

# Administration Linux

---





1. Administration Linux . . . . .	3	1.12 Commande pour gérer une machine à base de Debian . . . . .	21	2. Installation d'un serveur web local . . . . .	39
1.1 Linux ou GNU/LINUX . . . . .	4	1.13 Réseaux . . . . .	24	2.1 LEMP : Linux Nginx MySQL/MariaDB PHP . . . . .	40
1.2 Système d'exploitation Open Source . . . . .	5	1.14 For fun . . . . .	25	2.2 Installation de php . . . . .	40
1.3 Premier système d'exploitation au monde sur serveurs . . . . .	6	1.15 Commande d'informations sur un programme . . . . .	26	2.3 Installation du serveur web nginx . . . . .	43
1.4 Linux est distribué sous licence GNU GPL . . . . .	7	1.16 Commandes shell . . . . .	27	2.4 LAMP : Linux Apache MySQL/MariaDB PHP . . . . .	45
1.5 Unix / Windows . . . . .	8	1.17 Commande sur le système de fichier . . . . .	28	2.5 Installation du serveur web Apache . . . . .	46
1.6 Fonctionnement d'une machine sous Linux . . . . .	10	1.18 Information sur sa machine . . . . .	32	2.6 Installation d'un serveur de base de donnée mariadb/mysql . . . . .	49
1.7 Distribution Linux . . . . .	11	1.19 Commande diverses . . . . .	33	2.7 Tester des pages html,php . . . . .	52
1.8 Répertoires d'un système GNU/Linux . . . . .	12	1.20 Bash . . . . .	34	2.8 Compression . . . . .	53
1.9 SSH Connexion . . . . .	16	1.21 Les fichiers de configuration BASH . . . . .	35	2.9 Sauvegarder une base de donnée mariadb avec /mariadb-dump . . . . .	55
1.10 Oracle VM VirtualBox . . . . .	18	1.22 Piping . . . . .	36		
1.11 Debian . . . . .	19	1.23 Gestion des droits d'accès . . . . .	38		



2.10 Monitorer sa machine . . . . .	57
3. Vagrant . . . . .	60
3.1 Création de VMs . . . . .	61
3.2 Configuration . . . . .	62
3.3 Dépannage . . . . .	63
4. Parefeu (Firewall) . . . . .	64
4.1 UFW (UncomplicatedFirewall) . . . . .	65
5. Securiser son serveur . . . . .	66
5.1 Debian . . . . .	67
5.2 Simulation de charge du serveur . . . . .	68
6. Ressources . . . . .	69
6.1 Sécurité informatique . . . . .	70
6.2 Sources . . . . .	71

# **1. Administration Linux**

---

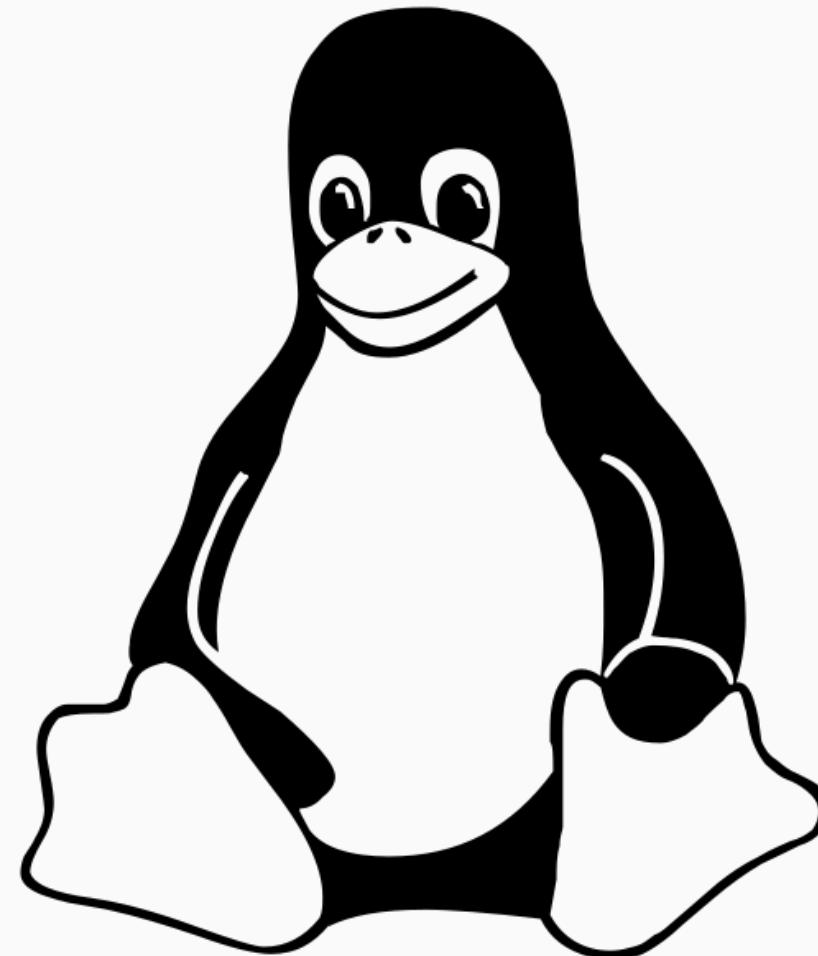


Figure 1: **Tux** le manchot mascotte officielle du **noyau Linux**



[https://fr.wikipedia.org/wiki/Linux\\_ou\\_GNU/Linux](https://fr.wikipedia.org/wiki/Linux_ou_GNU/Linux)

**OS** : Operating System

Crée en **1991** par **Linus Torvalds**

Création en tant que passe-temps !

