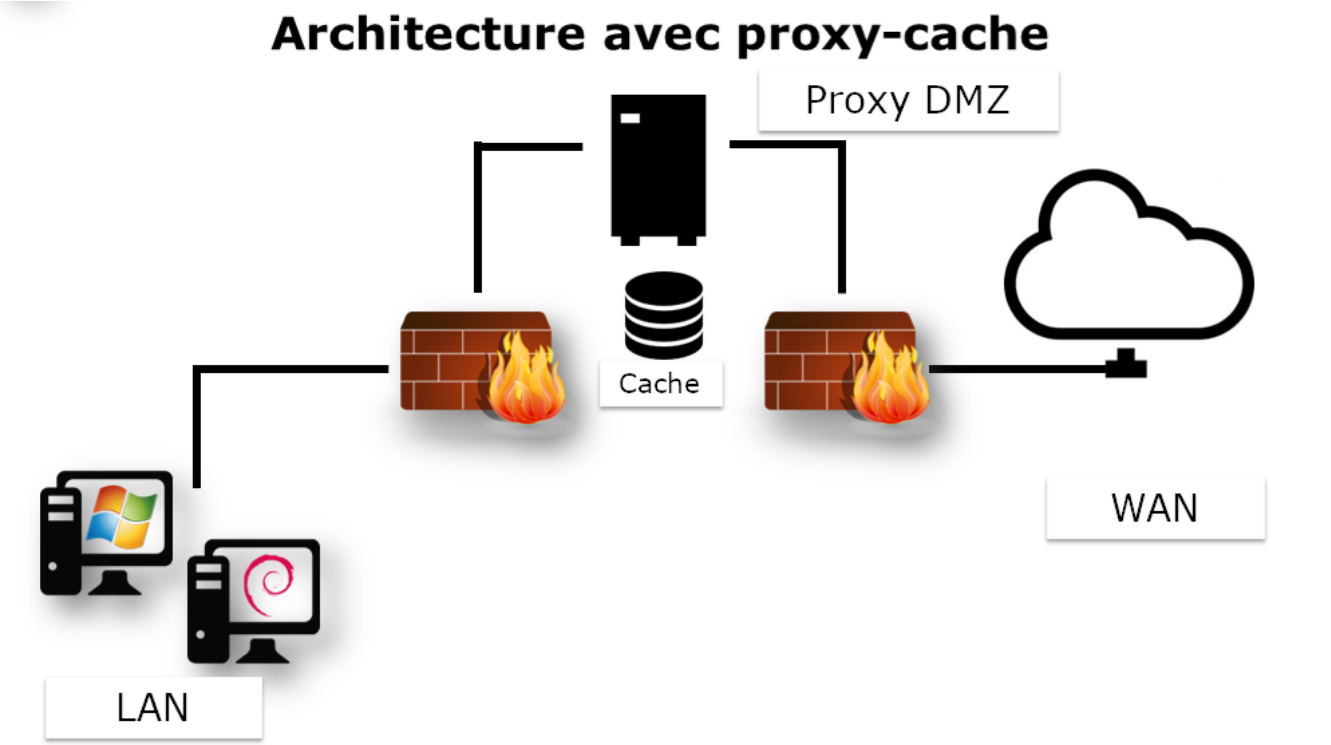
La configuration se passe au niveau de la passerelle, qui reçoit les demandes des clients, et qui va, de manière totalement transparente, réécrire les trames pour les envoyer vers le proxy.



Cette architecture nécessite une configuration spécifique sur le routeur.

Dans le cas de l'architecture proxy standard ou du proxy captif, l'un des premiers intérêts de ce genre de service est bien évidemment de faire office de cache.

Ainsi, un fichier qui aura été téléchargé une première fois depuis le WAN (et donc potentiellement une liaison plus lente que le LAN), sera conservé en mémoire par le proxy-cache pour être resservi au profit des clients suivants. Ainsi, la bande passante de la liaison lente est optimisée.



Comme nous le verrons dans la suite de ce chapitre, ce n'est bien évidemment pas la seule utilité d'un proxy.

Un proxy pourra être déployé pour :

- Interdire l'accès à certaines ressources en fonction de différents paramètres,
- Mettre en place une authentification et un suivi des activités sur internet des clients,
- Mettre en place une hiérarchie de cache distribués,
- Masquer l'architecture du LAN d'un point de vue WAN (combien y a-t-il de clients sur le LAN ?).

Les intérêts sont multiples :

- Anonymat sur Internet ;
- Authentification ;
- Journaliser les activités des clients ;
- Filtrage ;
- Limiter les accès ;
- Optimisation de la bande passante ;
- Sécurité.

Mettre en place l'authentification bloque une grande partie des effets malveillants des virus sur le LAN.

Le service proxy devient un service critique nécessitant une haute disponibilité.

Durant l'exploitation d'un serveur Proxy Squid, l'administrateur est amené à exploiter les logs. Il est donc primordiale de connaître les principaux codes réponses HTTP.

Table 94. Les codes réponses HTTP

Code	Catégories
1XX	Info
2XX	Succès
3XX	Redirection
4XX	Erreur de requête client
5XX	Erreur sur le serveur

Exemples :

- 200 : ok
- 301 : Moved Permanently
- 302 : Moved Temporatly
- 304 : Not modified
- 400 : Bad request
- 401 : Unauthorized
- 404 : Not found

10.2. Le serveur SQUID

Squid prend en charge les protocoles http et ftp.

Les intérêts d'installer une solution basée sur le serveur Squid :

- Les solutions matérielles sont coûteuses ;
- Il est développé depuis 1996 ;
- Il est publié sous licence GNU/GPL.

Dimensionnement

- Prévoir des solutions de haute disponibilité ;
- Privilégier des disques durs rapides pour le cache ;
- Mémoire vive et CPU correctement dimensionnés.

Il faut prévoir 14Mo de RAM par Go de cache sur le disque.

Installation

L'installation du serveur Squid se fait avec le paquet squid.

```
yum install squid
chkconfig squid on
```

Attention à ne pas démarrer le service tant que le cache n'a pas été initialisé !

Arborescence et fichiers du serveur Squid

Le fichier de configuration unique est le fichier **/etc/squid/squid.conf**.

Les logs du service (arrêt et relance) sont enregistré dans le fichier **/var/log/squid.cache.log** tandis que les requêtes des clients **/var/log/squid/access.log**. Les fichiers de cache seront par défaut stockés dans **/var/spool/squid/**.

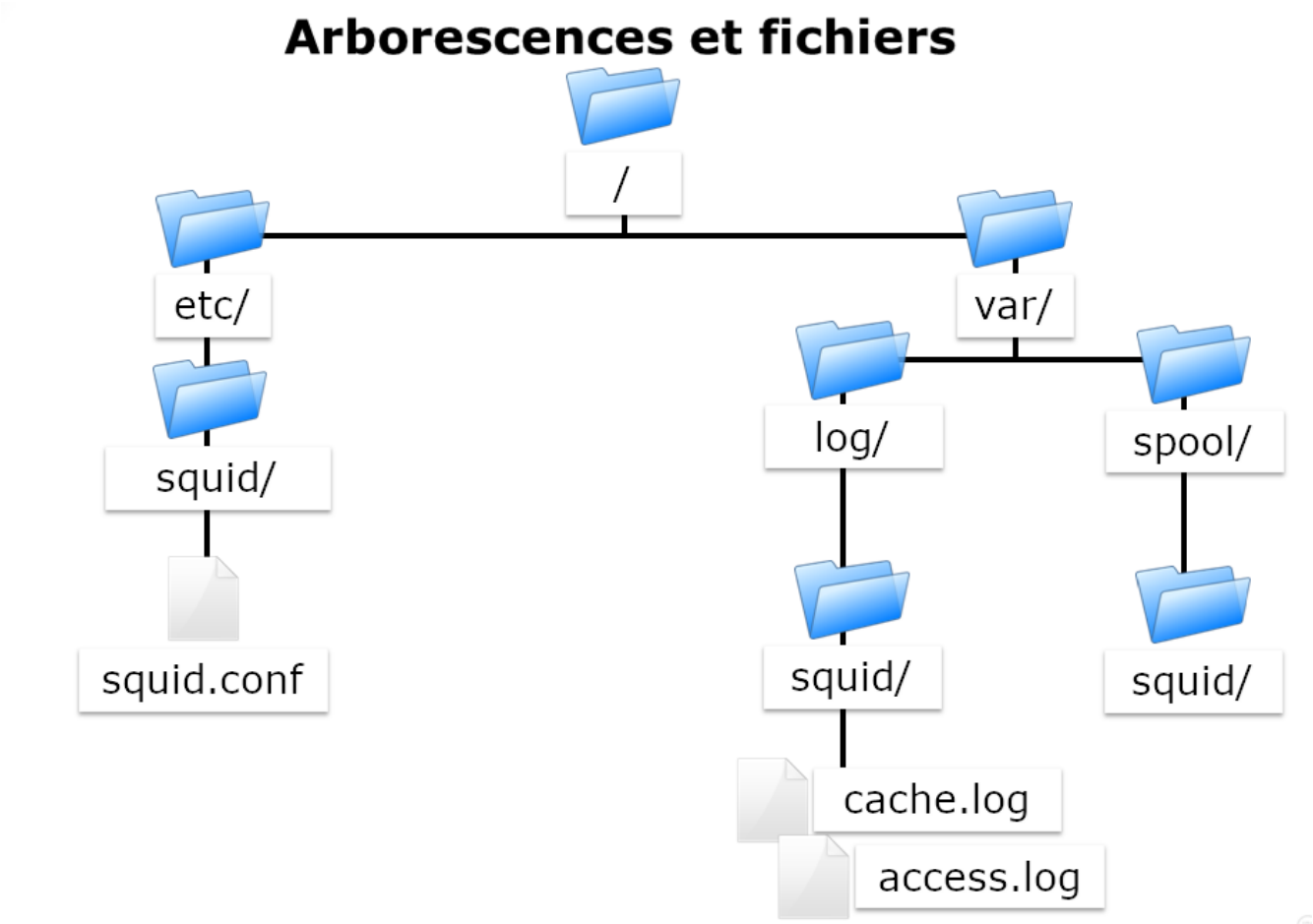


Figure 61. Arborescence et fichiers du serveur Squid

La commande squid

Commande squid permet de contrôler le serveur squid.

Syntaxe de la commande squid

```
squid [-z|-s|-k parse|-k rotate]
```

Option	Description
-z	Initialise les répertoires du cache
-s	Active la journalisation syslog
-k parse	Test le fichier de configuration
-k rotate	Effectue une rotation des logs

Journaliser les requêtes clientes peut rapidement entraîner le stockage des volumes conséquents de données.

Il est opportun de régulièrement créer un nouveau fichier de log et d'archiver l'ancien dans un format compressé.

Cette action peut être effectuée manuellement avec l'option **–k rotate** de la commande squid ou via le service Linux dédié **Logrotate**.

10.3. Configuration basique

La configuration de Squid se fait dans le fichier de configuration **/etc/squid/squid.conf**.

- Numéro de port du proxy (port d'écoute)

Syntaxe de la directive http_port

```
http_port num_port
```

Par défaut, le numéro de port est fixé à 3128 mais il est fréquemment changé à 8080. Il faudra penser à ouvrir le port correspondant du pare-feu !

Par exemple :

```
http_port 8080
```

Au redémarrage du service, le serveur Squid se mettra en écoute sur le port défini par la directive http_port.

- Réservation de la mémoire vive

Syntaxe de la directive cache_mem

```
cache_mem taille KB|taille MB|taille GB
```

Par exemple :

```
cache_mem 1 GB
```

Bonne pratique : 1/3 du total de la mémoire vive allouée

- Protocole de Cache Internet (ICP)

Le protocole ICP (Internet Cache Protocol) permet aux serveurs Squid voisins de s'échanger des requêtes. Il est courant de proposer une hiérarchie de proxy qui se partagent leurs bases d'informations.

La directive icp_port permet de définir le numéro de port sur lequel Squid envoie et reçoit les requêtes ICP des serveurs Squid voisins.

Par exemple :

```
icp_port 3130
```

Positionner à 0 pour le désactiver.

- Utilisateur FTP anonyme

La directive ftp_user permet d'associer un utilisateur FTP aux connexions FTP anonymes. L'utilisateur dit être une adresse de messagerie valide.

```
ftp_user bob@formatux.lan
```

- Mettre en place des Access Control List

Syntaxe des ACL

```
acl nom type argument
http_access allow|deny nomacl
```

Exemple :

```
acl REPAS time 12:00-14:00
http_access deny REPAS
```

Les ACL sont étudiées plus en détail dans la partie "Configuration avancée".

- Taille maximum d'un objet en cache

Syntaxe de la directive maximum_object_size

```
maximum_object_size size
```

Exemple :

```
maximum_object_size 32 MB
```

Si la taille de l'objet est supérieur à la limite maximum_object_size, l'objet n'est pas conservé en cache.

- Nom du serveur proxy

Syntaxe de la directive visible_hostname

```
visible_hostname nom
```

Exemple :

```
visible_hostname proxysquid
```

La valeur fournie peut être différente du nom d'hôte.

- Définir un cache pour squid

```
cache_ufs format chemin taille nbDossierNiv1 nbDossierNiv2
```

Plusieurs caches peuvent être définis sur différents systèmes de fichiers pour optimiser les temps d'accès.

Exemple :

```
cache_dir ufs /var/spool/squid/ 100 16 256
```

Option	Description
ufs	Unix File System
100	Taille en méga
16	16 dossiers de premier niveau
256	256 dossiers de second niveau

Au premier lancement du service, il faut initialiser le dossier de cache :

```
[root]# squid -z
[root]# service squid start
```

10.4. Configurations avancées

Les Access Control List (ACL)

Syntaxe de la directive http_access

```
http_access allow|deny [!]nom_acl
```

Exemple :

```
http_access allow REPAS
http_access deny !REPAS
```

L'ACL !nom_acl est le contraire de l'ACL nom_acl.

Syntaxe de la directive acl

```
acl nom type argument
```

L'ordre des ACL est cumulatif. Plusieurs ACL de même nom représentent une seule ACL.

Exemples :

- Autoriser à l'heure des repas :

```
acl REPAS time 12:00-14:00
http_access allow REPAS
```

- Interdire les vidéos :

```
acl VIDEOS rep_mime_type video/mpeg
acl VIDEOS rep_mime_type video/avi
http_access deny VIDEOS
```

- Gestion des adresses IP :

```
acl XXX src 192.168.0.0/255.255.255.0
acl XXX dst 10.10.10.1
```

- Gestion des FQDN :

```
acl XXX srcdomain .formatux.lan
acl XXX dstdomain .linux.org
```

- Gestion des ports :

```
acl XXX port 80 21
```

- Gestion des protocoles :

```
acl XXX proto HTTP FTP
```

Les algorithmes de cache

Il existe différents algorithmes de cache qui disposent de caractéristiques différentes :

- LRU - **Least Recently Used** : supprime les objets les plus anciens de la mémoire vive.
- LRU-THOLD : copie en fonction de sa taille un objet dans le cache.
- MRU : **Most Recently Used** : les données les moins demandées sont supprimées.
- GDSF : **Greedy Dual Size Frequency** : supprime en fonction de la taille et du temps d'accès d'origine. Les plus petits sont conservés.
- LFUDA : **Least Frequently Used With Dynamic Aging** : idem que GDSF sans notion de taille. Utile pour les caches avec des fichiers de grande taille.

10.5. Authentification des clients

Squid s'appuie sur des programmes externes pour gérer l'authentification. Il peut ainsi s'appuyer sur un simple fichier plat type htpasswd ou sur un service LDAP, SMB, PAM, etc.

L'authentification peut être une nécessité juridique : pensez à faire signer une charte d'usage à vos

utilisateurs !

10.6. Outils

La commande squidclient

La commande squidclient permet de tester une requête vers le serveur squid.

Syntaxe de la commande squidclient

```
squidclient [–s] [-h cible] [-p port] url
```

Exemple :

```
squidclient –s –h localhost –p 8080 http://localhost/
```

Option	Description
-s	Mode silencieux (n'affiche rien dans la console)
-h	Définir un proxy cible
-p	Port d'écoute (par défaut 3128)
-r	Forcer le serveur à recharger l'objet

Analyser les logs

Les enregistrements du journal de Squid peuvent être suivi avec la commande :

```
tail –f /var/log/squid/access.log
```

- Décomposition d'une ligne de log

Option	Description
Date	Horodatage du log
Tps reponse	Temps de réponse pour la requête
@ client	Adresse IP du client
Code status	Code HTTP de la réponse
Taille	Taille du transfert
Méthode	Méthode HTTP (Put / Get / Post / etc.)
URL	URL de la requête
Peer Code	Code de réponse inter-proxy

Option	Description
Type fichier	Type mime de la cible de la requête

La commande sarg

La commande sarg (**Squid Analysis Report Generator**) permet de générer un rapport au format HTML.

Syntaxe de la commande sarg

```
sarg -x
```

- Installer sarg :

```
yum install httpd
yum install sarg
```

- Configurer sarg :

```
vim /etc/sarg/sarg.conf
access_log /var/log/squid/access.log
```

Le rapport est généré sous `/var/www/html/squid_reports/`.

SquidGuard

SquidGuard permet de filtrer les URLs à partir de blacklists (éventuellement disponibles sur Internet).

Sa mise en œuvre dépasse toutefois le cadre de ce support.

Chapitre 11. Serveur de log Syslog

Le système génère des logs qu'il faut surveiller pour la sécurité du système ou pour réagir avant la panne.

Sous Linux, c'est le rôle du protocole Syslog.

11.1. Généralités

Syslog est le protocole de journalisation standard sous Linux. Syslog gère le journal d'événement Linux, que ce soit pour le noyau Linux ou pour les services hébergés sur la station.

- Les logs peuvent être archivés localement (dans ce cas il faut prévoir une rotation des logs).
- Syslog peut également fonctionner en local en mode client/serveur. Syslog utilise le port 514 en UDP ou TCP pour sa communication réseau.

Exemple de fichier /var/log/messages :

```
Nov 23 08:30:00 centos6 dhcp service[warning] 110 message
```

Sous CentOS 6, c'est le logiciel rsyslog qui est utilisé pour gérer les logs du système. A noter que la syntaxe rsyslog est compatible avec les clients syslog standard (syslog, syslog-ng), mais l'inverse n'est pas vrai.

Un journal au format syslog comporte dans l'ordre les informations suivantes :

- la date à laquelle a été émis le log,
- le nom de l'équipement ayant généré le log (hostname),
- une information sur le processus qui a déclenché cette émission,
- le niveau de priorité du log,
- un identifiant du processus ayant généré le log
- le corps de message.

Certaines de ces informations sont optionnelles.

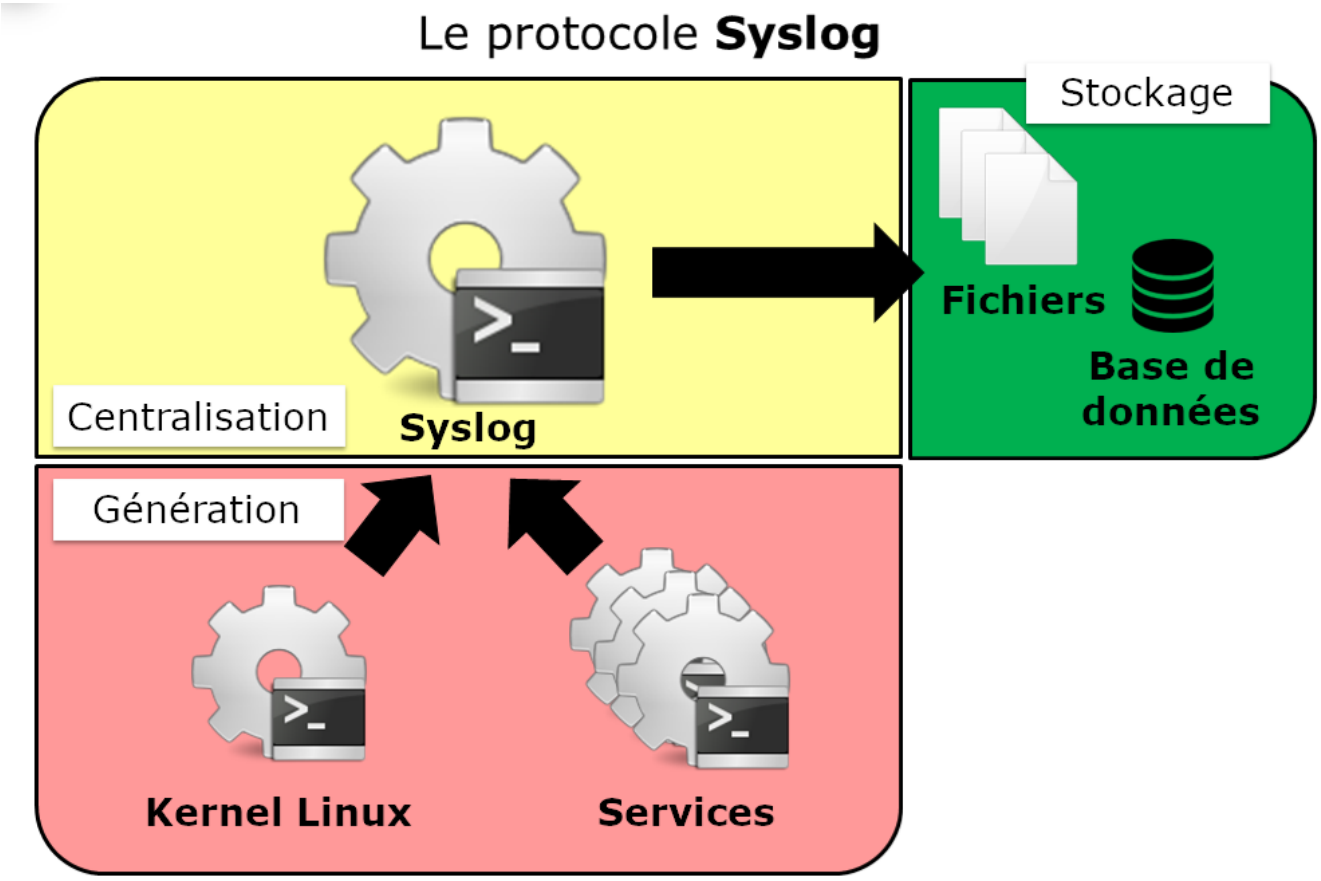


Figure 62. Fonctionnement du protocole Syslog

Le serveur Syslog centralise les messages du kernel Linux ou des services dans des fichiers. Des modules existent pour rediriger les logs vers une base de données.

En mode client/serveur, les clients envoient leurs logs vers un serveur syslog sur le port 514. Ce serveur peut ensuite stocker les logs de ses clients vers un serveur de base de données.

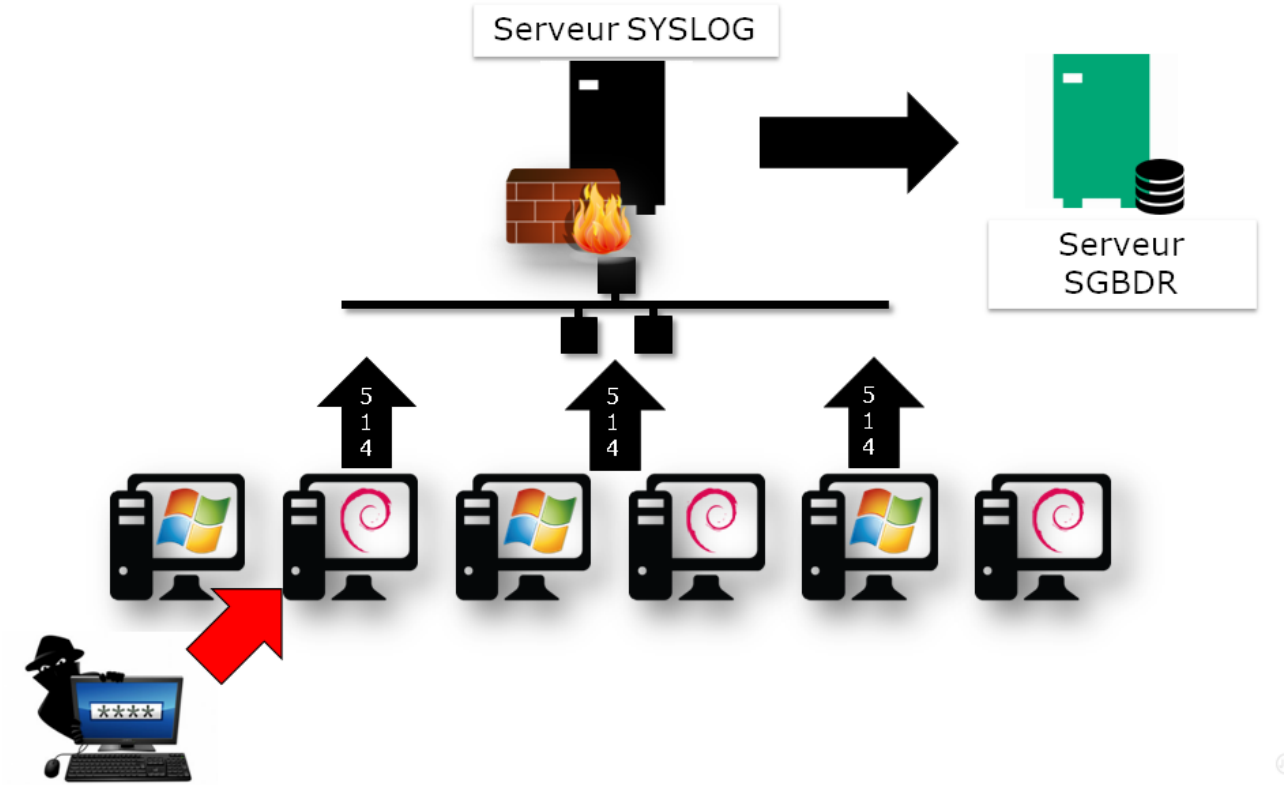


Figure 63. Mise en réseau du protocole Syslog

Ainsi, un attaquant ne peut pas effacer ces traces qui sont déportées sur un serveur distant.

Les catégories de messages

Les messages sont orientés selon leur origine et leur gravité (origine.gravité).

Table 95. Origine des messages Syslog

Code	Mot-clé	Description
0	kern	kernel messages
1	user	user-level messages
2	mail	mail system
3	daemon	system daemons
4	auth	security/authorization messages
5	syslog	messages generated internally by syslogd
6	lpr	line printer subsystem
7	news	network news subsystem
8	uucp	UUCP subsystem
9		clock daemon
10	authpriv	security/authorization messages

Code	Mot-clé	Description
11	ftp	FTP daemon
12	-	NTP subsystem
13	-	log audit
14	-	log alert
15	cron	clock daemon
16	local0	local use 0 (local0)
17	local1	local use 1 (local1)
18	local2	local use 2 (local2)
19	local3	local use 3 (local3)
20	local4	local use 4 (local4)
21	local5	local use 5 (local5)
22	local6	local use 6 (local6)
23	local7	local use 7 (local7)

Table 96. Gravité des messages syslog

Code	Gravité	Mot-clé	Description
0	Emergency	emerg (panic)	Système inutilisable.
1	Alert	alert	Une intervention immédiate est nécessaire.
2	Critical	crit	Erreur critique pour le système.
3	Error	err (error)	Erreur de fonctionnement.
4	Warning	warn (warning)	Avertissement (une erreur peut intervenir si aucune action n'est prise).
5	Notice	notice	Événement normal méritant d'être signalé.
6	Informational	info	Pour information.
7	Debugging	debug	Message de mise au point.

11.2. Client Syslog

La configuration du client et du serveur rsyslog est centralisée dans le fichier **/etc/rsyslog.conf**.

Après toute modification il faut redémarrer le service :

```
service rsyslog restart
```

La commande logger génère une ligne de log.

Syntaxe de la commande logger

```
logger texte
```

Exemple :

```
logger "====> Marqueur"
```

Fichier /var/log/messages après exécution de la commande logger

```
Nov 23 08:30:00 centos6 stagiaire ====> Marqueur
```

Il est possible de rediriger les logs du client vers le serveur :

Modification du fichier /etc/rsyslog.conf pour envoyer les logs vers le réseau

```
*.*     @IPServeur:514
```

```
service rsyslog restart
logger test
```

Le message test est envoyé vers le serveur.

@IPServeur = UDP @IPServeur = TCP

Pour différentier une redirection en TCP d'une redirection en UDP, il faudra doubler l'arobase présent devant l'adresse IP du serveur.

Par exemple :

```
mail.err* @@172.16.96.203
```

Après avoir ajouté cette ligne, le service syslog enverra les logs de la catégorie mail d'un niveau de gravité supérieur à erreur vers le serveur syslog 172.16.96.203 en TCP.

La commande logwatch

La commande logwatch effectue une synthèse journalière des logs et l'envoie par message.

Installation :

```
yum install logwatch
```

LogWatch analyse pour vous quotidiennement les logs pour en extraire les informations du jour, les trie et vous envoie une synthèse quotidienne.

Les logs des services étant généralement très copieux, un outil tel Logwatch (couplé avec la redirection des mails) est nécessaire pour rester informé en un seul coup d'œil.

Voici un exemple de rapport :

```
#################### Logwatch 7.3.6 (05/19/07) ####################
       Processing Initiated: Fri Oct 23 10:10:04 2015
       Date Range Processed: yesterday
                             ( 2015-Oct-22 )
                             Period is day.
     Detail Level of Output: 0
             Type of Output: unformatted
          Logfiles for Host: srv-instructeurs.formatux.lan
 ##################################################################

---------- Selinux Audit Begin ------------


---------- Selinux Audit End ------------


---------- Automount Begin ------------


---------- Automount End ------------


---------- Cron Begin ------------


---------- Cron End ------------


---------- httpd Begin ------------


Requests with error response codes
   403 Forbidden
      /: 1 Time(s)
   404 Not Found
      /favicon.ico: 2 Time(s)

---------- httpd End ------------
```

```
---------------------- Init Begin -----------------------


---------------------- Init End -----------------------


--------------------- Named Begin -----------------------


Received control channel commands
   reload: 8 Time(s)
   stop: 7 Time(s)

---------------------- Named End -----------------------


--------------------- pam_unix Begin -----------------------

su-l:
   Authentication Failures:
   Sessions Opened:
      pupitre -> root: 1 Time(s)

sudo:
   Authentication Failures:


---------------------- pam_unix End -----------------------


--------------------- Postfix Begin -----------------------

   3.957K  Bytes accepted                              4,052
   3.957K  Bytes delivered                             4,052
========   ==============================================

       4   Accepted                                  100.00%
--------   ----------------------------------------------
       4   Total                                     100.00%
========   ==============================================

       4   Removed from queue
       2   Sent via SMTP
       2   Forwarded

       6   Postfix start
       6   Postfix stop
       1   Postfix waiting to terminate
```

```
---------------------- Postfix End -------------------------


--------------------- Connections (secure-log) Begin ------------------------

New Users:
   postgres (26)

New Groups:
   postgres (26)


   groupadd: group added to /etc/group: name=postgres, GID=26: 1 Time(s)
   groupadd: group added to /etc/gshadow: name=postgres: 1 Time(s)
   webmin: Successful login as pupitre from 172.16.96.232: 1 Time(s)

---------------------- Connections (secure-log) End -------------------------


---------------------- SSHD Begin ------------------------


---------------------- SSHD End -------------------------


---------------------- Sudo (secure-log) Begin ------------------------



---------------------- Sudo (secure-log) End -------------------------


---------------------- yum Begin ------------------------


Packages Installed:
   postgresql-libs-8.4.20-3.el6_6.x86_64
   postgresql-server-8.4.20-3.el6_6.x86_64
   postgresql-8.4.20-3.el6_6.x86_64
   1:mod_ssl-2.2.15-47.el6.centos.x86_64
   1:net-snmp-libs-5.5-54.el6_7.1.x86_64
   policycoreutils-python-2.0.83-24.el6.x86_64

---------------------- yum End -------------------------


--------------------- Disk Space Begin ------------------------
```

```
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/vg_root-lv_root
                       27G  3.7G   22G  15% /
/dev/sda1             485M   34M  426M   8% /boot
/dev/sdb1             488M  154M  310M  34% /BoiteAOutils


---------------------- Disk Space End -------------------------


######################## Logwatch End #########################
```

11.3. Serveur Syslog

L'architecture de rsyslog est modulaire. Pour activer son mode serveur, il faut charger le module UDP ou TCP et le mettre en écoute sur le port 514 sans oublier de relancer le service.

Activer les serveurs UDP et TCP rsyslog

```
vim /etc/rsyslog.conf
$ModLoad imudp
$UDPServerRun 514

$ModLoad imtcp
$InputTCPServerRun 514
```

```
service rsyslog restart

netstat -tapn | grep 514
udp  0  0  0.0.0.0:514  0.0.0.0:*  LISTEN 3172/rsyslog
```

Stocker les logs dans des fichiers différenciés

Les modifications a apporter au fichier /etc/rsyslog.conf sont les suivantes :

```
## Rules
$template syslog,"/var/log/%fromhost%.log"

mail.* ?syslog
```

Explications :

- **$template** : définir un template qui s'appelera **syslog**
- la variable **%fromhost%** contient le nom du client à l'origine du message

- **mail.*** correspond à tout les messages d'origine mail qui seront redirigés vers le template que nous avons appelé syslog (?syslog)

11.4. Stockage en base de données

Il est particulièrement intéressant de stocker les enregistrements syslog en base de données. Il est ensuite possible de visualiser les logs dans des interfaces web spécialisées (ici LogAnalyzer) :

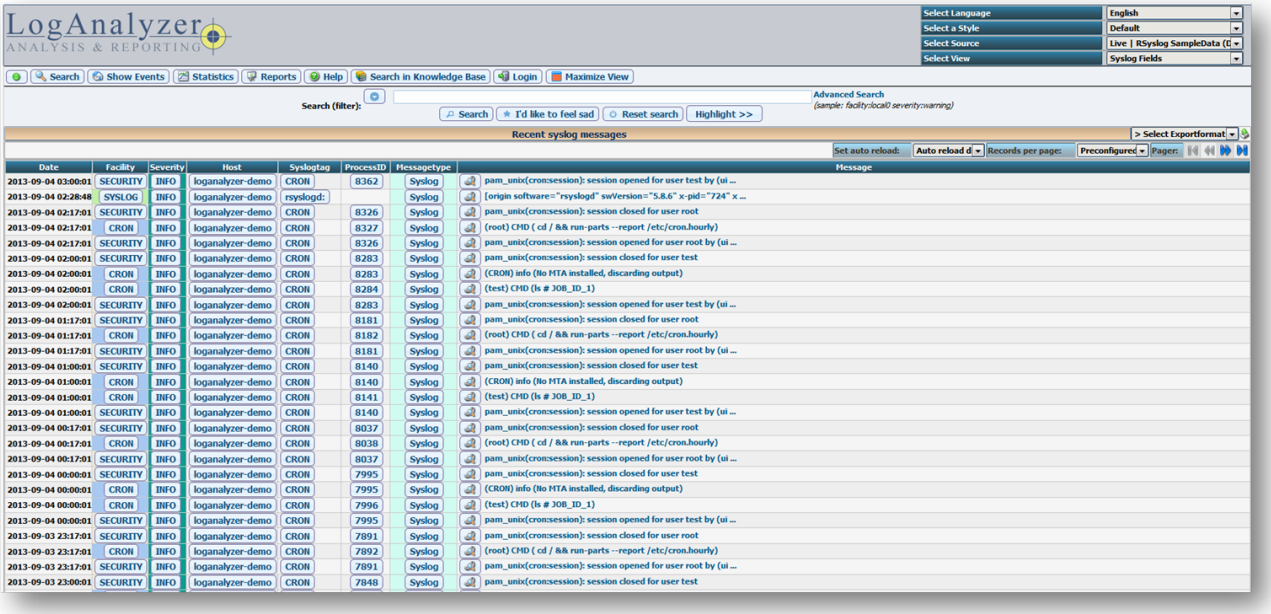


Figure 64. Interface du logiciel LogAnalyzer

Installer le module MySQL :

```
yum install rsyslog-mysql
```

Configurer le module dans /etc/rsyslog.conf :

```
$ModLoad MySQL
*.* > @IPServeur,base,USERMYSQL,PWDMYSQL
```

La création de la base MySQL sort du cadre de ce support.

```
service rsyslog restart
```

Chapitre 12. Serveur web Nginx

Nginx est un serveur web **HTTP libre sous licence BSD**. Son développement commence en Russie en 2002 par Igor Sysoev. Nginx dispose, en plus des fonctionnalités standards d'un serveur web, celles de **proxy inverse** (Reverse Proxy) pour le protocole **HTTP** mais aussi de **proxy** pour les protocoles de messageries **POP et IMAP**.

Le développement du serveur nginx est une réponse au problème **C10K** : supporter 10 000 connexions concurrentes (chose courante sur le web moderne) est un vrai challenge pour des serveurs web.

Un support commercial est possible par Nginx Inc.

12.1. Généralités

L'architecture interne du serveur permet des **performances très élevées** avec une **faible consommation de charge mémoire** en comparaison avec ce que peut faire le serveur web Apache notamment.

Les modules venant compléter les fonctions de base du noyau nginx sont liés à la compilation : ils ne peuvent pas être activés/désactivés à chaud.

Les processus serveurs sont contrôlés par un processus maître, rendant la **modification de la configuration ou la mise à jour du logiciel possible sans arrêt du service**.

Nginx détient une part de marché non négligeable de 28% sur les sites les plus chargés du marché, juste derrière Apache (41%).

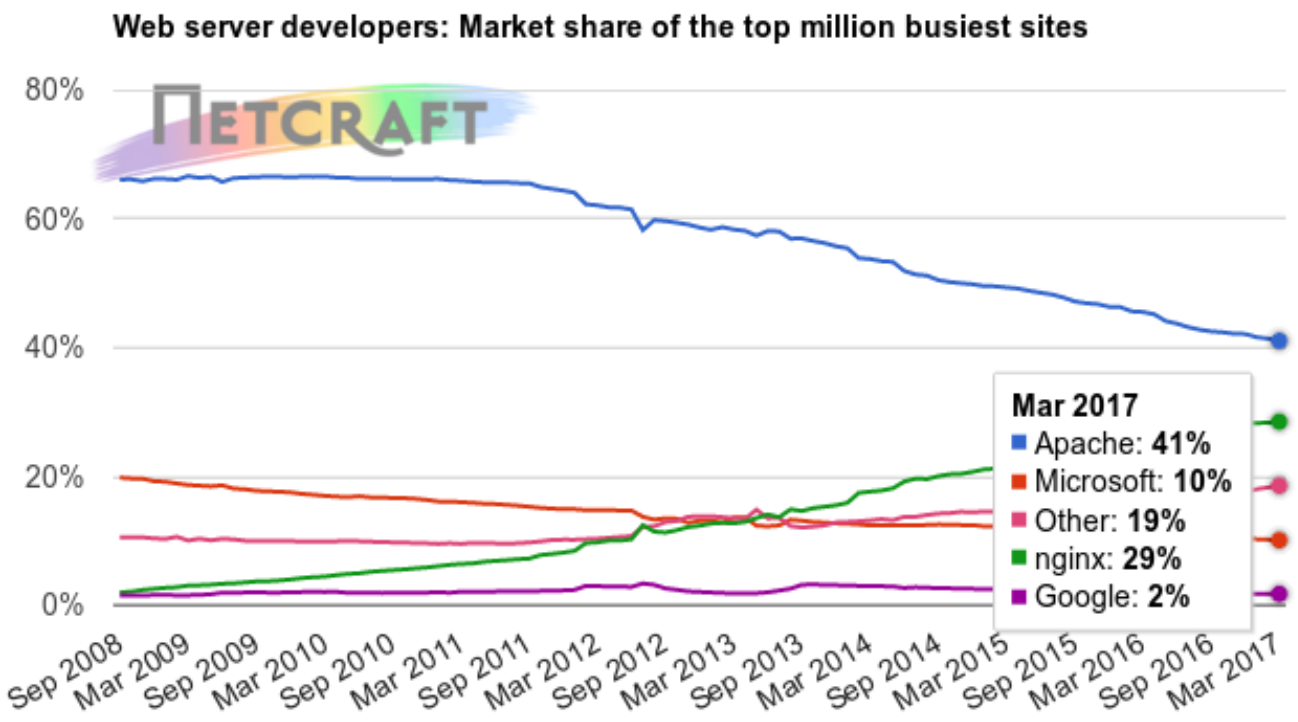


Figure 65. Statistiques NetCraft : top million busiest sites

Fonctionnalités

Nginx offre les fonctionnalités basiques suivantes :

- Hébergement de pages web statiques ;
- Génération de pages d'index automatique ;
- Proxy inverse accéléré avec cache ;
- Répartition de charge ;
- Tolérance de panne ;
- Support avec cache du FastCGI, uWSGI, SCGI et serveur de cache memcached ;
- Filtres divers pour gzip, xslt, ssi, transformation d'images, ...
- Support pour SSL/TLS et SNI ;
- Support du HTTP/2.

Autres fonctionnalités :

- Hébergement par nom ou par adresse IP ;
- Gestion du keepalive des connexions clientes ;
- Gestion des logs : syslog, rotation, buffer ;
- Ré-écriture d'URI ;

- Contrôle d'accès : par IP, mot de passe…
- Streaming FLV et MP4.

12.2. Installation du service

Nginx sous debian

Depuis Debian Wheezy, l'installation de Nginx est proposée par 3 paquets : nginx-light (le moins de modules), nginx-full (installé par le méta-paquet nginx - paquet par défaut), nginx-extras (le plus de modules).

Installation du metapaquet nginx

```
sudo apt-get install nginx
...
Les paquets supplémentaires suivants seront installés :
  libgd3 libvpx1 libxpm4 nginx-common nginx-full
Paquets suggérés :
  libgd-tools fcgiwrap nginx-doc ssl-cert
Les NOUVEAUX paquets suivants seront installés :
  libgd3 libvpx1 libxpm4 nginx nginx-common nginx-full
...
```

Nginx sous RedHat 7

Le dépôt de nginx pour RHEL/CentOS doit être ajouté aux dépôts existants. Créer le fichier /etc/yum.repos.d/nginx.repo:

```
$ sudo vi /etc/yum.repos.d/nginx.repo
```

Et ajouter le contenu suivant :

```
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/centos/7/$basearch/
gpgcheck=0
enabled=1
```

L'installation peut être lancée :

```
$ sudo yum update
$ sudo yum install nginx
```

Configuration de Nginx

La configuration de Nginx se situe sous ***/etc/nginx*** :

- Le fichier **/etc/nginx/nginx.conf** : fichier de configuration globale du serveur. Les paramètres impactent l'ensemble du serveur.
- le répertoire **sites-available** : contient les fichiers de configuration des sites.
- le répertoire **sites-enabled** : contient des liens symboliques vers les fichiers de **sites-available**, ce qui permet d'activer ou de désactiver les sites.
- le répertoire **conf.d** : répertoire contenant les paramètres communs à tous les sites.

La fonctionnalité de fichier .htaccess connues des administrateurs Apache, n'existe pas sous nginx !

Le fichier nginx.conf, épuré de tous ses commentaires, et fourni ci-dessous à titre indicatif :

Fichier nginx.conf par défaut

```
user www-data;
worker_processes 4;
pid /run/nginx.pid;

events {
    worker_connections 768;
}

http {
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;

    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers on;

    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    gzip on;
    gzip_disable "msie6";

    application/javascript text/xml application/xml application/xml+rss
text/javascript;

    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/sites-enabled/*;
}
```

Table 97. Directives de la configuration par défaut

Directives	Observations
user	Définit l'**utilisateur** et le **groupe** propriétaires du processus. Si le groupe n'est pas spécifié, le groupe du même nom que l'utilisateur est utilisé.

Directives	Observations
worker_processes	Définit le **nombre de processus**. La valeur optimale dépend de nombreux facteurs comme le nombre de coeurs CPU, les spécificités des disques durs, etc. En cas de doute, la documentation de nginx propose comme valeur de départ le nombre équivalent au nombre de coeurs CPU disponibles (la valeur **auto** essaiera de le déterminer).
pid	Définit un fichier pour stocker la valeur du **pid**.
worker_connections	Fixe le **nombre maximum de connexions simultanées** qu'un processus worker peut ouvrir (vers le client et vers les serveurs mandatés).
tcp_nopush	tcp_nopush est indissociable de l'option sendfile. Elle permet d'**optimiser la quantité d'information envoyée en une seule fois**. Les paquet ne sont envoyés que lorsque ils ont atteints leur taille maximale.
tcp_nodelay	Activer tcp_nodelay force l'**envoi immédiat des données** contenues dans la socket, quelle que soit la taille du paquet, ce qui est le contraire de ce que fait tcp_nopush.
sendfile	Optimiser l'envoi de **fichiers statiques** (option inutile dans le cadre d'une configuration en proxy-inverse). Si **sendfile** est activée, nginx s'assure que tous les paquets soient bien remplis avant d'être envoyés au client (grâce à tcp_nopush), puis, quand arrive le dernier paquet, nginx désactive tcp_nopush, et force l'envoi des données avec tcp_nodelay.
keepalive_timeout	**temps maximum avant fermeture** d'une connexion inactive.
types_hash_max_size	Nginx entretient des tables de hashage contenant des informations statiques. Permet de définir **la taille maximale de la table de hachage**.
include	Inclure un autre fichier ou d'autres fichiers qui correspondent au modèle fourni dans la configuration.
default_type	Type MIME par défaut d'une requête.
ssl_protocols	Versions du protocole TLS acceptés.
ssl_prefer_server_ciphers	Préférer l'utilisation de la cipher suite du serveur plutôt que celle du client.
access_log	Configurer les **journaux d'accès** (voir paragraphe "gestion des logs").

Directives	Observations
error_log	Configurer les **journaux d'erreurs** (voir paragraphe "gestion des logs").
gzip	Le module **ngx_http_gzip_module** est un filtre compressant les données transmises au format gzip.
gzip_disable	Désactiver gzip en fonction d'une expression régulière.

La configuration de nginx est articulée de la manière suivante :

```
# directives globales

events {
    # configuration du worker
}

http {
    # configuration du service http

    # Configuration du premier serveur en écoute sur le port 80
    server {
        listen 80 default_server;
        listen [::]:80 default_server;
        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;
        server_name _;
        location / {
            try_files $uri $uri/ =404;
        }
    }
}

mail {
    # configuration du service mail

    # directives globales du service mail

   server {
        # Un premier serveur en écoute sur le protocole pop
        listen     localhost:110;
        protocol   pop3;
        proxy      on;
   }


   server {
        # Un second serveur en écoute sur le protocole imap
       listen     localhost:143;
       protocol   imap;
       proxy      on;
   }
}
```

La configuration du premier serveur en écoute sur le port 80 se situe sous **/etc/nginx/sites-available/default**. Ce fichier est inclue au fichier nginx.conf grâce à la ligne **include /etc/nginx/sites-enabled/*;**

Configuration https

Pour configurer un service https, il faut ajouter un bloc serveur, ou modifier le bloc server existant (un bloc server peut à la fois écouter sur le port 443 et sur le port 80).

Ce bloc peut, par exemple, être ajouté au nouveau fichier *sites-available/default_https* :

```
server {
    listen              443 ssl default_server;
    ssl_protocols       TLSv1.2 TLSv1.1
    ssl_certificate     /chemin/vers/cert.pem;
    ssl_certificate_key /chemin/vers/key.key;
    root                /var/www/html;
    index               index.html index.htm index.nginx-debian.html;
    server_name         _;
    location / {
        try_files       $uri $uri/ =404;
    }
}
```

ou le server par défaut peut être modifié pour prendre en compte le https :

```
server {
    listen              80;
    listen              443 ssl;
    server_name         _;
    ssl_protocols       TLSv1.2 TLSv1.1
    ssl_certificate     /chemin/vers/cert.pem;
    ssl_certificate_key /chemin/vers/key.key;
    ...
}
```

La gestion des logs

La directive **error_log** permet de configurer les journaux d'erreurs.

Syntaxe de la directive error_log

```
error_log fichier [niveau];
```

Le premier paramètre définit un fichier qui va recevoir les logs.

Le second paramètre détermine le niveau des logs : debug, info, notice, warn, error, crit, alert ou emerg (voir le cours syslog).

L'envoi des enregistrements vers syslog peut être effectué en employant le préfixe "syslog:".

```
access_log syslog:server=192.168.1.100:5514,tag=nginx debug;
```

Nginx en proxy inverse

La fonctionnalité de proxy inverse est fourni par le module **ngx_http_upstream_module**. Il permet de définir des groupes de serveurs qui sont ensuite appelés par les directives proxy_pass ou fastcgi_pass, memcached_pass, etc.

Exemple de configuration basique, qui réparti la charge de 2/3 vers le premier serveur et d'1/3 vers le second serveur applicatif :

```
    upstream svrmetiers {
        server metiers1.formatux.fr:8080       weight=2;
        server metiers2.formatux.fr:8080       weight=1;
    }

    server {
        location / {
            proxy_pass http://svrmetiers;
        }
    }
```

Des serveurs peuvent être déclarés en secours :

```
    upstream svrmetiers {
        ...
        server secours1.formatux.fr:8080   backup;
        server secours2.formatux.fr:8080   backup;

    }
```

La directive serveur accepte de nombreux arguments :

- **max_fails=nombredetentative** : fixe le nombre de tentatives de connexion devant être en echec durant le laps de temps défini par la paramètre **fail_timeout** pour que le serveur soit considéré comme indisponible. La valeur par défaut est fixée à 1, la valeur à 0 désactive la fonctionnalité.
- **fail_timeout=time**: fixe la durée durant laquelle un nombre de connexion défini bascule le serveur comme indisponible et fixe la période de temps durant laquelle le serveur sera considéré comme indisponible. La valeur par défaut est de 10 secondes.

12.3. Sources

- https://t37.net/optimisations-nginx-bien-comprendre-sendfile-tcp-nodelay-et-tcp-nopush.html
- http://nginx.org/en/docs/

Chapitre 13. Service de cache HTTP avec Varnish

Varnish est un service de reverse-proxy-cache (mandataire inversé avec cache) HTTP, autrement dit un accélérateur de sites web.

Varnish reçoit les requêtes HTTP des visiteurs :

- s'il dispose de la réponse à la requête en cache, celle-ci est renvoyée directement au client depuis la mémoire du serveur,
- s'il ne dispose pas de la réponse, Varnish s'adresse alors au serveur web. Il lui transmet la requête, récupère la réponse, la stocke dans son cache, et répond au client.

Fournir la réponse depuis le cache en mémoire permet d'améliorer les temps de réponse aux clients. En effet, dans ce cas, il n'y a pas d'accès aux disques physiques.

Par défaut, Varnish écoute sur le port `6081` et utilise le langage VCL (*Varnish Configuration Language*) pour sa configuration. Grâce au langage VCL, il est possible de décider ce qui doit ou ne doit pas être transmit au client, ce qui doit être stocké en cache, depuis quel site et comment la réponse peut être modifiée.

Varnish est extensible par l'utilisation de modules VMOD (Varnish Modules).

13.1. Principe de fonctionnement

Dans un fonctionnement basique d'un service Web, le client communique en TCP sur le port 80 directement avec le service.

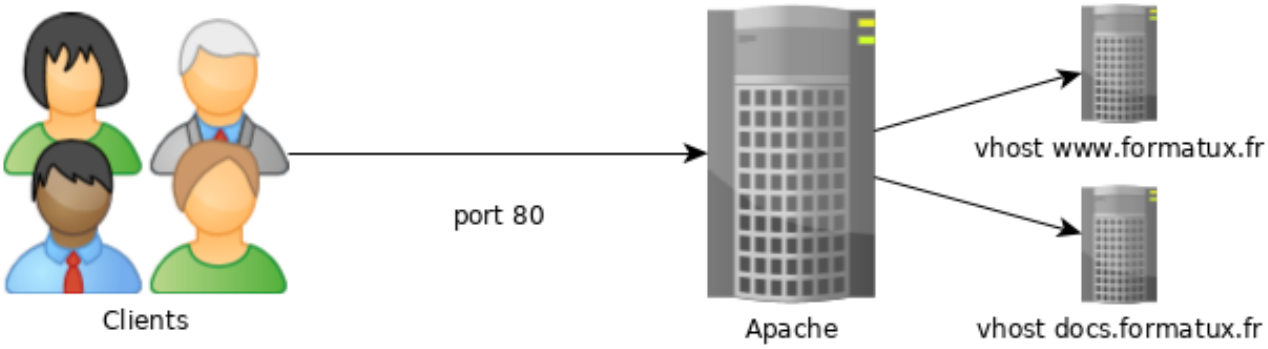


Figure 66. Fonctionnement d'un site web standard

Pour profiter du cache, le client doit communiquer avec le service web sur le port par défaut de Varnish 6081.

