

Rappel sur l'OS LINUX

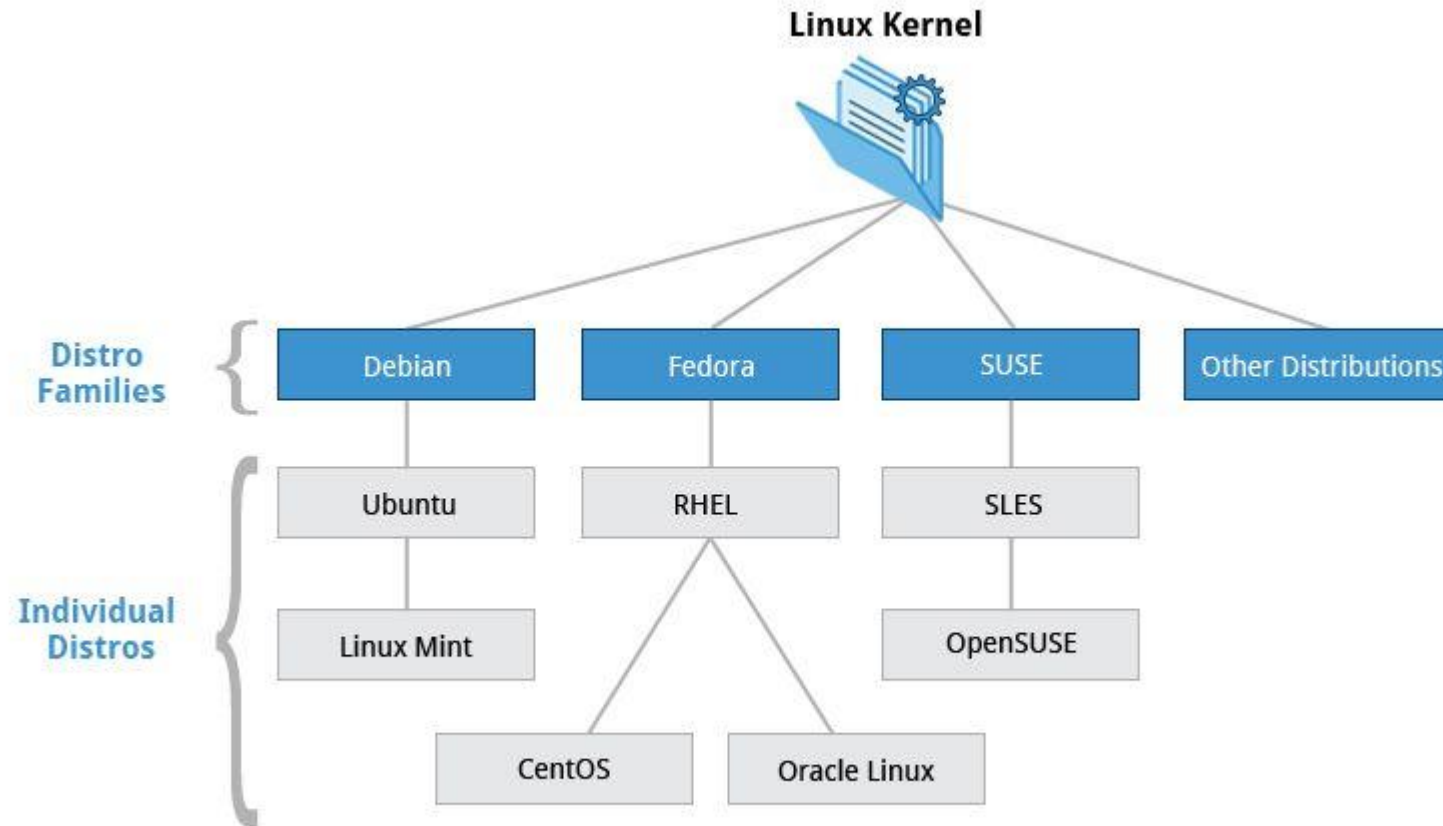
Plan

- ▶ Le système d'exploitation Linux
 - ▶ Les authentifications et l'autorisations
 - ▶ Les systèmes de fichiers
 - ▶ Le pare feu

Informations générales sur Linux

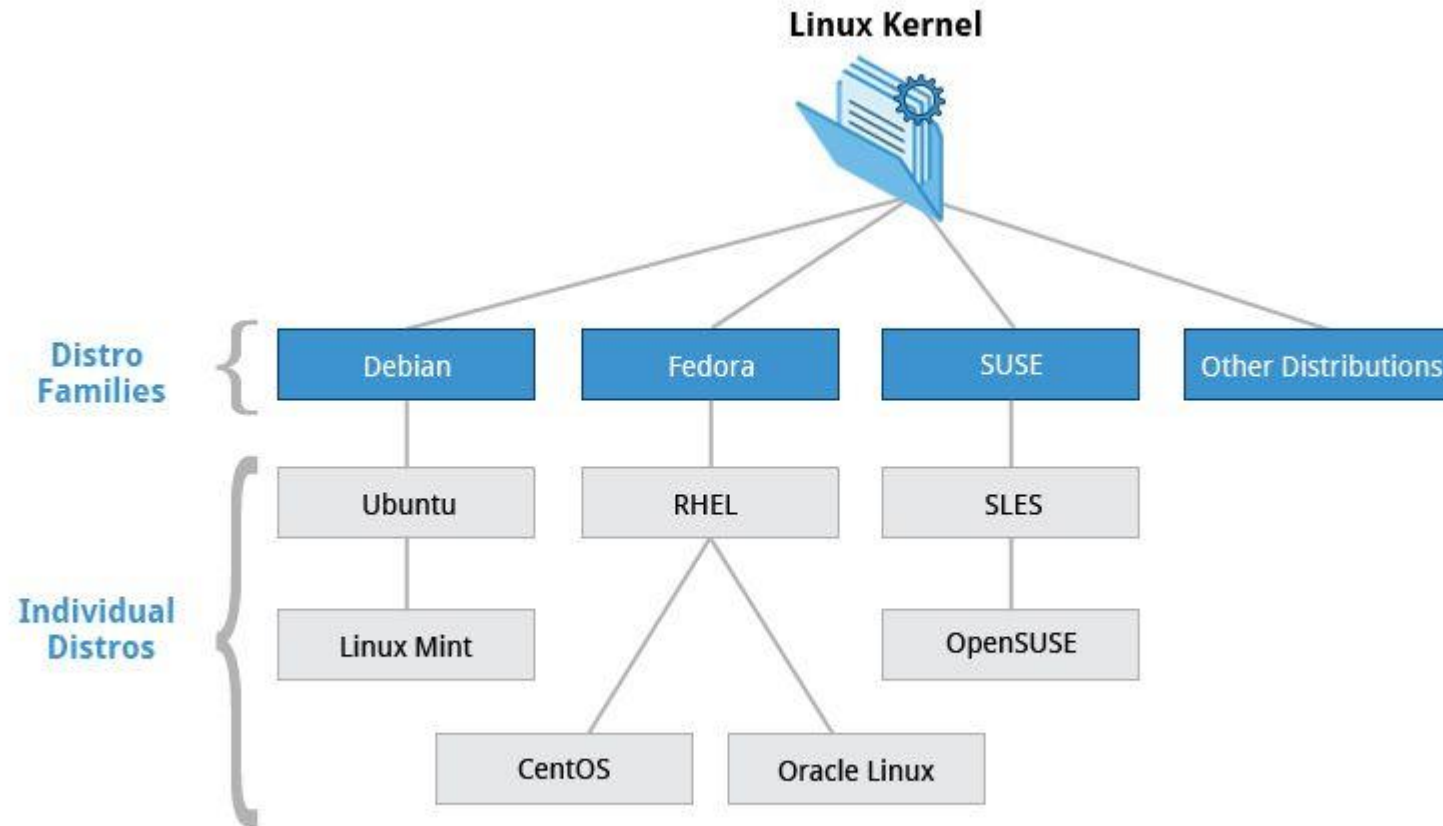
Les distributions linux

➤ Il y a 42 distributions linux



Les distributions linux

➤ Il y a 42 distributions linux



Linux informations générales

- Linux adopte une approche totalement différente en comparant avec Windows, Linux n'a pas de base de registre comme Windows
- Linux se base entièrement sur des fichiers
- Linux utilise le système de fichiers ext4 le successeur du système de fichiers ext3, principalement destiné aux systèmes basés sur GNU/Linux
- Linux utilise le système de fichiers ext4 le successeur du système de fichiers ext3, principalement destiné aux systèmes basés sur GNU/Linux



Linux les étapes de démarrage

1

BIOS

Le processus de démarrage commence par le système d'entrée/sortie de base (BIOS) ou l'interface UEFI (Unified Extensible Firmware Interface)

2

GRUB2

L'étape suivante est l'exécution d'un chargeur de démarrage. Les chargeurs de démarrage courants incluent GRUB (Grand Unified Bootloader)

3

Kernel

Le noyau Linux est le cœur du système d'exploitation. Il fournit des services essentiels tels que la gestion des processus, la gestion de la mémoire, les pilotes de périphériques et les appels système. Le noyau est chargé en mémoire par le chargeur de démarrage

4

Systemd

Systemd effectue l'initialisation du système et gère les services système.

5

Espace utilisateur

L'espace utilisateur comprend les bibliothèques système, les commandes et d'autres services qui ne font pas partie du noyau

6

Niveaux D'exécutions

Chargement des cibles

7

Connexion

Service de connexion d'utilisateur

8

Serveur X optionnel

Composant graphique Gnome ou KDE ou autre



Le système de fichiers

Les types de fichiers sous linux

Fichiers généraux : On les appelle également fichiers ordinaires. Il peut s'agir d'une image, d'une vidéo, d'un script ou d'un simple fichier texte. Ces types de fichiers peuvent être au format ASCII ou binaire. Il s'agit du fichier le plus couramment utilisé sous le système Linux.

Fichiers de répertoire : Ces types de fichiers constituent un entrepôt pour d'autres types de fichiers. Il peut s'agir d'un fichier répertoire dans un répertoire sous-répertoire.

Fichiers système: Ces types de fichiers assurent le bon fonctionnement du système

Fichiers de périphérique : Dans un système d'exploitation de type Windows, les périphériques tels que les CD-ROM et les disques durs sont représentés par des lettres de lecteur telles que F : G : H, tandis que dans le système Linux, les périphériques sont représentés sous forme de fichiers. Comme par exemple, /dev/sda1, /dev/sda2, etc



Les fichiers généraux

- La création d'un fichier standard se fait à l'aide d'un éditeur de choix VIM, ou NANO ou un éditeur graphique comme GEDIT ou EMACS



Les fichiers de répertoire

- La création d'un fichier répertoire c'est un répertoire qui sont créés selon des règles précises
 - Les noms de dossiers sous Linux sont sensibles à la casse . Par conséquent, « Folder1 » et « folder1 » représenteraient des fichiers différents
 - Les noms de fichiers commençant par un point (".folder1") sont masqués
 - Il est déconseillé d'utiliser des noms de dossier comprenant d'espaces
 - Limiter l'usage des caractères spéciaux aux tirets et traits



Les fichiers système

- Sur un système Linux, vous pouvez énumérer les types de systèmes de fichiers actuellement disponibles à partir de la ligne de commande **cat /proc/filesystems**

```
$ cat /proc/filesystems
nodev    sysfs
nodev    tmpfs
nodev    bdev
nodev    proc
nodev    cgroup
nodev    cgroup2
nodev    cpuset
nodev    devtmpfs
nodev    configfs
```

- Pour lister les fichiers système **findmnt**

```
ubuntu@ml:~$ findmnt
TARGET                                SOURCE                                FSTYPE
/                                     /dev/mapper/ubuntu--vg-ubuntu01    ext4
-/sys                                 sysfs                                sysfs
-/sys/kernel/security                 securityfs                            securityfs
-/sys/fs/cgroup                       cgroup2                              cgroup2
-/sys/fs/pstore                       pstore                              pstore
-/sys/fs/bpf                          bpf                                  bpf
-/sys/kernel/debug                    debugfs                              debugfs
-/sys/kernel/tracing                  tracefs                              tracefs
-/sys/fs/fuse/connections              fusectl                              fusectl
```



Les fichiers de périphériques

- Il est possible de lister les fichiers montés à l'aide de la commande **findmnt -t ext4**

```
$ findmnt -t ext4
TARGET SOURCE FSTYPE OPTIONS
/ /dev/mapper/ubuntu--vg-ubuntu--lv ext4 rw,relatime
/boot /dev/sda2 ext4 rw,relatime
```

- La commande **lsblk** permet de montrer les volumes disponibles

```
$ lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
loop0 7:0 0 63,3M 1 loop /snap/core20/1822
loop1 7:1 0 111,9M 1 loop /snap/lxd/24322
loop2 7:2 0 49,8M 1 loop /snap/snapd/18357
sda 8:0 0 10G 0 disk
├─sda1 8:1 0 1M 0 part
├─sda2 8:2 0 1,8G 0 part /boot
├─sda3 8:3 0 8,2G 0 part
└─ubuntu--vg-ubuntu--lv 253:0 0 8,2G 0 lvm /
sr0 11:0 1 1024M 0 rom
```

- Pour monter un CDROM **mkdir /mnt/cdrom**
- Pour monter une clé usb **mkdir -p /media/usb**



La structure du système de fichiers sous linux

- Linux dispose d'un système de fichiers EXT et non pas NTFS comme Windows
- Les principaux dossiers sous linux sont
 - **/bin**: Ce répertoire contient tous les fichiers programmes binaires ou exécutables pour démarrer le système
 - **/sbin**: Similaire à **bin** ce répertoire contient les commandes nécessaires au démarrage du système mais qui ne nécessitent pas un privilège élevés
 - **/usr/bin**: Ce répertoire contient tous les fichiers programmes binaires ne sont pas nécessaires aux démarrage
 - **/usr/sbin**: Ce répertoire contient tous les fichiers programmes binaires ne sont pas nécessaires aux démarrage et qui demandent un privilège élevé



La structure du système de fichiers sous linux

/bin(et /sbin) étaient destinés aux programmes qui devaient se trouver sur une petite /partition avant

De nos jours, ils servent principalement **comme emplacement standard pour des programmes clés tels /bin/sh** , bien que l'intention d'origine puisse toujours être pertinente, par exemple pour des installations sur de petits appareils embarqués.

/sbin, à la différence de /bin, est destiné aux programmes de gestion système (qui ne sont normalement pas utilisés par les utilisateurs ordinaires) nécessaires avant le montage **/usr**. **/usr/bin** est destiné aux programmes utilisateur normaux gérés par la distribution.

/usr/local/bin est destiné aux programmes utilisateur normaux non gérés par le gestionnaire de packages de distribution, par exemple les packages compilés localement. Vous ne devez pas les installer dans **/usr/bin** car les futures mises à niveau de la distribution peuvent les modifier ou les supprimer sans avertissement.

/usr/local/sbin, comme vous pouvez probablement le deviner, c'est pour exécuter les programmes locaux



La structure du système de fichiers sous linux

Il y a trois sous dossiers sous /usr qui sont d'importance considérable

/usr/bin: Contient les exécutables nécessaires pour le lancement du système

/usr/sbin: les binaires qui s'exécutent au lancement mais avec les privilèges de **super utilisateur**

/usr/local: est un endroit pour installer les fichiers créés par l'administrateur, exemple un fichier

Makefile

/var: Le contenu des fichiers qui ont tendance à grossir comme les fichiers de journaux se trouvent dans ce répertoire

- **/var/log :** fichiers journaux système générés par le système d'exploitation et d'autres applications.
- **/var/lib :** contient la base de données et les fichiers de packages.
- **/var/mail :** Contient les e-mails.
- **/var/tmp :** Contient les fichiers temporaires nécessaires au redémarrage



La structure du système de fichiers sous linux

- Linux dispose d'un système de fichiers EXT et non pas NTFS comme Windows
- Les principaux dossiers sous linux sont
 - **/:** C'est le répertoire racine, **.** C'est le point de départ du FHS
 - **/etc:** Les fichiers de configuration principaux sont stockés dans le répertoire **/etc**, si quelqu'un à l'accès à ce dossier, il pourra explorer les paramètres des applications comme les ports, les chemins etc...
 - **/home:** C'est le répertoire où l'utilisateur courant peut stocker ses informations personnelles y compris les certificats **.ssh**
 - **/opt:** Ce répertoire est utilisé pour installer les logiciels d'application de fournisseurs tiers qui ne sont pas disponibles dans la distribution Linux (exemple: Google Earth, Tomcat)
 - **/tmp:** Pour les fichiers temporaires, il s'efface lors du prochain redémarrage
 - **/usr:** Ce répertoire contient les bibliothèques, le code source, les binaires et la documentation nécessaires aux programmes de second niveau



La structure du système de fichiers sous linux

- **/var:** Les fichiers de journalisation
- **/var/lib:** Les librairies
- **/mnt:** Il contient des répertoires de montage temporaires pour monter le système de fichiers
- **/sys:** Il s'agit d'un système de fichiers virtuel que les distributions Linux modernes peuvent stocker et qui permet de modifier les appareils connectés au système
- **/srv:** Il contient des fichiers spécifiques au serveur et liés au serveur



Les commandes pour gérer les dossiers

- **cd**: pour changer de dossier
- **cd -**: pour retourner au répertoire précédemment accédé
- **cd ~**: pour accéder au home
- **cd ..** **Cd ../..** : pour retourner vers le répertoire parent
- **mkdir**: pour créer un dossier
- **mkdir -p**: pour créer des dossiers imbriqués
- **rmdir**: pour supprimer un dossier vide
- **rm -rf**: pour supprimer un dossier non vide content des dossiers et des fichiers
- **cp**: pour copier un dossier
- **cp -r**: pour copier un dossier d'une manière récursive
- **mv**: pour déplacer un dossier
- **pwd**: pour afficher ou exprimer le chemin du répertoire courant
- **!!**: afficher la commande précédente
- **HISTTIMEFORMAT="%Y-%m-%d %T "** | **history**: ajouter le temps à history à ajouter dans le `bachrc`
- **truncate -s 0 <fichier>** : vider le fichier



Les commandes pour gérer les dossiers

- **touch**: pour créer un fichier
- **cat**: pour afficher le contenu d'un fichier
- **head**: pour afficher les quelques premières lignes du fichier
- **tail**: pour afficher les quelques dernières lignes du fichier
- **tail -f** : pour afficher les quelques dernières lignes du fichier en mode dynamique
- **rm -r**: pour supprimer un dossier non vide content des dossiers
- **rm -rf**: pour supprimer un dossier non vide content des dossiers et des fichiers
- **cp**: pour copier un fichier
- **mv**: pour déplacer un fichier
- **diff**: pour comparer deux fichiers en affichant le contenu
- **cmp**: pour comparer deux fichiers en indiquant les différences au niveau des lignes
- **zip**: pour zipper des fichiers

```
ubuntu@m1:~$ zip files file1 file2 file3
  adding: file1 (stored 0%)
  adding: file2 (stored 0%)
  adding: file3 (stored 0%)
ubuntu@m1:~$ ls
file1 file2 file3 files.zip script.py
```

- **unzip**: pour dé zipper des fichiers



Les commandes pour gérer les dossiers

- **Tar cvf**: pour compresser des fichiers en format tar

```
ubuntu@m1:~$ ls
file1 file2 file3 files files.zip script.py
ubuntu@m1:~$ tar cvf {file1,file2,file3} files/
file2
file3
files/          ubuntu@m1:~$ tar cvf {file1,file2,file3} files/
files/file3
files/file1
files/files.tar
files/file2
ubuntu@m1:~$ ls
file1 file2 file3 files files.zip script.py
ubuntu@m1:~$ cd files
ubuntu@m1:~/files$ ls
file1 file2 file3 files.tar
```