Basé sur l'**OS MINIX**

Lui même basé sur les principes et la conception d'**Unix**



Seul **OS** à être utilisé sur les **500 Super Ordinateurs** les plus rapides au monde

<https://www.top500.org/statistics/sublist/>



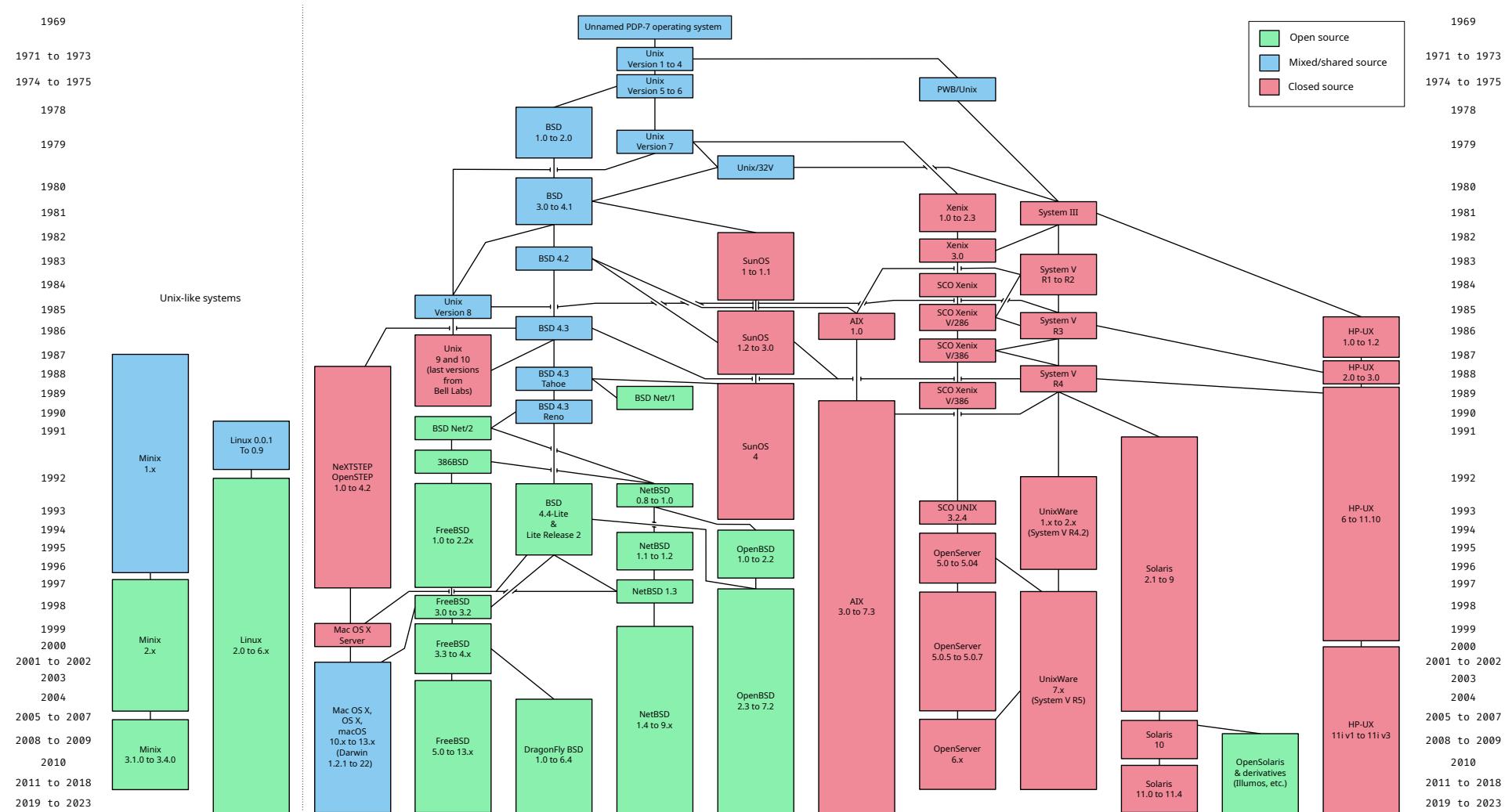
[https://fr.wikipedia.org/wiki/GNU\\_GPL](https://fr.wikipedia.org/wiki/GNU_GPL)