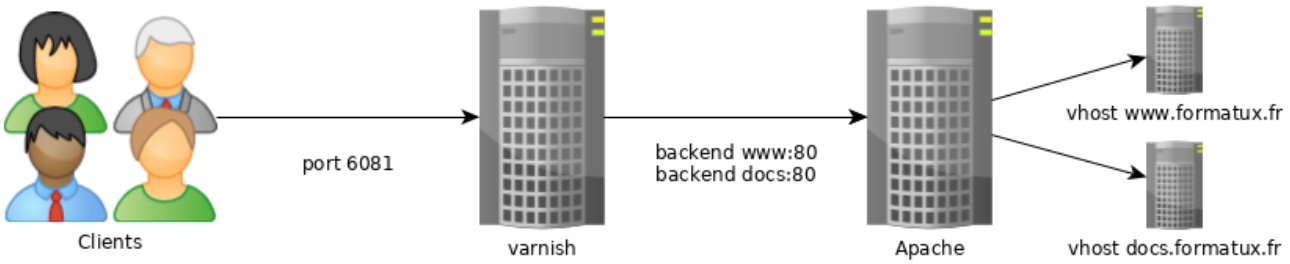


Figure 67. Fonctionnement par défaut de Varnish

Pour rendre le service transparent au client, il faudra changer le port d'écoute par défaut de Varnish et des vhosts du service web.

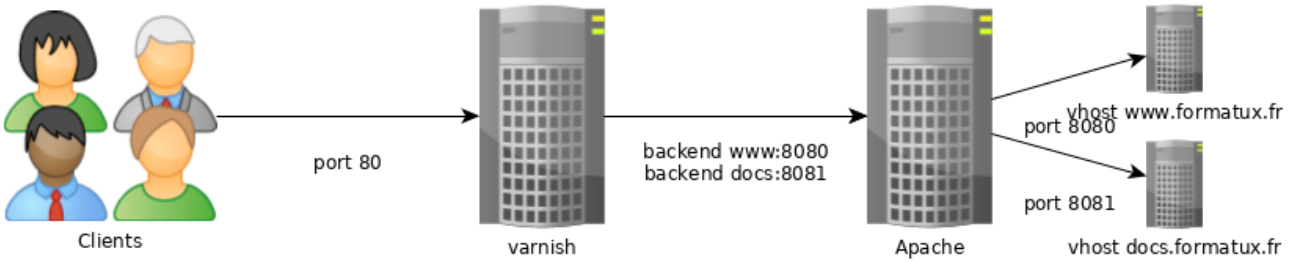


Figure 68. Mise en place transparente pour le client

13.2. Installation de Varnish

- Sous Debian :

```
# apt-get install varnish
```

- Sous RHEL :

```
# yum install varnish
```

13.3. Configuration du démon varnishd

Configuration du démon

La configuration du démon se fait dans le fichier `/etc/varnish/varnish.params` sur RedHat et dans `/etc/default/varnish` sous Debian :

1. Le fichier /etc/varnish/varnish.params sous RedHat

```
# Set this to 1 to make systemd reload try to switch VCL without restart.
RELOAD_VCL=1

# Main configuration file. You probably want to change it.
VARNISH_VCL_CONF=/etc/varnish/default.vcl

# Default address and port to bind to. Blank address means all IPv4
# and IPv6 interfaces, otherwise specify a host name, an IPv4 dotted
# quad, or an IPv6 address in brackets.
# VARNISH_LISTEN_ADDRESS=192.168.1.5
VARNISH_LISTEN_PORT=6081

# Admin interface listen address and port
VARNISH_ADMIN_LISTEN_ADDRESS=127.0.0.1
VARNISH_ADMIN_LISTEN_PORT=6082

# Shared secret file for admin interface
VARNISH_SECRET_FILE=/etc/varnish/secret

# Backend storage specification, see Storage Types in the varnishd(5)
# man page for details.
VARNISH_STORAGE="malloc,256M"

# User and group for the varnishd worker processes
VARNISH_USER=varnish
VARNISH_GROUP=varnish

# Other options, see the man page varnishd(1)
#DAEMON_OPTS="-p thread_pool_min=5 -p thread_pool_max=500 -p thread_pool_timeout=300"
```

Le fichier /etc/default/varnish sous Debian

```
DAEMON_OPTS="-a :6081 \
             -T localhost:6082 \
             -f /etc/varnish/default.vcl \
             -S /etc/varnish/secret \
             -s default,256m"
```

Depuis `systemctl`, changer les valeurs par défaut se fait en créant le fichier `/etc/systemd/system/varnish.service.d/customexec.conf` :

Le ficher `/etc/systemd/system/varnish.service.d/customexec.conf`

```
[Service]
ExecStart=
ExecStart=/usr/sbin/varnishd -a :80 -T localhost:6082 -f /etc/varnish/default.vcl -S
/etc/varnish/secret -s default,256m
```

Cela aura pour effet de surcharger les valeurs par défaut dans la partie `ExecStart` fournies avec Varnish.

Il faut ensuite relancer `systemctl daemon-reload` pour s'assurer que `systemd` prenne en compte les modifications avant de relancer Varnish.

Table 98. Les options du démon varnish

Option	Observation
`-a addr[:port]`	Ecoute les requêtes clientes sur les adresses IP et les ports spécifiés. Par défaut : toutes les adresses et sur le port `80`.
`-T addr[:port]`	Interface de gestion
`-f file`	Fichier de configuration
`-S`	Fichier contenant le secret permettant l'authentification sur le port de gestion
`-s`	Spécifier un backend de stockage du cache. L'option peut être spécifiée plusieurs fois. Les types de stockage possibles sont `malloc` (cache en mémoire puis si besoin dans la swap), ou `file` (création d'un fichier sur le disque puis mapping en mémoire). Les tailles sont exprimées en K/M/G/T (kilobytes, megabytes, ...)
`-C`	Compile la VCL en langage C et l'affiche à l'écran.

Test de la configuration et relance

Il est conseillé de vérifier la syntaxe de la VCL avant de relancer le démon varnishd.

Il s'agit de compiler en langage C le fichier de configuration VCL. Le service peut être redémarré si la compilation fonctionne et qu'aucune alarme n'est affichée.

Vérification de la syntaxe varnishd

```
varnishd -C -f /etc/varnish/default.vcl
```

ou utilisation du script init :

```
# /etc/init.d/varnish configtest
[...]
    .hit_func = VGC_function_vcl_hit,
    .init_func = VGC_function_vcl_init,
    .miss_func = VGC_function_vcl_miss,
    .pass_func = VGC_function_vcl_pass,
    .pipe_func = VGC_function_vcl_pipe,
    .purge_func = VGC_function_vcl_purge,
    .recv_func = VGC_function_vcl_recv,
    .synth_func = VGC_function_vcl_synth,
};
```

Puis purge du cache et rechargement de la configuration : (si `RELOAD_VCL=1`) :

Rechargement de la configuration

```
systemctl reload varnishd
```

ou

Redémarrage complet

```
systemctl restart varnishd
```

Il est possible de vérifier qu'une page provient du cache varnish depuis les en-têtes de la réponse HTTP :

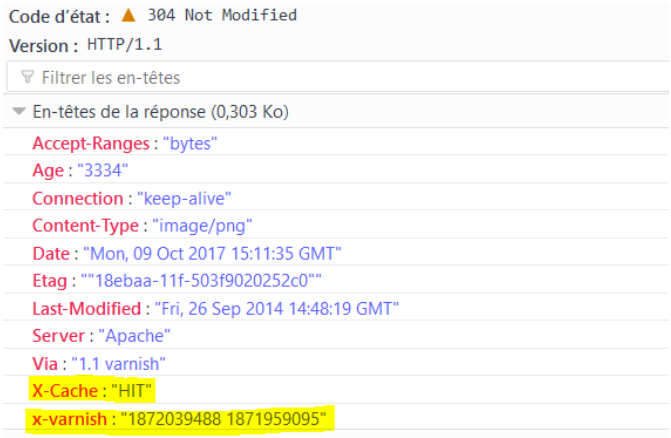


Figure 69. En-tête varnish et cache hit

13.4. La langage VCL

Les sous-routines

Varnish utilise des fichiers VCL, segmentés en sous-routines comportant les actions a exécuter. Ces

sous-routines sont exécutées uniquement dans les cas spécifiques qu'elles définissent. Dans le fichier par défaut `/etc/varnish/default.vcl`, nous trouvons les routines `vcl_recv`, `vcl_backend_response` et `vcl_deliver`.

```
# vim /etc/varnish/default.vcl
cat /etc/varnish/default.vcl
#
# This is an example VCL file for Varnish.
#
# It does not do anything by default, delegating control to the
# builtin VCL. The builtin VCL is called when there is no explicit
# return statement.
#
# See the VCL chapters in the Users Guide at https://www.varnish-cache.org/docs/
# and http://varnish-cache.org/trac/wiki/VCLExamples for more examples.

# Marker to tell the VCL compiler that this VCL has been adapted to the
# new 4.0 format.
vcl 4.0;

# Default backend definition. Set this to point to your content server.
backend default {
    .host = "127.0.0.1";
    .port = "8080";
}

sub vcl_recv {

}

sub vcl_backend_response {

}

sub vcl_deliver {

}
```

Table 99. Les sous-routines VCL

Sous-routine	Action
`vcl_recv`	Cette routine est appelée **avant** l'envoi de la requête vers le backend. Dans cette routine, il est possible de modifier les en-têtes HTTP, cookies, choisir le backend, etc. Voir actions `set req.`.
`vcl_backend_response`	Cette routine est appelée **après** la réception de la réponse du backend (`beresp` = BackEnd RESPonse). Voir actions `set bereq.` et `set beresp.`.

Sous-routine	Action
`vcl_deliver`	Cette routine est utile pour modifier la sortie de Varnish. Si besoin de modifier l'objet final (ajouter ou supprimer un en-tête, ...), il est possible de le faire dans `vcl_deliver`.

Les opérateurs VCL

Table 100. Les opérateurs VCL

Opérateur	Action
`=`	assignation
`==`	comparaison
`~`	comparaison en association avec une expression régulière et des ACL
`!`	négation
`&&`	et logique
`||`	ou logique

Les objets Varnish

`beresp` = BackEnd RESPonse

Table 101. Les différents objets Varnish

Objet	Observation
`req`	l'objet requête. Quand Varnish reçoit la requête, `req` est créé. La plupart du travail dans la sous-routine `vcl_recv` touche à cet objet.
`bereq`	l'objet requête à destination du serveur web. Varnish créé cet objet à partir de `req`.
`beresp`	l'objet réponse du serveur web. Il contient les entêtes de l'objet en provenance de l'application. Il est possible de modifier la réponse du serveur dans la sous-routine `vcl_backend_response`.
`resp`	la réponse HTTP qui va être envoyée au client. Cet objet est modifié dans la sous-routine `vcl_deliver`.
`obj`	l'objet tel que sauvegardé en cache. En lecture seule.

Les actions varnish

Table 102. Les actions les plus fréquentes

Action	Observation
`pass`	Quand `pass` est retourné, la requête et la réponse qui en suivront viendront du serveur d'application. Il n'y aura pas de cache appliqué. `pass` est retourné depuis la sous-routine `vcl_recv`.
`hash`	Quand `hash` est retourné depuis `vcl_recv`, Varnish servira le contenu depuis le cache même si la requête est configurée pour passer sans cache.
`pipe`	Cette action sert à gérer les flux. Varnish dans ce cas n'inspectera plus chaque requête mais laisse passer tous les bytes au serveur. `pipe` est utilisé par exemple par les websockets ou la gestion des flux vidéos.
`deliver`	Sert l'objet au client. En général depuis la sous-routine `vcl_backend_response`.
`restart`	Recommence le processus de traitement de la requête. Les modifications de l'objet req sont conservées.
`retry`	La requête est de nouveau transférée au serveur d'application. Utilisé depuis `vcl_backend_response` ou `vcl_backend_error` si la réponse de l'application n'est pas satisfaisante.

En résumé, les interactions possibles entre les sous-routines et les actions sont illustrées dans le schéma ci-dessous :

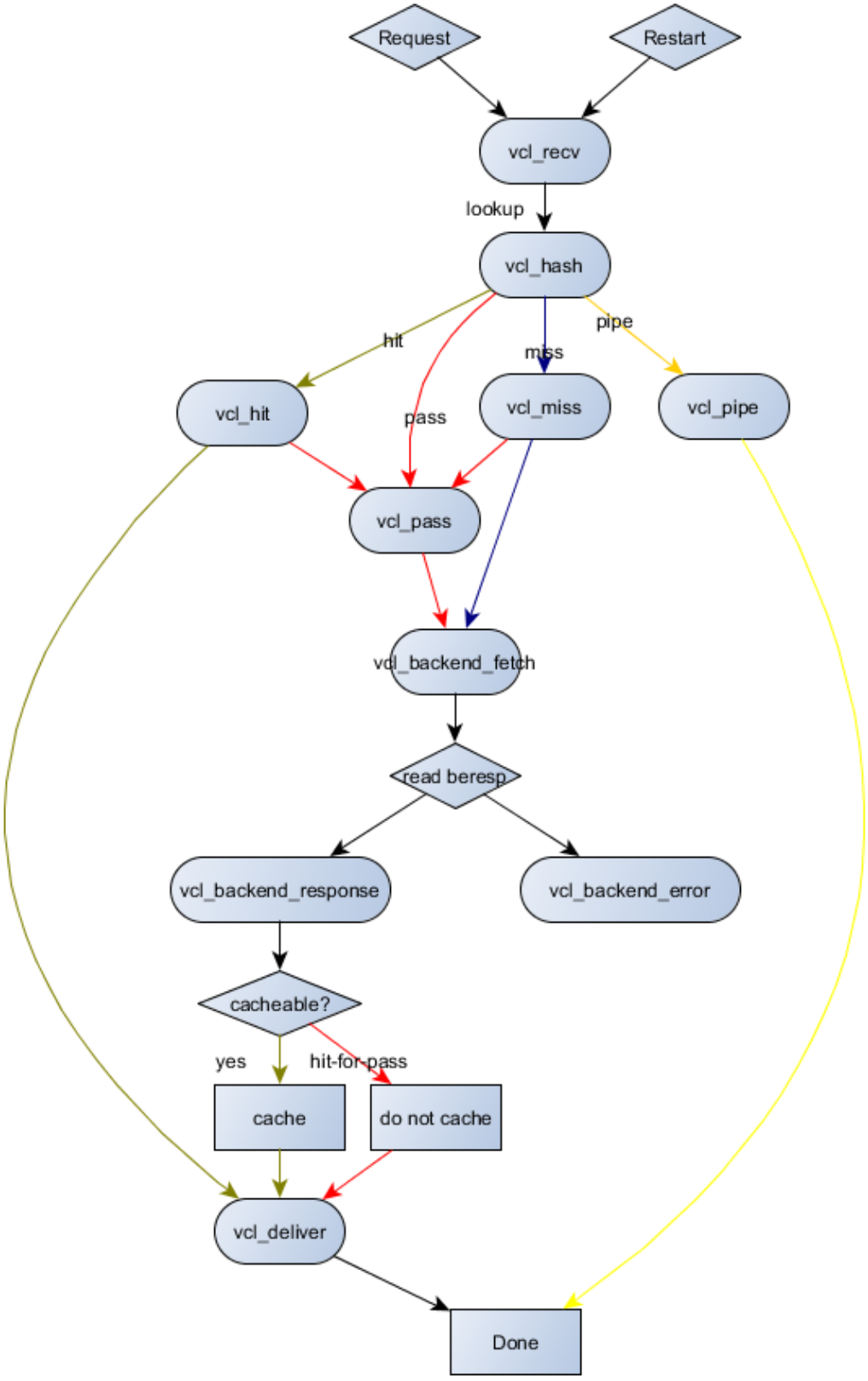


Figure 70. Fonctionnement simplifié de varnish

13.5. Configuration des backends

Varnish utilise le terme backend pour les vhosts qu'il doit mandater.

Plusieurs backend peuvent être défini sur le même serveur Varnish.

La configuration des backends se fait dans le fichier `/etc/varnish/default.vcl`.

Gestion des ACL

```
# ACL de deny
acl deny {
"10.10.0.10"/32;
"192.168.1.0"/24;
}
```

Appliquer l'ACL :

```
# Bloquer les IP de l'ACL deny
if (client.ip ~ forbidden) {
  error 403 "Acces interdit";
}
```

Ne pas cacher certaines pages :

```
# Ne pas mettre en cache les pages login et admin
if (req.url ~ "/(login|admin)") {
  return (pass);
}
```

Paramètres POST et cookies

Varnish ne met jamais en cache les requêtes HTTP de type POST ou celle contenant des cookies (qu'ils proviennent du client ou du backend).

Si le backend utilise des cookies, alors aucun contenu ne sera mis en cache.

Pour corriger ce comportement, il est possible de déréferrencer les cookies de nos requêtes :

```
sub vcl_recv {
    unset req.http.cookie;
}

sub vcl_backend_response {
    unset beresp.http.set-cookie;
}
```

Répartir les requêtes sur différents backend

Dans le cas de l'hébergement de plusieurs sites, comme par exemple un serveur de document (doc.formatux.fr) et un blog (blog.formatux.fr), il est possible de répartir les requêtes vers le bon backend.

Déclaration des backends

```
backend docs {
    .host = "127.0.0.1";
    .port = "8080";
}

backend blog {
    .host = "127.0.0.1";
    .port = "8081";
}
```

L'objet `req.backend` est modifié en fonction de l'hôte appelé dans la requête HTTP dans la sous-routine `vcl_recv` :

Sélection du backend

```
sub vcl_recv {
    if (req.http.host ~ "^doc.formatux.fr$") {
        set req.backend = nginx;
    }

    if (req.http.host ~ "^blog.formatux.fr$") {
        set req.backend = ghost;
    }
}
```

Répartir la charge

Varnish est capable de gérer la répartition de charge via des backends spécifiques appelés directors.

- Le director `round-robin` distribue les requêtes aux backends en round-robin (alternativement). Il est possible d'affecter une pondération à chaque backend.
- Le director `client` distribue les requêtes en fonction d'une affinité de session (sticky session) sur n'importe quel élément de l'en-tête (par exemple avec un cookie de session). Un client est dans ce cas toujours renvoyé vers le même backend.

Déclaration des backends

```
backend docs1 {
    .host = "10.10.11.10";
    .port = "8080";
}

backend docs2 {
    .host = "10.10.11.11";
    .port = "8080";
}
```

Le `director` permet d'associer les 2 backends définis :

Déclaration du director

```
director docs_director round-robin {
    { .backend = docs1; }
    { .backend = docs2; }
}
```

Reste à définir le `director` comme backend aux requêtes :

Association des requêtes au director

```
sub vcl_recv {
    set req.backend = docs_director;
}
```

13.6. Configuration du système

Configuration du port 80 et 443

Modifier, sous debian le fichier `/etc/default/varnish` :

```
DAEMON_OPTS="-a :80
-T localhost:6082
-f /etc/varnish/default.vcl
-S /etc/varnish/secret
...
```

Configuration d'un cache

Configuration d'un cache sur disque d'1G.

Modifier, sous debian le fichier `/etc/default/varnish` :

```
DAEMON_OPTS="-a :80
-T localhost:6082
-f /etc/varnish/default.vcl
-S /etc/varnish/secret
...
-s file,/var/lib/varnish/$INSTANCE/varnish_storage.bin,1G"
```

Adaptation d'Apache

Changement de ports réseau

Si le service Varnish est localisé sur le même serveur que le service Web (Apache ou Nginx), les deux services ne pourront plus écouter en même temps les ports par défaut 80 et 443.

Dans ce cas, il est d'usage de faire écouter le service web sur un port 8080, 8081, 8082 etc. et de laisser le port par défaut à Varnish.

```
#Listen 80
Listen 8080
```

Modification du LogFormat

Le service http étant reverse proxifié, le serveur web n'aura plus accès aux adresses IP du client mais à celui du service Varnish.

Pour prendre en compte le reverse proxy dans les logs Apache, modifier dans le fichier de configuration du serveur le format du journal d'évènement :

```
LogFormat "%{X-Forwarded-For}i %l %u %t "%r" %>s %b "%{Referer}i" "%{User-Agent}i""
varnishcombined
```

et prendre en compte ce nouveau format dans le vhost du site web :

```
CustomLog /var/log/httpd/www-access.log.formatux.fr varnishcombined
```

et rendre Varnish compatible :

```
# Compatibility with Apache format log

if (req.restarts == 0) {
  if (req.http.x-forwarded-for) {
    set req.http.X-Forwarded-For = req.http.X-Forwarded-For + ", " + client.ip;
  } else {
   set req.http.X-Forwarded-For = client.ip;
  }
}
```

13.7. Purge du cache

Le cache peut être purgé :

- en ligne de commande :

```
# varnishadm 'ban req.url ~ .'
```

- en utilisant un secret et un port différent du port par défaut :

```
# varnishadm -S /etc/varnish/secret -T 127.0.0.1:6082 'ban req.url ~ .'
```

- avec l'interface CLI:

```
# varnishadm

varnish> ban req.url ~ ".css$"
200

varnish> ban req.http.host == docs.formatux.fr
200

varnish> ban req.http.host ~ .
200
```

- via une requête HTTP PURGE :

```
$ curl -X PURGE http://www.example.org/foo.txt
```

Varnish doit être configuré pour accepter cette requête :

```
acl local {
    "localhost";
    "10.10.1.50";
}

sub vcl_recv {
    # directive a placer en premier,
    # sinon une autre directive risque de matcher avant
    # et la purge ne sera jamais effectuée
    if (req.method == "PURGE") {
        if (client.ip ~ local) {
            return(purge);
        }
    }
}
```

13.8. Gérer les backends par CLI

Les backends peuvent être marqués comme `sick` ou `healthy` en fonction des besoins d'administration ou de maintenance. Cette action permet de sortir un noeud du pool sans avoir à modifier la configuration du serveur Varnish (et donc sans le relancer) ou sans stopper le service du backend.

Visualiser le status du backend

La commande `backend.list` affiche l'ensemble des backends, même les backends qui ne disposent pas de `healthcheck` (`probe`).

```
$ varnishadm backend.list
Backend name                   Admin      Probe
site.default                   probe      Healthy (no probe)
site.front01                   probe      Healthy 5/5
site.front02                   probe      Healthy 5/5
```

Basculer un backend en status `sick`

La commande backend.set_health va permettre de basculer un backend d'un état à l'autre :

```
$ varnishadm backend.set_health site.front01 sick
```

Le backend est sorti du pool et ne reçoit plus de trafic :

```
$ varnishadm backend.list
Backend name                   Admin      Probe
site.default                   probe      Healthy (no probe)
site.front01                   sick       Sick 0/5
site.front02                   probe      Healthy 5/5
```

Notez que dans la colonne `Admin`, le backend a été marqué explicitement comme `sick`.

Rebasculer un backend en status `healthy`

De la même manière, le status peut être rebasculé en `healthy`, ce qui n'est toutefois pas encore le même status qu' `auto` (voir paragraphe suivant).

```
$ varnishadm backend.set_health site.front01 healthy
```

Le backend est de retour dans le pool, il reçoit à nouveau du trafic.

Retour à la normal en mode `auto`

Pour laisser décider varnish de l'état de ses backends, il faut impérativement rebasculer en mode `auto` les backends qui auraient étés basculés en `sick` ou `healthy` manuellement.

```
$ varnishadm backend.set_health site.front01 auto
```

13.9. La gestion des journaux

Varnish écrit ses journaux en mémoire et en binaire afin de ne pas pénaliser ses performances. Lorsqu'il ne dispose plus d'espace en mémoire, il réécrit les nouveaux enregistrements par dessus les anciens, en repartant du début de son espace mémoire.

Les journaux peuvent être consultés avec les outils varnishstat (statistiques), varnishtop (top pour Varnish), varnishlog (journalisation verbeuse) ou varnishnsca (journaux au format NCSA comme Apache) :

```
# varnishstat
# varnishtop -i ReqURL
# varnishlog
# varnishnsca
```

Filtrer les journaux

L'option `-q` permet d'appliquer des filtres sur les journaux via les commandes précédentes :

```
# varnishlog -q 'TxHeader eq MISS' -q "ReqHeader ~ '^Host: formatux\.fr$'"
# varnishncsa -q "ReqHeader eq 'X-Cache: MISS'"
```

Enregistrer les journaux sur disques

L'enregistrement des journaux sur disque est effectué par les démons `varnishlog` et `varnishnsca` en toute indépendance du démon varnishd. Le démon varnishd continue a renseigner ses journaux en mémoire sans pénaliser ses performances vers les clients, puis les autres démons se charge de copier les enregistrements sur disque.

13.10. Commandes Varnish

- `varnishlog`: Affichage du log du daemon Varnish.
- `varnishstat`: Affichage des statistiques d'utilisation de Varnish.
- `varnishhist`: Affiche un historique sous forme de graphe des requêtes faites à votre serveur Varnish.
- `varnishadm`: une interface d'administration locale de Varnish

13.11. Sources

- https://wooster.checkmy.ws/2014/03/varnish-performance-cache/
- https://wiki.evolix.org/HowtoVarnish

Chapitre 14. PHP-FPM

PHP-FPM (FastCGI Process Manager) est intégré à PHP depuis sa version 5.3.3. La version FastCGI de php apporte des fonctionnalités complémentaires.

14.1. Généralités

CGI (Common Gateway Interface) et **FastCGI** permettent la communication entre le serveur Web (Apache, Nginx) et un langage de développement (Php, Python, Java) :

- Dans le cas du **CGI**, chaque requête entraîne la création d'un **nouveau processus**, ce qui n'est pas efficace en terme de performance.
- **FastCGI** s'appuie, quant à lui, sur un **certain nombre de processus** pour le traitement de ses requêtes clientes.

PHP-FPM, apporte **en plus des meilleurs performances** :

- La possibilité de **mieux cloisonner les applications** : lancement des processus avec des uid/gid différents, avec des fichiers php.ini personnalisés,
- La gestion des statistiques,
- Gestion des journaux,
- Gestion dynamique des processus et redémarrage sans coupure de service ('graceful').

Apache possédant un module php, l'utilisation de php-fpm est moins intéressante que pour le serveur Nginx.

14.2. Installation

Debian 8

L'installation de php-fpm s'effectue depuis les dépôts apt :

```
$ sudo apt-get install php5-fpm
...
Les paquets supplémentaires suivants seront installés :
  libapparmor1 libonig2 libqdbm14 php5-cli php5-common php5-json php5-readline
Paquets suggérés :
  php-pear php5-user-cache
...
```

Arrêt et relance du service

Via systemd, la commande suivante stoppe le service :

```
$ sudo systemctl stop php5-fpm
```

La commande suivante relance le service :

```
$ sudo systemctl restart php5-fpm
```

Pour simplement recharger la configuration et prendre les modifications effectuées en compte :

```
$ sudo systemctl reload php5-fpm
```

14.3. Configuration

Les fichiers de configuration de php-fpm se situent sous ***/etc/php5/fpm***.

php-fpm utilise la syntaxe de php.ini pour ses fichiers de configuration (php-fpm.conf et fichier de configuration des pools).

Le fichier **/etc/php5/fpm/php-fpm.conf**, dans sa version minimale, contient :

```
[global]
pid = /run/php5-fpm.pid
error_log = /var/log/php5-fpm.log


include=/etc/php5/fpm/pool.d/*.conf
```

Le fichier ***/etc/php5/fpm/pool.d/www.conf*** contient, quant à lui, les quelques directives suivantes :

```
[www]
user = www-data
group = www-data
listen = /var/run/php5-fpm.sock
listen.owner = www-data
listen.group = www-data

pm = dynamic
pm.max_children = 5
pm.start_servers = 2
pm.min_spare_servers = 1
pm.max_spare_servers = 3

chdir = /
```

Table 103. Directives de la configuration par défaut

Directives	Observations
[pool]	Nom du pool de processus. Le fichier de configuration peut être composé de plusieurs pools de processus (le nom du pool entre crochet commence une nouvelle section)
listen	Définit l'interface d'écoute ou le socket unix utilisé. Exemple : **listen** = **127.0.0.1:9000** Ou via une socket Unix : **listen** = **/var/run/php5-fpm.sock**. L'utilisation d'une socket lorsque le serveur web et le serveur php sont sur la même machine permet de s'affranchir de la couche TCP/IP.
Pour une interface : listen.owner, listen.group, listen.mode	Spécifier le propriétaire, le groupe propriétaire et les droits de la socket Unix. Attention : les deux serveurs (web et php) doivent disposer des droits d'accès sur la socket.
Pour une socket : listen.allowed_clients	restreindre l'accès au serveur php à certaines adresses IP. Exemple : **listen.allowed_clients** = **127.0.0.1**

Configuration statique ou dynamique

Les processus de php-fpm peuvent être gérés de manière statique ou dynamique :

- En mode **static** : le nombre de processus fils est fixé par la valeur de pm.max_children ;

Configuration de php-fpm en mode static

```
pm = static
pm.max_children = 10
```

Cette configuration lancera 10 processus.

- En mode **dynamic** : php-fpm lancera au maximum le nombre de processus spécifié par la valeur de pm.max_children, en commençant par lancer un nombre de processus correspondant à pm.start_servers, et en gardant au minimum la valeur de pm.min_spare_servers de processus inactifs et au maximum pm.max_spare_servers processus inactifs.

Exemple :

```
pm                  = dynamic
pm.max_children     =  5
pm.start_servers    =  2
pm.min_spare_servers =  1
pm.max_spare_servers =  3
```

Php-fpm créera un nouveau processus en remplacement d'un processus qui aura traité un nombre de requêtes équivalent à pm.max_requests.

Par défaut, la valeur de pm.max_requests est à 0, ce qui signifie que les processus ne sont jamais recyclés. Utiliser l'option pm.max_requests peut être intéressant pour des applications présentant des fuites mémoires.

Configuration avancée

Status du processus

Php-fpm propose, à l'instar de Apache et de son module mod_status, une page indiquant l'état du processus.

Pour activer la page, il faudra fournir à nginx son chemin d'accès via la directive pm.status_path :

```
pm.status_path = /status
```

Journaliser les requêtes longues

La directive slowlog indique le fichier recevant la journalisation des requêtes trop longues (dont le temps dépasse la valeur de la directive request_slowlog_timeout).

Le fichier généré se situe par défaut ***/var/log/php5-fpm.log.slow***.

```
request_slowlog_timeout = 30
slowlog = /var/log/php5-fpm.log.slow
```

Une valeur à 0 de request_slowlog_timeout désactive la journalisation.

Configuration avec nginx

Le paramètrage par défaut de nginx intègre déjà la configuration nécessaire pour faire fonctionner php avec php-fpm.

Le fichier de configuration fastcgi.conf (ou fastcgi_params) se situe sous /**etc/nginx/** :

```
fastcgi_param  SCRIPT_FILENAME    $document_root$fastcgi_script_name;
fastcgi_param  QUERY_STRING       $query_string;
fastcgi_param  REQUEST_METHOD     $request_method;
fastcgi_param  CONTENT_TYPE       $content_type;
fastcgi_param  CONTENT_LENGTH     $content_length;

fastcgi_param  SCRIPT_NAME        $fastcgi_script_name;
fastcgi_param  REQUEST_URI        $request_uri;
fastcgi_param  DOCUMENT_URI       $document_uri;
fastcgi_param  DOCUMENT_ROOT      $document_root;
fastcgi_param  SERVER_PROTOCOL    $server_protocol;
fastcgi_param  HTTPS              $https if_not_empty;

fastcgi_param  GATEWAY_INTERFACE  CGI/1.1;
fastcgi_param  SERVER_SOFTWARE    nginx/$nginx_version;

fastcgi_param  REMOTE_ADDR        $remote_addr;
fastcgi_param  REMOTE_PORT        $remote_port;
fastcgi_param  SERVER_ADDR        $server_addr;
fastcgi_param  SERVER_PORT        $server_port;
fastcgi_param  SERVER_NAME        $server_name;

# PHP only, required if PHP was built with --enable-force-cgi-redirect
fastcgi_param  REDIRECT_STATUS    200;
```

Pour que nginx traite les fichiers .php, les directives suivantes doivent être ajoutées au fichier de configuration du site :

- Si php-fpm écoute sur le port 9000 :

```
location ~ \.php$ {
  include /etc/nginx/fastcgi_params;
  fastcgi_pass  127.0.0.1:9000;
}
```

- Si php-fpm écoute sur une socket unix :

```
location ~ \.php$ {
  include /etc/nginx/fastcgi_params;
  fastcgi_pass  unix:/var/run/php5-fpm.sock;
}
```

Chapitre 15. Serveur de base de données MySQL - MariaDB

MySQL et **MariaDB** sont des systèmes de gestion de base de données relationnelles (**SGBD-R**) open source.

La littérature anglophone utilise le terme RDBMS (Relational DataBase Managed System).

MySQL a été développé par **Michael "Monty" Widenius** (informaticien Finlandais), qui fonde MySQL AB en 1995. MySQL AB est rachetée par la société **SUN** en 2008, elle-même rachetée par **Oracle** en en 2009 qui est toujours propriétaire du logiciel MySQL et qui le distribue sous double licence GPL et propriétaire.

En 2009, Michael Widenius quitte SUN, fonde la société Monty Program AB et lance le développement de son fork communautaire de MySQL : **MariaDB** sous licence GPL. La gouvernance du projet est confiée à la fondation MariaDB, ce qui assure au projet de rester libre.

Rapidement, la majorité des distributions Linux proposent par défaut les paquets MariaDB à la place de ceux de MySQL et des grands comptes, comme Wikipedia ou Google, adoptent eux aussi le fork communautaire.

MySQL et MariaDB font parti des SGBR-R les plus utilisés au monde (professionnellement et par le grand public), notamment pour des applications web (**LAMP** : Linux+Apache+Mysql-MariaDB+Php).

Les concurrents principaux de Mysql-MariaDB sont :

- Oracle DB,
- Microsoft SQL Server.

Les services applicatifs sont multi-threads et multi-utilisateurs, ils fonctionnent sur la majorité des systèmes d'exploitations (Linux, Unix, BSD, Mac OSx, Windows) et sont accessibles depuis de nombreux langages de programmation (Php, Java, Python, C, C++, Perl, ...).

Plusieurs moteurs sont supportés, ce qui permet, au sein d'une même base, d'attribuer un moteur différent aux différentes tables en fonction du besoin :

- **MyISAM** : le plus simple mais ne supporte pas les transactions ni les clefs étrangères. Il s'agit d'un moteur séquentiel indexé.
- **InnoDB** : gestion de l'intégrité des tables (clé étrangères et transactions) mais occupe plus d'espace sur le disque. C'est le moteur **par défaut** depuis la version 5.6 de MySQL. Il s'agit d'un moteur transactionnel.
- **Memory** : les tables sont stockées en mémoire.
- **Archive** : compression des données à l'insertion ce qui fait gagner de l'espace disque mais

ralenti les requêtes de recherche (données froides).

- ...

Il s'agira d'adopter un moteur selon le besoin : Archive pour le stockage de log, Memory pour des données temporaires, etc.

MySQL utilise la port **3306/tcp** pour sa communication sur le réseau.

15.1. Installation

Sous CentOS

Le serveur MySQL est installé par le paquet **mysql-community-server**, mais le paquet distribué dans les dépôts de base de la distribution est le paquet mariadb-server :

```
$ yum search mysql-community-server mariadb-server

mariadb-server.x86_64 : The MariaDB server and related files

  Name and summary matches mostly, use "search all" for everything.
Warning: No matches found for: mysql-community-server
```

L'installation de mariadb-server peut donc se faire directement :

```
$ sudo yum install mariadb-server
$ sudo systemctl enable mariadb
Created symlink from /etc/systemd/system/multi-user.target.wants/mariadb.service to
/usr/lib/systemd/system/mariadb.service.
$ sudo systemctl start mariadb
```

L'installation ajoute au système un utilisateur :

```
$ cat /etc/passwd
...
mysql:x:27:27:MariaDB Server:/var/lib/mysql:/sbin/nologin
...
```

Pour installer le paquet mysql-server, il faudra passer par l'ajout du dépôt MySQL. Ce dépôt fourni les dernières versions des paquets logiciels pour les applications :

- Serveur MySQL,
- MySQL Cluster
- MySQL Workbench

- MySQL Fabric
- MySQL Router
- MySQL Utilities
- MySQL Connector / ODBC
- MySQL Connector / Python
- MySQL Shell

Les dépôts MySQL se situent sous : http://repo.mysql.com/.

- Pour RHEL 7

Installer le rpm contenant la définition du dépôt :

```
$ sudo yum install http://repo.mysql.com/mysql-community-release-el7.rpm
```

L'installation de ce RPM a pour effet de créer le fichier */etc/yum.repos.d/mysql-community.repo* :

Le fichier de dépôts /etc/yum.repos.d/mysql-community.repo

```
[mysql-connectors-community]
name=MySQL Connectors Community
baseurl=http://repo.mysql.com/yum/mysql-connectors-community/el/7/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql

[mysql-tools-community]
name=MySQL Tools Community
baseurl=http://repo.mysql.com/yum/mysql-tools-community/el/7/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql

# Enable to use MySQL 5.5
[mysql55-community]
name=MySQL 5.5 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.5-community/el/7/$basearch/
enabled=0
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql

# Enable to use MySQL 5.6
[mysql56-community]
name=MySQL 5.6 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.6-community/el/7/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql

[mysql57-community]
name=MySQL 5.7 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.7-community/el/7/$basearch/
enabled=0
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
```

Comme vous pouvez le constater, le dépôt activé par défaut et le dépôt contenant la version 5.6 du paquet. Pour installer la version 5.7 du serveur, il faudra activer le dépôt correspondant et désactiver les dépôts des autres versions.

```
$ sudo yum install mysql-community-server
$ sudo systemctl enable mysqld
$ sudo systemctl start mysqld
```

L'utilisateur sera cette fois-ci quelque peu différent (uniquement au niveau du commentaire et du shell) :

```
$ cat /etc/passwd
...
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/false
...
```

Si le serveur Mariadb est déjà installé, celui-ci sera remplacé par le serveur MySQL !

Sous Debian

Sous Debian, les 2 paquets mariadb-server et mysql-server sont disponibles dans les dépôts de base. Il n'est donc pas nécessaire d'ajouter le dépôt officiel.

```
$ sudo apt-cache search mysql-server
mysql-server - MySQL database server (metapackage depending on the latest version)
```

```
$ sudo apt-cache search mariadb-server
mariadb-server - MariaDB database server (metapackage depending on the latest version)
mariadb-server-10.0 - MariaDB database server binaries
```

Installer au choix un des deux paquets (selon vos préférences) :

```
$ sudo apt-get install mariadb-server
```

Un utilisateur est également créé :

```
cat /etc/passwd
...
mysql:x:112:116:MySQL Server,,,:/nonexistent:/bin/false
...
```

A l'installation du paquet mysql-server, l'installateur demande la saisie d'un mot de passe pour l'utilisateur root@localhost.

15.2. Gestion du service

Le nom de l'unité cible systemd est différente selon la distribution et le paquet :

- sous RedHat :
  - mysqld pour mysql
  - mariadb
- sous debian :
  - mysql
  - mariadb

15.3. Sécurisation

- Depuis la version 5.7 :

Au démarrage du serveur, une bi-clef ssl est générée dans le dossier de données, le plugin validate_password est installé puis activé. Un compte utilisateur '**root**' est créé et son mot de passe est stocké dans le fichier journal de log.

Extrait du fichier /var/log/mysqld.log

```
[Note] A temporary password is generated for root@localhost: akTjtLSPq6/5
```

Le plugin validate_password nécessite que le mot de passe soit long d'au moins 8 caractères et qu'il soit composé d'au moins une majuscule, une minuscule, un entier et un caractère spécial.

Ce mot de passe peut immédiatement être changé avec les commandes suivantes :

```
$ mysql -uroot -p
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'M1nNouve@uMDP';
```

Ne pas lancer la commande mysql_secure_installation pour un serveur en version 5.7 (la procédure est effectuée automatiquement au moment de l'installation).

- Avec la version 5.6 et précédente :

Le programme mysql_secure_installation effectue des opérations importantes comme attribuer un mot de passe à l'utilisateur root, supprimer les utilisateurs anonymes, etc.

Pour des raisons évidentes de sécurité, la commande mysql_secure_installation devrait toujours être lancée après l'installation :

```
$ mysql_secure_installation
```

15.4. Configuration

Le fichier ***/etc/my.cnf*** (redhat) ou ***/etc/mysql/my.cnf*** (debian) contient la configuration à la fois du client et du serveur.

Le fichier /etc/my.cf par défaut

```
cat /etc/my.cnf

[mysqld]
#
# Remove leading # and set to the amount of RAM for the most important data
# cache in MySQL. Start at 70% of total RAM for dedicated server, else 10%.
# innodb_buffer_pool_size = 128M
#
# Remove leading # to turn on a very important data integrity option: logging
# changes to the binary log between backups.
# log_bin
#
# Remove leading # to set options mainly useful for reporting servers.
# The server defaults are faster for transactions and fast SELECTs.
# Adjust sizes as needed, experiment to find the optimal values.
# join_buffer_size = 128M
# sort_buffer_size = 2M
# read_rnd_buffer_size = 2M
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock

# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0

log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

Le fichier /etc/mysql/my.cnf

```
# The MariaDB configuration file
#
# The MariaDB/MySQL tools read configuration files in the following order:
# 1. "/etc/mysql/mariadb.cnf" (this file) to set global defaults,
# 2. "/etc/mysql/conf.d/*.cnf" to set global options.
# 3. "/etc/mysql/mariadb.conf.d/*.cnf" to set MariaDB-only options.
# 4. "~/.my.cnf" to set user-specific options.
#
# If the same option is defined multiple times, the last one will apply.
#
# One can use all long options that the program supports.
# Run program with --help to get a list of available options and with
# --print-defaults to see which it would actually understand and use.

#
# This group is read both both by the client and the server
# use it for options that affect everything
#
[client-server]

# Import all .cnf files from configuration directory
!includedir /etc/mysql/conf.d/
!includedir /etc/mysql/mariadb.conf.d/
```

Alors que le serveur mysqld est en cours de fonctionnement, la commande *ps* nous renseigne sur les paramètres en cours :

```
ps -ef | grep mysql
root      3324     1  0 08:27 ?        00:00:00 /bin/bash /usr/bin/mysqld_safe
mysql     3468  3324  0 08:27 ?        00:00:01 /usr/sbin/mysqld --basedir=/usr
--datadir=/var/lib/mysql --plugin-dir=/usr/lib/mysql/plugin --user=mysql --skip-log
-error --pid-file=/var/run/mysqld/mysqld.pid --socket=/var/run/mysqld/mysqld.sock
--port=3306
root      3469  3324  0 08:27 ?        00:00:00 logger -t mysqld -p daemon error
```

Le processus **mysqld_safe** est le responsable du démarrage du serveur. La majorité des options de mysqld_safe sont les mêmes que celles de mysqld.

La démon **mysqld** est démarré avec les options suivantes :

- **--basedir=/usr** : chemin vers l'installation de MySQL.
- **--datadir=/var/lib/mysql** : chemin vers le dossier contenant les bases de données.
- **--plugin-dir=/usr/lib/mysql/plugin** : chemin vers le dossier contenant les plugins.
- **--user=mysql** : compte utilisateur sous lequel le serveur est lancé.

- **--skip-log-error** : n'enregistre pas dans le journal d'erreur mais utilise syslog à la place.
- **--pid-file=/var/run/mysqld/mysqld.pid** : fichier contenant le PID du service.
- **--socket=/var/run/mysqld/mysqld.sock** : chemin vers la socket Unix utilisée par les clients locaux.
- **--port=3306** : port TCP utilisé. Le port 3306 est le port par défaut.

La commande **mysqld --help --verbose** donne la liste complète des options disponibles et leurs valeurs actuelles.

Autres options utiles :

- **--bind-address** : par défaut, le serveur écoute sur le réseau et sur l'adresse 0.0.0.0 (toutes les adresses). IL est possible de restreindre l'écoute sur une ou plusieurs interface, comme Debian le fait en n'écoutant que sur l'adresse de loopback localhost.

15.5. Utilisation

La commande mysql

La commande **mysql** permet de se connecter à un serveur MySQL.

Syntaxe de la commande mysql

```
[root]# mysql [–u utilisateur] [-p[password]] [-D database] [-h host]
```

Exemples de la commande shutdown

```
[root]# mysql -u root -pmdparoot
```

Table 104. Options de la commande shutdown

Options	Observations
-u	L'utilisateur MySQL a utilisé lors de la connexion au serveur.
-p	Le mot de passe à utiliser lors de la connexion au serveur. Il ne doit pas y avoir d'espace entre l'option -p et le mot de passe. Sans mot de passe suivant l'option -p, MySQL en demandera un.
-D	La base de données à utiliser.
-h	Se connecter à un serveur distant.

Gestion des utilisateurs

Création/suppression d'un utilisateur et gestion de ses privilèges

- Connexion au serveur MySQL :

```
$ mysql -u root -p
```

- Créer un nouvel utilisateur :

```
CREATE USER 'utilisateur'@'localhost' IDENTIFIED BY 'password';
```

Dans la requête ci-dessus, l'utilisateur ne pourra se connecter au serveur MySQL que localement (localhost). Si l'utilisateur doit pouvoir se connecter à distance depuis l'adresse IP 172.16.1.100, il faudra utiliser la requête suivante :

```
CREATE USER 'utilisateur'@'172.16.1.100' IDENTIFIED BY 'password';
```

- Donner des droits à l'utilisateur :

Si l'utilisateur doit accéder à la totalité des bases en lecture :

```
GRANT SELECT ON *.* TO 'utilisateur'@'localhost';
```

Syntaxe de la requête GRANT

```
GRANT <permission type> ON <database>.<table> TO '<username>'@'<host>';
```

- Suppression des droits :

Pour supprimer les droits d'un utilisateur les mots clefs **GRANT** et **ON** seront remplacés par **REVOKE** et **TO**.

Syntaxe de la commande REVOKE

```
REVOKE <permission type> ON <database>.<table> FROM '<username>'@'<host>';
```

- Suppression d'un utilisateur :

Un utilisateur est supprimé avec le mot clé **DROP** :

```
DROP USER 'utilisateur'@'localhost';
```

- Appliquer les modifications :

Pour que les modifications prennent effet, il faut recharger tous les privilèges :

```
FLUSH PRIVILEGES;
```

- Quitter l'environnement mysql :

Pour quitter l'environnement mysql, il faudra saisir la commande :

```
exit;
```

Les types de permissions

Il existe différents types de permissions à offrir aux utilisateurs :

- **SELECT** : lecture sur les données
- **USAGE** : autorisation de se connecter au serveur (donné par défaut à la création d'un nouvel utilisateur)
- **INSERT** : ajouter de nouveaux tuples dans une table.
- **UPDATE** : modifier des tuples existantes
- **DELETE** : supprimer des tuples
- **CREATE** : créer des nouvelles tables ou des bases de données
- **DROP** : supprimer des tables ou des bases de données existantes
- **ALL PRIVILEGES** : tous les droits
- **GRANT OPTION** : donner ou supprimer des droits aux autres utilisateurs

Définition de **tuple** : Collection ordonnée des valeurs d'un nombre indéfini d'attributs relatifs à un même objet.

15.6. La gestion des journaux

MySQL renseigne différents journaux :

- Le journal des erreurs

Il contient les messages générés au démarrage et à l'arrêt du service ainsi que les événements importants (warning et error).

- Le journal binaire

Ce journal (au format binaire) conserve toutes les actions qui modifient la structure des bases ou les données. En cas de restauration d'une base de données, il faudra restaurer la sauvegarde ET rejouer le journal binaire pour retrouver l'état de la base de données avant le crash.

- Le journal des requêtes

Toutes les requêtes émises par les clients sont consignées dans ce journal.

- Le journal des requêtes lentes

Les requêtes lentes, c'est à dire qui mettent plus qu'un certain temps (ce temps est paramétrable) à s'exécuter sont consignées à part dans ce journal. Une analyse de ce fichier permettra éventuellement de prendre des mesures afin de réduire leur temps d'éxécution (mettre en place des index par exemple ou modifier l'application cliente).

Hormis le journal binaire, ces journaux sont au format texte, ce qui permet de les exploiter directement !

Les paramètres du démon **mysqld** concernant les journaux sont :

Table 105. Paramètres de gestion des journaux d'enregistrements

Paramètres	Observations
--log-error=pathtofile	le journal des erreurs
--log-bin=path	le journal binaire
--log	le journal des requêtes
--slow-queries=pathtofile	le journal des requêtes lentes
--long_query_time=secondes	durée à partir de laquelle une requête est considérée comme longue et donc consignée dans le journal des requêtes lentes.

15.7. La chasse aux requêtes longues

Afin d'activer le journal d'enregistrement des requêtes longues, éditez le fichier de configuration my.cnf pour ajouter les lignes suivantes :

```
slow_query_log      = 1
slow_query_log_file = /var/log/mysql/mysql-slow.log
long_query_time     = 2
```

La valeur minimale pour la variable long_query_time est de 0 et la valeur par défaut est fixée à 10 secondes.

Relancez le service pour que les modifications soient prises en compte.

Une fois que le fichier de log se rempli, il est possible de l'analyser avec la commande **mysqldumpslow**.

Syntaxe de la commande mysqldumpslow

```
mysqldumpslow [options] [log_file ...]
```

Table 106. Options de la commande mysqldumpslow

Option	Observation
-t n	N'affiche que les n premières requêtes
-s sort_type	Trie en fonction du nombre de requête (c), .
-r	Inverse l'affichage des résultats

Les types de tri peuvent être :

- c : en fonction du nombre de requête
- t ou at : en fonction du temps d'exécution ou de la moyenne du temps d'exécution (a pour average)
- l ou al : en fonction du temps de verrou ou de sa moyenne
- r ou ar : en fonction du nombre de lignes renvoyées ou de sa moyenne

15.8. La sauvegarde

Comme pour tout SGBD-R, la sauvegarde d'une base de données doit se faire alors que les données ne sont pas en cours de modification. Cela est possible :

- alors que le service est à l'arrêt : il s'agit d'une **sauvegarde offline** ;
- alors que le service fonctionne mais un verrou est posé (pour momentanément suspendre toutes modifications) : il s'agit d'une **sauvegarde online**
- en utilisant un instantané du système de fichiers type LVM, permettant de sauvegarder les données avec un système de fichiers à froid.

Le format de la sauvegarde peut être un fichier au format ASCII (texte), représentant l'état de la base et de ses données sous forme d'ordres SQL, ou un fichier binaire, correspondant aux fichiers de stockage de MySQL.

Tandis que la sauvegarde d'un fichier binaire peut se faire à l'aide des utilitaires courants comme tar ou cpio, la sauvegarde au format ASCII nécessite l'utilisation d'un utilitaire comme **mysqldump**.

N'oubliez pas qu'après la restauration d'une sauvegarde complète, la restauration des fichiers binaires permettent de compléter la reconstitution des données.

Exemple de sauvegarde d'une base de données mabase :

```
$ mysqldump -u root -p --opt mabase > /tmp/sauvegarde-mabase.sql
```

Lors de la sauvegarde, l'utilitaire mysqldump pose un verrou sur la base.

Le fichier obtenu permet de restaurer les données de la base (la base doit toujours exister ou être recréée au préalable !) :

```
$ mysql -u root -p mabase < /tmp/sauvegarde-mabase.sql
```

15.9. Outils de gestions

- PhpMyAdmin
- MySQL Workbench
- MySQL administrator

Chapitre 16. Serveurs MySQL/MariaDB - Multi-Maîtres

MySQL/MariaDB fonctionne par défaut en Standalone, chaque serveur étant autonome.

L'accès aux données étant crucial pour les entreprises, les administrateurs chercheront des solutions de **répartitions de charge** (répartition des requêtes en écriture sur le serveur maître et celles en lecture sur les serveurs esclaves).

MySQL propose un système de réplication à sens unique, basée sur :

- Un serveur **maître** contenant la base de données en **écriture**,
- Un ou plusieurs serveurs **esclaves**, qui vont se synchroniser sur le serveur maître.

Ce mode de réplication est dit "Asynchrone". La données consultée sur un noeud peut être différente de la valeur stockée sur le noeud maître le temps de la réplication.

Un serveur esclave peut devenir maître pour d'autres serveurs, cascadant ainsi la réplication d'une donnée.

Une telle infrastructure permet de répartir la charge de travail (requêtes en écriture sur le maître, les autres requêtes vers les esclaves, sauvegarde sur un esclave dédié) mais :

- l'application doit avoir été **développée spécifiquement** pour utiliser l'infrastructure,
- en cas de panne sur un des serveurs (panne du maître ou d'un esclave d'esclave), une **action de l'administrateur** sera nécessaire,
- en cas d'évolution de l'infrastructure, l'application devra être modifiée.

Dans ce type de réplication, l'état de la base de données est dans un premier temps dupliqué sur le serveur esclave. Ensuite le serveur esclave entre en contact avec le serveur maître et récupère les données binaires depuis une position de référence et se met à jour.

1. Le journal binaire doit être activé sur tous les serveurs. Un identifiant unique doit être attribué à chaque serveur.
2. Un compte de réplication avec le privilège de 'REPLICATION SLAVE' doit être créé sur le serveur maître permettant à l'esclave de s'y connecter.
3. La base de données du maître est sauvegardée puis exportée vers l'esclave (manuellement).
4. Sur le maître, le fichier binaire en cours et la position dans le journal doivent être récupérés (avec la commande SQL 'show master status;')
5. Sur l'esclave, le maître peut être configuré et le processus de réplication démarré.

Pour des besoins de **hautes disponibilités** (**HA** : High Availability), l'administrateur pourra :

- configurer ses serveurs en Multi-Maîtres : chaque esclave étant également le maître des autres

serveurs. Cette technique est limitée à 64 maîtres maximum.

- utiliser des technologies de clustering avec MySQL Cluster qui propose un système de réplication Synchrone et une répartition de la charge.

Dans ce chapitre, nous allons vous présenter la mise en oeuvre d'une **réplication Multi-Maître à deux noeuds**. La plate-forme de test est constituée de deux noeuds CentOS 7 :

- mirroir1 : 192.168.100.173
- mirroir2 : 192.168.100.211

16.1. Configuration des noeuds

Lors de la mise en place de cluster, il est recommandé de s'assurer de la bonne résolution des noms de machines de chaque membre du cluster dès le démarrage de la machine et même en absence de service DNS.

Configuration du fichier /etc/hosts sur chaque noeud :

Fichier /etc/hosts

```
...
192.168.100.173         mirroir1.local.lan      mirroir1
192.168.100.211         mirroir2.local.lan      mirroir2
```

Installation de mariadb-server :

```
yum install mariadb-server
```

Le service peut être démarré sur chacun des noeuds :

```
systemctl enable mariadb
systemctl start mariadb
```

16.2. Création de la base à répliquer

La base de données à répliquer est crée sur les deux noeuds :

```
mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 5.5.52-MariaDB MariaDB Server

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB > show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
4 rows in set (0.00 sec)

MariaDB > create database mabase;
Query OK, 1 row affected (0.01 sec)
```

16.3. Création des utilisateurs MySQL pour la réplication

Sur chaque noeud doit être créé un utilisateur qui disposera des droits de réplication sur le serveur distant.

- Sur le noeud 1 :

```
MariaDB > create user 'mirroir'@'mirroir2.local.lan' identified by 'm!rro!r';
Query OK, 0 rows affected (0.01 sec)

MariaDB > grant replication slave on *.* to 'mirroir'@'mirroir2.local.lan';
Query OK, 0 rows affected (0.00 sec)
```

- Sur le noeud 2 :

```
MariaDB > create user 'mirroir'@'mirroir1.local.lan' identified by 'm!rro!r';
Query OK, 0 rows affected (0.01 sec)

MariaDB > grant replication slave on *.* to 'mirroir'@'mirroir1.local.lan';
Query OK, 0 rows affected (0.00 sec)
```

16.4. Configuration de MySQL

Le fichier my.cnf peut être modifié comme suit :

- Sur le noeud 1 :

Fichier /etc/my.cnf sur le noeud 1

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
port=3306
innodb_file_per_table=ON
pid-file=/var/run/mariadb/mariadb.pid

server-id=10
log_bin=/var/log/mariadb/mariadb-bin.log
binlog_do_db=mabase
```

- **server-id** : l'identifiant du serveur pour la réplication. Il doit être différent sur chaque noeud.
- **log_bin** : Le fichier de log utilisé pour suivre l'activité de la réplication
- **binlog_do_db** : La base de données concernée par le processus de réplication
- Sur le noeud 2 :

Fichier /etc/my.cnf sur le noeud 2

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
port=3306
innodb_file_per_table=ON
pid-file=/var/run/mariadb/mariadb.pid

server-id=11
log_bin=/var/log/mariadb/mariadb-bin.log
binlog_do_db=mabase
```

Relancer les services sur les deux noeuds pour prendre en compte les modifications :

```
systemctl restart mariadb
```

- Vérification

Sur les différents noeuds, vérifier l'état de la réplication :

```
MariaDB > show master status;
+--------------------+----------+--------------+------------------+
| File               | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+--------------------+----------+--------------+------------------+
| mariadb-bin.000001 |      245 | mabase       |                  |
+--------------------+----------+--------------+------------------+
1 row in set (0.00 sec)

MariaDB > show slave status;
Empty set (0.00 sec)
```

Basculer les serveurs en multi-maîtres :

- Sur le noeud 1 :
  - Arrêter le processus de synchronisation
  - Ajouter le noeud 2 comme maître
  - Redémarrer le processus de synchronisation

Les valeurs de MASTER_LOG_POS et MASTER_LOG_FILE sont à récupérer sur le noeud 2.

