

Équida

Mise en production

MARTIN Justine
BOTTON Léa

1 Procédure

On se connecte au serveur en suivant les identifiants fournis dans [Ressources](#).

On va commencer par cloner le projet dans le home de root. En étant connecté en tant que root on fait donc :

- cd
- git clone https ://github.com/justine-martin-study/Equida

On va ensuite installer phpmyadmin, php, mysql, apache2 afin de pouvoir gérer la base de données de manière graphique grâce à PhpMyAdmin.

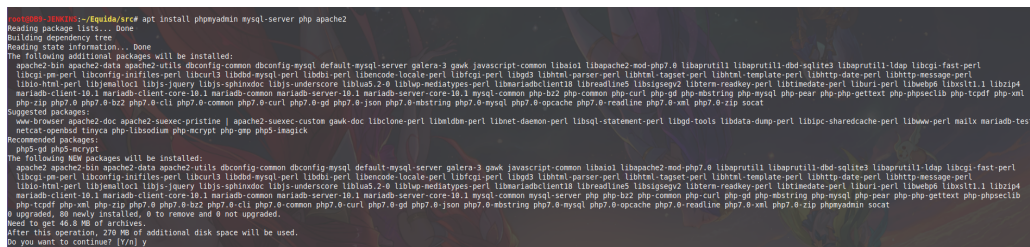


FIGURE 1.1 – Installation de lamm

Il faut maintenant autoriser l'utilisateur phpmyadmin à se connecter à la base de données. Ainsi, on va se connecter à la base de données en utilisant la commande mysql.

- mysql -u root -p

Après avoir saisi le mot de passe choisi à l'installation, on arrive sur l'invite de commande du serveur MySQL. Sur cette console il faut alors saisir les commandes suivantes :

- GRANT ALL PRIVILEGES ON *.* TO 'phpmyadmin'@'localhost' ;
- FLUSH PRIVILEGES ;
- EXIT

On peut maintenant exécuter les scripts SQL du dépôt Git par l'intermédiaire de PhpMyAdmin. On crée alors la base de données "equida" et on y exécute les scripts SQL.

Importation dans la base de données «equida»

Fichier à importer :

Le fichier peut être comprimé (gzip, bzip2, zip) ou non.
Le nom du fichier comprimé doit se terminer par **[format].[compression]**. Exemple: **.sql.zip**

Parcourir : bdd.sql (Taille maximum: 2 048Kio)

Vous pouvez également faire glisser et déposer un fichier sur n'importe quelle page.

Jeu de caractères du fichier :

FIGURE 1.2 – Execution des script SQL

NB : Tous les scripts ont auparavant été compilés en 1 seul nommé bdd.sql

On obtient alors la base de données avec toutes ses tables et ses enregistrements.

Table	Action	Lignes	Type	Interclassement	Taille	Perte
<input type="checkbox"/> CATEG_VENTE		4	InnoDB	utf8mb4_general_ci	16 Kio	-
<input type="checkbox"/> CHEVAL		5	InnoDB	utf8mb4_general_ci	80 Kio	-
<input type="checkbox"/> CLIENT		46	InnoDB	utf8mb4_general_ci	16 Kio	-
<input type="checkbox"/> CLIENT_CATEG_VENTE		11	InnoDB	utf8mb4_general_ci	48 Kio	-
<input type="checkbox"/> COMPTE		2	InnoDB	utf8mb4_general_ci	32 Kio	-
<input type="checkbox"/> COURRIEL		4	InnoDB	utf8mb4_general_ci	32 Kio	-
<input type="checkbox"/> COURSE		10	InnoDB	utf8mb4_general_ci	16 Kio	-
<input type="checkbox"/> DIRECTEUR_GENERAL		1	InnoDB	utf8mb4_general_ci	16 Kio	-
<input type="checkbox"/> ENCHERE		4	InnoDB	utf8mb4_general_ci	48 Kio	-
<input type="checkbox"/> JOINT		4	InnoDB	utf8mb4_general_ci	48 Kio	-
<input type="checkbox"/> LIEU		2	InnoDB	utf8mb4_general_ci	16 Kio	-
<input type="checkbox"/> LOT		8	InnoDB	utf8mb4_general_ci	48 Kio	-