- Exécuter
- Étudier
- Partager
- Modifier

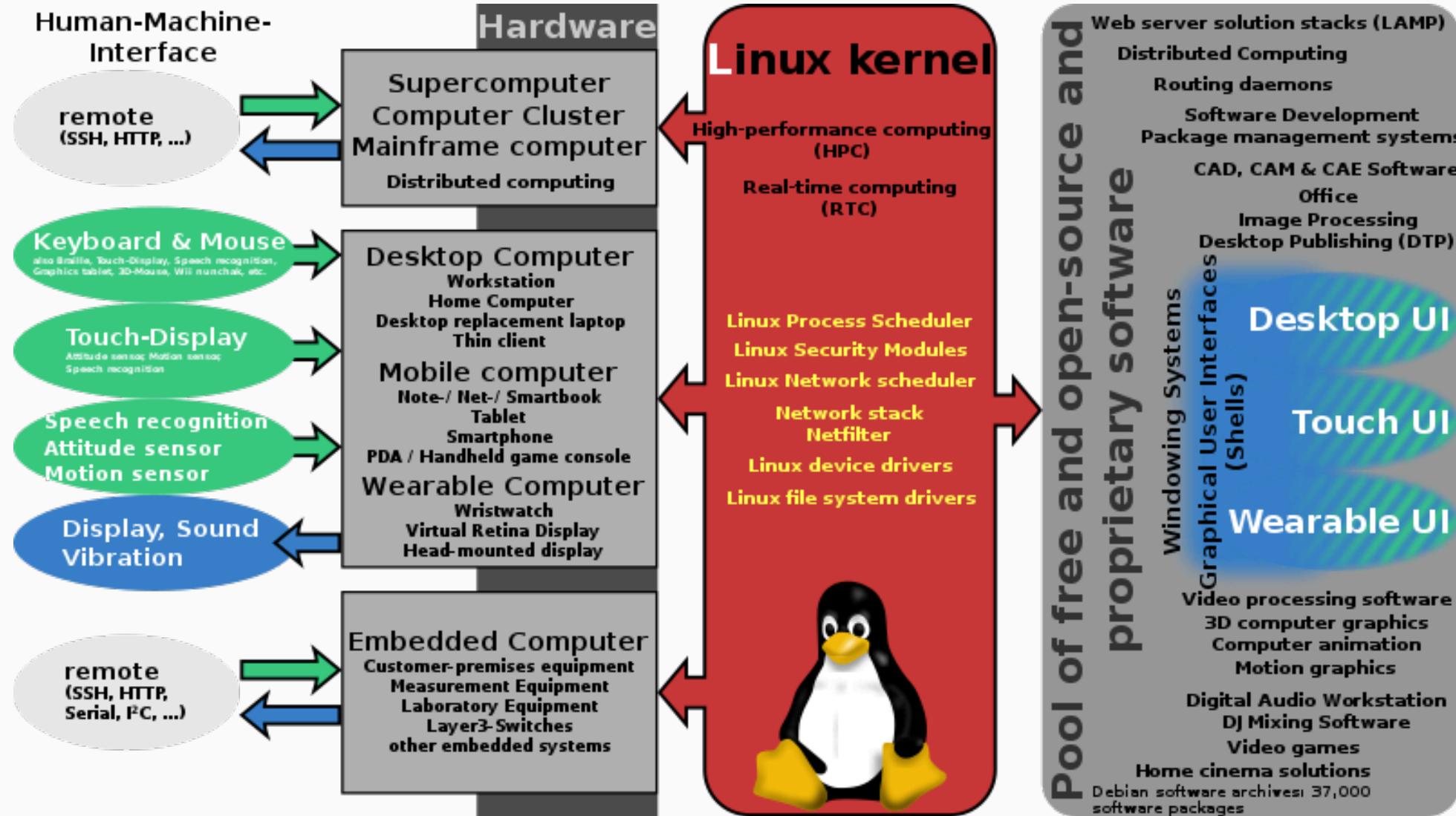


- Propriétaire
- Verrouillé
- Non modifiable

# 1.5 Unix / Windows



# 1.6 Fonctionnement d'une machine sous Linux





Une distribution Linux, appelée aussi distribution GNU/Linux lorsqu'elle contient les logiciels du projet GNU, est un ensemble cohérent de logiciels, la plupart étant des logiciels libres, assemblés autour du noyau Linux, et formant un système d'exploitation pleinement opérationnel. [Source Wikipédia](#)

### Liste des distributions GNU/Linux

**Distributions généralistes** : Debian, Ubuntu, Mageia, Fedora, Slackware, OpenSUSE, ArchLinux, Gentoo...

Les distributions orientées exclusivement vers l'**entreprise** avec un contrat de **support** annuel (à base de souscription par machine) par exemple **Red Hat Enterprise Linux**, **Ubuntu Long Term Support(LTS)** et **SUSE Linux Enterprise**.

### Référencement de toutes les distributions Unix/Linux : Distrowatch



### [Arborescence d'un système GNU/Linux](#)

Manuel Filesystem Hierarchy

man hier



## 1.8 Répertoires d'un système GNU/Linux

- **/** : Racine du système
- **/bin** (binaries) : Exécutables essentiels au système, utilisables par tous les utilisateurs (`ls`, `mkdir`, `touch`, `cp`, ...)
- **/boot** (bootstrap : initialisation) : fichiers permettant à Linux de démarrer (noyau,...)
- **/dev** (device) : Fichiers spéciaux des périphériques (disques durs, écrans, partitions, webcam,...)
- **/etc** (editing text config) : Fichiers de configuration au format textuel de plusieurs programmes et services du système (`passwd`, `fstab`, ...)
- **/home** : Répertoire personnel des utilisateurs
- **/lib** (librairies : bibliothèques) : Bibliothèques partagées essentielles et modules du noyau
- **/lib64** : idem `/lib` mais pour les 64bits (parfois, on trouvera `lib` et `lib32`. Dans ce cas, `lib` = 64bits et `lib32` = 32bits)
- **/media** : Contient les points de montages pour les médias amovibles
- **/mnt** (mount : montage) : Point de montage pour monter temporairement un système de fichiers
- **/opt** (optional) : Emplacement pour des applications installées hors gestionnaire de paquets (logiciels **optionnels**)



- **/proc** (process) : Répertoire virtuel ne prenant aucune place sur le disque, pour les informations système (noyau, processus)
- **/root** (racine) : Répertoire personnel du super utilisateur
- **/run** (runtime system : exécution système) : Informations relatives au système depuis son dernier démarrage (ex : utilisateurs actifs, services en cours d'exécution,...)
- **/sbin** (super binaries) : Programmes système essentiels utilisables par l'admin uniquement
- **/srv** (services) : N'est pas présent dans toutes les distributions. C'est un répertoire de données pour divers services (stockage des documents de comptes FTP, ou pages de sites web)
- **/sys** : Répertoire virtuel ne prenant aucune place sur le disque. Contient des informations entre le système et ses composants matériels
- **/tmp** (temporary) : Fichiers temporaires des applications
- **/usr** (Unix System Ressources) : Programmes installés (**/usr/bin**) avec leur librairies (**/usr/lib** ou **/usr/lib64**) tels que firefox, libreoffice, ... quelques programmes réservés à l'admin système (**/usr/sbin**) et les fichiers de code source (**/usr/src**).
  - On y retrouve **/usr/share** avec les éléments partagés indépendants de l'architecture (documentation, icônes, ...).
  - Dans **/usr/local** on pourra installer les programmes compilés manuellement sur le système.



