Manuel d'installation du portail **WalleSmart**

COOLS Aurélie, GAUTHIER Inès, GENIN Emilie, LERAT Jean-Sébastien
ROBBERTS François, ROMBAUX Michael, VANDEN DRIES Virginie

Juin 2019

Table des matières

1 Introduction **1**

2 Environnement de travail **1**
2.1 Ubuntu 1
2.2 Zeppelin 9
2.3 NFS 16
2.4 Jenkins 17
2.5 Apache 17
2.6 Sources de données 17
2.6.1 PostgreSQL 18
2.6.2 MariaDB 18
2.6.3 REST 19

3 Site Web **20**
3.1 Portail WalleSmart 20
3.2 Création des bases de données 21
3.3 Reverse Proxy 21

4 Sécurité **22**
4.1 Réseau 22
4.2 Apache 23
4.3 Zeppelin 23

5 Conclusion **23**

Liste des figures

1 Ubuntu, choix de la langue. 2
2 Ubuntu, choix de la disposition clavier. 2
3 Ubuntu, choix du mode de fonctionnement. 3
4 Ubuntu, configuration réseau. 3
5 Ubuntu, configuration serveur mandataire. 4
6 Ubuntu, choix du mirroir. 4
7 Ubuntu, partitionnement. 5
8 Ubuntu, bootloader. 5
9 Ubuntu, choix de la langue. 6
10 Ubuntu, confirmation. 6
11 Ubuntu, compte utilisateur. 7
12 Ubuntu, choix des packages. 7
13 Ubuntu, choix de la langue. 8
14 Zeppelin, accès aux interpréteurs. 11
15 Zeppelin, interpréteur `Access`. 11
16 Zeppelin, interpréteur `CSV`. 12
17 Zeppelin, interpréteur `MariaDB`. 13
18 Zeppelin, interpréteur `PostgreSQL`. 14
19 Zeppelin, interpréteur `Python`. 14
20 Zeppelin, exemple de source `Access`. 15
21 Zeppelin, exemple de source `CSV`. 15
22 Zeppelin, exemple de source `PostgreSQL`. 16
23 Zeppelin, exemple de source `REST`. 16
24 Sécurité, topologie proposée. 22

1 Introduction

L'objectif de ce document est de regrouper la procédure d'installation de chacun des composants du projet Wallesmart. Ce projet, en tant que *proof of concept*, a nécessité la mise en place d'une infrastructure de développement. La procédure d'installation des divers composants est expliquée de manière succincte mais explicite et est subdivisée en trois catégories :

Environnement de travail permet de préparer les sources de données et les machines afin d'accueillir le portail Web WalleSmart.

Site Web décrit l'installation du site Web sur l'environnement de travail.

Sécurité propose des recommandations d'installation dans un environnement de production.

2 Environnement de travail

L'environnement de travail se compose de deux machines virtuelles sous le système d'exploitation GNU/Linux. Celles-ci vont être nommées `apache-zeppelin` et `data-sources`.

La première machine va exploiter le logiciel Apache Zeppelin[1]; la seconde va reprendre diverses sources de données (fichiers, système de gestion de base de données relationnelles, ...). Ces deux machines communiquent dans le sous-réseau 192.168.2.0/24. Les adresses IP attribuées sont 192.168.2.168 pour `data-sources` et 192.168.2.169 pour `apache-zeppelin`. Lorsqu'un nom de domaine est nécessaire plutôt qu'une adresse IP, le domaine `umons-ig.lan` est utilisé.

Le reste de la section détaille les divers services à installer dont le système d'exploitation Ubuntu[2] et les sources de données d'exemple ainsi que le serveur Web Apache[3] et logiciel d'intégration continue Jenkins[4].

2.1 Ubuntu

Le système d'exploitation utilisé est Ubuntu Server (facilité d'utilisation dans les versions gratuites de type serveur) dans sa dernière version stable en début de projet : 18.04. Ubuntu Server 18.04.2 LTS est disponible sur `https://www.ubuntu.com/download/server`.

Suite à l'insertion du CD-ROM (ou clef USB bootable) d'installation, on obtient un menu qui permet de choisir la langue d'installation :

[1] `https://zeppelin.apache.org/`
[2] `https://www.ubuntu.com`
[3] `https://httpd.apache.org/`
[4] `https://jenkins.io/`

