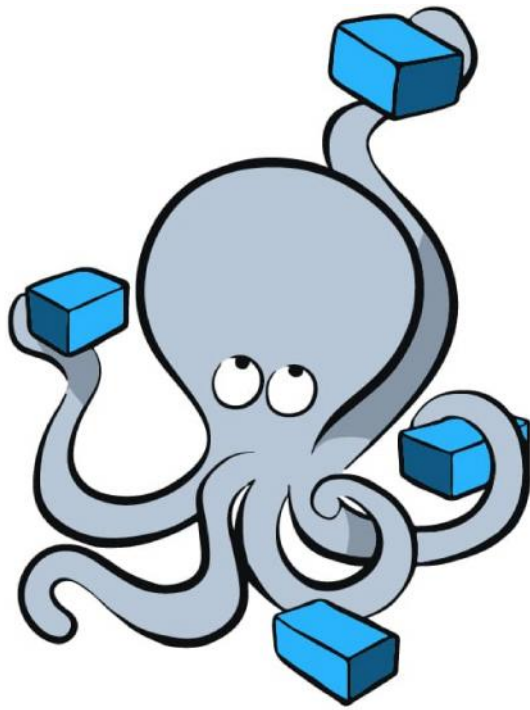


Docker Compose et Volume



docker

Compose et Volume

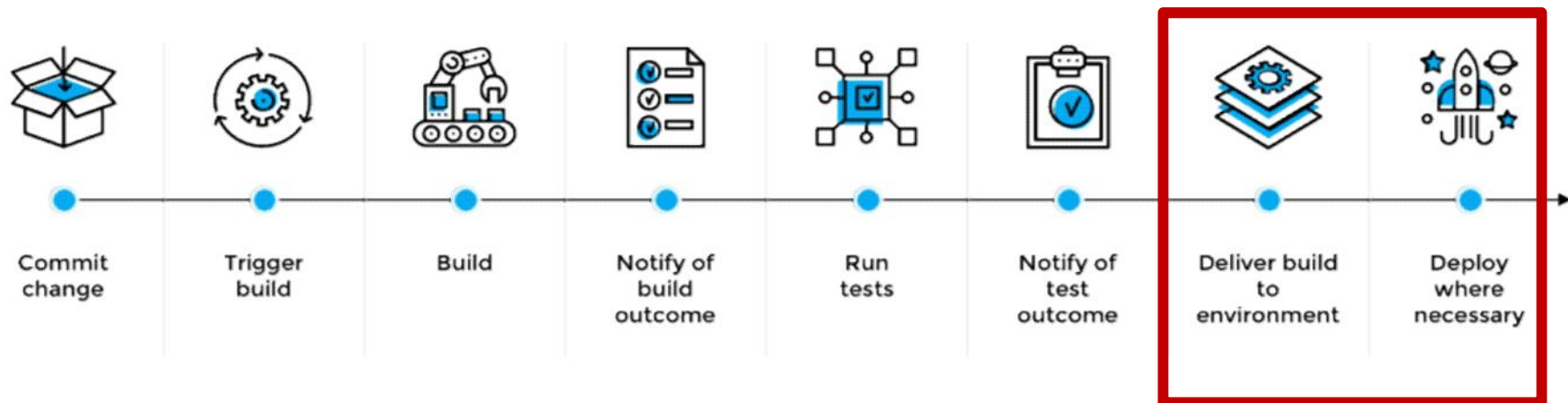
Bureau E204

Plan du cours

- Introduction
- Docker
- Docker Compose
- Docker Volume
- Docker Compose et Jenkins

Introduction

- Notre application Spring Boot codée, compilée et testée (unitairement et qualitativement) doit être intégrée dans une chaine DevOps complète (CI/CD).
- La chaine d'intégration (CI) continue a été réalisée grâce à Jenkins via la création d'un pipeline.
- Dans ce cours on va s'intéresser à **la chaine CD (Continuos delivery and deployment)**



Introduction

- Qu'est ce qu'une livraison continue ?
- Qu'est ce qu'on doit livrer ?
- Où dois-je livrer le livrable ?
- Quelle est la différence entre la livraison continue et le déploiement continu ?