- **/var** (variable) : Données variables (fichiers de log dans **/var/log**) mais parfois les bases de données (**/var/lib/mysql**) et les pages de site web (**/var/www/html**)



**Secure Shell (SSH)** est un protocole de **communication** sécurisé. Le protocole de connexion impose un échange de **clés de chiffrement** en début de connexion. Par la suite, tous les segments TCP sont authentifiés et chiffrés. Il devient donc impossible d'utiliser un analyseur de paquets (sniffer) pour voir ce que fait l'utilisateur.



- Installer **ssh** si pas présent sur le **serveur** :

**sudo apt install openssh-server**

- Sur le **serveur** vérifier le status de **ssh** :

**systemctl status ssh**

- Tester **ssh** en local :

**ssh localhost**

- Sur la machine hôte (**Windows** ou autre) générer un couple de **clefs publiques privées** :

**ssh-keygen -t ed25519 -C "mon@email.fr"**

- Copie de la **clé publique** sur le **serveur** :

**ssh-copy-id -i /.ssh/clef\_public.pub user@ip\_serveur**

- Connexion **ssh** :

**ssh user@ipserveur**



<https://www.virtualbox.org>

**Oracle VM VirtualBox** (anciennement **VirtualBox**) est un logiciel libre de **virtualisation** créé par la société Innotek (en) rachetée par Sun Microsystems et aujourd'hui publié par Oracle.



**debian**

<https://www.debian.org/releases/index.fr.html>



**Debian** (ou **Debian GNU/Linux**) est une distribution **Linux**, composée presque exclusivement de logiciels libres. Elle est aussi utilisée en tant que base de nombreuses autres distributions, comme **Ubuntu**.

**Debian** est disponible en trois versions. Ces trois versions possèdent des cycles de développement distincts (trois branches) :

- **stable** : c'est la version mise en avant par la communauté. Une version sort tous les deux ans. Utilisé sur les **serveurs en production**.
- **testing** : c'est la version en amont de la version stable. Elle offre des paquets extrêmement récents, tout en conservant une base de stabilité relative (rien de réellement bloquant).
- **unstable** : (surnommée **Sid**, pour « Still In Development ») Il s'agit de la version qui est en amont de la version **testing**, elle est en constante évolution, alimentée sans fin par des nouveaux paquets ou des mises à jour de paquets déjà existants (on parle de **rolling release**).



La liste des dépôts des logiciels se trouve dans **/etc/apt/source.list**  
ainsi que **/etc/apt/sources.list.d/fichier\_source.list**

- Met à jour la liste des logiciels disponibles **apt update**
- Mise à jour de la distribution **apt upgrade**
- Simuler une mise à jour **apt -simulate upgrade**



Installer des services,...

**tasksel**



Documentation sur la commande **apt** : <https://www.debian.org/doc/manuals/debian-faq/pkgtools.en.html#apt-get>

- Installer un logiciel avec ses dépendances **apt install logiciel**
- Dés-installer un logiciel avec ses dépendances **apt remove logiciel**
- Dés-installation complète d'un logiciel avec ses fichiers de configuration **apt remove - purge logiciel**
- Rechercher un logiciel **apt search logiciel**
- Information sur un logiciel installé **apt-cache policy logiciel**

Recherche de logiciel sur le site web : [Debian packages](#)



Le fichier **/etc/hostname** contient le nom de la machine

Le fichier **/etc/hosts** permet de lier une adresse ip à un nom hôte local.

Le fichier **/etc/network/interfaces** permet de mettre une ip statique à la place de **dhcp**



- `apt install cowsay`
- `apt install cmatrix`
- `apt install sl`
- `apt install toilet`
- `apt install figlet`
- `apt install lolcat`



- **man programme** : Lire le manuel d'utilisation d'une commande
- **whereis programme** : Permet de rechercher les fichiers exécutables, les sources et les pages de manuel d'une commande  
(**whereis ls** renvoie **ls: /usr/bin/ls /usr/share/man/man1/ls.1.gz**)
- **which programme** : Permet de savoir quel est le fichier exécuté  
(**which python** renvoie **/usr/bin/python**)
- **whatis programme** : Permet de savoir rapidement à quoi sert une commande  
(**whatis mkdir** renvoie **pwd (1) - print name of current/working directory**)



Une commande **shell** est une chaîne de caractères en minuscules qui peut être invoquée au travers d'une **invite de commande** ou d'un **script**. Des **options** et des **arguments** peuvent la compléter. Ceux-ci sont généralement appelés **paramètres** de la commande.



