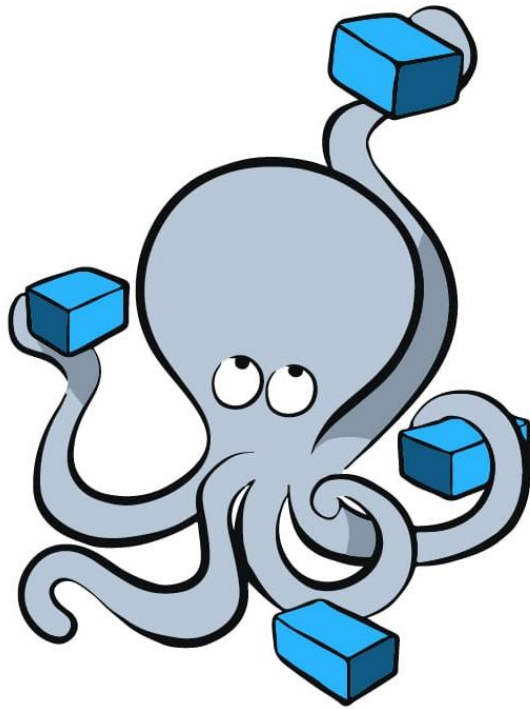


# Docker Compose et Volume



# docker

## Compose et Volume

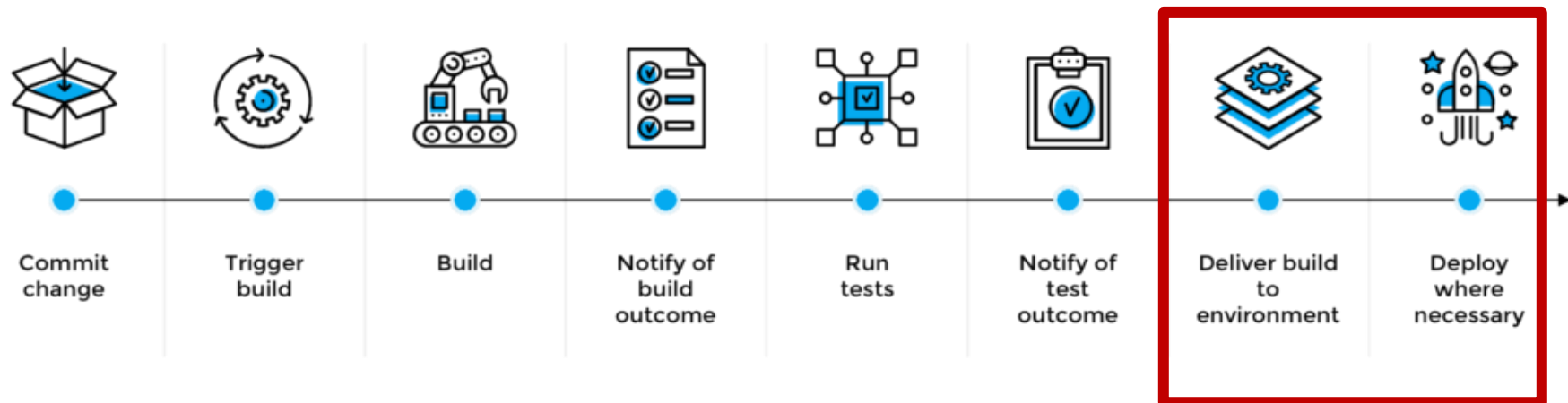
**Bureau E204**

# Plan du cours

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

# Introduction

- Notre application Spring Boot codé, compilé et testé (fonctionnellement 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'une pipeline automatisée déclenchée lors d'une détection d'un push dans le référentiel du code.
- 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
  - ✓ Une image Docker

# Introduction

Nous avons vu que nous pouvons isoler chaque application à l'intérieur d'une image où nous pouvons définir son environnement dans un Dockerfile.