FIGURE 1.3 – base de données Post execution des script SQL

Maintenant que la base de données est fonctionnelle on doit maintenant permettre l'exécution des applications développés. On va donc installer nodejs (et npm). Java 8 étant déjà présent par défaut, il n'est pas nécessaire de le préciser dans la commande. Sur notre version de Debian, NodeJs n'était pas fournis dans une version suffisamment haute pour faire tourner ionic. On a donc du ajouter le dépôt de ionic à la liste des dépôts utilisés.

- apt install curl
- curl -sL https://deb.nodesource.com/setup_8.x | bash -
- apt install nodejs npm
- npm install -g ionic

```
root@DB9-JENKINS:/etc/apt# curl -sL https://deb.nodesource.com/setup_8.x | bash -
## Installing the NodeSource Node.js 8.x LTS Carbon repo...

## Populating apt-get cache...

+ apt-get update
Ign:1 http://ftp.debian.org/debian stretch InRelease
Hit:2 http://ftp.debian.org/debian stretch-updates InRelease
Hit:3 http://ftp.debian.org/debian stretch Release
Ign:5 http://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:6 http://pkg.jenkins.io/debian-stable binary/ Release
Hit:8 http://security.debian.org stretch/updates InRelease
Reading package lists... Done

## Confirming "stretch" is supported...

+ curl -sLf -o /dev/null 'https://deb.nodesource.com/node_8.x/dists/stretch/Release'

## Adding the NodeSource signing key to your keyring...

+ curl -s https://deb.nodesource.com/gpgkey/nodesource.gpg.key | apt-key add -
OK

## Creating apt sources list file for the NodeSource Node.js 8.x LTS Carbon repo...

+ echo 'deb https://deb.nodesource.com/node_8.x stretch main' > /etc/apt/sources.list.d/nodesource.list
+ echo 'deb-src https://deb.nodesource.com/node_8.x stretch main' >> /etc/apt/sources.list.d/nodesource.list

## Running 'apt-get update' for you...

+ apt-get update
Hit:1 http://security.debian.org stretch/updates InRelease
Ign:2 http://ftp.debian.org/debian stretch InRelease
Ign:3 http://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:4 http://ftp.debian.org/debian stretch-updates InRelease
Hit:5 http://ftp.debian.org/debian stretch Release
Hit:6 http://pkg.jenkins.io/debian-stable binary/ Release
Get:9 https://deb.nodesource.com/node_8.x stretch InRelease [4620 B]
Get:10 https://deb.nodesource.com/node_8.x stretch/main Sources [762 B]
Get:11 https://deb.nodesource.com/node_8.x stretch/main amd64 Packages [1008 B]
Fetched 6390 B in 1s (3719 B/s)
Reading package lists... Done
```

FIGURE 1.4 – Ajout du dépôt officiel de NodeJs

Pour compiler les projets basés sur Spring Boot on doit maintenant installer Gradle sur le serveur.

- cd /tmp
- wget https://services.gradle.org/distributions/gradle-5.4.1-all.zip
- mkdir /opt/gradle
- unzip -d /opt/gradle gradle-5.4.1-all.zip

Pour que l'on puisse utiliser la commande "gradle" on doit donc modifier le .bashrc de root pour ajouter cette ligne : export PATH=\$PATH :/opt/gradle/gradle-5.4.1/bin

```
if [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
fi

PS1='${debian_chroot:+($debian_chroot)}\[\033[01;31m\]

export PATH=$PATH:/opt/gradle/gradle-5.4.1/bin
```

FIGURE 1.5 – Modification du PATH dans le .bashrc

Afin de prendre en compte les changements on doit alors se déconnecter puis se reconnecter en tant que root.