- **pwd** : Affiche le nom du répertoire courant
- **ls** : Lister les fichiers et dossiers présents dans un répertoire
- **cd** : Se déplacer dans un autre répertoire
- **df** : Indiquer l'espace occupé par les systèmes de fichiers
- **du** : Évaluer l'espace disque occupé par des fichiers



- **wc** : Compte le nombre de lignes, mots et caractères contenus dans un fichier
- **cat** : Concaténer des fichiers et les afficher sur la sortie standard
- **file** : Déterminer le type d'un fichier
- **stat** : Informations détaillées sur un fichier
- **free** : Connaître l'état de la mémoire du serveur
- **top ou htop** : Voir les processus en cours



- **cp** : Copier
- **mv** : Déplacer ou renommer
- **rm** : Suppression
- **mkdir** : Création de répertoire
- **rmdir** : Suppression de répertoire
- **ln** : Créer des liens physiques ou symboliques
- **find** : Rechercher des fichiers
- **grep** : Faire des recherches plein texte de chaînes de caractères



- **nl** : Affichage du contenu d'un fichier avec les numéros de lignes
- **head** : Affichage par défaut des 10 premières lignes d'un fichier
- **tail** : Affichage par défaut des 10 dernières lignes d'un fichier



**last** : Afficher les **dernières connexions** à votre machine

**lastlog2** : Afficher les **dernières connexions** de chaque **utilisateur** à votre machine

Liste des **ports** ouvert sur votre machine **apt install nmap**

**nmap localhost**



**su** : sans préciser un utilisateur ,

**su** permet d'exécuter des commandes en mode **root** mais sans charger les variables d'environnement à la différence de **su -**

**useradd** : créer un nouvel utilisateur , pas de création d'un dossier pour cet utilisateur dans **/home** , pas de création de mot de passe

**adduser** : à la différence de **useradd** un dossier dans **/home** est crée ainsi qu'un mot de passe

**deluser -remove-all-files** : supprimer un utilisateur avec tous ses fichiers et répertoires



## BASH

**Bash** (acronyme de Bourne-Again shell) est un interpréteur en ligne de commande de type script. C'est le [shell](#) Unix du projet [GNU](#).

Le terme **shell** désigne un logiciel fournissant une interface à l'utilisateur pour des composants d'un ensemble informatique .Il est plus généralement employé pour désigner un interpréteur de lignes de commandes pouvant accéder aux services et interagir avec le noyau d'un système d'exploitation. Dans le cas de **debian**, un **shell** interagit avec le noyau **Linux**.



- **~/.bashrc** : Fichier permettant la configuration lors du lancement du programme bash
- **~/.bash\_aliases** : Fichier contenant des **alias** (substitutions abrégées de commandes à taper dans la console)
- **~/.bash\_history** : Historique des commandes tapés en ligne de commande , voir ce fichier avec **history**
- **~/.profile** : Ce fichier n'est lu que si **~/.bash\_profile** ou **~/.bash\_login** n'existe pas. il permet de prendre en compte de nouveau chemin pour la variable d'environnement **PATH**.



Ce qu'une commande lit au clavier peut être dirigé vers un fichier. Le flux sur lequel une commande lit est nommé le fichier standard d'entrée (**stdin**).

Ce fichier standard d'entrée peut lui-aussi être dirigée à l'aide du méta-caractère **<**.

Ainsi, la commande **wc < un-fichier** fait comme si le contenu du fichier **un-fichier** était tapé sur le clavier alors que la commande **wc** (sans argument) a été invoquée.

Dans le cas de la commande **wc**, faire **wc un-fichier** ou **wc < un-fichier** est totalement équivalent.

**cat mon-fichier** et **cat < mon-fichier** auront exactement le même comportement.

On ne peut diriger l'entrée standard (sortie standard) que de commandes qui y lisent (écrivent) des données.

Par exemple cela n'a pas de sens de diriger l'entrée de **ls** puisque **ls** ne lit jamais de données sur son entrée standard.

On peut diriger la sortie standard d'une commande vers l'entrée standard d'une autre **ls | wc**

Cette opération (connecter la sortie d'une commande à l'entrée d'une autre) se nomme un **pipeline**.

Exécutions **simultanées** de commandes **mkdir dossierA & touch fichierB.txt & ... & ...**

Exécutions **successives** de commandes **ls ; df ; ... ; ...**



Enchaînements **conditionnels** de commandes **ls && df && ... && ...**



<https://doc.ubuntu-fr.org/permissions>

[File-system permissions : https://en.wikipedia.org/wiki/File-system\\_permissions](https://en.wikipedia.org/wiki/File-system_permissions)

Chaque fichier du système est associé à des droits d'accès.

Résultat de la commande **ls** :

```
-rwx-rw-r-- 1 marc marc 1.2K Dec 1 11:37 fichier.txt
```

Ce fichier a les droits suivants :

- pour le premier groupe de droit de **rwx** , il s'agit des droits du **propriétaire** , dans ce cas le propriétaire à les droits :
  - de lire **r (read)**
  - d'écrire **w (write)**
  - d'executer **x (execute)**
- pour le deuxième groupe de droit , il s'agit du **groupe** dans lequel se trouve le **propriétaire**
- pour le troisième groupe de droit , il s'agit du **reste du monde**

On gère les droits sur les fichiers avec la commande **chmod**.

## **2. Installation d'un serveur web local**

---



## 2.2 Installation de php

<https://www.php.net>

**PHP** (officiellement, ce sigle est un acronyme récursif pour **PHP Hypertext Preprocessor**) est un langage de scripts généraliste et Open Source, spécialement conçu pour le développement d'applications web. Il peut être intégré facilement au **HTML**.