Introduction

- L'objectif de la partie CD (déploiement et livraison continu) est de placer notre application dans une machine de production et d'assurer son fonctionnement (Communication avec la base de données, web services fonctionnels, etc..)
- La machine de production peut être:
 - ✓ Une machine physique
 - ✓ Une machine virtuelle
 - ✓ Un conteneur Docker

Introduction

Chaque application a besoin de connecter à un serveur base de données.

→ Pour que ces deux-là puissent communiquer, il faut les mettre sous le même réseau et lancer la base de données avant le démarrage de l'application.



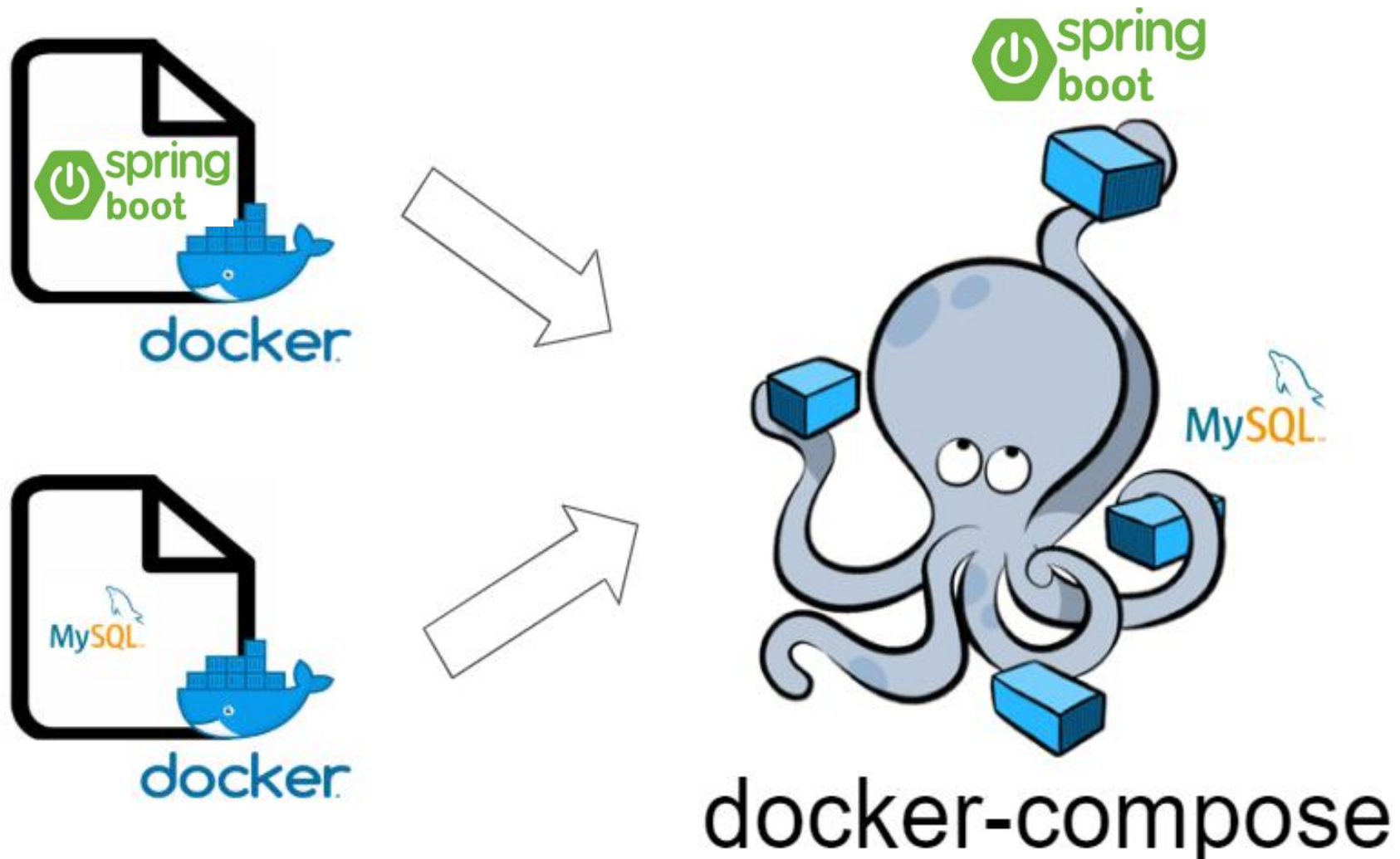
```
docker run -p 9090:9090 --network mynetwork -d app-image-name
```



```
docker run --name mysqldb --network mynetwork -e MYSQL_ROOT_PASSWORD=my-secret-pw -v /home/mysql/data:/var/lib/mysql -d mysql:8
```

Introduction

→ Et là, il nous faut le docker compose.