Le choix a alors été fait de créer des services pour gérer plus efficacement le démarrage automatique des applications lors d'un démarrage, le redémarrage en cas d'interruption lors d'un crash par exemple, etc. Les services ajoutés sont donc equidaIonic pour démarrer le projet Ionic, equidaRest pour démarrer l'API, equidaWebApp pour démarrer le site web. On a donc le code suivant pour les services

```
[Unit]
Description=Equida Rest

[Service]
Type=simple

User=root
Group=root
UMask=007

ExecStart=/usr/bin/java -jar /var/www/html/Equida-Rest.jar

Restart=on-failure

# Configures the time to wait before service is stopped forcefully.
TimeoutStopSec=300

[Install]
WantedBy=multi-user.target
```

FIGURE 1.6 – Service equidaRest

```

[Unit]
Description=Equida WebApp

[Service]
Type=simple

User=root
Group=root
UMask=007

ExecStart=/usr/bin/java -jar /var/www/html/Equida-WebApp.jar

Restart=on-failure

# Configures the time to wait before service is stopped forcefully.
TimeoutStopSec=300

[Install]
WantedBy=multi-user.target

```

FIGURE 1.7 – Service equidaWebApp

```

[Unit]
Description=Equida Ionic

[Service]
Type=simple

User=root
Group=root
UMask=007

ExecStart= /usr/local/bin/equidaIonicStart

Restart=on-failure

# Configures the time to wait before service is stopped forcefully.
TimeoutStopSec=300

[Install]
WantedBy=multi-user.target

```

FIGURE 1.8 – Service equidaIonic

Le script equidaIonicStart est le suivant :

```

#!/bin/bash

cd /var/www/html/ionic
ionic serve --prod --port=8100

```

FIGURE 1.9 – Script equidaIonicStart

Ce script est situé dans /usr/local/bin et est bien entendu executable.

Afin d'automatiser la mise en production des applications, un script "updateEquida" est créée. Voici son code :

```
#!/bin/sh

#Mise a jour du depot
cd /root/Equida
git checkout master
git pull

#Compilation WebApp
cd src/Spring/webApp
gradle bootJar

#Compilation REST
cd ..
cd rest
gradle bootJar

#Mise en production WebApp
cp /root/Equida/src/Spring/webApp/build/libs/webApp-* /var/www/html/Equida-WebApp.jar

#Mise en production REST
cp /root/Equida/src/Spring/rest/build/libs/rest-* /var/www/html/Equida-Rest.jar

#Mise en production ionic
cp /root/Equida/src/ionic /var/www/html/ionic -r
cd /var/www/html/ionic/
npm install

#Redemarrage des services
service equidaWebApp restart
service equidaRest restart
service equidaIonic restart
```

FIGURE 1.10 – Script updateEquida

Ce script est également situé dans /usr/local/bin et est executable lui aussi.

2 Gestion des applications

La gestion des applications est donc simplifiée. On peut gérer leur exécution avec l'utilisation de la commande "service" ou "systemctl".

Concernant la mise à jour des application en production, il suffit d'exécuter la commande "updateEquida" qui va alors pull la branche master, car c'est celle qui est considéré comme étant stable, puis remplacer les anciennes applications par les nouvelles fraîchement compilés. Cependant, il est très important de noter que le script ne permet pas de modifier la ligne de la classe "rest-api.service" qui indique l'url de l'api Rest dans le projet Ionic. il faut donc modifier manuellement cette ligne pour le moment.

3 Ressources

Accès serveur en local (ssh) : 172.20.0.249 :22

Accès serveur à distance (ssh) : nas.inforostand14.net :2249

Compte utilisateur : leaju/mpLeaju

Compte Admin : root/adm5RV

Compte BDD : phpmyadmin/mpLeaju