**PHP** est principalement conçu pour servir de **langage de script** coté **serveur**, il peut :

- collecter des données de formulaire
- générer du contenu dynamique
- gérer des cookies



## 2.2 Installation de php

Il y a deux domaines différents où **PHP** peut s'illustrer :

- Langage de **script** côté **serveur**. C'est l'utilisation la plus traditionnelle, et aussi le principal objet de **PHP**. Trois composants sont nécessaires pour l'exploiter :
  - un analyseur **PHP** (CGI ou module serveur)
  - un **serveur web**
  - un navigateur web
- Langage de programmation en **ligne de commande**. Un **script PHP** peut être **exécuté en ligne de commande**, sans l'aide du **serveur** web et d'un navigateur. Il suffit de disposer de l'exécutable **PHP**. Cette utilisation est idéale pour les scripts qui sont exécutés régulièrement avec un **cron** sous **Unix** ou **Linux** ou un **gestionnaire de tâches** (sous **Windows**).

**FPM (FastCGI Process Manager)** est une version améliorée de **FastCGI** pour **PHP**, conçue pour gérer efficacement de lourdes charges de trafic. Il offre une gestion fine des **processus**, une configuration flexible et des outils avancés de **monitoring** et de **journalisation**, ce qui le rend particulièrement adapté aux environnements web hautes performances.



## 2.2 Installation de php

On installe **php** :

```
apt install php php-cli php-fpm php-mysql
```

On vérifie la version de **php** qu'on vient d'installer :

```
php -v
```

On vérifie que le service **php-fpm** est bien lancé:

```
systemctl status php8.xxx-fpm
```



## 2.3 Installation du serveur web nginx

<https://nginx.org/>

**Nginx** est un logiciel libre de serveur **Web** (ou **HTTP**) ainsi qu'un **proxy inverse** écrit par Igor Sysoev, dont le développement a débuté en 2002.



## 2.3 Installation du serveur web nginx

Installation : `apt-get install nginx`

On démarre le serveur **nginx** `systemctl start nginx`

On configure **nginx** pour prendre en charge les fichiers **php** , éditer le fichier :  
`nano /etc/nginx/sites-available/default`

Changer la ligne `try_files $uri $uri/ =404;` par `try_files $uri $uri/ /index.php?$args;`

Ajouter :

```
location ~ \.php$ {  
    include snippets/fastcgi-php.conf;  
    fastcgi_pass unix:/run/php/php8.x-fpm.sock;  
}
```

On test le fichier de configuration à savoir si il n'y a pas d'erreur `nginx -t`

On relance le serveur **nginx** `systemctl restart nginx`

Pour ne pas afficher les informations du serveur , éditer



<https://fr.wikipedia.org/wiki/LAMP>



<https://httpd.apache.org>

Le logiciel libre **Apache HTTP Server (Apache)** est un **serveur HTTP** créé et maintenu au sein de la fondation **Apache**.



## 2.5 Installation du serveur web Apache

Installation : `apt install apache2`

On crée le dossier du site web :

```
mkdir /var/www/www.monsuperbesiteweb.fr
```

On crée une page **index.html** dans ce dossier

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Mon site</title>
  </head>
  <body>
    <p>Bienvenue sur mon site.</p>
  </body>
</html>
```



## 2.5 Installation du serveur web Apache

Chaque site web qu'on héberge aura un hôte virtuel [VirtualHost](#)  
avec son fichier de configuration `www.monsuperbesiteweb.fr.conf`

```
<VirtualHost 192.168.0.34:80>
    ServerName www.monsuperbesiteweb.fr.conf
    DocumentRoot /var/www/www.monsuperbesiteweb.fr
    DirectoryIndex index.html
    <Directory /var/www/www.monsuperbesiteweb.fr>
        Options -Indexes
    </Directory>
    ErrorDocument 404 /404.html
    ErrorLog ${APACHE_LOG_DIR}/error_www.monsuperbesiteweb.fr
    CustomLog ${APACHE_LOG_DIR}/access_www.monsuperbesiteweb.fr combined
</VirtualHost>
```

Activer le site `a2ensite www.monsuperbesiteweb.fr.conf`

Redémarrez le serveur `systemctl restart apache2`

Ajouter votre site au fichier `/etc/hosts`, dans Windows `C:/windows/system32/drivers/etc/hosts`

`192.168.0.34 www.monsuperbesiteweb.fr`



## 2.6 Installation d'un serveur de base de donnée mariadb/mysql

<https://mariadb.com>

MariaDB Server Documentation : <https://mariadb.com/kb/en/documentation>

**MariaDB** est un système de gestion de base de données **relationnelle** édité sous licence **GPL**.  
Il s'agit d'un dérivé (**fork**) communautaire de **MySQL**.



## 2.6 Installation d'un serveur de base de donnée mariadb/mysql

Installation : `apt install mariadb-server`

On vérifie si mariadb est bien lancé : `systemctl status mariadb`

Sinon le démarrer :

`service mariadb start` ou `systemctl start mariadb`

On vérifie la version de **mariadb** : `mariadb --version`

Sécuriser **mariadb** via le script : `mariadb-secure-installation`

Se connecter à **mariadb** : `mariadb -u root -p`

Créer une base de données `CREATE database testdb;`

Créer un utilisateur avec tous les privilèges pour cette base de données

`CREATE USER 'testdbuser'@'localhost' IDENTIFIED BY 'UN_MOT_DE_PASSE';`

`GRANT ALL PRIVILEGES ON testdb.* TO 'testdbuser'@'localhost';`

`FLUSH PRIVILEGES;`

Sortir de **mariadb** `exit;`

Sélectionnez une base de données `USE DataBaseName;`



Lister toutes les bases de données `SHOW DATABASES;`

Lister toutes les tables `SHOW TABLES;`

Voir le **schéma** d'une table `DESC DataBaseName.Table;`

Lister tous les users dans **mariadb** `SELECT user FROM mysql.user;`



## 2.7 Tester des pages html,php

Donner des droits à l'utilisateur pour écrire dans le répertoire où se trouve les fichiers du serveur `chown -R $USER:$USER /var/www/html`

Créer une page de test `php echo "<?php phpinfo(); ?>" > /var/www/html/info.php`

Tester la connexion à la base de données : `testdb.php`