Docker Compose

- Docker Compose est un outil permettant de définir et d'exécuter des applications Docker multi-conteneurs.
- Dans cette logique, chaque partie de l'application (code, base de données, serveur web, ...) sera hébergée par un conteneur.
- Cet outil repose sur le langage YAML pour décrire l'architecture physique de l'application.
- Le fichier Compose comporte la **version** (OBSOLÈTE), les **services** (REQUIS), les **réseaux**, les **volumes**, les **configurations** et les **secrets**.
- Après la configuration du fichier YAML, une seule commande à exécuter pour créer et démarrer tous les services.

Docker Compose

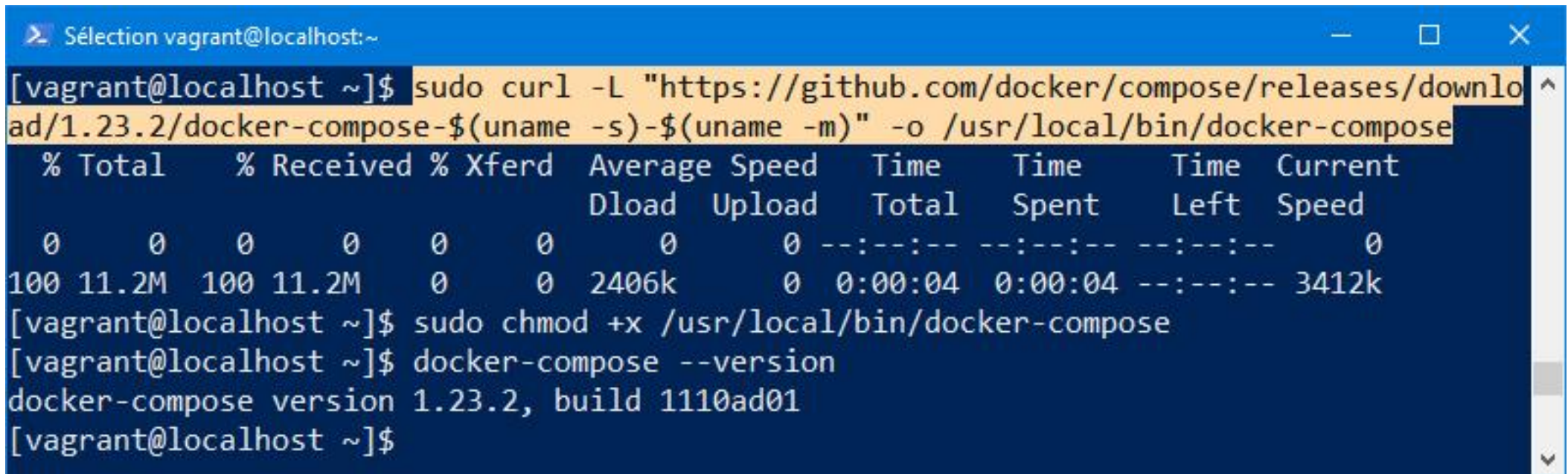


- L'utilisation de Docker Compose se résume à un processus en trois étapes :
 1. Définir l'environnement de votre application à l'aide d'un « DockerFile » afin qu'il puisse être reproduit partout.
 2. Définir les services qui composent votre application dans « Docker-compose.yml » afin qu'ils puissent être exécutés ensemble dans un environnement isolé.
 3. Exécuter la commande « docker compose up », c'est la commande pour lancer votre application entière.

