**Technologie Docker**

Docker est une technologie de **conteneurisation** qui permet d’exécuter des applications dans des environnements isolés appelés **conteneurs**.

**Fonctionnement de Docker**

**Explication** : Docker utilise des **images** (modèles immuables contenant une application et ses dépendances) pour créer des **conteneurs** (instances exécutables des images). Les conteneurs sont isolés mais partagent le système d’exploitation hôte.

=> Docker utilise des **images** pour créer des **conteneurs.** Les conteneurs sont isolés mais partagent le système d’exploitation hôte.

**Concepts clés** :

* **Image** : Fichier statique (ex. : image Ubuntu ou PostgreSQL).
* **Conteneur** : Instance active d’une image, exécutée de manière isolée.
* **Docker Hub** : Plateforme pour télécharger des images (ex. : docker pull alpine).

**Télécharger et exécuter une image**

docker pull alpine

docker run -dit --name conteneur-1 --network git-network alpine

**Manipulation des réseaux avec Docker**

**Explication** : Docker propose plusieurs types de réseaux pour permettre aux conteneurs de communiquer entre eux ou avec l’extérieur.

**Types de réseaux Docker** :

* **Bridge** : Réseau par défaut, les conteneurs sur le même hôte communiquent via une IP privée.
* **Host** : Le conteneur utilise directement le réseau de l’hôte.
* **None** : Pas de réseau, le conteneur est isolé.
* **Overlay** : Pour la communication entre plusieurs hôtes (utilisé dans Docker Swarm).

**Exemple de configuration réseau**

# Créer un réseau personnalisé

docker network create git-network

# Lancer deux conteneurs sur ce réseau

docker run -dit --name conteneur-1 --network git-network alpine

docker run -dit --name conteneur-2 --network git-network alpine

# Vérifier la connectivité

docker exec conteneur-1 ping -c1 172.18.0.3

**Différence entre image et conteneur**

* **Image** : Modèle statique, comme un blueprint. Ex. : Image PostgreSQL sur Docker Hub.
* **Conteneur** : Instance exécutable créée à partir de l’image. Plusieurs conteneurs peuvent être créés à partir de la même image.

# Télécharger l'image

docker pull postgres

# Créer un conteneur à partir de l'image

docker run --name mon-postgres -e POSTGRES\_PASSWORD=secret -d postgres

**Manipulation de bases de données avec Docker**

**Explication** : Docker permet de déployer des bases de données (ex. : PostgreSQL) dans des conteneurs.

**Étapes** :

1. Télécharger l’image de la base de données.
2. Lancer un conteneur avec les variables d’environnement nécessaires.
3. Accéder au conteneur pour gérer la base de données.

# Télécharger l'image PostgreSQL

docker pull postgres

# Lancer un conteneur

docker run --name ca-postgres -e POSTGRES\_PASSWORD=secret -d postgres

# Accéder au conteneur

docker exec -it ca-postgres psql -U postgres

# Créer une base de données

psql> CREATE DATABASE gift;

# Créer une table

psql> CREATE TABLE students (studentid int, studentname char(20), studentgroup char(20));

# Insérer des données

psql> INSERT INTO students (studentid, studentname, studentgroup) VALUES (1001, 'student1', 'git1'), (1002, 'karim', 'git1');

**Volumes Docker**

**Explication** : Les volumes permettent un stockage persistant pour les données des conteneurs, qui sont autrement éphémères. Les volumes sont stockés sur l’hôte (ex. : /var/lib/docker/volumes) et peuvent être partagés entre conteneurs.

# Créer un volume

docker volume create voldata

# Lancer un conteneur avec le volume

docker run -it --name conteneur-ubuntu -v voldata:/data ubuntu

# Écrire des données dans le volume

echo "ceci est un stockage persistant" > /data/test.txt

# Vérifier les données

cat /data/test.txt

**Concepts Dockerfile**

**Explication** : Un **Dockerfile** est un fichier texte contenant des instructions pour construire une image Docker personnalisée. Il définit l’environnement, les dépendances et les commandes à exécuter.

**Structure de base** :

* FROM : Image de base (ex. : ubuntu).
* RUN : Exécute des commandes pendant la construction.
* COPY : Copie des fichiers dans l’image.
* CMD : Commande par défaut exécutée au démarrage du conteneur.

FROM ubuntu:20.04

RUN apt update && apt install -y python3

COPY app.py /app/

CMD ["python3", "/app/app.py"]

**Structure de fichier Docker Compose**

**Explication** : **Docker Compose** est un outil pour définir et gérer plusieurs conteneurs à l’aide d’un fichier YAML (docker-compose.yml). Il est utilisé pour configurer des applications multi-conteneurs.