```
mariadb > stop slave;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mariadb > CHANGE MASTER TO MASTER_HOST = 'mirroir2.local.lan', MASTER_PORT = 3306,
MASTER_USER = 'mirroir', MASTER_PASSWORD = 'm!rro!r', MASTER_LOG_FILE = 'mariadb-
bin.000001', MASTER_LOG_POS = 245;
Query OK, 0 rows affected, 2 warnings (0.25 sec)

mariadb > start slave;
Query OK, 0 rows affected (0.03 sec)

MariaDB [(none)]> show slave status;
+---------------------------------+-------------+-------------+-------------
+---------------+--------------------+---------------------+--------------------------
+---------------+----------------------+------------------+-------------------
+-----------------+---------------------+--------------------+------------------------
+-------------------------+-----------------------------+------------+------------
+-------------+--------------------+----------------+----------------
+----------------+--------------+--------------------+-------------------
```

```
+-------------------+----------------+-------------------+---------------
+----------------------+------------------------------+--------------
+--------------+---------------+---------------+---------------------------
+------------------+
| Slave_IO_State                   | Master_Host | Master_User | Master_Port |
Connect_Retry | Master_Log_File    | Read_Master_Log_Pos | Relay_Log_File           |
Relay_Log_Pos | Relay_Master_Log_File | Slave_IO_Running | Slave_SQL_Running |
Replicate_Do_DB | Replicate_Ignore_DB | Replicate_Do_Table | Replicate_Ignore_Table |
Replicate_Wild_Do_Table | Replicate_Wild_Ignore_Table | Last_Errno | Last_Error |
Skip_Counter | Exec_Master_Log_Pos | Relay_Log_Space | Until_Condition |
Until_Log_File | Until_Log_Pos | Master_SSL_Allowed | Master_SSL_CA_File |
Master_SSL_CA_Path | Master_SSL_Cert | Master_SSL_Cipher | Master_SSL_Key |
Seconds_Behind_Master | Master_SSL_Verify_Server_Cert | Last_IO_Errno | Last_IO_Error
| Last_SQL_Errno | Last_SQL_Error | Replicate_Ignore_Server_Ids | Master_Server_Id |
+---------------------------------+-------------+-------------+-------------
+--------------+-------------------+--------------------+-------------------------
+--------------+----------------------+------------------+------------------
+----------------+----------------------+--------------------+-----------------------
+-------------------------+-----------------------------+------------+------------
+--------------+---------------------+-----------------+-----------------
+----------------+---------------+--------------------+--------------------
+--------------------+-----------------+-------------------+---------------
+----------------------+------------------------------+--------------
+--------------+---------------+---------------+---------------------------
+------------------+
| Waiting for master to send event | mirroir2    | mirroir     |        3306 |
60 | mariadb-bin.000001 |                509 | mariadb-relay-bin.000002 |
531 | mariadb-bin.000001    | Yes              | Yes               |                 |
|                    |                       |                          |
|          0 |           |           0 |                509 |             827 |
None            |               |            0 | No                |
|                    |               |                  |               |
0 | No                           |            0 |              |             0 |
|                             |              11 |
+---------------------------------+-------------+-------------+-------------
+--------------+-------------------+--------------------+-------------------------
+--------------+----------------------+------------------+------------------
+----------------+----------------------+--------------------+-----------------------
+-------------------------+-----------------------------+------------+------------
+--------------+---------------------+-----------------+-----------------
+----------------+---------------+--------------------+--------------------
+--------------------+-----------------+-------------------+---------------
+----------------------+------------------------------+--------------
+--------------+---------------+---------------+---------------------------
+------------------+
1 row in set (0.00 sec)
```

La requête **CHANGE MASTER [nom_connexion] TO** créé ou modifie une connexion à un serveur

maître. Elle change les paramètres que le serveur esclave utilise pour se connecter et communiquer avec le serveur maître durant les réplications. Sans spécifier de nom de connexion, la connexion par défaut est modifiée.

Les valeurs **MASTER_LOG_FILE** et **MASTER_LOG_POS** sont les coordonnées à partir desquelles l'esclave doit commencer à lire depuis le maître lors de la prochaine réplication.

Sur le noeud 2 :

```
mariadb > stop slave;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mariadb > CHANGE MASTER TO MASTER_HOST = 'mirroir1.local.lan', MASTER_PORT = 3306,
MASTER_USER = 'mirroir', MASTER_PASSWORD = 'm!rro!r', MASTER_LOG_FILE = 'mariadb-
bin.000001', MASTER_LOG_POS = 245;
Query OK, 0 rows affected, 2 warnings (0.07 sec)

mariadb > start slave;
Query OK, 0 rows affected (0.04 sec)
```

16.5. Tests de bon fonctionnement

Pour tester le bon fonctionnement du cluster, une table va être créée sur le noeud 1. Après vérification sur le noeud 2 que la table a bien été répliquée, des données y seront rajoutées. La présence des données sur le noeud 1 permettra de valider la réplication multi-maître.

- Sur le noeud 1 :

```
[root@mirroir1 ~]# mysql -u root -p mabase
Enter password:

mariadb > create table table1( id int(11) primary key auto_increment, nom varchar(30
));
Query OK, 0 rows affected (0.22 sec)

mariadb > show tables in mabase;
+------------------+
| Tables_in_mabase |
+------------------+
| table1 |
+------------------+
1 row in set (0.01 sec)
```

- Vérifier le noeud 2 la présence de la table et ajouter des données :

```
[root@mirroir2 ~]# mysql -u root -p mabase
Enter password:

mariadb > show tables in mabase;
+--------------------+
| Tables_in_test_rep |
+--------------------+
| table1 |
+--------------------+
1 row in set (0.00 sec)

mariadb > insert into table1 ( nom ) values ('antoine'), ('xavier'), ('patrick') ;
Query OK, 3 rows affected (0.05 sec)
Records: 3 Duplicates: 0 Warnings: 0

mariadb > select * from table1;
+----+----------+
| id | fullname |
+----+----------+
| 1 | antoine |
| 2 | xavier |
| 3 | patrick |
+----+----------+
3 rows in set (0.00 sec)

mariadb > commit;
Query OK, 0 rows affected (0.00 sec)
```

- Retour sur le noeud 1 :

```
[root@DB1 ~]# mysql -u root -p mabase
Enter password:

mariadb > select * from table1;
+----+----------+
| id | fullname |
+----+----------+
| 1 | antoine |
| 2 | xavier |
| 3 | patrick |
+----+----------+
3 rows in set (0.01 sec)
```

Chapitre 17. La mise en cluster sous Linux

> La **haute disponibilité** est un terme souvent utilisé en informatique, à propos d'architecture de système ou d'un service pour désigner le fait que cette architecture ou ce service a un taux de disponibilité convenable.
>
> — wikipedia

Cette disponibilité est une mesure de performance exprimée en pourcentage obtenue par le ration **Durée de fonctionnement** / **Durée total de fonctionnement souhaité**.

Table 107. Taux de disponibilité

Taux	Temps d'arrêt annuel
90%	876 heures
95%	438 heures
99%	87 heures et 36 minutes
99,9%	8 heures 45 minutes 36 secondes
99,99%	52 minutes, 33 secondes
99,999%	5 minutes, 15 secondes
99,9999%	31,68 secondes

La "Haute Disponibilité" (en anglais "**High Availability**" - **HA**) concerne donc toutes les mesures prises visant à garantir la plus haute disponibilité d'un service, c'est à dire son bon fonctionnement 24H/24H.

17.1. Généralités

Un **cluster** est une ***"grappe d'ordinateurs"***, un groupe de deux machines ou plus.

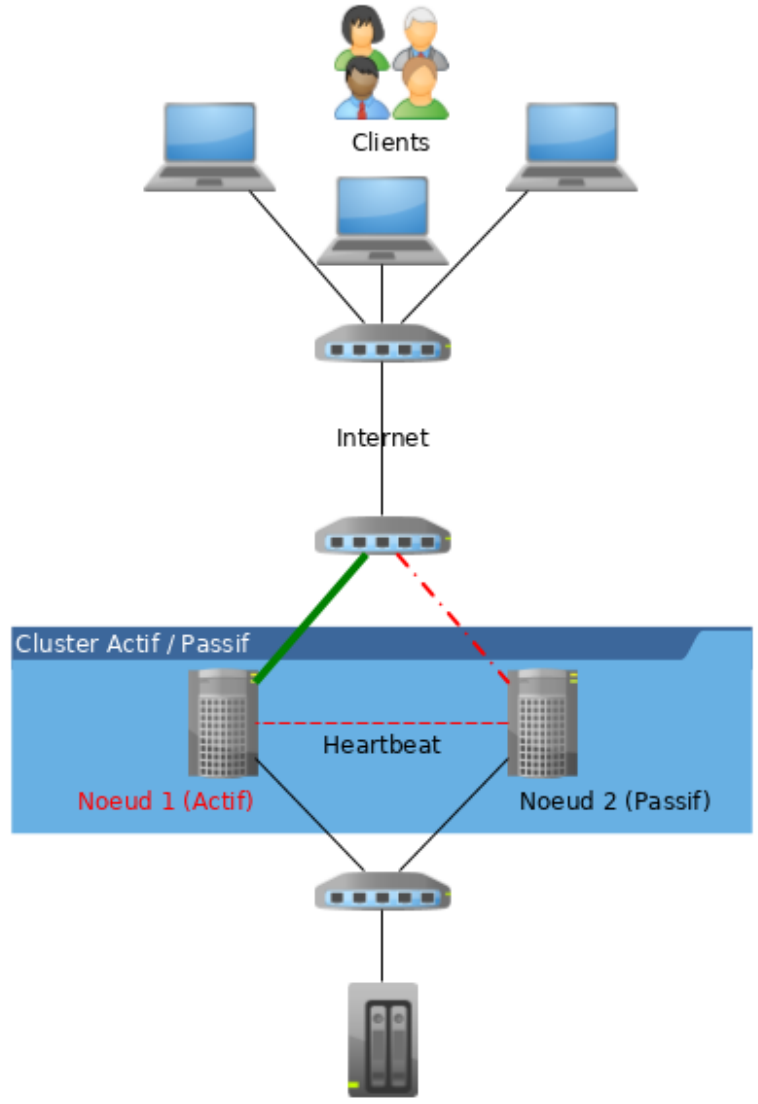


Figure 71. Schéma d'un cluster actif/passif

Un cluster permet :

- le **calcul distribué** utilisant la puissance de calcul de l'ensemble des noeuds,
- la **haute disponibilité**, la continuité de service et le basculement automatique des services en cas de panne d'un noeud.

Les types de services

- Services Actif / Passif

L'installation d'un cluster avec deux noeuds actif/passif utilisant Pacemaker et DRBD est une solution à faible coût pour de nombreux cas de figure nécessitant un système de haute disponibilité.

- Services N+1

Grâce à plusieurs noeuds, Pacemaker peut réduire le coût matériel en permettant à plusieurs

clusters actif/passif de se combiner et de partager un noeud de secours.

- Services N TO N

Avec un stockage partagé, chaque noeud peut potentiellement être utilisé pour la tolérance de panne. Pacemaker peut également faire fonctionner plusieurs copies des services pour répartir la charge de travail.

- Services Sites Distants

Pacemaker inclus des améliorations pour simplifier la création de clusters sur plusieurs sites.

Les VIP

La **VIP** est une adresse IP virtuelle. Cette adresse est attribuée à un cluster Actif/Passif. Elle est assignée au noeud du cluster qui est actif. En cas de rupture de service, la VIP est désactivée sur le noeud en échec et activée sur le noeud prenant le relais : c'est la **tolérance de panne** (ou **failover**).

Les clients s'adressent toujours au cluster en utilisant la VIP, ce qui rend transparent à leurs yeux les bascules de serveur actif.

Le split-brain

Le ***split-brain*** est le risque principal que peut rencontrer un cluster. Cet état arrive lorsque plusieurs nœuds d'un cluster pense son voisin inactif. Le noeud tente alors de démarrer le service redondant et plusieurs noeuds fournissent alors le même service, ce qui peut entraîner des effets de bords génant (VIP en double sur le réseau, accès concurrent aux données, etc.).

Les solutions techniques possibles pour éviter ce problème sont :

- de séparer le trafic réseau public du trafic réseau du cluster,
- d'utiliser un agrégat de lien (bonding) réseau.

17.2. La gestion des ressources : Pacemaker

Pacemaker est la partie logicielle du cluster qui gère ses ressources (les VIP, les services, les données). Il est chargé de faire démarrer, arrêter et superviser les ressources du cluster. Il est le garant de la haute disponibilité des noeuds.

Pacemaker utilise la couche de message fournit par **corosync** (par défaut) ou par **Heartbeat**.

Pacemaker est composé de **5 composants clés** :

- La base d'information du cluster (Cluster Information Base - **CIB**)
- Le démon de gestion des ressources du cluster (Cluster Resource Management daemon - **CRMd**)
- Le démon de gestion des ressources locales (Local Resource Management daemon - **LRMd**)

- Le moteur de stratégie (Policy Engine - **PEngine** ou **PE**)
- Le démon de protection (Fencing daemon - **STONITHd**)

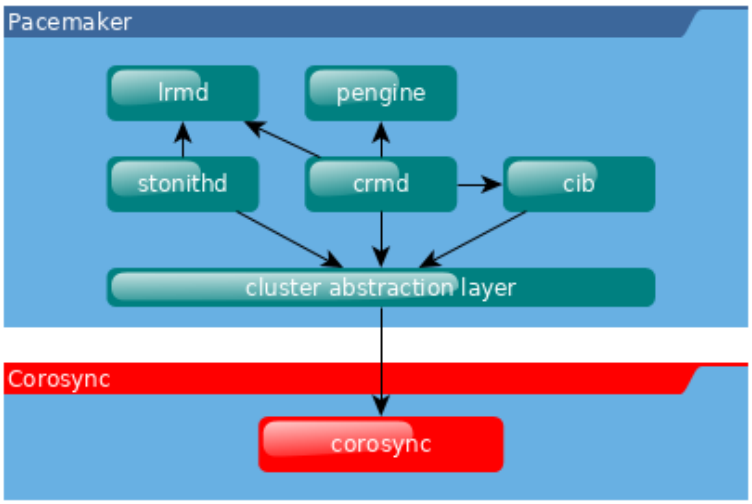


Figure 72. Les composants logiciels du cluster

La **CIB** représente la configuration du cluster et l'état en cours de toutes les ressources du cluster. Le contenu de la CIB est automatiquement synchronisé sur le cluster complet et est utilisé par le **PEngine** pour calculer l'état idéal du cluster et comment il doit être obtenu.

La liste des instructions est alors fournie au controlleur désigné (**Designated Controller** - **DC)**. Pacemaker centralise toutes les décisions du cluster en élisant en tant que maître une des instances de **CRMd**.

Le **DC** execute dans l'ordre requis les instructions du **PEngine** en les transmettant soit au **LRMd** local ou aux **CRMd** des autres noeuds via **Corosync** ou **Heartbeat**.

Dans certains cas, il est nécessaire de stopper des noeuds pour protéger les données partagées ou permettre leur récupération. Pour cela, Pacemaker est livré avec **STONITHd**.

Stonith

Stonith est un composant de Pacemaker. Il est l'acronyme de **Shoot-The-Other-Node-In-The-Head**, une pratique recommandée pour que le noeud dysfonctionnant soit le plus rapidement isolé (éteint ou au moins déconnecté des ressources partagées), ce qui évite la corruption des données.

Un nœud qui ne répond pas ne signifie pas qu'il n'accède plus aux données. La seule solution pour s'assurer qu'un nœud n'accède plus aux données avant de donner la main à un autre nœud et d'utiliser STONITH, qui va soit éteindre, soit redémarrer le serveur en échec.

STONITH a également un rôle à jouer dans le cas où un service en cluster n'arrive pas à s'arrêter. Dans ce cas, Pacemaker utilise STONITH pour forcer le nœud entier à s'arrêter.

La gestion du quorum

Le **quorum** représente le nombre de **noeuds minimum** en fonctionnement qui permettent de valider une décision, comme décider quel noeud de secours doit prendre le relais lorsqu'un des noeuds est en erreur. Pacemaker, par défaut, exige que plus de la moitié des nœuds soient en ligne.

Lorsque des problèmes de communication séparent un cluster en plusieurs groupes de noeuds, le quorum est alors utilisé pour prévenir le démarrage des ressources sur plus de noeuds que prévu. Un cluster a le quorum lorsque plus de la moitié de tous les noeuds étant connus en ligne se retrouvent dans son groupe (noeuds_actifs_groupe > noeuds_total_actifs / 2)

La décision par défaut lorsque le quorum n'est pas atteint est de désactiver toutes les ressources.

```
Exemple de cas :
```

- Sur un cluster à **deux noeuds**, le quorum ne pouvant **jamais être atteint** en cas de panne d'un noeud, il doit donc être ignoré sous peine que le cluster complet soit stoppé.
- Si un cluster à 5 noeuds est coupé en 2 groupes de 3 et 2 noeuds, le groupe de 3 noeuds disposera du quorum et continuera à gérer les ressources.
- Si un cluster à 6 noeuds est coupé en 2 groupes de 3 noeuds alors aucun groupe ne disposera du quorum. Dans ce cas, le comportement par défaut de pacemaker est de stopper toutes les ressources pour éviter la corruption de données.

17.3. Communication du clusters

Pacemaker s'appuie au choix sur **Corosync** ou **Heartbeat** (du projet linux-ha) pour assurer la communication entre les nœuds et la gestion du cluster.

Corosync

Corosync Cluster Engine est une couche de messagerie entre les membres du cluster et intègre des fonctionnalités additionnelles pour l'implémentation de la haute disponibilité au sein des applications. Le projet Corosync est dérivé du projet OpenAIS.

La communication entre les noeuds se fait en mode Client/Serveur via le protocole UDP.

Il permet de gérer des cluster composés de plus de 16 noeuds dans les modes Actif/Passif ou Actif/Actif.

Heartbeat

La technologie Heartbeat est plus limitée que Corosync. Il n'est pas possible de créer un cluster de plus de 2 noeuds et les règles de gestion sont moins abouties que son concurrent.

Le choix de pacemaker/corosync aujourd'hui semble plus opportun, c'est le choix par défaut des distributions RedHat, Debian et Ubuntu.

17.4. La gestion des données

Le raid en réseau drdb

DRDB est un driver de périphérique de type **bloc** qui permet la mise en oeuvre de **RAID 1** (mirroir) **via le réseau**.

La mise en oeuvre de DRDB peut être intéressante lorsque des technologies NAS ou SAN ne sont pas disponibles mais que les données doivent tout de même être synchronisées.

GFS2

17.5. Ateliers Dirigés Cluster

Ces ateliers dirigés s'appuient sur une infrastructure à deux noeuds :

- Noeud 1 :
  - Nom de machine : node1.formatux.fr
  - Disque système 10G ou plus
  - Disque 2 : 1G (pour DRDB)
  - Distribution CentOS7 à jour
  - Adresse IP publique : 192.168.1.100
  - Adresse IP privée : 192.168.100.100
- Noeud 2 :
  - Nom de machine : node2.formatux.fr
  - Disque système 10G ou plus
  - Disque 2 : 1G (pour DRDB)
  - Distribution CentOS7 à jour
  - Adresse IP publique : 192.168.1.101
  - Adresse IP privée : 192.168.100.101

Préparation des hôtes

Sur chaque hôte sera configuré :

- Le fichier /etc/hosts pour assurer la résolution de nom dès le démarrage de la machine et en toute indépendance du serveur DNS

- Le service Corosync
- Le service Pacemaker

A l'issue de cette configuration, les mises en oeuvre de clusters avec VIP, avec services puis avec DRDB pourront être envisagés.

Configuration du fichier /etc/hosts

Le fichier */etc/hosts* doit permettre la résolution des adresses IP publiques et des adresses IP privées.

Le fichier /etc/hosts ressemblera à l'exemple ci-dessous sur l'ensemble des noeuds du cluster :

Fichier /etc/hosts

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.1.100           node1.formatux.fr       node1
192.168.1.101           node2.formatux.fr       node2
192.168.100.100         node1-priv.formatux.fr  node1-priv
192.168.100.101         node2-priv.formatux.fr  node2-priv
```

Depuis le noeud 1, le noeud 2 peut être joint soit par son nom publique ou son nom privé :

```
$ ping -c 4 node2
$ ping -c 4 node2-priv
```

Informations sur les paquets Pacemaker et corosync

Quelques informations sur le paquet pacemaker :

```
$ yum info pacemaker
Paquets disponibles
Nom                 : pacemaker
Résumé              : Scalable High-Availability cluster resource manager
URL                 : http://www.clusterlabs.org
Licence             : GPLv2+ and LGPLv2+
Description         : Pacemaker is an advanced, scalable High-Availability
                    : cluster resource manager for Corosync, CMAN and/or
                    : Linux-HA.
                    :
                    : It supports more than 16 node clusters with significant
                    : capabilities for managing resources and dependencies.
                    :
                    : It will run scripts at initialization, when machines go up
                    : or down, when related resources fail and can be configured
                    : to periodically check resource health.
```

Grâce à la commande repoquery (voir le chapitre commandes avancées), il est possible de connaître les dépendances du paquet pacemaker :

```
$ sudo yum install yum-utils
$ repoquery --requires pacemaker
corosync
pacemaker-cli = 1.1.15-11.el7_3.5
resource-agents
...
```

L'installation de pacemaker installera donc automatiquement corosync et une interface CLI pour pacemaker.

Quelques informations sur le paquet corosync :

```
yum info corosync
Paquets disponibles
Nom                 : corosync
Résumé              : The Corosync Cluster Engine and Application Programming
                    : Interfaces
URL                 : http://corosync.github.io/corosync/
Licence             : BSD
Description         : This package contains the Corosync Cluster Engine
                    : Executive, several default APIs and libraries, default
                    : configuration files, and an init script.
```

Installation des logiciels corosync et pacemaker

```
# yum install pacemaker
```

- Ouverture du firewall :

```
# firewall-cmd --permanent --add-service=high-availability
success
# firewall-cmd --reload
success
```

Les services peuvent maintenant être activés pour le prochain démarrage :

```
# systemctl enable corosync
# systemctl enable pacemaker
```

Gestion du cluster

Le paquet **pcs** fournit des outils la gestion du cluster. La commande **pcs** est une interface en ligne de commande pour gérer la **stack** de haute disponibilité de Pacemaker.

La configuration du cluster pourrait éventuellement être faite à la main, mais le paquet pcs facilite grandement la gestion (création, configuration et dépannage) d'un cluster !

Il existe des alternatives à pcs.

Installation de pcs

Installer le paquet sur l'ensemble des noeuds et activer le démon :

```
# yum install pcs
# systemctl start pcsd
# systemctl enable pcsd
```

L'installation du paquet a créé un utilisateur **hacluster** avec un mot de passe vide. Pour effectuer les tâches de synchronisation des fichiers de configuration de corosync ou redémarrer les noeuds distants, un mot de passe doit lui être attribué.

Sur tous les noeuds, attribuer un mot de passe identique à l'utilisateur hacluster :

```
# echo "mdphacluster" | passwd --stdin hacluster
```

Administration du cluster

Depuis n'importe quel noeud, il est possible de s'authentifier comme utilisateur hacluster sur l'ensemble des noeuds, puis d'utiliser les commandes pcs sur ceux-ci :

```
pcs cluster auth node1-priv node2-priv
Username: hacluster
Password:
node2-priv: Authorized
node1-priv: Authorized
```

Depuis le noeud sur lequel pcs est authentifié, lancer la configuration du cluster :

```
pcs cluster setup --name moncluster node1-priv node2-priv
```

Le cluster peut maintenant être démarré :

```
pcs cluster start --all
node1-priv: Starting Cluster...
node2-priv: Starting Cluster...
```

La commande **pcs cluster setup** prend en charge le problème du quorum des clusters à deux noeuds. Un tel cluster fonctionnera donc correctement en cas de panne d'un des deux noeuds. Si vous configurez manuellement corosync ou utilisez un autre shell de gestion du cluster, vous devrez configurer corosync correctement par vous-même.

Vérifications

La commande **pcs cluster setup** a eu pour effet de générer un fichier **/etc/corosync/corosync.conf** :

```
totem {
    version: 2
    secauth: off
    cluster_name: moncluster
    transport: udpu
}

nodelist {
    node {
        ring0_addr: node1-priv
        nodeid: 1
    }

    node {
        ring0_addr: node2-priv
        nodeid: 2
    }
}

quorum {
    provider: corosync_votequorum
    two_node: 1
}

logging {
    to_logfile: yes
    logfile: /var/log/cluster/corosync.log
    to_syslog: yes
}
```

Des exemples de fichiers de configuration plus complets se trouvent sous /etc/corosync.

La commande **pcs status** renseigne sur l'état global du cluster :

```
pcs status
Cluster name: moncluster
WARNING: no stonith devices and stonith-enabled is not false
Stack: corosync
Current DC: node1-priv (version 1.1.15-11.el7_3.5-e174ec8) - partition with quorum
Last updated: Wed Jul  5 18:22:47 2017      Last change: Wed Jul  5 17:56:27 2017 by
hacluster via crmd on node2-priv

2 nodes and 0 resources configured

Online: [ node1-priv node2-priv ]

No resources


Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

Comme vous pouvez le constater dans le retour de la commande, le processus **stonith** est activé mais non configuré :

```
WARNING: no stonith devices and stonith-enabled is not false
```

Dans un premier temps, nous allons désactiver stonith en attendant d'apprendre à le configurer :

```
pcs property set stonith-enabled=false
```

Attention à ne pas laisser stonith désactivé sur un environnement de production !!!

La commande **pcs status corosync** nous renseigne sur l'état des noeuds corosync :

```
pcs status corosync

Membership information
----------------------
    Nodeid      Votes Name
         1          1 node1-priv (local)
         2          1 node2-priv
```

Les outils standards peuvent également être utilisés :

- La commande **crm_mon** renvoie une configuration correcte du cluster :

```
# crm_mon -1
Stack: corosync
Current DC: node2-priv (version 1.1.15-11.el7_3.5-e174ec8) - partition with quorum
Last updated: Wed Jul  5 15:57:18 2017      Last change: Wed Jul  5 16:08:39 2017 by
hacluster via crmd on node2-priv

2 nodes and 0 resources configured

Online: [ node1-priv node2-priv ]

No active resources
```

- La commande **corosync-cfgtool** verifie si la configuration est correcte et si la communication avec le cluster se fait bien:

```
$ corosync-cfgtool -s

Printing ring status.
Local node ID 1
RING ID 0
        id      = 192.168.122.100
        status  = ring 0 active with no faults
```

- La commande **corosync-cmapctl** est un outil pour accéder à la base d'objets. Elle permet, par exemple, de vérifier le status des noeuds membres du cluster :

```
corosync-cmapctl  | grep members
runtime.totem.pg.mrp.srp.members.1.config_version (u64) = 0
runtime.totem.pg.mrp.srp.members.1.ip (str) = r(0) ip(192.168.100.100)
runtime.totem.pg.mrp.srp.members.1.join_count (u32) = 1
runtime.totem.pg.mrp.srp.members.1.status (str) = joined
runtime.totem.pg.mrp.srp.members.2.config_version (u64) = 0
runtime.totem.pg.mrp.srp.members.2.ip (str) = r(0) ip(192.168.100.101)
runtime.totem.pg.mrp.srp.members.2.join_count (u32) = 2
runtime.totem.pg.mrp.srp.members.2.status (str) = joined
```

TD Configuration d'une VIP

La première ressource que nous allons créer sur notre cluster est une VIP.

Cette VIP, correspondant à l'adresse IP utilisée par les clients pour accéder aux futurs services du cluster, sera attribuée à un des noeuds, puis, en cas de défaillance, le cluster basculera cette ressource d'un noeud à l'autre pour assurer la continuité du service.

```
pcs resource create monclusterVIP ocf:heartbeat:IPaddr1 ip=192.168.1.99 cidr_netmask
=24 op monitor interval=30s
```

L'argument **ocf:heartbeat:IPaddr2** est composé de 3 champs qui fournissent à pacemaker :

1. le standard qui est suivi par le script de ressource (ici ocf),
2. l'espace de nom du script,
3. le nom du script de la ressource

Les ressources standards disponibles sont fournies par la commande **pcs resource standards** :

```
pcs resource standards
ocf
lsb
service
systemd
stonith
```

Le résultat est l'ajout d'une adresse IP virtuelle sur un des noeuds :

```
pcs status
Cluster name: moncluster
Stack: corosync
Current DC: node1-priv (version 1.1.15-11.el7_3.5-e174ec8) - partition with quorum
Last updated: Wed Jul  5 18:30:54 2017      Last change: Wed Jul  5 18:29:50 2017 by
root via cibadmin on node1-priv

2 nodes and 1 resource configured

Online: [ node1-priv node2-priv ]

Full list of resources:

 monclusterVIP  (ocf::heartbeat:IPaddr2):   Started node1-priv

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

Dans le cas présent, la VIP est active sur le noeud 1, ce qui est vérifiable avec la commande **ip** :

```
node1 # ip add sh enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen
1000
    link/ether 08:00:27:83:83:11 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.100/24 brd 192.168.1.255 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet 192.168.1.99/24 brd 192.168.1.255 scope global secondary enp0s3
       valid_lft forever preferred_lft forever
```

Tests de bascule

- Depuis un poste du réseau, lancer la commande ping sur la VIP :

```
ping 192.168.1.99
```

- Redémarrer le noeud 1 :

```
node1 # reboot
```

- Sur le noeud 2 :

Constater que le noeud 1 est offline et que la bascule de la ressource s'est correctement effectuée :

```
# pcs status
Cluster name: moncluster
Stack: corosync
Current DC: node2-priv (version 1.1.15-11.el7_3.5-e174ec8) - partition with quorum
Last updated: Wed Jul  5 18:34:19 2017      Last change: Wed Jul  5 18:29:51 2017 by
root via cibadmin on node1-priv

2 nodes and 1 resource configured

Online: [ node2-priv ]
OFFLINE: [ node1-priv ]

Full list of resources:

 monclusterVIP  (ocf::heartbeat:IPaddr2):   Started node2-priv

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

- Vérifier avec la commande ip sur le noeud 2 :

```
node2 # ip add show enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen
1000
    link/ether 08:00:27:55:3d:ca brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.101/24 brd 192.168.1.255 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet 192.168.1.99/24 brd 192.168.1.255 scope global secondary enp0s3
       valid_lft forever preferred_lft forever
```

- Depuis le poste du réseau, constater qu'aucun ping n'a été perdu durant la bascule.
- Noter qu'une fois le noeud 1 redémarré, le cluster retrouve son état normal, la ressource n'est toutefois pas rebasculée vers le noeud 1 : elle reste sur le noeud 2.

```
node2 # pcs status
Cluster name: moncluster
Stack: corosync
Current DC: node2-priv (version 1.1.15-11.el7_3.5-e174ec8) - partition with quorum
Last updated: Wed Jul  5 18:37:23 2017      Last change: Wed Jul  5 18:29:51 2017 by
root via cibadmin on node1-priv

2 nodes and 1 resource configured

Online: [ node1-priv node2-priv ]

Full list of resources:

 monclusterVIP  (ocf::heartbeat:IPaddr2):   Started node2-priv

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

TD Configuration d'un service Apache

L'objectif de cet atelier est d'installer le service Apache sur les deux noeuds de notre cluster. Ce service ne sera démarré que sur le noeud actif est basculera de noeud en même temps que la VIP en cas de panne du noeud actif.

Installation et configuration d'Apache

L'installation doit être faîte sur les 2 noeuds :

```
# yum install -y httpd
# firewall-cmd --permanent --add-service=http
# firewall-cmd --reload
```

Une page HTML contenant le nom du serveur sera affichée par défaut :

```
echo "<html><body>Noeud $(hostname -f)</body></html>" > "/var/www/html/index.html"
```

L'agent de ressource de Pacemaker utilise la page /server-status (voir chapitre apache) pour déterminer son état de santé. Il doit être activé en créant le fichier ***/etc/httpd/conf.d/status.conf*** :

Activation du server-status Apache

```
<Location /server-status>
  SetHandler server-status
  Require local
</Location>
```

Le service ne doit pas être démarré ni activé. La main est totalement laissé à Pacemaker, qui au démarrage du serveur prendra en charge le démarrage du service sur l'un des noeuds !

Configuration du service dans le cluster

Pour créer une ressource que nous appelerons "SiteSeb", nous allons faire appel au script apache de la ressource OCF et dans l'espace de nom de heartbeat.

```
# pcs resource create SiteWeb ocf:heartbeat:apache
configfile=/etc/httpd/conf/httpd.conf statusurl="http://localhost/server-status" op
monitor interval=1min
```

Le cluster va vérifier l'état de santé d'Apache toutes les minutes (**op monitor interval=1min**).

Enfin, pour que le service Apache soit démarré sur le même noeud que l'adresse VIP, il faut ajouter une contrainte au cluster :

```
# pcs constraint colocation add SiteWeb with monclusterVIP INFINITY
```

Le service Apache peut également être configuré pour démarrer après la VIP, ce qui peut être intéressant si les VHosts Apache sont configurés pour écouter l'adresse de la VIP (**Listen 192.168.1.99**) :

```
pcs constraint order monclusterVIP then SiteWeb
```

La page Apache affichée est gérée par le noeud 1 :

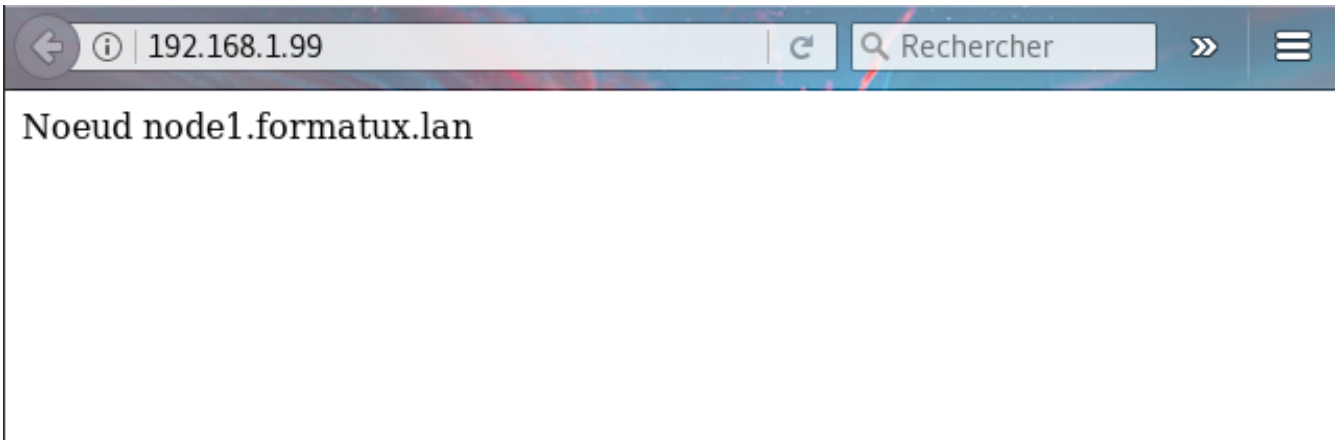


Figure 73. Le service Apache pris en charge par le noeud 1

Après le redémarrage du noeud 1, le service est pris en charge par le noeud 2 :

```
pcs status
Cluster name: moncluster
Stack: corosync
Current DC: node2-priv (version 1.1.15-11.el7_3.5-e174ec8) - partition with quorum
Last updated: Thu Jul  6 15:31:45 2017      Last change: Thu Jul  6 15:13:56 2017 by
root via cibadmin on node1-priv

2 nodes and 2 resources configured

Online: [ node2-priv ]
OFFLINE: [ node1-priv ]

Full list of resources:

 moncluster (ocf::heartbeat:IPaddr2):   Started node2-priv
 SiteWeb    (ocf::heartbeat:apache):    Started node2-priv

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

Et la page affichée provient du noeud 2 :

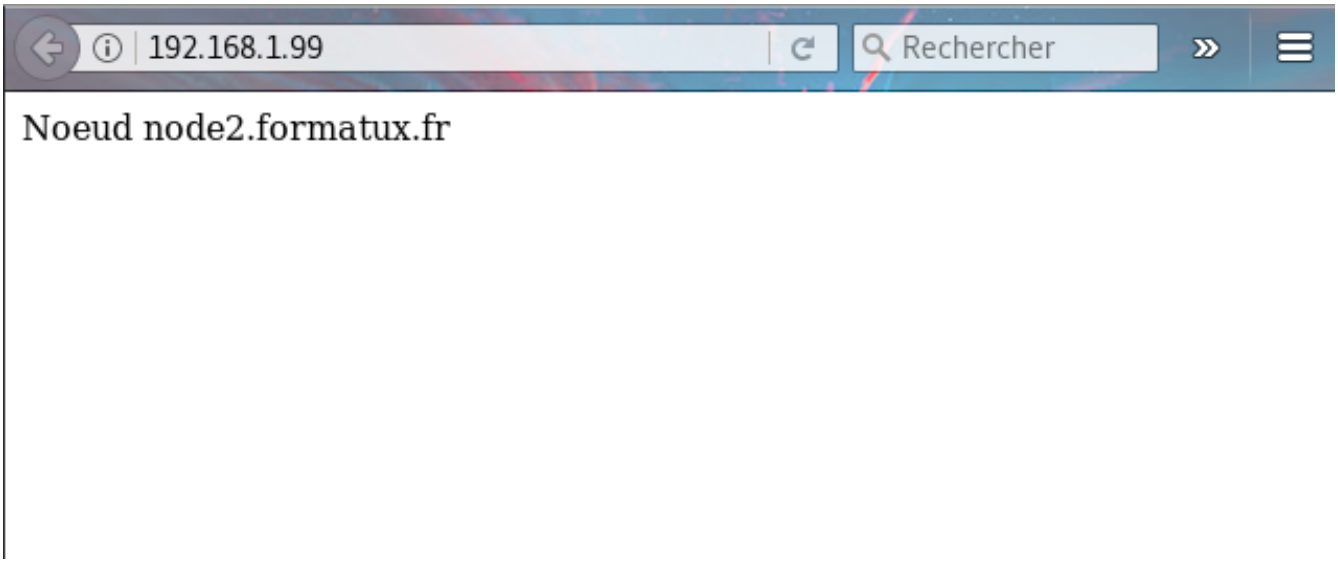


Figure 74. Le service Apache après bascule sur le noeud 2

17.6. Répliquer les données avec DRDB

DRDB permet la synchronisation, en l'absence de stockage partagé type NAS ou SAN, de données entre deux serveurs.

DRDB peut être vu comme un mirroir Réseau de type RAID 1.

Installation

Le module DRBD est inclu dans le noyau Linux, mais il est nécessaire d'installer des outils pour le contrôler.

Les outils sont disponibles depuis le dépôt **el** :

```
# yum install http://www.elrepo.org/elrepo-release-7.0-2.el7.elrepo.noarch.rpm
# yum install kmod-drbd84 drbd84-utils
```

Rendre SELinux permissif uniquement pour DRBD :

```
# yum install policycoreutils-python
# semanage permissive -a drbd_t
```

et configurer le pare-feu :

```
firewall-cmd --permanent --add-port="7789/tcp"
firewall-cmd --reload
```

Configuration des disques

A l'aide de **cfdisk**, créer une partition sur le disque (ou éventuellement la partition) dédiée sur les 2 noeuds :

```
# cfdisk /dev/sdb
```

Créer le fichier de configuration /etc/drbd.d/siteweb.res sur les 2 noeuds :

```
# vim /etc/drbd.d/siteweb.res
resource siteweb {
 protocol C;
 meta-disk internal;
 device /dev/drbd1;
 syncer {
  verify-alg sha1;
 }
 net {
  allow-two-primaries;
 }
 on node1 {
  disk   /dev/sdb1;
  address  192.168.100.100:7789;
 }
 on node2 {
  disk   /dev/sdb1;
  address  192.168.100.101:7789;
 }
}
```

Initialiser et activer le disque sur les 2 noeuds :

```
# drbdadm create-md siteweb
initializing activity log
NOT initializing bitmap
Writing meta data...
New drbd meta data block successfully created.
success

# modprobe drbd
# drbdadm up siteweb
```

Comme les disques ne sont pas encore initialisés, il faut dire à drbd lequel des deux noeuds est la référence (le primary). Depuis le noeud 1 :

```
drbdadm primary --force website
```

La synchronisation des disques peut être suivie avec la commande :

```
cat /proc/drbd
version: 8.4.9-1 (api:1/proto:86-101)
GIT-hash: 9976da086367a2476503ef7f6b13d4567327a280 build by akemi@Build64R7, 2016-12-
04 01:08:48

 1: cs:SyncSource ro:Primary/Secondary ds:UpToDate/Inconsistent C r-----
    ns:1120 nr:0 dw:0 dr:2032 al:16 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:1047356
    [>....................] sync'ed:  0.4% (1047356/1048476)K
    finish: 0:14:32 speed: 1,120 (1,120) K/sec
```

Lorsque la synchronisation est finie :

```
cat /proc/drbd
version: 8.4.9-1 (api:1/proto:86-101)
GIT-hash: 9976da086367a2476503ef7f6b13d4567327a280 build by akemi@Build64R7, 2016-12-
04 01:08:48

 1: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C r-----
    ns:0 nr:1065560 dw:1065560 dr:0 al:8 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:0
```

Le formatage des disques en ext4 ne permettrait pas de monter le système de fichiers en même temps sur les deux noeuds.

Pour monter le même système de fichiers sur deux noeuds en même temps, il faut utiliser des systèmes de fichiers spécialisés, type ocfs ou gfs2, ainsi qu'un gestionnaire de verroux distribués **dlm (Distributed Lock Manager)** qu'il faut installer sur les 2 noeuds :

```
yum install gfs2-utils dlm
```

La partition peut être formatée en gfs2 :