- **tar -xvf** : pour décompresser des fichiers
- **wget**: pour télécharger des fichiers depuis internet
- **whereis**: pour trouver l'emplacement d'un fichier



La commande ls pour lister les fichiers et les dossiers

- La commande **ls** est l'une des commandes les plus basiques de Linux. Il est conçu pour répertorier les noms et les caractéristiques des fichiers et des répertoires.
- Pour découvrir **ls** tapez **man ls**
- **ls -l** : lister les fichiers et les dossiers avec les droits
- **ls -a** : afficher les fichiers et les dossiers y compris les fichiers et dossiers cachés
- **ls -s** : afficher les fichiers avec les tailles allouées

Les droits	Nombre de liens physiques	propriétaire	groupe	dernière modification	Nom du fichier
Lien symbolique					
Dossier					
<pre>ubuntu@m1:/\$ ls -l total 1371220 lrwxrwxrwx 1 root root 7 févr. 17 2023 bin -> usr/bin drwxr-xr-x 4 root root 4096 oct. 27 13:57 boot drwxr-xr-x 20 root root 4100 oct. 28 10:05 dev drwxr-xr-x 2 root root 4096 oct. 28 10:32 directory1 drwxr-xr-x 2 root root 4096 oct. 28 10:32 directory2 drwxr-xr-x 2 root root 4096 oct. 28 10:32 directory3 drwxr-xr-x 2 root root 4096 oct. 28 10:32 directory4</pre>					



La commande **ls** pour lister les fichiers et les dossiers

- La commande **ls** est l'une des commandes les plus basiques de Linux. Il est conçu pour répertorier les noms et les caractéristiques des fichiers et des répertoires.
- Pour découvrir **ls** tapez **man ls**
- **ls -l** : lister les fichiers et les dossiers avec les droits
- **ls -a** : afficher les fichiers et les dossiers y compris les fichiers et dossiers cachés
- **ls -s** : afficher les fichiers avec les tailles allouées



L'éditeur VIM

- **vi**: suivit par le nom du fichier pour l'éditer
- **i,a**: passer en mode insertion pour ajouter le contenu
- **:q!**: quitter vim sans enregistrer
- **:w**: enregistrer vim
- **:wq**: enregistrer et quitter
- **A**: déplacer le curseur à la fin de la ligne
- **o**: Nouvelle ligne en bas
- **O**: Nouvelle ligne en haut
- **w**: sauter mot par mot vers l'avant
- **B**: sauter mot par mot vers l'arrière
- **r**: remplacer un mot
- **C**: effacer tout ce qui vient après
- **dd**: supprimer une ligne ajouter un nombre pour supprimer le nombre de lignes
- **u**: annuler une action ajouter un nombre annuler le nombre de actions
- **zz**: centrer la vue
- **ggdG**: Effacer tout
- **/:** chercher une occurrence n occurrence suivante
N occurrence précédente
dernière occurrence
+ première occurrence



L'éditeur VIM

- **s/ancien/nouveau/g**: changer une occurrence une fois
- **%s/ancien/nouveau/g**: changer toutes les occurrences
- **set number**: numéroter les lignes
- **:colorscheme** : changer la couleur
- **set mouse=a** : activer la souris pour désactiver mettre un set mouse = " "



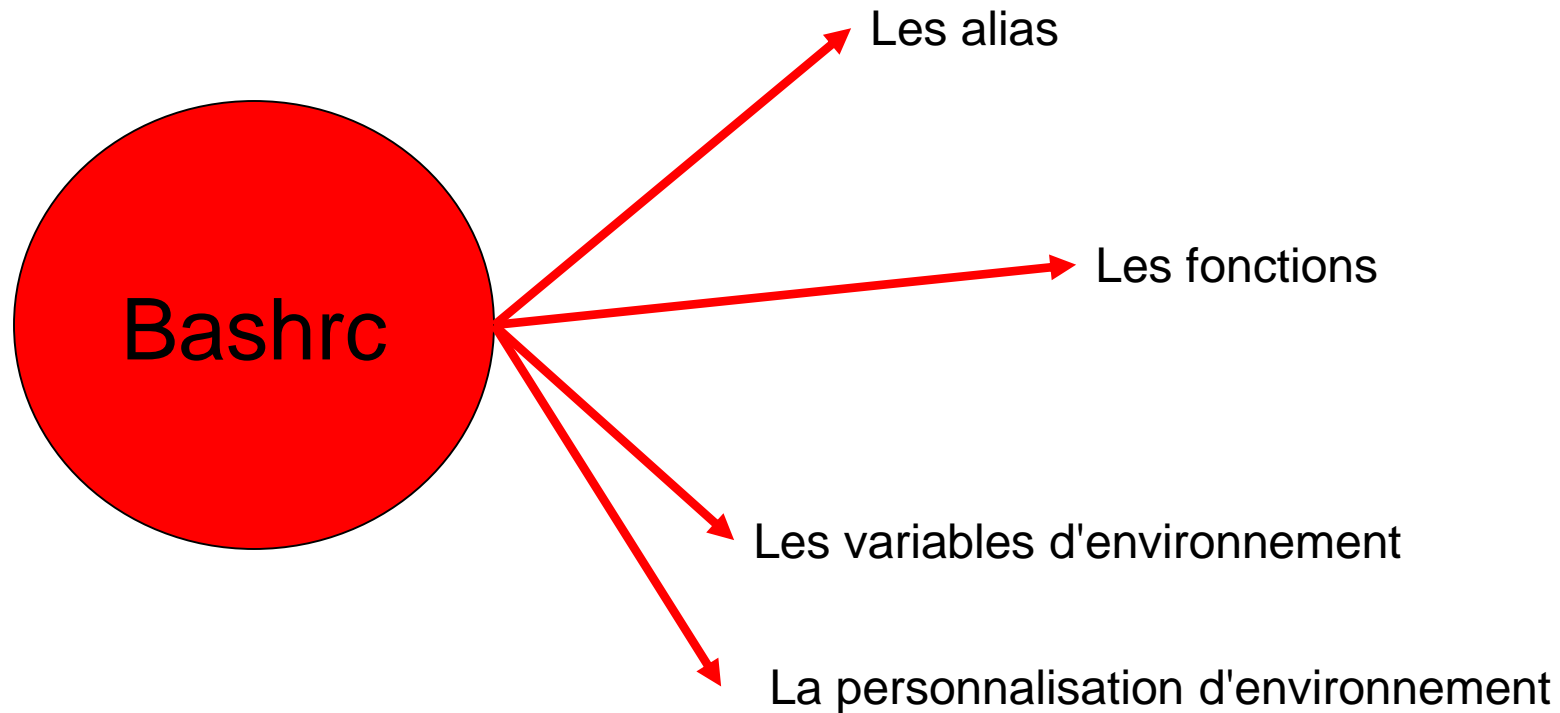
L'éditeur NANO

- **Ctrl + O**: sauvegarder un fichier en cliquant **Enter**
- **Ctrl + R**: ouvrir le contenu d'un autre fichier dans le fichier courant
- **Ctrl + A**: sélectionner les données avec les flèches
- **Alt + ^** : copier les données
- **Ctrl + U**: coller les données
- **Ctrl + K**: couper les données
- **Alt + **: naviguer vers le début du fichier
- **Alt + /**: naviguer vers la fin du fichier
- **Alt + G**: naviguer vers une ligne
- **Ctrl + W**: chercher une occurrence
- **Alt + R**: remplacer une occurrence



Les fichiers et dossiers particuliers (**bashrc** et **bashrc_profile**)

- Le fichier **.bashrc** abréviation de **bash read command** est un fichier de configuration pour l'environnement shell Bash. Chaque fois qu'une session interactive du shell Bash démarre, le fichier de script *.bashrc* s'exécute.



Les fichiers et dossiers particuliers(bashrc & bash_profile)

- La modification des variables d'environnement y compris \$PATH se fait avec la commande **export** mais une fois la session est fermée, il n'est plus possible de lancer le fichier de n'importe quel emplacement
- Il faut ajouter la valeur dans le fichier ~/.bashrc

```
#export /home/ubuntu  
export PATH="/home/ubuntu:$PATH"
```



Les fichiers et dossiers particuliers (bashrc & bash_profile)

- Les variables d'environnement sont définies dans le **.bashrc** y compris la variable **PATH**

```
[centos@localhost Folder1]$ pwd  
/home/centos/Folder1  
[centos@localhost Folder1]$ ls  
bechir.sh
```

```
[centos@localhost ~]$ echo "PATH=$PATH:/home/bechir/Folder1" >> .bashrc | source .bashrc
```

- Maintenant **bechir.sh** est appelé à partir de n'importe quel emplacement tout le temps

Note: Pour l'appel des fichiers bash il faut les appeler sans **./** exemple test.sh au lieu de ./test.sh



Les fichiers et dossiers particuliers (Le **bashrc** & **bash_profile**)

- Les deux fichiers servent le même but mais **bash_profile** s'exécute une seule fois à l'ouverture d'une session, **bashrc** se lance avec chaque nouveau terminal
- **bashrc** est couramment utilisé pour définir des alias, définir des fonctions et personnaliser l'invite de commande
- **bash_profile** est couramment utilisé pour définir la variable **PATH** et pour exécuter des commandes qui ne sont nécessaires qu'une seule fois au début de la session

Exemple 1 : Définition d'alias

Supposons que vous souhaitiez définir un alias pour la commande **ls** afin de répertorier les fichiers au format long. Vous pouvez le faire en ajoutant la ligne suivante à votre fichier **bashrc**

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



Les fichiers et dossiers particuliers (Le bashrc & bash_profile)

Exemple 2 : Définition de la variable PATH

Supposons que vous ayez installé une application personnalisée dans votre répertoire personnel et que vous souhaitiez ajouter son emplacement à votre variable PATH afin de pouvoir l'exécuter depuis n'importe où dans votre shell

Exemple 3 : Personnalisation de l'invite

Supposons que vous souhaitiez personnaliser l'invite qui apparaît dans la fenêtre de votre terminal pour inclure la date et l'heure actuelles. Vous pouvez le faire en ajoutant la ligne suivante à votre fichier bashrc -
`export PS1='\u@\h \D{%F %T} \w\$ '`



Les fichiers et dossiers particuliers (Sudoers)

- Le fichier **Sudoers** est ouvert avec la commande **visudo** ou **vi /etc/sudoers**
- Le fichier **Sudoers** est ouvert également avec la commande suivante

```
cloudsigma@mserver:~$ sudo update-alternatives --config editor
There are 4 choices for the alternative editor (providing /usr/bin/editor).

  Selection    Path                        Priority  Status
-----
*  0            /bin/nano                   40       auto mode
   1            /bin/ed                     -100     manual mode
   2            /bin/nano                   40       manual mode
   3            /usr/bin/vim.basic          30       manual mode
   4            /usr/bin/vim.tiny           15       manual mode

Press <enter> to keep the current choice[*], or type selection number: █
```



Les fichiers et dossiers particuliers (Sudoers)

- Le fichier **Sudoers** est utilisé pour ajouter un utilisateur standard à la liste des super utilisateurs
- Le fichier se trouve sous `/etc/sudoers` et accessible également par la commande **sudo visudo**

```
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root    ALL=(ALL)        ALL
centos  ALL=(ALL)        ALL

## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES, LOCATE, DRIVERS

## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)        ALL

## Same thing without a password
# %wheel    ALL=(ALL)        NOPASSWD: ALL

## Allows members of the users group to mount and unmount the
## cdrom as root
# %users    ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mnt/cdrom

## Allows members of the users group to shutdown this system
# %users    localhost=/sbin/shutdown -h now
```



Les fichiers et dossiers particuliers (Sudoers)

- Certaines commandes tel que les mises à jour et les installations demandent des privilèges administrateurs, les utilisateurs doivent être des **Sudoers**
- Pour vérifier si un utilisateur fait partie des super utilisateurs c'est avec `sudo -l`

```
[centos@localhost ~]$ sudo -l
[sudo] Mot de passe de centos : 
Entrées par défaut pour centos sur localhost :
!visiblepw, always_set_home, m[bechir@localhost centos]$ sudo -l
HOSTNAME HISTSIZE KDEDIR LS_CO[sudo] Mot de passe de bechir : 
env_keep+="LC_COLLATE LC_IDENTDésolé, l'utilisateur bechir ne peut pas utiliser sudo sur localhost.
LC_TELEPHONE", env_keep+="LC_T[bechir@localhost centos]$ 
secure_path=/sbin\:/bin\:/usr/

L'utilisateur centos peut utiliser les commandes suivantes sur localhost :
(ALL) ALL
```

- Les super utilisateurs font partie des groupes
 - Wheel : Cas de Centos
 - Sudo: Cas de ubuntu
- Sinon il faut les ajouter à l'un de ces groupes avec `usermod -aG <nom du groupe> <nom du user>`
Wheel|sudo



Les fichiers et dossiers particuliers (Sudoers)

- Il est possible d'accorder l'exécution de certaines commandes à un utilisateur via le **sudo**

```
## Allow root to run any commands anywhere
root    ALL=(ALL)        ALL
centos  ALL=(ALL)        ALL
bechir  ALL=(ALL)        /usr/bin/ls,/usr/bin/rm,/usr/bin/rmdir,/usr/bin/mkdir
```

- Pour éviter la demande d'entrer un mot de passe à chaque exécution de commande en ajoutant **PASSWORD:**

```
## Same thing without a password
# %wheel    ALL=(ALL)        NOPASSWD: ALL
```



Les fichiers et dossiers particuliers (.ssh)

- Un autre dossier important est le dossier .ssh qui contient les clés ssh qui permettent de se connecter à distance à un serveur distant à travers l'exécution des commandes
ssh-keygen -t rsa
ssh-copy-id utilisateur@IP
- Et puis se connecter à travers la commande **ssh 'user@serverip'**

```
bechir@PC2023:/mnt/c/Users/DELL/.ssh$ ssh-copy-id ubuntu@192.168.1.17
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/bechir/.ssh/id_rsa.pub"
The authenticity of host '192.168.1.17 (192.168.1.17)' can't be established.
ED25519 key fingerprint is SHA256:Ix4MtBuMBTcJPcSOvMBXHtYt5sbr3KckE+ByJ2EPbh4.
```

```
bechir@PC2023:/mnt/c/Users/DELL/.ssh$ ssh 'ubuntu@192.168.1.17'
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 6.2.0-35-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of sam. 28 oct. 2023 21:22:12 UTC

System load:  0.0419921875      Processes:           118
```



Les fichiers et dossiers particuliers (.ssh)

- La clé publique sera stockée dans la machine distante dans le fichier **authorized_key** sous le répertoire **.ssh** de l'utilisateur distant tant quelle est existante et valide il est possible de se connecter à distance

```
[centos@localhost .ssh]$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGCZYeo5DaDnMP1s4hkpWuRpP0IQyx0YvcI/zyydUYaCWx9eR731PDxtVQTx32xg9R/xLuqmiGA+CIwGjFmvpEgC
1Dzk0PR4eIfCcJ9ySti1c38Xsb20tW3uoZwLNlIjL/lFe1iQGcwl1J8fybDBhcgSxUfjcvFB5+IzGU/5Gajv7aNWx3aNkJA4mEMr+K0gyj3wrAlprcDndYYk1iEv
B0jtI5lVtJwzL7ZqWepNUoQTuHLvmh2gyoiNKYCIj3UCrtuM6Bu0ZfguTpDf2KtAJsF0yC2RBB3CKomPhllmKuT6F3F0pyvvnv88IQIAiGxod2yq+HLZ0iWjvNtR
avTI8pfB8IzRkFK//MQkkCNxDzB2eZTAhwwwPNGtVgbq32C3whBUdp/pbLb3GCLxAN/RaSHepm4EbNE8ZeqjVD44PpPU1yH/7oM/2d7eJmy+WR3K6g9JWqxDoPtX
tHAIN7zl/IzwNNwfjQh/Zv9B1z9NHInLOULMsxAEhBofYSwvDByh+I8= bechir@PC2023
[centos@localhost .ssh]$
```



Les fichiers et dossiers particuliers

root ALL=(ALL:ALL) ALL

- Le premier champ indique le nom d'utilisateur auquel la règle s'appliquera (root).
- Tout d'abord « ALL » indique que cette règle s'applique à tous les hôtes.
- Le deuxième « ALL » indique que l'utilisateur peut exécuter des commandes en tant que tous les utilisateurs en cas de "Impersonation".
- Le troisième « ALL » indique que l'utilisateur peut exécuter des commandes avec tous les groupes.
- Le quatrième « ALL » indique que ces règles s'appliquent à toutes les commandes.



Les fichiers et dossiers particuliers

root ALL=(ALL:ALL) ALL

- Le premier champ indique le nom d'utilisateur auquel la règle s'appliquera (root).
- Tout d'abord « ALL » indique que cette règle s'applique à tous les hôtes.
- Le deuxième « ALL » indique que l'utilisateur peut exécuter des commandes en tant que tous les utilisateurs en cas de "Impersonation".
- Le troisième « ALL » indique que l'utilisateur peut exécuter des commandes avec tous les groupes.
- Le quatrième « ALL » indique que ces règles s'appliquent à toutes les commandes.