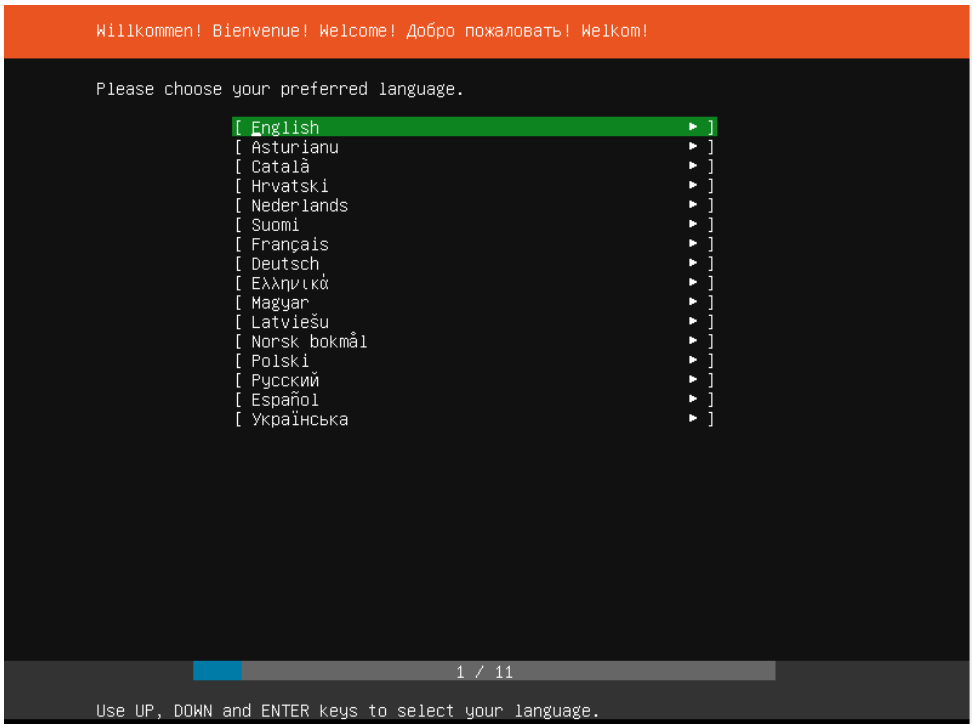

Figure 1: Ubuntu, choix de la langue.

Le menu suivant permet de sélectionner la disposition du clavier qui sera utilisée par le système.

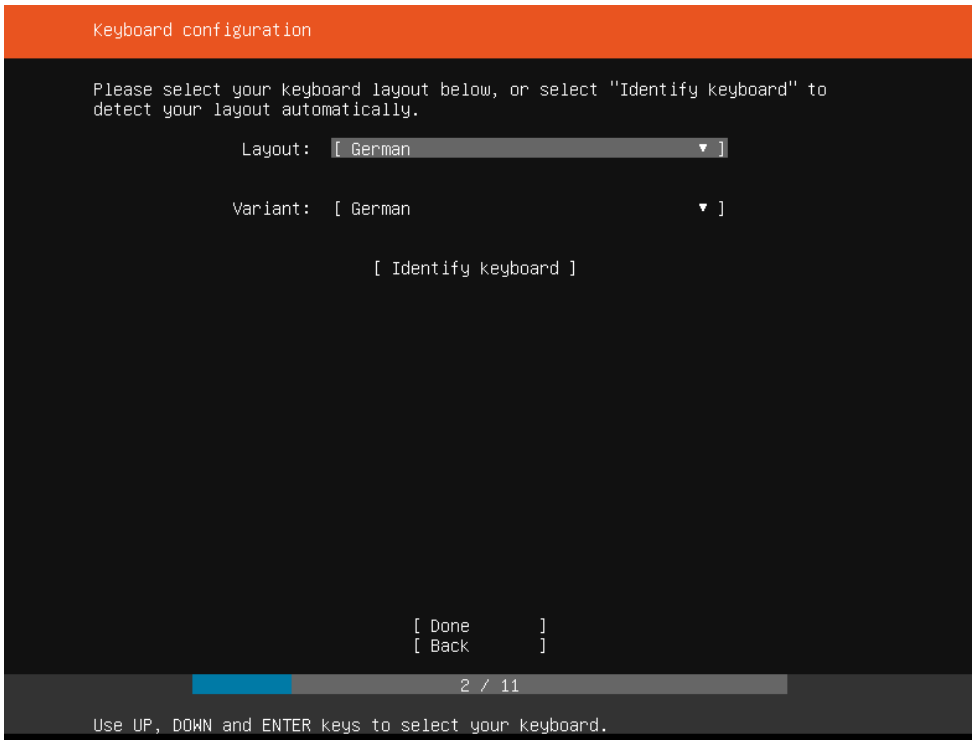

Figure 2: Ubuntu, choix de la disposition clavier.

Ensuite, il faut sélectionner le mode d'installation. Nous avons choisi le mode par défaut étant donné que nous n'avons pas de service cloud :

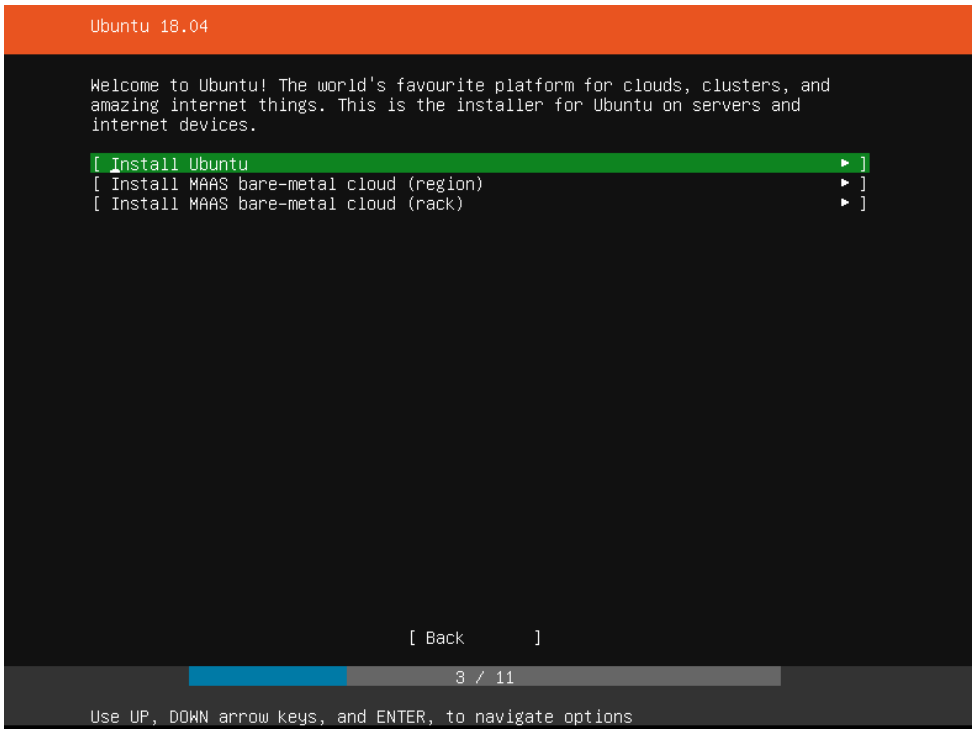

Figure 3: Ubuntu, choix du mode de fonctionnement.