```
mkfs.gfs2 -p lock_dlm -j 2 -t moncluster:siteweb /dev/drbd1
It appears to contain an existing filesystem (ext4)
This will destroy any data on /dev/drbd1
Are you sure you want to proceed? [y/n]y
Discarding device contents (may take a while on large devices): Done
Adding journals: Done
Building resource groups: Done
Creating quota file: Done
Writing superblock and syncing: Done
Device:                    /dev/drbd1
Block size:                4096
Device size:               1,00 GB (262119 blocks)
Filesystem size:           1,00 GB (262117 blocks)
Journals:                  2
Resource groups:           5
Locking protocol:          "lock_dlm"
Lock table:                "moncluster:siteweb"
UUID:                      23405f79-cc0c-3dfa-8554-a0cb6dce40ad
```

Table 108. Options de la commande mkfs.gfs2

Options	Commentaires
-j n	Le nombre de journaux à créer. Un journal est nécessaire par noeuds qui monteront simultanément le système de fichiers. Il est préférable de bien réfléchir à choisir le bon nombre de noeuds maximum dès la création plutôt que d'ajouter des journaux par la suite.
-p lock_type	Spécifier le protocole de verroux à utiliser lorsqu'aucun protocole n'est spécifié au moment du montage : soit lock_dlm, soit lock_nolock.
-t clustername:lockspace	La table de verroux utilisée pour identifier le système de fichiers dans le cluster. Le clustername doit correspondre au nom donné à votre cluster durant la configuration. Les seuls membres du cluster sont autorisés à utiliser le système de fichiers. Le lockspace est un nom unique pour le système de fichiers gfs2.

17.7. Sources

- http://clusterlabs.org/doc/en-US/Pacemaker/1.1/html/Clusters_from_Scratch/
- https://binbash.fr/2011/09/19/des-clusters-avec-pacemaker/
- https://binbash.fr/2011/10/27/cluster-pacemaker-apache-actif/passif/
- https://www.sebastien-han.fr/blog/2011/07/04/introduction-au-cluster-sous-linux/
- https://www.yanx.eu/tag/pacemaker/

Partie 4 : Automatisation - DevOPS.

Découvrir la philosophie **devops**.

Objectifs pédagogiques

devops, **automatisation**, **gestion de configuration**, **intégration continue**, **déploiement continu**, **DSL**.

Connaissances : ⚙⚙⚙

Niveau technique : ☆

Temps de lecture : 5 minutes

Le mouvement **devops** cherche à optimiser le travail de toutes les équipes intervenant sur un système d'information.

- Les développeurs (les ***dev***) cherchent à ce que leurs applications soient déployées le plus souvent et le plus rapidement possible.
- Les administrateurs systèmes, réseaux ou de bases de données (les ***ops***) cherchent à garantir la stabilité, la sécurité de leurs systèmes et leur disponibilité.

Les objectifs des **dev** et des **ops** sont donc bien souvent opposés, la communication entre les équipes parfois difficile : les dev et les ops n'utilisent pas toujours les mêmes éléments de langage.

- Il n'y a rien de plus frustrant pour un développeur que de devoir attendre la disponibilité d'un administrateur système pour voir la nouvelle fonctionnalité de son application être mise en ligne ;
- Quoi de plus frustrant pour un administrateur système de devoir déployer une nouvelle mise à jour applicative manuellement alors qu'il vient de finir la précédente ?

La philosophie devops regroupe l'ensemble des outils des deux mondes, offre un langage commun, afin de faciliter le travail des équipes avec comme objectif la performance économique pour l'entreprise.

Le travail des développeurs et des administrateurs doit être simplifié afin d'être automatisé avec des outils spécifiques.

== Le vocabulaire DEVOPS

- le **build** : concerne la conception de l'application ;
- le **run** : concerne la maintenance de l'application ;
- le **change** : concerne l'évolution de l'application.
- l'**intégration continue** (*Continuous Integration CI*) : chaque modification d'un code source entraîne une vérification de non-régression de l'application.

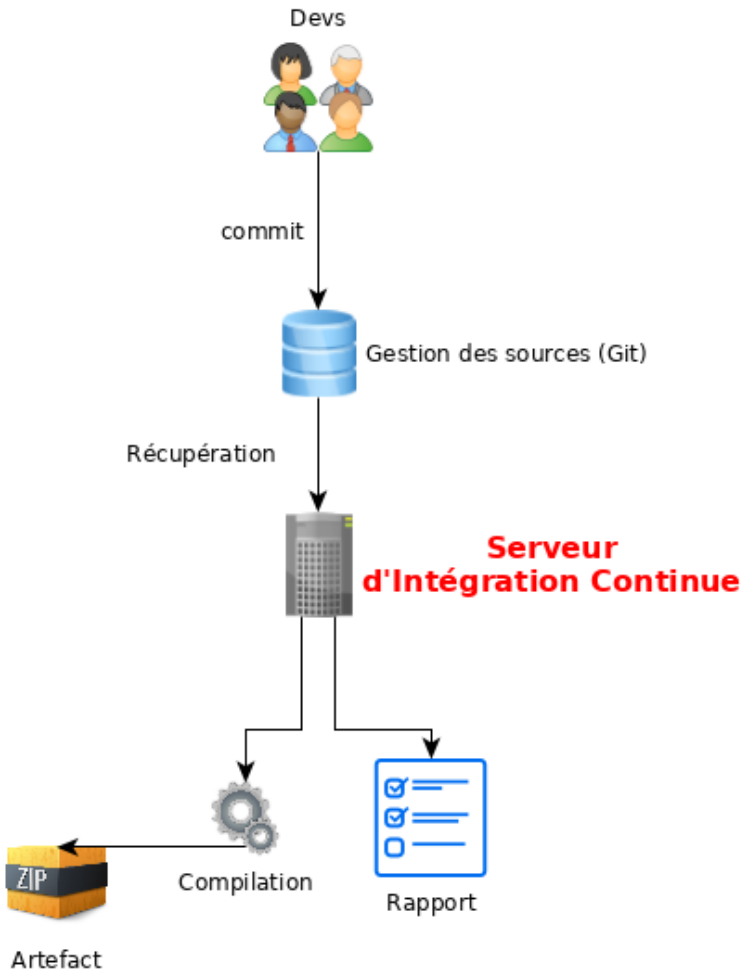


Figure 75. Schéma de fonctionnement de l'intégration continue

- **automation** (*automatisation*) : fonctionnement d'un système sans intervention humaine, automatisation d'une suite d'opération.
- **idempotence** : une opération est idempotente si elle a le même effet quelle soit appliquée une ou plusieurs fois. Les outils de gestion de configuration sont généralement idempotent.
- **orchestration** : processus automatique de gestion d'un système informatique.
- **provisionning** : processus d'allocation automatique de s ressources.

Pourquoi scripter en Bash n'est pas considéré comme de l'automatisation ?

Les langages impératifs contre les languages déclaratifs…

Même si la connaissance du **Bash** est une exigence de base pour un administrateur système, celui-ci est un langage de programmation interprété "**impératif**". Il exécute les instructions les unes à la suite des autres.

Les langages dédiés au domaine (**DSL** Domain Specific Language) comme ceux utilisés par Ansible, Terraform, ou Puppet, quant à eux, ne spécifient pas les étapes à réaliser mais l'état à obtenir.

Parce qu'Ansible, Terraform ou Puppet utilisent un langage déclaratif, ils sont très simples. Il suffit de leur dire "Fais cette action" ou "Met le serveur dans cet état". Ils considéreront l'état désiré indépendamment de l'état initial ou du contexte. Peu importe dans quel état le serveur se situe au départ, les étapes à franchir pour arriver au résultat, le serveur est mis dans l'état désiré avec un rapport de succès (avec ou sans changement d'état) ou d'échec.

La même tâche d'automatisation en bash nécessiterait de vérifier tous les états possibles, les autorisations, etc. afin de déterminer la tâche à accomplir avant de lancer l'exécution de la commande, souvent en imbriquant de nombreux "si", ce qui complique la tâche, l'écriture et la maintenance du script.

== Les outils devops

- Outils de gestion de configuration
  - Puppet
  - Ansible
  - Saltstack
  - Chef
- Intégration continue
  - Jenkins
  - Gitlab-ci
- Orchestration des tâches
  - Rundeck
  - Ansible Tower
- Infrastructure As Code
  - Terraform
- Docs as Code

- Asciidoctor
- Markdown
- ReSTructured Text

Chapitre 1. Généralités DevOPS

Découvrir la philosophie **devops**.

Objectifs pédagogiques

devops, **automatisation**, **gestion de configuration**, **intégration continue**, **déploiement continu**, **DSL**.

Connaissances : ⚙⚙⚙

Niveau technique : ☆

Temps de lecture : 5 minutes

Le mouvement **devops** cherche à optimiser le travail de toutes les équipes intervenant sur un système d'information.

- Les développeurs (les ***dev***) cherchent à ce que leurs applications soient déployées le plus souvent et le plus rapidement possible.
- Les administrateurs systèmes, réseaux ou de bases de données (les ***ops***) cherchent à garantir la stabilité, la sécurité de leurs systèmes et leur disponibilité.

Les objectifs des **dev** et des **ops** sont donc bien souvent opposés, la communication entre les équipes parfois difficile : les dev et les ops n'utilisent pas toujours les mêmes éléments de langage.

- Il n'y a rien de plus frustrant pour un développeur que de devoir attendre la disponibilité d'un administrateur système pour voir la nouvelle fonctionnalité de son application être mise en ligne ;
- Quoi de plus frustrant pour un administrateur système de devoir déployer une nouvelle mise à jour applicative manuellement alors qu'il vient de finir la précédente ?

La philosophie devops regroupe l'ensemble des outils des deux mondes, offre un langage commun, afin de faciliter le travail des équipes avec comme objectif la performance économique pour l'entreprise.

Le travail des développeurs et des administrateurs doit être simplifié afin d'être automatisé avec des outils spécifiques.

1.1. Le vocabulaire DEVOPS

- le **build** : concerne la conception de l'application ;
- le **run** : concerne la maintenance de l'application ;
- le **change** : concerne l'évolution de l'application.

- l'**intégration continue** (*Continuous Integration CI*) : chaque modification d'un code source entraîne une vérification de non-régression de l'application.

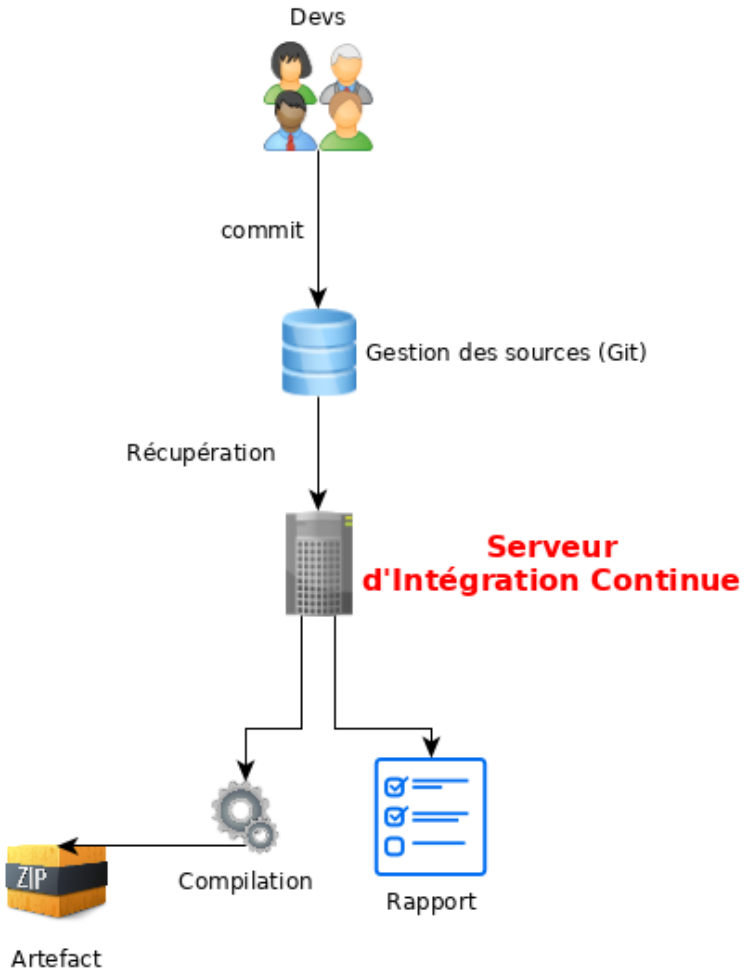


Figure 76. Schéma de fonctionnement de l'intégration continue

- **automation** (*automatisation*) : fonctionnement d'un système sans intervention humaine, automatisation d'une suite d'opération.
- **idempotence** : une opération est idempotente si elle a le même effet quelle soit appliquée une ou plusieurs fois. Les outils de gestion de configuration sont généralement idempotent.
- **orchestration** : processus automatique de gestion d'un système informatique.
- **provisionning** : processus d'allocation automatique de s ressources.

Pourquoi scripter en Bash n'est pas considéré comme de l'automatisation ?

Les langages impératifs contre les languages déclaratifs...

Même si la connaissance du **Bash** est une exigence de base pour un administrateur système, celui-ci est un langage de programmation interprété "**impératif**". Il exécute les instructions les unes à la suite des autres.

Les langages dédiés au domaine (**DSL** Domain Specific Language) comme ceux utilisés par Ansible, Terraform, ou Puppet, quant à eux, ne spécifient pas les étapes à réaliser mais l'état à obtenir.

Parce qu'Ansible, Terraform ou Puppet utilisent un langage déclaratif, ils sont très simples. Il suffit de leur dire "Fais cette action" ou "Met le serveur dans cet état". Ils considéreront l'état désiré indépendamment de l'état initial ou du contexte. Peu importe dans quel état le serveur se situe au départ, les étapes à franchir pour arriver au résultat, le serveur est mis dans l'état désiré avec un rapport de succès (avec ou sans changement d'état) ou d'échec.

La même tâche d'automatisation en bash nécessiterait de vérifier tous les états possibles, les autorisations, etc. afin de déterminer la tâche à accomplir avant de lancer l'exécution de la commande, souvent en imbriquant de nombreux "si", ce qui complique la tâche, l'écriture et la maintenance du script.

1.2. Les outils devops

- Outils de gestion de configuration
  - Puppet
  - Ansible
  - Saltstack
  - Chef
- Intégration continue
  - Jenkins
  - Gitlab-ci
- Orchestration des tâches
  - Rundeck
  - Ansible Tower
- Infrastructure As Code
  - Terraform
- Docs as Code

- Asciidoctor
- Markdown
- ReSTructured Text

Chapitre 2. Premiers pas avec git

Ces articles ont été initialement rédigés par Carl Chenet et repris par formatux avec son aimable autorisation.

Git reste un programme incompris et craint par beaucoup alors qu'il est aujourd'hui indistinctement utilisé par les développeurs et les sysadmins. Afin de démystifier ce formidable outil, je vous propose une série d'articles à l'approche très concrète pour débuter avec Git et l'utiliser efficacement ensuite.

2.1. Créer un dépôt

La première partie de cette série va se concentrer sur la création d'un dépôt Git. Afin de réfléter l'usage actuel qui allie le plus souvent un dépôt Git local et un dépôt distant, nous nous appuierons sur Gitlab.com, excellente forge logicielle en ligne basée sur le logiciel libre Gitlab.

Vous pouvez également utiliser l'instance gitlab de framagit qui héberge notamment les sources de Framatux.

Qu'est-ce qu'un dépôt de sources ?

Le dépôt de sources est très concrètement un répertoire sur votre système, contenant lu-même un répertoire caché nommé `.git` à sa racine. Vous y stockerez votre code source et il enregistrera les modifications apportées au code dont il a la charge.

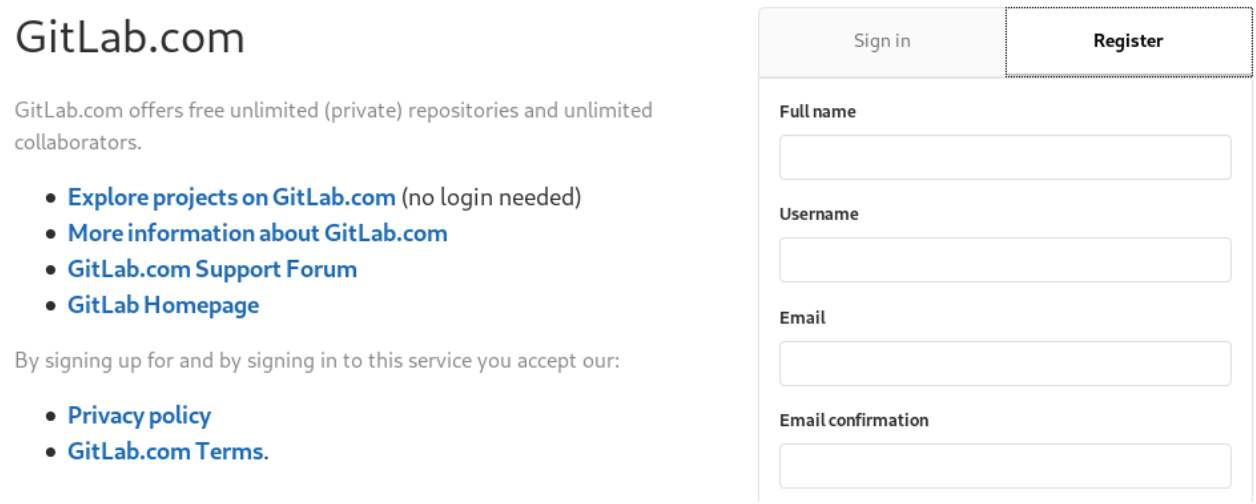
Pour prendre directement de bonnes habitudes, nous allons créer un nouveau projet à partir de notre compte Gitlab.com. Un projet Gitlab offre un dépôt distant et une suite d'outils autour. Dans un premier temps nous ne nous intéresserons qu'au dépôt. Le but est, dans un premier temps, de montrer le lien entre le dépôt distant et le dépôt local.

Pourquoi un dépôt distant ?

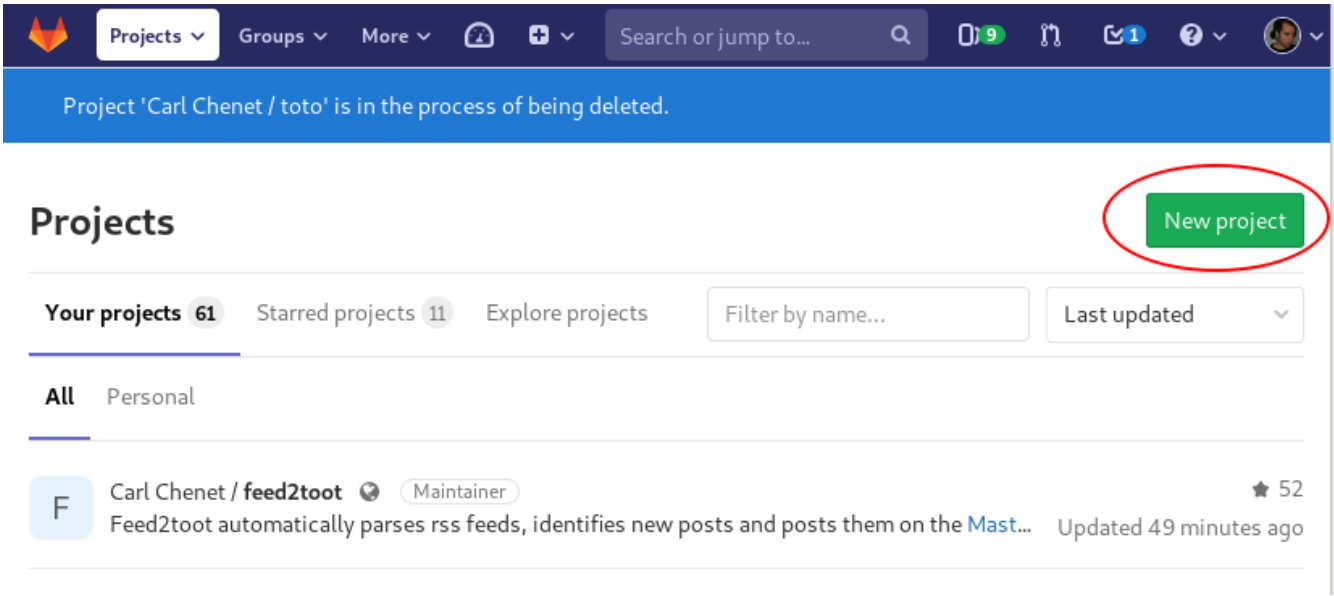
Tout simplement dans un premier temps pour sauver votre code sur un autre ordinateur que le vôtre, et dans un second temps de permettre le travail collaboratif.

Créer un projet sur Gitlab.com

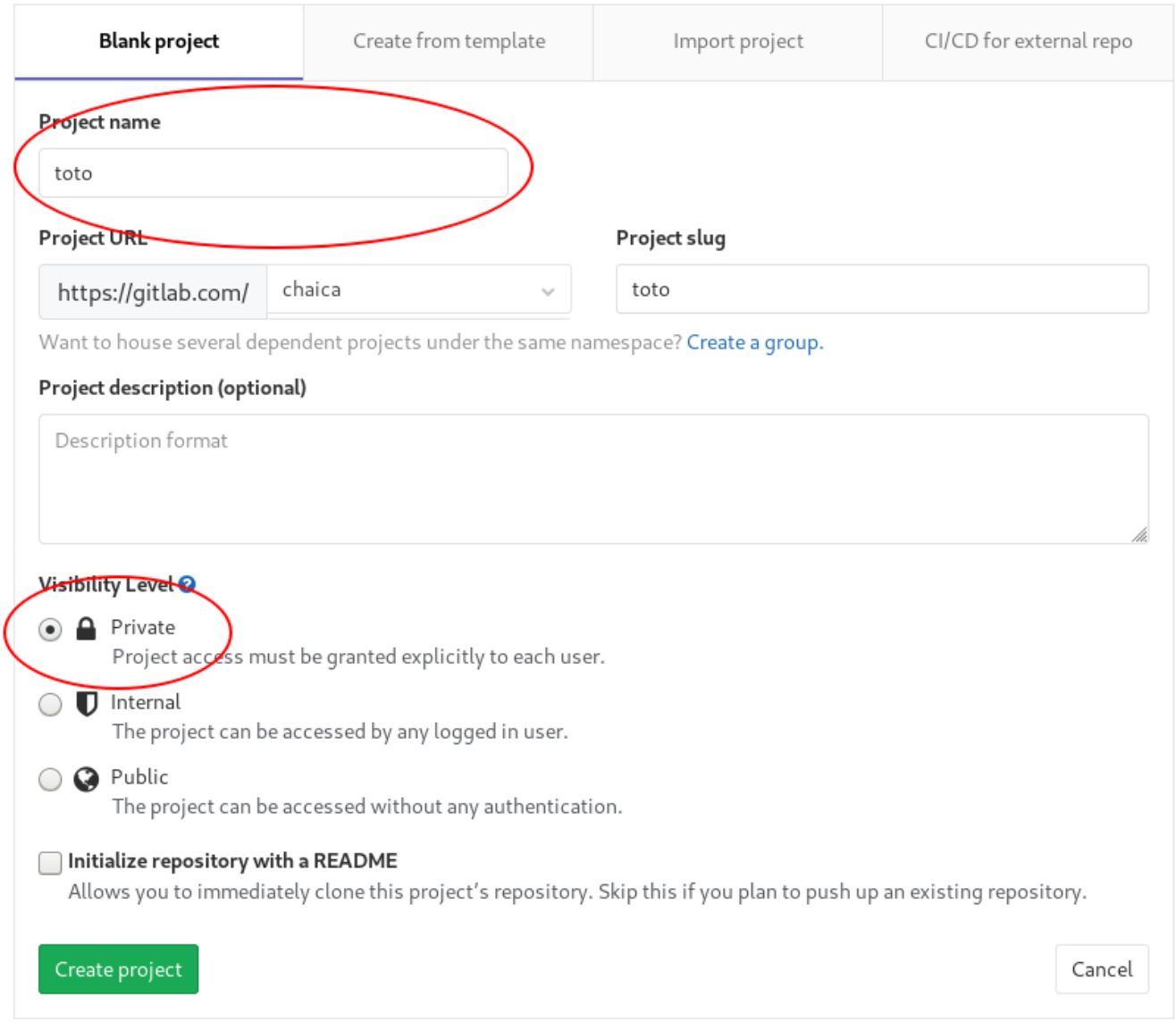
Nous commençons par créer un compte sur Gitlab.com.



Une fois connecté, nous créons maintenant un nouveau projet.



Nous lui donnons également un nom. Ce projet n'ayant pas (encore) comme but de devenir public, nous restons en dépôt privé, ce qui signifie que l'accès au dépôt sera restreint par un identifiant et un mot de passe.



Nous en avons fini avec Gitlab.com. De retour à Git et à la ligne de commande.

Cloner un dépôt distant

Sur notre poste, nous allons commencer par cloner le dépôt distant depuis Gitlab.com vers notre poste local :

```
$ git clone https://gitlab.com/chaica/toto.git
Cloning into 'toto'...
warning: You appear to have cloned an empty repository.
$ cd toto
$ ls -a
.  ..  .git
```

Comment se souvenir d'où vient ce code ? La commmande `git remote` va nous permettre de voir le lien entre notre dépôt local et le dépôt distant de Gitlab.com :

```
$ git remote -v
origin https://gitlab.com/chaica/toto (fetch)
origin https://gitlab.com/chaica/toto (push)
```

Nous détaillerons plus tard, l'important est de voir qu'il y a bien un lien entre le dépôt local et le dépôt distant.

S'identifier

Le dépôt Git dans lequel vous travaillez va bientôt chercher à identifier la personne qui procède à des modifications. Pour cela nous allons définir notre identité au niveau local – c'est-à-dire de ce dépôt – avec les commandes suivantes :

```
$ git config --local user.name "Carl Chenet"
$ git config --local user.email "chaica@ohmytux.com"
```

Git créé en fait ici un fichier `.git/config` contenant les informations fournies.

Vous pouvez également définir votre identité au niveau global, pour ré-utiliser cette configuration pour tous les dépôts que vous créerez avec cet utilisateur du système :

```
$ git config --global user.name "Carl Chenet"
$ git config --global user.email "chaica@ohmytux.com"
```

Git crée en fait ici un fichier `~/.gitconfig` contenant les informations fournies.

Saisir le mot de passe le moins possible

Lorsque nous avons cloné notre dépôt plus haut dans l'article, vous avez dû saisir un mot de passe. J'imagine que vous avez choisi un mot de passe à 24 caractères pour protéger votre compte Gitlab.com, qui est donc également le mot de passe de votre dépôt distant. Dans les prochains articles, nous allons beaucoup interagir avec le dépôt distant et il serait assez pénible d'avoir à le saisir régulièrement. Pour remédier à cela nous allons immédiatement passer la commande suivante :

```
$ git config --local credential.helper cache
```

Le dernier argument indique le type de stockage du mot de passe. Ici il s'agit uniquement d'un cache d'une durée de vie égale à 15 minutes. Si vous souhaitez stocker définitivement votre mot de passe, vous pouvez utiliser la commande suivante :

```
$ git config credential.helper store
```

Vos mots de passe seront stockés par défaut dans ~/.git-credentials. Attention, les mots de passe sont sauvés en clair dans ce fichier.

Conclusion

Pour une première approche de Git, nous avons appris à créer un projet sur Gitlab.com, à rapatrier localement le dépôt créé puis à le configurer pour l'utiliser le plus simplement possible. Nous verrons les premières utilisations de ce dépôt dans une prochaine partie.

2.2. Premier ajout de code

Après avoir vu comment débuter avec Git, nous allons détailler comment ajouter pour la première fois du code dans un dépôt de code source et les notions fondamentales de Git qui accompagnent ces opérations.

État initial du dépôt

Dans un premier temps, nous allons nous positionner dans le dépôt que nous avons créé durant la première partie sur notre poste de travail local et vérifier l'état courant de notre dépôt :

```
$ cd ~/toto
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

Plusieurs notions importantes dans le résultat de cette commande :

- Le message `On branch master` vous signale la branche dans laquelle vous êtes positionné.
- Mais qu'est-ce qu'une branche ? Retenons ici essentiellement que nous travaillons dans un état particulier du dépôt appelé `branche` et qu'il s'agit de celle par défaut, nommée `master`. Nous serons beaucoup plus précis sur cette notion dans une prochaine partie.
- Le message `No commits yet` est assez explicite et indique qu'aucun `commit` n'a encore été enregistré dans le dépôt, ce qui est cohérent.
- Mais qu'est-ce qu'un `commit` ? Un `commit` est l'état du code enregistré dans votre dépôt à un moment T, un instantané de tous les éléments dans votre dépôt au moment où vous effectuez le commit.

Allons donc maintenant ajouter un fichier dans la branche master de notre dépôt :

```
$ touch README.md
$ git add README.md
```

Créons maintenant un fichier vide nommé `README.md`. Nous utilisons ensuite la commande `git add README.md` afin d'ajouter le fichier sur la branche. Après cela, le nouvel état du dépôt est le suivant :

```
$ git status
On branch master

No commits yet

Changes to be committed:
(use "git rm --cached <file>..." to unstage)

new file: README.md
```

Nous observons que nous sommes toujours sur la branche `master` et qu'aucun `commit` n'a toujours été effectué sur cette branche. La section suivante indique par contre qu'un changement est désormais pris en compte par Git et qu'il s'agit d'un nouveau fichier nommé `README.md`.

Notion importante ici à retenir : la commande `git add` nous a permis de faire passer une modification – ici l'ajout d'un fichier – de l'état où ce fichier n'était pas pris en compte par Git vers un espace de travail, appelé ici `stage`, ce qui signifie donc que ce fichier est désormais surveillé par Git.

Premier commit

Premier grand événement de la vie de notre dépôt, nous allons maintenant effectuer notre premier `commit` avec la commande suivante :

```
$ git commit README.md -m "add README.md"
[master (root-commit) 505ace4] add README.md
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README.md
```

La command `git commit` nous permet de spécifier un fichier à prendre en compte pour le commit en question. Ici c'était facile, nous n'en avons qu'un. L'option `-m` accompagnée d'un texte permet de spécifier un message explicatif à associer à ce commit. Git nous retourne différentes informations sur le commit effectué puis nous vérifions de nouveau l'état du dépôt :

```
$ git status
On branch master
nothing to commit, working tree clean
```

Nous n'apprenons pas grand-chose. On peut remarquer que le message `No commit yet` a disparu. Nous allons passer une commande spécifique pour consulter l'historique des commits :

```
$ git log
commit 5c1b4e9826a147aa1e16625bf698b4d7af5eca9b
Author: Carl Chenet <chaica@ohmytux.com>
Date: Mon Apr 29 22:12:08 2019 +0200

add README.md
```

La commande `git log` nous apprend l'identifiant, l'auteur et la date du commit, suivi par le message explicatif dudit commit.

Pousser son commit vers le dépôt distant

Dans la première partie de cette série, nous avions créé un dépôt distant à partir duquel avait été cloné le dépôt local dans lequel nous venons de travailler. Il s'agit maintenant de synchroniser le dépôt distant avec le dépôt local. Pour cela, nous utilisons la commande suivante :

```
$ git push --set-upstream origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 220 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://gitlab.com/chaica/toto
* [new branch] master -> master
Branch master set up to track remote branch master from origin.
```

La commande `git push` nous permet de pousser nos modifications d'un dépôt local vers un dépôt distant. L'option `--set-upstream` permet d'indiquer que notre branche courante dans le dépôt local (désigné par le terme `origin` et qui est donc ici nommée `master`) sera poussée vers la branche du dépôt distant nommée elle aussi `master`. Cette option n'est obligatoire qu'à votre premier push.

Git nous indique ici `[new branch]` car, pour rappel, nous avions cloné un dépôt vide. Il créé donc la branche du dépôt distant nommé master. Refaisons appel à la commande `git remote` que nous avions déjà utilisée dans la première partie. En effet elle va nous permettre de mieux appréhender le rapport entre la branche locale et la branche distante :

```
$ git remote -v
origin https://gitlab.com/chaica/toto (fetch)
origin https://gitlab.com/chaica/toto (push)
```

Nous voyons sur la seconde ligne qu'en effet l'origine pour la branche en cours, ici l'url https://gitlab.com/chaica/toto, pour l'action `push` est bien notre dépôt distant.

Nous pouvons maintenant utiliser une version beaucoup plus simple de la commande `git push` se limitant à deux mots :

```
$ git push
Everything up-to-date
```

Le message de Git est clair, notre dépôt local est à jour par rapport au dépôt distant. Nos récentes modifications, représentées au niveau de Git par le commit que nous venons de créer, ont été poussées vers le dépôt distant, assurant la redondance des données et permettant – nous le verrons plus tard dans quelques parties – le travail collaboratif.

Conclusion

Cette deuxième partie sur comment débuter avec Git nous a permis de réaliser les toutes premières opérations d'ajouts de code sur notre dépôt local et de s'assurer de sa synchronisation avec le dépôt distant. Les notions de branche et de commit ont également été abordées pour la première fois. Dans une prochaine partie nous présenterons comment gérer un commit plus complexe.

2.3. Un commit plus complexe

Après avoir réalisé notre premier commit, nous continuons en nous intéressant aux différentes opérations possibles pour réaliser un commit plus complexe que le simple ajout d'un fichier.

Objectif

Pour bien débuter avec Git, nous avons vu dans la partie précédente comment réaliser un commit très simple, consistant à enregistrer un seul fichier. Nous allons aller plus loin pour coller davantage à la complexité du monde réel en constituant un nouveau commit avec plusieurs modifications différentes issues de plusieurs fichiers.

État initial du dépôt

Nous allons commencer avec un dépôt Git contenant un seul fichier texte nommé `README` qui est assez long.

```
$ cd commit-complexe
$ ls
README
$ wc -l README
19 README
```

Premiers changements

Nous allons ajouter sur la première ligne la phrase "do not forget to read the file foo!" et sur la

dernière ligne la phrase “do not forget to read the file bar”.

Voyons maintenant les modifications par rapport au commit précédent :

```
$ cd commit-complexe
$ git diff
diff --git a/README b/README
index d012f47..737bc05 100644
--- a/README
+++ b/README
@@ -1,5 +1,7 @@
this is a README

+do not forget to read the file foo!
+
Lots of interesting stuff here

Let's work
@@ -17,3 +19,5 @@ These criteria eliminated every then-extant version-control system, so immediate
The development of Git began on 3 April 2005.[16] Torvalds announced the project on 6 April;[17] it became self-hosting as of 7 April.[16] The first merge of multiple branches took place on 18 April.[18] Torvalds achieved his performance goals; on 29 April, the nascent Git was benchmarked recording patches to the Linux kernel tree at the rate of 6.7 patches per second.[19] On 16 June Git managed the kernel 2.6.12 release.[20]

Torvalds turned over maintenance on 26 July 2005 to Junio Hamano, a major contributor to the project.[21] Hamano was responsible for the 1.0 release on 21 December 2005 and remains the project's maintainer.[22]
+
+To finish, do not forget to read file bar!
```

Git nous indique les changements intervenus avec des `+` devant les ajouts. Nous avons donc rajouté deux lignes et deux sauts de lignes.

Choix de ce qu’on souhaite mettre dans son commit : l’index

Bien, il nous reste maintenant à écrire les nouveaux fichiers `foo` et `bar`. Mais il est 19h, je suis fatigué et je ne vais avoir le temps que d’écrire le premier fichier `foo`. Nous procédons comme suit pour écrire et ajouter le fichier `foo` :

```
$ echo "very interesting stuff" > foo
$ cat foo
very interesting stuff
```

Une fois créé, nous indiquons à Git que nous souhaitons que ce nouveau fichier soit pris en compte au prochain commit. On appelle cette opération indexer un fichier :

```
$ git add foo
```

Une furieuse envie d'enregistrer le travail en cours en faisant un `git commit -a` est notre premier réflexe, mais le fichier `README` va donc évoquer un fichier `bar` qui n'existe pas encore dans le dépôt. Ce n'est pas propre, surtout si un collègue relit immédiatement mon travail.

Dans notre situation, la solution est de choisir ce que l'on veut ajouter au commit, c'est à dire le fichier `foo` (ce que nous venons de faire) et seulement une partie des modifications du fichier `README`. C'est possible avec l'option `--patch` ou `-p` de `git add` qui va nous permettre d'indexer seulement les modifications qui nous intéressent pour préparer le prochain commit.

```
$ git add -p README
```

La commande va identifier vos différentes modifications du fichier en question et vous demander lesquelles vous souhaitez indexer pour le prochain commit.

```
@@ -1,5 +1,7 @@
this is a README

+do not forget to read the file foo!
+
Lots of interesting stuff here

Let's work
Stage this hunk [y,n,q,a,d,j,J,g,/,e,?]? y
```

Cette commande nous présente notre première modification dans le fichier `README`. Un `hunk` est ici donc une modification unitaire identifiée par Git. Le contenu ajouté apparaît avec un symbole `+` en tout début de ligne. Nous acceptons d'indexer la première modification en appuyant sur `y` (yes).

```
+
+To finish, do not forget to read file bar!
Stage this hunk [y,n,q,a,d,K,g,/,e,?]? n
```

Git nous affiche ensuite la seconde modification identifiée dans le fichier. Nous refusons la seconde en appuyant sur `n` (no), que nous ajouterons demain quand nous aurons écrit le fameux fichier bar dans un futur commit, parce qu'il est 19h et qu'à chaque jour suffit sa peine.

L'index avant le commit

Bien, après cette sélection, où en sommes-nous ? Un `git status` va nous aider.

```
$ git status
On branch master
Changes to be committed:
(use "git reset HEAD <file>..." to unstage)

modified: README
new file: foo

Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git checkout -- <file>..." to discard changes in working directory)

modified: README
```

La commande est très claire si on la lit rigoureusement : sur la branche master, les modifications suivantes sont prêtes à être committées : le fichier `README` a été modifié et un nouveau fichier `foo` a été créé.

Git nous indique également qu'un changement est survenu mais qu'il n'a pas été indexé (sous la phrase Changes not staged for commit) : c'est le fameux changement que nous souhaitons faire demain.

Le `commit` lui-même

Il est grand temps de valider nos modifications et de créer un commit. Il ne faut ici surtout pas utiliser l'option `-a` de `git commit` sous peine de valider indistinctement toutes les modifications présentes dans le dépôt, ce que nous ne voulons absolument pas faire.

On va vérifier puis valider notre commit grâce à la commande suivante :

```
$ git commit -v
```

Un éditeur de texte s'ouvre et nous affiche la sortie suivante :

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
# modified: README
# new file: foo
#
# Changes not staged for commit:
# modified: README
#
# ------------------------ >8 ------------------------
# Do not modify or remove the line above.
# Everything below it will be ignored.
diff --git a/README b/README
index d012f47..6c50e6b 100644
--- a/README
+++ b/README
@@ -1,5 +1,7 @@
this is a README

+do not forget to read the file foo!
+
Lots of interesting stuff here

Let's work
diff --git a/foo b/foo
new file mode 100644
index 0000000..7c50c6a
--- /dev/null
+++ b/foo
@@ -0,0 +1 @@
+very interesting stuff
```

Nous retrouvons ici le résultat du précédent `git status` et le détail des modifications qui vont être validées. Nous remarquons que seule la mention du fichier `foo` dans le `README` et la création de ce fichier `foo` y figurent. Nous avons donc réussi notre commit.

Si ça n'était pas le cas, vous pouvez quitter votre éditeur de texte sans enregistrer, cela annulera le commit. Si tout est bon, entrez simplement un message de commit pertinent comme “mention foo in README and add the foo file” et sauvegardez.

Le résultat de la commande est le suivant :

```
$ git commit -v
[master 077f1f6] mention foo in README and add the foo file
2 files changed, 3 insertions(+)
create mode 100644 foo
```

Nous voyons notre message de validation et également que deux fichiers ont été modifiés.

État de notre dépôt après ce commit

Dans quel état est notre dépôt après ce commit ? Encore une fois l'analyse de la commande `git status` va nous aider.

```
$ git status
On branch master
Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git checkout -- <file>..." to discard changes in working directory)

modified: README

no changes added to commit (use "git add" and/or "git commit -a")
```

En effet nous avons la seconde modification du fichier README que nous avions faite qui n'a pas encore été enregistrée dans aucun commit. Nous n'avons pas indexé cette modification. Elle demeure donc dans notre dépôt en attente d'un prochain traitement.

Un `git diff` nous montre cette modification :

```
$ git diff
diff --git a/README b/README
index 6c50e6b..737bc05 100644
+
+To finish, do not forget to read file bar!
```

Il s'agit bien de la modification que nous ne souhaitions pas encore enregistrer.

J'ai simplifié l'affichage de la commande précédente pour aller à l'essentiel.

Sauvegarde du travail

Nous avons un beau commit, mais il n'existe pour l'instant que sur notre ordinateur personnel. Un malheureux accident ou un vol est si vite arrivé, il est indispensable de sauver notre travail sur notre dépôt Git distant (voir la création du dépôt distant dans la partie 1). Nous procédons avec un simple `git push` vers notre dépôt distant:

```
$ git push
```

Conclusion

Bien débuter avec Git nécessite de comprendre ce que l'on fait et pour cela nous abordons les notions fondamentales une par une.

La notion essentielle abordée aujourd'hui est l'ajout de modifications à l'index. L'index constitue l'ensemble des modifications à prendre en compte dans le prochain commit. Nos modifications ont donc pour l'instant 3 statuts possible dans Git : non-indexées, indexées ou enregistrées dans un commit. À chaque statut correspond une palette d'opérations possibles et Git vous tient en permanence au courant du statut de vos différentes modifications en cours, en vous proposant souvent des actions possibles.

Nous aborderons bientôt dans une prochaine partie une autre notion fondamentale de Git : les branches.

2.4. Les commits et les branches

Dans cette série consacrée à l'apprentissage pratique de Git à travers des exemples, après avoir vu ce qu'est un commit, nous étudierons comment s'organisent les commits et comment passer de l'un à l'autre.

Objectif

Comme nous l'avons vu à la partie 2 et à la partie 3, Git enregistre les modifications qui surviennent au code dans le dépôt sous forme de commits.

Au fil des commits, nous construisons donc un historique des nos modifications. Git nous permet de naviguer entre ces modifications et donc de retrouver les états antérieurs des sources dans notre dépôt. Nous allons aujourd'hui détailler les possibilités offertes par cette organisation des commits.

État initial du dépôt

Nous nous positionnons dans un dépôt Git contenant actuellement deux fichiers.

```
$ cd historique-commit
$ ls
file1 file2
```

Le dépôt Git a connu 4 commits, comme nous l'indique la commande `git log`.

```
$ git log
commit ab63aad1cfa5dd4f33eae1b9f6baf472ec19f2ee (HEAD -> master)
Author: Carl Chenet <chaica@ohmytux.com>
Date: Tue May 28 20:46:53 2019 +0200

adding a line into file 2

commit 7b6882a5148bb6a2cd240dac4d339f45c1c51738
Author: Carl Chenet <chaica@ohmytux.com>
Date: Tue May 28 20:46:14 2019 +0200

add a second file

commit ce9804dee8a2eac55490f3aee189a3c67865481c
Author: Carl Chenet <chaica@ohmytux.com>
Date: Tue May 28 20:45:21 2019 +0200

adding a line in file 1

commit 667b2590fedd4673cfa4e219823c51768eeaf47b
Author: Carl Chenet <chaica@ohmytux.com>
Date: Tue May 28 20:44:30 2019 +0200

first commit
```

La commande `git status` nous précise quant à elle qu'aucun travail n'est en cours.

```
$ git status
On branch master
nothing to commit, working tree clean
```

Affichons le dernier fichier modifié pour la suite de l'article.

```
$ cat file2
this is the number 2 file

adding a line into file 2
```

Retrouver un état antérieur

Nous allons maintenant tenter de retrouver un état antérieur de notre dépôt, à savoir l'état de notre dépôt au précédent commit.

La commande `git checkout` va nous permettre de revenir à l'état de notre dépôt à un certain commit. Nous pouvons utiliser pour ça un nombre de commits antérieurs, par exemple juste 1

commit avant :

```
$ git checkout HEAD~1
```

Nous pouvons également utiliser l'identifiant du commit.

```
$ git checkout 7b6882a5148bb6a2cd240dac4d339f45c1c51738
Note: checking out '7b6882a5148bb6a2cd240dac4d339f45c1c51738'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

git checkout -b <new-branch-name>

HEAD is now at 7b6882a add a second file
```

La sortie de Git est un peu difficile à comprendre tout de suite, mais le fait est que nous sommes revenus à l'état du dépôt à l'avant-dernier commit.

Affichons le dernier fichier que nous avons modifié.

```
$ cat file2
this is the number 2 file
```

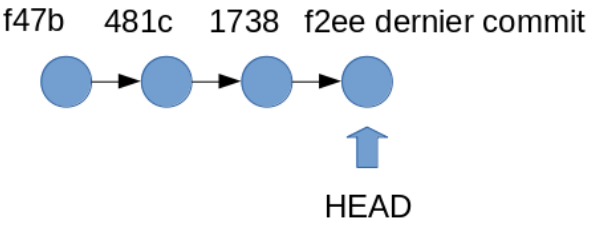
Son contenu a bien changé, nous sommes donc bien revenus en arrière dans l'histoire des modifications de notre dépôt.

Le pointeur HEAD

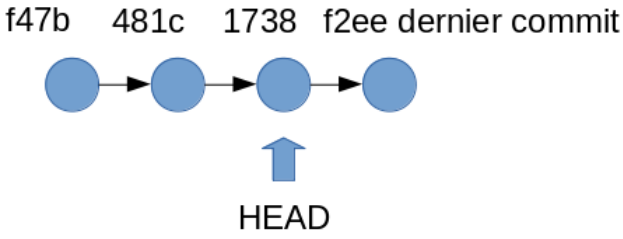
Un composant important de la commande `git` précédente reste assez obscure : que signifie `HEAD` ? Et pourquoi `~1` ?

Il s'agit tout simplement d'un pointeur de position parmi les commits de notre dépôt. Un pointeur est ici un repère logique, un petit drapeau au sein de Git, qui indique un commit et que l'on peut déplacer pour indiquer un autre commit.

Un schéma va nous aider à comprendre. Nous identifions les commits par les 4 derniers caractères de leurs identifiants.



Avant notre checkout, `HEAD` pointait sur le dernier commit réalisé. Après le `git checkout HEAD~1`, nous avons positionné `HEAD` sur l'avant-dernier commit.



Nous entrons dans un mode spécial de Git (`detached head` ou tête détachée), qui nous permet de retrouver les données du dépôt telles qu'elles étaient au moment de ce commit. À partir de cet état du dépôt, nous pourrons évoluer vers un autre état sans modifier les commits déjà existants.

Différences entre deux commits

Nous allons maintenant observer les différences entre le commit sur lequel nous sommes positionnés et le commit le plus avancé que nous avons écrit, à l'aide de la commande `git diff`.

```
$ git diff HEAD master
diff --git a/file2 b/file2
index a21d8c9..040c455 100644
--- a/file2
+++ b/file2
@@ -1 +1,3 @@
this is the number 2 file
+
+adding a line into file 2
```

Nous voyons bien la ligne ajoutée au fichier `file2` lors du dernier commit.

Remarquons que nous avons utilisé dans notre commande `master`, avec `HEAD`. Ici `HEAD` point sur l'avant-dernier commit de notre liste. Nous voyons les différences entre l'avant-dernier et le dernier commit. Or le dernier argument de notre ligne de commande était `master`. Il s'agit donc aussi, comme `HEAD`, d'un pointeur, mais sur le dernier commit réalisé. Nous y reviendrons.

Cette commande `git diff` marche bien sûr avec n'importe quel identifiant de commit, par exemple voici la différence entre le premier et le second commit, en utilisant ici leur identifiant unique.

```
$ git diff 667b2590fedd4673cfa4e219823c51768eeaf47b
ce9804dee8a2eac55490f3aee189a3c67865481c
diff --git a/file1 b/file1
index 9dd524a..2788b18 100644
--- a/file1
+++ b/file1
@@ -1 +1,3 @@
this is the number 1 file
+
+adding a line in file 1
```

Les différences entre le premier et le second commit apparaissent bien.

Écrire une suite différente : une nouvelle branche

Nous sommes donc positionnés sur l'avant-dernier commit. Nous nous apercevons que nous aimerions continuer avec un contenu différent que celui enregistré dans le dernier commit, sans toutefois effacer ce dernier commit. Pour résumer nous voulons créer un embranchement dans l'histoire de nos commits pour créer une suite différente. Nous allons donc créer notre première branche.

Pour cela il suffit de relire le dernier message affichée lors de notre commande `git checkout HEAD~1` :

```
If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>
```

Nous allons donc passer la commande suivante afin de créer une nouvelle branche dans laquelle nous écrirons la nouvelle suite des modifications que nous souhaitons.

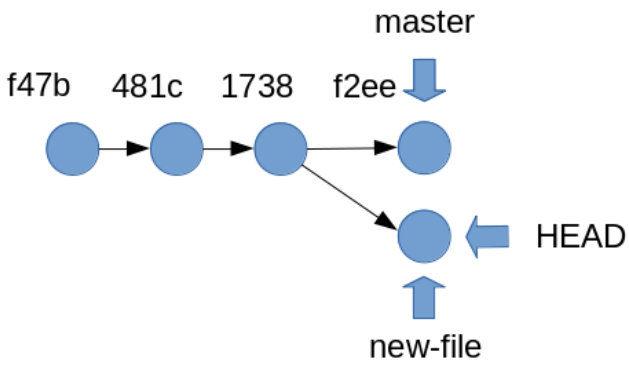
```
$ git checkout -b new-file
$ git status
On branch new-file
nothing to commit, working tree clean
```

Remarquons la première ligne `On branch new-file` alors que jusqu'ici nous avions `On branch master`. Nous avons donc bien créé une nouvelle branche nommée `new-file`.

Nous créons maintenant un nouveau commit contenant un nouveau fichier et l'ajoutons au dépôt.

```
$ echo "An unexpected new file" > file3
$ git add file3
$ git commit file3 -m "create an unexpected file"
[new-file a2e05c3] create an unexpected file
1 file changed, 1 insertion(+)
create mode 100644 file3
```

Où en sommes-nous ? Un schéma vaut mieux qu'un long discours.



Ce schéma nous apprend beaucoup de choses :

- notre première série de commits finit par un commit `f2ee` sur lequel un pointeur nommé `master` est positionné. Il s'agit de la branche `master`
- De la même façon, la branche `new-file` pointe sur notre dernier commit.
- Le pointeur `HEAD` indique l'état du dépôt sur lequel on travaille.

Une branche est donc définie par une série de commits et un pointeur du nom de cette branche sur le dernier commit de cette branche.

Conclusion

Arrêtons-nous là pour l'instant. Nous avons vu une notion fondamentale, à savoir ce qu'est réellement une branche Git et les principes sous-jacents à une branche, le lien entre les commits et les pointeurs. Il était malheureusement difficile de parler des branches précisément (ce que nous avions fait dans notre première partie) sans toutes ces notions.

Dans un dépôt Git, l'unité est le `commit`, qui est un ensemble de modifications survenus sur le code dans ce dépôt. Un `commit` et ses prédecesseurs représentent une `branche`. On positionne sur certains `commits` des `pointeurs`, ayant chacun un rôle distinct :

- Le pointeur nommé `master` pointe sur le dernier commit de la branche `master`.
- Le pointeur `new-file` pointe sur le dernier commit de la branche éponyme.
- Un pointeur spécial nommé `HEAD` indique en permanence l'état du dépôt au dernier commit pointé par le pointeur `HEAD`.

Nous verrons dans une prochaine partie comment les branches interagissent entre elles et comment les utiliser pour multiplier les différentes versions d'un travail en cours.

2.5. Fusionner des branches

Nous avons vu ce que sont les branches Git. Nous allons maintenant présenter pourquoi fusionner des branches et comment s'y prendre.

Mise en place

Pour cette partie nous créons un dépôt distant sur notre Gitlab (comme expliqué dans la partie 1), puis nous le clonons sur notre machine locale.

```
$ git clone https://gitlab.com/chaica/merge-branches.git
Cloning into 'merge-branches'…
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
$ cd merge-branches/
$ ls
README.md
```

Notre dépôt initial contient simplement un fichier `README.md`, créé par Gitlab, contenant un titre en langage `Markdown`

```
$ cat README.md
# merge-branch
```

Nous allons maintenant ajouter un sous-titre à ce fichier pour un prochain exemple.

```
$ echo -e "\n## sous-titre" >> README.md
$ git commit README.md -m "add first subtitle"
[master 2059805] add first subtitle
 1 file changed, 2 insertions(+)
```

À ce niveau, nous avons donc deux commits qui constituent notre branche master, comme nous l'indique la commande `git log`.

```
$ git log
commit 20598053fb2c3e55f95e2521dfa804739abd7d8a
Author: Carl Chenet chaica@ohmytux.com
Date:   Fri Jun 21 10:22:00 2019 +0200

 add first subtitle

commit 11cb68a24bed5236972138a1211d189adb4512a8 (origin/master, origin/HEAD)
Author: Carl Chenet chaica@ohmytux.com
Date:   Fri Jun 21 08:18:56 2019 +0000

 Initial commit
```

Mise en place un peu longue, mais qui nous a permis de réviser quelques commandes fondamentales. Nous entrons dans le vif du sujet.

Création d'une nouvelle branche

Un client nous demande une évolution du code. Afin de ne pas toucher à la branche `master` qui contient le code de référence courant, nous allons compliquer maintenant un peu les choses en créant une nouvelle branche nommée `branch-with-foo`. Cette étape a déjà été expliquée dans la partie 4 plus en détail.

```
$ git checkout -b branch-with-foo
Switched to a new branch 'branch-with-foo'
```

Nous créons immédiatement un fichier nommé `foo` dans cette branche que nous ajoutons et enregistrons dans la foulée.

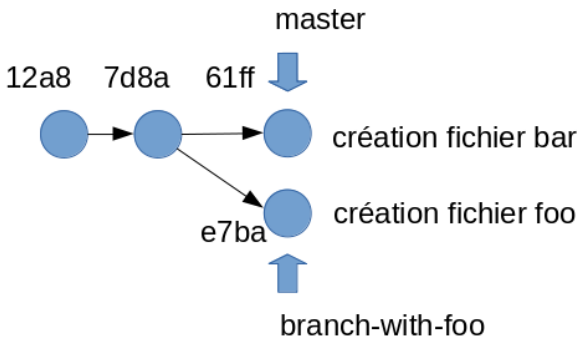
```
$ echo "this is a foo file" > foo
$ git add foo && git commit foo -m "add foo file"
[branch-with-foo d9afaa2] add foo file
 1 file changed, 1 insertion(+)
 create mode 100644 foo
```

Divergence des branches

Nous revenons maintenant sur master et nous créons un fichier `bar` que nous enregistrons aussi dans la foulée.

```
$ git checkout master
$ echo "this is a bar file" > bar
$ git add bar && git commit bar -m "add bar file"
[master 222c618] add bar file
 1 file changed, 1 insertion(+)
 create mode 100644 bar
```

Faisons une pause, nous venons de créer notre première divergence entre nos branches, nous avons créé un embranchement dans l'historique de nos commits, les deux derniers commits n'appartenant plus aux mêmes branches.



Le schéma présente la divergence entre la branche `master` et la branche `branch-with-foo`. La première contient un fichier `bar`, la seconde un fichier `foo`. Bien, il est temps de passer aux choses sérieuses.

Fuuuuuuuuuusion

Le besoin que nous avions de créer une nouvelle branche a disparu, le client a annulé le projet.

Bon, nous allons réintégrer les modifications de cette branche dans la branche `master`. Nous nous positionnons dans la branche `master`, ou d'une manière générale la branche à laquelle nous souhaitons réintégrer les modifications d'une autre, et nous passons la commande suivante :

```
$ git checkout master
$ git merge branch-with-foo -m "Merge branch 'branch-with-foo'"
Merge made by the 'recursive' strategy.
  foo | 1 +
  1 file changed, 1 insertion(+)
  create mode 100644 foo
```

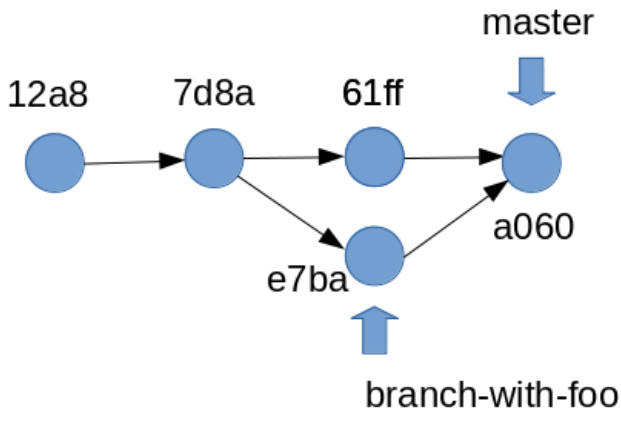
La sortie de la commande nous précise ce qui s'est passé : un fichier `foo` (celui de la branche `branch-with-foo`) a été créé dans la branche courante `master`.

Jetons un oeil à l'historique avec la commande `git log` avec l'option `--graph` qui va nous présenter une représentation graphique de notre historique et l'option `--oneline` afin de rendre la commande moins verbeuse.

```
$ git log --graph --oneline
* 69fa060 (HEAD -> master) Merge branch 'branch-with-foo'
|\
| * d9afaa2 (branch-with-foo) add foo file
* |222c618 add bar file
|/
* 2059805 add first subtitle
* 11cb68a (origin/master, origin/HEAD) Initial commit
```

Cette représentation est très parlante. Au niveau de l'histoire elle va de haut en bas, le haut étant le `commit` le plus récent et le plus bas le plus vieux. Une étoile (`*`) est un `commit`, avec son message à droite , et le nom de la branche s'il y ambiguité.

Nous reprenons notre dessin précédent et le faisons évoluer.



Nous avons bien fusionné la branche `branch-with-foo` dans `master`. **Fusion réussie**.

Sauver son code et ses branches

Avant de s'arrêter aujourd'hui, n'oublions pas de pousser tout ce que nous avons fait vers notre

dépôt Gitlab distant. Nous commençons par la branche `master`.