Puis, avec un simple “docker build” et “docker run”, notre application sera accessible via le port que nous avons exposé:

- `docker build -t image_name .`
- `docker run -p 8080:8080 image_name`



# 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.



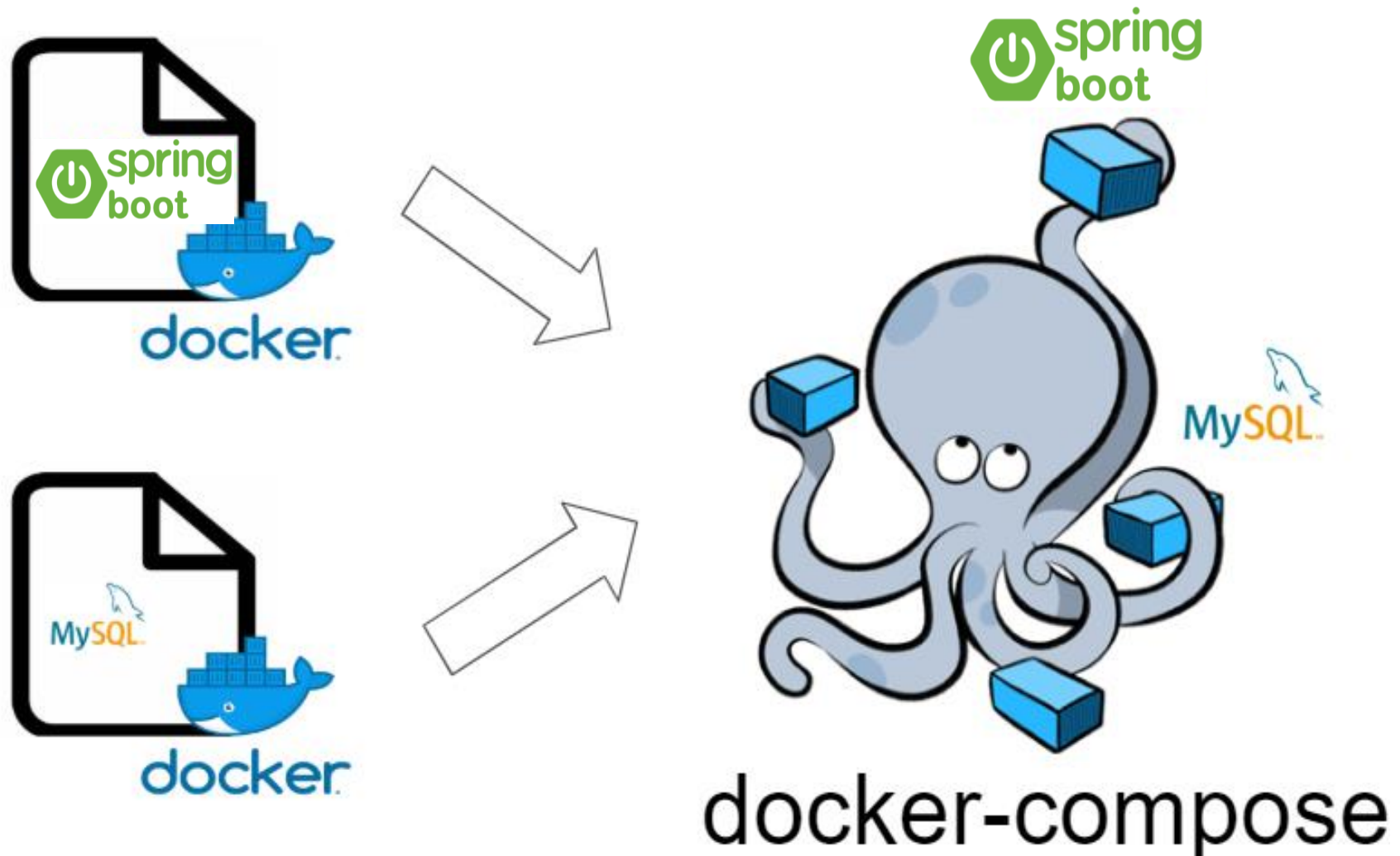
```
docker run -p 9090:9090 --network mynetwork -d app-image-name
```



```
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 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, il suffit d'exécuter une seule commande 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 » pour lancer votre application entière.

# Docker Compose - Exemple

▶ docker-compose.yml x

1 # docker-compose.yml

2 version: '2'

La version du format de fichier  
Compose (1,2 ou 3)

3 services:

Définir les images à exécuter simultanément

4 image\_node:

5 image: node:12

Le nom de base de  
l'image (node.js 12.x)

6 working\_dir: /app

Répertoire de travail  
des commandes

7 ports:

8 - '8080:8081'

Exposer le port du  
conteneur à l'hôte

La définition  
d'un conteneur

# Docker Compose – Installation (Méthode 01)

- Installer le plugin Docker Compose:

✓ `sudo yum install docker-compose-plugin`