Installation Docker Compose

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.23.2/docker-  
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

```
sudo chmod +x /usr/local/bin/docker-compose
```



```
Sélection vagrant@localhost:~  
[vagrant@localhost ~]$ sudo curl -L "https://github.com/docker/compose/releases/download/1.23.2/docker-  
ad/1.23.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose  
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current  
           % Dload  % Upload   Dload  Upload   Total   Spent    Left     Speed  
0     0     0     0     0     0      0     0  --:--:-- --:--:-- --:--:--     0  
100 11.2M 100 11.2M    0     0 2406k     0  0:00:04  0:00:04 --:--:-- 3412k  
[vagrant@localhost ~]$ sudo chmod +x /usr/local/bin/docker-compose  
[vagrant@localhost ~]$ docker-compose --version  
docker-compose version 1.23.2, build 1110ad01  
[vagrant@localhost ~]$
```

Les 3 fonctions principales

Les 3 fonctions principales de docker-compose sont :

- Comment lancer un docker-compose? (se mettre dans le dossier contenant le fichier docker-compose.yml) :
docker-compose up -d
- Comment vérifier les logs des conteneurs qui ont été lancé?
docker-compose logs
- Comment arrêter un docker compose ?
docker-compose down

docker-compose.yml

```
version: "3.9"

services:
  mysqlldb:
    container_name: mysqlldb
    image: mysql:8
    restart: unless-stopped
    environment:
      MYSQL_ROOT_PASSWORD: root
    volumes:
      - /home/mysql/data:/var/lib/mysql

  spring_app:
    image: achatproject
    container_name: achatproject
    restart: on-failure
    ports:
      - 9090:9090
    depends_on:
      - mysqlldb
```

Par défaut, Compose crée un réseau pour vos services. Chaque conteneur d'un service rejoint ce réseau et devient alors accessible aux autres conteneurs.

Service base de données.

Cette partie remplace la commande:

```
docker run --name mysqlldb --network mynetwork
-e MYSQL_ROOT_PASSWORD=root
-v /home/mysql/data:/var/lib/mysql -d mysql:8
```

L'application Backend.

```
docker run -p 9090:9090 --network mynetwork -d
app-image-name
```

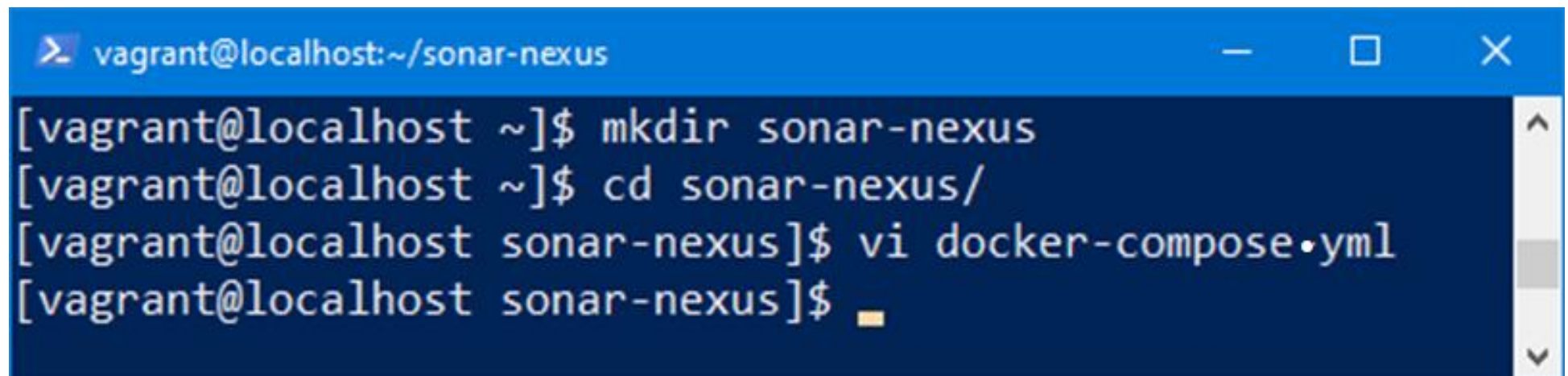
TP en Classe

Essayez de lancer Nexus et Sonar via un `docker-compose.yml`.