L'étape 4 de l'installation consiste à configurer le réseau. Dans un premier temps, nous utilisons le mode de base (ethernet en DHCP) et nous le configurerons par la suite :

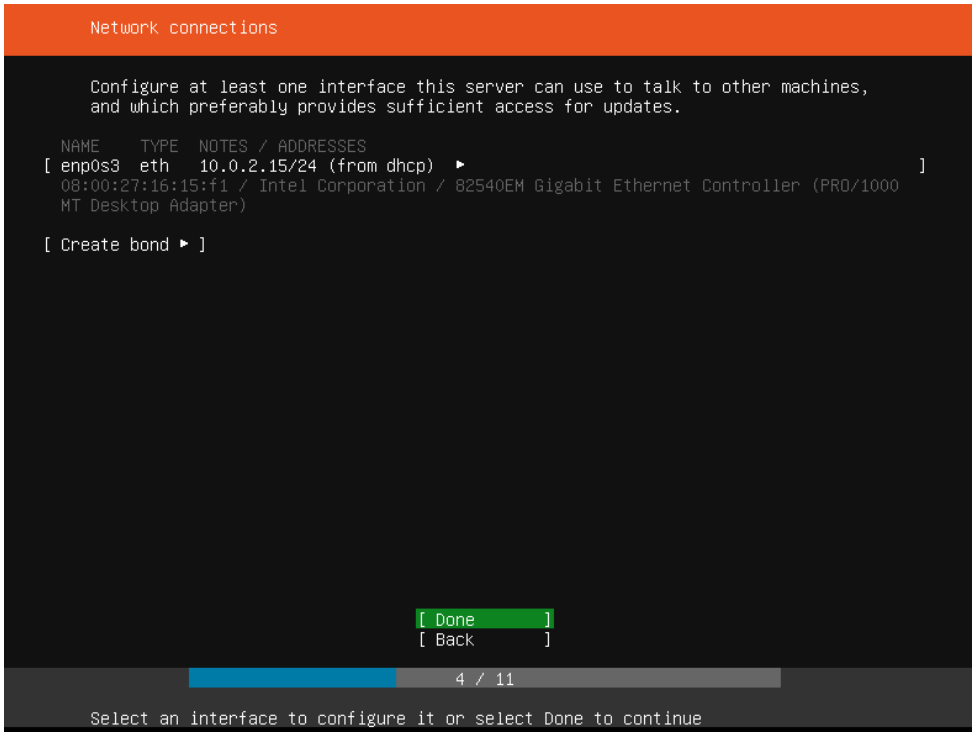

Figure 4: Ubuntu, configuration réseau.

L'étape suivante permet de définir un serveur mandataire à utiliser. Nous n'en avons pas eu besoin, il faut passer au menu suivant :

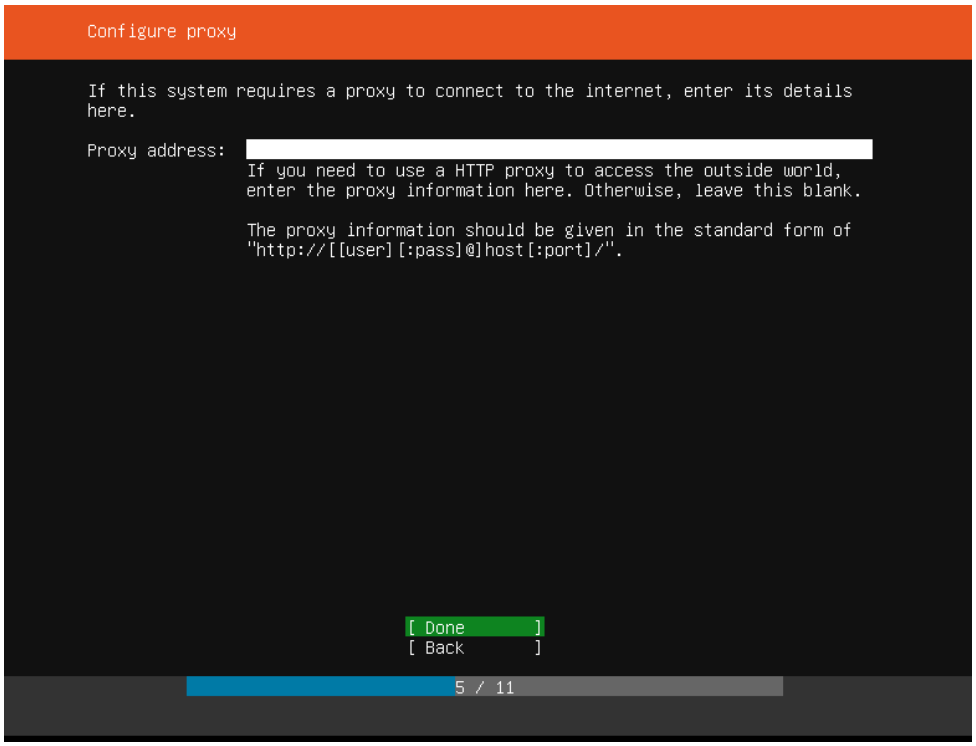

Figure 5: Ubuntu, configuration serveur mandataire.