```
<?php

try{
    $dbconn = new pdo( 'mysql:host=127.0.0.1;dbname=testdb',
                      'testdbuser',
                      'testdbuserModeDePasse',
                      [pdo::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]);
    die(json_encode(array('connexion db' => true)));
}
catch(PDOexception $e){
    die(json_encode(['connexion db' => false, 'message' => 'unable to connect']));
}
?>
```



## 2.8 Compression

### zip

```
zip -r dossier.zip dossier/
```

### unzip

```
unzip dossier.zip
```

### tar

```
tar -cvf backup.tar /home/user
```

### tar.gz

```
tar -cvzfp backup.tar.gz /home/user
```

Backup d'une machine entière avec exclusion de dossier :

```
tar -cvzfp backup.tgz --exclude=/proc --exclude=/lost+found --exclude=/mnt --exclude=/media --exclude=backup.tgz  
--exclude=/sys /
```

### Exclude des fichiers lors de la compression tar.gz

```
tar --exclude file.txt --exclude file.sh -cvzf backup.tar.gz
```

### Extraire tar.gz

```
tar -xvzf backup.tar.gz
```



### Lister le contenu d'un tar.gz

```
tar -ztvf backup.tar.gz
```

### Découper une archive tar.gz en plus petit morceaux

```
tar cvf - /home/user | split --bytes=200MB - backup.tar
```

### Tester si un fichier tar.gz n'est pas corrompu

```
gunzip -t backup.tar.gz
```

### Tester l'intégrité d'un fichier tar ou tar.gz

```
gunzip -c backup.tar.gz | tar t > /dev/null
```

```
tar -tvWF backup.tar
```



## 2.9 Sauvegarder une base de donnée mariadb avec /mariadb-dump

```
mariadb-dump --user=user_name --password=user_password database_name > database_name.sql
```

Avec compression :

```
mariadb-dump --user=user_name --password=user_password database_name | gzip > database_name.sql.gz
```

Personnaliser son fichier de sauvegarde avec la date (Jour Mois Année Heure Minute Seconde) :

```
mariadb-dump --user=user_name --password=user_password database_name > database_name-$(date ++%d%m%Y%H%M%S).sql
```

Sauvegarder plusieurs bases de données :

```
mariadb-dump --user=user_name --password=user_password --databases database_name_1 database_name_2 database_name_3 > database_names.sql
```

Sauvegarder toutes les bases de données :

```
mariadb-dump --user=user_name --password=user_password --all-databases > database_name_complete.sql
```

Sauvegarder toutes les bases de données dans des fichiers séparées à l'aide d'une boucle for :

```
for database in $(mysql/mariadb --user=user_name --password=user_password -e 'show databases' -s --skip-column-names); do
```



## 2.9 Sauvegarder une base de donnée mariadb avec /mariadb-dumb

```
mariadb-dump $database > "$database.sql";
done
```

Restaurer une sauvegarde de base de donnée depuis un fichier sql :

```
mysql/mariadb --user=user_name --password=user_password < database_name.sql
```

Restaurer une sauvegarde complète de base de donnée depuis un fichier sql :

```
mysql/mariadb --user=user_name --password=user_password --one-database database_names <
database_name_complete.sql
```

Supprimer les sauvegardes datant de plus de 30 jours à l'aide de find :

```
find /mes_sauvegardes -type f -name "*.sql" -mtime +30 -delete
```



## 2.10 Monitorer sa machine

**Liste** des répertoires qui prennent le plus de place :

```
du -ch --max-depth=2 / 2>/dev/null | sort -rh | head -15
```

Liste les **derniers** fichiers **crées** ou **modifiés** dans les dernières 5 minutes :

```
find / -mmin 5 > liste.txt
```

Voir les **ports** ouverts sur son serveur :

```
ss -tulpn
```

Lister tous les **services actifs** :

```
systemctl -t service
```

Les options de **systemctl** pour un **service** comme **nginx** :

- demarrer : `systemctl start nginx`
- arreter : `systemctl stop nginx`
- redemarrer : `systemctl restart nginx`
- status : `systemctl status nginx`
- activer au demarrage de la machine : `systemctl enable nginx`
- desactiver au demarrage de la machine : `systemctl disable nginx`



## 2.10 Monitorer sa machine

Information sur la machine : `hostnamectl`

**journalctl** permet d'**afficher** les **logs systemes (syslog)**.

Afficher les **logs** de sa machine en **live** : `journalctl -f`

Afficher les **logs** de sa machine en **live** aujourd'hui : `journalctl --since=today -f`

Afficher les **logs** de sa machine en **live** à une date précise :

`journalctl --since="2025-12-02 09:00:00" -f`

Afficher les **logs** de sa machine en **live** il y a une heure : `journalctl --since="1 hour ago" -f`

Afficher les logs d'un **service nginx** en live : `journalctl -u nginx -f`

Afficher les logs de **plusieurs service nginx** en live : `journalctl -u nginx -u mysql -f`

Afficher les **logs** de sa machine en **live** avec les **erreurs** : `journalctl -f -p err`

Afficher les 10 **denieres erreurs** dans les **logs** : `journalctl -p err -n 10 --no-pager`

Afficher les **erreurs** avec une **couleur** en recherchant avec **grep** :

`journalctl | grep --color "error"`



## 2.10 Monitorer sa machine

Combiner l'affichage des **erreurs** avec une **couleur** en recherchant avec **grep** :