```
$ git push
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (11/11), 975 bytes | 975.00 KiB/s, done.
Total 11 (delta 2), reused 0 (delta 0)
To https://gitlab.com/chaica/merge-branches.git
   11cb68a..69fa060  master -> master
```

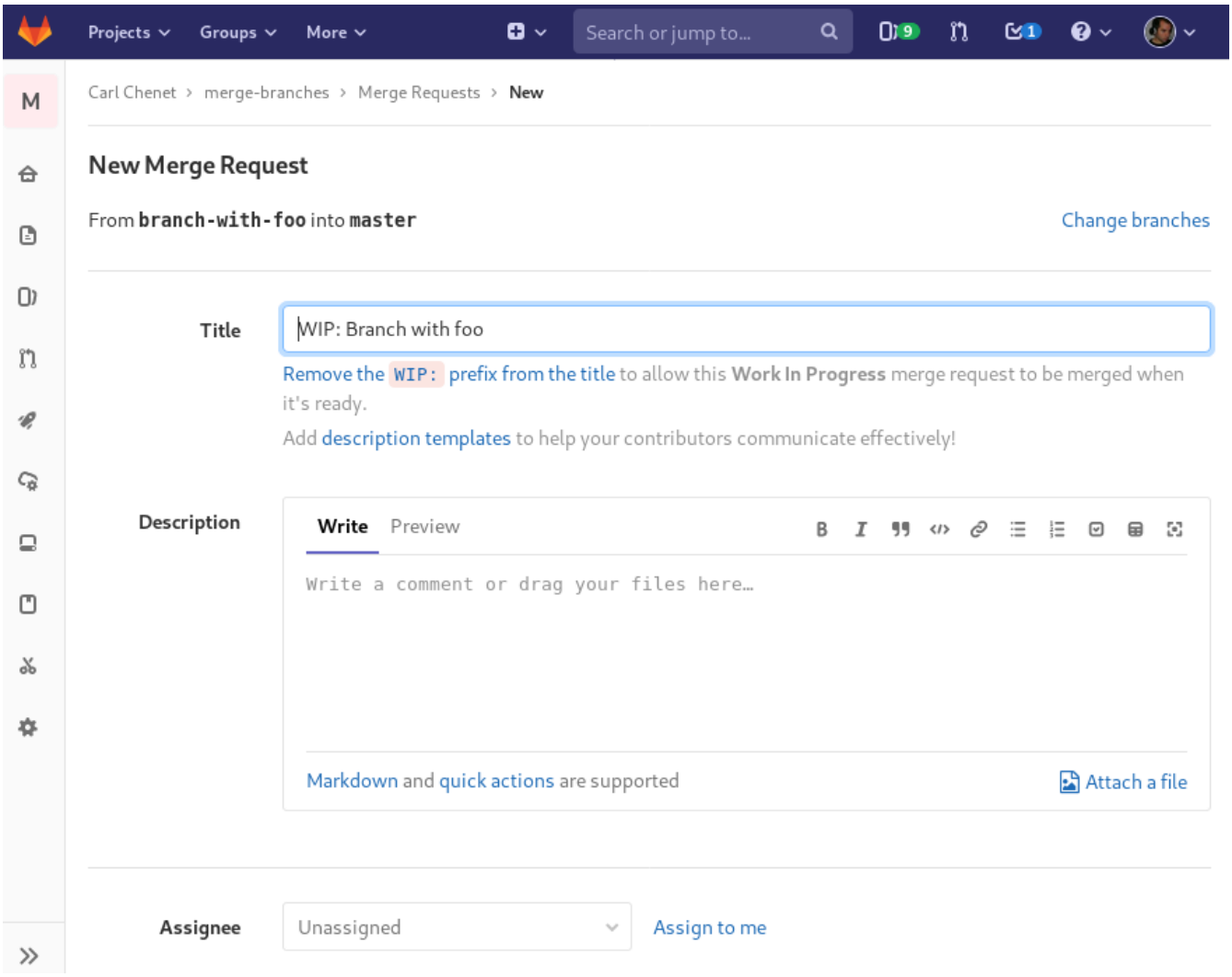
La dernière ligne indique bien que nous avons poussé depuis notre branche `master` locale vers notre branche `master` distante présent sur notre Gitlab.

Passons à la branche `branch-with-foo`, qui, même si elle a été fusionnée dans `master`, existe toujours. Pourquoi ? Car le nom de la branche `branch-with-foo` est un pointeur, un indicateur qui désigne le dernier commit connu de cette branche. Rien de plus.

Pour changer un peu nous varions la syntaxe de notre commande `git push` en utilisant l'option `--all` afin de pousser toutes les branches locales vers le dépot distant. Nous n'en avons qu'une ici, `branch-with-foo`.

```
$ git push --all
Total 0 (delta 0), reused 0 (delta 0)
remote:
remote: To create a merge request for branch-with-foo, visit:
remote:   https://gitlab.com/chaica/merge-
branches/merge_requests/new?merge_request%5Bsource_branch%5D=branch-with-foo
remote:
To https://gitlab.com/chaica/merge-branches.git
[new branch]      branch-with-foo -> branch-with-foo
```

Point très intéressant, nous voyons que les lignes commençant par `remote:` proviennent du Gitlab, qui nous indiquent comment créer une demande de fusion (`Merge Request`). Inutile, nous avons déjà fusionné. Mais ce sera intéressant dans le cadre du travail collaboratif. Ce sera pour un prochain article.



La dernière ligne nous confirme que nous avons bien poussé la branche locale `branch-with-foo` vers la branche du dépôt Gitlab distant nommée également `branch-with-foo`.

Conclusion

Nous avons vu aujourd'hui la fusion de branches Git. Cette opération permet de récupérer le travail réalisé dans une autre branche, en divergence du code "principal" que nous conservons – comme bonne pratique dans l'industrie en général – dans la branche `master`. C'est le cas le plus courant.

Vous devriez quasi systématiquement commencer par créer une nouvelle branche quand vous envisagez d'introduire du nouveau code dans un dépôt Git, afin de ne pas travailler directement vous-même dans `master`. Pourquoi pas ? C'est ce que nous verrons dans le prochain article de cette série.

2.6. Une fusion de branches échoue

Nous allons démystifier une situation qui provoque un stress important auprès des nouveaux utilisateurs de Git : lorsqu'une fusion de branches échoue.

Il arrive malheureusement qu'une fusion de branches échoue. Ce cas se présente surtout dans l'utilisation collaborative de Git, où différentes personnes interviennent en même temps sur le code.

En effet Git est un outil très bien conçu, mais il ne prendra jamais une décision à la place de ses utilisateurs. Or, si la même partie du code est modifiée par deux utilisateurs différents, qui a raison ? Qui a tort ? Qui Git doit-il croire ?

Mise en place du conflit de fusion

Nous allons tout d'abord créer un fichier `README.md` dans un nouveau dépôt Git avec un titre et deux sections.

```
$ mkdir fusion-echoue
$ cd fusion-echoue/
$ git init .
Initialized empty Git repository in /tmp/fusion-echoue/.git/
$ vi README.md
$ cat README.md
# README

## installation

apt-get install whatever

## use

whatever param1 param3 param3
$ git add README.md && git commit README.md -m "add README.md"
add README.md
 1 file changed, 9 insertions(+)
 create mode 100644 README.md
```

Nous créons maintenant une branche dédiée à l'ajout d'une nouvelle section pour indiquer la licence du projet dans le `README.md`.

```
$ git checkout -b new-license
Switched to a new branch 'new-license'
$ vi README.md
$ git commit README.md -m "license section"
license section
 1 file changed, 3 insertions(+)
```

Jetons un oeil à l'état actuel de notre fichier sur a branche `new-license` :

```
# README

## installation

apt-get install whatever

## use

whatever param1 param3 param3

# license
BSD
```

Un code, deux choix différents

Pendant ce temps, dans la branche `master`, un développeur semble avoir fait un choix différent au niveau de la licence.

```
git checkout master
Switched to branch 'master'
$ vi README.md
$ git commit README.md -m "license section - use gpl"
[master bf15ff3] license section - use gpl
 1 file changed, 3 insertions(+)
```

Voici l'état actuel du fichier sur la branche master.

```
# README

## installation

apt-get install whatever

## use

whatever param1 param3 param3

# license
GPLv3
```

Une fusion destinée à échouer

L'événement-déclencheur du drame va être la volonté de supprimer la branche créée auparavant et de réintégrer son code dans `master`. Pour cela, rien de plus simple, nous utilisons `git merge` comme

vu dans l'article précédent.

```
$ git merge new-license
 Auto-merging README.md
 CONFLICT (content): Merge conflict in README.md
 Automatic merge failed; fix conflicts and then commit the result.
```

Oups, le `CONFLICT` en majuscule nous annonce que Git a rencontré un conflit en tentant de fusionner la branche `new-license` dans la branche `master`.

La dernière phrase est très claire : résoudre les conflits est un préalable indispensable à continuer la fusion.

Résoudre le conflit… ou pas

Si vous avez déjà rencontré ce problème en utilisant Git, vous avez sûrement éprouvé ce sentiment : la grosse panique. Vous vous voyez déjà accuser des pires manipulations par vos collègues et de casser l'outil de travail commun. **DU CALME, PAS DU TOUT**.

Avant de céder à la panique, repassons la commande `git status`.

```
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)
Unmerged paths:
  (use "git add …" to mark resolution)
both modified:   README.md
```

La sortie de la commande est très claire : un simple `git merge --abort` vous fera revenir à la situation précédant le `git merge`. Voyons cela.

```
$ git merge --abort
$ git status
On branch master
nothing to commit, working tree clean
```

Voilà, vous pouvez vous détendre et ranger votre lette de démission. Tout va bien se passer.

Résoudre le conflit

Nous allons maintenant tenter de réellement résoudre notre conflit. Avant cela, nous devons prendre conscience de la tâche qui nous attend.

```
$ git merge new-license
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)
Unmerged paths:
  (use "git add …" to mark resolution)
both modified:   README.md
```

La sortie de la commande `git status` nous indique qu'un seul fichier pose problème par la mention sur la dernière ligne `both modified: README.md`.

Ce n'est pas nos quelques lignes dans le `README.md` qui vont nous décourager. Jetons donc un oeil au fichier en question.

```
$ cat README.md
# README

## installation

apt-get install whatever

## use

whatever param1 param2 param3

## license
<<<<<<< HEAD
GPLv3
=======
BSD
>>>>>>> new-license
```

Surprise, Git nous a maché le travail.

Dans la section license du fichier, nous voyons que la partie entre `<<<<<<< HEAD` et `=======` provient de `HEAD`, qui pointe actuellement sur `master`, donc le code provenant de `master`. Vous avez oublié ce qu'est `HEAD` ? On en parle dans la partie 4.

À l'opposé, la partie entre `=======` et `>>>>>>> new-license` provient donc de la branche `new-license`. C'est maintenant à vous de jouer et de faire un choix, Git ne le fera pas à votre place, une décision humaine est attendue.

Après consultation avec vos collègues, vous choisissez la licence GPLv3. Vous allez donc éditer le code de la section license pour ne laisser que la partie suivante :

```
## license
GPLv3
```

La commande `git status` précédente nous indiquait la marche à suivre une fois le fichier édité.

```
Unmerged paths:
   (use "git add ⋯" to mark resolution)
 both modified:   README.md
```

Nous suivons donc ces instructions à la lettre.

```
$ git add README.md
$ git status
On branch master
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)
```

Le conflit est résolu. Nous allons maintenant utiliser `git commit` pour terminer la fusion.

```
$ git commit -a -m "use GPLv3 license. merge from new-license branch"
```

Si vous n'utilisez que la commande `git commit` sans argument, votre éditeur de texte s'ouvre et propose d'enregistrer le message de commit suivant :

```
Merge branch 'new-license'
```

Conclusion

Nous avons brilliamment résolu notre premier conflit de fusion entre deux branches. Il était franchement facile à régler, mais il nous a permis de détailler toutes les étapes nécessaires. Bien évidemment nous pouvons avoir des conflits bien plus sévères à régler, mais nous avons vu comment interrompre une fusion, ce qui nous aidera quoiqu'il arrive à revenir à un état stable du dépôt.

S'engager dans une fusion complexe réclame un très bonne connaissance du code manipulé et de ce qu'on fait vos collègues, pour ne pas écraser du code utile par inadvertance.

Chapitre 3. Gestion de configurations avec Puppet

Il est difficile de s'appuyer sur des processus manuels ou des scripts personnalisés pour accomplir des tâches répétitives.

Lorsque les environnements grossissent ou que les équipes accueillent de plus en plus de techniciens, ces méthodes sont difficiles à maintenir et à optimiser. Elles peuvent causer des risques pour la sécurité du système, incluant des erreurs de configuration, ce qui au final, peut faire réduire la productivité.

La gestion automatique des configurations élimine beaucoup de travail manuel et augmente la réactivité des équipes. Cette réactivité devient de plus en plus importante avec la montée en puissance de la virtualisation, qui révolutionne notre gestion du cycle de vie des serveurs : durée de vie plus courte, déploiement plus rapide, configurations plus standardisées et conformes.

Parmi les systèmes de gestion de configurations, plusieurs systèmes ont fait leur apparition :

- puppet ;
- chef ;
- ansible ;
- etc.

Des outils ont également fait leur apparition pour encore faciliter l'administration de ces systèmes :

- geppetto : un environnement de développement (IDE) pour puppet ;
- the foreman : un gestionnaire de cycle de vie complet des serveurs.

3.1. La gestion de configuration

La gestion de configuration est le processus de standardisation des configurations de ressources et l'assurance de leur état dans l'infrastructure informatique, avec des méthodes automatisées mais agiles. Le management de configurations est devenu critique pour le succès des projets informatiques.

Concrètement, lors de l'ajout d'un nouveau serveur au sein d'une infrastructure informatique complexe, les administrateurs système ne doivent pas perdre de temps pour la configuration des briques de base du système : la configuration des services NTP, DNS, SMTP, SSH, la création des comptes utilisateurs, etc… doit être totalement automatisée et transparente aux équipes.

L'utilisation d'un gestionnaire de configuration doit également permettre d'installer un clone de serveur d'une manière totalement automatisée, ce qui peut être pratique pour des environnements multiples (Développement → Intégration → Pré-production → Production).

La combinaison d'outils de gestion de configuration avec l'utilisation de dépôts de gestion de versions, comme « git », permet de conserver un historique des modifications apportées au système.

3.2. Puppet

Puppet a été conçu pour fonctionner en mode client-serveur. Son utilisation en mode de fonctionnement « autonome » est également possible et facile. La migration vers un système de clients « Puppet / Master Puppet » n'est pas d'une réalisation complexe.

Puppet est un logiciel d'automatisation qui rend simple pour l'administrateur système le provisionnement (la description matérielle d'une machine virtuelle), la configuration et la gestion de l'infrastructure tout au long de son cycle de vie. Il permet de décrire l'état de configuration d'un ensemble hétérogène de stations de travail ou de serveurs et de s'assurer que l'état réel des machines correspond bien à l'état demandé.

Par sa structure de langage, il fait le lien entre les bonnes pratiques, le cahier de procédures et l'état effectif des machines.

Vocabulaire Puppet

- **Noeud** (Node) : serveur ou poste de travail administré par Puppet ;
- **Site** : ensemble des noeuds gérés par le Puppet Master ;
- **Classe** : moyen dans Puppet de séparer des morceaux de code ;
- **Module** : unité de code Puppet qui est réutilisable et pouvant être partagé ;
- **Catalogue** : ensemble des classes de configuration à appliquer à un nœud ;
- **Facter** : librairie multi-plateforme qui fournit à Puppet sous forme de variables les informations propres au système (nom, adresse ip, système d'exploitation, etc.) ;
- **Ressource** (Resource): objet que Puppet peut manipuler (fichier, utilisateur, service, package, etc.) ;
- **Manifeste** (Manifest) : regroupe un ensemble de ressource.

Architecture

Puppet conseille de coupler son fonctionnement avec un gestionnaire de version type « git ».

Un serveur PuppetMaster contient la configuration commune et les points de différence entre machines ;

Chaque client fait fonctionner puppetd qui :

- applique la configuration initiale pour le nœud concerné ;
- applique les nouveautés de configuration au fil du temps ;

- s'assure de manière régulière que la machine correspond bien à la configuration voulue.

La communication est assurée via des canaux chiffrés, en utilisant le protocole https et donc TLS (une mini-pki est fournie).

Toute la configuration (le référentiel) de Puppet est centralisée dans l'arborescence /etc/puppet du serveur de référence :

- **/etc/puppet/manifests/site.pp** : est le premier fichier analysé par Puppet pour définir son référentiel. Il permet de définir les variables globales et d'importer des modules ;
- **/etc/puppet/manifests/node.pp** : permet de définir les nœuds du réseau. Un nœud doit être défini par le nom FQDN de la machine ;
- **/etc/puppet/modules/<module>** : sous-répertoire contenant un module.

3.3. Installation

Les dépôts Puppets doivent être activés :

```
[root]# vim /etc/yum/yum.repos.d/puppetlabs.repo
[puppetlabs-products]
name=Puppet Labs Products El 6 - $basearch
baseurl=http://yum.puppetlabs.com/el/6/products/$basearch
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs
enabled=1
gpgcheck=1

[puppetlabs-deps]
name=Puppet Labs Dependencies EL 6 - $basearch
baseurl=http://yum.puppetlabs.com/el/6/dependencies/$basearch
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs
enabled=1
gpgcheck=1
```

puis :

```
[root]# yum update
[root]# yum install puppet
```

3.4. Hello world

Pour fonctionner, le client autonome puppet a besoin d'un fichier appelé manifeste, dont l'extension sera en « .pp ».

Le langage utilisé est le ruby.

Créer un fichier /root/config-init.pp :

```
[root]# vim /root/config-init.pp
file {'helloworld':
  path => '/tmp/helloworld',
  ensure => present,
  mode => 0640,
  content => "Helloworld via puppet ! "
}
```

Exécuter ce manifeste avec la commande puppet :

```
[root]# puppet apply /root/config-init.pp
Notice : Compiled catalog for centos6 in environnement production in 0.31 seconds
Notice: /Stage[main]/main/File[helloworld]/ensure: created
Notice: Finished catalog run in 0.07 seconds
```

3.5. Les modules Puppet

Les modules complémentaires à Puppet peuvent être téléchargés sur le site puppetlabs.

L'installation d'un module complémentaire pourra se faire, soit directement depuis internet, soit par le téléchargement manuel de l'archive .tar.gz.

Depuis internet :

```
puppet module install nomdumodule
```

Une commande de recherche de modules existe :

```
puppet module search nomdumodule
```

Pour rechercher les modules installés :

```
puppet module list
```

Et pour les mettre à jour :

```
puppet module upgrade nomdumodule
```

Pensez à préciser le proxy dans la commande puppet en exportant les variables http_proxy et https_proxy :

```
export http_proxy=http://10.10.10.7:8080
export https_proxy=http://10.10.10.7:8080
```

Sans connexion internet

Sans connexion internet, un module peut être installé en fournissant dans la commande puppet le chemin vers le fichier tar.gz :

```
puppet module install ~/nomdumodule.tar.gz --ignore-dependencies
```

Grâce aux modules du dépôt Puppet, les possibilités de l'outil sont quasiment infinies. Le téléchargement et l'utilisation des modules du dépôt permettent un gain de temps confortable car l'outil nécessite peu de compétences en développement.

3.6. Documentation

La liste des types et leurs attributs est consultable en ligne :

- https://docs.puppetlabs.com/references/latest/type.html

Une documentation de formation est également consultable :

- https://doc.puppetlabs.com/learning/introduction.html

3.7. Commandes utiles

Lister les objets connus par puppet :

```
puppet describe -l
```

Lister les valeurs possibles d'une ressource :

```
puppet describe user
```

Lister les objets du système :

```
puppet resource user
```

3.8. Cas concrets

La création d'un noeud (node)

Le code du manifeste doit être découpé en classe pour des raisons de maintenance et d'évolutivité.

Un objet de type node recevra les classes à exécuter. Le node « default » est automatiquement exécuté.

Le fichier site.pp contiendra l'ensemble du code :

```
node default {
  include init_services
}
```

Gestion des services

La classe init_services contient les services qui doivent être lancés sur le nœud et ceux qui doivent être stoppés :

```
class init_services {

  service { ["sshd","NetworkManager","iptables","postfix","puppet","rsyslog","sssd",
"vmware-tools"]:
    ensure => 'running',
    enable => 'true',
  }

  service { ["atd","cups","bluetooth","ip6tables","ntpd","ntpdate","snmpd","snmptradp"]:
    ensure => 'stopped',
    enable => 'false',
  }
}
```

La ligne ensure ⇒ a pour effet de démarrer ou non un service, tandis que la ligne enable ⇒ activera ou non le démarrage du service au démarrage du serveur.

Gestion des utilisateurs

La classe create_users contiendra les utilisateurs de notre système. Pensez à ajouter l'appel de cette classe dans le node default !

```
class create_users {
  user { 'antoine':
    ensure => present,
    uid => '5000',
    gid => 'users',
    shell => '/bin/bash',
    home => '/home/antoine',
    managehome => true,
  }
}
```

```
node default {
  include init_services
  include create_users
}
```

Au départ du personnel pour mutation, il sera aisé de supprimer son compte en remplaçant la ligne ensure ⇒ present par ensure ⇒ absent et en supprimant le reste des options.

La directive managehome permet de créer les répertoires personnels à la création des comptes.

La création des groupes est toute aussi aisée :

```
group { "DSI":
  ensure => present,
  gid => 1001
}
```

Gestion des dossiers et des fichiers

Un dossier peut être créé avec la ressource « file » :

```
file { '/etc/skel/boulot':
  ensure => directory,
  mode   => 0644,
}
```

Un fichier peut être copié d'un répertoire vers un autre :

```
file { '/STAGE/utilisateurs/gshadow':
  mode   => 440,
  owner  => root,
  group  => root,
  source => "/etc/gshadow"
}
```

Modification de valeurs

La commande augeas, développée par la société RedHat, permet de modifier les valeurs des variables dans les fichiers de configuration. Son utilisation peut s'avérer autant puissante que complexe.

Un usage minimaliste serait par exemple :

```
augeas { "Modification default login defs" :
    context => "/files/etc/login.defs",
    changes => ["set UID_MIN 2000","set GID_MIN 700","set PASS_MAX_DAYS 60"],
}
```

Le contexte est suffixé de « /files/ » pour préciser qu'il s'agit d'un système de fichiers local.

Exécution d'une commande externe

La commande exec est utilisée pour lancer une commande externe :

```
exec { "Redirection":
  command => "/usr/sbin/useradd -D > /STAGE/utilisateurs/defaut",
}
```

De manière générale, les commandes et les fichiers doivent être décrits en absolu dans les manifestes.

Rappel : la commande whereis fournit le chemin absolu d'une commande.

Chapitre 4. Ansible

🎓 Objectifs

✔ Mettre en oeuvre Ansible ;
✔ Appliquer des changements de configuration sur un serveur ;
✔ Créer des playbooks Ansible.

Ansible centralise et automatise les tâches d'administration. Il est :

- sans **agent** (il ne nécessite pas de déploiements spécifiques sur les clients),
- **idempotent** (effet identique à chaque exécution)
- et utilise le protocole **SSH** pour configurer à distance les clients Linux.

L'interface graphique web Ansible Tower est payante.

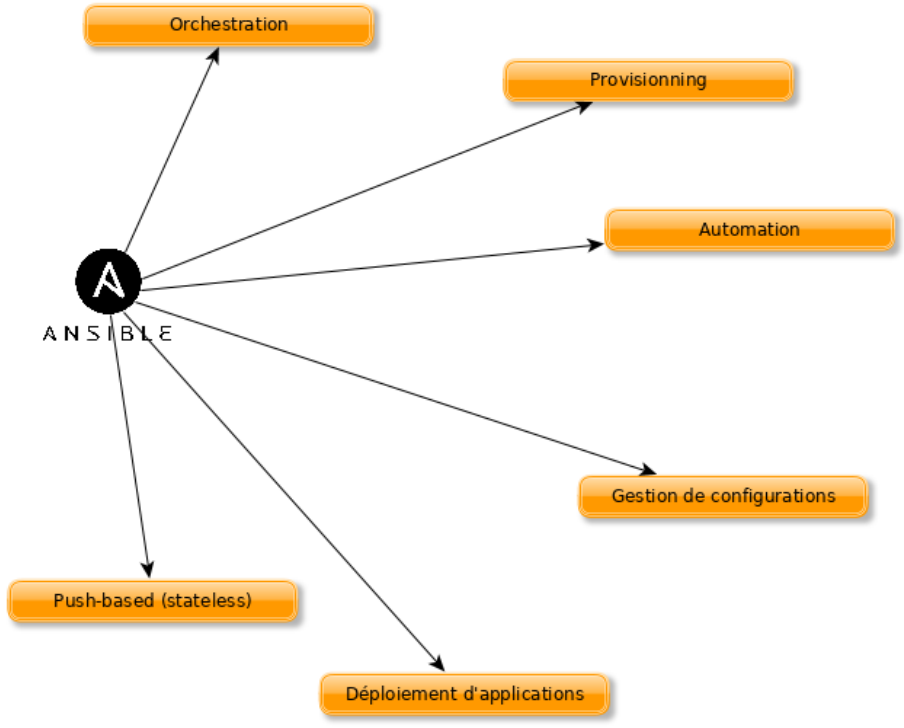


Figure 77. Les fonctionnalités d'Ansible

L'ouverture des flux SSH vers l'ensemble des clients depuis le serveur Ansible font de lui un elément critique de l'architecture qu'il faudra attentivement surveiller.

4.1. Le vocabulaire ansible

- Le **poste de gestion** : la machine sur laquelle Ansible est installée. Ansible étant **agentless**, aucun logiciel n'est déployé sur les serveurs gérés.
- L'**inventaire** : un fichier contenant les informations concernant les serveurs gérés.
- Les **tâches** (tasks) : une tâche est un bloc définissant une procédure à exectuer (par exemple créer un utilisateur ou un groupe, installer un paquet logiciel, etc.).
- Un **module** : un module rend abstrait une tâche. Il existe de nombreux modules fournis par Ansible.
- Les **playbooks** : un fichier simple au format yaml définissant les serveurs cibles et les tâches devant être effectuées.
- Un **rôle** : un rôle permet d'organiser les playbooks et tous les autres fichiers nécessaires (modèles, scripts, etc.) pour faciliter le partage et la réutilisation du code.
- Les **facts** : ce sont des variables globales contenant des informations à propos du système (nom de la machine, version du système, interface et configuration réseau, etc.).
- les **handlers**: il sont utilisés pour provoquer un arrêt ou un redémarrage d'un service en cas de changement.

4.2. Installation sur le serveur de gestion

Ansible est disponible dans le dépôt *EPEL* :

- Installation d'EPEL :

```
$ sudo yum install epel-release
```

La configuration du serveur se situe sous `/etc/ansible`.

Deux fichiers de configuration :

- Le fichier de configuration principal `ansible.cfg` : commandes, modules, plugins, configuration ssh ;
- Le fichier inventaire de gestion des machines clientes `hosts` : déclaration des clients, des groupes.

Le fichier /etc/ansible/hosts

```
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
#   - Comments begin with the '#' character
#   - Blank lines are ignored
#   - Groups of hosts are delimited by [header] elements
#   - You can enter hostnames or ip addresses
#   - A hostname/ip can be a member of multiple groups

# Ex 1: Ungrouped hosts, specify before any group headers.

## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10

# Ex 2: A collection of hosts belonging to the 'webservers' group

## [webservers]
## alpha.example.org
## beta.example.org
## 192.168.1.100
## 192.168.1.110

...
```

Par exemple, un groupe `centos7` est créé en insérant le bloc suivant dans ce fichier :

Création d'un groupe d'hôtes dans /etc/ansible/hosts

```
[centos7]
172.16.1.217
172.16.1.192
```

4.3. Utilisation en ligne de commande

La commande `ansible` lance une tâche sur un ou plusieurs hôtes cibles.

Syntaxe de la commande ansible

```
ansible <host-pattern> [-m module_name] [-a args] [options]
```

Exemples :

- Lister les hôtes appartenant à un groupe :

```
ansible {{group}} --list-hosts
```

- Pinger un groupe d'hôtes avec le module `ping` :

```
ansible {{group}} -m ping
```

- Afficher des facts d'un groupe d'hôtes avec le module `setup` :

```
ansible {{group}} -m setup
```

- Exécuter une commande sur un groupe d'hôtes en invoquant le module `command` avec des arguments :

```
ansible {{group}} -m command -a '{{commande}}'
```

- Exécuter une commande avec les privilèges d'administrateur :

```
ansible {{group}} --become --ask-become-pass -m command -a '{{commande}}'
```

- Exécuter une commande en utilisant un fichier d'inventaire personnalisé :

```
ansible {{group}} -i {{inventory_file}} -m command -a '{{commande}}'
```

Table 109. Options principales de la commande ansible

Option	Information
`-a 'arguments'`	Les arguments à passer au module.
`-b -K`	Demande un mot de passe et lance la commande avec des privilèges supérieurs.
`--user=utilisateur`	Utilise cet utilisateur pour se connecter à l'hôte cible au lieu d'utiliser l'utilisateur courant.
`--become -user=utilisateur`	Exécute l'opération en tant que cet utilisateur (défaut : `root`).
`-C`	Simulation. Ne fait pas de changement sur la cible mais la teste pour voir ce qui devrait être changé.
`-m module`	Exécute le module appelé

Tester avec le module ping

Par défaut la connexion par mot de passe n'est pas autorisée par Ansible.

Décommenter la ligne suivante de la section `[defaults]` dans le fichier de configuration `/etc/ansible/ansible.cfg` :

```
ask_pass      = True
```

Lancer un `ping` sur chacun des serveurs du groupe CentOS 7 :

```
# ansible centos7 -m ping
SSH password:
172.16.1.192 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
172.16.1.217 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Le mot de passe `root` des serveurs distants vous est demandé, ce qui pose un problème de sécurité...

4.4. Authentification par clef

L'authentification par mot de passe va être remplacée par une authentification par clefs privée/publique beaucoup plus sécurisée.

Création d'une clef SSH

La bi-clefs va être générée avec la commande `ssh-keygen` :

```
# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:RpYuJzkkaeZzve8La8Xd/8kTTE8t43DeS+L7WB26WF8 root@ansible-srv
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|     .   .       |
|    = . +       .|
|   + o *    . +.o|
|    o * S. . *o*.|
|     o * .o ..=+=|
|        o.  .oooE|
|        .+  o.*o+|
|       ...+o +o=+|
+----[SHA256]-----+
```

La clef publique peut être copiée sur les serveurs :

```
# ssh-copy-id root@172.16.1.192
# ssh-copy-id root@172.16.1.217
```

Re-commenter la ligne suivante de la section `[defaults]` dans le fichier de configuration `/etc/ansible/ansible.cfg` pour empêcher l'authentification par mot de passe :

```
#ask_pass      = True
```

Test d'authentification par clef privée

Pour le prochain test, le module `shell`, permettant l'exécution de commandes à distance, est utilisé :

```
# ansible centos7 -m shell -a "uptime"
172.16.1.192 | SUCCESS | rc=0 >>
 12:36:18 up 57 min,  1 user,  load average: 0.00, 0.00, 0.00

172.16.1.217 | SUCCESS | rc=0 >>
 12:37:07 up 57 min,  1 user,  load average: 0.00, 0.00, 0.00
```

Aucun mot de passe n'est demandé, l'authentification par clé privée/publique fonctionne !

Exemple de connexion à une instance Cloud Amazon ECS

Lors de la création d'une instance Amazon, une clef privée est créée et téléchargée sur le poste local.

Ajout de la clef dans l'agent SSH :

```
ssh-add path/to/fichier.pem
```

Lancement des `facts` sur les serveurs aws :

```
ansible aws --user=ec2-user --become -m setup
```

Pour une image ubuntu, il faudra utiliser l'utilisateur ubuntu :

```
ansible aws --user=ubuntu --become -m setup
```

4.5. Utilisation

Ansible peut être utilisé depuis l'interpréteur de commandes ou via des playbooks.

Les modules

La liste des modules classés par catégories se trouve à l'adresse http://docs.ansible.com/ansible/modules_by_category.html. Ansible en propose plus de 750 !

Un module s'invoque avec l'option `-m` de la commande ansible.

Il existe un module pour chaque besoin ou presque ! Il est donc conseillé, au lui d'utiliser le module shell, de chercher un module adapté au besoin.

Chaque catégorie de besoin dispose de son module. En voici une liste non exhaustive :

Table 110. Catégories de modules

Type	Exemples
Gestion du système	`user` (création des utilisateurs), `group` (gestion des groupes), etc.
Gestion des logiciels	`yum`, `apt`, `pip`, `npm`
Gestion des fichiers	`copy`, `fetch`, `lineinfile`, `template`, `archive`
Gestion des bases de données	`mysql`, `postgresql`, `redis`

Type	Exemples
Gestion du cloud	`amazon S3`, `cloudstack`, `openstack`
Gestion d'un cluster	`consul`, `zookeeper`
Envoyer des commandes	`shell`, `script`, `expect`
Gestion des messages	
Gestion du monitoring	
Gestion du réseau	`get_url`
Gestion des notifications	
Gestion des sources	`git`, `gitlab`

Exemple d'installation logiciel

Le module `yum` permet d'installer des logiciels sur les clients cibles :

```
# ansible centos7 -m yum -a name="httpd"
172.16.1.192 | SUCCESS => {
    "changed": true,
    "msg": "",
    "rc": 0,
    "results": [
      ...
      \n\nComplete!\n"
    ]
}
172.16.1.217 | SUCCESS => {
    "changed": true,
    "msg": "",
    "rc": 0,
    "results": [
      ...
    \n\nComplete!\n"
    ]
}
```

Le logiciel installé étant un service, il faut maintenant le démarrer avec le module `service` (CentOS 6) ou `systemd` (CentOS 7) :

```
# ansible centos7 -m systemd -a "name=httpd state=started"
172.16.1.192 | SUCCESS => {
    "changed": true,
    "name": "httpd",
    "state": "started"
}
172.16.1.217 | SUCCESS => {
    "changed": true,
    "name": "httpd",
    "state": "started"
}
```

4.6. Les playbooks

Les `playbooks` ansible décrivent une politique à appliquer à des systèmes distants, pour forcer leur configuration. Les playbooks sont écrits dans un format texte facilement compréhensible regroupant un ensemble de tâches : le format `yaml`.

En savoir plus sur le yaml : http://docs.ansible.com/ansible/YAMLSyntax.html

Syntaxe de la commande ansible-playbook

```
ansible-playbook <fichier.yml> ... [options]
```

Les options sont identiques à la commande ansible.

La commande renvoi les codes d'erreurs suivants :

Table 111. Codes de sortie de la commande ansible-playbook

Code	Erreur
`0`	OK ou aucun hôte correspondant
`1`	Erreur
`2`	Un ou plusieurs hôtes sont en échecs
`3`	Un ou plusieurs hôtes ne sont pas joignables
`4`	Erreur d'analyse
`5`	Mauvaises options ou options incomplètes
`99`	Execution interrompue par l'utilisateur
`250`	Erreur inattendue

Exemple de playbook Apache et MySQL

Le playbook suivant permet d'installer Apache et MySQL sur nos serveurs cibles :

```
---
- hosts: centos7
  remote_user: root

  tasks:
  - name: ensure apache is at the latest version
    yum: name=httpd,php,php-mysqli state=latest
  - name: ensure httpd is started
    systemd: name=httpd state=started
  - name: ensure mysql is at the latest version
    yum: name=mysql-server state=latest
  - name: ensure mysqld is started
    systemd: name=mysqld state=started
```

L'exécution du playbook s'effectue avec la commande `ansible-playbook` :

```
$ ansible-playbook test

PLAY [centos7] ***************************************************************

TASK [setup] *****************************************************************
ok: [172.16.1.192]
ok: [172.16.1.217]

TASK [ensure apache is at the latest version] ********************************
ok: [172.16.1.192]
ok: [172.16.1.217]

TASK [ensure httpd is started] ***********************************************
changed: [172.16.1.192]
changed: [172.16.1.217]

TASK [ensure mysql is at the latest version] *********************************
changed: [172.16.1.192]
changed: [172.16.1.217]

TASK [ensure mysqld is started] **********************************************
changed: [172.16.1.192]
changed: [172.16.1.217]

PLAY RECAP *******************************************************************
172.16.1.192               : ok=5    changed=3    unreachable=0    failed=0
172.16.1.217               : ok=5    changed=3    unreachable=0    failed=0
```

Exemple de préparation d'un noeud MySQL

Dans ce playbook, un serveur va être installé et configuré pour héberger une base de données MySQL.

Le playbook utilise :

- Des variables ;
- Ajoute des lignes dans le fichier `/etc/hosts` ;
- Installe et démarre MariaDB ;
- Créer une base de données, un utilisateur et lui donne tous les droits sur les bases de données.

```
---

- hosts: centos7
  become: yes
  vars:
    mysqlpackage: "mariadb-server,MySQL-python"
    mysqlservice: "mariadb"
    mysql_port: "3306"
    dbuser: "synchro"
    dbname: "mabase"
    upassword: "M!rro!r"

  tasks:
    - name: configurer le noeud 1 dans /etc/hosts
      lineinfile:
              dest: /etc/hosts
              line: "13.59.197.48 mirroir1.local.lan mirroir1"
              state: present
    - name: configurer le noeud 2 dans /etc/hosts
      lineinfile:
              dest: /etc/hosts
              line: "52.14.125.109 mirroir2.local.lan mirroir2"
              state: present
    - name: mariadb installe et a jour
      yum: name="{{ mysqlpackage }}" state=latest
    - name: mariadb est demarre
      service: name="{{ mysqlservice }}" state=started
    - name: creer la base de donnee
      mysql_db: name="{{ dbname }}" state=present
    - name: creer un utilisateur
      mysql_user: name="{{ dbuser }}" password="{{ upassword }}" priv=*.*:ALL host='%'
state=present
    - name: restart mariadb
      service: name="{{ mysqlservice }}" state=restarted
...
```

4.7. La gestion des boucles

Il existe plusieurs type de boucles sous Ansible :

- `with_items`
- `with_file`
- `with_fileglob`
- ...

Exemple d'utilisation, création de 3 utilisateurs :

```
- name: ajouter des utilisateurs
  user:
    name: "{{ item }}"
    state: present
    groups: "users"
  with_items:
     - antoine
     - xavier
     - patrick
```

4.8. Les rôles

Un rôle Ansible est une unité favorisant la réutilisabilité des playbooks.

Un squelette de rôle, servant comme point de départ du développement d'un rôle personnalisé, peut être généré par la commande `ansible-galaxy` :

```
$ ansible-galaxy init formatux
```

La commande aura pour effet de générer l'arborescence suivante pour contenir le rôle `formatux` :

```
$ tree formatux
formatux/
├── defaults
│   └── main.yml
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

La commande ansible-galaxy

La commande `ansible-galaxy` gère des rôles en utilisant le site galaxy.ansible.com.

Syntaxe de la commande ansible-galaxy

```
ansible-galaxy [import|init|install|login|remove|...]
```

Table 112. Sous-commandes de la commande ansible-galaxy

Sous-commandes	Observations
`install`	installe un rôle
`remove`	retire un ou plusieurs rôles
`init`	génère un squelette de nouveau rôle
`import`	importe un rôle depuis le site web galaxy. Nécessite un login au préalable.

Chapitre 5. Ansible Niveau 2

🎓 Objectifs

✔ Utiliser les variables ;
✔ Mettre en oeuvre des boucles et des conditions ;
✔ Gérer les fichiers ;
✔ Envoyer des notifications et réagir ;
✔ Gérer les fichiers ;
✔ Créer des tâches asynchrones.

5.1. Les variables

Plus d'informations sur http://docs.ansible.com/ansible/latest/playbooks_variables.html

Sous Ansible, il existe deux types de variables :

- la valeur simple
- le dictionnaire

Une variable peut être définie dans un playbook, dans un rôle ou depuis la ligne de commande.

Par exemple, depuis un playbook :

```
---
- hosts: apache1
  remote_user: root
  vars:
    port_http: 80
    service:
      debian: apache2
      centos: httpd
```

ou depuis la ligne de commande :

```
$ ansible-playbook deploy-http.yml --extra-vars "service=httpd"
```

Une fois définie, une variable peut être utilisée en l'appellant entre doubles accolades :

- {{ port_http }} pour la valeur simple
    - {{ service['centos'] }} ou {{ service.centos }} pour le dictionnaire.

Par exemple :

```
  tasks:
  - name: make sure apache is started
    service: name={{ service['centos'] }} state=started
```

Evidemment, il est également possible d'accéder aux variables globales d'Ansible (type d'OS, adresses IP, nom de la VM, etc.).

Externaliser les variables

Les variables peuvent être déportées dans un fichier externe au playbook, auquel cas il faudra définir ce fichier dans le playbook avec la directive `vars_files` :

```
---
- hosts: apache1
  remote_user: root
  vars_files:
    - mesvariables.yml
```

Le fichier mesvariables.yml

```
---
port_http: 80
service:
  debian: apache2
  centos: httpd
```

Afficher une variable

Pour afficher une variable, il faut activer le mode `debug` de la façon suivante :

Afficher une variable

```
- debug:
  msg: "Afficher la variable : {{ service['debian'] }}"
```

Enregistrer le retour d'une tâche

Pour enregistrer le retour d'une tâche et pouvoir y accéder plus tard, il faut utiliser le mot clef `register` à l'intérieur même de la tâche.

Utilisation d'une variable stockée

```
  tasks:
  - name: contenu de /home
    shell: ls /home
    register: homes

  - name: affiche le premier repertoire
    debug:
      var: homes.stdout_lines[0]

  - name: affiche le second repertoire
    debug:
      var: homes.stdout_lines[1]
```

Les chaînes de caractères composant la variable enregistrée sont accessibles via la valeur `stdout` (ce qui permet de faire des choses comme `homes.stdout.find("core") != -1`), de les exploiter en utilisant une boucle (voir `with_items`), ou tout simplement par leurs indices comme vu dans l'exemple précédent.

5.2. La gestion des boucles

Plus d'informations sur http://docs.ansible.com/ansible/latest/playbooks_loops.html

Il existe plusieurs types de boucles sous Ansible, en fonction de l'objet que vous voulez manipuler :

- `with_items`
- `with_file`
- `with_dict`
- `with_fileglob`
- ...

Exemple d'utilisation, création de 3 utilisateurs :

```
- name: ajouter des utilisateurs
  user:
    name: "{{ item }}"
    state: present
    groups: "users"
  with_items:
     - antoine
     - kevin
     - nicolas
```

Nous pouvons reprendre l'exemple vu durant l'étude des variables stockées pour l'améliorer :

Utilisation d'une variable stockée

```
  tasks:
  - name: contenu de /home
    shell: ls /home
    register: homes

  - name: affiche le nom des repertoires
    debug:
      msg: "Dossier => {{ item }}"
    with_items:
      - "{{ homes.stdout_lines }}"
```

Une fois un dictionnaire créé, celui-ci peut être parcouru en utilisant la variable `item` comme index :

```
---
- hosts: ansiblecli
  remote_user: ansible
  become: true
  vars:
    users:
      antoine:
        group: users
        state: present
      erwan:
        group: users
        state: absent

  tasks:

  - name: creer les utilisateurs
    user:
      name: "{{ item }}"
      group: "{{ users[item]['group'] }}"
      state: "{{ users[item]['state'] }}"
    with_items: "{{ users }}"
```

5.3. Les conditions

Plus d'informations sur http://docs.ansible.com/ansible/latest/playbooks_conditionals.html

L'instruction `when` est très pratique dans de nombreux cas : ne pas effectuer certaines actions sur certains type de serveur, si un fichier ou un n'utilisateur n'existe pas, etc.

Derrière l'instruction `when` les variables ne nécessitent pas de doubles accolades (il s'agit en fait d'expressions Jinja2...).

```
tasks:
  - name: "ne redemarre que les OS Debian"
    command: /sbin/shutdown -r now
    when: ansible_os_family == "Debian"
```

Les conditions peuvent être regroupées avec des parenthèses :

```
tasks:
  - name: "ne redémarre que les CentOS version 6 et Debian version 7"
    command: /sbin/shutdown -r now
    when: (ansible_distribution == "CentOS" and ansible_distribution_major_version ==
"6") or
          (ansible_distribution == "Debian" and ansible_distribution_major_version ==
"7")
```

Les conditions correspondant à un ET logique peuvent être fournies sous forme de liste :

```
tasks:
  - name: "ne redémarre que les CentOS 6"
    command: /sbin/shutdown -r now
    when:
      - ansible_distribution == "CentOS"
      - ansible_distribution_major_version == "6"
```

Le résultat de la variable peut aussi être utilisé conjointement aux conditions `when` et son absence de contenu évalué :

```
  tasks:

  - name: check if /data exists
    command: find / -maxdepth 1 -type d -name data
    register: datadir

  - name: print warning if /data does not exist
    debug: msg="Le dossier /data n'existe pas..."
    when: datadir.stdout == ""
```

5.4. La gestion des fichiers

Plus d'informations sur http://docs.ansible.com/ansible/latest/list_of_files_modules.html

En fonction de votre besoin, vous allez être amenés à utiliser différents modules Ansible pour modifier les fichiers de configuration du système.

Le module `ini_file`

Lorsqu'il s'agit de modifier un fichier de type INI (section entre [] puis paires de clef=valeur), le plus simple est d'utiliser le module ini_file.

Le module nécessite :

- La valeur de la section
- Le nom de l'option
- La nouvelle valeur

Exemple d'utilisation :

```
- name: change value on inifile
  ini_file: dest=/path/to/file.ini section=SECTIONNAME option=OPTIONNAME value
=NEWVALUE
```

Le module `lineinfile`

Pour s'assurer qu'une ligne est présente dans un fichier, ou lorsqu'une seule ligne d'un fichier doit être ajoutée ou modifiée.

Voir http://docs.ansible.com/ansible/latest/lineinfile_module.html.

Dans ce cas, la ligne à modifier d'un fichier sera retrouvée à l'aide d'une regexp.

Par exemple, pour s'assurer que la ligne commençant par 'SELINUX=' dans le fichier /etc/selinux/config contiennent la valeur `enforcing` :

```
- lineinfile:
    path: /etc/selinux/config
    regexp: '^SELINUX='
    line: 'SELINUX=enforcing'
```

Le module `copy`

Lorsqu'un fichier doit être copié depuis le serveur Ansible vers un ou plusieurs hosts, dans ce cas il est préférable d'utiliser le module `copy` :

```
- copy:
    src: /data/ansible/sources/monfichier.conf
    dest: /etc/monfichier.conf
    owner: root
    group: root
    mode: 0644
```

Le module `fetch`

Lorsqu'un fichier doit être copié depuis un serveur distant vers le serveur local.

Ce module fait l'inverse du module `copy`.

```
- fetch:
    src: /etc/monfichier.conf
    dest: /data/ansible/backup/monfichier-{{ inventory_hostname }}.conf
    flat: yes
```

Le module `template`

Ansible et son module `template` utilisent le système de template Jinja2 (http://jinja.pocoo.org/docs/) pour générer des fichiers sur les hôtes cibles.

```
- template:
    src: /data/ansible/templates/monfichier.j2
    dest: /etc/monfichier.conf
    owner: root
    group: root
    mode: 0644
```

Il est possible d'ajouter une étape de validation si le service ciblé le permet (par exemple apache avec la commande `apachectl -t`) :

```
- template:
    src: /data/ansible/templates/vhost.j2
    dest: /etc/httpd/sites-available/vhost.conf
    owner: root
    group: root
    mode: 0644
    validate: '/usr/sbin/apachectl -t'
```

Le module `get_url`

Pour télécharger vers un ou plusieurs hôtes des fichiers depuis un site web ou ftp.

```
- get_url:
    url: http://site.com/archive.zip
    dest: /tmp/archive.zip
    mode: 0640
    checksum: sha256:f772bd36185515581aa9a2e4b38fb97940ff28764900ba708e68286121770e9a
```

En fournissant un checksum du fichier, ce dernier ne sera pas re-téléchargé s'il est déjà présent à l'emplacement de destination et que son checksum correspond à la valeur fournie.

5.5. Les handlers

Plus d'informations sur http://docs.ansible.com/ansible/latest/playbooks_intro.html#handlers-running-operations-on-change

Les handlers permettent de lancer des opérations, comme relancer un service, lors de changements.

Un module, étant idempotent, un playbook peut détecter qu'il y a eu un changement significatif sur un système distant, et donc déclencer une opération en réaction à ce changement. Une notification est envoyée à la fin d'un bloc de tâche du playbook, et l'opération en réaction ne sera déclenchée qu'une seule fois même si plusieurs tâches différentes envoient cette notification.

Par exemple, plusieurs tâches peuvent indiquer que le service httpd nécessite une relance à cause d'un changement dans ses fichiers de configuration. Mais le service ne sera redémarré qu'une seule fois pour éviter les multiples démarrages non nécessaires.