Les dossiers partagés NFS

- La configuration d'un partage NFS (Network File System) pour partager le répertoire ou d'exporter le répertoire pour que les clients puissent y accéder

```
sudo yum install nfs-utils nfs-utils-lib
```

```
sudo mkdir /shared_folder
```

```
sudo chmod -R 755 /shared_folder
```

```
sudo chown nfsnobody:nfsnobody /shared_folder
```

```
sudo vi /etc/exports <= /shared_folder *(rw,sync,no_root_squash)
```

```
sudo systemctl start nfs-server
```

```
sudo systemctl enable nfs-server
```

```
sudo firewall-cmd --permanent --zone=public --add-service=nfs
```

```
sudo firewall-cmd --permanent --zone=public --add-service=mountd
```

```
sudo firewall-cmd --permanent --zone=public --add-service=rpc-bind
```

```
sudo firewall-cmd --reload
```

```
sudo exportfs -a
```

```
sudo yum install nfs-utils
```

```
sudo mount -t nfs SERVER_IP:/shared_folder /mnt
```

```
/etc/fstab <= SERVER_IP:/shared_folder /mnt nfs defaults 0 0
```

SERVER_IP est l'adresse du serveur



Le Shell Scripting

Les shells sous Linux

- Pour lister les shells sous Linux **cat /etc/shells**

```
[centos@localhost ~]$ cat /etc/shells
/bin/sh
/bin/bash
/usr/bin/sh
/usr/bin/bash
```

les shells sous centos/red hat

```
ubuntu@m1:~$ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/bash
/usr/bin/bash
/bin/rbash
/usr/bin/rbash
/usr/bin/sh
/bin/dash
/usr/bin/dash
/usr/bin/tmux
/usr/bin/screen
```

les shells sous ubuntu

- La commande **chsh** permet de passer d'un Shell à un autre

```
ubuntu@m1:~$ chsh
Password:
Changing the login shell for ubuntu
Enter the new value, or press ENTER for the default
    Login Shell [/bin/bash]: /bin/sh
ubuntu@m1:~$
```



Les shells sous Linux

- Pour connaître la version actuelle du Shell **ps -p \$\$**

```
ubuntu@m1:~$ ps -p $$
  PID TTY          TIME CMD
   980 pts/0    00:00:00 bash
```

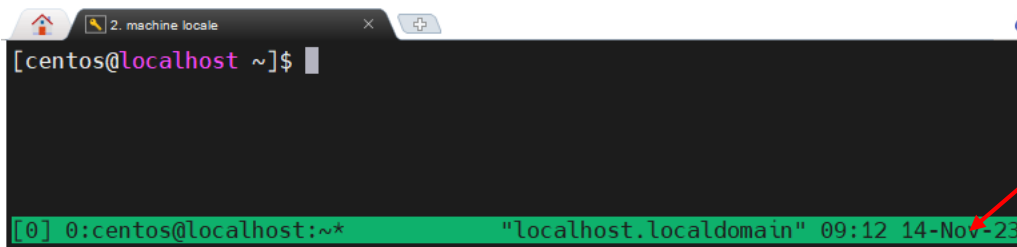


Les shell tmux

- Le Shell **tmux Terminal Multiplexer** n'est pas installé par défaut sous **Centos** il faut l'installer via les commandes

```
yum install epel-release  
yum install tmux
```

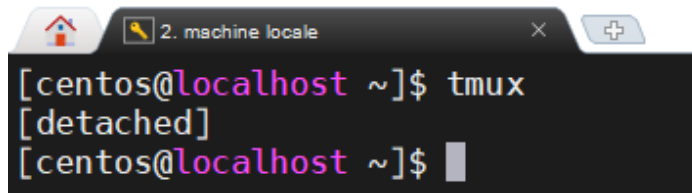
- Le Shell **tmux** est lancé via la commande **tmux**

A terminal window titled "2. machine locale" showing a tmux session. The prompt is [centos@localhost ~]\$ and the cursor is at the end of the line. Below the main terminal area, a green status bar displays [0] 0:centos@localhost:~* "localhost.localdomain" 09:12 14-Nov-23. A red arrow points from the text "La barre verte" to this status bar.

```
[centos@localhost ~]$  
[0] 0:centos@localhost:~* "localhost.localdomain" 09:12 14-Nov-23
```

La barre verte

- Le Shell **tmux** est quitté via le raccourcis clavier Ctrl + B et puis D

A terminal window titled "2. machine locale" showing the execution of the 'tmux' command. The prompt is [centos@localhost ~]\$ and the output is [detached]. The prompt then changes to [centos@localhost ~]\$ with a cursor at the end of the line.

```
[centos@localhost ~]$ tmux  
[detached]  
[centos@localhost ~]$
```

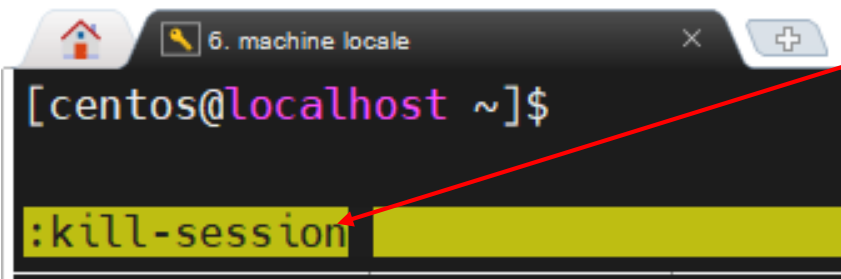


Les Shell tmux

- Le détachement du Shell **tmux** ne veut pas dire que la session est rompue

```
[centos@localhost ~]$ tmux ls  
0: 1 windows (created Tue Nov 14 09:24:28 2023) [110x31]  
[centos@localhost ~]$ tmux a -t 0  
[detached]  
[centos@localhost ~]$
```

- Afficher la liste des sessions tmux avec la commande **tmux ls**
- Lancer la session tmux à nouveau avec la commande **tmux a -t <identifiant de session>**
- Tuer la session tmux avec le raccourcis **Ctrl + B** et puis taper **: kill-session**



A terminal window titled "6. machine locale" showing a shell prompt [centos@localhost ~]\$. The command :kill-session is entered and highlighted in yellow. A red arrow points from the text ": kill-session" in the list above to this command in the terminal.

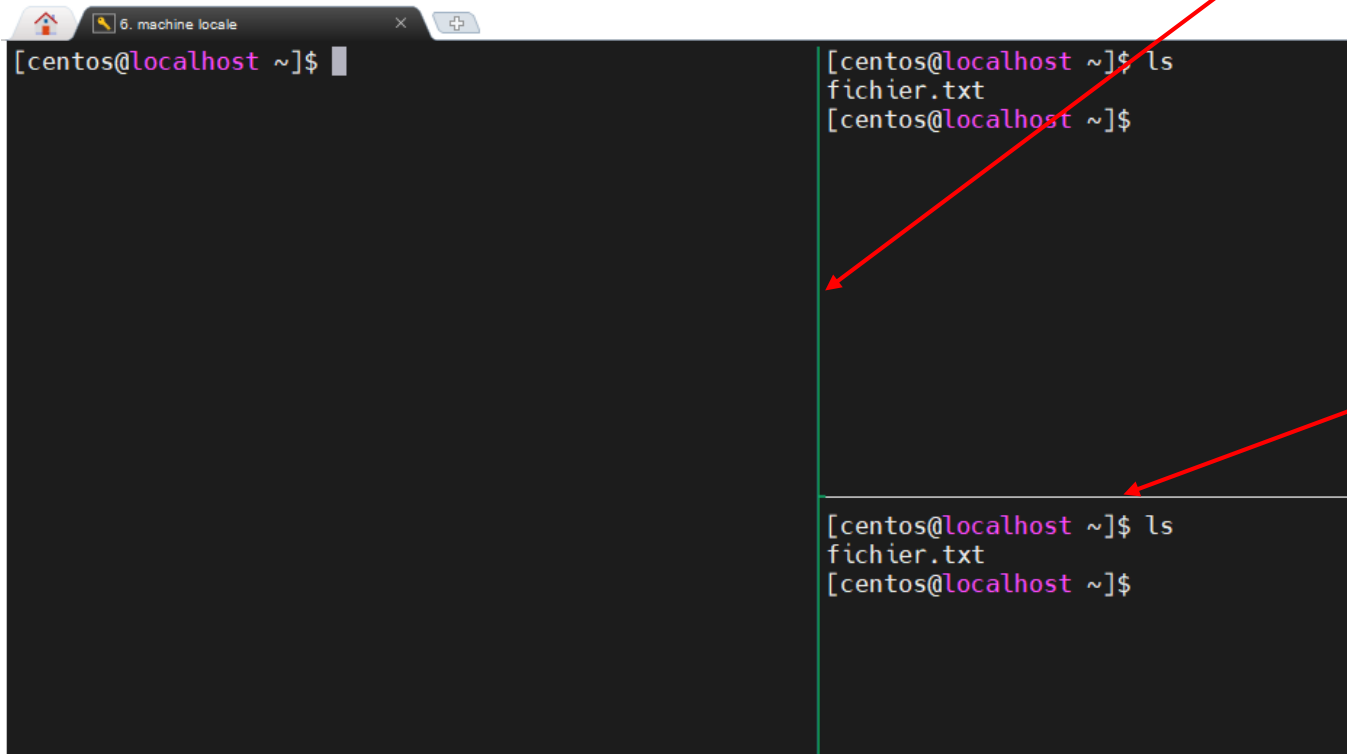
- Tuer la session tmux également avec **exit**



Les Shell tmux (Multi écran)

- Pour diviser l'écran en verticalement **Ctrl + B** et puis **%**
- Pour diviser l'écran en horizontalement **Ctrl + B** et puis **"**

Split vertical



```
[centos@localhost ~]$  
[centos@localhost ~]$ ls  
fichier.txt  
[centos@localhost ~]$  
[centos@localhost ~]$ ls  
fichier.txt  
[centos@localhost ~]$
```

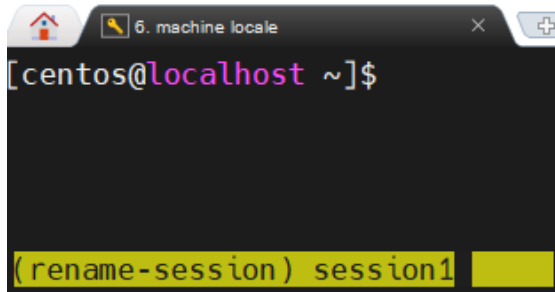
Split horizontal

- Pour naviguer entre les fenêtres c'est avec **Ctrl +B** et les flèches
- Pour fermer une fenêtre c'est avec la commande **exit**



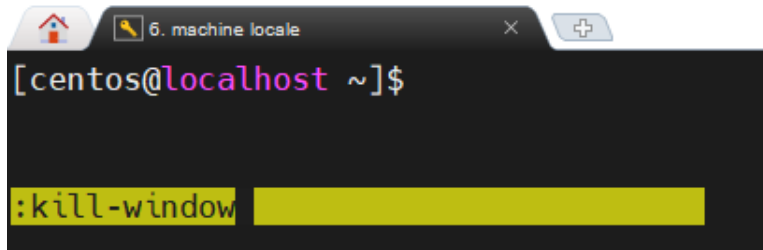
Les Shell tmux (Multi fenêtres)

- Pour créer des fenêtres multiples indépendantes, entrer dans en tmux et puis presser **Ctrl + C**
- Pour naviguer entre les fenêtres en avant et en arrière respectivement avec **Ctrl +B** et puis **N "Next"** ou **Ctrl +B** et puis **P "Previous"**
- Pour renommer une fenêtre **Ctrl + B** et presser **,** ou à l'intérieur de la session taper **tmux rename-session <nom de session>** ou **Ctrl + B** et presser **\$**



```
[centos@localhost ~]$  
(rename-session) session1
```

- Pour supprimer une fenêtre **Ctrl + B** et **:kill-window**

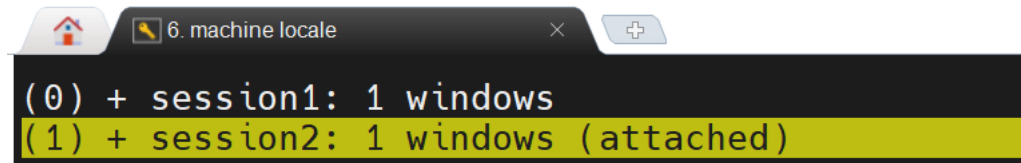


```
[centos@localhost ~]$  
:kill-window
```



Les Shell tmux (Multi fenêtres)

- Pour naviguer entre les sessions **Ctrl + B** et presser **S** et naviguer à travers les flèches

A screenshot of a terminal window with a dark background. The title bar at the top shows a home icon, a yellow lightning bolt icon, and the text "6. machine locale". The terminal content shows two lines: "(0) + session1: 1 windows" and "(1) + session2: 1 windows (attached)". The second line is highlighted in yellow.

```
(0) + session1: 1 windows  
(1) + session2: 1 windows (attached)
```

- Pour tuer toutes les sessions presser **kill -f tmux**



Linux informations générales

- La notion d'exécutable n'existe pas sous linux
- L'équivalent de MS-DOS bat files sont les fichiers bash, sh et zsh (Z shell)
 - Shell
 - Bourne Again Shell (history,
 - Z Shell



Etapes d'écriture du script shell

Les scripts Bash commencent par un shebang. Shebang est une combinaison bash #et bang ! suivi du chemin du shell bash

```
#!/bin/bash
```

La commande **which bash** permet de localiser le bash

Notre premier script invite l'utilisateur à saisir un chemin. En contrepartie, son contenu sera répertorié

```
#!/bin/bash  
echo "Today is " date
```

```
echo -e "\nenter the path to directory"  
read the_path
```

```
echo -e "\n you path has the following files and folders: "  
ls $the_path
```

Pour rendre le script exécutable, attribuez les droits d'exécution à votre utilisateur à l'aide de cette commande
chmod u+x run_all.sh



Etapes d'écriture du script shell

Pour exécuter le script **shell** `./<nom du fichier>.sh`



Les variables

➤ **Créer une variable:**

Nom de la variable = valeur (pas d'espaces, pas de \$)
lire le nom de la variable (pas de \$)

➤ **Accéder à la valeur d'une variable:**

\$ nom de la variable

➤ **Lister toutes les variables:**

Set | more

➤ Les variables sont partagées uniquement avec leur propre processus, sauf si elles sont exportées

- x=2 – définir x dans le processus en cours
- sh – lancer un nouveau processus
- echo \$x – impossible de voir x depuis le processus parent
- x = au revoir
- <ctrl d> -- quitter le nouveau processus
- export x
- sh – lancer un nouveau processus
- echo \$x –x est maintenant visible



La commande read

- **read x:** Lit la variable x



Les variables numériques

➤ Créer une variable numérique:

```
declare -i x=1          declare -i number
declare -i y=2          number=6/3
declare -i z=0          echo $number
z=$(( x + y ))
echo "$x + $y = $z"
```

➤ Créer une variable en lecture seule:

```
declare -r var1=1
```

```
[centos@localhost ~]$ declare -r var1=1
[centos@localhost ~]$ echo "var1 = $var1"    # var1 = 1
var1 = 1
[centos@localhost ~]$ (( var1++ ))
-bash: var1 : variable en lecture seule
```

Note: Le **Bash** ne prend pas les nombres flottants par défaut



Les flux conditionnels

➤ La structure conditionnelle:

If: Exécutez un ensemble de commandes si un test est vrai

Else: Si le test n'est pas vrai, exécuter un autre ensemble de commandes

Elif: Si le test précédent a donné faux, essayer celui-ci

&&: Effectuer l'opération et

||: Effectuer l'opération ou

Case: Choisir un ensemble de commandes à exécuter en fonction d'une chaîne correspondant à un modèle particulier



Les flux conditionnels

➤ La structure conditionnelle (if - fi):

```
#!/bin/bash
```

```
# block if basique
```

```
declare -i number
```

```
echo "Entrer un nombre entre 1 et 1000"
```

```
read number
```

```
if [ $number -gt 100 ];
```

```
then
```

```
echo $USER ce nombre est supérieur à 100
```

```
fi
```

```
if [ $number -gt 100 ] then
```

```
echo $USER ce nombre est supérieur à 100
```

```
fi
```

Cette forme génère une erreur car le point virgule est absent



Les flux conditionnels

➤ La structure conditionnelle (if – else -fi):

```
#!/bin/bash
```

```
# block if basique
```

```
declare -i number
```

```
echo "Entrer un nombre entre 1 et 1000"
```

```
read number
```

```
if [ $number -gt 100 ];
```

```
then
```

```
echo $USER ce nombre est supérieur à 100
```

```
fi
```



Les flux conditionnels

➤ La structure conditionnelle (if – elif -else -fi):

```
#!/bin/bash
```

```
declare -i number
```

```
echo "Entrer un nombre entre 1 et 1000"
```

```
read number
```

```
if [ $number -gt 100 -a $number -lt 200 ]; then  
echo $USER ce nombre est supérieur à 100 mais  
inférieur à 200
```

```
elif [ $number -gt 200 ]; then  
echo $USER ce nombre est supérieur à 200
```

```
else  
echo $USER ce nombre est inférieur à 100
```

```
fi
```



Les flux conditionnels

➤ La structure conditionnelle (case):

```
#!/bin/bash
```

```
# Exemple de case
```

```
echo "entrer une couleur primaire"  
read valeur
```

```
case $valeur in
```

```
rouge)
```

```
echo "La couleur est rouge"
```

```
;;
```

```
vert)
```

```
echo "La couleur est verte"
```

```
;;
```

```
bleu)
```

```
echo "La couleur est bleue"
```

```
;;
```

```
*)
```

```
echo "La couleur n'est pas primaire"
```

```
;;
```

```
esac
```



La clause select

```
#!/bin/bash
```

```
select character in Sheldon Leonard Penny Howard Raj
do
    echo "Selected character: $character"
    echo "Selected number: $REPLY"
    break
done
```

Les flux conditionnels

➤ La structure conditionnelle (if – elif -else -fi):

```
#!/bin/bash
```

```
declare -i number  
echo "Entrer un nombre entre 1 et 1000"  
read number
```

```
if [ $number -gt 100 -a $number -lt 200 ]; then  
echo $USER ce nombre est supérieur  
à 100 mais inférieur à 200  
elif [ $number -gt 200 ]; then  
echo $USER ce nombre est supérieur à 200  
else  
echo $USER ce nombre est inférieur à 100  
fi
```

```
#!/bin/bash
```

```
declare -i number  
echo "Entrer un nombre entre 1 et 1000"  
read number
```

```
if [ $number -gt 100 ] && [ $number -lt 200 ]; then  
echo $USER ce nombre est supérieur  
à 100 mais inférieur à 200  
elif [ $number -gt 200 ]; then  
echo $USER ce nombre est supérieur à 200  
else  
echo $USER ce nombre est inférieur à 100  
fi
```



Les flux conditionnels (Les conditions)

Opérateur de tests	Tests Vrai si
<code>[chaîne1 = chaîne2]</code>	String1 est égal à String2 (espace entourant = est nécessaire)
<code>[chaîne1 != chaîne2]</code>	String1 n'est pas égal à String2 (espace entourant != n'est pas nécessaire)
<code>[chaîne]</code>	La chaîne n'est pas nulle.
<code>[-z chaîne]</code>	La longueur de la chaîne est nulle.
<code>[-n chaîne]</code>	La longueur de la chaîne est différente de zéro.
<code>[-l chaîne]</code>	Longueur de la chaîne (nombre de caractères)

Les flux conditionnels (Les conditions)

```
#!/bin/bash
```

```
echo "entrer string1"
read string1
```

```
echo "entrer string2"
read string2
```

```
# Verifie si string1 égale à string2
```

```
if [ "$string1" = "$string2" ]; then
    echo "Strings are equal."
```

```
else
```

```
    echo "Strings are not equal."
```

```
fi
```

```
# Verifie si string1 n'est pas égale à string2
```

```
if [ "$string1" != "$string2" ]; then
    echo "Strings are not equal."
```

```
else
```

```
    echo "Strings are equal."
```

```
fi
```

Exemple 1

```
#!/bin/bash
```

```
echo "Entrer la première chaine"
read string1
```

```
echo "Entrer la deuxième chaine"
read string2
```

```
# Converti les deux chaines tout d'abord en miniscule
```

```
if [[ ${string1,,} == ${string2,,} ]]; then
```

```
    echo "Les chaines sont égales (case non considérée)."
```

```
else
```

```
    echo "Les chaines ne sont pas égales (casse non considérée)."
```

```
fi
```

```
# Converti les deux chaines tout d'abord en majuscule
```

```
if [[ ${string1^^} == ${string2^^} ]]; then
```

```
    echo "Les chaines sont égales (casse non considérée)."
```

```
else
```

```
    echo "Les chaines ne sont pas égales (casse non considérée)."
```

```
fi
```

Exemple 2

Les flux conditionnels (Les conditions)

```
#!/bin/bash
```

```
echo "Entrer la phrase"
```

```
read string
```

```
echo "Entrer le mot recherché"
```

```
read substring
```

```
if [[ "$string" == *"$substring"* ]]; then
    echo "Substring found: $substring"
else
    echo "Substring not found: $substring"
fi
```

Exemple 3

```
#!/bin/bash
```

```
echo "Entrer un Email"
```

```
read email
```

```
# Check if the string matches the pattern for an email address
```

```
if [[ "$email" =~ ^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$ ]]; then
    echo "Valid email address: $email"
else
    echo "Invalid email address: $email"
fi
```

Exemple 4

Les flux conditionnels (Les conditions)

Opérateur de tests	Test vrai si
[<i>cond1</i> -a <i>cond2</i>]	condition1 et condition 2 sont vraies.
[<i>cond1</i> -o <i>cond2</i>]	La condition 1 ou la condition 2 sont toutes deux vraies.
[! <i>chaîne</i>]	Pas une correspondance condition1

Opérateur de test	Tests Vrai si
[[<i>cond1</i> && <i>cond2</i>]]	condition1 et condition2 sont vrais
[[<i>cond1</i> <i>cond2</i>]]	Soit le condition 1, soit le condition 2 est vrai
[[! <i>cond</i>]]	Pas une correspondance de condition

Les flux conditionnels (Les conditions)

Opérateur de test	Tests Vrai si
[<i>int1</i> -eq <i>int2</i>]	int1 = int2
[<i>int1</i> -ne <i>int2</i>]	int1 ≠ int2
[<i>int1</i> -gt <i>int2</i>]	int1 > int2
[<i>int1</i> -ge <i>int2</i>]	int1 ≥ int2
[<i>int1</i> -lt <i>int2</i>]	int1 < int2
[<i>int1</i> -le <i>int2</i>]	int1 ≤ int2

Opérateur de tests	Test vrai si
[<i>fichier1</i> -nt <i>fichier2</i>]	Vrai si le fichier1 est plus récent que le fichier2*
[<i>fichier1</i> -ot <i>fichier2</i>]	Vrai si le fichier1 est plus ancien que le fichier2*
[<i>fichier1</i> -ef <i>fichier2</i>]	Vrai si file1 et file2 ont les mêmes numéros de périphérique et d'inode .

