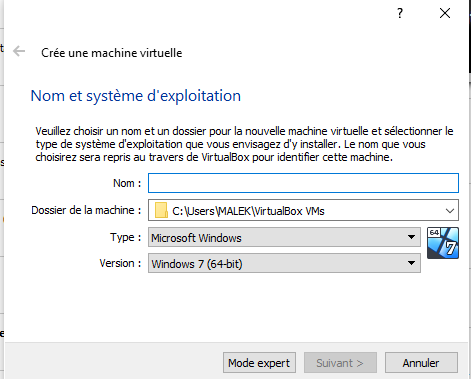
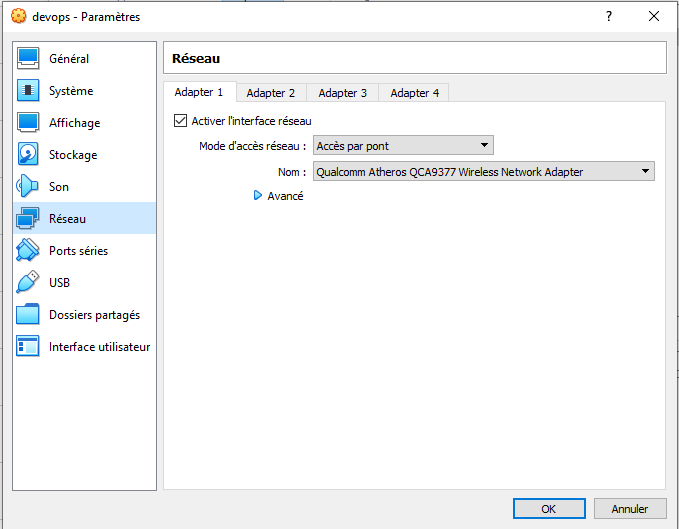
**TP N 1**

**Installer ubuntu**

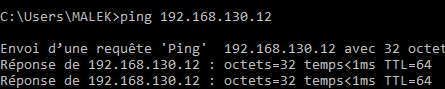
1. Installer Oracle Virtual Box
2. Créer une nouvelle machine virtuelle



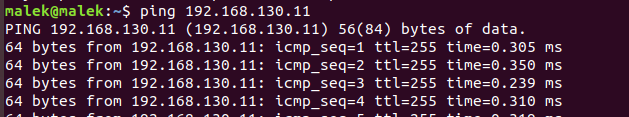
1. Installer ubuntu
2. Configurer Réseau : une interface réseau Accès par pont