De manière similaire, le miroir par défaut a été sélectionné :

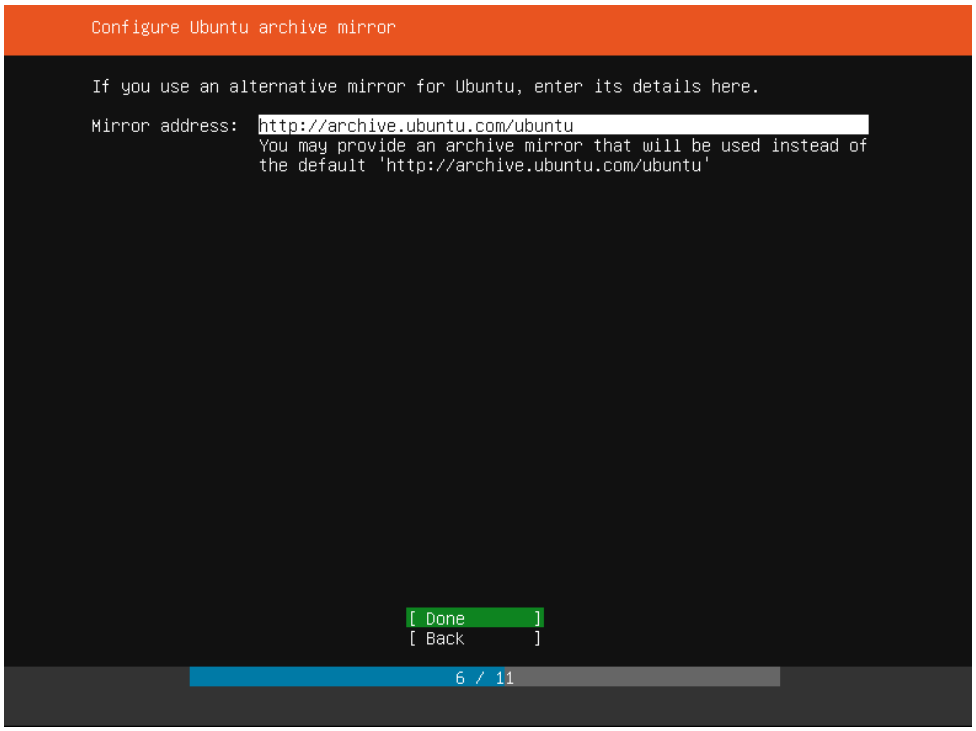

Figure 6: Ubuntu, choix du mirroir.

L'étape 7 permet de définir précisément le partitionnement à utiliser et ainsi diviser son disque en différentes partitions. A des fins de tests, cela ne nous est pas nécessaire et nous sélectionnons donc le partitionnement automatique :

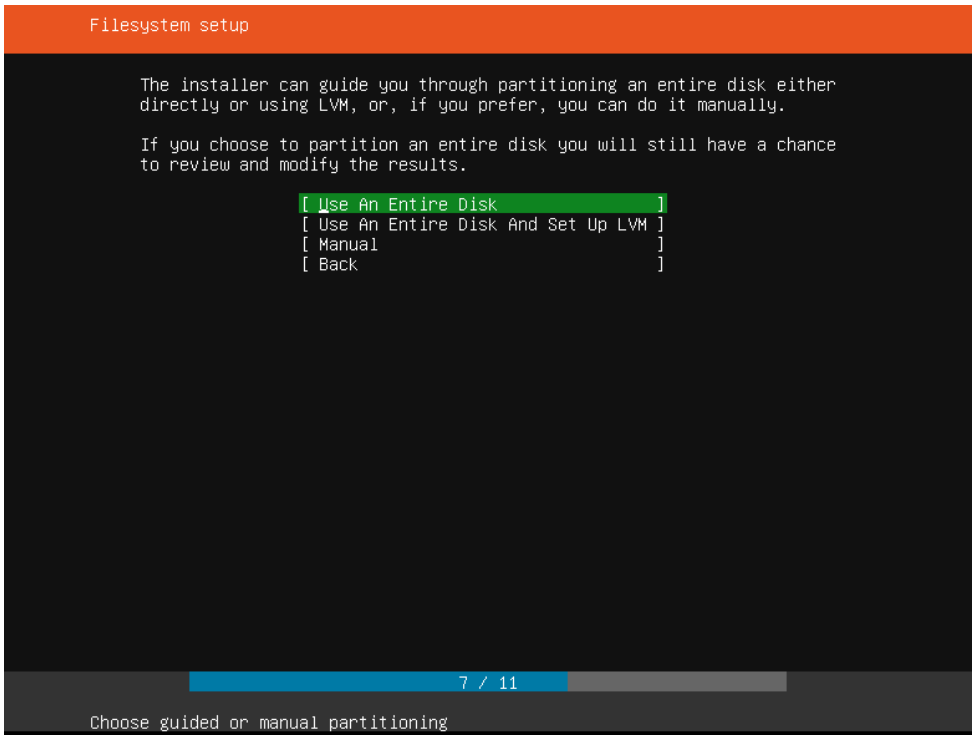

Figure 7: Ubuntu, partitionnement.

De manière similaire, il faut sélectionner le disque qui contiendra le bootloader :