Test des fichiers

Opérateur de tests	Testez Vrai si :
-d nom de fichier	Existence de dossier
-e nom de fichier	Existence de fichier
-f nom de fichier	Existence régulière d'un fichier et non d'un répertoire
-G nom de fichier	Vrai si le fichier existe et appartient à l'identifiant de groupe effectif
-g nom de fichier	Set-group-ID est défini
-L nom de fichier	Le fichier est un lien symbolique

Test des fichiers

Opérateur de tests	Testez Vrai si :
-p nom de fichier	Le fichier est un named pipe
-O nom de fichier	Le fichier existe et appartient à l'ID utilisateur effectif
-r nom de fichier	le fichier est lisible
-S nom de fichier	le fichier est un socket
-s nom de fichier	le fichier est de taille différente de zéro
-t fd	Vrai si fd (descripteur de fichier) est ouvert sur un terminal
-u nom de fichier	Set-user-id est défini
-w nom de fichier	Le fichier est accessible en écriture
-x nom de fichier	Le fichier est exécutable

Les boucles

```
#!/bin/bash
```

```
counter=1
while [ $counter -le 10 ]
do
echo $counter
((counter++))
done
echo All done
```

La boucle tant que

```
#!/bin/bash
```

```
counter=1
until [ $counter -gt 10 ]
do
echo $counter
((counter++))
done
echo All done
```

La boucle repeater jusqu'a

```
#!/bin/bash
```

```
for value in {1..5}
do
echo $value
done
echo All done
```

```
#!/bin/bash
```

```
for value in {10..5..2}
do
echo $value
done
echo All done
```

La boucle pour

```
#!/bin/bash
```

```
for ((num = 1; num <= 5;
num++))
do
echo $num
done
echo All done
```

Les boucles

```
#!/bin/bash
```

```
for v in 1 2 3 4 5 6 7 8 9  
do  
    echo $v  
done
```

```
#!/bin/bash
```

```
s=(1 2 3)  
for n in ${s[@]};  
do  
    echo $n  
done
```

La boucle pour

Les fonctions

- Les fonctions de Bash Scripting sont un excellent moyen de réutiliser du code
- Considérer une fonction comme un petit script dans un script. Il s'agit d'un petit morceau de code que vous pouvez appeler plusieurs fois dans votre script

```
#!/bin/bash
```

```
# Fonction basique
```

```
declare -i n1
```

```
declare -i n2
```

```
echo "Entrer une première valeur"
```

```
read n1
```

```
echo "Entrer une première valeur"
```

```
read n2
```

```
print_somme () {
```

```
echo La somme de $n1 et $n2 est $((n1 + n2))
```

```
}
```

```
print_somme
```

Les fonctions (Le passage des paramètres)

- Voici un exemple de passage de paramètres et récupération des valeurs de retour

```
#!/bin/bash
# Setting a return status for a function
print_something () {
echo Hello $1
return 5
}
print_something Mars
print_something Jupiter
echo The previous function has a return value of $?
```


Les fonctions (Les variables globales vs les variables locales)

- Voici un exemple de passage de paramètres et récupération des valeurs de retour

```
#!/bin/bash
```

```
# Les portées de variables
```

```
var_change () {
```

```
local var1='local 1'
```

```
echo A l'intérieur de la fonction: var1 est $var1 : var2 est $var2
```

```
var1='changed again'
```

```
var2='2 changed again'
```

```
}
```

```
var1='global 1'
```

```
var2='global 2'
```

```
echo Avant appel de fonction: var1 est $var1 : var2 est $var2
```

```
var_change
```

```
echo Après appel de fonction: var1 est $var1 : var2 est $var2
```

Les fonctions enveloppe

- Il est possible de créer une enveloppe de commande

```
# Create a wrapper around the command ls
ls () {
  command ls -lh
}
ls
```

```
[root@localhost ~]# ./loops.sh
total 16K
-rw-----. 1 root root 1,5K 27 oct. 10:51 anaconda-ks.cfg
drwxr-xr-x. 2 root root  6 14 nov. 16:54 folder
drwxr-xr-x. 2 root root  6 14 nov. 17:09 folder2
-rwxr-xr-x. 1 root root 69 15 nov. 17:53 loops.sh
-rw-r--r--. 1 root root  0 14 nov. 16:03 mypasswd.txt
-rwxr-xr-x. 1 root root 212 15 nov. 15:34 test.sh
-rwxr-xr-x. 1 root root 256 15 nov. 16:24 tom.sh
-rw-r--r--. 1 root root  0 14 nov. 16:05 unshadowed_password
```

Quelques exemples utiles de script bash (Généraux)

Crypter et décrypter un fichier

```
#!/usr/bin/env bash
echo "Entrer le nom exact et complet du fichier"
read -r file

if [ -e $file ]; then
    echo "Pour crypter le fichier taper 0, pour décrypter le fichier taper 1"
    read choix

    if [ $choix = 0 ];then
        gpg -c "$file"
        rm -rf "$file"
        echo "Fichier encrypté le $(date)"
    elif [ $choix = 1 ]; then
        gpg -d $file > file
        echo "Fichier décrypté le $(date)"
    else
        echo "Choix incorrect, il faut choisir 0 pour encrypter et 1 pour décrypter"
    fi
else
    echo "Nom du fichier incorrect ou fichier non existant"
fi
```

Quelques exemples utiles de script bash (Généraux)

Calculer la taille d'un dossier (simple)

```
#!/bin/bash
echo -n "Enter le nom du dossier: "
read -r x
du -sh "$x"
```

Calculer la taille d'un dossier (passage de paramètre de position)

```
#!/bin/bash
du -sh $1
```

Quelques exemples utiles de script bash (Généraux)

Encrypter décrypter un fichier (passage d'arguments avec flag)

```
#!/usr/bin/env bash
```

```
file = $1
```

```
if [ -e $file ]; then
```

```
    e_flag="
```

```
    d_flag="
```

```
    verbose='false'
```

```
    while getopts 'edf:v' flag; do
```

```
        case "${flag}" in
```

```
            e) e_flag='true' ;;
```

```
            d) d_flag='true' ;;
```

```
            f) file=${OPTARG}
```

```
;;
```

```
            v) verbose='true' ;;
```

```
            *) exit 1;;
```

```
        esac
```

```
    done
```

```
    if [ $e_flag = 'true' ];then
```

```
        gpg -c "$file"
```

```
        rm -rf "$file"
```

```
        if [ verbose = 'true' ]; then
```

```
            echo "Fichier encrypté le $(date)"
```

```
        fi
```

```
    elif [ $d_flag = 'true' ]; then
```

```
        gpg -d $file > file
```

```
        if [ verbose = 'false' ]; then
```

```
            echo "Fichier décrypté le $(date)"
```

```
        fi
```

```
    else
```

```
        echo "Choix incorrect, il faut choisir 0 pour encrypter
```

```
et 1 pour décrypter"
```

```
    fi
```

```
else
```

```
    echo "Nom du fichier incorrect ou fichier non existant"
```

```
fi
```

Utilisateurs et groupes

Gérer les utilisateurs et les groupes

- **/etc/passwd**: contient la liste des utilisateurs

```
cpdump:x:109:115::/nonexistent:/usr/sbin/nologin
ss:x:110:116:TPM software stack,,,:/var/lib/tpm:/bin/false
andscape:x:111:117::/var/lib/landscape:/usr/sbin/nologin
wupd-refresh:x:112:118:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
sbmux:x:113:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
buntu:x:1000:1000:bechir:/home/ubuntu:/bin/bash
```

- **/etc/group**: contient la liste des groupes

```
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,ubuntu
tty:x:5:
disk:x:6:
```

- **sudo useradd -m visiteur**: ajouter l'utilisateur visiteur avec le répertoire home
- **sudo adduser visiteur**: ajouter l'utilisateur visiteur avec le répertoire home également
- **sudo deluser visiteur**: supprimer l'utilisateur visiteur avec le répertoire home également



Gérer les utilisateurs et les groupes

- **id visiteur**: montre l'identifiant de l'utilisateur visiteur
- **adduser -g users visiteur**: crée et ajoute visiteur au groupe users
- **useradd**: Ajoute un utilisateur sans home **useradd ubuntu2 -g ubuntu -d /home/ubuntu2 -s /bin/bash**
- **useradd -g users -G root,mail visiteur**: crée visiteur avec le groupe users comme groupe principal et root mail comme les autres groupes
- **passwd visiteur**: crée un mot de passe pour l'utilisateur visiteur
- **Useradd -s /usr/bin/sh visiteur**: crée visiteur avec le **shell** par défaut sh
exécuter `grep visiteur /etc/passwd` pour afficher les infos sur visiteur
- **Useradd -e 2019-01-22 visiteur**: crée un utilisateur avec date d'expiration utiliser **chage -l** pour montrer les infos sur la date d'expiration

```
root@m1:/home/ubuntu# useradd -e 2019-01-22 visiteur
root@m1:/home/ubuntu# su visiteur
Your account has expired; please contact your system administrator.
su: Authentication failure
```

- **Useradd -D -s /bin/bash visiteur**: modifier l'utilisateur
- **Userdel visiteur**: supprimer l'utilisateur



Gérer les utilisateurs et les groupes

- **groupadd**: crée un groupe, ajouter `-g` suivi d'un numéro pour affecter un nombre spécifique
`groupadd -g 1111 visiteurs`
- **groupmod**: modifie un groupe, ajouter `-g` suivi d'un numéro pour affecter un nombre spécifique
`groupmod -g 1112 visiteurs` `groupmod -n invités visiteurs`
- **groupdel** : supprime un groupe
`groupdel visiteurs`
- **usermod -aG root,sudo visiteur** : ajoute visiteur au groupe root et le groupe sudo
- **gpasswd -d visiteur root** : supprime visiteur du groupe root



Les jeux de permissions

Les permissions

- Il y a trois actions que peut faire un utilisateur sous linux
- Lire
 - Ecrire
 - Exécuter

Numéro	Type d'autorisation	Symbole
0	Aucune autorisation	—
1	Exécuter	-x-
2	Écrire	-w-
4	Lire	-r-
3	Exécuter + Ecrire	-wx
5	Lire + executer	r-x
6	Lire + Écrire	rw-
7	Lire + Ecrire + Exécuter	rwX



Les permissions

- Il y a trois catégories d'objets qu'on pourra contrôler à travers un jeu de permissions
 - L'utilisateur **u**: généralement c'est l'utilisateur courant ou le root, ou le propriétaire de la ressource
 - Le groupe **g**: Généralement c'est le groupe auquel appartient l'utilisateur
 - Les autres **o**: Ce sont les autres utilisateurs non propriétaires de la ressource (Fichier, dossier)
- Par exemple , l'autorisation **777 sur le dossier /etc** signifie que le dossier dispose de toutes les autorisations de **lecture, d'écriture et** d'exécution pour le propriétaire, le groupe et tous les utilisateurs

Note: Pour voir les jeux de permissions d'un il faut utiliser la commande **ls -l**

Note: Pour changer le propriétaire d'un fichier ou dossier utiliser la commande **chown utilisateur:groupe fichier**

Note: Pour changer l'utilisateur propriétaire d'un fichier ou dossier utiliser la commande **chown utilisateur:groupe fichier**

Note: Pour changer le groupe propriétaire d'un fichier ou dossier utiliser la commande **chown utilisateur:groupe fichier**





La commande **chmod**

- La commande **chmod** est utilisée pour modifier les permissions d'un fichier
- Pour ajouter/retirer des autorisations d'exécution pour le propriétaire d'un fichier à un utilisateur
chmod u+x nom_fichier / chmod u-x nom_fichier
- Ou, pour ajouter/retirer des autorisations de lecture et d'écriture pour le groupe propriétaire du fichier
chmod g+rw nom_fichier / chmod g-rw nom_fichier
- Au lieu d'ajouter/retirer des autorisations, la syntaxe symbolique de chmod peut également être utilisée pour soustraire ou définir une valeur absolue
chmod ow nom_fichier ou **chmod u=rwx,g=rx,o= nom_fichier**
- La commande **chmod** peut également définir explicitement des autorisations à l'aide d'une représentation numérique
chmod 774 nom_fichier



Attribuer les permissions

<div><div>File Permissions Cheat Sheet LinuxSeb</div></div>		
What it Means: drwxrwxrwx or -rwxrwxrwx r = read w = write x = execute - = file d = directory	The Command: sudo chmod -R 755 (dirname) <div><div>755</div><div><div>↙</div><div>↓</div><div>↘</div></div><div>rwxxrwx</div><div>usergroupother</div></div>	Numerical Presentation: 7 = rwx 6 = rw- 5 = r-x 4 = r-- 3 = -wx 2 = -w- 1 = --x 0 = ---



La commande **chown**

- La commande **chown** est utilisée pour changer le propriétaire du fichier.
- **chown visiteur fichier1** : change la propriété du fichier1 à l'utilisateur visiteur il est possible d'écrire également **chown visiteur:visiteurs fichier1**
- **chown :visiteurs fichier1**: change la propriété du fichier1 au groupe visiteurs
- Pour inspecter les permissions utiliser la commande **getfacl** <nom répertoire> | <nom fichier>

```
ubuntu@m1:~$ getfacl /home/ubuntu/user_folder
getfacl: Removing leading '/' from absolute path names
# file: home/ubuntu/user_folder
# owner: ubuntu
# group: usrs
user::rwx
group::rwx
other::r-x
```

Remarque: Il faut installer le module **acl** avec **sudo apt install acl**



La commande chgrp

- **chgrp:** Change la propriété du groupe à un autre groupe exemple **sudo chgrp -R admins /test/permissions**



La commande **chown**

- La commande **chown** est utilisée pour changer le propriétaire du fichier.
- **chown visiteur fichier1** : change la propriété du fichier1 à l'utilisateur visiteur il est possible d'écrire également **chown visiteur:visiteurs fichier1**
- **chown :visiteurs fichier1**: change la propriété du fichier1 au groupe visiteurs
- Ajouter les deux options **-c** ou **-v** pour afficher des informations de sortie



La gestion des process

La commande top

➤ Un processus est un terme utilisé pour décrire une application ou un programme en cour d'exécution

➤ Les types de processus

Les processus de premier plan

dépendent de l'utilisateur pour la saisie,
également appelés processus interactifs.

Les processus en arrière-plan

s'exécutent indépendamment de l'utilisateur,
appelés processus non interactifs ou automatiques.

➤ Les états de processus

Running : Processus en cour d'exécution

Sleeping : Processus en état de pause

Interruptible sleep : dont l'état de pause peut être interrompu

Uninterruptible sleep : don't l'état de pause ne peut pas être interrompu

Stopped : Processus en état d'arrêt

Zombie : Processus est mort mais que l'entrée du processus est toujours présente dans le tableau



La commande top

- Suivre les processus en cours sur la machine avec la commande **top**

identifiant utilisateur priorité Valeur Nice Mémoire virtuelle Mémoire résidente Mémoire partagée CPU consommé % / total

```
ubuntu@ml:~$ top
top - 19:32:49 up 4:22, 2 users, load average: 0,30, 0,13, 0,03
Tasks: 109 total, 1 running, 107 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0,3 us, 0,7 sy, 0,0 ni, 98,8 id, 0,2 wa, 0,0 hi, 0,0 si, 0,0 st
MiB Mem : 1959,4 total, 1322,9 free, 190,8 used, 445,7 buff/cache
MiB Swap: 1339,0 total, 1339,0 free, 0,0 used, 1619,3 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
15	root	20	0	0	0	0	I	0,3	0,0	0:04.26	rcu_preempt
516	root	20	0	0	0	0	I	0,3	0,0	0:32.97	kworker/0:4-events
961	ubuntu	20	0	17464	8444	5888	S	0,3	0,4	0:50.27	sshd
1017	ubuntu	20	0	7368	3584	3328	S	0,3	0,2	0:37.05	bash
61270	root	20	0	0	0	0	I	0,3	0,0	0:00.94	kworker/u4:0-events
1	root	20	0	166308	11488	8288	S	0,0	0,6	0:02.39	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.01	kthreadd
3	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_par_gp
5	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	slub_flushwq

Mémoire consommée % / total

Durée d'exécution

La commande qui a lancé le process



Gérer les processus

- Voici quelques options utiles
 - **top -u nom d'utilisateur**: permet de filtrer les processus lancés par l'utilisateur spécifique
 - **top -n nombre d'itérations**: permet de définir le nombre d'itérations avant de retourner la commande
- Par défaut, la sortie de la commande supérieure est actualisée toutes les 3 secondes
- Pour modifier cet intervalle, appuyez sur la touche **d** pendant que la commande top est en cours d'exécution. Vous pouvez ensuite saisir la nouvelle heure

```
Tasks: 122 total, 1 running, 121 sleeping,  
%Cpu(s): 0,3 us, 0,3 sy, 0,0 ni, 99,4 id, 0  
MiB Mem : 1959,4 total, 1134,8 free, 192  
MiB Swap: 1339,0 total, 1339,0 free, 0  
Change delay from 1,0 to 3  
PID USER PR NI VIRT RES SHR  
961 ubuntu 20 0 17464 8572 6016
```



La commande top

- Lorsque vous appuyez sur la touche **z** pendant que votre commande top est en cours d'exécution, les processus actuellement actifs seront affichés en couleur

```
Centos [En fonction] - Oracle VM VirtualBox
Fichier Machine Écran Entrée Périphériques Aide
top - 22:08:34 up 4:47, 1 user, load average: 0,00, 0,01, 0,05
Tasks: 99 total, 1 running, 98 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0,2 us, 0,2 sy, 0,0 ni, 99,7 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem : 1881892 total, 1585596 free, 152092 used, 144204 buff/cache
KiB Swap: 1138684 total, 1138684 free, 0 used. 1580148 avail Mem