```
[root@localhost vagrant]# sudo yum install docker-compose-plugin
Loaded plugins: fastestmirror, product-id, search-disabled-repos,
               : subscription-manager

This system is not registered with an entitlement server. You can use subscription-manager to register.

Loading mirror speeds from cached hostfile
 * base: miroir.univ-lorraine.fr
 * extras: miroir.univ-lorraine.fr
 * updates: fr2.rpmfind.net
base
docker-ce-stable
extras
|
```

- Vérifier l'installation:

✓ `docker-compose --version`

```
[root@localhost vagrant]# docker-compose --version
docker-compose version 1.23.2, build 1110ad01
```

# Docker Compose – Installation (Méthode 02)

- Installer Docker Compose à partir du binaire du dépôt GitHub de Docker:

✓ `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`

```
[root@localhost vagrant]# 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
```

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
			Dload Upload	Total	Spent	Left	Speed
0	0	0	0	0	--:--:--	0:00:02	0
10	11.2M	10	1199k	0	0	142k	0
				0:01:20	0:00:08	0:01:12	227k

- Définir les permissions pour rendre le binaire exécutable:

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

```
[root@localhost vagrant]# sudo chmod +x /usr/local/bin/docker-compose
```

- Vérifier l'installation:

✓ `docker-compose --version`

```
[root@localhost vagrant]# docker-compose --version
docker-compose version 1.23.2, build 1110ad01
```

# 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 « SonarAndNexus » et ensuite aller dans ce répertoire

```
[root@localhost vagrant]# mkdir SonarAndNexus  
[root@localhost vagrant]# cd SonarAndNexus/  
[root@localhost SonarAndNexus]#
```

- Créez maintenant le fichier YAML en utilisant votre éditeur de texte préféré:

```
[root@localhost SonarAndNexus]# nano docker-compose.yml
```

# Docker Compose - Exécuter des conteneurs ensemble

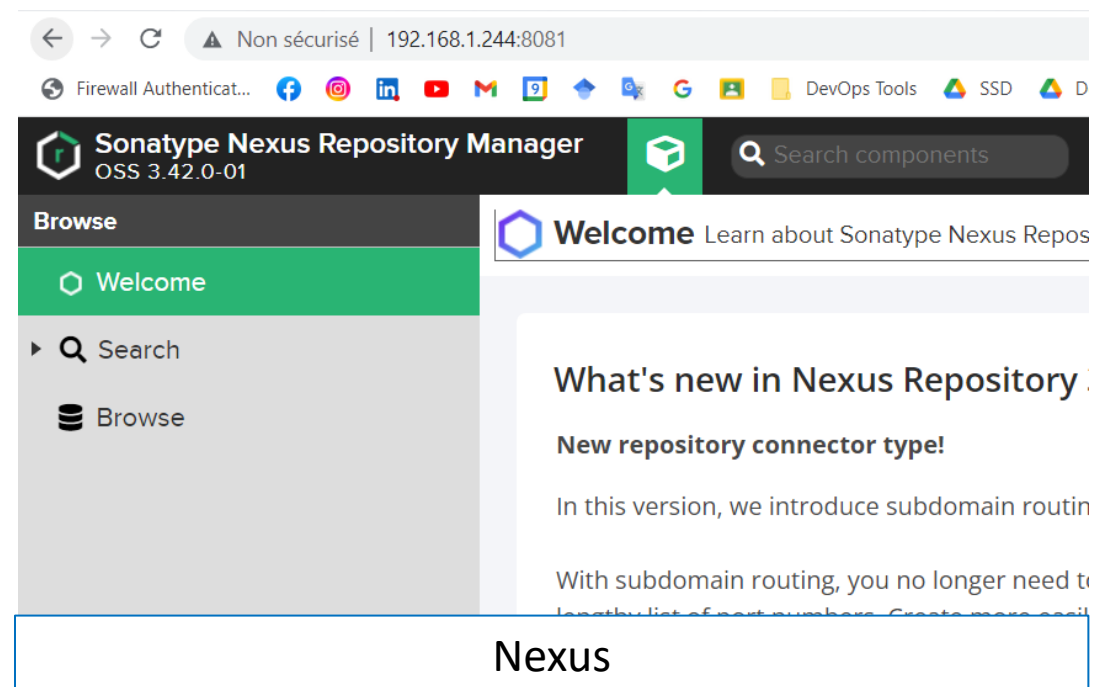
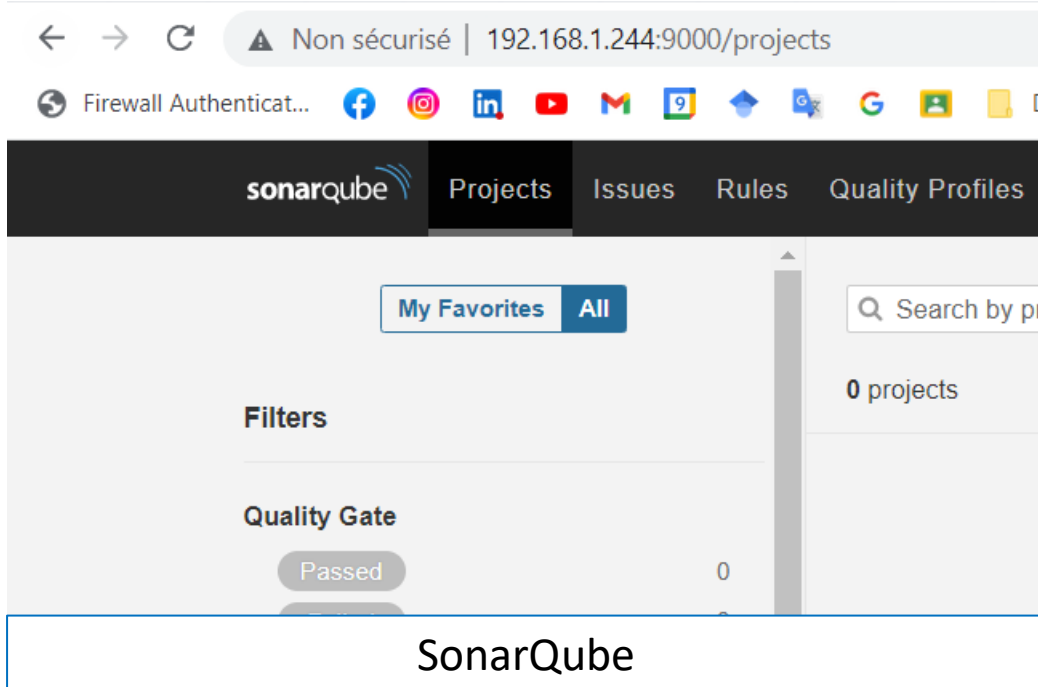
- Rédiger le fichier « docker-compose.yml »:

```
# 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"
```

# Docker Compose - Exécuter des conteneurs ensemble

- Exécuter la commande suivante pour créer le conteneur

```
[root@localhost SonarAndNexus]# docker-compose up
Creating sonarandnexus_sonarqube_1 ... done
Creating sonarandnexus_nexus_1 ... done
Attaching to sonarandnexus_nexus_1, sonarandnexus_sonarqube_1
sonarqube_1 | 2022.10.09 21:19:48 INFO app[o.s.a.AppFileSystem] Cleaning or creating temp directory
sonarqube_1 | 2022.10.09 21:19:48 INFO app[o.s.a.es.EsSettings] Elasticsearch listening on [HTTP: 1
sonarqube_1 | 2022.10.09 21:19:48 INFO app[o.s.a.ProcessLauncherImpl] Launch process[[key='es', ipc
elasticsearch: /opt/sonarqube/elasticsearch/bin/elasticsearch
```





# Docker Compose - Exécuter des conteneurs ensemble

- Comment vérifier les logs des conteneurs qui ont été lancés ?

docker-compose logs

