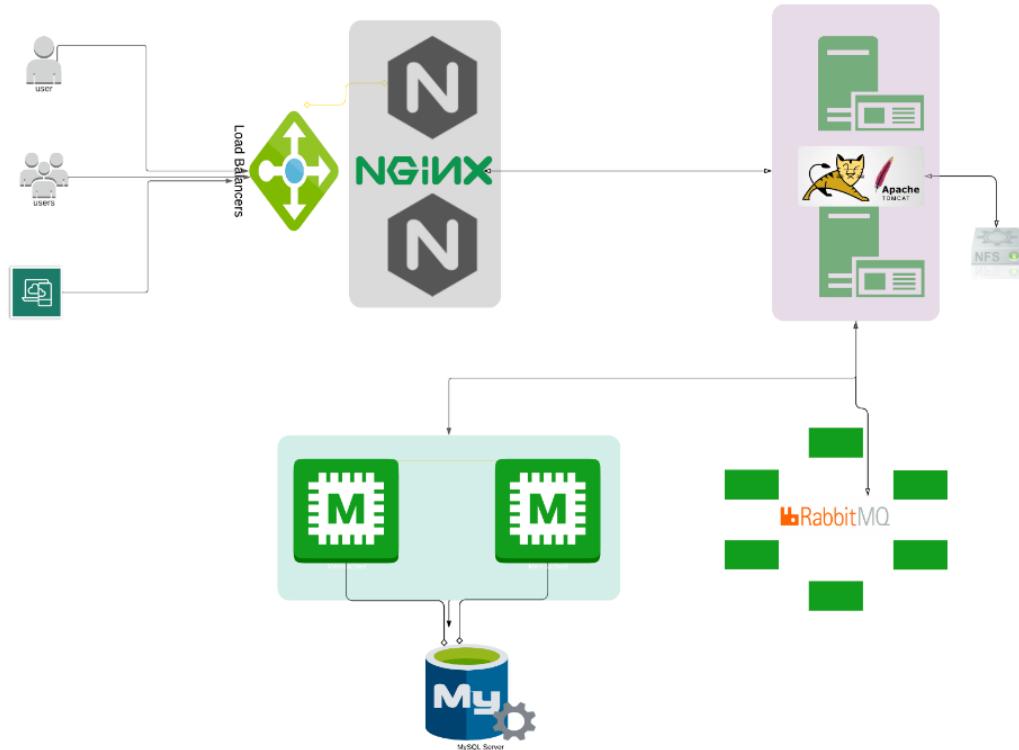


# Projet DevOps – Infrastructure Microservices avec Docker et Vagrant

Ce projet met en place une infrastructure complète de type microservices, déployée via Docker Compose et provisionnée automatiquement avec Vagrant. Il comprend un backend Java, un serveur web, une base de données, du cache, un broker de messages et un load balancer – le tout dans un environnement reproductible.

## Architecture du projet



Cette image représente l'infrastructure complète du projet.

## Environnement de Développement (Vagrant)

Le fichier Vagrantfile automatise la création d'une VM Ubuntu 20.04 avec Docker et Docker Compose préinstallés.

### Configuration principale

#### Installation des prérequis

```
sudo apt-get install ca-certificates curl gnupg -y
```

Installe les outils nécessaires :

- *ca-certificates* : gestion des certificats SSL pour les connexions sécurisées
- *curl* : téléchargement de fichiers depuis internet
- *gnupg* : pour vérifier les signatures GPG (authenticité des dépôts)

#### Création du répertoire pour les clés

```
sudo install -m 0755 -d /etc/apt/keyrings
```

Crée un répertoire pour stocker les clés GPG de manière sécurisée (*/etc/apt/keyrings*) avec les droits 0755 (lecture/écriture pour root, lecture/exécution pour les autres).