  PID USER      PR  NI   VIRT   RES    SHR S  %CPU  %MEM    TIME+  COMMAND
 9145 centos    20   0 162068   2184   1540 R   0,7   0,1   0:01.81 top
 9149 root      20   0     0      0      0 S   0,3   0,0   0:00.62 kworker/0:0
    1 root      20   0 128024   6656   4172 S   0,0   0,4   0:01.00 systemd
    2 root      20   0     0      0      0 S   0,0   0,0   0:00.01 kthreadd
    4 root       0 -20     0      0      0 S   0,0   0,0   0:00.00 kworker/0:0H
    5 root      20   0     0      0      0 S   0,0   0,0   0:00.07 kworker/u4:0
    6 root      20   0     0      0      0 S   0,0   0,0   0:00.60 ksoftirqd/0
    7 root      rt   0     0      0      0 S   0,0   0,0   0:00.01 migration/0
    8 root      20   0     0      0      0 S   0,0   0,0   0:00.00 rcu_bh
```



La commande top

- Si vous souhaitez afficher le chemin absolu des processus en cours d'exécution, appuyez sur la touche c

```
top - 20:10:24 up 4:59, 3 users, load average: 0,01, 0,05, 0,02
Tasks: 117 total, 1 running, 116 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0,9 us, 0,9 sy, 0,0 ni, 98,1 id, 0,0 wa, 0,0 hi, 0,2 si, 0,0 st
MiB Mem : 1959,4 total, 1131,3 free, 196,4 used, 631,7 buff/cache
MiB Swap: 1339,0 total, 1339,0 free, 0,0 used. 1612,1 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
24	root	20	0	0	0	0	I	0,7	0,0	0:28.99	[kworker/1:0-events]
336	root	rt	0	289452	27648	8960	S	0,3	1,4	0:07.21	/sbin/multipathd -d -s
961	ubuntu	20	0	17464	8572	6016	S	0,3	0,4	0:59.98	sshd: ubuntu@pts/0
81791	ubuntu	20	0	7368	3584	3328	S	0,3	0,2	0:03.14	bash -c while true; do sleep 1;head -v -n 8 /pr+
85768	root	20	0	0	0	0	I	0,3	0,0	0:01.03	[kworker/0:0-events]
1	root	20	0	166308	11488	8288	S	0,0	0,6	0:02.48	/sbin/init
2	root	20	0	0	0	0	S	0,0	0,0	0:00.01	[kthreadd]



La commande top

- Appuyez sur la touche k pendant que la commande supérieure est en cours d'exécution. Une invite vous posera des questions sur le PID que vous souhaitez supprimer. Entrez l'ID de processus requis en l'affichant dans la liste

```
Mem Swap: 1339,0 total, 1339,0 free, 0,0 used. 1616,4 avail Mem
PID to signal/kill [default pid = 637] 81556
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
892	ubuntu	20	0	17080	9728	8064	S	0,0	0,5	0:00.08	systemd
893	ubuntu	20	0	159360	5576	1792	S	0,0	0,3	0:00.00	(sd-pam)
899	ubuntu	20	0	8740	5504	3840	S	0,0	0,3	0:00.02	bash
980	ubuntu	20	0	8864	5632	3968	S	0,0	0,3	0:00.16	bash
1014	ubuntu	20	0	17312	8192	5760	S	0,0	0,4	0:00.00	sshd
1016	ubuntu	20	0	7768	5632	4608	S	0,0	0,3	0:00.00	sftp-server
81457	ubuntu	20	0	17464	8572	6016	S	0,0	0,4	0:00.06	sshd
81459	ubuntu	20	0	2888	1920	1792	S	0,0	0,1	0:00.00	sh
81506	ubuntu	20	0	17312	8324	5888	S	0,0	0,4	0:00.00	sshd
81507	ubuntu	20	0	2888	1664	1664	S	0,0	0,1	0:00.00	sh
81508	ubuntu	20	0	7768	5632	4608	S	0,0	0,3	0:00.00	sftp-server
81556	ubuntu	20	0	21360	10880	7168	S	0,0	0,5	0:00.03	vi
89692	ubuntu	20	0	10480	3968	3328	R	0,0	0,2	0:00.02	top
89758	ubuntu	20	0	5768	1920	1920	S	0,0	0,1	0:00.00	sleep



La commande top

- Vous pouvez enregistrer l'état actuel de votre système pour une utilisation ultérieure si vous enregistrez la sortie de la commande top dans un fichier texte

top -n 1 -b > top.txt

Le nombre d'états

Option pour sauvegarder dans un fichier

Le fichier de stockage



La commande top

➤ Voici quelques touches utiles à utiliser avec **top**

- H ou ?: Afficher une fenêtre d'aide avec toutes les commandes et autres informations utiles.
- Espace: Appuyez dessus pour mettre à jour la liste des processus.
- F : Ajouter des champs ou supprime certains champs de la table d'affichage
- Q :quitte l'application top ou une fenêtre rattachée à top
- L : Affiche les informations relatives à la disponibilité et l'utilisation moyenne.
- M : Permet d'afficher des informations sur la mémoire.
- P (Shift + p) : Trier les processus en fonction de l'utilisation du processeur.

Autres usages utiles de top



La commande ps

- La commande ps est l'abréviation de « Statut du processus ». Il affiche les processus en cours d'exécution

```
ubuntu@m1:~$ ps
  PID TTY          TIME CMD
  980 pts/0        00:00:00 bash
 91948 pts/0        00:00:00 ps
```

- La commande ps -u affiche les processus en cours par utilisateur

```
ubuntu@m1:~$ ps -u
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
ubuntu    899  0.0  0.2   8740  5504 tty1     S+   15:11    0:00 -bash
ubuntu    980  0.0  0.2   8864  5632 pts/0    Ss   15:12    0:00 -bash
ubuntu   81459  0.0  0.0   2888   1920 pts/1    Ss   19:47    0:00 -sh
ubuntu   81556  0.0  0.5  21360 10880 pts/1    S+   19:47    0:00 vi test
ubuntu   92288  0.0  0.1  10068   3456 pts/0    R+   20:21    0:00 ps -u
ubuntu@m1:~$ ps -u root
```



La commande ps

➤ Voici quelques options utiles à utiliser avec **ps**

-e : Affiche tous les processus.

-f : Listing complet.

-r : Affiche uniquement les processus en cours d'exécution.

-u : Possibilité d'utiliser un nom d'utilisateur (ou plusieurs) en particulier.

-pid : Option de filtrage par PID

-ppid : Option de filtrage par PPID



La commande ps

`ps -ef` – répertorie les processus en cours d'exécution. (Une autre commande similaire est `ps aux`)

`ps -f -u user1,user2` – Affiche tous les processus basés sur un ou des UID en particulier (User ID ou nom d'utilisateur).

`ps aux --sort=-pcpu,+pmem` – Affiche les processus consommant la plus grande quantité de CPU.

`ps -e -o pid,uname,pcpu,pmem,comm` – Utilisé pour afficher certaines colonnes seulement.

`ps -e -o pid,comm,etime` – Affiche le temps depuis lequel le processus a démarré.

Nous vous recommandons de consulter la page aide « `man ps` » pour plus d'informations et l'utilisation de la commande `ps`

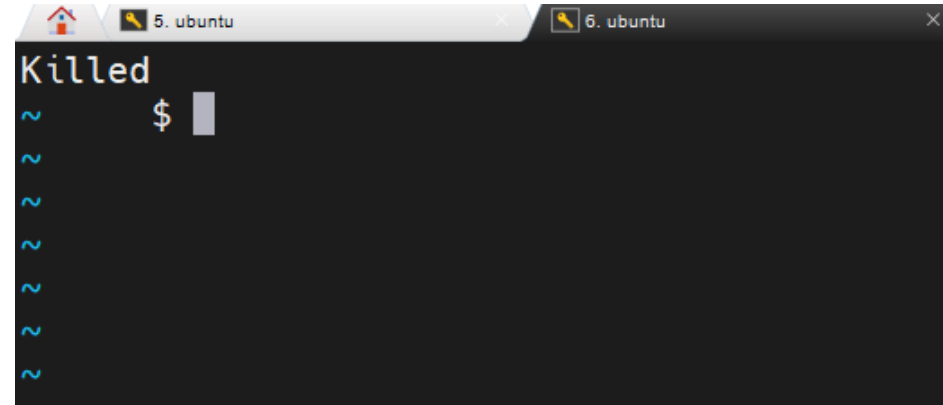


La commande kill

- La commande **kill -9** arrête un processus

```
ubuntu@m1:~$ ps -u ubuntu
  PID TTY          TIME CMD
   892 ?            00:00:00 systemd
   893 ?            00:00:00 (sd-pam)
   899 tty1        00:00:00 bash
   961 ?            00:01:03 sshd
   980 pts/0        00:00:00 bash
  1014 ?            00:00:00 sshd
  1016 ?            00:00:00 sftp-server
 81457 ?            00:00:00 sshd
 81459 pts/1        00:00:00 sh
 81506 ?            00:00:00 sshd
 81507 ?            00:00:00 sh
 81508 ?            00:00:00 sftp-server
 81556 pts/1        00:00:00 vi
 93752 ?            00:00:00 bash
 93822 ?            00:00:00 sleep
 93823 pts/0        00:00:00 ps

ubuntu@m1:~$ kill -9 81556
ubuntu@m1:~$
```



The image shows a terminal window with two tabs labeled '5. ubuntu' and '6. ubuntu'. The '6. ubuntu' tab is active and displays the word 'Killed' in a large font, followed by a prompt character '\$' and a cursor. The '5. ubuntu' tab is inactive and shows a tilde '~' character.



La commande nice

- Pour démarrer un processus et lui donner une belle valeur autre que celle par défaut, utilisez :

```
$ nice -n 5 vi
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
892	ubuntu	20	0	17080	9728	8064	S	0,0	0,5	0:00.08	systemd
893	ubuntu	20	0	169360	5576	1792	S	0,0	0,3	0:00.00	(sd-pam)
899	ubuntu	20	0	8740	5504	3840	S	0,0	0,3	0:00.02	bash
961	ubuntu	20	0	17464	8572	6016	S	0,0	0,4	1:04.39	sshd
980	ubuntu	20	0	8864	5632	3968	S	0,0	0,3	0:00.18	bash
1014	ubuntu	20	0	17312	8192	5760	S	0,0	0,4	0:00.00	sshd
1016	ubuntu	20	0	7768	5632	4608	S	0,0	0,3	0:00.00	sftp-server
95045	ubuntu	20	0	2888	1664	1664	S	0,0	0,1	0:00.00	sh
95076	ubuntu	20	0	17316	8328	5888	S	0,0	0,4	0:00.00	sshd
95077	ubuntu	20	0	2888	1664	1664	S	0,0	0,1	0:00.00	sh
95078	ubuntu	20	0	7768	5632	4608	S	0,0	0,3	0:00.00	sftp-server
95370	ubuntu	25	5	21264	10880	7168	S	0,0	0,5	0:00.02	vi
95613	ubuntu	20	0	2888	1664	1664	S	0,0	0,1	0:00.00	sh
95641	ubuntu	20	0	5768	1920	1920	S	0,0	0,1	0:00.00	sleep

- Pour modifier la valeur intéressante d'un processus déjà en cours d'exécution, utilisez :

```
renice 8 -p 96551
```

95045	ubuntu	20	0	2888	1664	1664	S	0,0	0,1	0:00.00	sh
95076	ubuntu	20	0	17316	8328	5888	S	0,0	0,4	0:00.00	sshd
95077	ubuntu	20	0	2888	1664	1664	S	0,0	0,1	0:00.00	sh
95078	ubuntu	20	0	7768	5632	4608	S	0,0	0,3	0:00.00	sftp-server
96551	ubuntu	28	8	21396	10880	7168	S	0,0	0,5	0:00.01	vi
96743	ubuntu	20	0	17464	8444	5888	S	0,0	0,4	0:00.15	sshd
96744	ubuntu	20	0	2888	1920	1792	S	0,0	0,1	0:00.00	sh

La nouvelle
valeur

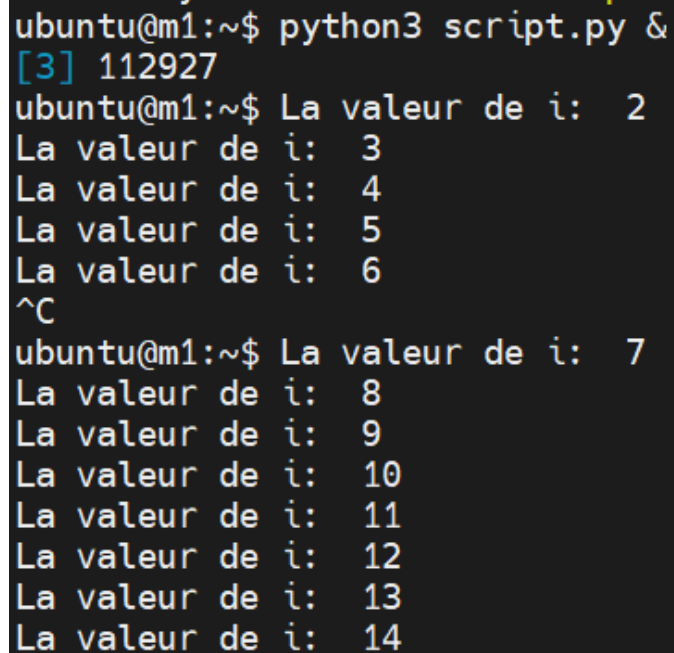
L'option

L'identifiant du process



Le contrôle des processus

- Un processus est démarré par défaut en **Foreground** c'est-à-dire apparent, Il faut utiliser Ctrl+Z pour suspendre l'invite de commande ou Ctrl+C pour l'arrêter
- Pour démarrer un processus en arrière-plan, il faut utiliser le signe « & », Ctrl+Z et Ctrl+C demeurent inactives



```
ubuntu@m1:~$ python3 script.py &  
[3] 112927  
ubuntu@m1:~$ La valeur de i: 2  
La valeur de i: 3  
La valeur de i: 4  
La valeur de i: 5  
La valeur de i: 6  
^C  
ubuntu@m1:~$ La valeur de i: 7  
La valeur de i: 8  
La valeur de i: 9  
La valeur de i: 10  
La valeur de i: 11  
La valeur de i: 12  
La valeur de i: 13  
La valeur de i: 14
```

- Un processus est arrêté avec `kill -9 pid`

Où pid est l'identifiant du processus

La gestion de services

La présentation des services

- Les services permettent de démarrer automatiquement des programmes lors du démarrage du système d'exploitation comme un serveur de base de données ou un serveur web
- Il existe deux types de services
 - Les services systèmes
 - Les services personnalisés
- La configuration des services se trouve par défaut dans le répertoire **/lib/systemd/system** (Ubuntu, Linuxmint) ou **/usr/lib/systemd/system** (centos)
- Un service est un fichier qui présente généralement une extension **.service** comme par exemple le service **network.service** responsable de la gestion du réseau

La présentation des services

C'est quoi un process?

- Un processus un programme au cour d'exécution

C'est quoi un service?

- Un service est un ou un ensemble de processus ou une application qui s'exécute en arrière-plan, soit en effectuant une tâche planifiée, soit en attendant un événement

C'est quoi un démon?

- Démon est le terme réel désignant un processus d'arrière-plan de longue durée. Un service est en fait constitué d'un ou plusieurs démons, généralement le nom d'un daemon finit avec "**d**" comme httpd, sshd ...

C'est quoi les fichiers d'unité?

- Les fichiers qui définissent la manière dont systemd gère les ressources systèmes, il y a plusieurs types d'unités, parmi les quelles les plus importantes sont *.services *.timer *.mount *.target

La gestion des états de services

- Les services permettent de démarrer automatiquement des programmes lors du démarrage du système d'exploitation comme un serveur de base de données ou un serveur web
- **sudo systemctl list-units --type service**: Lister tout les services
- **systemctl start nom_du_service**: lance le service
- **systemctl stop nom_du_service**: arrête le service
- **systemctl restart nom_du_service**: relance le service
- **systemctl reload nom_du_service**: recharge les fichiers de configuration du service sans l'arrêter
- **systemctl enable nom_du_service**: active le service
- **systemctl disable nom_du_service**: désactive le service
- **systemctl kill nom_du_service**: arrête le service
- **systemctl status nom_du_service**: inspecte le service
- **systemctl - -failed - -type=service**: énumérer les services qui représentent une anomalie
- **systemctl cat nom_du_service**: Afficher le contenu d'un service

La modification des services

- **systemctl edit nom_du_service:** Pour apporter une modification partielle à un fichier unité
- **systemctl edit - -full nom_du_service:** Pour apporter une modification intégrale à un fichier unité

Note: Après avoir modifié un fichier unité, vous devez recharger le systemd

systemctl daemon-reload

La gestion des états de services

➤ Les services sont contrôlés à travers la commande **journalctl**

➤ **journalctl -u <nom de service>**

```
[root@localhost centos]# sudo journalctl -u sshd
-- Logs begin at jeu. 2023-11-16 17:03:55 CET, end at jeu. 2023-11-16 22:20:18 CET. --
nov. 16 17:04:07 localhost.localdomain systemd[1]: Starting OpenSSH server daemon...
nov. 16 17:04:07 localhost.localdomain sshd[1052]: Server listening on 0.0.0.0 port 22.
nov. 16 17:04:07 localhost.localdomain sshd[1052]: Server listening on :: port 22.
nov. 16 17:04:07 localhost.localdomain systemd[1]: Started OpenSSH server daemon.
nov. 16 17:04:43 localhost.localdomain sshd[1572]: Accepted password for centos from 192.168.56.1 port 59399 ssh2
nov. 16 17:04:43 localhost.localdomain sshd[1576]: Accepted password for centos from 192.168.56.1 port 59400 ssh2
```

➤ **journalctl -n 10**: Utiliser l'argument **-n** de la commande **journalctl** pour afficher uniquement les N derniers nombres d'entrées

Le service sshd

```
[root@localhost centos]# sudo journalctl -n 10 -u sshd
-- Logs begin at jeu. 2023-11-16 17:03:55 CET, end at jeu. 2023-11-16 22:23:51 CET. --
nov. 16 17:04:07 localhost.localdomain systemd[1]: Starting OpenSSH server daemon...
nov. 16 17:04:07 localhost.localdomain sshd[1052]: Server listening on 0.0.0.0 port 22.
nov. 16 17:04:07 localhost.localdomain sshd[1052]: Server listening on :: port 22.
nov. 16 17:04:07 localhost.localdomain systemd[1]: Started OpenSSH server daemon.
nov. 16 17:04:43 localhost.localdomain sshd[1572]: Accepted password for centos from 192.168.56.1 port 59399 ssh2
nov. 16 17:04:43 localhost.localdomain sshd[1576]: Accepted password for centos from 192.168.56.1 port 59400 ssh2
```

Tout les services

```
[root@localhost centos]# sudo journalctl -n 10
-- Logs begin at jeu. 2023-11-16 17:03:55 CET, end at jeu. 2023-11-16 22:21:51 CET. --
nov. 16 22:20:00 localhost.localdomain systemd[1]: Starting Network Manager Script Dispatcher Service...
nov. 16 22:20:00 localhost.localdomain dbus[693]: [system] Successfully activated service 'org.freedesktop.nm_dispatcher'
nov. 16 22:20:00 localhost.localdomain systemd[1]: Started Network Manager Script Dispatcher Service.
nov. 16 22:20:00 localhost.localdomain nm-dispatcher[14074]: req:1 'dhcp4-change' [enp0s8]: new request (3 scripts)
nov. 16 22:20:00 localhost.localdomain nm-dispatcher[14074]: req:1 'dhcp4-change' [enp0s8]: start running ordered scripts...
```