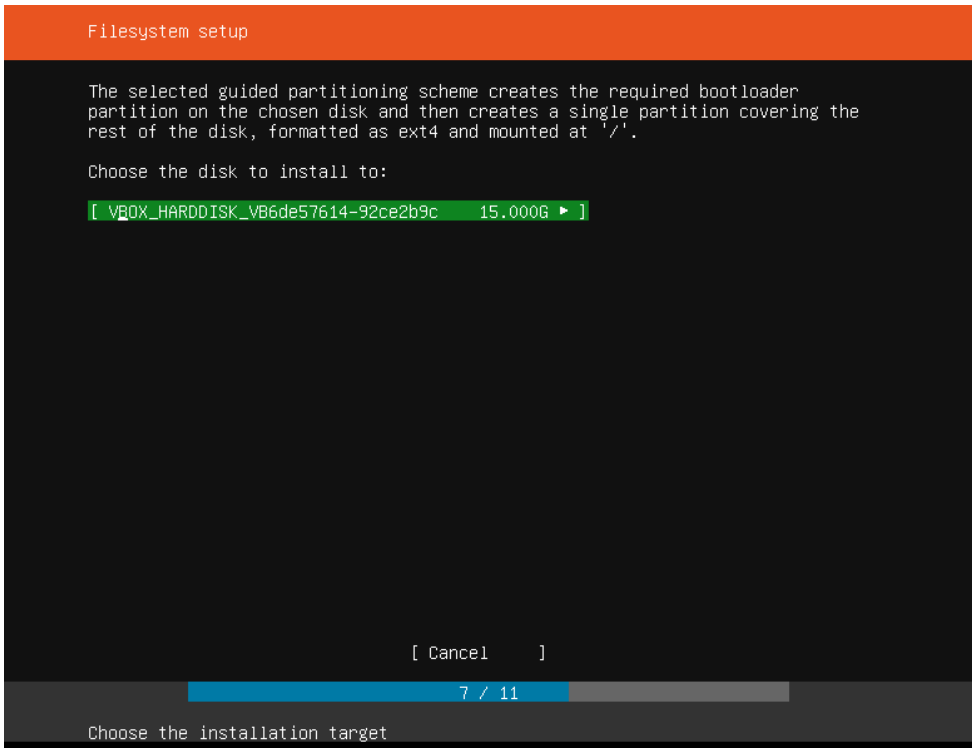

Figure 8: Ubuntu, bootloader.

Un récapitulatif du partitionnement est ensuite affiché :

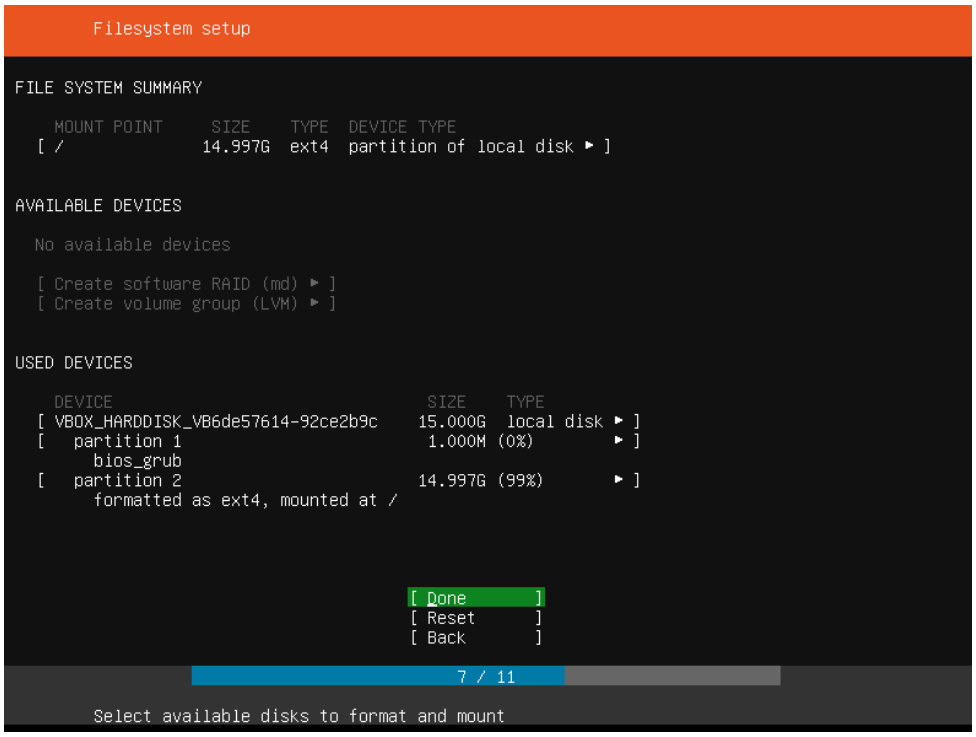

Figure 9: Ubuntu, choix de la langue.

Il faut ensuite confirmer la demande de partitionnement :

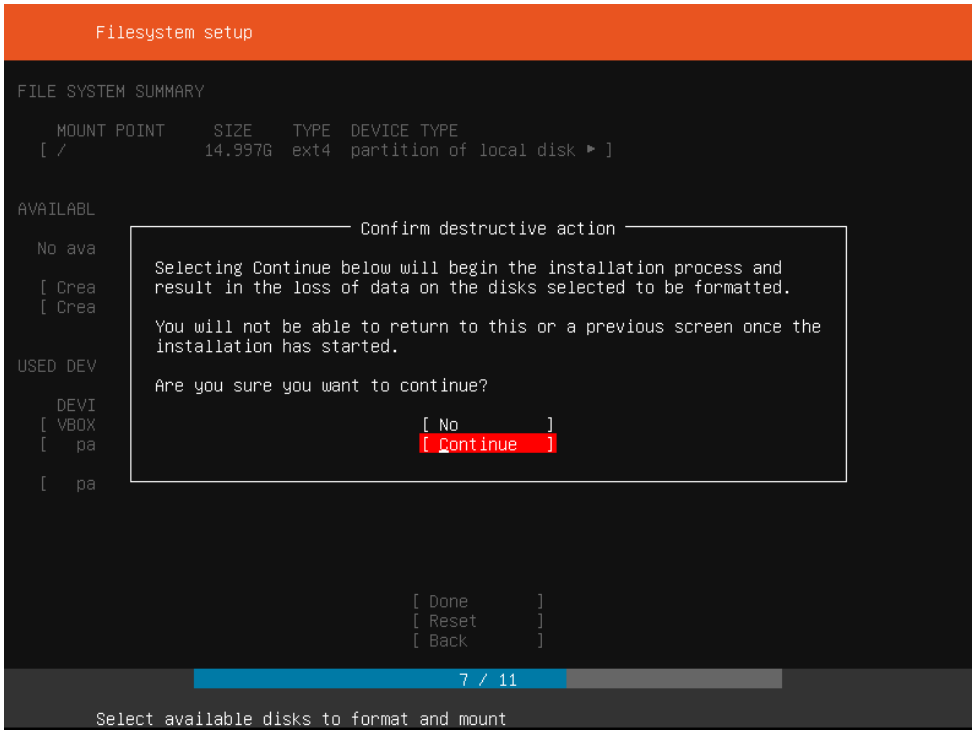

Figure 10: Ubuntu, confirmation.