1. Sous windows vérifier la connexion



Dans la VM

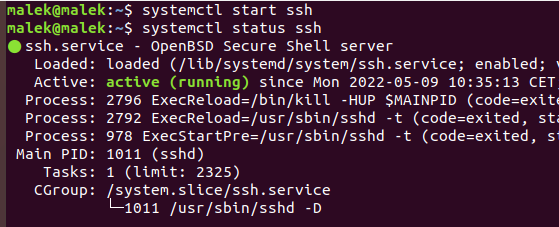


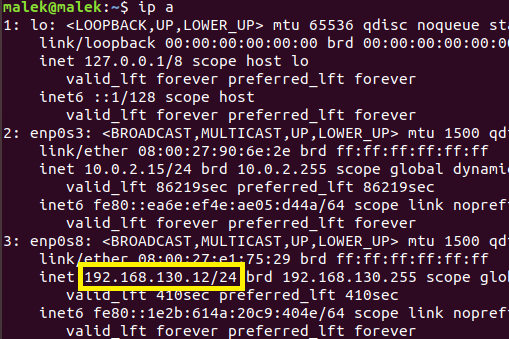
1. Installer open ssh

sudo apt-get update

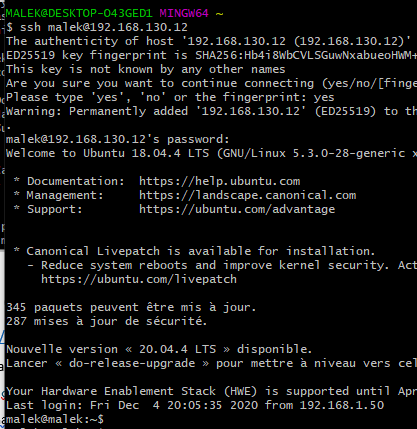
sudo apt-get upgrade

sudo apt-get install openssh-server





1. Démarrer Git



**Installer docker**

<https://docs.docker.com/engine/install/ubuntu/>

1. sudo apt-get install curl

sudo curl -fsSL https://get.docker.com -o get-docker.sh

télécharger docker

ls

*sudo sh get-docker.sh*

docker version : pour vérifier l’installation de docker

**En cas de problème**

sudo rm /var/lib/dpkg/lock-frontend

sudo rm /var/lib/apt/lists/lock

sudo rm /var/cache/apt/archives/lock

sudo rm /var/lib/dpkg/lock

**Windows Télécharger des images Docker**

1. Hub.docker.com

Télécharger hello-world

1. Dans git



**TP N 2**

sudo docker images

La commande sudo docker images affiche la liste de toutes les images Docker présentes localement sur la machine, avec des détails comme le nom, la balise (tag), l'ID de l'image, et la date de création.

sudo docker run -d nginx

La commande sudo docker run -d nginx lance un conteneur en arrière-plan (-d pour détaché) à partir de l'image Docker officielle de Nginx, qui sert un serveur web.

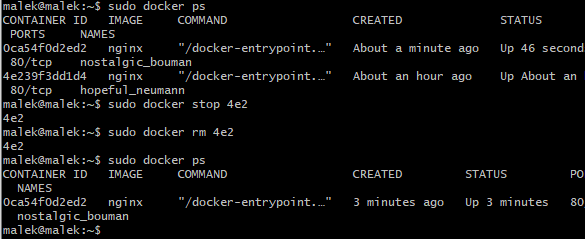
sudo Docker ps

La commande sudo docker ps affiche la liste des conteneurs Docker en cours d'exécution, avec des détails comme l'ID du conteneur, l'image utilisée, l'état, le port, et la commande exécutée.

Sudo docker stop

La commande sudo docker stop arrête un ou plusieurs conteneurs Docker en cours d'exécution, en les identifiant par leur nom ou leur ID.

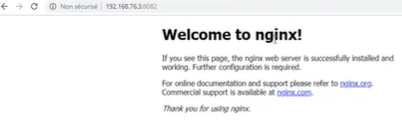
La commande sudo docker rm supprime un ou plusieurs conteneurs Docker arrêtés, identifiés par leur nom ou leur ID.



sudo docker stop ad2b

sudo docker run -d -p 8082:80 nginx

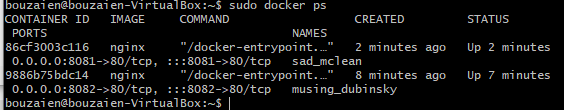
La commande sudo docker run -d -p 8082:80 nginx lance un conteneur Nginx en arrière-plan (-d), tout en mappant le port 8082 de la machine hôte au port 80 du conteneur, permettant d'accéder au serveur Nginx via le port 8082 sur l'hôte.



Mapper le port pour accéder de l’extérieur

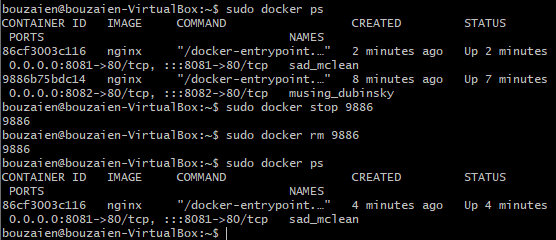
Démarrer un autre serveur web nginx sur le port 8081