La gestion des états de services

- **journalctl -f -u sshd**: Surveiller les journaux d'un service, c'est-à-dire continuer à lire les journaux en temps réel avec l'option *-f*

```
[root@localhost centos]# journalctl -f -u sshd
-- Logs begin at jeu. 2023-11-16 17:03:55 CET. --
nov. 16 17:04:07 localhost.localdomain systemd[1]: Starting OpenSSH server daemon...
nov. 16 17:04:07 localhost.localdomain sshd[1052]: Server listening on 0.0.0.0 port 22.
nov. 16 17:04:07 localhost.localdomain sshd[1052]: Server listening on :: port 22.
nov. 16 17:04:07 localhost.localdomain systemd[1]: Started OpenSSH server daemon.
nov. 16 17:04:43 localhost.localdomain sshd[1572]: Accepted password for centos from 192.168.56.1 port 59399 ssh2
nov. 16 17:04:43 localhost.localdomain sshd[1576]: Accepted password for centos from 192.168.56.1 port 59400 ssh2
```

- **journalctl - -since yesterday | "2023-11-10 14:00:00"**: Surveiller les journaux d'un service, c'est-à-dire continuer à lire les journaux en temps réel avec l'option *-f*

```
[root@localhost centos]# journalctl --since yesterday
-- Logs begin at jeu. 2023-11-16 17:03:55 CET, end at jeu. 2023-11-16 22:29:36 CET. --
nov. 16 17:03:55 localhost.localdomain systemd-journal[96]: Runtime journal is using 8.0M (max allowed 91.8M, trying to
nov. 16 17:03:55 localhost.localdomain kernel: Initializing cgroup subsys cpuset
nov. 16 17:03:55 localhost.localdomain kernel: Initializing cgroup subsys cpu
nov. 16 17:03:55 localhost.localdomain kernel: Initializing cgroup subsys cpuacct
nov. 16 17:03:55 localhost.localdomain kernel: Linux version 3.10.0-1160.102.1.el7.x86_64 (mockbuild@kbuilder.bsys.cent
nov. 16 17:03:55 localhost.localdomain kernel: Command line: BOOT_IMAGE=/vmlinuz-3.10.0-1160.102.1.el7.x86_64 root=/dev
nov. 16 17:03:55 localhost.localdomain kernel: e820: BIOS-provided physical RAM map:
```

Les services personnalisés

- Il existe plusieurs façons d'exécuter votre programme en tant que service, même si le serveur redémarre pour une raison quelconque, le script s'exécutera en arrière-plan malgré tout
- Vérifier que le module **systemd** existe avec `systemd --version` sinon il faut l'installer avec **apt-get install (ubuntu)** ou **yum install (centos)**
- Créer un fichier avec l'extension `service` dans `/etc/systemd/system/nom_de_service.service`

```
[Unit]
Description=My test service
After=multi-user.target
[Service]
Type=simple
Restart=always
ExecStart=/usr/bin/python3    Chemin du script ici
[Install]
WantedBy=multi-user.target
```


Les services personnalisés

- Relancer le daemon **sudo systemctl daemon-reload**
- Activer le service **sudo systemctl enable nom_de_service.service**

Les types de services

simple: Pour les executables sans demonization

forking: Pour les executables avec demonization

oneshot: Pour les exécutable à courte durée de vie

notify: Pour les exécutable qui notifient systemd lors de leur démarrage

Les priorités de services par ordre

/etc/systemd/system

/run/systemd/system

/lib/systemd/system

Les services personnalisés

- Un exemple de service commencer par créer le script :

```
#!/bin/bash
while true
do
    echo La date est $(date)
    sleep 1
done
```

- Donner le droit d'exécution avec chmod au script
- Créer un fichier qui porte l'extension .service sous /etc/systemd/system

```
[Service]
ExecStart=/scripts/montimer.sh
```

- Lancer les commandes
systemctl daemon-reload
systemctl start <nom du service>
- Inspecter le service avec la commande **systemctl status <nom du service>**

La présentation des UNIT

C'est quoi un target?

- Dans systemd, une unité cible est un concept utilisé pour regrouper et définir un ensemble de services ou d'unités qui doivent être démarrés ou arrêtés ensemble
- Une unité cible dans systemd est représentée par un fichier « cible », se terminant généralement par .target
- Par exemple **multi-user.target**, qui peut inclure les services nécessaires à un environnement multi-utilisateurs, tels que la mise en réseau, la connexion et d'autres services essentiels
- Afficher le contenu d'une cible avec **systemctl cat nom_du_target**

La présentation des UNIT

Exemple d'un target?

[Unit]

Description=Foofoo boot target

Requires=multi-user.target

Wants=foofoo.service

Conflicts=rescue.service rescue.target

After=multi-user.target rescue.service rescue.target

[Install]

WantedBy=default.target

Pour expliquer les options:

Description: Décrit la cible.

Requires: Les dépendances matérielles de la cible.

Wants: Dépendances logicielles. La cible n'exige pas que ceux-ci démarrent.

Conflicts: Si une unité peut avoir un conflit avec une autre unité, le démarrage de la première arrêtera la seconde et vice versa.

After: Lancement après les services mentionnés

La présentation des UNIT

C'est quoi un timer?

- C'est une unité de minuterie est utilisée pour configurer et contrôler l'exécution d'autres unités à des intervalles spécifiques ou à certaines heures du calendrier, elle est représentée par un fichier *.**timer**
- Un fichier d'unité de minuterie contient des options de configuration spécifiant quand et à quelle fréquence l'unité associée doit être activée. Il comprend des paramètres tels que:

OnBootSec: Le délai d'attente après le démarrage

OnUnitActiveSec: Le délai d'attente après la dernière activation de l'unité

OnCalendar: La spécification d'événements de calendrier spécifiques

La présentation des UNIT

Exemple d'un timer?

- Il faut commencer par créer un service sous **/etc/systemd/system** dont le nom est **ls-service.service**

[Unit]

Description=Tester un service avec timer

Wants=myMonitor.timer

[Service]

Type=oneshot  Le service s'éteint après l'exécution

ExecStart=/usr/bin/ls

[Install]

WantedBy=multi-user.target

La présentation des UNIT

Exemple d'un timer?

- Ensuite créer un timer sous `/etc/systemd/system` dont le nom est **myMonitor.timer**

[Unit]

Description=Exemple de timer

Requires=ls-service.service

[Timer]

Unit=ls-service.service

OnCalendar=*-*-* *: *:00  Déclenchement toutes les minutes

[Install]

WantedBy=timers.target

La présentation des UNIT

Exemple d'un timer?

- Ensuite il faut recharger le systemd via **systemctl daemon-reload**
- Activer le service **systemctl enable ls-service.service**
- Activer le timer **systemctl enable myMonitor.timer**
- Lancer le service **systemctl start ls-service.service**
- Vérifier le status du service **systemctl status ls-service.service**

- Vérifier que le service est lancé chaque minute **journalctl -S today -f -u ls-service.service**

La présentation des UNIT

➤ Exemples d'horodatages valides et leur forme normalisée :

Minutieusement → *-*-* *:*:00

Horaire → *-*-* *:00:00

Quotidien → *-*-* 00:00:00

Mensuel → *-*-01 00:00:00

Hebdomadaire tout les Lundis → Mon *-*-* 00:00:00

Annuel → *-01-01 00:00:00

Trimestriel → *-01,04,07,10-01 00:00:00

Semestriel → *-01,07-01 00:00:00

➤ Autres exemples :

Sat,Thu,Mon..Wed,Sat..Sun → Mon..Thu,Sat,Sun *-*-* 00:00:00

Mon,Sun 12-*-* 2,1:23 → Mon,Sun 2012-*-* 01,02:23:00

Wed *-1 → Wed *-*-01 00:00:00

Wed..Wed,Wed *-1 → Wed *-*-01 00:00:00

Wed, 17:48 → Wed *-*-* 17:48:00

La présentation des UNIT

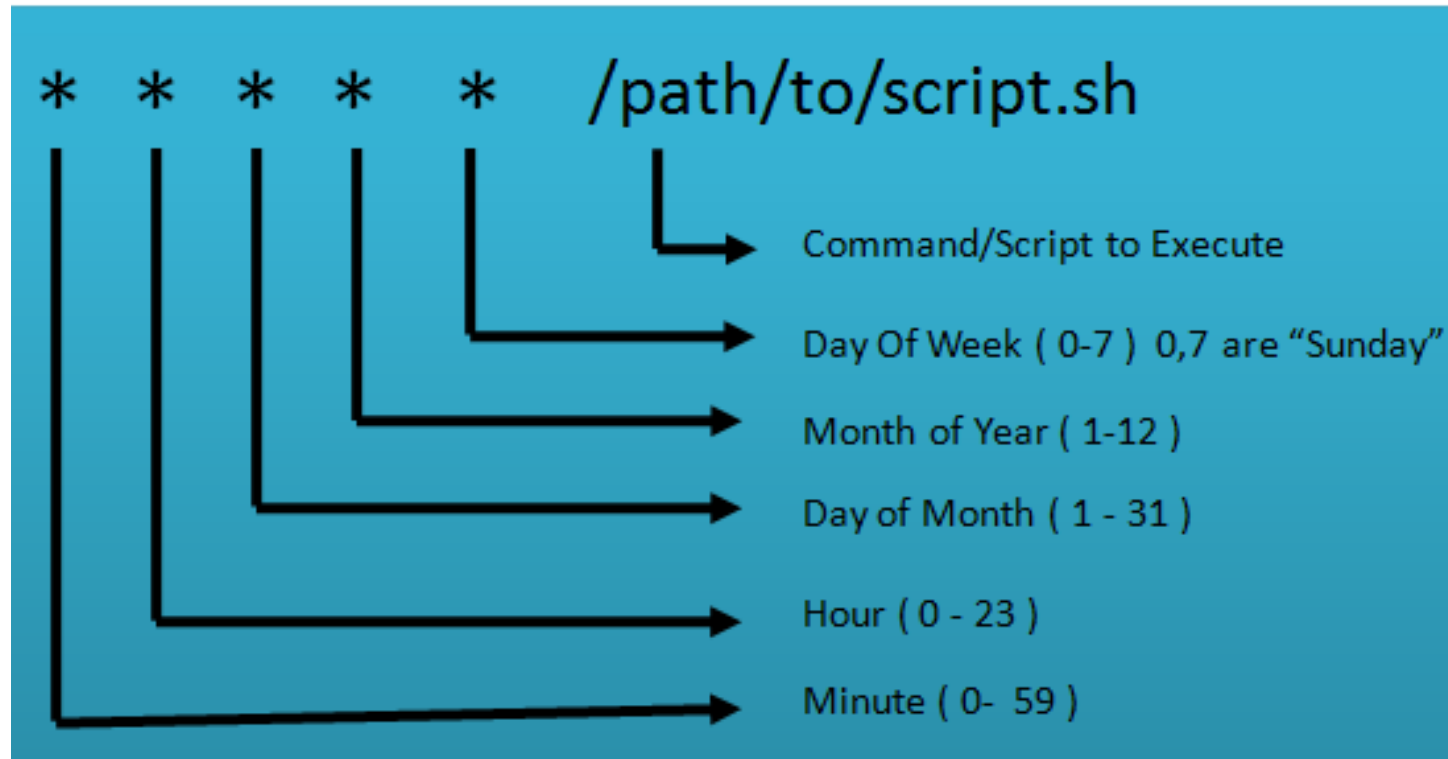
➤ Tableau qui montre des exemples de configuration de Calendar

Spécification des événements du calendrier	Description
--* 00:15:30	Chaque jour de chaque mois de chaque année à 15 minutes et 30 secondes après minuit
Hebdomadaire	Tous les lundis à 00:00:00
--* 00:00:00	Lundi chaque semaine à minuit
Mon	Lundi chaque semaine à minuit
Wed 2020-*-*	Tous les mercredis de l'année 2020 à 00:00:00
Mon..Fri 2021-*-*	Tous les jours de la semaine en 2021 à 00:00:00
2022-6,7,8-1,15 01:15:00	Les 1er et 15 juin, juillet et août 2022 à 01h15
Mon *-05~03	Occurrence suivante d'un lundi de mai d'une année qui est également le 3ème jour à compter de la fin du mois.
Mon..Fri *-08~04	Le 4ème jour précédant la fin août pour les années où il tombe également un jour de semaine.
*-05~03/2	Le 3ème jour à partir de la fin du mois de mai puis de nouveau deux jours plus tard. Se répète chaque année. Notez que cette expression utilise le Tilde (~).
*-05-03/2	Le troisième jour du mois de mai puis tous les 2 jours pour le reste du mois de mai. Se répète chaque année. Notez que cette expression utilise le tiret (-).

Le CRON

Qu'est-ce que CRON ?

- CRON est un planificateur de tâches basé sur le temps dans les systèmes d'exploitation de type Unix, notamment Linux. Il permet aux utilisateurs de planifier et d'automatiser l'exécution de commandes, de scripts et de programmes



CRON vs Service

- Les services sont intégrés au système d'initialisation du système, tel que systemd, , garantissant qu'ils démarrent automatiquement et peuvent être contrôlés à l'aide de commandes à l'échelle du système telles que service ou **systemctl**
- Cron est basé sur le temps, Les services sont basés sur des événements et répondent aux événements du système ou aux demandes des utilisateurs
- Les tâches Cron sont exécutées dans le contexte de l'utilisateur qui les a créées, tandis que les services s'exécutent généralement en tant que service au niveau du système

Comprendre la syntaxe CRON

- La commande **crontab** est l'outil de ligne de commande utilisé pour gérer les tâches CRON.
Pour ouvrir le fichier cron qui porte le nom de l'utilisateur sous **/var/spool/cron/crontabs** pour le modifier

```
root@m1:/tmp# cd /var/spool/cron/crontabs
root@m1:/var/spool/cron/crontabs# ls
root  ubuntu
```

- La commande **c** a des options
 - -e: éditer le fichier cron
 - -l: lister les cron jobs lancés
 - -r: supprimer le fichier cron
 - -i: pauser la question avant de supprimer
- Le root et l'utilisateur courant sont par défaut les uniques executeurs du cron

```
root@m1:/var/spool/cron/crontabs# ls
root  ubuntu
```

- La commande **crontab -u <nom de l'utilisateur> -e** ajoute l'utilisateur au club du cron exemple **crontab -u ubuntu2 -e**

```
root@m1:/var/spool/cron/crontabs# ls
root  ubuntu  ubuntu2
```


Comprendre la syntaxe CRON

- Chaque tâche CRON se compose de cinq composants :
 - Minute : la minute de l'heure (0-59).
 - Heure : L'heure de la journée (0-23).
 - Jour du mois : Le jour du mois (1-31).
 - Mois : le mois de l'année (1-12).
 - Jour de la semaine : le jour de la semaine (0-7, où 0 et 7 représentent le dimanche)

- Les symboles suivants ont des significations particulières dans la syntaxe CRON :
 - L'astérisque (*) représente toutes les périodes * * * * *
 - La virgule (,) permet de spécifier une liste discrète de valeurs
 - Le trait d'union (-) spécifie une intervalle de valeurs.
 - La barre oblique (/) spécifie un pas de période */5 * * * *
 - La (NL) où N est un nombre entier entre 0 et 7 spécifique au jour de la semaine 2L -> chaque Mardi
 - Le hash (#) spécifie le jour de la semaine

Le réseau

Le rappel de l'adressage

Les adresses IPV4

- Sont composés de segments
- Représentent deux parties
 - Partie réseau
 - Partie sous réseau (optionnelle en cas de sous réseau)
 - Partie hôte
- Classification des adresses IP
 - Classe A 1.0.0.0 – 126.255.255.255 masque sous réseau par défaut 255.0.0.0
 - LoopBack 127.0.0.0 – 127.255.255.255 masque sous réseau par défaut 255.0.0.0
 - Classe B 128.0.0.0 – 191.255.255.255 masque sous réseau par défaut 255.255.0.0
 - Classe C 192.0.0.0 – 223.255.255.255 masque sous réseau par défaut 255.255.255.0
 - Classe D 224.0.0.0 – 239.255.255.255 La classe D est réservée au multicasting. En multidiffusion, les données ne sont pas destinées à un hôte particulier, c'est pourquoi il n'est pas nécessaire d'extraire l'adresse de l'hôte de l'adresse IP et la classe D n'a pas de masque de sous-réseau

Les adresses IPV4

- Exemple1 d' adressage et sous réseau Classe A. Soit le réseau 10.0.0.0, il est demandé de le décomposer en deux réseaux

Partie réseaux	Partie hôte	Partie hôte	Partie hôte
10	0	0	0
00001010	00000000	00000000	00000000

Partie réseaux	Partie sous réseau	Partie hôte	Partie hôte	Partie hôte
00001010	0	0000000	00000000	00000000
00001010	1	0000000	00000000	00000000

Les adresses IPV4

Partie réseaux	Partie sous réseau	Partie hôte	Partie hôte	Partie hôte
00001010	0	0000000	00000000	00000000
00001010	1	0000000	00000000	00000000

Partie réseaux	Partie sous réseau	Partie hôte	Partie hôte	Partie hôte
10	0	0	0	0
10	128	0	0	0

Le réseaux	L'adresse réseau	L'adresse Mask	
10.0.0.0	10.0.0.0	10.127.255.255	
10.128.0.0	10.128.0.0	10.128.255.255	

Les adresses IPV4

- Exemple2 d' adressage et sous réseau Classe B 172.16.0.0. Soit le réseau 10.0.0.0, il est demandé de le décomposer en quatre réseaux

Partie réseaux	Partie réseaux	Partie hôte	Partie hôte
172	16	0	0
00001010	00010000	00000000	00000000

Partie réseaux	Partie réseau	Partie sous réseau	Partie hôte	Partie hôte
172	16	00000000 = 0	0	0
172	16	01000000 = 64	00000000	00000000
172	16	11000000 = 128	00000000	00000000
172	16	11000000 = 192	00000000	00000000

Les adresses IPV4

➤ Exemple2 d' adressage et sous réseau Classe B 172.16.0.0. Soit le réseau 10.0.0.0, il est demandé de le décomposer en quatre réseaux

Partie réseaux	Partie réseau	Partie sous réseau	Partie hôte	Partie hôte
172	16	00000000 = 0	0	0
172	16	01000000 = 64	00000000	00000000
172	16	11000000 = 128	00000000	00000000
172	16	11000000 = 192	00000000	00000000

Le réseaux	L'adresse réseau	L'adresse Mask	
172.16.0.0	172.16.0.0	172.16.63.255	
172.16.64.0	172.16.64.0	172.16.127.255	
172.16.128.0	172.16.128.0	172.16.191.255	
172.16.192.0	172.16.192.0	172.16.192.255	

Les adresses et VLSM

- Les fournisseurs de services Internet peuvent être confrontés à une situation dans laquelle ils doivent allouer des sous-réseaux IP de différentes tailles selon les besoins du client et pas conformément aux standards
- Pour un FAI, il n'est pas possible de diviser les adresses IP en sous-réseaux de taille fixe ; il peut plutôt souhaiter diviser les sous-réseaux en sous-réseaux de manière à minimiser le gaspillage d'adresses IP
- Soit l'exemple du réseau 192.168.1.0/24 que nous divisons en