```
[root@localhost SonarAndNexus]# docker-compose logs
Attaching to sonarandnexus_nexus_1, sonarandnexus_sonarqube_1
sonarqube_1 | 2022.10.09 21:19:48 INFO app[][o.s.a.AppFileSystem] Cleaning or cre
sonarqube_1 | 2022.10.09 21:19:48 INFO app[][o.s.a.es.EsSettings] Elasticsearch T
sonarqube_1 | 2022.10.09 21:19:48 INFO app[][o.s.a.ProcessLauncherImpl] Launch p
elasticsearch]: /opt/sonarqube/elasticsearch/bin/elasticsearch
sonarqube_1 | 2022.10.09 21:19:48 INFO app[][o.s.a.SchedulerImpl] Waiting for Ela
sonarqube_1 | warning: no-jdk distributions that do not bundle a JDK are deprecate
sonarqube_1 | 2022.10.09 21:19:58 INFO es[][o.e.n.Node] version[7.16.2], pid[40],
:42:46.604893745Z], OS[Linux/3.10.0-1160.76.1.el7.x86_64/amd64], JVM[Eclipse Adopt
sonarqube_1 | 2022.10.09 21:19:58 INFO es[][o.e.n.Node] JVM home [/opt/java/openj
nexus_1 | 2022-10-09 21:20:09,885+0000 INFO [FelixStartLevel] *SYSTEM org.son
nexus_1 | 2022-10-09 21:20:11,535+0000 WARN [CM Event Dispatcher (Fire Config
s not writeable: file:/opt/sonatype/nexus/etc/karaf/jmx.acl.cfg
nexus_1 | 2022-10-09 21:20:12,130+0000 WARN [CM Event Dispatcher (Fire Config
tall - File is not writeable: file:/opt/sonatype/nexus/etc/karaf/org.apache.karaf.T
nexus_1 | 2022-10-09 21:20:12,142+0000 WARN [CM Event Dispatcher (Fire Config
```

# Docker Compose - Exécuter des conteneurs ensemble

- Comment arrêter un docker compose ?

`docker-compose down`

```
[root@localhost SonarAndNexus]# docker-compose down
Stopping sonarandnexus_nexus_1      ... done
Stopping sonarandnexus_sonarqube_1  ... done
Removing sonarandnexus_nexus_1      ... done
Removing sonarandnexus_sonarqube_1  ... done
Removing network sonarandnexus_default
```

# Docker Compose - Exécuter des conteneurs ensemble

- Une fois que nous arrêtons l'exécution du « Docker-compose » et nous le démarrons une autre fois, nous devons **refaire** la configuration.
- En fait, la configuration est stockée dans le conteneur. Mais, si nous le supprimons, nous supprimons aussi les données de configuration.

Comment palier à ce problème ?

→ Docker Volume.

# Docker Volume

- Les volumes sont le mécanisme privilégié pour la persistance des données générées et utilisées par les conteneurs Docker.
- Les volumes permettent de garder en mémoire des données de manière permanente.
- Le volume est une fonctionnalité très intéressante dans Docker. Il rend l'utilisation des conteneurs encore plus attrayante.
- Avec des volumes bien configurés, il est possible de réutiliser certaines données dans un autre conteneur, de les exporter ailleurs ou de les importer.

# Docker Volume – Configuration dans Docker-Compose