Solution ci-dessous :

Docker Compose - Exécuter des conteneurs ensemble

- Pour utiliser « Docker-compose », nous allons configurer les deux images 'Sonarqube' et 'Nexus' afin de les lancer simultanément.
- Créer un dossier nommé « sonar-nexus » et ensuite aller dans ce répertoire
- Créez maintenant le fichier YAML en utilisant votre éditeur de texte préféré:

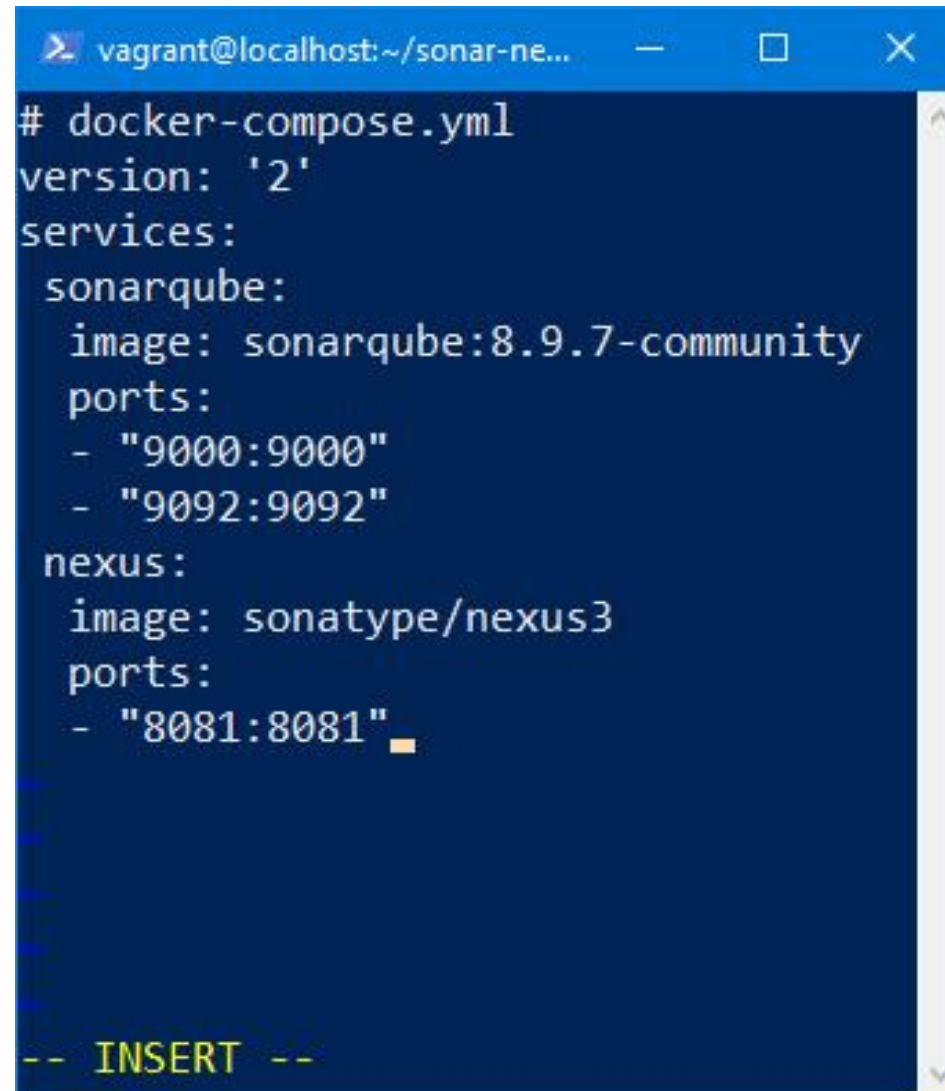


```
vagrant@localhost:~/sonar-nexus

[vagrant@localhost ~]$ mkdir sonar-nexus
[vagrant@localhost ~]$ cd sonar-nexus/
[vagrant@localhost sonar-nexus]$ vi docker-compose.yml
[vagrant@localhost sonar-nexus]$
```

Docker Compose - Exécuter des conteneurs ensemble

- Contenu du fichier « docker-compose.yml »:

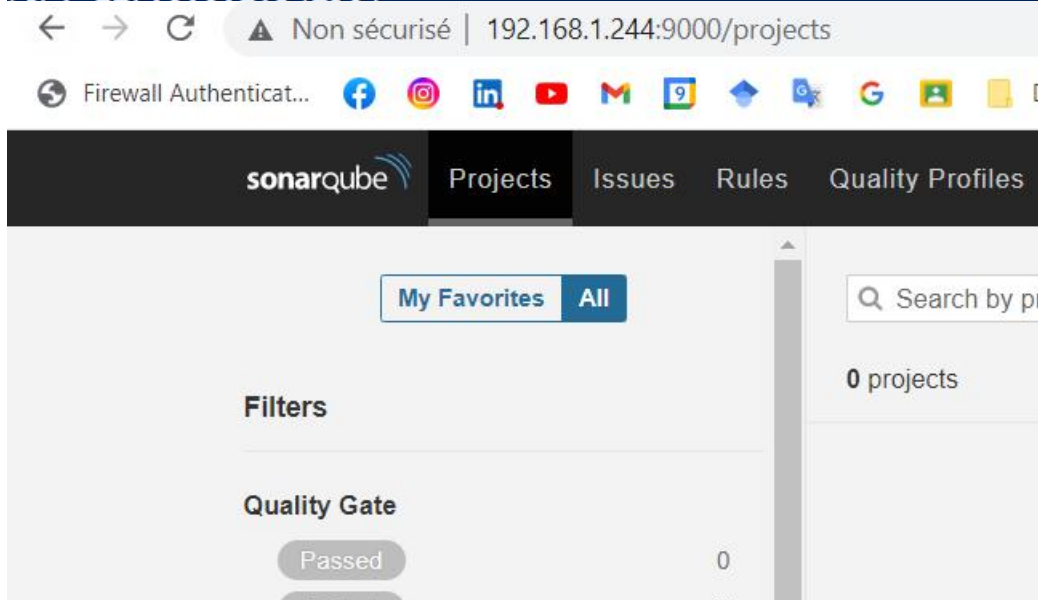
A screenshot of a terminal window with a blue title bar. The title bar text is 'vagrant@localhost:~/sonar-ne...'. The terminal content shows a YAML configuration for Docker Compose. It starts with a comment '# docker-compose.yml', followed by 'version: '2'', and a 'services:' section. Under 'services:', there are two services: 'sonarqube' and 'nexus'. 'sonarqube' has 'image: sonarqube:8.9.7-community' and 'ports' mapping '9000:9000' and '9092:9092'. 'nexus' has 'image: sonatype/nexus3' and 'ports' mapping '8081:8081'. At the bottom, there is a yellow prompt '-- INSERT --'.

```
vagrant@localhost:~/sonar-ne...  
# docker-compose.yml  
version: '2'  
services:  
  sonarqube:  
    image: sonarqube:8.9.7-community  
    ports:  
      - "9000:9000"  
      - "9092:9092"  
  nexus:  
    image: sonatype/nexus3  
    ports:  
      - "8081:8081"  
-- INSERT --
```