```
- name: template configuration file
  template: src=modele-site.j2 dest=/etc/httpd/sites-availables/site-test.conf
  notify:
     - restart memcached
     - restart httpd
```

Un handler est une sorte de tâche référencé par un nom unique global :

- Il est activé par ou un plusieurs notifiers.
- Il ne se lance pas immédiatement, mais attend que toutes les tâches soient complètes pour s'exécuter.

Exemple de handlers

```
handlers:
    - name: restart memcached
      service: name=memcached state=restarted
    - name: restart httpd
      service: name=httpd state=restarted
```

Depuis la version 2.2 d'Ansible, les handlers peuvent se mettre directement à l'écoute des tâches :

```
handlers:
    - name: restart memcached
      service: name=memcached state=restarted
      listen: "restart web services"
    - name: restart apache
      service: name=apache state=restarted
      listen: "restart web services"

tasks:
    - name: restart everything
      command: echo "this task will restart the web services"
      notify: "restart web services"
```

5.6. Les rôles

Plus d'informations sur http://docs.ansible.com/ansible/latest/playbooks_reuse_roles.html

Un rôle Ansible est une unité favorisant la réutilisabilité des playbooks.

Un squelette de rôle, servant comme point de départ du développement d'un rôle personnalisé, peut être généré par la commande `ansible-galaxy` :

```
$ ansible-galaxy init claranet
```

La commande aura pour effet de générer l'arborescence suivante pour contenir le rôle `claranet` :

```
$ tree claranet
claranet/
├──── defaults
│     └──── main.yml
├──── handlers
│     └──── main.yml
├──── meta
│     └──── main.yml
├──── README.md
├──── tasks
│     └──── main.yml
├──── tests
│     ├──── inventory
│     └──── test.yml
└──── vars
      └──── main.yml
```

Les rôles permettent de s'affranchir totalement de l'inclusion des fichiers. Plus besoin de spécifier les chemins d'accès aux fichiers, les directives `include` dans les playbook. Il suffit de spécifier une tâche, ansible s'occupe des inclusions.

La commande ansible-galaxy

La commande `ansible-galaxy` gère des rôles en utilisant le site galaxy.ansible.com.

Syntaxe de la commande ansible-galaxy

```
ansible-galaxy [import|init|install|login|remove|...]
```

Table 113. Sous-commandes de la commande ansible-galaxy

Sous-commandes	Observations
`install`	installe un rôle
`remove`	retire un ou plusieurs rôles
`init`	génère un squelette de nouveau rôle
`import`	importe un rôle depuis le site web galaxy. Nécessite un login au préalable.

5.7. Les tâches asynchrones

Plus d'informations sur http://docs.ansible.com/ansible/latest/playbooks_async.html

Par défaut, les connexions SSH vers les hosts restent ouvertes durant l'exécution des différentes tâches d'un playbook sur l'ensemble des noeuds.

Cela peut poser quelques problèmes, notamment :

- si la durée d'exécution de la tâche est supérieure au timeout de la connexion SSH
- si la connexion est interrompue durant l'action (reboot du serveur par exemple)

Dans ce cas, il faudra basculer en mode asynchrone et spécifier un temps maximum d'exécution ainsi que la fréquence (par défaut à 10s) avec laquelle vous allez vérifier le status de l'hôte.

En spécifiant une valeur de poll à 0, Ansible va exécuter la tâche et continuer sans se soucier du résultat.

Voici un exemple mettant en oeuvre les tâches asynchrones, qui permet de relancer un serveur et d'attendre que le port 22 soit de nouveau joignable :

```
# On attend 2s et on lance un reboot
- name: Reboot system
  shell: sleep 2 && shutdown -r now "Ansible reboot triggered"
  async: 1
  poll: 0
  ignore_errors: true
  become: true
  changed_when: False

  # On attend que ça revienne
  - name: Waiting for server to restart (10 mins max)
    wait_for:
      host: "{{ inventory_hostname }}"
      port: 22
      delay: 30
      state: started
      timeout: 600
    delegate_to: localhost
```

5.8. Connexion à une instance Cloud Amazon ECS

Lors de la création d'une instance Amazon, une clef privée est créée et téléchargée sur le poste local.

Ajout de la clef dans l'agent SSH :

```
ssh-add path/to/fichier.pem
```

Lancement des `facts` sur les serveurs aws :

```
ansible aws --user=ec2-user --become -m setup
```

Pour une image ubuntu, il faudra utiliser l'utilisateur ubuntu :

```
ansible aws --user=ubuntu --become -m setup
```

Chapitre 6. Ansible Ansistrano

🎓 Objectifs

✔ Mettre en oeuvre Ansistrano ;
✔ Configurer Ansistrano ;
✔ Utiliser des dossiers et fichiers partagés entre versions déployées ;
✔ Déployer différentes versions d'un site depuis git ;
✔ Réagir entre les étapes de déploiement.

6.1. Introduction

Ansistrano est un rôle Ansible pour déployer facilement des applications PHP, Python, etc. Il se base sur le fonctionnement de Capistrano

Ansistrano nécessite pour fonctionner :

- Ansible sur la machine de déploiement,
- `rsync` ou `git` sur la machine cliente.

Il peut télécharger le code source depuis `rsync`, `git`, `scp`, `http`, `S3`, ...

Dans le cadre de notre exemple de déploiement, nous allons utiliser le protocole `git`.

Ansistrano déploie les applications en suivant ces 5 étapes :

- `Setup` : création de la structure de répertoire pour accueillir les releases
- `Update Code` : téléchargement de la nouvelle release sur les cibles
- `Symlink Shared` et `Symlink` : après avoir déployé la nouvelle release, le lien symbolique `current` est modifié pour pointer vers cette nouvelle release
- `Clean Up` : pour faire un peut de nettoyage (suppression des anciennes versions)

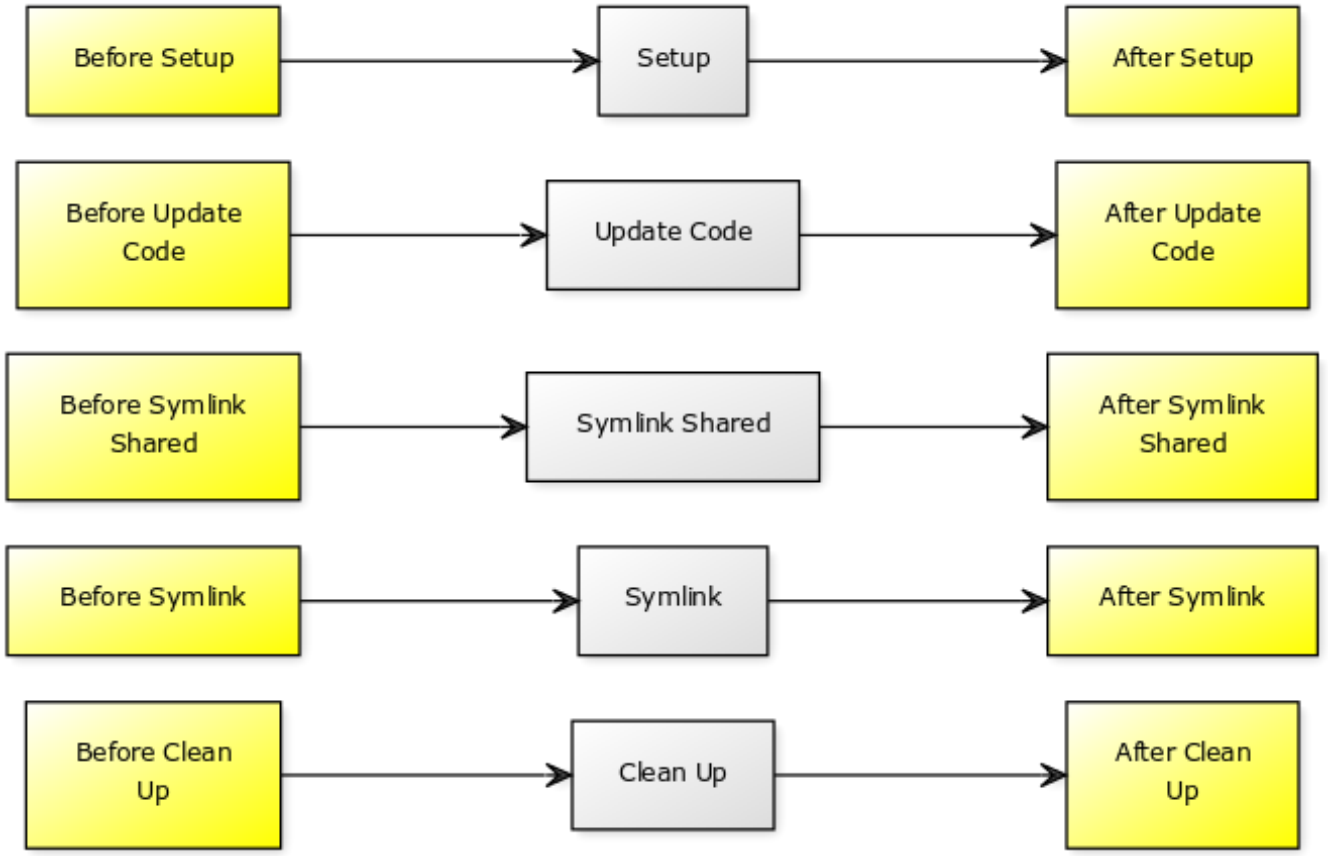


Figure 78. Etapes d'un déploiement

Le squelette d'un déploiement avec ansistrano ressemble à :

Squelette d'un déploiement avec ansistrano

```
/var/www/site/
├─── current -> ./releases/20180821070043Z
├─── releases
│    └─── 20180821070043Z
│        ├─── css -> ../../shared/css/
│        ├─── img -> ../../shared/img/
│        └─── REVISION
├─── repo
└─── shared
    ├─── css/
    └─── img/
```

Vous retrouverez toute la documentation ansistrano sur son dépôt Github.

6.2. Module 1 : Prise en main de la plateforme

Exercice 1.1 : Déploiement de la plateforme

Vous allez travailler sur 2 VM :

- La VM de gestion :
  - Vous devrez installer ansible et déployer le rôle `ansistrano-deploy`
- La VM cliente :
  - Cette VM n'a aucun logiciel spécifique.
  - Vous devrez installer Apache et déployer le site du client

6.3. Module 2 : Déployer le serveur Web

Exercice 2.1 : Utiliser le rôle `geerlingguy.apache` pour configurer le serveur

Pour gérer notre serveur web, nous allons utiliser le rôle `geerlingguy.apache` qu'il faut installer sur le serveur :

```
[ansiblesrv] $ ansible-galaxy install geerlingguy.apache
```

Une fois le rôle installé, nous allons pouvoir créer la première partie de notre playbook, qui va :

- Installer Apache,
- Créer un dossier cible pour notre `vhost`,
- Créer un `vhost` par défaut,
- Démarrer ou redémarrer Apache.

Considérations techniques :

- Nous allons déployer notre site dans le dossier `/var/www/site/`.
- Comme nous le verrons plus loin, `ansistrano` va créer un lien symbolique `current` vers le dossier de la release en cours.
- Le code source à déployer contient un dossier `html` sur lequel le vhost devra pointer. Sa `DirectoryIndex` est `index.htm`.
- Le déploiement se faisant par `git`, le paquet sera installé.

La cible de notre vhost sera donc : `/var/www/site/current/html`.

Playbook de configuration du serveur playbook-config-serveur.yml

```
---
- hosts: ansiblecli
  become: yes
  vars:
    dest: "/var/www/site/"
    apache_global_vhost_settings: |
      DirectoryIndex index.php index.htm
    apache_vhosts:
      - servername: "website"
        documentroot: "{{ dest }}current/html"

  tasks:

    - name: create directory for website
      file:
        path: /var/www/site/
        state: directory
        mode: 0755

    - name: install git
      package:
        name: git
        state: latest

  roles:
    - { role: geerlingguy.apache }
```

Le playbook peut être appliqué au serveur :

```
[ansiblesrv] $ ansible-playbook -i hosts playbook-config-serveur.yml
```

Notez l'exécution des tâches suivantes :

```
TASK [geerlingguy.apache : Ensure Apache is installed on RHEL.] ****************
TASK [geerlingguy.apache : Configure Apache.] **********************************
TASK [geerlingguy.apache : Add apache vhosts configuration.] *******************
TASK [geerlingguy.apache : Ensure Apache has selected state and enabled on boot.] ***
RUNNING HANDLER [geerlingguy.apache : restart apache] **************************
```

Le rôle `geerlingguy.apache` nous facililte grandement la tâche en s'occupant de l'installation et la configuration d'Apache.

6.4. Module 3 : Déployer le logiciel

Notre serveur étant maintenant configuré, nous allons pouvoir déployer l'applicatif.

Pour cela, nous allons utiliser le rôle `ansistrano.deploy` dans un second playbook dédié au déploiement applicatif (pour plus de lisibilité).

Exercice 3.1 : Installer le rôle `ansistrano-deploy`

```
[ansiblesrv] $ ansible-galaxy install ansistrano.deploy
```

Exercice 3.2 : Utiliser le rôle `ansistrano-deploy` pour déployer le logiciel

Les sources du logiciel se trouvent dans le dépôt :

- sur framagit https://framagit.org/alemorvan/demo-ansible.git accessible depuis internet.

Nous allons créer un playbook `playbook-deploy.yml` pour gérer notre déploiement :

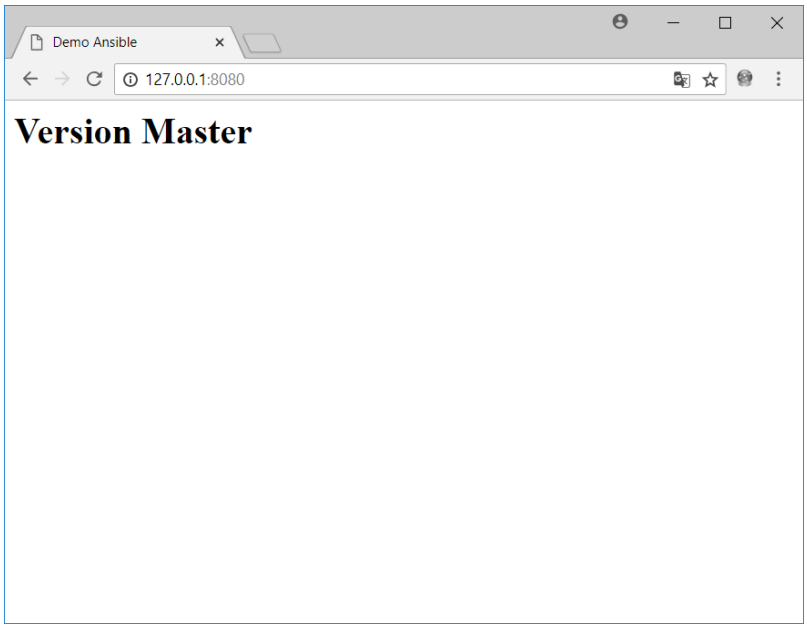
```
---
- hosts: ansiblecli
  become: yes
  vars:
    dest: "/var/www/site/"
    ansistrano_deploy_via: "git"
    ansistrano_git_repo: https://framagit.org/alemorvan/demo-ansible.git
    ansistrano_deploy_to: "{{ dest }}"

  roles:
     - { role: ansistrano.deploy }
```

```
[ansiblesrv] $ ansible-playbook -i hosts playbook-deploy.yml
```

Exercice 3.3 : Visualiser le résultat dans un navigateur

Une redirection du port `http` a été mis en place pour accéder depuis l'extérieur à votre serveur.

- Ouvrir un navigateur web vers l'url http://@IPPUBLIQUE:PORTREDIR/ :



Et voir apparaître le déploiement fonctionnel de la branche `master` de notre dépôt.

Exercice 3.4 : Vérification sur le serveur

- Se connecter en ssh sur le serveur client :

```
[ansiblesrv] $ ssh ansible@192.168.10.11
```

- Faire un `tree` sur le repertoire `/var/www/site/` :

```
[ansiblecli] $ tree /var/www/site/
/var/www/site/
├── current -> ./releases/20180821070043Z
├── releases
│   └── 20180821070043Z
│       ├── html
│       │   └── index.htm
│       └── REVISION
├── repo
│   └── html
│       └── index.htm
└── shared
```

Notez :

- le lien symbolique `current` vers la release `./releases/20180821070043Z`
- la présence d'un dossier `shared`
- la présence du dépôt git dans `./repo/`

- Depuis le serveur ansible, relancer **3** fois le déploiement, puis vérifier sur le client.

```
[ansiblecli] $ tree /var/www/site/
/var/www/site/
├── current -> ./releases/20180821112348Z
├── releases
│   ├── 20180821070043Z
│   │   ├── html
│   │   │   └── index.htm
│   │   └── REVISION
│   ├── 20180821112047Z
│   │   ├── html
│   │   │   └── index.htm
│   │   └── REVISION
│   └── 20180821112100Z
│       ├── html
│       │   └── index.htm
│       └── REVISION
│   ├── 20180821112348Z
│   │   ├── html
│   │   │   └── index.htm
│   │   └── REVISION
├── repo
│   └── html
│       └── index.htm
└── shared
```

Notez :

- `ansistrano` a conservé les 4 dernières releases,
- le lien `current` pointe maintenant vers la dernière release exécutée

6.5. Module 4 : Ansistrano

Exercice 4.1 : Limiter le nombre de releases

La variable `ansistrano_keep_releases` permet de spécifier le nombre de releases à conserver.

- En utilisant la variable `ansistrano_keep_releases`, ne conserver que 3 releases du projet. Vérifier.

```
---
- hosts: ansiblecli
  become: yes
  vars:
    dest: "/var/www/site/"
    ansistrano_deploy_via: "git"
    ansistrano_git_repo: https://framagit.org/alemorvan/demo-ansible.git
    ansistrano_deploy_to: "{{ dest }}"
    ansistrano_keep_releases: 3

  roles:
     - { role: ansistrano.deploy }
```

```
---
[ansiblesrv] $ ansible-playbook -i hosts playbook-deploy.yml
```

Sur le client :

```
[ansiblecli] $ tree /var/www/site/
/var/www/site/
├── current -> ./releases/20180821113223Z
├── releases
│   └── 20180821112100Z
│       ├── html
│       │   └── index.htm
│       └── REVISION
│   ├── 20180821112348Z
│   │   ├── html
│   │   │   └── index.htm
│   │   └── REVISION
│   └── 20180821113223Z
│       ├── html
│       │   └── index.htm
│       └── REVISION
├── repo
│   └── html
│       └── index.htm
└── shared
```

Exercice 4.2 : Utiliser des `shared_paths` et `shared_files`

```
---
- hosts: ansiblecli
  become: yes
  vars:
    dest: "/var/www/site/"
    ansistrano_deploy_via: "git"
    ansistrano_git_repo: https://framagit.org/alemorvan/demo-ansible.git
    ansistrano_deploy_to: "{{ dest }}"
    ansistrano_keep_releases: 3
    ansistrano_shared_paths:
      - "img"
      - "css"
    ansistrano_shared_files:
      - "logs"

  roles:
     - { role: ansistrano.deploy }
```

Sur le client, créer le fichier `logs` dans le répertoire `shared` :

```
sudo touch /var/www/site/shared/logs
```

Puis lancer l'exécution du job :

```
TASK [ansistrano.deploy : ANSISTRANO | Ensure shared paths targets are absent]
*********************************************************
ok: [192.168.10.11] => (item=img)
ok: [192.168.10.11] => (item=css)
ok: [192.168.10.11] => (item=logs/log)

TASK [ansistrano.deploy : ANSISTRANO | Create softlinks for shared paths and files]
*************************************************
changed: [192.168.10.11] => (item=img)
changed: [192.168.10.11] => (item=css)
changed: [192.168.10.11] => (item=logs)
```

Sur le client :

```
[ansiblecli] $  tree -F /var/www/site/
/var/www/site/
├── current -> ./releases/20180821120131Z/
├── releases
│   ├── 20180821112348Z/
│   │   ├── html/
│   │   │   └── index.htm
│   │   └── REVISION
│   └── 20180821113223Z/
│       ├── html/
│       │   └── index.htm
│       └── REVISION
│   └── 20180821120131Z/
│       ├── css -> ../../shared/css/
│       ├── html
│       │   └── index.htm
│       ├── img -> ../../shared/img/
│       ├── logs -> ../../shared/logs
│       └── REVISION
├── repo/
│   └── html
│       └── index.htm
└── shared/
    ├── css/
    ├── img/
    └── logs
```

Notez que la dernière release contient :

- Un répertoire `css`, un répertoire `img`, et un fichier `logs`
- Des liens symboliques :
  - du dossier `/var/www/site/releases/css/` vers le dossier `../../shared/css/`.
  - du dossier `/var/www/site/releases/img/` vers le dossier `../../shared/img/`.
  - du fichier `/var/www/site/releases/logs/` vers le fichier `../../shared/logs`.

Dès lors, les fichiers contenus dans ces 2 dossiers et le fichier `logs` sont toujours accessibles via les chemins suivants :

- `/var/www/site/current/css/`,
- `/var/www/site/current/img/`,
- `/var/www/site/current/logs`,

mais surtout ils seront conservés d'une release à l'autre.

Exercice 4.3 : Utiliser un sous répertoire du dépôt pour le déploiement

Dans notre cas, le dépôt contient un dossier `html`, qui contient les fichiers du site.

- Pour éviter ce niveau supplémentaire de répertoire, utiliser la variable `ansistrano_git_repo_tree` en précisant le path du sous répertoire à utiliser. Ne pas oublier de modifier la configuration d'apache pour prendre en compte ce changement !

Playbook de configuration du serveur playbook-config-serveur.yml

```
---
- hosts: ansiblecli
  become: yes
  vars:
    dest: "/var/www/site/"
    apache_global_vhost_settings: |
      DirectoryIndex index.php index.htm
    apache_vhosts:
      - servername: "website"
        documentroot: "{{ dest }}current/" ①

  tasks:

    - name: create directory for website
      file:
        path: /var/www/site/
        state: directory
        mode: 0755

    - name: install git
      package:
        name: git
        state: latest

  roles:
    - { role: geerlingguy.apache }
```

① Modifier cette ligne

Playbook de déploiement playbook-deploy.yml

```
---
- hosts: ansiblecli
  become: yes
  vars:
    dest: "/var/www/site/"
    ansistrano_deploy_via: "git"
    ansistrano_git_repo: https://framagit.org/alemorvan/demo-ansible.git
    ansistrano_deploy_to: "{{ dest }}"
    ansistrano_keep_releases: 3
    ansistrano_shared_paths:
      - "img"
      - "css"
    ansistrano_shared_files:
      - "log"
    ansistrano_git_repo_tree: 'html' ①

  roles:
     - { role: ansistrano.deploy }
```

① Modifier cette ligne

- Vérifier sur le client :

```
[ansiblecli] $  tree -F /var/www/site/
/var/www/site/
├── current -> ./releases/20180821120131Z/
├── releases
│   └── 20180821113223Z/
│   │   ├── html/
│   │   │   └── index.htm
│   │   └── REVISION
│   └── 20180821120131Z/
│   │   ├── css -> ../../shared/css/
│   │   ├── html
│   │   │   └── index.htm
│   │   ├── img -> ../../shared/img/
│   │   ├── logs -> ../../shared/logs
│   │   └── REVISION
│   └── 20180821130124Z/
│       ├── css -> ../../shared/css/
│       ├── img -> ../../shared/img/
│       ├── index.htm  ①
│       ├── logs -> ../../shared/logs
│       └── REVISION
├── repo/
│   └── html
│       └── index.htm
└── shared/
    ├── css/
    ├── img/
    └── logs
```

① Notez l'absence du dossier `html`

6.6. Module 5 : La gestion des branches ou des tags git

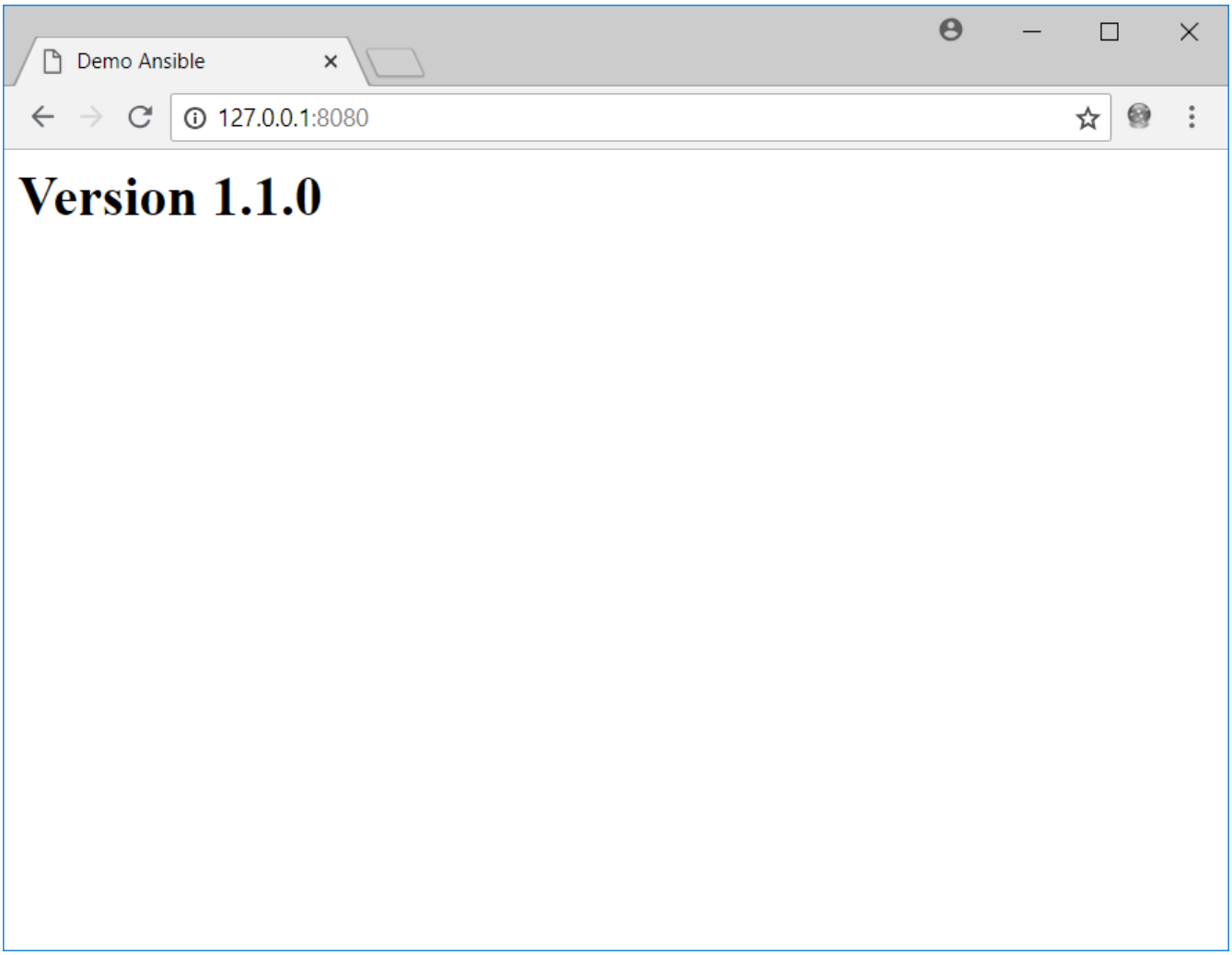
La variable `ansistrano_git_branch` permet de préciser une `branch` ou un `tag` à déployer.

Exercice 5.1 : Déployer une branche

- Déployer la branche `releases/v1.1.0` :

```
---
- hosts: ansiblecli
  become: yes
  vars:
    dest: "/var/www/site/"
    ansistrano_deploy_via: "git"
    ansistrano_git_repo: https://framagit.org/alemorvan/demo-ansible.git
    ansistrano_deploy_to: "{{ dest }}"
    ansistrano_keep_releases: 3
    ansistrano_shared_paths:
      - "img"
      - "css"
    ansistrano_shared_files:
      - "log"
    ansistrano_git_repo_tree: 'html'
    ansistrano_git_branch: 'releases/v1.1.0'

  roles:
     - { role: ansistrano.deploy }
```

Vous pouvez vous amuser, durant le déploiement, à rafraichir votre navigateur, pour voir en 'live' le changement s'effectuer.

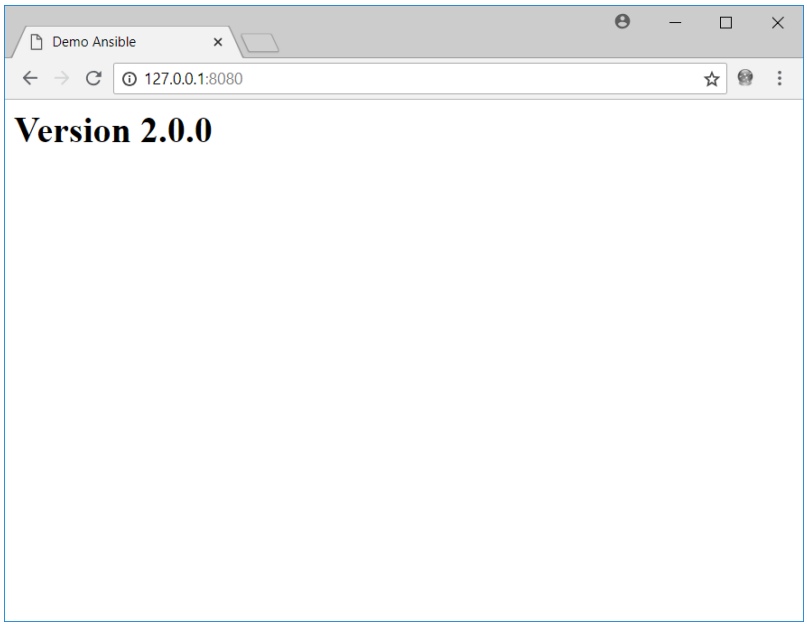


Exercice 5.2 : Déployer un tag

- Déployer le tag `v2.0.0` :

```
---
- hosts: ansiblecli
  become: yes
  vars:
    dest: "/var/www/site/"
    ansistrano_deploy_via: "git"
    ansistrano_git_repo: https://framagit.org/alemorvan/demo-ansible.git
    ansistrano_deploy_to: "{{ dest }}"
    ansistrano_keep_releases: 3
    ansistrano_shared_paths:
      - "img"
      - "css"
    ansistrano_shared_files:
      - "log"
    ansistrano_git_repo_tree: 'html'
    ansistrano_git_branch: 'v2.0.0'

  roles:
     - { role: ansistrano.deploy }
```

Vous pouvez vous amuser, durant le déploiement, à rafraichir votre navigateur, pour voir en 'live' le changement s'effectuer.



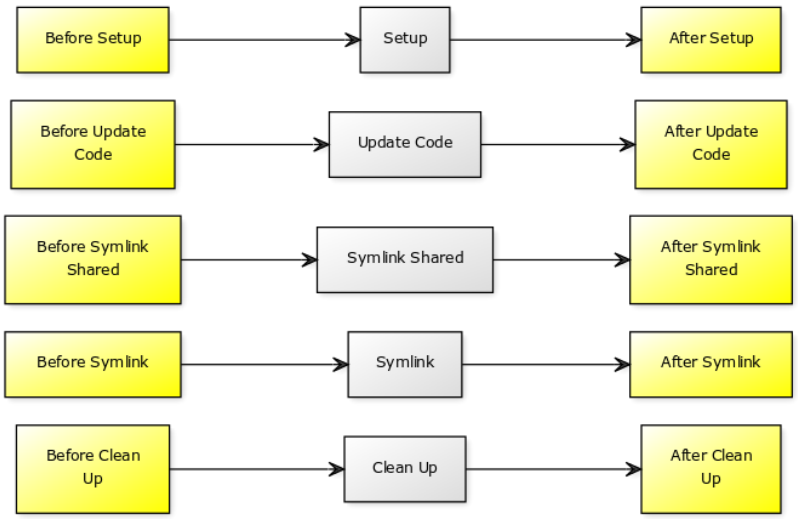
6.7. Module 6 : Actions entre les étapes de déploiement

Un déploiement avec Ansistrano respecte les étapes suivantes :

- `Setup`
- `Update Code`

- `Symlink Shared`
- `Symlink`
- `Clean Up`

Il est possible d'intervenir avant et après chacune de ses étapes.



Un playbook peut être inclu par les variables prévues à cet effet :

- ansistrano_before_<task>_tasks_file
- ou ansistrano_after_<task>_tasks_file

Exercice 6.1 : Envoyer un mail en début de MEP

```
---
- hosts: ansiblecli
  become: yes
  vars:
    dest: "/var/www/site/"
    ansistrano_deploy_via: "git"
    ansistrano_git_repo: https://framagit.org/alemorvan/demo-ansible.git
    ansistrano_deploy_to: "{{ dest }}"
    ansistrano_keep_releases: 3
    ansistrano_shared_paths:
      - "img"
      - "css"
    ansistrano_shared_files:
      - "logs"
    ansistrano_git_repo_tree: 'html'
    ansistrano_git_branch: 'v2.0.0'
    ansistrano_before_setup_tasks_file: "{{ playbook_dir }}/deploy/before-setup-
tasks.yml"

  roles:
     - { role: ansistrano.deploy }
```

Le fichier deploy/before-setup-tasks.yml

```
---
- name: Envoyer un mail
  mail:
    subject: Debut de MEP sur {{ ansible_hostname }}.
  delegate_to: localhost
```

```
TASK [ansistrano.deploy : include]
*********************************************************************************************
included: /home/ansible/deploy/before-setup-tasks.yml for 192.168.10.11

TASK [ansistrano.deploy : Envoyer un mail]
*********************************************************************************************
ok: [192.168.10.11 -> localhost]
```

```
[root] # mailx
Heirloom Mail version 12.5 7/5/10.  Type ? for help.
"/var/spool/mail/root": 1 message 1 new
>N  1 root@localhost.local  Tue Aug 21 14:41  28/946   "Debut de MEP sur localhost."
```

Exercice 6.2 : Redémarrer apache en fin de MEP

```
---
- hosts: ansiblecli
  become: yes
  vars:
    dest: "/var/www/site/"
    ansistrano_deploy_via: "git"
    ansistrano_git_repo: https://framagit.org/alemorvan/demo-ansible.git
    ansistrano_deploy_to: "{{ dest }}"
    ansistrano_keep_releases: 3
    ansistrano_shared_paths:
      - "img"
      - "css"
    ansistrano_shared_files:
      - "logs"
    ansistrano_git_repo_tree: 'html'
    ansistrano_git_branch: 'v2.0.0'
    ansistrano_before_setup_tasks_file: "{{ playbook_dir }}/deploy/before-setup-
tasks.yml"
    ansistrano_after_symlink_tasks_file: "{{ playbook_dir }}/deploy/after-symlink-
tasks.yml"

  roles:
     - { role: ansistrano.deploy }
```

Le fichier deploy/after-symlink-tasks.yml

```
---
- name: restart apache
  service:
    name: httpd
    state: restarted
```

```
TASK [ansistrano.deploy : include]
*********************************************************************************
included: /home/ansible/deploy/after-symlink-tasks.yml for 192.168.10.11

TASK [ansistrano.deploy : restart apache]
*********************************************************************************
changed: [192.168.10.11]
```

Chapitre 7. Ordonnanceur centralisé Rundeck

🎓 Objectifs

✔ installer un serveur d'ordonnancement Rundeck ;
✔ créer un premier projet et une tâche simple.

Rundeck est un ordonnanceur centralisé open source (licence Apache) écrit en Java.

Depuis l'interface de Rundeck, il est possible de gérer les différents travaux (commandes ou scripts) à exécuter sur les serveurs distants.

Fonctionnalités :

- Ordonnancement des tâches selon un scénario ;
- Exécution de scripts/tâches distantes à la demande ou de manière planifiée ;
- Notifications ;
- Interface CLI (ligne de commande) et API.

Rundeck peut être intégré aux autres outils de gestion de configuration comme Puppet, Ansible et d'intégration continue comme Jenkins, etc.

La connexion avec les hôtes clients est gérée en SSH.

7.1. Installation

Rundeck utilisant le protocle SSH pour se connecter aux systèmes distants, un compte avec les droits sudo est nécessaire sur chacun des stations clientes.

- sur Debian 8 (Jessie) :

Rundeck étant écrit en Java, l'installation du JDK est nécessaire :

```
# apt-get install openjdk-7-jdk
```

Rundeck peut être téléchargé puis installé :

```
# wget http://dl.bintray.com/rundeck/rundeck-deb/rundeck-2.6.7-1-GA.deb
# dpkg -i ./rundeck-2.6.7-1-GA.deb
```

- Sur CentOS 7 :

Rundeck étant écrit en Java, l'installation du JDK est nécessaire :

```
# yum install java-1.8.0
```

Rundeck peut être téléchargé puis installé :

```
# rpm -Uvh http://repo.rundeck.org/latest.rpm
# yum install rundeck
```

7.2. Configuration

Par défaut, Rundeck n'est configuré en écoute que sur l'adresse de boucle interne (`localhost`).

Si vous ne souhaitez pas mettre en place un proxy inverse (Apache, Nginx ou HAProxy), éditez les fichiers `/etc/rundeck/framework.properties` et `/etc/rundeck/rundeck-config.properties` et remplacer `localhost` par l'adresse IP du serveur, pour mettre en écoute Rundeck sur son interface réseau :

Modification du fichier /etc/rundeck/framework.properties

```
framework.server.url = http://xxx.xxx.xxx.xxx:4440
```

Modification du fichier /etc/rundeck/rundeck-config.properties

```
grails.serverURL=http://xxx.xxx.xxx.xxx:4440
```

Remplacer `xxx.xxx.xxx.xxx` par l'adresse IP publique du serveur.

Rundeck est ensuite lancé par systemctl :

```
# systemctl start rundeckd
# systemctl enable rundeckd
```

L'interface d'administration de Rundeck est accessible depuis un navigateur internet à l'adresse `http://your_server:4440`.

7.3. Utiliser RunDeck

Le compte par défaut pour se connecter à l'interface web est `admin` (mot de passe : `admin`). Pensez à changer le mot de passe le plus rapidement possible.

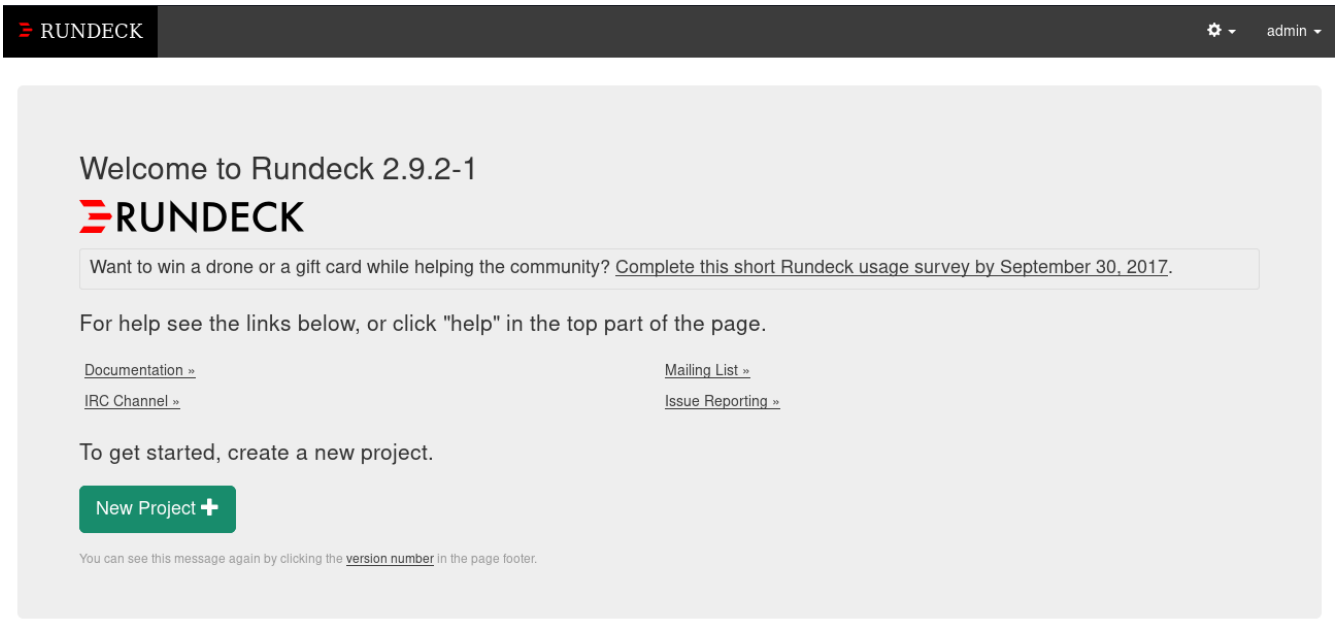


Figure 79. La page d'accueil de RunDeck

Créer un projet

Un nouveau projet peut être ajouté en cliquant sur le lien `Nouveau projet`.

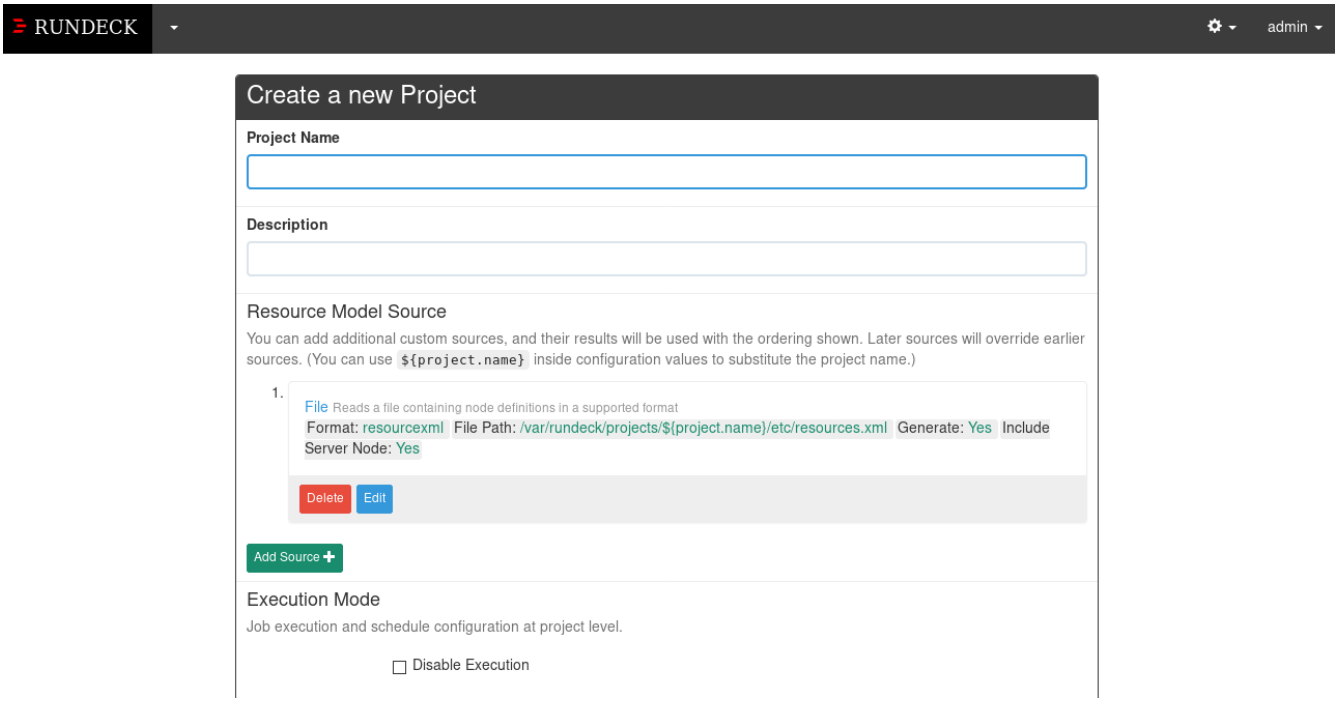


Figure 80. L'assistant de création de nouveau projet Rundeck

Vous devez fournir au minimum un nom de projet.

Dans la section `Resource Model Source`, cliquer sur le bouton `Edit` et choisir `Require File Exists `, puis cliquer sur `Save`.

Dans la section `Default Node Executor`, choisir `password` pour l'authentification par SSH (pour une meilleure gestion de la sécurité, l'utilisation d'une clef SSH est recommandée).

Cliquer sur `Create` pour créer le projet.

Créer une tâche

Le serveur est prêt à recevoir une première tâche. Cette tâche consiste en une connection SSH pour lancer une commande distante.

- Cliquer sur `Create a new job` et saisir un nom pour la tâche.

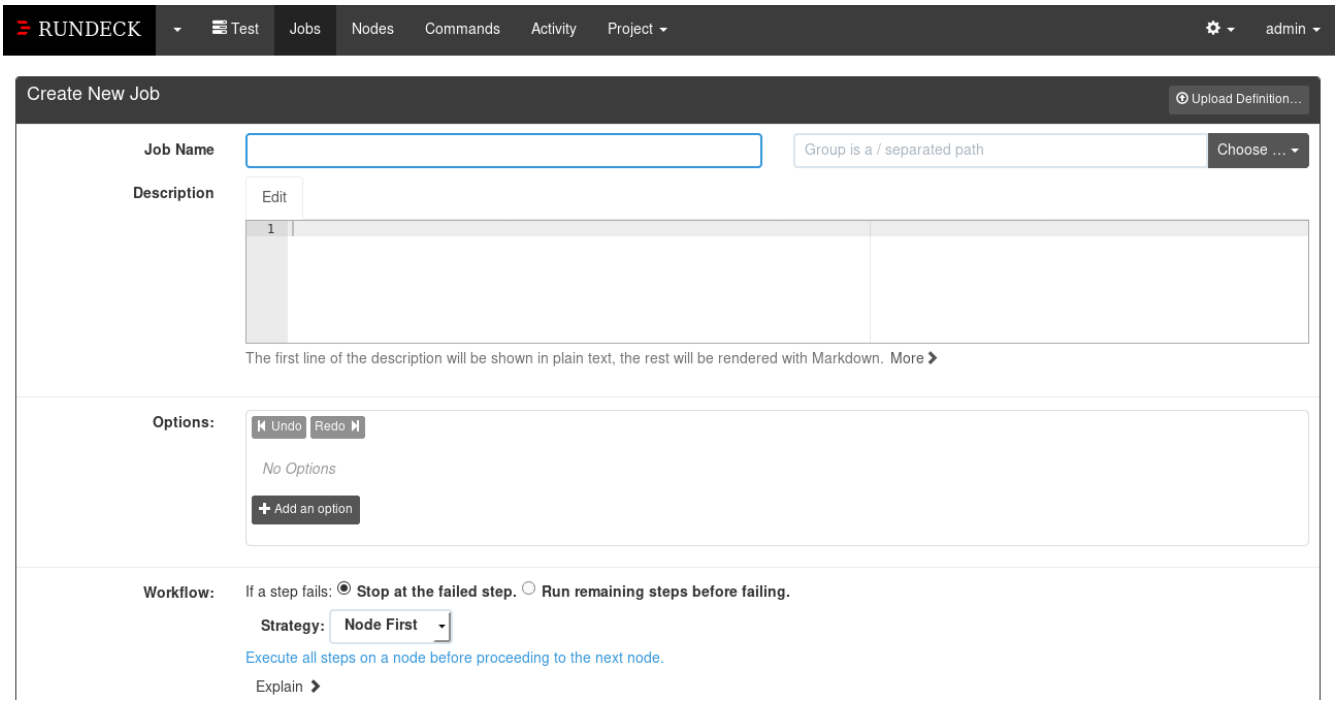
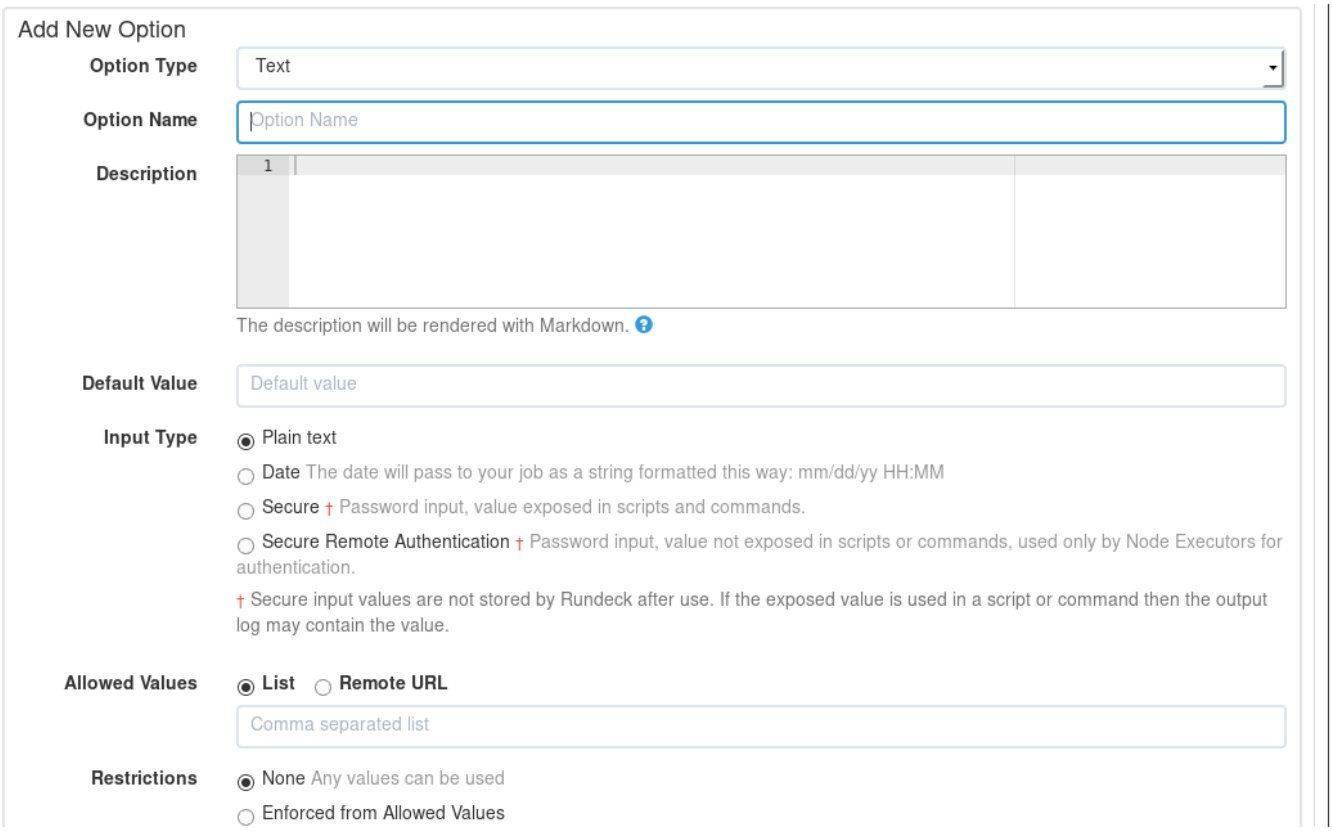


Figure 81. L'assistant de création de tâche de RunDeck

Il va falloir fournir à RunDeck le mot de passe de l'utilisateur distant permettant de se connecter au serveur ainsi que le mot de passe sudo pour lancer la commande. Pour cela :

- Ajouter une option

Cliquer sur `Add New Option`.



Donner un nom pour l'option (par exemple sshPassword) et une valeur par défaut qui correspond à votre mot de passe.

Dans le champ `Input Type`, choisir `Secure Remote Authentication` et mettre `Required` à `Yes`.

Répéter l'opération avec l'option `sudoPassword`.

Dans la section `Add a Step`, choisir `Command`. Fournir la commande dans le champ `Command`. Par exemple :

```
sudo "top -b -n 1 | head -n 5"
```

Cliquer sur `Save` puis `Create` pour finaliser la nouvelle tâche.

Pour appliquer cette tâche à un système distant (appelé noeud), éditer le fichier de configuration du noeud :

```
vi /var/rundeck/projects/your_project_name/etc/resources.xml
```

Ajouter à la ligne commençant par `localhost` : `ssh-authentication="password" ssh-password-option="option.sshPassword" sudo-command-enabled="true" sudo-password-option="option.sudoPassword"`, pour activer l'authentification par SSH et fournir les valeurs des options pour l'authentification.

Retourner dans l'interface web est executer la tâche.

Le serveur RunDeck est prêt à recevoir vos projets d'ordonnancement.

Chapitre 8. Serveur d'Intégration Continue Jenkins

🎓 Objectifs

✔ installer un serveur d'intégration continue Jenkins ;
✔ configurer l'accès au service par un mandataire proxy inverse ;
✔ sécuriser les accès au service ;
✔ créer une première tâche simple.

Jenkins est un outil Open Source d'**intégration continue** écrit en **Java**.

Interfacé avec des systèmes de gestion de version tel que Git, Subversion etc., il peut être utilisé pour compiler, tester et déployer des scripts shell ou des projets Ant, Maven, etc, selon des planifications ou des requêtes à des URLs.

Le principe de l'intégration continue est de vérifier, idéalement à chaque modification du code, la non-régression sur l'application des modifications apportées. Cette vérification systématique est primordiale pour une équipe de développeur : le projet est stable tout au long du développement et peut être livré à tout moment.

Le code source de l'application doit être :

- **partagé** avec un système de gestion versions ;
- **testé** automatiquement ;
- les modifications doivent être **poussées** régulièrement.

Ainsi, les problèmes d'intégrations peuvent être corrigés au plus tôt et la dernière version stable de l'application est connue.

8.1. Installation

Jenkins peut fonctionner :

- en mode **standalone** : avec le serveur web intégré ;
- en tant que **servlet** : avec le serveur applicatif tomcat.

En mode standalone

Installation de Jenkins :

- Sous debian :

Installation de la clé et ajout du dépot dans /etc/apt/sources.list :

```
# wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | # sudo apt-key add -
```

```
# deb http://pkg.jenkins-ci.org/debian binary/
```

Installation :

```
# apt-get update
# apt-get install jenkins
```

- Sous RedHat :

Installation du dépôt et ajout de la clé GPG :

```
# wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat/jenkins.repo
# sudo rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key
```

Installation du paquet :

```
# yum update
# yum install java-1.8.0-openjdk jenkins
```

Jenkins peut maintenant être démarré :

```
# systemctl start jenkins
# systemctl enable jenkins
```

Jenkins est directement accessible à l'adressse :

- http://IPSERVEUR:8080/jenkins

En tant que servlet tomcat

Installation de tomcat :

- Sous debian :

```
apt-get install tomcat
```

- Sous redhat :

```
yum install java-1.8.0-openjdk tomcat
```

Téléchargement de l'application :

```
cd /var/lib/tomcat6/webapps
wget http://mirrors.jenkins-ci.org/war/latest/jenkins.war
```

Si tomcat est en cours de fonctionnement, l'application sera automatiquement déployée, sinon lancer/relancer tomcat.

Jenkins est accessible à l'adressse :

- http://IPSERVEUR:8080/jenkins

8.2. Installer Nginx

Un proxy inverse Nginx (mais Apache ou HAproxy sont de bonnes alternatives) va permettre d'accéder à l'application sur le port standard 80.