```
# docker-compose.yml
version: '2'
services:
  sonarqube:
    image: sonarqube:8.9.7-community
    ports:
      - "9000:9000"
      - "9092:9092"
    volumes:
      - 'SonarQube_data:/opt/SonarQube/data'
      - 'SonarQube_extensions:/opt/SonarQube/extensions'
      - 'SonarQube_logs:/opt/SonarQube/logs'
  nexus:
    image: sonatype/nexus3
    ports:
      - "8081:8081"
    volumes:
      - 'nexus-data:/nexus-data'
volumes:
  SonarQube_data:
  SonarQube_extensions:
  SonarQube_logs:
  nexus-data:
```

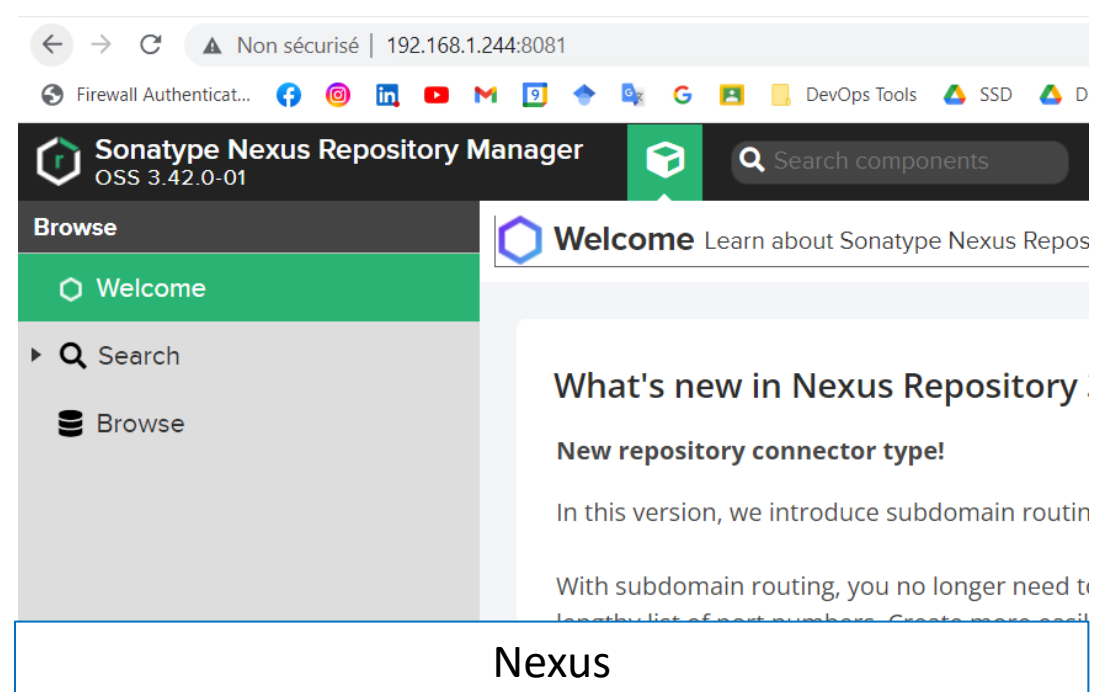
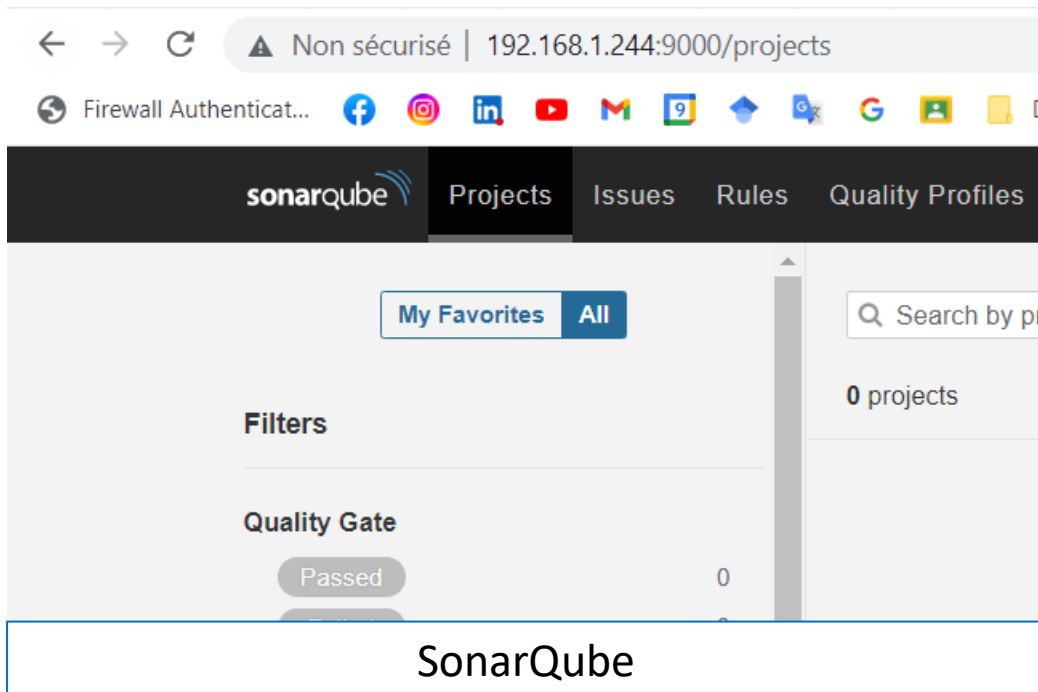
Les espaces de stockage réservés pour 'SonarQube'

L'espace de stockage réservé pour 'Nexus'

Déclaration des espaces de stockage

# Docker Volume – Configuration dans Docker-Compose