Le compte utilisateur par défaut est `user`, avec la même valeur en tant que nom d'utilisateur, mot de passe et nom de serveur. Le nom de serveur sera configuré plus tard :

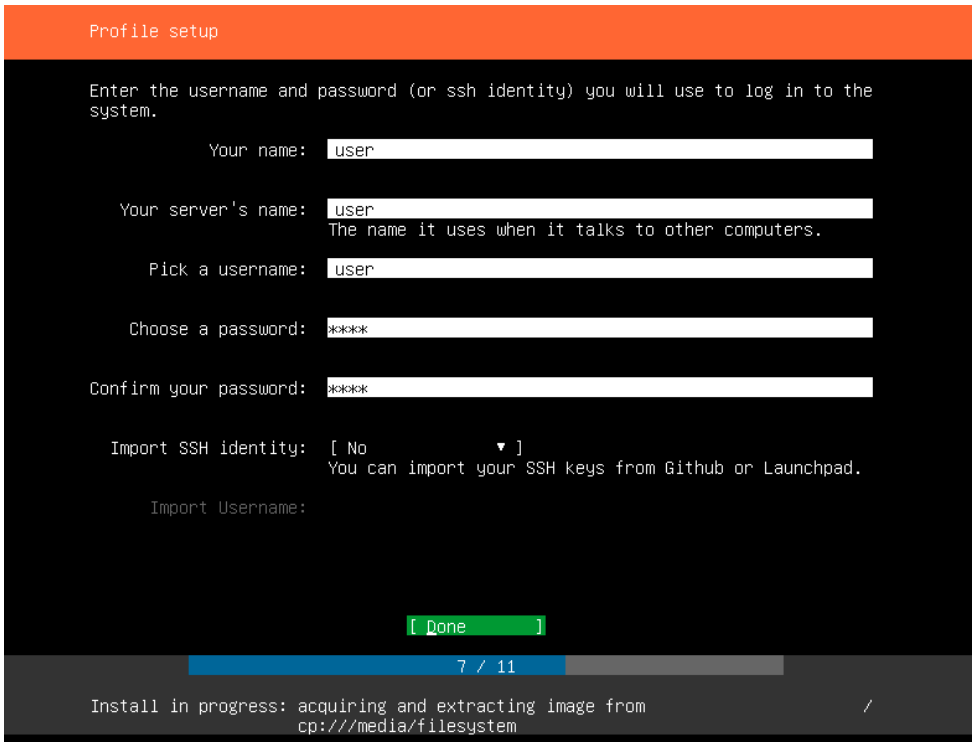

Figure 11: Ubuntu, compte utilisateur.

Il est possible d'ajouter des fonctionnalités (d'installer des services supplémentaires), cela sera effectué manuellement plus tard :

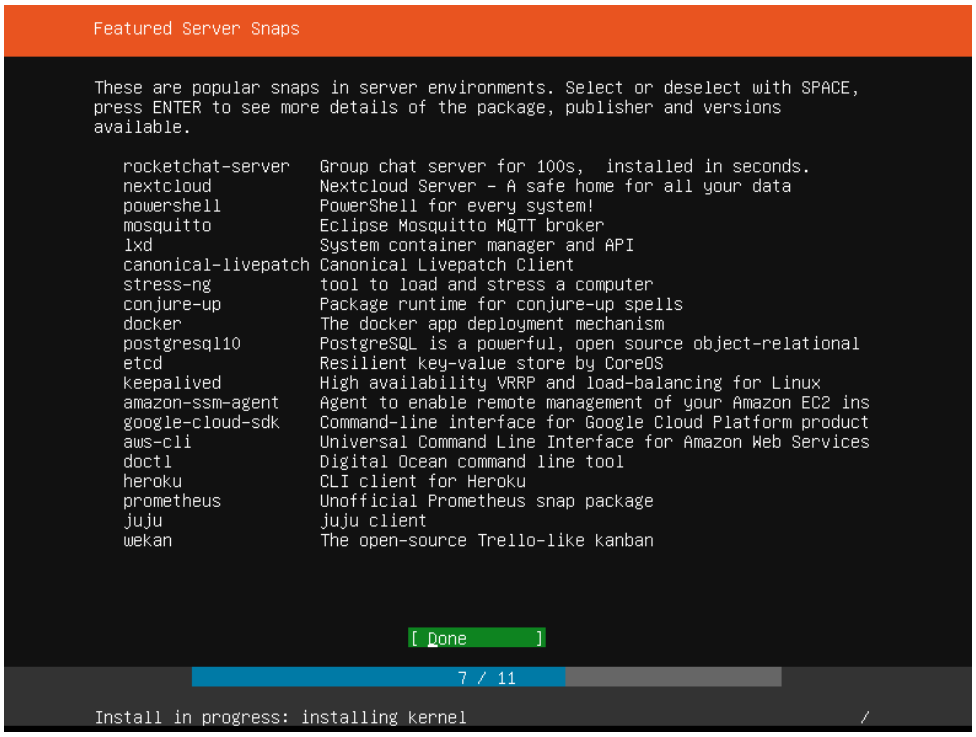

Figure 12: Ubuntu, choix des packages.