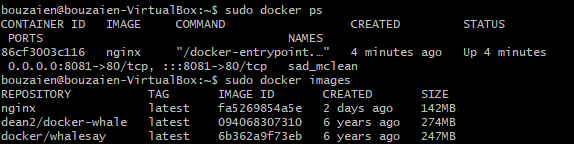
sudo docker run -d -p 8081:80 nginx





Avant de supprimer un docker il faut l’arrêter

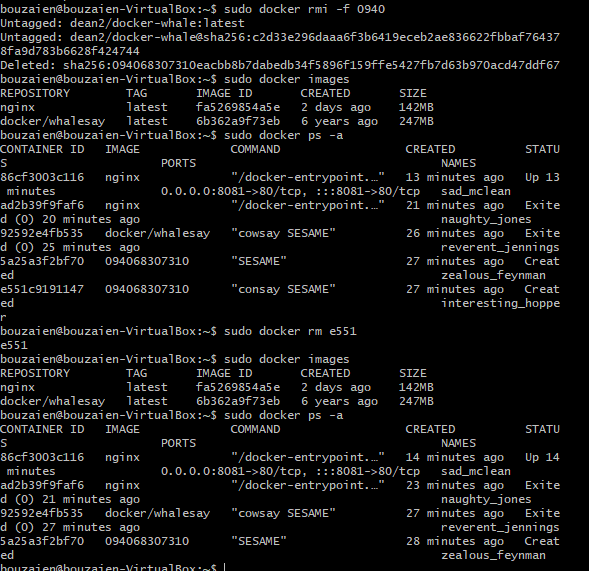




La commande sudo docker ps -a affiche la liste de tous les conteneurs Docker, qu'ils soient en cours d'exécution, arrêtés ou ayant terminé leur exécution, avec des détails comme l'ID, le nom, l'image, l'état et l'heure de création.

La commande sudo docker rmi supprime une ou plusieurs images Docker locales, identifiées par leur nom, balise (tag) ou ID, afin de libérer de l'espace ou de nettoyer des images inutilisées.

La commande sudo docker rmi -f force la suppression d'une ou plusieurs images Docker, même si elles sont utilisées par des conteneurs (en arrêtant et supprimant ces conteneurs au besoin).

p

Modifier la structure du fichier docker.service





Redémarrer docker

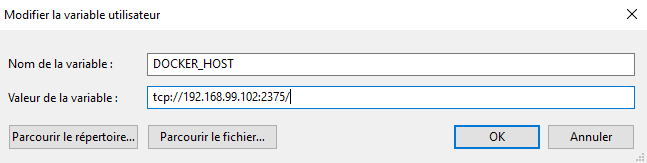


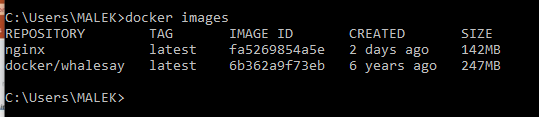


**Partie 2 : cli-docker**

Pour commencer il faut installer choco

Après veuillez taper la commande suivante : choco install docker-cli





**Application Spring Boot**

**Télécharger le dossier exp1-spring-master.rar du lien suivant :**

<https://drive.google.com/file/d/1lqNI6IkEpmdk-8Uzs5mlyXX7KIH-ZA7f/view?usp=sharing>

une fois vous avez télécharger le dossier créer le fichier Dockerfile et ajouter les instructions suivantes :

1. **FROM adoptopenjdk:11-jre-hotspot**  
   Spécifie l'image de base à utiliser pour le conteneur, qui est ici l'image AdoptOpenJDK avec la version 11 de Java Runtime Environment (JRE) et HotSpot comme machine virtuelle Java.
2. **VOLUME /tmp**  
   Crée un volume Docker pour le répertoire /tmp, permettant de partager des fichiers entre le conteneur et l'hôte ou d'autres conteneurs, ainsi que de persister des données générées au cours de l'exécution.
3. **ADD target/\*.jar app.jar**  
   Ajoute le fichier JAR compilé de votre application (qui se trouve dans le répertoire target) dans le conteneur et le renomme app.jar.
4. **CMD ["java", "-jar", "app.jar", "--spring.profiles.active=prod"]**  
   Définit la commande à exécuter lorsque le conteneur démarre, ici pour lancer l'application Java avec le fichier app.jar et en spécifiant le profil Spring actif comme prod.
5. **EXPOSE 8091**  
   Indique que le conteneur écoute sur le port 8091, permettant à d'autres services ou à l'hôte d'accéder à votre application via ce port.

**Dockerfile**

FROM adoptopenjdk:11-jre-hotspot

VOLUME /tmp

ADD target/\*.jar app.jar

CMD ["java", "-jar", "app.jar", "--spring.profiles.active=prod"]

EXPOSE 8080