```
[root@localhost SonarAndNexus]# docker-compose up
Creating volume "sonarandnexus_SonarQube_data" with default driver
Creating volume "sonarandnexus_SonarQube_extensions" with default driver
Creating volume "sonarandnexus_SonarQube_logs" with default driver
Removing sonarandnexus_sonarqube_1
sonarandnexus_nexus_1 is up-to-date
Recreating a13b6c82d625_sonarandnexus_sonarqube_1 ... done
```



# Docker et Jenkins


- Pour automatiser la création des images « Docker » dans « Jenkins »

1. Installer le plugin « Docker Pipeline »:

## Installation/Mise à jour des Plugins

Préparation

- Vérification de la connexion à internet
- Vérification de la connexion à jenkins-ci.org
- Succès

Authentication Tokens API	 Succès
Docker Commons	 En cours d'installation <div><div></div></div>
Docker Pipeline	 En cours
Loading plugin extensions	 Pending

➡ [Revenir en haut de la page](#)  
(vous pouvez commencer à utiliser les plugins installés dès maintenant)

➡ ☐ Redémarrer Jenkins quand l'installation est terminée et qu'aucun job n'est en cours

# Docker et Jenkins

1. Implémenter le fichier « DockerFile » pour créer l'image pour déployer la partie « BackEnd ».
2. Ajouter les « stages » nécessaires pour créer, construire et déposer l'image à déployer (Partie Spring) dans « DockerHub »

```
stage('Building image') {  
    steps{  
  
    }  
}
```

« A Compléter ...»