Lorsque l'installation est terminée, il faut redémarrer :

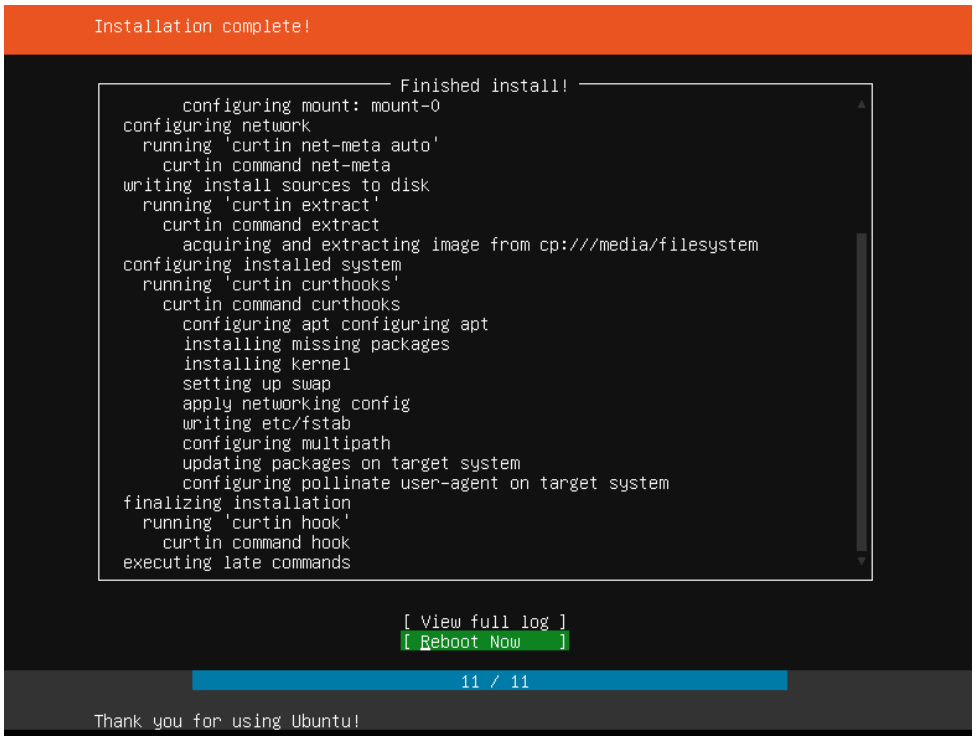

Figure 13: Ubuntu, choix de la langue.

A présent, les deux machines ont comme système d'exploitation Ubuntu 18.04 LTS avec le compte utilisateur `user`. Lors du redémarrage, une invite de commande propose de s'authentifier et d'entrer des commandes. N'ayant pas de serveur de nom de domaines (DNS), l'association adresse IP et nom de domaine se fait statiquement via le fichier `/etc/hosts`. Tout d'abord, il faut s'authentifier en tant que `user` et rentrer la commande :

```
sudo -s
```

pour devenir administrateur. Afin de le modifier sur les deux machines, il faut entrer les commandes :

```
echo "192.168.2.168   data-sources.umons-ig.lan" >> /etc/hosts
echo "192.168.2.169   apache-zeppelin.umons-ig.lan" >> /etc/hosts
```

A présent, il est important d'identifier les deux machines. A cette fin, le nom d'hôte doit être changé via la commande :

```
echo "data-sources" > /etc/hostname
```

et la commande

```
echo "apache-zeppelin" > /etc/hostname
```

Le réseau n'est pas encore configuré. Ubuntu 18.04 utilise désormais l'utilitaire `netplan`. Il faut donc modifier le fichier `/etc/netplan/50-cloud-init.yaml` de la manière suivante sur la machine `data-sources` :

```
echo "network:" > /etc/netplan/50-cloud-init.yaml
echo "   ethernets:" >> /etc/netplan/50-cloud-init.yaml
echo "      enp0s8:" >> /etc/netplan/50-cloud-init.yaml
echo "      addresses: [192.168.2.168/24]" >> /etc/netplan/50-cloud-init.yaml
echo "      gateway4: 192.168.2.1" >> /etc/netplan/50-cloud-init.yaml
echo "      dhcp4: true" >> /etc/netplan/50-cloud-init.yaml
echo "      nameservers:" >> /etc/netplan/50-cloud-init.yaml
echo "         addresses: [8.8.8.8]" >> /etc/netplan/50-cloud-init.yaml
echo "      optional: true" >> /etc/netplan/50-cloud-init.yaml
echo "   version: 2" >> /etc/netplan/50-cloud-init.yaml
```

et

```
echo "network:" > /etc/netplan/50-cloud-init.yaml
echo "   ethernets:" >> /etc/netplan/50-cloud-init.yaml
echo "      enp0s8:" >> /etc/netplan/50-cloud-init.yaml
echo "      addresses: [192.168.2.169/24]" >> /etc/netplan/50-cloud-init.yaml
echo "      gateway4: 192.168.2.1" >> /etc/netplan/50-cloud-init.yaml
echo "      dhcp4: true" >> /etc/netplan/50-cloud-init.yaml
echo "      nameservers:" >> /etc/netplan/50-cloud-init.yaml
echo "         addresses: [8.8.8.8]" >> /etc/netplan/50-cloud-init.yaml
echo "      optional: true" >> /etc/netplan/50-cloud-init.yaml
echo "   version: 2" >> /etc/netplan/50-cloud-init.yaml
```

sur la machine `data-sources`. La passerelle utilisée est l'adresse `192.168.2.1` et l'interface réseau est nommée `enp0s8`[5].