**Structure de base** :

* version : Version du format Docker Compose.
* services : Liste des conteneurs.
* networks : Configuration des réseaux.
* volumes : Configuration des volumes.

Exemple de docker-compose.yml :

version: '3'

services:

web:

image: nginx:latest

ports:

- "8080:80"

db:

image: postgres:latest

environment:

POSTGRES\_PASSWORD: secret

volumes:

- db-data:/var/lib/postgresql/data

volumes:

db-data:

* Commande pour lancer : docker-compose up -d

**Configuration d’un serveur web et service de base de données**

**Explication** : Docker permet de déployer un serveur web (ex. : Nginx) et une base de données (ex. : PostgreSQL) ensemble. Le fichier Docker Compose facilite leur configuration.

**Exemple avec Docker Compose** :

version: '3'

services:

web:

image: nginx:latest

ports:

- "80:80"

volumes:

- ./html:/usr/share/nginx/html

db:

image: postgres:latest

environment:

POSTGRES\_PASSWORD: secret

volumes:

- db-data:/var/lib/postgresql/data

volumes:

db-data:

**Commande** :

# Lancer les services

docker-compose up -d

# Vérifier les conteneurs

docker ps

**Système de gestion de versions**

**Explication** : Un système de gestion de versions (VCS) comme **Git** permet de suivre les modifications du code source. Git est largement utilisé pour la collaboration et le contrôle des versions.

**Commandes de base :**

# Initialiser un dépôt

git init

# Ajouter des fichiers

git add .

# Créer un commit

git commit -m "Premier commit"

# Pousser vers un dépôt distant

git push origin main

**Configuration :**

git config --global user.name "Votre Nom"

git config --global user.email [votre.email@example.com](mailto:votre.email@example.com)

**Partage NFS (Network File System)**

**Qu’est-ce que c’est ?**  
NFS permet à un **serveur** de partager un dossier sur un réseau. Les **clients** y accèdent comme s’il était local. Idéal pour partager des fichiers entre machines Linux.

**Pourquoi l’utiliser ?**

* Centraliser les données sur un serveur.
* Faciliter le partage dans un réseau.
* Éviter les doublons de fichiers.

**Exemple de configuration :**

# Sur le serveur (Ubuntu/Debian) :

# Installer NFS

sudo apt install nfs-kernel-server

# Créer un dossier à partager

sudo mkdir /var/nfs/partage

# Donner les permissions

sudo chown nobody:nogroup /var/nfs/partage

sudo chmod 775 /var/nfs/partage

# Configurer le partage dans /etc/exports

echo "/var/nfs/partage 192.168.1.0/24(rw,sync,no\_subtree\_check)" | sudo tee -a /etc/exports

# Appliquer et redémarrer NFS

sudo exportfs -ra

sudo systemctl restart nfs-kernel-server

Sur le client (Ubuntu/Debian) :

# Installer le client NFS

sudo apt install nfs-common

# Créer un point de montage

mkdir /mnt/partage

# Monter le dossier partagé (192.168.1.10 = IP du serveur)

sudo mount 192.168.1.10:/var/nfs/partage /mnt/partage

# Vérifier le montage

df -h /mnt/partage

**Notes de ton carnet :**

* Les ports TCP utilisés par NFS ne sont pas spécifiés dans tes notes, mais ils incluent généralement le port **2049** pour le service principal.
* NFS utilise une communication entre serveur et client, souvent sur un réseau local.

**Résumé :**  
NFS partage des dossiers facilement. Configure un dossier sur le serveur avec /etc/exports, et monte-le sur le client avec mount.

**Serveur DHCP (Dynamic Host Configuration Protocol)**

**Qu’est-ce que c’est ?**  
DHCP donne **automatiquement des adresses IP** aux appareils dans un réseau. Ça simplifie la configuration manuelle.

**Pourquoi l’utiliser ?**

* Évite les conflits d’IP.
* Gère les IP dynamiques ou fixes.
* Simplifie l’administration réseau.