Département	Nombre
Ventes	100
Production	50
Finance	25
Gestion	5

Les adresses et VLSM

- Allouons la plage d'adresses IP la plus élevée aux exigences les plus élevées, attribuons donc 192.168.1.0 /25 (255.255.255.128) au service Ventes
- Ce sous-réseau IP avec le numéro de réseau 192.168.1.0 possède 126 adresses IP d'hôte valides
- 1: Attribuer la plage suivante la plus élevée, attribuons donc 192.168.1.128 /26 (255.255.255.192) au service des Ventes qui contient 126 adresses
- 2: Attribuer la plage suivante la plus élevée, attribuons donc 192.168.1.128 /27 (255.255.255.224) au service Production qui contient 62 adresses
- 3: Attribuer la plage suivante la plus élevée, attribuons donc 192.168.1.128 /28 (255.255.255.248) au service Finance qui contient 25 adresses
- 4: Attribuer la plage suivante la plus élevée, attribuons donc 192.168.1.128 /29 (255.255.255.248) au service Gestion qui contient 6 adresses

Les adresses IPV6

- IPv6 est une nouvelle version du protocole Internet qui remplacera à terme IPv4
- IPv6 utilise des adresses de 128 bits par opposition aux adresses de 32 bits utilisées par IPv4, cela permet théoriquement 2^{128} combinaisons ou 340 000 milliards d'adresses
- Pour concevoir un sous-réseau plus grand ou plus petit, il suffit d'ajuster le préfixe par multiple de quatre

Préfixe	Exemple de sous-réseau	Adresses IP totales	# de /64 moustiquaires
4	X::	2^{124}	2^{60}
8	xx::	2^{120}	2^{56}
12	xxx::	2^{116}	2^{52}
16	xxxx::	2^{112}	2^{48}
20	xxxx:x::	2^{108}	2^{44}
124	xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxx::	2^4 (16)	0
128	xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx	2^0 (1)	0

Les adresses IPV6

➤ Les sous réseaux spéciaux de IPv6

Réseau	But
2001:db8::/32	Préfixe de documentation utilisé pour les exemples
::1	Hôte local
fc00::/7	Adresses locales uniques (ULA) - également connues sous le nom d'adresses IPv6 « privées ».
fe80::/10	Lier des adresses locales, valables uniquement à l'intérieur d'un seul domaine de diffusion.
ff00 : 0/8	Adresses de multidiffusion

Configuration des interfaces réseaux

La configuration des interfaces réseau (La commande nmcli)

- Commencer par découvrir les interfaces via la commande **nmcli d**

```
[root@localhost network-scripts]# nmcli d
```

DEVICE	TYPE	STATE	CONNECTION
enp0s9	ethernet	connecté	Connexion filaire 2
enp0s8	ethernet	connecté	Connexion filaire 1
enp0s3	ethernet	déconnecté	--
lo	loopback	non-géré	--

- Pour configurer une interface particulière il faut la repérer dans le dossier **/etc/sysconf ig/network-scripts**

```
[root@localhost network-scripts]# cd /etc/sysconfig/network-scripts
```

```
[root@localhost network-scripts]# ls
```

ifcfg-enp0s3	ifdown-ppp	ifup-eth	ifup-sit
ifcfg-lo	ifdown-routes	ifup-ipp	ifup-Team
ifdown	ifdown-sit	ifup-ipv6	ifup-TeamPort
ifdown-bnep	ifdown-Team	ifup-isdn	ifup-tunnel
ifdown-eth	ifdown-TeamPort	ifup-plip	ifup-wireless
ifdown-ipp	ifdown-tunnel	ifup-plusb	init.ipv6-global
ifdown-ipv6	ifup	ifup-post	network-functions
ifdown-isdn	ifup-aliases	ifup-ppp	network-functions-ipv6
ifdown-post	ifup-bnep	ifup-routes	

La configuration des interfaces réseau (La commande nmcli)

- Le contenu du fichier particulier contient des attributs parmi les quels il y a les suivants

```
BOOTPROTO= ... IPADDR=... NETMASK=...  
GATEWAY=... DNS1=... DNS2=.
```

- Après modification il faut relancer le service réseau
systemctl restart
- Pour montrer les informations d'une carte particulière exemple
nmcli dev show enp0s3

```
[root@localhost network-scripts]# nmcli dev show enp0s3  
GENERAL.DEVICE:                enp0s3  
GENERAL.TYPE:                  ethernet  
GENERAL.HWADDR:                08:00:27:D2:C9:A4  
GENERAL.MTU:                   1500  
GENERAL.STATE:                 30 (déconnecté)  
GENERAL.CONNECTION:            --  
GENERAL.CON-PATH:              --  
WIRED-PROPERTIES.CARRIER:     marche
```

La configuration des interfaces réseau (La commande nmcli)

- Il est possible de configurer l'interface réseau pour avoir soit une adresse IP statique soit récupérer une adresse du DHCP

```
TYPE="Ethernet"
PROXY_METHOD="none"
BROWSER_ONLY="no"
BOOTPROTO="static"
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_FAILURE_FATAL="no"
IPV6_ADDR_GEN_MODE="stable-privacy"
NAME="enp0s3"
UUID="1ea6bded-eb13-4e48-af4e-47a72e9794e8"
DEVICE="enp0s3"
ONBOOT="yes"
IPADDR=123.456.7.890
NETMASK=123.456.789.0
GATEWAY=123.456.7.8
DNS1=8.8.8.8
DNS2=8.8.4.4
```

Attribution d'adresse statique

```
TYPE="Ethernet"
PROXY_METHOD="none"
BROWSER_ONLY="no"
BOOTPROTO="dhcp"
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_FAILURE_FATAL="no"
IPV6_ADDR_GEN_MODE="stable-privacy"
NAME="enp0s3"
UUID="1ea6bded-eb13-4e48-af4e-47a72e9794e8"
DEVICE="enp0s3"
ONBOOT="yes"
IPADDR=123.456.7.890
NETMASK=123.456.789.0
GATEWAY=123.456.7.8
DNS1=8.8.8.8
DNS2=8.8.4.4
```

Attribution d'adresse récupérée du DHCP

La configuration des interfaces réseau (La commande nmcli)

- Il est possible de configurer l'interface réseau sans passer par le fichier de configuration

- Pour changer l'adresse IP

```
nmcli connection modify enp0s3 ipv4.addresses 10.0.2.15
```

- Pour changer la passerelle par défaut

```
nmcli connection modify enp0s3 ipv4.gateway 10.0.2.1
```

- Pour configurer le DNS

```
nmcli connection modify enp0s3 ipv4.dns 8.8.8.8
```

```
nmcli connection modify enp0s3 ipv4.dns "8.8.8.8 8.8.4.4"
```

- Pour configurer l'attribution de IP quelle soit statique

```
nmcli connection modify enp0s3 ipv4.method manual
```

- Pour configurer l'attribution de IP quelle soit attribuée via DHCP

```
nmcli connection modify enp0s3 ipv4.method auto
```

La configuration des interfaces réseau (La commande nmcli)

- Il est possible de configurer l'interface réseau sans passer par le fichier de configuration

- Pour changer l'adresse IP

```
nmcli connection modify enp0s3 ipv4.addresses 10.0.2.15
```

- Pour changer la passerelle par défaut

```
nmcli connection modify enp0s3 ipv4.gateway 10.0.2.1
```

- Pour configurer le DNS

```
nmcli connection modify enp0s3 ipv4.dns 8.8.8.8
```

```
nmcli connection modify enp0s3 ipv4.dns "8.8.8.8 8.8.4.4"
```

- Pour configurer l'attribution de IP quelle soit statique

```
nmcli connection modify enp0s3 ipv4.method manual
```

- Pour configurer l'attribution de IP quelle soit attribuée via DHCP

```
nmcli connection modify enp0s3 ipv4.method auto
```

La configuration des interfaces réseau (La commande ifconfig)

- La configuration de l'adresse IP et du masque de réseau de n'importe quelle interface réseau et la désactivation ou l'activation de n'importe quelle interface sont quelques utilisations de ifconfig
- Pour voir l'état de toutes les interfaces réseau actives
ifconfig
- Pour répertorier toutes les interfaces actuellement disponibles
ifconfig -a
- Pour attribuer une adresse IP à une interface
ifconfig eth0 195.167.54.6 netmask 255.255.255.0
- Pour activer une interface
ifconfig up eth0
- Pour désactiver une interface
ifconfig down eth0

La résolution des noms DNS

Le fichier hosts

- Le fichier hosts donne un moyen d'assurer la résolution de noms, de donner un nom FQDN à un hôte

```
ubuntu@m1:/$ cat /etc/hosts
127.0.0.1 localhost
127.0.1.1 m1

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```



Le fichier networks

- Le fichier networks donne un moyen qui permet d'affecter un nom logique à un réseau

```
ubuntu@m1:/$ cat /etc/networks  
# symbolic names for networks, see networks(5) for more information  
link-local 169.254.0.0
```



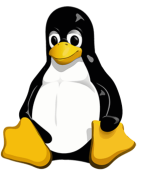
Les commandes

- **ifconfig**: afficher les informations sur les cartes réseaux
- **Route**: montre la table de routage
- **ip route**: montre la table de routage si route n'est pas installé
- **ip a**: avec l'option -4 pour afficher les ipv4 et -6 pour afficher les ipv6
- **ip a show nom_de_l'interface**: montrer les informations concernant l'interface réseau
- **hostname -I**: montrer l'adresse IP
- **ip a add @Ip/masque dev nom_de_l'interface**: affecter une adresse à une carte réseau
- **ip link set dev ens3 down/up**: pour activer/désactiver une interface réseau
- **traceroute**: montre l'itinéraire vers une adresse web
- **nslookup**: permet de montrer @IP à partir du nom de domaine
- **netstat**: permet d'afficher des informations sur les ports
- **ip address show dev enp0s3**: afficher les information sur la carte réseau



Les commandes

- **nmap**: afficher les ports occupés sur un serveur en relation avec une destination particulière exemple **nmap google.com**



Le pare feu

Le pare feu sous Linux (Ubuntu)

- Pour installer UFW si vous ne l'avez pas déjà, exécutez
`sudo apt update && sudo apt install ufw -y`
- UFW est désactivé par défaut dans Ubuntu et vous devez l'activer
`sudo ufw enable`
- Le fichier de configuration du pare feu :
`/etc/ufw/ufw.conf`
- Il est possible de changer l'état du pare feu à partir du fichier de configuration
`ENABLED=yes|no`
- Afficher l'état du pare feu
`sudo ufw status`
- Si vous souhaitez que la sortie soit numérotée
`sudo ufw status numbered`



Le pare feu sous Linux(Ubuntu)

- refuser tout par défaut
ufw default deny
- Permettre tout par défaut
ufw default allow
- Refuser la connexion à travers un port
ufw deny numero|protocole
- Permettre la connexion à travers un port
ufw allow numero/protocole
- Refuse la connexion à partir d'une adresse IP
ufw deny @IP

Note: Pour la règle **deny** il faut pas oublier de lancer la commande **ufw reload**



Le pare feu sous Linux(Ubuntu)

- permettre la connexion à partir d'une adresse IP
`ufw allow @IP`
- refuse la connexion à partir d'une plage d'adresses IP
`ufw deny @Reseau/masque`
- permettre la connexion à partir d'une plage d'adresses IP
`ufw allow @Reseau/masque`
- permettre la connexion à partir d'une plage d'adresses IP sur un port donné
`sudo ufw allow from @IP/masque to any port PORT`
- Refuser la connexion à partir d'une plage d'adresses IP sur un port donné
`sudo ufw deny from @IP/masque to any port PORT`



Le pare feu sous Linux(Centos)

- Vérifier l'état du pare-feu
`sudo systemctl status firewalld`

- Activer le parefeu
`sudo systemctl enable firewalld`

- Pour énumérer les zones
`sudo firewall-cmd --get-zones`

```
[centos@localhost ~]$ sudo firewall-cmd --get-zones  
block dmz drop external home internal public trusted work
```

- Pour avoir la zone par défaut
`sudo firewall-cmd --get-default-zone`

- Pour vérifier quelle zone est active
`sudo firewall-cmd --get-active-zones`



Le pare feu sous Linux(Centos)

- Lister les règles de la zone par défaut
`sudo firewall-cmd --list-all`
- Pour lister les règles d'une zone donnée
`sudo firewall-cmd --zone=work --list-all`
- Changer la zone de pare-feu par défaut
`sudo firewall-cmd --set-default-zone=work`
- Pour changer la zone d'une interface
`sudo firewall-cmd --zone=work --change-interface=eth0ens3`



IP tables

Le module IPTABLES

- **iptables** est le programme de ligne de commande de l'espace utilisateur utilisé pour configurer l'ensemble de règles de filtrage de paquets Linux

Comment ça fonctionne:

IP tables filtre les paquets en fonction de :

Tables : les tables sont des fichiers composée de plusieurs **chaînes**

Chaînes : Une chaîne est un ensemble de **règles** . Lorsqu'un paquet est reçu, iptables trouve la table appropriée, puis l'exécute dans la chaîne de **règles** jusqu'à ce qu'il trouve une correspondance

Règles : une règle est une instruction qui indique au système quoi faire avec un paquet. Les règles peuvent bloquer un type de paquet ou le transférer. La destination vers la quelle un paquet est envoyé est appelée une **cible**

Cibles : une cible est la destination du packet



Le module IPTABLES

- Pour vérifier si **iptables** est installée
iptables

```
root@m1:~# iptables
iptables v1.8.7 (nf_tables): no command specified
```

- Pour afficher l'état des chaines
iptables -L
iptables -L --line-numbers

- Le système affiche l'état de vos chaînes. La sortie listera trois chaînes :
Chain INPUT (policy ACCEPT)
Chain FORWARD (policy ACCEPT)
Chain OUTPUT (policy ACCEPT)

- Pour autoriser le trafic Web HTTP
`sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT`

- Pour autoriser le trafic SSH
`sudo iptables -A INPUT -p tcp --dport 22 -j ACCEP`

- Pour autoriser le trafic Htps
`sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT`



Le module IPTABLES

- Utiliser la commande suivante pour accpter|refuser le trafic provenant d'une adresse IP spécifique
`sudo iptables -A INPUT -s 192.168.1.14 -j ACCEPT | DROP`
- Utiliser la commande suivante pour accpter|refuser le trafic provenant d'une rangée d'IP spécifiques
`sudo iptables -A INPUT -m iprange --src-range 192.168.1.1-192.168.1.255 -j REJECT`

Iptables ne conserve pas les règles que créées au redémarrage du système. Pour enregistrer les règles dans les systèmes basés

Sur Debian:

`/sbin/iptables-save`

Sur Centos

`service iptables save`



Des règles de sécurité générales en Linux

Les règles de sécurité

- **Crypter la communication de données pour le serveur Linux:** Toutes les données transmises sur un réseau sont ouvertes à la surveillance. **Chiffrez les données transmises dans la mesure du possible** avec un mot de passe ou à l'aide de clés / certificats
- Évitez d'utiliser les services **FTP** et **Telnet**
- Minimiser les logiciels pour minimiser la vulnérabilité sous Linux
- Garder le noyau Linux et les logiciels à jour
- Comptes d'utilisateur Linux et politique de mot de passe fort
- Configurer l'expiration du mot de passe pour les utilisateurs Linux pour une meilleure sécurité avec la commande **chage**
Note: Pour obtenir les informations d'expiration du mot de passe, entrer **chage -l utilisateur**
- Restreindre l'utilisation des mots de passe précédents sous Linux
- Verrouillage des comptes d'utilisateurs après des échecs de connexion avec **faillog**
Note: Pour voir toutes les tentatives échouées d'un utilisateur **faillog -r -u utilisateur**
- Bloquer les comptes inactif ou douteux avec **passwd -l utilisateur**
Note: Pour débloquer **passwd -u utilisateur**



Les règles de sécurité

- Eviter les comptes avec mot de passes vides
- Utilisez sudo plutôt que de se connecter en tant que root pour effectuer des tâches qui demandent des privilèges élevés
- Sécurité du serveur physique
- Désactiver les services Linux indésirables
- Verrouiller les ports non utilisés
- Renforcer le noyau Linux à travers l'ajout des règles de démarrage `/etc/sysctl.conf`
- Etablir un quota de disque bien défini
- Désactivez IPv6 uniquement si elle ne sont pas utilisées sous Linux
- Installer et utiliser un système de détection d'intrusion
- Désactiver les périphériques USB/firewire/thunderbolt non utilisés
- Utiliser des serveur web sécurisés Apache/PHP/Nginx
- Faire toujours des sauvegardes



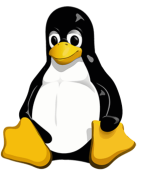
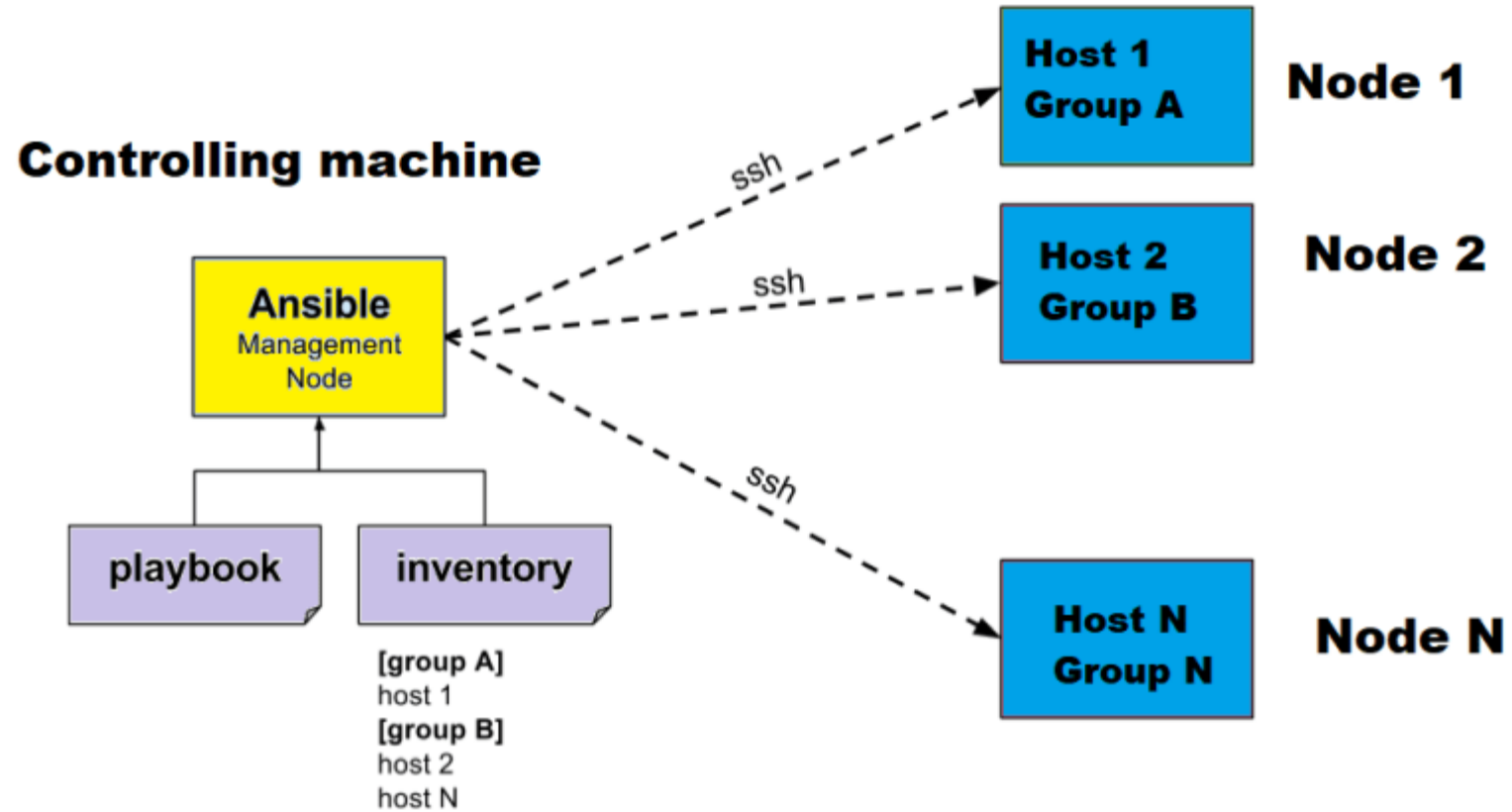
Ansible

PourquoiAnsible ?

- C'est une application open source gratuite
- Sans agent – Pas besoin d'installation et de gestion d'agent
- Basé sur Python / Yaml
- Gestion très flexible et de la configuration des systèmes
- Grand nombre de modules prêts à l'emploi pour la gestion du système
- Des modules personnalisés peuvent être ajoutés si nécessaire
- Restauration de la configuration en cas d'erreur
- Simple et lisible par l'homme
- Auto-documentation



Architecture de Ansible



Installation de Ansible

➤ Pré installation côté serveur

`adduser ansadmin`

`ssh-keygen -t rsa`

`ssh-copy-id -i /home/ansadmin/.ssh/id_rsa.pub ansadmin@IP`

➤ Pré installation côté client

`adduser ansadmin`

`/etc/sudoers <= ansadmin ALL=(ALL) NOPASSWD:ALL`

Note: Vérifier le paramètre **PasswordAuthentication** à **yes** au niveau de `/etc/ssh/sshd_config` au niveau du client, si non il faut le reconfigurer à **yes** et appliquer les changements via la commande **systemctl reload sshd**

➤ Installation Ansible

`apt install ansible -y`

➤ Post installation côté serveur

Editer le fichier inventaire `/etc/ansible/hosts` et ajouter les clients exemple

`[clients]`

`172.31.4.210`



Les playbooks de ansible

- Un Playbook Ansible est un fichier yml qui encapsule un ensemble de commandes à exécuter sur la machine cible

- name: Setup web servers
hosts: ubuntu
tasks:
 - name: Ensure Nginx is installed
apt:
 - name: nginx
state: present
 - name: Ensure Nginx is running
service:
 - name: nginx
state: started
 - name: Copy index.html
copy:
 - src: index.html
dest: /var/www/html/