```
# yum -y install epel-release
# yum -y install nginx
# systemctl enable nginx
```

Créer un nouveau serveur dans la configuration de Nginx :

```
# vim /etc/nginx/conf.d/jenkins.conf
upstream jenkins{
    server 127.0.0.1:8080;
}

server{
    listen      80;
    server_name jenkins.formatux.fr;

    access_log  /var/log/nginx/jenkins.access.log;
    error_log   /var/log/nginx/jenkins.error.log;

    proxy_buffers 16 64k;
    proxy_buffer_size 128k;

    location / {
        proxy_pass  http://jenkins;
        proxy_next_upstream error timeout invalid_header http_500 http_502 http_503
http_504;
        proxy_redirect off;

        proxy_set_header    Host            $host;
        proxy_set_header    X-Real-IP       $remote_addr;
        proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header    X-Forwarded-Proto https;
    }

}
```

Lancer le serveur Nginx :

```
# systemctl start nginx
# systemctl enable nginx
```

Vous pouvez alternativement installer un proxy inversé Apache. Dans ce cas, le VHOST suivant pourra être utilisé :

```
<Virtualhost *:80>
    ServerName        jenkins.formatux.fr
    ProxyRequests     Off
    ProxyPreserveHost On
    AllowEncodedSlashes NoDecode

    <Proxy http://localhost:8080/*>
      Order deny,allow
      Allow from all
    </Proxy>

    ProxyPass         /  http://localhost:8080/ nocanon
    ProxyPassReverse  /  http://localhost:8080/
    ProxyPassReverse  /  http://jenkins.formatux.fr/
</Virtualhost>
```

Ouvrir les ports du parefeu :

- Sur un serveur en standalone sans proxy-inverse ou sous tomcat :

```
# firewall-cmd --zone=public --add-port=8080/tcp --permanent
```

Sur un serveur avec proxy inverse :

```
# firewall-cmd --zone=public --add-service=http --permanent
# firewall-cmd --reload
```

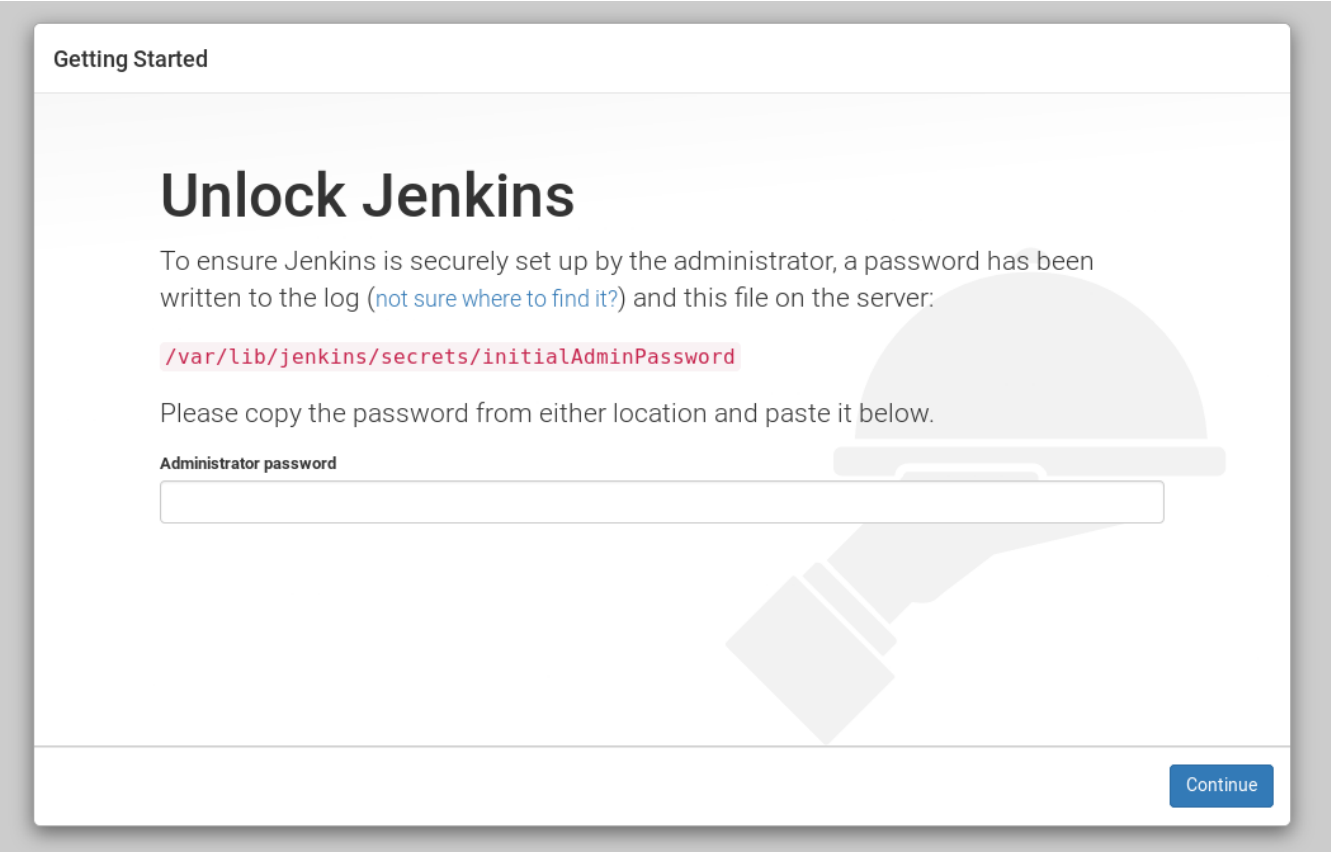
Autoriser Nginx à se connecter au serveur Jenkins d'un point de vue SELinux :

```
# setsebool httpd_can_network_connect 1 -P
```

8.3. Configuration de Jenkins

L'interface de gestion web du serveur d'intégration continue Jenkins est accessible à l'adresse `http://jenkins.formatux.fr` si le proxy inverse a été configuré ou à l'adresse `http://jenkins.formatux.fr:8080` dans les autres cas.

La page par défaut suivante s'affiche :



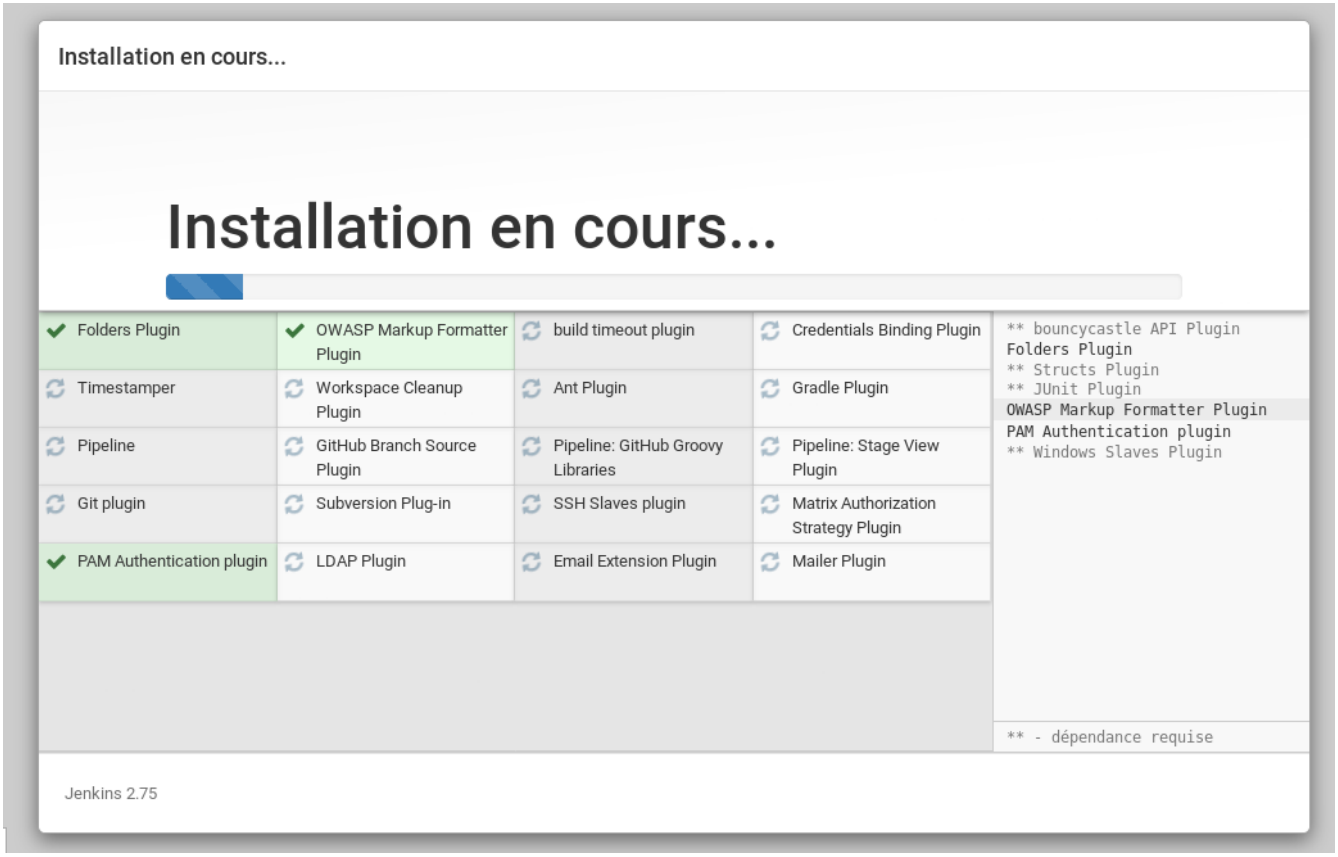
Cette page demande un mot de passe `admin` initial, qui a été généré lors de l'installation et qui est stocké dans le fichier `/var/lib/jenkins/secrets/initialAdminPassword`.

```
cat /var/lib/jenkins/secrets/initialAdminPassword
```

Utiliser le mot de passe pour ce connecter à l'interface de gestion.

L'assistant va ensuite demander quels plugins doivent être installés et proposer l'installation des plugins suggérés.

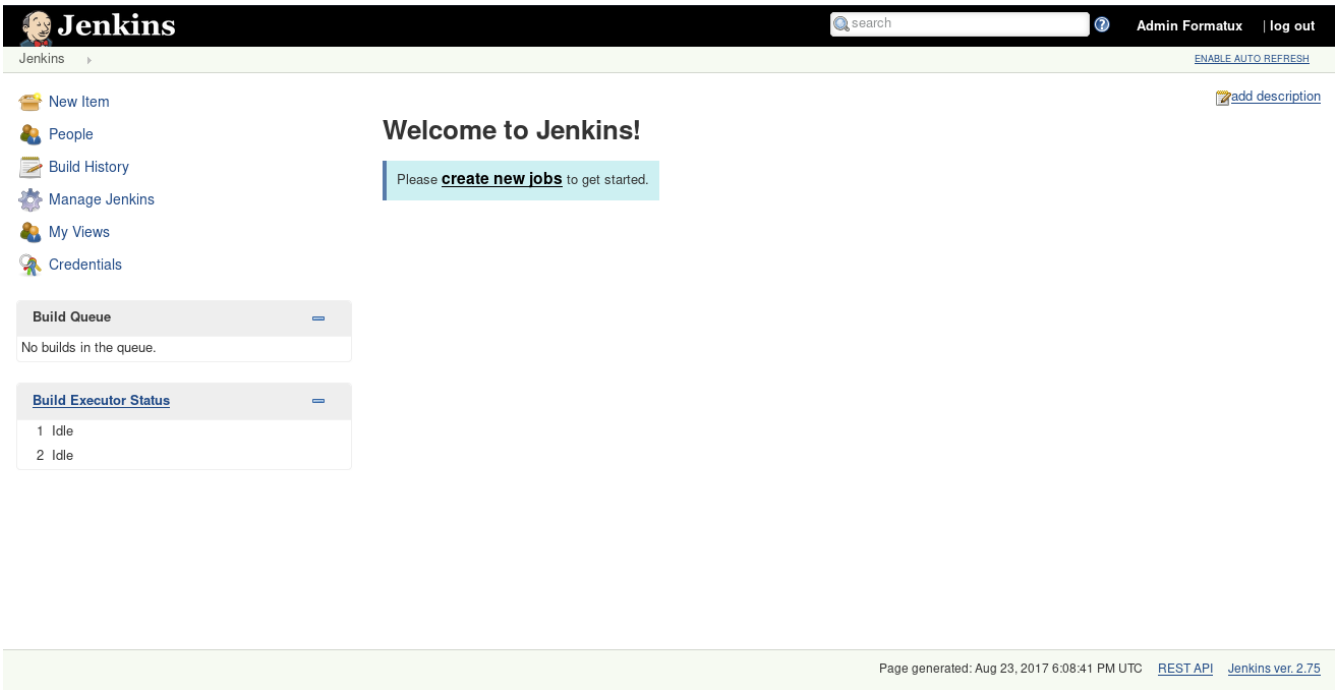
En choisissant `Installer les plugins suggérés`, tous les plugins nécessaire pour bien commencer seront installés :



L'étape suivante consiste à créer un utilisateur avec les droits d'administration pour l'accès à la

console de gestion :

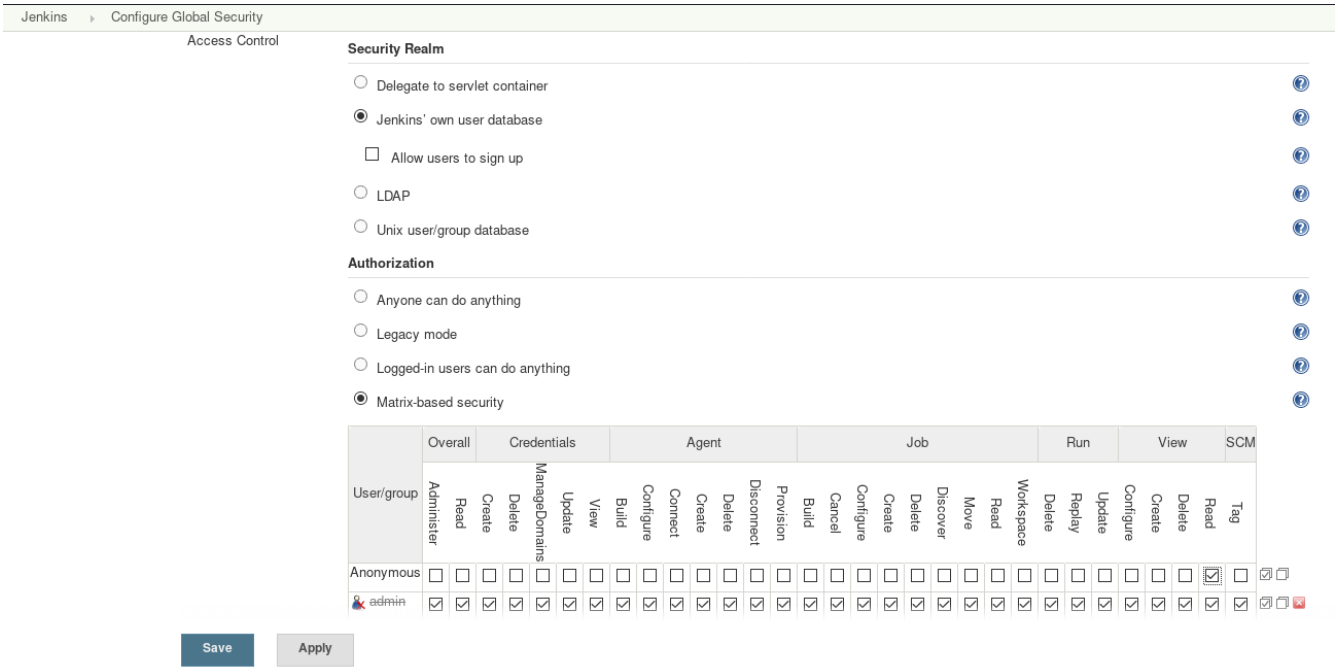
La configuration terminée, la console de gestion s'affiche :



8.4. La sécurité et la gestion des utilisateurs

Depuis l'interface de gestion, configurer les options de sécurité de Jenkins : cliquer sur Manage

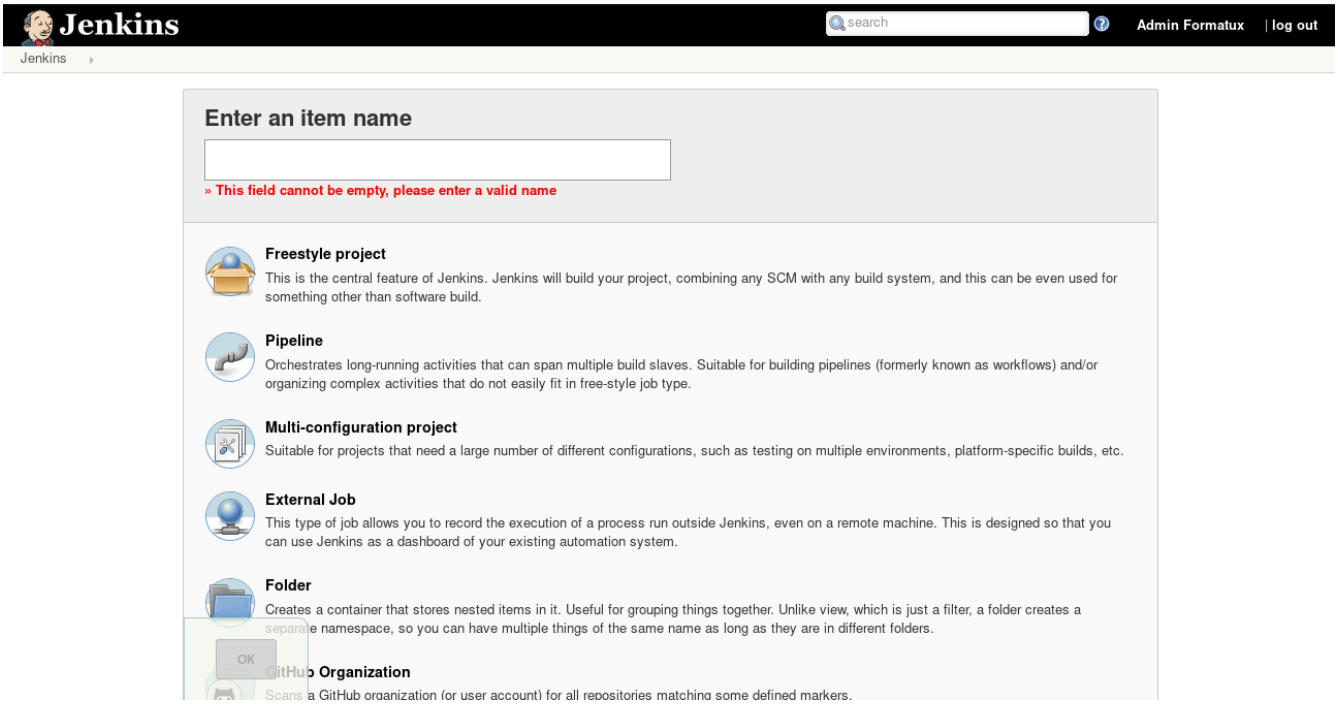
`Jenkins`, puis `Configure Global Security`.



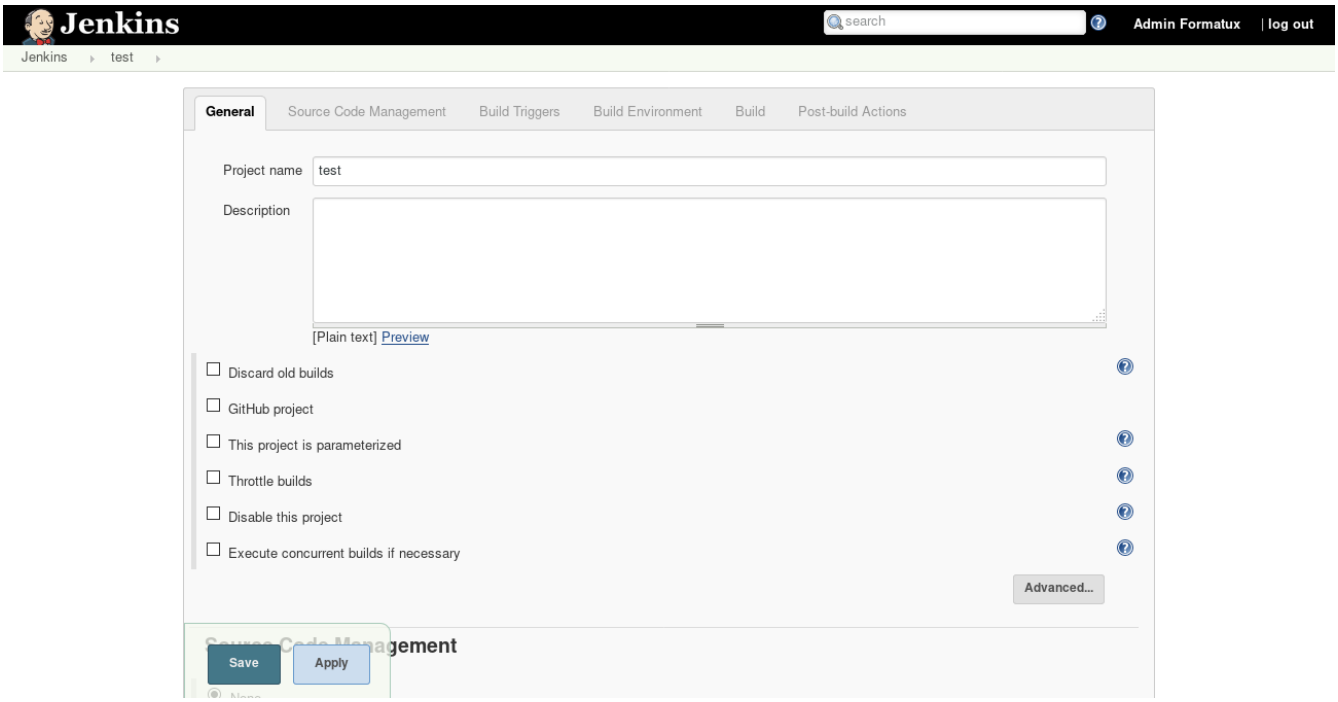
Plusieurs méthodes d'autorisations peuvent être utilisées. En sélectionnant `Matrix-based Security`, les droits des utilisateurs peuvent être gérés plus finement. Activer l'utilisateur `admin` et cliquer sur `Add`. Lui donner tous les droits en sélectionnant toutes les options. Donner à l'utilisateur `anonymous` uniquement les droits de lecture (`read`). Ne pas oublier de cliquer sur `Save`.

8.5. Ajouter une tâche d'automatisation simple

Dans l'interface de gestion, cliquer sur `Create new jobs`.



Saisir un nom pour la tâche, par exemple `test`. Choisir `Freestyle Project` et cliquer sur `OK`.

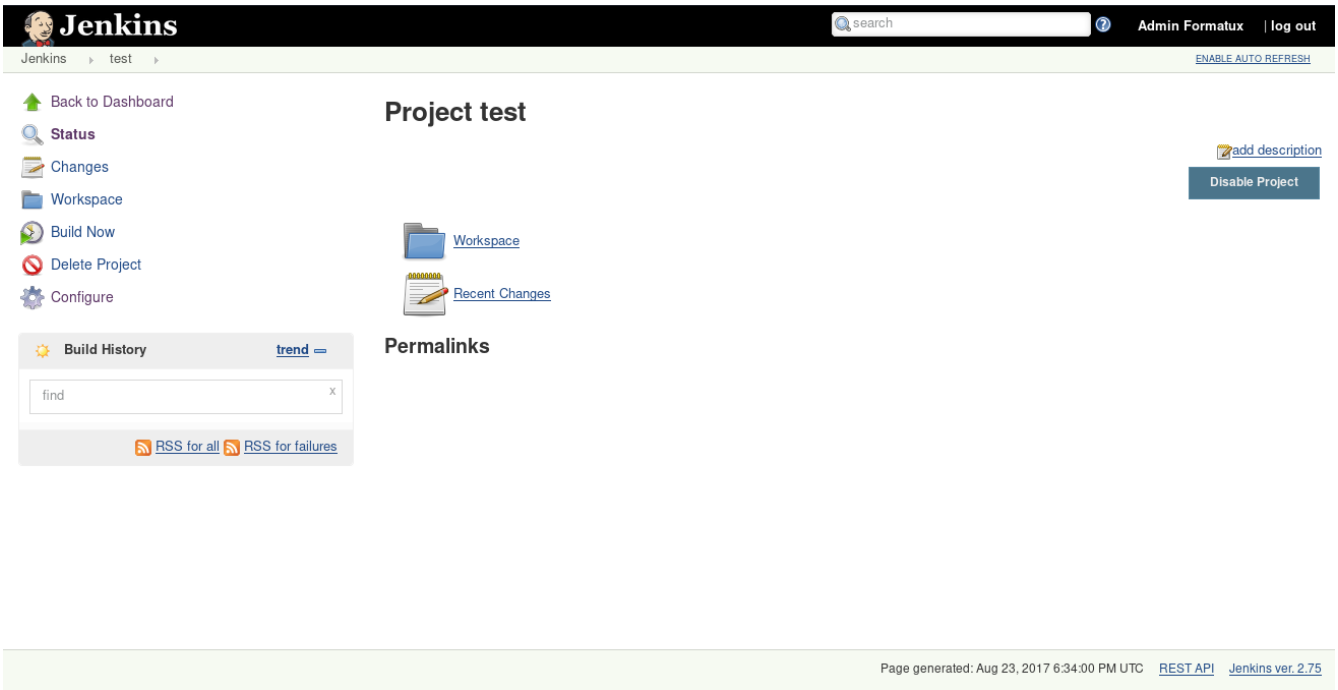


Aller dans l'onglet `Build`. Dans `Add build step`, selectionner l'option `Execute shell`.

Saisir la commande suivante dans le champ texte :

```
top -b -n 1 | head -n 5
```

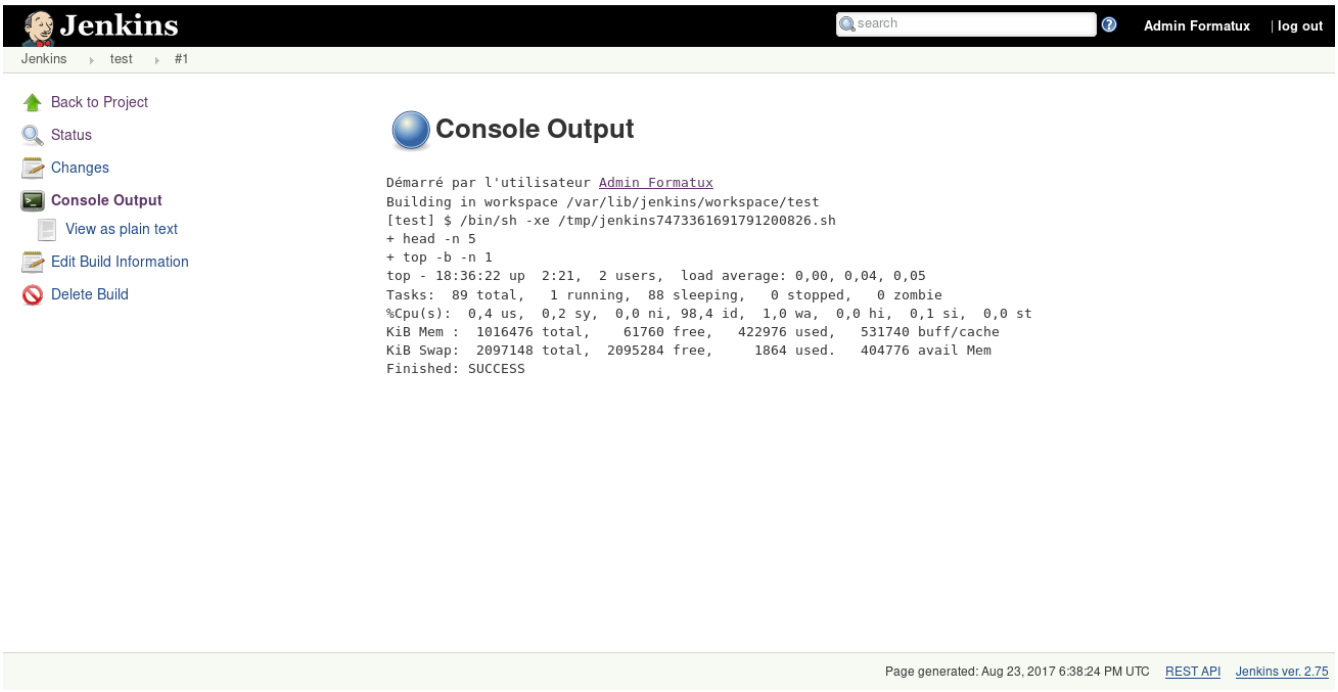
Cliquer sur `Save`.



Dans la page dédiée à la tâche `test`, cliquer sur `Build Now` pour executer la tâche `test`.

Une fois que la tâche a été executée, vous verrez l'historique du Build. Cliquer sur la première tâche

pour visualiser les résultats.



8.6. Sources

Source principale : howtoforge.

Chapitre 9. DocAsCode : le format Asciidoc

9.1. Introduction

La **Document as Code** (*Docs as Code*) est une philosophie de rédaction documentaire.

Pour favoriser la rédaction de documentations de qualité, les mêmes outils que ceux utilisés pour coder sont utilisés :

- gestionnaire de bugs,
- gestionnaire de version (`git`),
- revue de code,
- tests automatiques.

Plusieurs langages de balisage légers répondant à ces besoins sont apparus :

- le Markdown,
- le reStructuredText,
- l'Asciidoc.

> Le contenu avant la forme

Les avantages d'un tel dispositif sont nombreux :

- Le rédacteur se concentre uniquement sur le contenu et non sur la mise en forme comme c'est malheureusement trop souvent le cas avec les éditeurs de texte WYSIWYG (openoffice, word, etc.),
- Le même code source permet de générer des formats différents : pdf, html, word, confluence, epub, manpage, etc.
- Le travail collaboratif est grandement simplifié : code review, merge request, etc.
- Gestion des versions par branches ou tags git (pas une copie de fichier .doc).
- Edition de la documentation accessible à tous avec un simple éditeur de texte puisque la documentation est composée de fichiers plats (texte).

9.2. Le format asciidoc

Ce langage de balisage léger a été créé en 2002 !

Le processeur initial était développé en python mais à depuis été réécrit en ruby (https://asciidoctor.org/).

La syntaxe asciidoc a eu du mal à percer au bénéfice du markdown, mais sa grande lisibilité et le

lead de Dan Allen (https://twitter.com/mojavelinux) lui permettent de rattraper aujourd'hui son retard.

Voici un petit exemple de source asciidoc :

```
= Hello, AsciiDoc!
Doc Writer <doc@example.com>

An introduction to http://asciidoc.org[AsciiDoc].

== First Section

* item 1
* item 2
```

Comme vous pouvez le constater, le texte (malgré le balisage) reste lisible, la syntaxe est facilement assimilable et n'est pas trop lourde comparée à ce qui se faisait en latex.

> Use AsciiDoc for document markup. Really. It's actually readable by humans, easier to parse and way more flexible than XML.
>
> — Linus Torvalds

Une page référence les possibilités de la syntaxe asciidoc :

- https://asciidoctor.org/docs/asciidoc-syntax-quick-reference/

Compiler sa documentation

La documentation d'installation est disponible ici : https://asciidoctor.org/docs/install-toolchain/.

Comme pour un programme, la documentation nécessite une phase de compilation.

Après avoir installé l'environnement asciidoc, la commande suivante permet de compiler son document `.adoc` en `html5` (format par défaut) :

```
asciidoctor index.adoc
```

Une image docker existant, il est facile d'utiliser la CI d'un environnement tel que GitLab pour générer automatiquement la documentation à chaque commit, tag, etc.

```
build:
  stage: build
  image: asciidoctor/docker-asciidoctor
  script:
    - asciidoctor-pdf -a icons="font" -a lang="fr" -D public index.adoc
    only:
     - master
    artifacts:
      name: "compilation-pdf"
      paths:
        - public/
```

Une proposition d'environnement de travail

Atom (https://atom.io/) est un éditeur de code.

Après avoir installé Atom en suivant cette documentation : https://flight-manual.atom.io/getting-started/sections/installing-atom/, activer les packages suivants :

- `asciidoc-image-helper`
- `asciidoc-assistant`
- `asciidoc-preview`
- `autocomplete-asciidoc`
- `language-asciidoc`

9.3. Références

- http://www.writethedocs.org/guide/docs-as-code/
- https://www.technologies-ebusiness.com/enjeux-et-tendances/moi-code-madoc

Chapitre 10. Infrastructure as Code : Terraform

10.1. Introduction

Terraform est un produit de la société **HashiCorp** (*Terraform*, *Vault*, *Consul*, *Packer*, *Vagrant*), permettant de :

- **décrire** dans un langage humainement compréhensible l'infrastructure cible : la **HCL** (**H**ashiCorp **C**onfiguration **L**anguage),
- **versionner** les changements,
- **planifier** le déploiement,
- **créer** l'infrastructure.

Les changements apportés par une modification de code à l'infrastructure sont **prédictifs**.

> If it can be codified, it can be automated.

Le même outil permet de créer des ressources chez les plus grands fournisseurs d'infrastructure :

- AWS,
- GCP,
- Azure,
- OpenStack,
- VMware,
- etc.

L'infrastructure devient reproductible (intégration ⇒ preproduction ⇒ production) et facilement scalable. Les erreurs de manipulations humaines sont réduites.

Un exemple de création de ressouces :

```
# Une machine virtuelle
resource "digitalocean_droplet" "web" {
  name   = "tf-web"
  size   = "512mb"
  image  = "centos-7-5-x86"
  region = "sfo1"
}

/* Un enregistrement DNS
   en IPV4 type "A"
  */
resource "dnsimple_record" "hello" {
  domain = "example.com"
  name   = "test"
  value  = "${digitalocean_droplet.web.ipv4_address}"
  type   = "A"
}
```

Retrouvez plus d'informations sur le site https://www.terraform.io/, dans la documentation https://www.terraform.io/intro/index.html et dans l'espace de formation https://learn.hashicorp.com/terraform/getting-started/install.html.

10.2. La HCL

Le langage **H**ashiCorp **C**onfiguration **L**anguage est spécifique. Voici quelques points d'attention :

- Les commentaires sur une seule ligne commencent avec un `#`
- Les commentaires sur plusieurs lignes sont encadrés par `/*` et `*/`
- Les variables sont assignées avec la syntaxe `key = value` (aucune importance concernant les espaces). Les valeurs peuvent être des primitives `string`, `number` ou `boolean` ou encore une `list` ou une `map`.
- Les chaînes de caractères sont encadrées par des doubles-quotes.
- Les valeurs booléennes peuvent être `true` ou `false`.

10.3. Les providers

Terraform est utilisé pour gérer des ressources qui peuvent être des serveurs physiques, des VM, des équipements réseaux, des containeurs, mais aussi pourquoi pas des utilisateurs grafana, etc.

La liste des providers disponibles est consultable ici : https://www.terraform.io/docs/providers/index.html.

En voici quelques-uns :

- AWS,
- Azure,
- VMware vSphere
- OpenStack, CloudStack, OVH, DigitalOcean, etc.
- Gitlab,
- Datadog, PagerDuty,
- MySQL,
- Active Directory
- ...

Chaque provider venant avec ses propres ressources, il faut lire la doc !

10.4. Les actions

- `terraform init`

La première commande à lancer pour une nouvelle configuration qui va initialiser la configuration locale (import de modules par exemple).

La commande `terraform init` va automatiquement télécharger et installer les binaires des providers nécessaires.

- `terraform plan`

La commande `terraform plan` permet d'afficher le plan d'exécution, qui décrit quelles actions Terraform va prendre pour effectuer les changements réels de l'infrastructure.

Si une valeur est affichée comme `<computed>`, cela veut dire que cette valeur ne sera connue qu'au moment de l'exécution du plan.

- `terraform apply`

La commande `terraform apply` va réellement appliquer les changements tels qu'ils ont été décrits par la commande `terraform plan`.

- `terraform show`

La commande `terraform show` permet d'afficher l'état courant de l'infrastructure.

Une fois que l'infrastructure est gérée via Terraform, il est préférable d'éviter de la modifier manuellement.

Terraform va inscrire des données importantes dans un fichier `terraform.tfstate`. Ce fichier va stocker les ID des ressources créées de façon à savoir quelles ressources sont gérées par Terraform, et lesquelles ne le sont pas. Ce fichier doit donc à son tour être conservé et partagé avec toutes les

personnes devant intervenir sur la configuration.

Ce fichier `terraform.tfstate` contient des données sensibles. Il doit donc être partagé mais de manière sécurisée (des modules existent), éventuellement ailleurs que dans le repo du code de l'infrastructure.

Des ressources déjà existantes peuvent être importées avec la commande `terraform import ressourcetype.name id_existant`.

- `terraform destroy`

Avec l'avènement du cloud, le cycle de vie d'un serveur et notre façon de consommer les ressources ont considérablement changé. Une VM ou une infrastructure doit tout aussi facilement pouvoir être créée que supprimée.

Avec Terraform, une infrastructure complète peut être déployée juste à l'occasion des tests de non régréssion lors de la création d'une nouvelle version logicielle par exemple et être totalement détruite à la fin de ceux-ci pour réduire les coûts d'infrastructure au plus juste.

La commande `terraform destroy` est similaire à la commande `terraform apply`.

10.5. Dépendances des ressources

Lorsqu'une ressource dépend d'une autre ressource, la ressource parent peut être appelée comme ceci :

```
# Le VPC de la plateforme
resource "cloudstack_vpc" "default" {
  name         = "our-vpc"
  cidr         = "10.1.0.0/16"
  vpc_offering = "Default VPC offering"
  zone         = "EU-FR-IKDC2-Z4-ADV"
}

# Un réseau en 192.168.1.0/24 attaché à notre VPC
resource "cloudstack_network" "default" {
  name              = "our-network"
  cidr              = "10.1.1.0/24"
  network_offering  = "Isolated VPC tier (100MBps) with SourceNAT and StaticNAT"
  zone              = "EU-FR-IKDC2-Z4-ADV"
  vpc_id            = "${cloudstack_vpc.default.id}"
}
```

Dans l'exemple ci-dessus, un VPC (Virtual Private Cloud) est créé ainsi qu'un réseau privé. Ce réseau privé est rattaché à son VPC en lui fournissant son `id` connu dans terraform en tant que `${cloudstack_vpc.default.id}` où :

- `cloudstack_vpc` est le type de ressource,
- `default` est le nom de la ressource,
- `id` est l'attribut exporté de la ressource.

Les informations de dépendances déterminent l'ordre dans lequel Terraform va créer les ressources.

10.6. Les provisionners

Les provisionners permettent d'initialiser les instances une fois qu'elles ont été créées (lancer un playbook ansible par exemple).

Afin de mieux comprendre l'utilité d'un provisionner, étudiez l'exemple ci-dessous :

```
resource "aws_instance" "example" {
  ami           = "ami-b374d5a5"
  instance_type = "t2.micro"

  provisioner "local-exec" {
    command = "echo ${aws_instance.example.public_ip} > ip_address.txt"
  }
}
```

Lors de la création de la nouvelle VM, son adresse IP publique est stockée dans un fichier `ip_address.txt`, mais elle aurait très bien pu être envoyée à la CMDB par exemple.

Le provisioner `local-exec` exécute une commande localement, mais il existe de nombreux autres provisionners : https://www.terraform.io/docs/provisioners/index.html.

10.7. Les variables

Une variable est définie comme suit :

```
# variable de type string
variable "region" {
  default = "us-east-1"
}

# variable de type list
# definition implicite
variable "cidrs" { default = [] }

# definition explicite
variable "cidrs" { type = "list" }
```

Pour ensuite être utilisée comme suit :

```
provider "aws" {
  access_key = "${var.access_key}"
  secret_key = "${var.secret_key}"
  region     = "${var.region}"
}
```

Vous pouvez stocker vos variables dans un fichier externe (par exemple `variables.tf`) sachant que tous fichiers ayant pour extension `.tf` du répertoire courant seront chargés.

L'exemple du chapitre précédent peut être repris pour variabiliser la zone de déploiement de nos ressources :

```
# La zone de deploiement cible
variable "zone" {
  default = "EU-FR-IKDC2-Z4-ADV"
}

# Le VPC de la plateforme
resource "cloudstack_vpc" "default" {
  name         = "our-vpc"
  cidr         = "10.1.0.0/16"
  vpc_offering = "Default VPC offering"
  zone         = "${var.zone}"
}

# Un réseau en 192.168.1.0/24 attaché à notre VPC
resource "cloudstack_network" "default" {
  name              = "our-network"
  cidr              = "10.1.1.0/24"
  network_offering  = "Isolated VPC tier (100MBps) with SourceNAT and StaticNAT"
  zone              = "${var.zone}"
  vpc_id            = "${cloudstack_vpc.default.id}"
}
```

Les variables peuvent être également définies depuis la ligne de commande ou un fichier externe `terraform.tfvars` qui sera chargé automatiquement à l'exécution.

Il est d'usage de stocker les variables critiques (api key, mots de passe, etc.) dans un fichier terraform.tfvars et d'exclure ce fichier de votre configuration git.

Voir la documentation pour plus d'informations sur les variables (Maps, etc.) : https://learn.hashicorp.com/terraform/getting-started/variables

10.8. TD

Plusieurs possibilités :

- Créer un compte gratuit chez AWS puis :
  - Suivre le module `getting-started` d'hashicorp : https://learn.hashicorp.com/terraform/getting-started/install
  - Créer votre propre infrastructure sur AWS.
- Créer un compte chez Ikoula et gérer une infrastructure CloudStack (https://cloudstack.apache.org/).
- Piloter votre infrastructure VMWare.
- Installer grafana ou un autre logiciel disposant d'un provider sur une de vos VM et utiliser les ressources de ce provider.

Partie 5 : Shell.

Le scripting BASH sous LINUX.

Dans cette partie, vous allez pouvoir vous initier au scripting Bash, exercice que tout administrateur devra réaliser un jour ou l'autre.

Un TP Final vous est proposé avec des pistes de réalisation.

Chapitre 1. Les scripts shell - Niveau 1

🎓 Objectifs pédagogiques

Dans ce chapitre, vous allez apprendre à rédiger votre premier script, à utiliser des variables et également à interagir avec les utilisateurs.

✔ Rédiger son premier script en bash ;
✔ Utiliser des variables ;
✔ Interagir avec un utilisateur ;
✔ Transformer le contenu d'un fichier ou d'une variable ;
✔ Gérer les arguments d'un script.

🏷 **script**, **shell**, **bash**

Connaissances : ⚙
Niveau technique : ☆
Temps de lecture : 20 minutes

Le shell est **l'interpréteur de commandes** de Linux. C'est un binaire qui ne fait pas partie du noyau, mais forme une couche supplémentaire, d'où son nom de "*coquille*".

Il analyse les commandes saisies par l'utilisateur puis les fait exécuter par le système.

Il existe plusieurs shells, tous partageant des points communs. L'utilisateur est libre d'utiliser celui qui lui convient le mieux parmi (entre autres) :

- le **Bourne-Again shell** (`bash`),
- le **Korn shell** (`ksh`),
- le **C shell** (`csh`),
- etc.

Le `bash` est présent par défaut sur les principales distributions Linux. Il se caractérise par ses fonctionnalités pratiques et conviviales.

Le shell est aussi un **langage de programmation** basique qui, grâce à quelques commandes dédiées, permet :

- l'utilisation de **variables**,
- l'exécution **conditionnelle** de commandes,
- la **répétition** de commandes.

Les scripts en shell ont l'avantage d'être réalisables **rapidement** et de manière **fiable**, **sans compilation** ni installation de commandes supplémentaires. Un script shell n'est qu'un fichier

texte sans enluminures (gras, italique, etc.).

Si le shell est un langage de programmation « basique », il n'en reste pas moins très puissant et parfois plus rapide qu'un mauvais code compilé. Si vous n'en êtes pas convaincu, vous pouvez lire, même s'il commence à dater, l'article suivant : Entretien avec un débutant en bash de Étienne Dublé. Cela vous permettra de réfléchir à votre façon de coder.

Pour écrire un script shell, il suffit de réunir dans un même fichier texte toutes les commandes nécessaires. En rendant ce fichier exécutable, le shell le lira séquentiellement et exécutera une à une les commandes le comportant. Il est aussi possible de l'exécuter en passant le nom du script comme un argument au binaire `bash`.

Lorsque le shell rencontre une erreur, il affiche un message permettant d'identifier le problème mais continue l'exécution du script. Mais il existe des mécaniques pour stopper l'exécution d'un script en cas de survenance d'une erreur. Les erreurs propres aux commandes sont également affichées à l'écran ou à l'intérieur de fichiers.

Qu'est ce qu'un bon script ? C'est un script :

- **fiable** : son fonctionnement est irréprochable même en cas de mauvaise utilisation ;
- **commenté** : son code est annoté pour en faciliter la relecture et les futures évolutions ;
- **lisible** : le code est indenté à bon escient, les commandes sont aérées, ...
- **portable** : le code fonctionne sur tout système Linux, gestion des dépendances, gestion des droits, etc.

1.1. Premier script

Pour commencer l'écriture d'un script shell, il est pratique d'utiliser un éditeur de texte gérant la coloration syntaxique.

`vim`, par exemple, est un outil adapté à cela.

Le nom du script devra respecter quelques règles :

- pas de nom de commandes existantes ;
- ne contenant que des caractères alphanumériques, c'est-à-dire pas de caractère accentué ni d'espace ;
- extension en `.sh` pour indiquer qu'il s'agit d'un script shell.

hello-world.sh

```
1 #!/usr/bin/env bash
2 #
3 # Auteur : Antoine Le Morvan
4 # Date : Janvier 2019
5 # Version 1.0.0 : Affiche le texte "Hello world !"
6 #
7
8 # Affiche un texte à l'écran :
9 echo "Hello world !"
```

Pour pouvoir exécuter ce script, en argument du `bash` :

```
$ bash hello-world.sh
Hello world !
```

Ou, plus simplement, après lui avoir donné le droit d'exécution :

```
$ chmod u+x ./hello-world.sh
$ ./hello-world.sh
Hello world !
```

Notez que pour exécuter le script, celui-ci a été appelé avec « `./` » avant son nom. L'interpréteur pourra refuser d'exécuter un script présent dans le répertoire courant sans indiquer un chemin (ici avec le « `./` » devant). La commande `chmod` n'est à passer qu'une seule fois sur un script nouvellement créé.

La première ligne à écrire dans tout script permet d'indiquer le nom du binaire du shell à utiliser pour l'exécuter. Si vous désirez utiliser le shell `ksh` ou le langage interprété `python`, vous remplaceriez la ligne :

```
#!/usr/bin/env bash
```

par :

```
#!/usr/bin/env ksh
```

ou par :

```
#!/usr/bin/env python
```

Tout au long de l'écriture, il faudra penser à la relecture du script en utilisant notamment des commentaires :

- une présentation générale, en début, pour indiquer le but du script, son auteur, sa version, son utilisation, etc.
- au cours du texte pour aider à la compréhension des actions.

Les commentaires peuvent être placés sur une ligne à part ou bien à la fin d'une ligne contenant une commande.

Exemple :

```
# Ce programme affiche la date
date    # Cette ligne est la ligne qui affiche la date !
```

1.2. Variables

Comme dans tout langage de programmation, le script shell utilise des **variables**. Elles servent à stocker des informations en mémoire pour les réutiliser à volonté au cours du script.

Une variable est créée au moment où elle reçoit son contenu. Elle reste valide jusqu'à la fin de l'exécution du script ou à la demande explicite de l'auteur du script. Puisque le script est exécuté séquentiellement du début à la fin, il est impossible de faire appel à une variable avant qu'elle ne soit créée.

Le contenu d'une variable peut être modifié au cours du script, la variable continue d'exister. Si le contenu est supprimé, la variable reste active mais ne contient rien.

La notion de type de variable en script shell est possible mais est très peu utilisée. Le contenu d'une variable est **toujours un caractère ou une chaîne de caractères**.

01-backup.sh

```
 1 #!/usr/bin/env bash
 2
 3 #
 4 # Auteur : Antoine Le Morvan
 5 # Date : Janvier 2019
 6 # Version 1.0.0 : Sauvegarde dans /root les fichiers passwd, shadow, group et
gshadow
 7 #
 8
 9 # Variables globales
10 FICHIER1=/etc/passwd
11 FICHIER2=/etc/shadow
12 FICHIER3=/etc/group
13 FICHIER4=/etc/gshadow
14
15 # Dossier destination
16 DESTINATION=/root
17
18 # Nettoie l'écran :
19 clear
20
21 # Lancer la sauvegarde
22 echo "La sauvegarde de $FICHIER1, $FICHIER2, $FICHIER3, $FICHIER4 vers $DESTINATION
va commencer :"
23
24 cp $FICHIER1 $FICHIER2 $FICHIER3 $FICHIER4 $DESTINATION
25
26 echo "La sauvegarde est terminée !"
```

Ce script fait usage de variables. Le nom d'une variable doit commencer par une lettre mais peut ensuite contenir n'importe quelle suite de lettres ou de chiffres. Hormis le tiret bas « _ », les caractères spéciaux ne sont pas utilisables.

Par convention, les variables créées par un utilisateur ont un nom en minuscules. Ce nom doit être choisi avec précaution pour n'être ni trop évasif ni trop compliqué. Une variable peut toutefois être nommée avec des majuscules, comme c'est le cas ici, s'il s'agit d'une variable globale qui ne doit pas être modifiée par le programme.

Le caractère « = » affecte du contenu à une variable :

```
variable=valeur
nom_rep="/home"
```

Il n'y a pas d'espace ni avant ni après le signe =.

Pour afficher du texte en même temps que le contenu d'une variable, il est obligatoire d'utiliser les guillemets et non les apostrophes.

L'usage des apostrophes inhibe l'interprétation des caractères spéciaux.

```
$ message="Bonjour"
$ echo "Voici le contenu de la variable message : $message"
Voici le contenu de la variable message : Bonjour
$ echo 'Voici le contenu de la variable message : $message'
Voici le contenu de la variable message : $message
```

Pour isoler le nom de la variable, il faut utiliser les apostrophes ou les accolades :

```
$ fichier=nom_fichier
$ touch "$fichier"1
$ touch ${fichier}1
```

L'utilisation systématique des accolades est conseillée.

Supprimer et verrouiller les variables

La commande `unset` permet de supprimer une variable.

Exemple :

```
$ nom="NOM"
$ prenom="Prenom"
$ echo "$nom $prenom"
NOM Prenom
$ unset prenom
$ echo "$nom $prenom"
NOM
```

La commande `readonly` ou `typeset -r` verrouille une variable.

Exemple :

```
$ nom="NOM"
$ readonly nom
$ nom="AUTRE NOM"
bash: nom: variable en lecture seule
$ unset nom
bash: nom: variable en lecture seule
```

Un `set -u` en début de script va arrêter l'exécution du script en cas d'utilisation des variables non déclarées.

Variables d'environnements

Les variables **d'environnements** et les variables **systèmes** sont des variables utilisées par le système pour son fonctionnement. Par convention elles portent un nom en majuscules.

Comme toutes variables, elles peuvent être affichées à l'exécution d'un script. Même si cela est fortement déconseillé, elles peuvent aussi y être modifiées.

- La commande `env` permet d'afficher toutes les variables d'environnement utilisées.
- La commande `set` permet d'afficher toutes les variables système utilisées.

Parmi les dizaines de variables d'environnement, plusieurs ont un intérêt à être utilisées dans un script shell :

Table 114. Variables d'environnement

Variable	Observation
`HOSTNAME`	Nom d'hôte de la machine.
`USER`, `USERNAME` et `LOGNAME`	Nom de l'utilisateur connecté sur la session.
`PATH`	Chemin où trouver les commandes.
`PWD`	Répertoire courant, mis à jour à chaque exécution de la commande `cd`.
`HOME`	Répertoire de connexion.
`$$`	Numéro du processus de l'exécution du script.
`$?`	Code retour de la dernière commande exécutée.

Exporter une variable

La commande `export` permet d'exporter une variable.

Une variable n'est valable que dans l'environnement du processus du script shell. Pour que les **processus fils** du script puissent connaître les variables et leurs contenus, il faut les exporter.

La modification d'une variable exportée dans un processus fils ne peut pas remonter au processus père.

Sans option, la commande `export` affiche le nom et les valeurs des variables exportées dans l'environnement.

La substitution de commande

Il est possible de stocker le résultat d'une commande dans une variable.

Cette opération n'est valable que pour les commandes qui renvoient un message à la fin de leur exécution.

La syntaxe pour sous-exécuter une commande est la suivante :

Syntaxes pour la substitution de commandes

```
variable=`commande`
variable=$(commande) # Syntaxe à privilégier
```

Exemple :

```
$ jour=`date +%j`
$ homedir=$(pwd)
```

Améliorations du script de sauvegarde

Quelques pistes d'améliorations

```
 1 #!/usr/bin/env bash
 2
 3 #
 4 # Auteur : Antoine Le Morvan
 5 # Date : Janvier 2019
 6 # Version 1.0.0 : Sauvegarde dans /root les fichiers passwd, shadow, group et
gshadow
 7 # Version 1.0.1 : Création d'un répertoire avec le quantième du jour.
 8 #                  Améliorations diverses
 9
10 # Variables globales
11
12 ## Fichiers a sauvegarder
13 FICHIER1=/etc/passwd
14 FICHIER2=/etc/shadow
15 FICHIER3=/etc/group
16 FICHIER4=/etc/gshadow
17
18 ## Dossier destination
19 DESTINATION=/root
20
21 ## Variables en readonly
22 readonly FICHIER1
23 readonly FICHIER2
```

```
24 readonly FICHIER3
25 readonly FICHIER4
26 readonly DESTINATION
27
28 # Un nom de dossier avec le quantième du jour
29 rep="backup-$(date +%j)"
30
31 # Nettoie l'écran :
32 clear
33
34 # Lancer la sauvegarde
35 echo "********************************************************************"
36 echo "     Script de sauvegarde - Sauvegarde sur la machine $HOSTNAME "
37 echo "********************************************************************"
38 echo "La sauvegarde sera faîte dans le dossier ${rep}."
39 echo "Création du répertoire..."
40 mkdir -p $DESTINATION/$rep
41 echo "                                                        [ OK ]"
42 echo "La sauvegarde de ${FICHIER1}, ${FICHIER2}, ${FICHIER3}, ${FICHIER4} vers
${DESTINATION}/$rep va commencer :"
43
44 cp $FICHIER1 $FICHIER2 $FICHIER3 $FICHIER4 $DESTINATION/$rep
45
46 echo "La sauvegarde est terminée !"
47
48 # La sauvegarde est notée dans le journal d'évènements du système :
49 echo "La sauvegarde est renseignée dans syslog :"
50 logger "Sauvegarde des fichiers systèmes par ${USER} sur la machine ${HOSTNAME}
dans le dossier ${DESTINATION}/$rep."
51 echo "                                                        [ OK ]"
```

Exécution de notre script de sauvegarde

```
root # ./02-backup-enhanced.sh
********************************************************************
     Script de sauvegarde - Sauvegarde sur la machine formateur1
********************************************************************
La sauvegarde sera faîte dans le dossier backup-262.
Création du répertoire...
                                                        [ OK ]
La sauvegarde de /etc/passwd, /etc/shadow, /etc/group, /etc/gshadow vers /root/backup-
262 va commencer :
La sauvegarde est terminée !
La sauvegarde est renseignée dans syslog :
                                                        [ OK ]
```

Grace à la commande `logger`, les affichages de l'exécution du script sont écrites dans le journal `syslog` :

Événement dans syslog

```
root # tail -f /var/log/messages
janvier. 02 19:35:35 formateur1 antoine[9712]: Sauvegarde des fichiers systèmes par
antoine sur la machine formateur1 dans le dossier /root/b...
```

1.3. Saisie et manipulations

Selon l'objet du script, il peut être nécessaire de lui envoyer des informations lors de son lancement ou durant son exécution.

Ces informations, non connues lors de l'écriture du script, peuvent être extraites à partir de fichiers ou saisies par l'utilisateur.

Il est aussi possible d'envoyer ces informations sous forme d'arguments lors de la saisie de la commande du script. C'est le mode de fonctionnement de nombreuses commandes Linux.

La commande `read`

La commande `read` permet de saisir une chaîne de caractères pour la stocker dans une variable.

Syntaxe de la commande read

```
read [-n X] [-p] [-s] [variable]
```

Exemple de la commande read

```
$ read nom prenom
$ read -p "Veuillez saisir votre nom : " nom
```

Table 115. Options de la commande read

Option	Observation
`-p`	Affiche un message de prompt
`-n`	Limite le nombre de caractères à saisir
`-s`	Masque la saisie

Lors de l'utilisation de l'option `-n`, le shell valide automatiquement la saisie au bout du nombre de caractères précisés. L'utilisateur n'a pas à appuyer sur la touche `[ENTREE]`.