#### Ajout de la clé GPG officielle de Docker

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

Télécharge la **clé GPG** de Docker et la convertit (*dearmor*) pour qu'elle soit lisible par apt, puis la stocke dans le dossier de keyrings.

```
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

Donne les droits de lecture à tous les utilisateurs pour la clé (important pour qu'apt puisse l'utiliser sans erreur).

### Ajout du dépôt officiel Docker

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Crée un nouveau dépôt apt pointant vers les paquets Docker :

- *arch=\$(dpkg --print-architecture)* : détecte l'architecture
- *\$VERSION\_CODENAME* : extrait la version Ubuntu
- Ce dépôt sera signé avec la clé GPG ajoutée.

### Installation de Docker

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y
```

Installe :

- *docker-ce* : moteur Docker Community Edition
- *docker-ce-cli* : l'interface en ligne de commande Docker
- *containerd.io* : runtime de conteneurs léger
- *docker-buildx-plugin* : outil de build avancé
- *docker-compose-plugin* : plugin officiel pour docker compose

### Installation manuelle de Docker Compose

```
sudo curl -L "https://github.com/docker/compose/releases/download/v2.1.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

Télécharge manuellement Docker Compose v2.1.1 depuis GitHub, adapté au système (uname -s pour l'OS, uname -m pour l'architecture CPU), et l'enregistre dans /usr/local/bin.

## Docker Compose

Le fichier docker-compose.yml permet de déployer tous les services nécessaires à l'architecture.

### Explication des services

| Service     | Rôle                                |
|-------------|-------------------------------------|
| vprodb      | Base de données MySQL               |
| vprocache01 | Service de cache Memcached          |
| vpromq01    | Message broker RabbitMQ             |
| vproapp     | Application backend Java sur Tomcat |
| vproweb     | Serveur NGINX pour load balancing   |

### Lancement du projet

```
vagrant up  
vagrant ssh
```

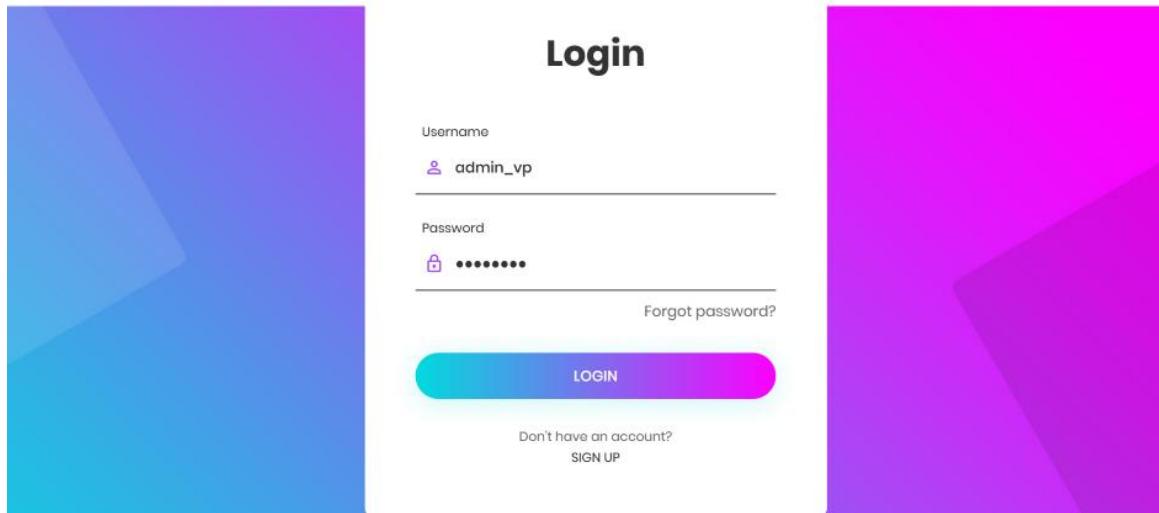
```
sudo -i  
  
mkdir compose  
cd /compose
```

```
vim docker-compose.yml => Copier le contenu à partir de mon github  
docker-compose up -d
```

### Résultat attendu

Une fois lancé :

- L'interface web est accessible sur <http://192.168.56.82>



Username : admin\_vp

Password : admin\_vp

Une vue d'ensemble du site

- RabbitMQ accessible sur <http://192.168.56.82:15672>
- L'application communique entre services via RabbitMQ et Memcached

## **Technologies utilisées**

- Docker / Docker Compose
- Vagrant / VirtualBox
- Ubuntu 20.04
- NGINX / Apache Tomcat
- RabbitMQ / Memcached / MySQL

## **Auteur**

Projet réalisé par : [Badaoudou Barro]

Projet pédagogique démontrant une architecture DevOps microservices.