**Exemple de configuration :**

# Installer le serveur DHCP (Ubuntu/Debian)

sudo apt install isc-dhcp-server

# Configurer /etc/default/isc-dhcp-server (interface réseau, ex. eth0)

echo "INTERFACESv4=\"eth0\"" | sudo tee /etc/default/isc-dhcp-server

# Configurer /etc/dhcp/dhcpd.conf

cat << EOF | sudo tee /etc/dhcp/dhcpd.conf

option domain-name "helo.ma";

option domain-name-servers 8.8.8.8, 8.8.4.4;

subnet 192.168.1.0 netmask 255.255.255.0 {

range 192.168.1.100 192.168.1.200;

option routers 192.168.1.1;

option broadcast-address 192.168.1.255;

default-lease-time 600;

max-lease-time 7200;

}

# IP fixe pour une machine (ex. Debian 12)

host debian12 {

hardware ethernet 00:26:40:29:32:AA;

fixed-address 192.168.1.200;

}

# Bloquer un appareil (blacklist)

host intrus {

hardware ethernet 08:27:00:AA:BB:CC;

deny booting;

}

EOF

# Démarrer et activer le service

sudo systemctl start isc-dhcp-server

sudo systemctl enable isc-dhcp-server

# Vérifier les baux attribués

cat /var/lib/dhcp/dhcpd.leases

**Notes de ton carnet :**

* **Processus DHCP** : Un client envoie un **DHCP Discover** (broadcast), le serveur répond avec un **DHCP Offer**, le client demande avec **DHCP Request**, et le serveur confirme avec **DHCP ACK**.
* **Bail** : L’IP est valide 50% du temps par défaut, puis renouvelée. Si ça échoue, une nouvelle IP est demandée par diffusion.
* **Configuration** : Définir l’interface dans /etc/default/isc-dhcp-server (ex. eth0).
* **Client** :
  + Linux : sudo dhclient -v (obtenir IP), sudo dhclient -r -v (libérer IP).
  + Windows : ipconfig /renew (renouveler), ipconfig /release (libérer), ipconfig /all (voir config).

**Résumé :**  
DHCP automatise les IP. Configure /etc/dhcp/dhcpd.conf pour des plages d’IP, des fixes, ou des blocages, et démarre le service.

**Protocole SSH (Secure Shell)**

**Qu’est-ce que c’est ?**  
SSH est un protocole sécurisé pour se connecter à un serveur à distance via un **shell**. Il chiffre les communications pour protéger les données.

**Pourquoi l’utiliser ?**

* Accéder à un serveur distant en toute sécurité.
* Gérer des machines à distance (ex. : commandes, fichiers).
* Utilisé par défaut sur le port **22**.

**Exemple de configuration :**

# Installer le serveur SSH (Ubuntu/Debian)

sudo apt install openssh-server

# Vérifier que SSH est actif

sudo systemctl status ssh

# Connexion à un serveur (ex. 192.168.1.100, port 22 par défaut)

ssh user@192.168.1.100

# Changer le port (ex. 2222) dans /etc/ssh/sshd\_config

sudo sed -i 's/#Port 22/Port 2222/' /etc/ssh/sshd\_config

# Redémarrer SSH après modification

sudo systemctl restart sshd

# Vérifier le port utilisé

sudo netstat -tuln | grep 2222

**Notes de ton carnet :**

* SSH s’établit avec un canal de communication chiffré.
* Par défaut, il utilise l’authentification par mot de passe ou clé.
* Le port peut être changé (ex. : 2222) en modifiant /etc/ssh/sshd\_config et redémarrant le service.
* Communication entre deux réseaux via port 22, sauf si modifié.

**Résumé :**  
SSH sécurise l’accès distant. Installe openssh-server, connecte-toi avec ssh, et ajuste le port si besoin.

**Configuration d’un serveur Samba**

**Qu’est-ce que c’est ?**  
Samba permet de partager des fichiers et imprimantes entre Linux et Windows via le protocole SMB.

**Pourquoi l’utiliser ?**

* Partager des dossiers avec des systèmes Windows.
* Centraliser les fichiers dans un réseau mixte.
* Facile à configurer pour les utilisateurs.

**Exemple de configuration :**

# Installer Samba (Ubuntu/Debian)

sudo apt install samba

# Créer un utilisateur Samba

sudo adduser --quiet --disabled-password --shell /bin/false smbuser

sudo smbpasswd -a smbuser

# Configurer un partage dans /etc/samba/smb.conf

cat << EOF | sudo tee -a /etc/samba/smb.conf

[Partage]

path = /srv/samba/partage

writable = yes

browsable = yes

valid users = smbuser

EOF

# Créer le dossier de partage

sudo mkdir -p /srv/samba/partage

sudo chown smbuser:smbuser /srv/samba/partage

# Redémarrer Samba

sudo systemctl restart smbd

# Vérifier la configuration

testparm

**Notes de ton carnet :**

* Installation avec apt install samba.
* Création d’utilisateur avec sudo adduser et mot de passe avec smbpasswd.
* Configuration dans /etc/samba/smb.conf pour définir les partages.
* Redémarrer avec systemctl restart smbd après modifications.

**Résumé :**  
Samba partage des fichiers avec Windows. Installe Samba, configure /etc/samba/smb.conf, et redémarre le service.