```
ansuser@PC2023:~$ ansible-playbook web.yml

PLAY [Setup web servers] *****

TASK [Gathering Facts] *****
ok: [192.168.1.14]

TASK [Ensure Nginx is installed] *****
ok: [192.168.1.14]

TASK [Ensure Nginx is running] *****
ok: [192.168.1.14]

TASK [Copy index.html] *****
changed: [192.168.1.14]

PLAY RECAP *****
192.168.1.14 : ok=4 changed=1
```

Le résultat attendu



Les playbooks de ansible

- Exécuter la même commande avec l'option `-b` pour que l'utilisateur infère le root **ansible-playbook -b web.yml**

```
ansuser@PC2023:~$ ansible-playbook -b web.yml

PLAY [Setup web servers] *****

TASK [Gathering Facts] *****
fatal: [192.168.1.14]: FAILED! => {"msg": "Missing sudo password"}

PLAY RECAP *****
192.168.1.14      : ok=0    changed=0    unreachable=0    failed=1
```

- Il faut installer dans ce cas sshpass sur le serveur ansible **apt install sshpass** et puis exécuter avec l'option `-kK`
`ansible-playbook -b web.yml -kK`



Les handlers de ansible

- Les **Handler** assurent que certaines tâches sont réussies tout d'abord pour continuer l'exécution du reste des tâches

```
- name: Setup web servers with handler
hosts: web_servers
tasks:
```

```
  - name: Ensure Nginx is installed
    apt:
      name: nginx
      state: present
    notify: Restart Nginx
```

```
  - name: Ensure Nginx is running
    service:
      name: nginx
      state: started
```

```
- name: Copy index.html
  copy:
    src: index.html
    dest: /var/www/html/
  notify: Restart Nginx
```

handlers:

```
- name: Restart Nginx
  service:
    name: nginx
    state: restarted
```



Les variables de ansible

- Les **Variables** assurent la réutilisabilité du playbook

```
- name: exemple de variables  
hosts: ubuntu  
become: true
```

```
vars:
```

```
- var1: Hello  
- var2: World
```

```
tasks:
```

```
- name: echo text  
  shell: echo "{{var1}} {{var2}}" > content.txt
```



Le module setup de ansible

- Le module **Setup** permet de collecter des informations sur les machines cibles
ansible ubuntu -m setup

```
ansuser@PC2023:/etc/ansible$ ansible ubuntu -m setup
192.168.1.14 | SUCCESS => {
  "ansible_facts": {
    "ansible_all_ipv4_addresses": [
      "192.168.1.14"
    ],
    "ansible_all_ipv6_addresses": [
      "fe80::a00:27ff:fe03:9464"
    ],
    "ansible_apparmor": {
      "status": "enabled"
    }
  },
  "changed": false
}
```

- Avec le setup module peut connaître le type de l'os de la machine distante
ansible ubuntu -m setup -a "filter=*family*"

```
ansuser@PC2023:/etc/ansible$ ansible ubuntu -m setup -a "filter=*family*"
192.168.1.14 | SUCCESS => {
  "ansible_facts": {
    "ansible_os_family": "Debian",
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false
}
ansuser@PC2023:/etc/ansible$ |
```

Fact variable



Le module setup de ansible

- Il est possible d'utiliser les Fact variables directement dans le Playbook

```
---  
- name: exemple avec ansible variable  
  hosts: Tomcat  
  become: true  
  vars:  
    - var1: Hello devops1  
    - var2: Good Morning devops2  
  
  tasks:  
    - name: echo text  
      shell: echo "{{var1}} is var1, but var2 is {{var2}}" > /home/ansadmin/{{ansible_os_family}}.txt
```



Les facts variables

- Il est possible d'utiliser les Fact variables directement dans le Playbook

```
---  
- name: exemple avec ansible variable  
  hosts: Tomcat  
  become: true  
  vars:  
    - var1: Hello devops1  
    - var2: Good Morning devops2  
  
  tasks:  
    - name: echo text  
      shell: echo "{{var1}} is var1, but var2 is {{var2}}" > /home/ansadmin/{{ansible_os_family}}.txt
```



La flexibilité des Playbooks avec jinja2

- La création de fichiers statiques pour chacune de ces configurations n'est pas une solution efficace. Jinja2 est un moteur de création de modèles utilisé par Ansible pour générer dynamiquement des configurations basées sur des variables et des expressions au niveau des PlayBooks
- Voici quelques notes
 - `{{ }}` : Ces doubles accolades sont les balises largement utilisées dans un fichier modèle et elles sont utilisées pour intégrer des variables et finalement imprimer leur valeur lors de l'exécution du code
 - `{% %}` : Ces accolades sont principalement utilisés pour les instructions de contrôle telles que les boucles et les instructions if-else
 - `{# #}` : Ces accolades désignent des commentaires qui décrivent une tâche.
- Les fichiers modèles portent l'extension **.j2** , ce qui implique que le modèle Jinja2 est utilisé



Les fichiers templating à l'aide des variables de Playbooks avec jinja2

- Pour comprendre voici un exemple
- Commencer par enregistrer ce contenu sous un fichier **test.j2**

Ce ci est un fichier template version_number et server changent à partir du Playbook

La `{{ version_number }}` est exécutée sur le serveur `{{ server }}`!!!

- Exécuter ce Playbook

- hosts: ubuntu

vars:

version_number: 'v1.0.0'

server: 'Ubuntu'

tasks:

- name: tester jinja 2

template:

src: test.j2

dest: /home/ansuser/resultat.txt



Les fichiers templating à l'aide des variables de Playbooks avec jinja2

- Voici un deuxième exemple **exemple2.j2**

La liste des légumes

```
{% for item in legumes %}  
    {{ item }}  
{% endfor %}
```

- Exécuter ce Playbook

- hosts: ubuntu

vars:

legumes: ['Piments','Tomates','Onions']

tasks:

- name: test de l'exemple 2 jija

template:

src: exemple2.j2

dest: /home/ansuser/légumes.txt



Les filtres de jinja2

- Les filtres sont utilisés pour modifier l'apparence des données de sortie ou de formatage
- Par exemple, pour imprimer les valeurs de la liste précédente en majuscules à l'aide du modèle, dirigez l'élément variable vers l'argument ' `upper` ' comme indiqué : `{{ item | upper }}`
- Pour imprimer les valeurs de la liste précédente en miniscule à l'aide du modèle, dirigez l'élément variable vers l'argument ' `lower` ' comme indiqué : `{{ item | lower }}`
- Par exemple, pour imprimer la valeur minimale d'une liste, transmettez la liste entière au filtre ' `min` ' comme indiqué `{{ 100,33,45,65,60,78 | min }}` => 33
- Par exemple, pour imprimer la valeur maximale d'une liste, transmettez la liste entière au filtre ' `max` ' comme indiqué `{{ 888,37,45,65,60,78 | max }}` => 888
- Pour remplacer une chaîne par une nouvelle
`{{ 888,37,45,65,60,78 | replace("37","38") }}` => 888,38,45,65,60,78
- Pour obtenir des valeurs uniques à partir d'une liste de valeurs en double dans un tableau
`{{ 1,1,1,2,2,3,3,4,4,5,5 | unique }}` => 1,2,3,4,5



Le débogage des Playbooks ansible

- Il est possible de déboguer les Playbook ansible avec le mot clé debug

Option1 :

```
---
- name: exemple avec ansible debug
  hosts: Tomcat
  become: true
  vars:
    - var1: Hello devops1
  tasks:
    - name: echo text
      command: echo -e "{{var1}} is var1 it works"
      register: results

    - name: show results
      debug: msg={{results}}
```

```
ansadmin@ansible-server:~# vim debug_variable.yaml
ansadmin@ansible-server:~# ansible-playbook debug_v

PLAY [exemple avec ansible debug] *****

TASK [Gathering Facts] *****
ok: [172.31.94.73]

TASK [echo text] *****
changed: [172.31.94.73]

TASK [show results] *****
ok: [172.31.94.73] => {
  "msg": {
    "changed": true,
    "cmd": [
      "echo",
      "-e",
      "Hello devops1 is var1 it works"
    ],
    "delta": "0:00:00.005415",
    "end": "2020-11-06 10:39:04.990618",
    "failed": false,
    "rc": 0,
    "start": "2020-11-06 10:39:04.985203",
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Hello devops1 is var1 it works",
    "stdout_lines": [
```

Option2 :

```
---
- name: exemple avec ansible debug
  hosts: Tomcat
  become: true
  vars:
    - var1: Hello devops1
  tasks:
    - name: echo text
      command: echo -e "{{var1}} is var1 it works"
      register: results

    - name: show results
      debug: msg={{results.stdout_lines}}
```



Le clause when ansible

- Parfois des commandes échouent, il est préférable de fournir une alternative avec la commande **when**

```
---  
- name: exemple avec condition when  
  hosts: Tomcat  
  become: true  
  
  tasks:  
    - name: install apache2 on ubuntu  
      apt: name=apache2 state=latest  
        register: results  
  
    - name: install httpd on centos  
      yum: name=httpd state=latest  
        when: results is failed
```

```
when: results is failed
```

```
ansadmin@ansible-server:~# vi when.yaml  
ansadmin@ansible-server:~# ansible-playbook when.yaml  
  
PLAY [exemple avec condition when] *****  
  
TASK [Gathering Facts] *****  
ok: [172.31.94.73]  
  
TASK [install apache2 on ubuntu] *****  
[WARNING]: Updating cache and auto-installing missing dependency: python-apt  
fatal: [172.31.94.73]: FAILED! => {"changed": false, "cmd": "apt-get update", "msg": "[Errno 2] No such file or directory", "rc": 2}  
  
PLAY RECAP *****  
172.31.94.73 : ok=1    changed=0    unreachable=0    failed=1  
kipped=0    rescued=0    ignored=0
```

- Le comportement par défaut de **Ansible** lorsqu'il rencontre une erreur est ne pas continuer l'exécution du reste des tâches sauf si **ignore_errors: yes** est ajoutée à la fin de la tâche douteuse



Le clause ignore_errors

- Le comportement par défaut de **Ansible** lorsqu'il rencontre une erreur est ne pas continuer l'exécution du reste des tâches sauf si **ignore_errors: yes** est ajoutée à la fin de la tâche douteuse

```
---
- name: exemple avec condition when
  hosts: Tomcat
  become: true

  tasks:
    - name: install apache2 on ubuntu
      apt: name=apache2 state=latest
      register: results
      ignore_errors: yes

    - name: install httpd on centos
      yum: name=httpd state=latest
      when: results is failed
```

```
ansadmin@ansible-server:~# vi when.yaml
ansadmin@ansible-server:~# ansible-playbook when.yaml

PLAY [exemple avec condition when] *****

TASK [Gathering Facts] *****
ok: [172.31.94.73]

TASK [install apache2 on ubuntu] *****
[WARNING]: Updating cache and auto-installing missing dependency: python-apt
fatal: [172.31.94.73]: FAILED! => {"changed": false, "cmd": "apt-get update", "msg": "[Errno 2] No such file or directory", "rc": 2}
...ignoring

TASK [install httpd on centos] *****
changed: [172.31.94.73]

PLAY RECAP *****
172.31.94.73 : ok=3 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=1

ansadmin@ansible-server:~#
```



Les boucles dans ansible

- Il est possible de lancer une boucle dans Ansible avec la clause **with_items** et **with_sequence**

```
---
- hosts: all
  tasks:
    - name: Ansible create multiple files example
      file:
        path: "{{item}}"
        state: touch
        mode: 0775
      with_items:
        - init1.txt
        - init2.txt
        - init3.txt
        - init4.txt
```

```
---
- hosts: all
  tasks:
    - name: Ansible create multiple files example
      file:
        path: "Devops {{item}}"
        state: touch
        mode: 0775
      with_sequence: start=20 end=27
```



Protéger les Playbook ansible en les cryptant en utilisant ansible-vault

- La commande **ansible-vault** permet de crypter le contenu d'un Playbook pour la protection
sudo ansible-vault encrypt légumes.yml

```
ansuser@PC2023:~$ sudo cat légumes.yml
$ANSIBLE_VAULT;1.1;AES256
63383832643233393037356564613463303630616234646531333535363536383134626266613336
66636461626663333626435663937363630393738393663610a396531613938313464623937656431
66333831333932373534616466303338356138363535333530373034613132306238343032323465
3633663432393630360a643433396239626564616233646538393530623438303366306230326366
31666264666632366539353763306432623163393961396331376264663933306230646662636565
3233346333616439316133316430308633131303163383966646235306361613865623763383037
36346466396166643431366337626364303866396436343239363533393933663833343035643261
63303232316530356631333762373337643732353562303138643463306634656633633864353734
326234303035643237626162383734636265663365353434306464356436633565313934623336
31633231363432346661623038373932313434353736663538316137363034316662663434356264
39623565343261663062666331633633396537376133643133313236313939366430366336643830
62636536306366303934313838363738383031373633376437323836383765353130303936396138
32353437363963656362663430386263336436376565363138363064643766666639633938366366
3736646637623434663664323866306663636161353037376635
```

- Pour éditer le fichier convenablement
sudo ansible-vault edit légumes.yml

```
---
- hosts: ubuntu
  vars:
    legumes: ['Piments', 'Tomates', 'Onions']

  tasks:
    - name: test de l'exemple 2 jija
      template:
        src: exemple2.j2
        dest: /home/ansuser/légumes.txt
~
```



Protéger les Playbook ansible en les cryptant en utilisant ansible-vault

- Pour utiliser le playbook
sudo ansible-vault edit légumes.yml --ask-vault-pass
- Pour utiliser le playbook il est également possible de décrypter le Playbook avant de l'utiliser
sudo ansible-vault decrypt légumes.yml



Appler un playbook à partir d'un autre playbook

- Importer playbook first et playbook second dans playbook main

```
---  
- import_playbook: first.yml  
- import_playbook: second.yml
```

main.yml

```
---  
- name: first playbook  
  hosts: all  
  become: true
```

```
tasks:  
- name: First task  
  file:  
    path: "first_file"  
    state: touch
```

first.yml

```
---  
- name: second playbook  
  hosts: all  
  become: true
```

```
tasks:  
- name: First task  
  file:  
    path: "second_file"  
    state: touch
```

second.yml



14/05/
2022

Appler un playbook à partir d'un autre playbook

- Importer la tâche task1 et la tâche task2 second dans le playbook main

```
---  
- name: main task  
  hosts: all  
  become: true  
  
  tasks:  
    - include: task1.yml  
    - include: task2.yml
```

main.yml

```
---  
- name: First task  
  file:  
    path: "first_file"  
    state: touch
```

task1.yml

```
---  
- name: Second task  
  file:  
    path: "second_file"  
    state: touch
```

task2.yml



Appliquer un playbook à partir d'un autre playbook

- Un rôle Ansible est une unité logique de code qui encapsule un ensemble d'actions et de tâches spécifiques
- L'utilisation des rôles Ansible facilite la gestion, Ils permettent également de séparer les responsabilités
- Commencer par créer un dossier roles
- Ensuite un dossier dans roles qui porte le nom du role exemple monrole
- Ensuite un dossier tasks dans monrole ainsi que handlers, créer des fichiers main.yml respectivement au niveau de ces deux fichiers

```
ansuser@m1:~/roles$ tree
.
├── monrole
│   ├── handlers
│   │   └── main.yml
│   └── tasks
│       └── main.yml
```



Appliquer un playbook à partir d'un autre playbook

- Ajouter du script dans le fichier main.yml au niveau du dossier tasks

```
- name: first task
  file:
    path: "monfichier"
    state: touch
```

- Créer un playbook qui fait appel au rôle monrole



```
---
- name: main task
  hosts: all
  become: true

  roles:
    - monrole
```



Chercher un playbook dans Ansible Galaxy

- Ansible Galaxy est une plateforme communautaire permettant de partager et de trouver des rôles Ansible





Galaxy NG


Search

Collections >

Roles >


Documentation 

Terms of Use 


Thanks for trying out the new and improved Galaxy, please share your feedback on forum.ansible.com 

Welcome to Galaxy

Filter by keywords



Download

 To be able to download content from galaxy it is required to have `ansible-core >= 2.13.9`. Please, check it running the command: `ansible --version`

Jump-start your automation project with great content from the community

```
ansuser@m1:~$ ansible-galaxy search hello

Found 51 roles matching your search:

Name                                     Description
----                                     -
adelaidearnauld.helloweb                your description
ansibleplaybookbundle.hello-world-apb  Deploys hello-world web appl
benthomasson.hello_role                 Hello World role
benthomasson.hello_role2                Hello World role 2
benthomasson.hello_role3                Hello World role
carlosgo13.hello_world                  An Ansible role to install a
chengzhang1016.hello_world_db_apb      A sample APB which deploys H
chusiang.helloworld                    An Ansible role example for
c-mart.hello-test                       test
cschatano.htnhello                     Getting started
damiennevaldo.galaxy-helloweb           your description
dizager.hello_world_k8s                 Manage hello-world applicati
```