```
journalctl -f | grep --color -E "error|warning|critical"
```

Coloriser les **logs** avec **ccze** :

```
sudo apt install ccze
```

```
journalctl -f | ccze -A
```

Exporter les **logs** au format **json** : `journalctl -o json`

Afficher les **logs** sous differentes formes :

- Que le message : `journalctl -o cat`
- Tous les champs : `journalctl -o verbose`
- Format cours (par defaut) : `journalctl -o short`
- Format cours à la microseconde : `journalctl -o short-precise`

Lister les derniers boot : `journalctl --list-boots`

```
journalctl -u ssh.service --since "1 week ago" | grep -E "(shutdown|reboot)"
```

```
grep -E "(shutdown|reboot)" /var/log/auth.log
```

### **3. Vagrant**

---



### 3.1 Création de VMs

**Vagrant** est un logiciel anciennement libre et open-source pour la **création** et la **configuration** des environnements de développement virtuels.

```
vagrant init [nom-de-boîte]
```

Initialise un nouvel environnement Vagrant en créant un Vagrantfile

```
vagrant up
```

Démarre et provisionne l'environnement Vagrant

```
vagrant halt
```

Arrête l'environnement Vagrant en cours d'exécution

```
vagrant reload
```

Redémarre l'environnement Vagrant

```
vagrant destroy
```

Détruit l'environnement Vagrant

```
vagrant ssh
```

Se connecte à l'environnement Vagrant via SSH

```
vagrant status
```

Affiche le statut de l'environnement Vagrant



## 3.2 Configuration

### Vagrantfile

Le fichier de configuration pour un environnement Vagrant

configvmbox = "nom-de-boîte"

Spécifie la boîte de base pour l'environnement Vagrant

configvmnetwork "private\_network", ip: "1921683310"

Configure un réseau privé pour l'environnement Vagrant

configvmsynced\_folder "src/", "/srv/website"

Configure un dossier synchronisé entre l'hôte et l'environnement Vagrant

configvmprovider "virtualbox" do |vb|

  vbmemory = "1024"

  vbcpus = 2

end

Configure les paramètres spécifiques au fournisseur pour l'environnement Vagrant



### 3.3 Dépannage

```
vagrant global-status
```

Liste tous les environnements Vagrant gérés par l'utilisateur actuel

```
vagrant global-status --prune
```

Supprime les entrées du statut global qui ne sont plus valides

```
vagrant box list
```

Liste toutes les boîtes Vagrant disponibles

```
vagrant box remove [nom-de-boîte]
```

Supprime une boîte Vagrant

## **4. Parefeu (Firewall)**

---



## 4.1 UFW (Uncomplicated Firewall)

<https://help.ubuntu.com/community/UFW>

Le noyau **Linux** fournit un système de filtrage de paquets appelé **netfilter**, et l'interface traditionnelle pour manipuler **netfilter** est la suite de commandes **iptables**. **iptables** offre une solution de **pare-feu** complète, à la fois hautement configurable et très flexible.

- Activer le **pare-feu** : `sudo ufw enable`
- Désactiver le **pare-feu** : `sudo ufw disable`
- Afficher le **statut** (actif/inactif) : `sudo ufw status`
- Définir la politique par défaut pour bloquer tout trafic entrant (**Recommandé**) :
  - `sudo ufw default deny incoming`
  - `sudo ufw default allow outgoing`
- Ouvrir un **port** spécifique : `sudo ufw allow 80` (**HTTP**)
- Ouvre un port pour un protocole spécifique : `sudo ufw allow 443/tcp` (**HTTPS**)
- Ouvre le port d'un service : `sudo ufw allow ssh`
- Bloque le trafic sur un port spécifique : `sudo ufw deny 25` (**SMTP**)
- **ufw** a la possibilité de refuser les connexions à partir d'une adresse IP qui a tenté d'initier 6 connexions ou plus au cours des 30 dernières secondes : `sudo ufw limit ssh`

## **5. Securiser son serveur**

---



**Automatiser** l'installation des **mises à jour** de **sécurité** :

```
sudo apt-get install unattended-upgrades apt-listchanges
```

Valider par **yes** l'installation **automatique** des **mises à jour** :

```
sudo dpkg-reconfigure --priority=low unattended-upgrades
```



## 5.2 Simulation de charge du serveur

Installer **siege** sur une machine **hôte** :

```
sudo apt-get install siege
```

Simuler **200 utilisateurs virtuels** pendant **1 minute** vers le site web **cible** :

```
siege -c 200 -t 1M http://votre-serveur-nginx.com
```

## **6. Ressources**

---



- SecNumAcadémie, la formation en ligne sur la sécurité informatique gratuite et ouverte à tous : <https://secnumacademie.gouv.fr>
- Les bonnes pratiques de sécurité informatique : <https://www.ssi.gouv.fr/administration/bonnes-pratiques>



Les références et textes de ce support viennent de différentes sources :

- <https://www.wikipedia.fr>
- <https://ww.php.net>
- <http://www.github.fr>

Tous les textes de ce support sous licence **CC BY-NC-ND 4.0** - <https://creativecommons.org/licenses/by-nc-nd/4.0/deed.fr>.