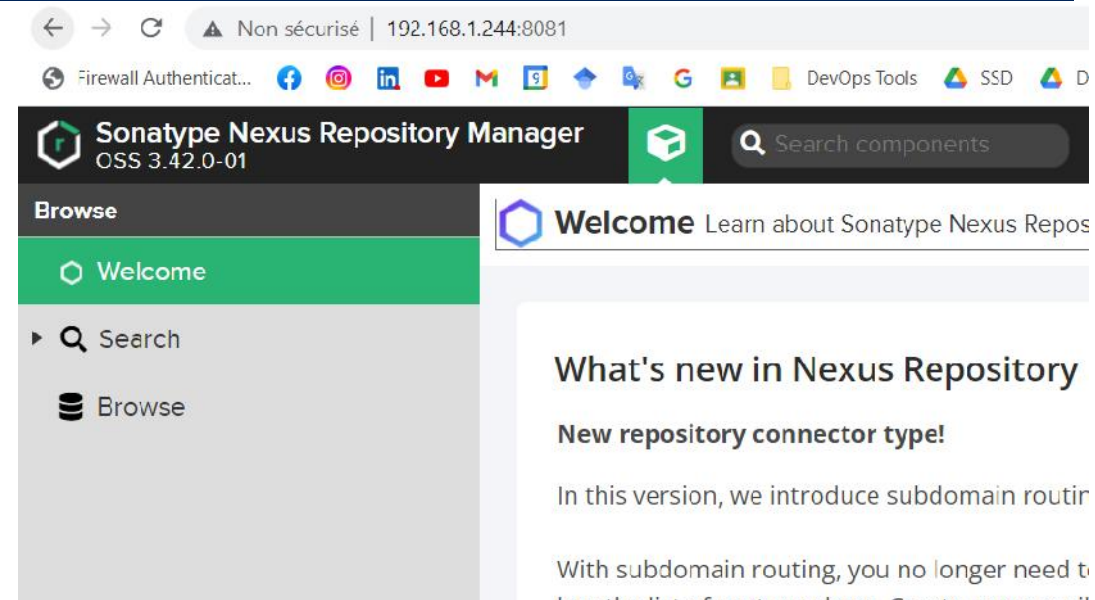

Docker Compose - Exécuter des conteneurs ensemble

- Exécuter la commande suivante pour créer le conteneur

```
Sélection vagrant@localhost:~/sonar-nexus  
[vagrant@localhost sonar-nexus]$ docker-compose up  
Creating network "sonar-nexus_default" with the default driver  
Creating sonar-nexus_nexus_1 ... done  
Creating sonar-nexus_sonarqube_1 ... done  
Attaching to sonar-nexus_sonarqube_1, sonar-nexus_nexus_1  
sonarqube_1 | 2022.10.10 23:11:50 INFO app[][o.s.a.AppFileSystem] Cleaning or creating temp directory /opt/sonarqube  
/temp  
sonarqube_1 | 2022.10.10 23:11:50 INFO app[][o.s.a.es.EsSettings] Elasticsearch listening on [HTTP: 127.0.0.1:9001,  
TCP: 127.0.0.1:34250]
```



SonarQube

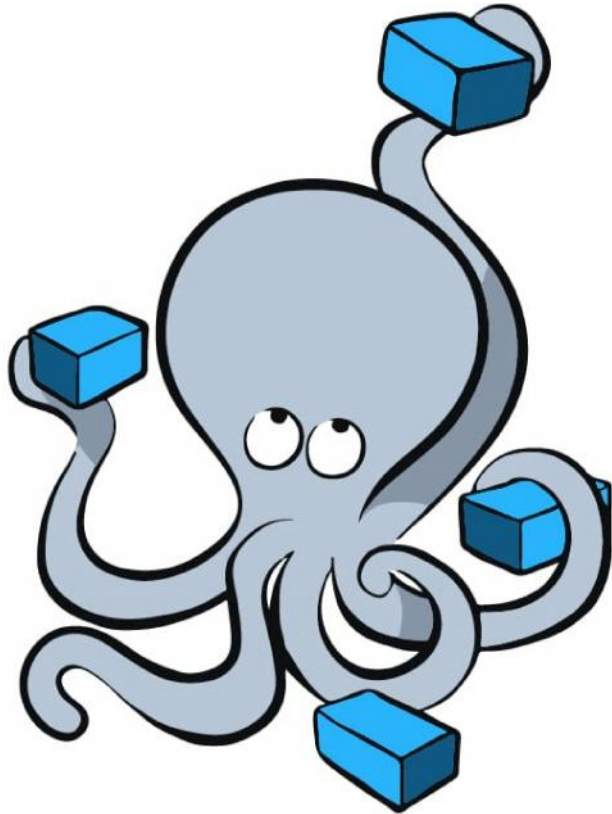


Nexus

TP – Projet final

- Créer un Dockerfile dans votre projet achat pour permettre la création de l'image.
- Créer un docker-compose.yml pour faire tourner votre application achat avec une base de données MySQL (inspirez vous de l'exemple ci-dessus).
- Ajouter un troisième conteneur avec l'application Angular (code source sur le Drive).

Docker Compose



docker
Compose