```
$ read -n5 nom
```

La commande `read` permet d'interrompre l'exécution du script le temps que l'utilisateur saisisse des informations. La saisie de l'utilisateur est découpée en mots affectés à une ou plusieurs variables

prédéfinies. Les mots sont des chaînes de caractères séparées par le séparateur de champs.

La fin de la saisie est déterminée par la frappe sur la touche `[ENTREE]` ou le caractère spécial de fin de ligne.

Une fois la saisie validée, chaque mot sera stocké dans la variable prédéfinie.

Le découpage des mots est défini par le caractère séparateur de champs. Ce séparateur est stocké dans la variable système `IFS` (*Internal Field Separator*).

```
$ set | grep IFS
IFS=$' \t\n'
```

Par défaut, l'IFS contient l'espace, la tabulation `\t` et le saut de ligne `\n`.

Utilisée sans préciser de variable, cette commande met simplement le script en pause. Le script continue son exécution lors de la validation de la saisie.

Cette utilisation permet de faire une pause lors du débogage d'un script ou pour inciter l'utilisateur à appuyer sur *ENTREE* pour continuer.

```
$ echo -n "Appuyer sur [ENTREE] pour continuer..."
$ read
```

La commande `cut`

La commande `cut` permet d'isoler une colonne dans un fichier ou dans un flux.

Syntaxe de la commande cut

```
cut [-cx] [-dy] [-fz] fichier
```

Exemple d'utilisation de la commande cut

```
$ cut -d: -f1 /etc/passwd
```

Table 116. Options de la commande cut

Option	Observation
`-c`	Spécifie les numéros d'ordre des caractères à sélectionner
`-d`	Spécifie le séparateur de champs
`-f`	Spécifie le numéro d'ordre des colonnes à sélectionner

Le principal intérêt de cette commande sera son association avec un flux, par exemple la

commande `grep` et le pipe `|`.

- La commande `grep` travaille verticalement (*isolation d'une ligne parmi toutes celles du fichier*).
- La combinaison des deux commandes permet **d'isoler un champ précis du fichier**.

Syntaxe de la commande cut

```
# grep "^root:" /etc/passwd | cut -d: -f3
0
```

Les fichiers de configurations comportant une structure unique utilisant le même séparateur de champs sont des cibles idéales pour cette combinaison de commandes.

La commande `tr`

La commande `tr` permet de convertir une chaîne de caractères.

Syntaxe de la commande tr

```
tr [-csd] chaîne1 chaîne2
```

Table 117. Options de la commande tr

Option	Observation
`-c`	Tous les caractères qui ne sont pas spécifiés dans la première chaîne sont convertis selon les caractères de la seconde.
`-d`	Efface le caractère spécifié.
`-s`	Réduire à une seule unité le caractère spécifié.

Exemple d'utilisation de la commande tr

```
$ grep root /etc/passwd
root:x:0:0:root:/root:/bin/bash
$ grep root /etc/passwd | tr -s "o"
rot:x:0:0:rot:/rot:/bin/bash
```

Exercice : Extraire le niveau d'exécution du fichier `/etc/inittab`

```
#!/usr/bin/env bash

#
# Auteur : Antoine Le Morvan
# Date : Janvier 2019
# Version 1.0.0 : Extrait le niveau d'exécution du fichier /etc/inittab

# Variables globales

INITTAB=/etc/inittab

niveau=`grep "^id" $INITTAB | cut -d: -f2`

# Affichage du résultat :
echo "Le niveau d'init au démarrage est : $niveau"
```

Extraire le nom et le chemin d'un fichier

- La commande `basename` permet d'extraire le nom du fichier à partir d'un chemin.
- La commande `dirname` permet d'extraire le chemin parent d'un fichier.

Exemple :

```
$ echo $FICHIER=/usr/bin/passwd
$ basename $FICHIER
passwd
$ dirname $FICHIER
/usr/bin
```

Arguments d'un script

La demande de saisie d'informations grâce à la commande `read` interrompt l'exécution du script tant que l'utilisateur ne fait pas de saisie.

Cette méthode, bien que très conviviale, présente des limites s'il s'agit d'un script à l'exécution programmée la nuit par exemple. Afin de palier ce problème, il est possible d'injecter les informations souhaitées via des **arguments**.

De nombreuses commandes Linux fonctionnent sur ce principe.

Cette façon de faire à l'avantage qu'une fois le script exécuté, il n'aura plus besoin d'intervention humaine pour se terminer.

Son inconvénient majeur est qu'il faudra prévenir l'utilisateur de la syntaxe du script pour éviter des erreurs.

Les arguments sont renseignés lors de la saisie de la commande du script. Ils sont séparés par un espace.

```
$ ./script argument1 argument2
```

Une fois exécuté, le script enregistre les arguments saisis dans des variables prédéfinies : les **variables positionnelles**.

Ces variables sont utilisables dans le script comme n'importe quelle autre variable, à l'exception faite qu'elles ne peuvent pas être affectées.

- Les variables positionnelles non utilisées existent mais sont vides.
- Les variables positionnelles sont toujours définies de la même façon :

Table 118. Les variables positionnelles

Variable	Observation
`$0`	contient le nom du script tel qu'il a été saisi.
`$1` à `$9`	contiennent les valeurs du 1er et du 9ème argument.
`${x}`	contient la valeur de l'argument `x`, supérieur à 9.
`$#`	contient le nombre d'arguments passés.
`$*` ou `$@`	contient en une variable tous les arguments passés

Exemples :

un_deux_trois.sh

```
 1 #!/usr/bin/env bash
 2 #
 3 # Auteur : Damien dit LeDub
 4 # Date : septembre 2019
 5 # Version 1.0.0 : Affiche la valeur des arguments positionnels
 6 # de 1 à 3
 7
 8 # Le séparateur de champ sera "," ou l'espace
 9 # Important pour visualiser la différence en $* et $@
10 IFS=", "
11
12 # Affiche un texte à l'écran :
13 echo "Le nombre d'argument (\$#) = $#"
14 echo "Le nom du script     (\$0) = $0"
15 echo "Le 1er argument      (\$1) = $1"
16 echo "Le 2e argument       (\$2) = $2"
17 echo "Le 3e argument       (\$3) = $3"
18 echo "Tous séparés par IFS (\$*) = $*"
19 echo "Tous sans séparation (\$@) = $@"
```

```
$ ./un_deux_trois.sh un deux "trois quatre"
Le nombre d'argument ($#) = 3
Le nom du script     ($0) = ./un_deux_trois.sh
Le 1er argument      ($1) = un
Le 2e argument       ($2) = deux
Le 3e argument       ($3) = trois quatre
Tous séparés par IFS ($*) = un,deux,trois quatre
Tous sans séparation ($@) = un deux trois quatre
```

Attention à la différence entre **`$@`** et **`$*`**.

Elle se fait au niveau du format de stockage des arguments :

- **`$*`** : Contient les arguments au format **`"$1 $2 $3 ⋯"`**
- **`$@`** : Contient les arguments au format **`"$1" "$2" "$3" ⋯`**

C'est en modifiant la variable d'environnement IFS que la différence est visible.

La commande `shift`

La commande `shift` permet de **décaler les variables positionnelles**.

Exemples :

un_deux_trois_shift.sh

```
#!/usr/bin/env bash
#
# Auteur : Damien dit LeDub
# Date : septembre 2019
# Version 1.0.0 : Affiche la valeur des arguments positionnels
# de 1 à 3 avant et après un «shift»

# Le séparateur de champ sera "," ou l'espace
# Important pour visualiser la différence en $* et $@
IFS=", "

# Récupère les arguments avant le «shift»
AVANT_SHIFT_NB="$#"
AVANT_SHIFT_NOM="$0"
AVANT_SHIFT_IER="$1"
AVANT_SHIFT_2E="$2"
AVANT_SHIFT_3E="$3"
AVANT_SHIFT_TOUS_IFS="$*"
AVANT_SHIFT_TOUS="$@"

# Récupère les arguments après le «shift»
shift 2
APRES_SHIFT_NB="$#"
APRES_SHIFT_NOM="$0"
APRES_SHIFT_IER="$1"
APRES_SHIFT_2E="$2"
APRES_SHIFT_3E="$3"
APRES_SHIFT_TOUS_IFS="$*"
APRES_SHIFT_TOUS="$@"

# Affiche toutes les informations récupérées
echo "Le nombre d'argument (\$#) : ${AVANT_SHIFT_NB}"
echo "  après le shift 2        : ${APRES_SHIFT_NB}"
echo "Le nom du script     (\$0) : ${AVANT_SHIFT_NOM}"
echo "  après le shift 2        :${APRES_SHIFT_NOM}"
echo "Le 1er argument      (\$1) : ${AVANT_SHIFT_IER}"
echo "  après le shift 2        :${APRES_SHIFT_IER}"
echo "Le 2e argument       (\$2) :${AVANT_SHIFT_2E}"
echo "  après le shift 2        :${APRES_SHIFT_2E}"
echo "Le 3e argument       (\$3) : ${AVANT_SHIFT_3E}"
echo "  après le shift 2        :${APRES_SHIFT_3E}"
echo "Tous séparé par IFS  (\$*) : ${AVANT_SHIFT_TOUS_IFS}"
echo "  après le shift 2        :${APRES_SHIFT_TOUS_IFS}"
echo "Tous sans séparation (\$@) : ${AVANT_SHIFT_TOUS}"
echo "  après le shift 2        :${APRES_SHIFT_TOUS} "
```

```
$ ./un_deux_trois_shift.sh un deux "trois quatre"
Le nombre d'argument ($#) : 3
  après le shift 2        : 1
Le nom du script     ($0) : ./un_deux_trois_shift.sh
  après le shift 2        :./un_deux_trois_shift.sh
Le 1er argument      ($1) : un
  après le shift 2        :trois quatre
Le 2e argument       ($2) :deux
  après le shift 2        :
Le 3e argument       ($3) : trois quatre
  après le shift 2        :
Tous séparé par IFS  ($*) : un,deux,trois quatre
  après le shift 2        :trois quatre
Tous sans séparation ($@) : un deux trois quatre
  après le shift 2        :trois quatre
```

Lors de l'utilisation de la commande `shift`, les variables `$#` et `$*` sont modifiées en conséquence.

La commande `set`

La commande `set` découpe une chaîne en variables positionnelles.

Syntaxe de la commande set

```
set [valeur] [$variable]
```

Exemple :

```
$ set un deux trois
$ echo $1 $2 $3 $#
un deux trois 3
$ variable="un deux trois"
$ set $variable
$ echo $1 $2 $3 $#
un deux trois 3
```

Ci-dessous, la version de notre script de sauvegarde mettant en oeuvre les variables positionnelles :

```
#!/usr/bin/env bash

#
# Auteur : Antoine Le Morvan
# Date : Janvier 2019
# Version 1.0.0 : Sauvegarde dans /root les fichiers passwd, shadow, group et gshadow
# Version 1.0.1 : Création d'un répertoire avec le quantième du jour.
#         Améliorations diverses
# Version 1.0.2 : Modification pour utiliser les variables positionnelles
#                 Limitation à 5 fichiers

# Variables globales

## Dossier destination
DESTINATION=/root

# Un nom de dossier avec le quantieme du jour
rep="backup-$(date +%j)"

# Nettoie l'écran :
clear

# Lancer la sauvegarde
echo "**********************************************************************"
echo "     Script de sauvegarde - Sauvegarde sur la machine $HOSTNAME "
echo "**********************************************************************"
echo "La sauvegarde sera faîte dans le dossier ${rep}."
echo "Création du répertoire..."
mkdir -p $DESTINATION/$rep
echo "                                                    [ OK ]"
echo "La sauvegarde de ${1} ${2} ${3} ${4} ${5} vers ${DESTINATION}/$rep va commencer :"

cp $1 $2 $3 $4 $5 $DESTINATION/$rep

echo "La sauvegarde est terminée !"
```

Tester vos connaissances

✔ Parmi ces 4 shells, lequel n'existe pas :
◯ Bash
◯ Ksh
◯ Tsh
◯ Csh

✔ Quelle est la bonne syntaxe pour affecter un contenu à une variable :
◯ `variable:=valeur`
◯ `variable := valeur`
◯ `variable = valeur`
◯ `variable=valeur`

✔ Comment stocker le retour d'une commande dans une variable :
◯ `fichier=$(ls)`
◯ `fichier=\`ls\``
◯ `fichier:=$ls`
◯ `fichier = $(ls)`

✔ La commande `read` permet de lire le contenu d'un fichier :
◯ Vrai
◯ Faux

✔ Parmi les propositions ci-dessous, laquelle est la bonne syntaxe pour la commande `cut` :
◯ `cut -f: -D1 /etc/passwd`
◯ `cut -d: -f1 /etc/passwd`
◯ `cut -d1 -f: /etc/passwd`
◯ `cut -c ":" -f 3 /etc/passwd`

✔ Quelle commande permet de décaller des variables positionnelles :
◯ `left`
◯ `shift`
◯ `set`
◯ `decalle`

✔ Quelle commande transforme une chaîne de caractères en variables positionnelles :
◯ `left`
◯ `shift`
◯ `set`
◯ `array`

Chapitre 2. Les scripts shell - Instructions de contrôle

🎓 Objectifs pédagogiques

Dans ce chapitre, vous allez apprendre à utiliser des structures de tests, des structures conditionnelles et des boucles.

✔ Tester des variables, des fichiers ;
✔ Utiliser une structure conditionnelle ;
✔ Faire des boucles `while`, `until`, `select`, `for`.

script, **shell**, **bash**, **structure**, **boucle**

Connaissances : ⚙⚙
Niveau technique : ☆☆
Temps de lecture : 20 minutes

Lorsqu'elles se terminent, toutes les commandes exécutées par le shell renvoient un **code de retour** (également appelé **code de statut** ou de **sortie**).

2.1. Les tests

- Si la commande s'est correctement exécutée, la convention veut que le code de statut ait pour valeur **zéro**.
- Si la commande a rencontré un problème lors de son exécution, son code de statut aura une valeur **différente de zéro**.
  Les raisons peuvent être nombreuses : manque de droits d'accès, absence de fichier, saisie incorrecte, etc.

Il faut se référer au manuel de la commande `man commande` pour connaître les différentes valeurs du code de retour prévues par les développeurs.

Le code de retour n'est pas visible directement, mais est enregistré dans une variable spéciale : `$?`.

```
$ mkdir repertoire
$ echo $?
0
$ mkdir /repertoire
mkdir: impossible de créer le répertoire
$ echo $?
1
$ commande_qui_n_existe_pas
commande_qui_n_existe_pas : commande introuvable
$ echo $?
127
```

L'affichage du contenu de la variable **$?** avec la commande **echo** se fait immédiatement après la commande que l'on souhaite évaluer car cette variable est mise à jour après chaque exécution d'une commande, d'une ligne de commandes ou encore d'un script.

Puisque la valeur de **$?** change après chaque exécution de commande, il est préférable de mettre sa valeur dans une variable qui sera utilisée par la suite, pour un test ou afficher un message.

```
$ ls fichier_absent
ls: impossible d'accéder à 'fichier_absent': Aucun fichier ou dossier
de ce type
$ RETOUR=$?
$ echo $?
0
$ echo $RETOUR
2
```

Il est également possible de créer des codes de retour dans un script. Il suffit pour cela d'ajouter un argument numérique à la commande **exit**.

```
$ bash        # pour éviter d'être déconnecté après le « exit 2 »
$ exit 2
$ echo $?
2
```

Outre la bonne exécution d'une commande, le shell offre la possibilité d'exécuter des tests sur de nombreux motifs :

- **Fichiers** : existence, type, droits, comparaison ;
- **Chaînes de caractères** : longueur, comparaison ;

- **Numériques entiers** : valeur, comparaison.

Le résultat du test :

- `$?=0` : le test s'est correctement exécuté et est vrai ;
- `$?=1` : le test s'est correctement exécuté et est faux ;
- `$?=2` : le test ne s'est pas correctement exécuté.

Tester le type d'un fichier

Syntaxe de la commande test pour un fichier

```
test [-d|-e|-f|-L] fichier
```

Table 119. Options de la commande test sur les fichiers

Option	Observation
`-e`	Teste si le fichier existe
`-f`	Teste si le fichier existe et est de type normal
`-d`	Teste si le fichier existe et est de type répertoire
`-L`	Teste si le fichier existe et est de type lien symbolique
`-b`	Teste si le fichier existe et est de type spécial mode bloc
`-c`	Teste si le fichier existe et est de type spécial mode caractère
`-p`	Teste si le fichier existe et est de type tube
`-S`	Teste si le fichier existe et est de type socket
`-t`	Teste si le fichier existe et est de type terminal
`-r`	Teste si le fichier existe et est accessible en lecture
`-w`	Teste si le fichier existe et est accessible en écriture
`-x`	Teste si le fichier existe et est est exécutable
`-g`	Teste si le fichier existe et est a un SGID positionné
`-u`	Teste si le fichier existe et est a un SUID positionné
`-s`	Teste si le fichier existe et est non vide (taille > 0 octets)

Comparer deux fichiers

La commande `test` peut également comparer des fichiers :

Syntaxe de la commande test pour la comparaison de fichiers

```
test fichier1 [-nt|-ot|-ef] fichier2
```

Table 120. Options de la commande test pour la comparaison de fichiers

Option	Observation
`-nt`	Teste si le premier fichier est plus récent que le second
`-ot`	Teste si le premier fichier est plus ancien que le second
`-ef`	Teste si le premier fichier est un lien physique du second

Tester une variable

Syntaxe de la commande test pour les variables

```
test [-z|-n] $variable
```

Table 121. Options de la commande test pour les variables

Option	Observation
`-z`	Teste si la variable est vide
`-n`	Teste si la variable n'est pas vide

Tester une chaîne de caractères

Syntaxe de la commande test pour les chaînes de caractères

```
test chaîne1 [=|!=] chaîne2
```

Exemple :

```
$ test "$var" = "Hello world !"
$ echo $?
0
```

Table 122. Options de la commande test pour les variables

Option	Observation
`=`	Teste si la première chaîne de caractères est égale à la seconde
`!=`	Teste si la première chaîne de caractères est différente de la seconde
`<`	Teste si la première chaîne de caractères est avant la seconde dans l'ordre ASCII
`>`	Teste si la première chaîne de caractères est après la seconde dans l'ordre ASCII

Comparaison de numériques entiers

Syntaxe de la commande test pour les entiers

```
test "num1" [-eq|-ne|-gt|-lt] "num2"
```

Exemple :

```
$ var=1
$ test "$var" -eq "1"
$ echo $?
0
$ var=2
$ test "$var" -eq "1"
$ echo $?
1
```

Table 123. Options de la commande test pour les entiers

Option	Observation
-eq	Teste si le premier nombre est égal au second
-ne	Teste si le premier nombre est différent au second
-gt	Teste si le premier nombre est supérieur au second
-lt	Teste si le premier nombre est inférieur au second

Les numériques étant traités par le shell comme des caractères (ou chaînes de caractères) classiques, un test sur un caractère peut renvoyer le même résultat qu'il soit traité en tant que numérique ou non.

```
$ test "1" = "1"
$ echo $?
0
$ test "1" -eq "1"
$ echo $?
0
```

Mais le résultat du test n'aura pas la même signification :

- Dans le premier cas, il signifiera que les deux caractères ont la même valeur dans la table ASCII.
- Dans le second cas, il signifiera que les deux nombres sont égaux.

Combinaison de tests

La combinaison de tests permet d'effectuer plusieurs tests en une seule commande. Il est possible de tester plusieurs fois le même argument (fichier, chaîne ou numérique) ou des arguments différents.

```
test option1 argument1 [-a|-o] option2 argument 2
```

```
$ ls -lad /etc
drwxr-xr-x 142 root root 12288 sept. 20 09:25 /etc
$ test -d /etc -a -x /etc
$ echo $?
0
```

Table 124. Options de combinaison de tests

Option	Observation
`-a`	**ET** : Le test sera vrai si tous les motifs le sont.
`-o`	**OU** : Le test sera vrai si au moins un motif l'est.

Les tests peuvent ainsi être groupé avec des parenthèses `( )` pour leur donner une priorité.

```
(TEST1 -a TEST2) -a TEST3
```

Le caractère `!` permet d'effectuer le test inverse de celui demandé par l'option :

```
$ test -e /fichier   # vrai si fichier existe
$ ! test -e /fichier # vrai si fichier n'existe pas
```

Les opérations numériques

La commande `expr` effectue une opération avec des entiers numériques.

Syntaxe de la commande expr

```
expr num1 [+] [-] [\*] [/] [%] num2
```

Exemple :

Exemple d'utilisation de la commande expr

```
$ expr 2 + 2
4
```

Attention à bien encadrer le signe d'opération par une espace, vous obtiendrez un message d'erreur en cas d'oubli.
Dans le cas d'une multiplication, le caractère joker `*` est précédé par `\` pour éviter une mauvaise interprétation.

Table 125. Opérateurs de la commande expr

Opérateur	Observation
`+`	Addition
`-`	Soustraction
`\*`	Multiplication
`/`	Quotient de la division
`%`	Modulo de la division

La commande `typeset`

La commande `typeset -i` déclare une variable comme un entier.

Exemple :

Exemple d'utilisation de la commande typeset

```
$ typeset -i var1
$ var1=1+1
$ var2=1+1
$ echo $var1
2
$ echo $var2
1+1
```

La commande `let`

La commande `let` teste si un caractère est numérique.

Exemple :

Exemple d'utilisation de la commande let

```
$ var1="10"
$ var2="AA"
$ let $var1
$ echo $?
0
$ let $var2
$ echo $?
1
```

La commande `let` ne retourne pas un code retour cohérent lorsqu'elle évalue le numérique `0`.

```
$ let 0
$ echo $?
1
```

La commande `let` permet également d'effectuer des opérations mathématiques :

```
$ let var=5+5
$ echo $var
10
```

`let` peut être substituée par `$(( ))`.

```
$ echo $((5+2))
7
$ echo $((5*2))
10
$ var=$((5*3))
$ echo $var
15
```

2.2. Structures conditionnelles

Si la variable `$?` permet de connaître le résultat d'un test ou de l'exécution d'une commande, elle ne peut qu'être affichée et n'a aucune incidence sur le déroulement d'un script.

Mais nous pouvons nous en servir dans une condition. **Si** le test est bon **alors** je fais cette action **sinon** je fais telle autre action.

Syntaxe de l'alternative conditionnelle if

```
if commande
then
    commande si $?=0
else
    commande si $?!=0
fi
```

La commande placée après le mot **if** peut être n'importe quelle commande puisque c'est son code de retour (**$?**) qui sera évalué. Il est souvent pratique d'utiliser la commande **test** pour définir plusieurs actions en fonction du résultat de ce test (fichier existe, variable non vide, droits en écriture positionnés).

Utiliser une commande classique (**mkdir**, **tar**, ...) permet de définir les actions à effectuer en cas de succès ou les messages d'erreur à afficher en cas d'échec.

Exemples d'utilisation de la structure conditionnelle if

```
if test -e /etc/passwd
then
    echo "Le fichier existe"
else
    echo "Le fichier n'existe pas"
fi

if mkdir rep
then
    cd rep
fi
```

La commande `test` peut être substituée par `[[ condition_de_test ]]`.

Ainsi :

```
if test -e /etc/passwd
then
    echo "Le fichier existe"
else
    echo "Le fichier n'existe pas"
fi
```

peut devenir :

```
if [[ -e /etc/passwd ]]
then
    echo "Le fichier existe"
else
    echo "Le fichier n'existe pas"
fi
```

Si le bloc `else` commence par une nouvelle structure `if`, il est possible de fusionner `else` et `if` :

```
[…]
else
  if test -e /etc/
[…]

[…]
# est équivalent à
elif test -e /etc
[…]
```

La structure `if` / `then` / `else` / `fi` évalue la commande placée après `if` :

- Si le code retour de cette commande est 0 (vrai) le shell exécutera les commandes placées après `then` ;
- Si le code retour est différent de 0 (faux) le shell exécutera les commandes placées après `else`.

Le bloc `else` est facultatif.

Il existe un besoin d'effectuer certaines actions uniquement si l'évaluation de la commande est vraie et n'avoir rien à faire si elle est fausse.

Le mot `fi` ferme la structure.

Lorsqu'il n'y a qu'une seule commande à exécuter dans le bloc `then`, il est possible d'utiliser une syntaxe plus simple.

La commande à exécuter si `$?` est vrai est placée après `&&` tandis que la commande à exécuter si `$?` est faux est placée après `||` (*facultatif*).

Par exemple :

```
$ test -e /etc/passwd && echo "Le fichier existe" || echo "Le fichier n'existe pas"
$ mkdir repert && echo "Le répertoire est créé"
```

Il est possible d'évaluer et de remplacer une variable avec une structure plus légère que `if`.

Cette syntaxe met en œuvre les accolades :

- Affiche une valeur de remplacement si la variable est vide :

```
${variable:-valeur}
```

- Affiche une valeur de remplacement si la variable n'est pas vide :

```
${variable:+valeur}
```

- Affecte une nouvelle valeur à la variable si elle est vide :

```
${variable:=valeur}
```

Exemples :

```
$ nom=""
$ echo ${nom:-linux}
linux
$ echo $nom

$ echo ${nom:=linux}
linux
$ echo $nom
linux
$ echo ${nom:+tux}
tux
$ echo $nom
linux
```

Structure alternative conditionnelle `case`

Une succession de structures `if` peut vite devenir lourde et complexe. Lorsqu'elle concerne l'évaluation d'une même variable, il est possible d'utiliser une structure conditionnelle à plusieurs branches. Les valeurs de la variable peuvent être précisées ou appartenir à une liste de possibilités.

Les **caractères jokers sont utilisables**.

La structure `case ··· in` / `esac` évalue la variable placée après `case` et la compare aux valeurs définies.

À la première égalité trouvée, les commandes placées entre `)` et `;;` sont exécutées.

La variable évaluée et les valeurs proposées peuvent être des chaînes de caractères ou des résultats de sous-exécutions de commandes.

Placé en fin de structure, le choix `*` indique les actions à exécuter pour toutes les valeurs qui n'ont pas été précédemment testées.

Syntaxe de l'alternative conditionnelle case

```
case $variable in
  valeur1)
    commandes si $variable = valeur1
    ;;
  valeur2)
    commandes si $variable = valeur2
    ;;
  [..]
  *)
    commandes pour toutes les valeurs de $variable != de valeur1 et valeur2
    ;;
esac
```

Lorsque la valeur est sujette à variation, il est conseillé d'utiliser les caractères jokers `[]` pour spécifier les possibilités :

```
[Oo][Uu][Ii])
  echo "oui"
  ;;
```

Le caractère `|` permet aussi de spécifier une valeur ou une autre :

```
"oui" | "OUI")
  echo "oui"
  ;;
```

2.3. Boucles

Le shell bash permet l'utilisation de **boucles**. Ces structures permettent l'exécution d'un **bloc de commandes plusieurs fois** (de 0 à l'infini) selon une valeur définie statiquement, dynamiquement ou sur condition :

- `while`
- `until`
- `for`
- `select`

Quelle que soit la boucle utilisée, les commandes à répéter se **placent entre les mots `do` et `done`**.

La structure boucle conditionnelle `while`

La structure `while` / `do` / `done` évalue la commande placée après `while`.

Si cette commande est vrai (`$? = 0`), les commandes placées entre `do` et `done` sont exécutées. Le script retourne ensuite au début évaluer de nouveau la commande.

Lorsque la commande évaluée est fausse (`$? != 0`), le shell reprend l'exécution du script à la première commande après `done`.

Syntaxe de la structure boucle conditionnelle while

```
while commande
do
  commande si $? = 0
done
```

Exemple :

Exemple d'utilisation de la structure conditionnelle while

```
while test -e /etc/passwd
do
  echo "Le fichier existe"
done
```

Si la commande évaluée ne varie pas, la boucle sera infinie et le shell n'exécutera jamais les commandes placées à la suite du script. Cela peut être volontaire, mais aussi être une erreur.

Il faut donc **faire très attention à la commande qui régit la boucle et trouver un moyen d'en sortir**.

Pour sortir d'une boucle `while`, il faut faire en sorte que la commande évaluée ne soit plus vraie, ce qui n'est pas toujours possible.

Il existe des commandes qui permettent de modifier le comportement d'une boucle :

- `exit`
- `break`
- `continue`

La commande `exit`

La commande `exit` termine l'exécution du script.

Syntaxe de la commande exit

```
exit [n]
```

Exemple :

Exemple d'utilisaton de la commande exit

```
$ bash        # pour éviter d'être déconnecté après le « exit 1 »
$ exit 1
$ echo $?
1
```

La commande `exit` met fin au script immédiatement. Il est possible de préciser le code de retour du script en le précisant en argument (*de 0 à 255*). Sans argument précisé, c'est le code de retour de la dernière commande du script qui sera transmise à la variable `$?`.

Cette commande est utile dans le cas d'un menu proposant la sortie du script dans les choix possibles.

La commande `break` / `continue`

La commande `break` permet d'interrompre la boucle en allant à la première commande après `done`.

La commande `continue` permet de relancer la boucle en revenant à la première commande après `do`.

```
while test -d /
do
  echo "Voulez-vous continuer ? (oui/non)"
  read rep
  test $rep = "oui" && continue
  test $rep = "non" && break
done
```

Les commandes `true` / `false`

La commande `true` renvoie toujours vrai tandis que la commande `false` renvoie toujours faux.

```
$ true
$ echo $?
0
$ false
$ echo $?
1
```

Utilisées comme condition d'une boucle, elles permettent soit d'exécuter une boucle infinie soit de désactiver cette boucle.

Exemple :

```
while true
do
  echo "Voulez-vous continuer ? (oui/non)"
  read rep
  test $rep = "oui" && continue
  test $rep = "non" && break
done
```

La structure boucle conditionnelle `until`

La structure `until` / `do` / `done` évalue la commande placée après `until`.

Si cette commande est fausse (`$? != 0`), les commandes placées entre `do` et `done` sont exécutées. Le script retourne ensuite au début évaluer de nouveau la commande.

Lorsque la commande évaluée est vraie (`$? = 0`), le shell reprend l'exécution du script à la première commande après `done`.

Syntaxe de la structure boucle conditionnelle until

```
until commande
do
  commande si $? != 0
done
```

Exemple :

Exemple d'utilisation de la structure conditionnelle until

```
until test -e /etc/passwd
do
  echo "Le fichier n'existe pas"
done
```

La structure choix alternatif `select`

La structure `select` / `do` / `done` permet d'afficher rapidement un menu avec plusieurs choix et une demande de saisie.

À chaque élément de la liste correspond un choix numéroté. À la saisie, la valeur choisie est affectée à la variable placée après `select` (*créée à cette occasion*).

Elle exécute ensuite les commandes placées entre `do` et `done` avec cette valeur.

- La variable `PS3` contient le message d'invitation à entrer le choix ;
- La variable `REPLY` va permettre de récupérer le numéro du choix.

Il faut une commande `break` pour sortir de la boucle.

La structure `select` est très utile pour de petits menus simples et rapides. Pour personnaliser un affichage plus complet, il faudra utiliser les commandes `echo` et `read` dans une boucle `while`.

Syntaxe de la structure boucle conditionnelle select

```
PS3="Votre choix :"
select variable in var1 var2 var3
do
  commandes
done
```

Exemple :

Exemple d'utilisation de la structure conditionnelle select

```
PS3="Votre choix : "
select choix in café thé chocolat
do
  echo "Vous avez choisi le $REPLY : $choix"
done
```

ce qui donne à l'exécution :

```
1) Café
2) Thé
3) Chocolat
Votre choix : 2
Vous avez choisi le choix 2 : thé
Votre choix :
```

La structure boucle sur liste de valeurs `for`

La structure `for` / `do` / `done` affecte le premier élément de la liste à la variable placée après `for` (*créée à cette occasion*).

Elle exécute ensuite les commandes placées entre `do` et `done` avec cette valeur. Le script retourne ensuite au début affecter l'élément suivant de la liste à la variable de travail.

Lorsque le dernier élément a été utilisé, le shell reprend l'exécution à la première commande après `done`.

Syntaxe de la structure boucle sur liste de valeurs for

```
for variable in liste
do
  commandes
done
```

Exemple :

Exemple d'utilisation de la structure conditionnelle for

```
for fichier in /home /etc/passwd /root/fic.txt
do
  file $fichier
done
```

Toute commande produisant une liste de valeurs peut être placée à la suite du `in` à l'aide d'une sous-exécution.

- Avec la variable `IFS` contenant `$' \t\n'`, la boucle `for` prendra **chaque mot** du résultat de cette commande comme liste d'éléments sur laquelle boucler .
- Avec la variable `IFS` contenant `$'\t\n'` (c'est-à-dire sans espace), la boucle `for` prendra **chaque ligne** du résultat de cette commande.

Cela peut être les fichiers d'un répertoire. Dans ce cas, la variable prendra comme valeur chacun des mots des noms des fichiers présents :

```
for fichier in $(ls -d /tmp/*)
do
  echo $fichier
done
```

Cela peut être un fichier. Dans ce cas, la variable prendra comme valeur chaque mot contenu dans le fichier parcouru, du début à la fin :

```
$ cat mon_fichier.txt
première ligne
seconde ligne
troisième ligne
$ for LIGNE in $(cat mon_fichier.txt); do echo $LIGNE; done
première
ligne
seconde
ligne
troisième
ligne
```

Pour lire ligne par ligne un fichier, il faut modifier la valeur de la variable d'environnement `IFS`.

```
$ IFS=$'\t\n'
$ for LIGNE in $(cat mon_fichier.txt); do echo $LIGNE; done
première ligne
seconde ligne
troisième ligne
```

Tester vos connaissances

✔ Toute commande renvoie obligatoirement un code de retour à la fin de son exécution :
☐ Vrai
☐ Faux

✔ Un code de retour à `0` indique une erreur d'exécution :
☐ Vrai

☐ Faux

✔ Le code de retour est stocké dans la variable `$@` :
☐ Vrai
☐ Faux

✔ La commande `test` permet de :
☐ Tester le type d'un fichier
☐ Tester une variable
☐ Comparer des numériques
☐ Comparer le contenu de 2 fichier

✔ La commande `expr` :
☐ Concatène 2 chaînes de caractères
☐ Effectue des opérations mathématiques
☐ Affiche du texte à l'écran

✔ La syntaxe de la structure conditionnelle ci-dessous vous semble-t'elle correcte ? Expliquez pourquoi.
☐ Vrai
☐ Faux

```
if commande
    commande si $?=0
else
    commande si $?!=0
fi
```

✔ Que sigifie la syntaxe suivante : `${variable:=valeur}`
☐ Affiche une valeur de remplacement si la variable est vide
☐ Affiche une valeur de remplacement si la variable n'est pas vide
☐ Affecte une nouvelle valeur à la variable si elle est vide

✔ La syntaxe de la structure alternative conditionnelle ci-dessous vous semble-t'elle correcte ? Expliquez pourquoi.
☐ Vrai
☐ Faux

```
case $variable in
  valeur1)
    commandes si $variable = valeur1
  valeur2)
    commandes si $variable = valeur2
  *)
    commandes pour toutes les valeurs de $variable != de valeur1 et valeur2
    ;;
esac
```

✔ Parmi les propositions ci-dessous, laquelle n'est pas une structure pour faire une boucle :
☐ while
☐ until
☐ loop
☐ for

✔ La commande `true` renvoit toujours `1`:
☐ Vrai
☐ Faux

Chapitre 3. TP Scripting shell

Votre entreprise a besoin d'une solution sécurisée permettant aux personnels de la supervision d'intervenir dans un cadre maîtrisé sur les serveurs.

3.1. Étude du besoin

Votre responsable vous demande de développer un outil destiné aux superviseurs. Ils pourront effectuer quelques actions d'administration ainsi que les premiers diagnostics avant de faire intervenir le personnel d'astreinte.

Le personnel doit pouvoir se connecter aux serveurs via un compte générique : **supervision**.

Lorsque l'utilisateur se connecte, un menu est proposé, lui permettant :

- De gérer les utilisateurs :
  - afficher le nombre d'utilisateurs du serveur et les afficher sous formes de 2 listes :
    - les utilisateurs systèmes,
    - les utilisateurs standards ;
  - afficher les groupes du serveur et les afficher sous forme de 2 listes :
    - les groupes systèmes,
    - les groupes standards ;
  - créer un groupe : le superviseur devra fournir le GID ;
  - créer un utilisateur : le superviseur devra fournir l'UID, le GID, etc. ;
  - changer le mot de passe d'un utilisateur ; l'utilisateur sera forcé de changer son mot de passe lors de sa prochaine connexion.
- De gérer les services :
  - relancer le serveur apache ;
  - relancer le serveur postfix.
- De tester le réseau :
  - Afficher les informations du réseau (Adresse IP, masque, passerelle, serveurs DNS) ;
  - Tester le réseau (localhost, ip, passerelle, serveur distant, résolution DNS).
- Actions diverses :
  - redémarrer le serveur ;
  - quitter le script (l'utilisateur est déconnecté).

Les actions du superviseur devront être renseignées dans les journaux systèmes.

3.2. Consignes

- Les scripts sont stockés dans /opt/supervision/scripts/ ;
- Effectuer tous les tests que vous jugerez nécessaires ;
- Découper le code en plusieurs scripts ;
- Utiliser des fonctions pour organiser le code ;
- Commenter le code.

3.3. Pistes de travail

- L'utilisateur supervision aura besoin des droits sudo pour les commandes réservées à root.
- Le système attribue le shell /bin/bash à un utilisateur standard, tentez d'attribuer votre script à la place !
- Utilisez la commande logger pour suivre les actions des superviseurs.
- Visitez le site : https://www.shellcheck.net/

3.4. Proposition de correction

Le code présenté ci-dessous n'est qu'une ébauche effectuée en TP par des stagiaires après 12 heures de cours de script. Il n'est pas parfait mais peut servir de base de correction ou de départ pour l'élaboration d'un travail plus complet.

Création de l'utilisateur

L'utilisateur doit être créé en remplaçant son shell (option -s) par le script que nous allons créer :

```
useradd -s /opt/supervision/scripts/supervision.sh -g users supervision
```

Il faut autoriser l'utilisateur supervision à utiliser sudo mais seulement pour les commandes autorisées. Pour cela, nous allons créer un fichier /etc/sudoers.d/supervision contenant les directives nécessaires :

```
# Liste les commandes autorisees aux superviseurs
Cmnd_Alias SUPERVISION = /sbin/reboot, /sbin/ip

# Autorise le superviseur a lancer les commandes precedentes sans saisir de mot de
passe
supervision    ALL=NOPASSWD:SUPERVISION
```

Menu

Créer le fichier /opt/supervision/scripts/supervision.sh et lui donner les droits en exécution :

```
mkdir -p /opt/supervision/scripts
touch /opt/supervision/scripts/supervision.sh
chown supervision /opt/supervision/scripts/*
chmod u+x /opt/supervision/scripts/*
```

La même opération sera effectuée pour chaque script créé.

```
#!/bin/bash

# Base des scripts

BASE=$(dirname "$0")
readonly BASE

. $BASE/utils.sh

function print_menu {
  while (true)
  do
    clear
    banner
    warning
    echo "Vous pouvez : "
    echo ""
    echo "  => 1) Relancer le serveur"
    echo "  => 2) Afficher la conf IP"
    echo "  => 3) Tester le reseau  "
    echo "  => 4) Afficher les utilisateurs"
    echo "  => 5) Relancer le service apache"
    echo "  => 6) Relancer le service postfix"
    echo "  => Q) Quitter ce super programme "
    echo ""
    read -p "Que voulez vous faire : " choix
    echo ""
    case $choix in
      "1")
        sudo reboot
        ;;
      "2")
        $BASE/print-net.sh
        ;;
      "3")
        $BASE/check-net.sh
```

```
        ;;
      "4")
        $BASE/affiche-utilisateurs.sh
        ;;
      "5")
        $BASE/gestion-services.sh "httpd"
        ;;
      "6")
        $BASE/gestion-services.sh "postfix"
        ;;
      "q" | "Q" | "quitter" | "quit")
        exit 0
        ;;
      *)
        echo "Cette fonction n'est pas encore developpee"

    esac
    pause
  done
}
banner

echo "Bienvenue sur votre console d'administration"
echo ""
echo "Vous pouvez effectuer quelques diagnostics avant d'appeler le personnel
d'astreinte"
echo ""
warning

pause

print_menu

exit 0
```

Quelques fonctions utilitaires

Le fichier utils.sh contient des fonctions que nous utiliserons dans chaque script :

```
#!/bin/bash
#
# Fonctions utilitaires et variables globales
# Version 1
#
ok="                                          [OK]"
nok="                                         [NOK]"

# Affiche ok ou nok
# Arguments :
# $1 = 0 ou 1
# $2 = message a imprimer
function printOK {
  echo "$1"
  if test "$2" = "0"
  then
    echo "$ok"
  else
    echo "$nok"
  fi
}

function banner {
echo "*            Bienvenue dans l'outil de la          *"
echo "               S U P E R V I S I O N               "
echo "                                    _     _          "
echo " ___ _   _ _ __   ___ _ ____   _(_)___(_) ___  _ __  "
echo "/ __| | | | '_ \ / _ \ '__\ \ / / / __| |/ _ \| '_ \ "
echo "\__ \ |_| | |_) |  __/ |   \ V /| \__ \ | (_) | | | |"
echo "|___/\__,_| .__/ \___|_|    \_/ |_|___/_|\___/|_| |_|"
echo "          |_|                                       "
}



function pause {
  echo "Appuyer sur Entree pour continuer..."
  read
}

function warning {
  echo "ATTENTION !!!"
  echo "Toutes les actions entreprises sont renseignees dans le journal d'evenement !"
  echo ""
}
```

Le fichier net-utils.sh contient les fonctions liées au réseau :

```
#!/bin/bash

#
# Fonction utilitaires du reseau
# Version 1
# Depends de utils.sh

function getGateway {
  gateway=$(sudo ip route | grep default | cut -d" " -f3)
  echo $gateway
}

function getDNS {
  DNS=$(grep "nameserver" /etc/resolv.conf | tail -1 | cut -d" " -f2)
  echo $DNS
}

# Test une adresse IP
function checkIP {
  ip=$1
  msg="Test de l'adresse ip : $ip"
  ping -c 1 $ip 1> /dev/null 2>&1
  printOK "$msg" "$?"
}

# test une resolution DNS
function checkDNS {
  res=$(dig +short www.free.fr | wc -l)
  if test "$res" -gt "0"
  then
    printOK "La resolution DNS fonctionne" "0"
  else
    printOK "La resolution DNS ne fonctionne pas" "1"
  fi
}

function getPrefix {
  sudo ip add sh | grep " inet " | grep -v "127.0.0.1" | tr -s ' ' | cut -d" " -f 3 |
cut -d "/" -f2
}
```

La gestion du réseau

Le fichier print-net.sh :

```
#!/bin/bash

#
# Test du reseau
# Version 1
#
# Arguments :
#
ici=$(dirname "$0")
. $ici/utils.sh
. $ici/net-utils.sh

echo "L'adresse IP de votre serveur est     : $(hostname -i)"
echo "L'adresse IP de votre gateway est     : $(getGateway)"
echo "L'adresse IP de votre serveur DNS est : $(getDNS)"
echo -n "Votre prefix est : "
getPrefix

echo ""
```

Le fichier check-net.sh :

```
#!/bin/bash

#
# Test du reseau
# Version 1
#
# Arguments :
#
ici=$(dirname "$0")
. $ici/utils.sh
. $ici/net-utils.sh

# Gestion du service fourni en argument
checkIP 127.0.0.1
checkIP $(hostname -i)
checkIP $(getGateway)
checkIP $(getDNS)
checkDNS
```

La gestion des services

```
#!/bin/bash
```

```
#
# Gestion des services
# Version 1
#
# Arguments :
# $1 : le nom du service a relancer
#
. ./utils.sh

# Test l'etat du service
# Si le service est demarre, il propose de le relancer
# Sinon le service est demarre
function startService {
  service=$1
  service $service status 1> /dev/null 2>&1
  status=$?
  if test "$status" = "0"
  then
    # Le service fonctionne deja
    # Faut-il le relancer ?
    echo "Le service $service fonctionne..."
    read -p "Voulez vous le relancer ? O/N " rep
    if test "$rep" = "O" -o "$rep" = "o"
    then
      # L'utilisateur a demande a le relancer
      logger "SUPP -> Relance d'apache"
      msg="Relance du serveur $service"
      service $service restart 1> /dev/null 2>&1
      printOK "$msg" "$?"
    else
      # L'utilisateur ne veut pas le relancer
      msg="Le service ne sera pas relance"
      printOK "$msg" "0"
    fi
  else
    # Le service ne fonctionne pas
    # Demarrage
    logger "SUPP -> Demarrage d'apache"
    msg="Lancement du serveur $service"
    service $service start 1> /dev/null 2>&1
    printOK "$msg" "$?"
  fi
}

# Gestion du service fourni en argument
service=$1
startService $service
```

L’affichage des utilisateurs

Le fichier affiche-utilisateur.sh :

```
#!/bin/bash

# Extrait du fichier /etc/passwd la liste :
# - des utilisateurs du systeme
# - des utilisateurs standards
# Chaque liste est affiche sur une ligne
#
# Version 1.0
# Date : 24/11/2016

# usersys : la liste des utilisateurs systemes
usersys="Voici la liste des utilisateurs systemes :\n"
# userstd : la liste des utilisateurs standards
userstd="Voici la liste des utilisateurs standard :\n"

# Stocker l'IFS dans une variable
OLDIFS='$IFS'
# Pour que la commande for fonctionne, il faut supprimer l'espace comme caractere de
separation
IFS=$'\n'
# On boucle sur chaque ligne du fichier /etc/passwd
while read -r ligne
do
  # Isoler l'UID
  uid=$(echo $ligne | cut -d: -f3)
  # Isoler le Nom
  nom=$(echo $ligne | cut -d: -f1)
  # Si uid < 500 => Utilisateur systeme
  if test "$uid" -lt "500"
  then
    # Ajouter le nom a la liste
    usersys="${usersys}${nom}, "
  else
    # Ajouter le nom a la liste
    userstd="${userstd}${nom}, "
  fi
done < /etc/passwd


# Affichage de la liste
echo -e "$usersys"
echo ""
echo -e "$userstd"

IFS=$OLDIFS
```

Partie 6 : Annexes

Chapitre 1. Memento VI

Lancer `vim` ou `vimtutor`

```
vi [-c commande] [fichier]
$ vi -c "set nu" /home/stagiaire/fichier
$ vimtutor
```

1.1. Mode Commandes

Table 126. Commandes vi

Caractères	Actions
[←] ou [n][←]	Déplacement d'un ou n caractères vers la gauche
[→] ou [n][→]	Déplacement d'un ou n caractères vers la droite
[↑] ou [n][↑]	Déplacement d'un ou n caractères vers le haut
[↓] ou [n][↓]	Déplacement d'un ou n caractères vers le bas
[$] ou [FIN]	Déplacement à la fin de la ligne
[0] ou [POS1]	Déplacement au début de la ligne
[w] ou [n][w]	Déplacement d'un ou n mots vers la droite
[b] ou [n][b]	Déplacement d'un ou n mots vers la gauche
[G]	Déplacement à la dernière ligne du texte
[n][G]	Déplacement à la ligne n
[H]	Déplacement à la première ligne de l'écran
[M]	Déplacement à la ligne du milieu de l'écran
[L]	Déplacement à la dernière ligne de l'écran
[i]	Insertion de texte avant un caractère
[a]	Insertion de texte après un caractère
[I]	Insertion de texte au début d'une ligne
[A]	Insertion de texte à la fin d'une ligne
[O]	Insertion de texte avant une ligne
[o]	Insertion de texte après une ligne
[x] ou *[n][x]	Supprimer un ou n caractères
[r][caractère]	Remplacer un caractère par un autre
[R][caractères][ECHAP]	Remplacer plus d'un caractère par d'autres

Caractères	Actions
[d][w] ou [n][d][w]	Supprimer (couper) un ou n mots
[y][w] ou [n][y][w]	Copier un ou n mots
[p] ou [n][p]	Coller un mot une ou n fois après le curseur
[P] ou [n][P]	Coller un mot une ou n fois avant le curseur
[c][w][mot][ECHAP]	Remplacer un mot
[d][d] ou [n][d][d]	Supprimer (couper) une ou n lignes
[y][y] ou [n][y][y]	Copier une ou n lignes
[p] ou [n][p]	Coller ce qui a été copié ou supprimé une ou n fois après la ligne courante
[P] ou [n][P]	Coller ce qui a été copié ou supprimé une ou n fois avant la ligne courante
[d][0]	Supprimer (couper) du début de la ligne jusqu'au curseur
[d][$]	Supprimer (couper) du curseur jusqu'à la fin de la ligne
[y][0]	Copier du début de la ligne jusqu'au curseur
[y][$]	Copier du curseur jusqu'à la fin de la ligne
[d][L] ou [d][G]	Supprimer (couper) le texte à partir de la ligne courante
[y][L] ou [y][G]	Copier le texte à partir de la ligne courante
[u]	Annuler la dernière action
[U]	Annuler les actions sur la ligne courante

1.2. Mode EX

Table 127. Commandes du mode Ex

Caractères	Actions
:set nu	Afficher la numérotation
:set nonu	Mmasquer la numérotation
/chaîne	Rechercher une chaîne de caractères à partir du curseur
?chaîne	Rechercher une chaîne de caractères avant le curseur
[n]	Aller à l'occurrence trouvée suivante
[N]	Aller à l'occurence trouvée précédente
/[Mm]ot	Recherche d'un unique caractère dont les valeurs possibles sont précisées
/Mot,^	Recherche d'une chaîne débutant la ligne

Caractères	Actions
`/Mot,$`	Recherche d'une chaîne finissant la ligne
`/M*t`	Recherche d'un ou de plusieurs caractères, quels qu'ils soient
`:1,$s/recherche/remplace`	De la 1ère à la dernière ligne du texte, remplacer la chaîne recherchée par la chaîne précisée
`:n,ms/recherche/remplace`	De la ligne `n` à la ligne `m`, remplacer la chaîne recherchée par la chaîne précisée
`:n,ms/recherche/remplace/g`	Par défaut, seule la première occurence trouvée de chaque ligne est remplacée. Pour forcer le remplacement de chaque occurence, il faut ajouter `/g` à la fin de la commande
`:w`	Enregistrer le fichier
`:w fichier`	Enregistrer sous un autre nom
`:n,mw fichier`	Enregistrer de la ligne `n` à la ligne `m` dans un autre fichier
`e!`	Recharger le dernier enregistrement du fichier
`:r fichier`	Coller le contenu d'un autre fichier après le curseur
`:q`	Quitter le fichier sans enregistrer
`:wq` ou `:x`	Quitter le fichier et enregistrer

Glossaire et index

Glossaire

BASH

Bourne Again SHell

Index

T

U

V

W