```
stage('Deploy Image') {  
    steps{  
  
    }  
}
```

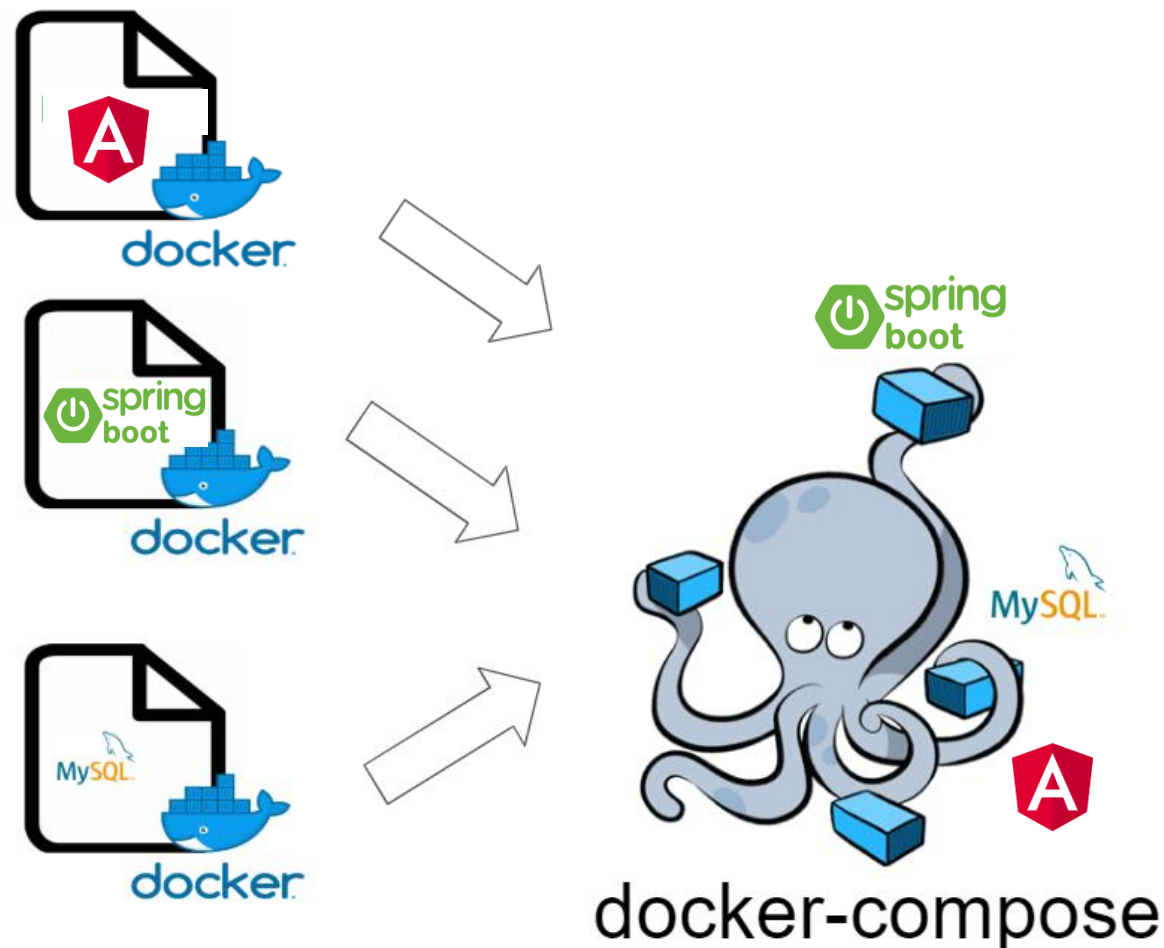
« A Compléter ...»

3. Implémenter le fichier « Docker-compose » adéquat pour déployer l'application « TP Achat »



# Docker et Jenkins

4. Ajouter le « stage » nécessaire pour lancer le fichier « Docker-compose » automatiquement avec l'orchestrateur Jenkins.



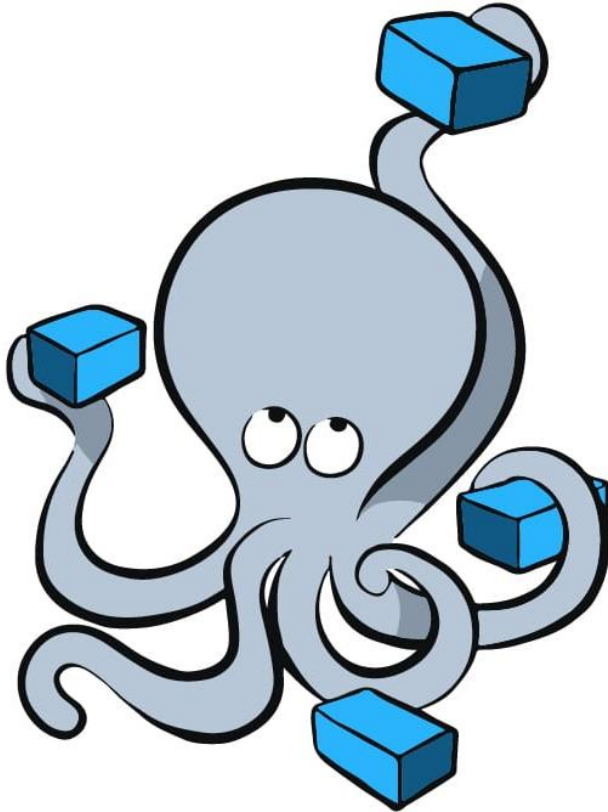
# Docker Compose

Si vous avez des questions, n'hésitez pas à nous contacter :

**Département Informatique**  
**UP ASI**

Bureau E204

# Docker Compose



# docker Compose