Le système est installé et configuré. Il est à présent temps de mettre à jour le système via les commande :

```
apt update
apt-get upgrade -y
```

Et de désactiver le firewall dans un soucis de simplicité. Ce point sera abordé plus tard afin de sécuriser le système :

```
ufw disable
```

Finalement, le système doit être redémarré pour prendre en compte tous les changements :

```
reboot
```

2.2 Zeppelin

L'utilisation de Zeppelin requière Java[6]. Après s'être authentifié en tant qu'administrateur sur la machine `apache-zeppelin`, il faut utiliser les commandes suivantes pour installer Java d'Oracle en version 11 :

[5] Il est possible d'obtenir son nom à l'aide du programme `ifconfig`.

[6] `https://www.oracle.com/technetwork/java/javase/downloads/index.html`

```
add-apt-repository ppa:linuxuprising/java
apt update
apt install oracle-java11-installer
```

Pour enfin passer à l'étape d'installation de Zeppelin dans le dossier `/opt` :

```
cd /root
wget http://apache.belnet.be/zeppelin/zeppelin-0.8.1/zeppelin-0.8.1-bin-all.tgz
tar xf zeppelin-*-bin-all.tgz -C /opt
mv /opt/zeppelin-*-bin-all /opt/zeppelin
rm zeppelin*.tgz
```

Une règle de bonne pratique est de cadenasser les droits. Cela est obtenu via la création d'un compte propre à Zeppelin et de lui donner les droits sur le logiciel Zeppelin situé dans `/opt/zeppelin`

```
useradd -d /opt/zeppelin -s /bin/false zeppelin
chown -R zeppelin:zeppelin /opt/zeppelin
```

Zeppelin est à présent installé mais n'est pas exécuté au démarrage. Étant un utilitaire démon à exécuter en arrière plan, il faut créer un service zeppelin via les commandes suivantes :

```
echo "[Unit]" >> /etc/systemd/system/zeppelin.service
echo "Description=Zeppelin service" >> /etc/systemd/system/zeppelin.service
echo "After=syslog.target network.target" >> /etc/systemd/system/zeppelin.service
echo "" >> /etc/systemd/system/zeppelin.service
echo "[Service]" >> /etc/systemd/system/zeppelin.service
echo "Type=forking" >> /etc/systemd/system/zeppelin.service
echo "ExecStart=/opt/zeppelin/bin/zeppelin-daemon.sh start"
   >> /etc/systemd/system/zeppelin.service
echo "ExecStop=/opt/zeppelin/bin/zeppelin-daemon.sh stop"
   >> /etc/systemd/system/zeppelin.service
echo "ExecReload=/opt/zeppelin/bin/zeppelin-daemon.sh reload"
   >> /etc/systemd/system/zeppelin.service
echo "User=zeppelin" >> /etc/systemd/system/zeppelin.service
echo "Group=zeppelin" >> /etc/systemd/system/zeppelin.service
echo "Restart=always" >> /etc/systemd/system/zeppelin.service
echo "" >> /etc/systemd/system/zeppelin.service
echo "[Install]" >> /etc/systemd/system/zeppelin.service
echo "WantedBy=multi-user.target" >> echo /etc/systemd/system/zeppelin.service
```

Le service est créé mais il n'est pas automatiquement exécuté au démarrage, ce qui se fait via la commande :

```
systemctl enable zeppelin
```

Et peut être exécuté directement via la commande :

```
systemctl start zeppelin
```

Ensuite, il faut ajouter et configurer les différents interpréteurs comme sur les figures suivantes :

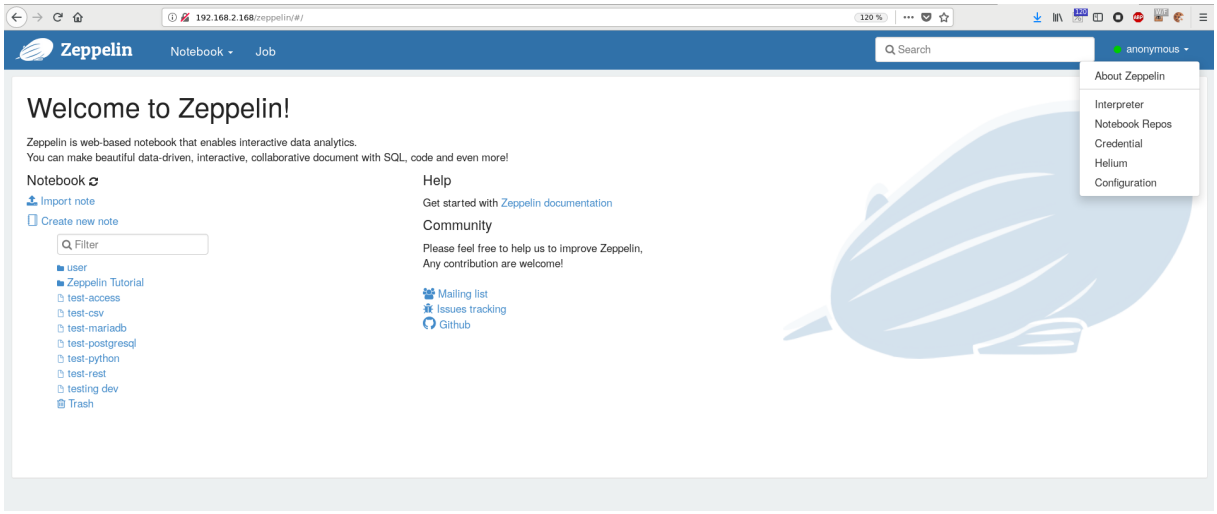

Figure 14: Zeppelin, accès aux interpréteurs.

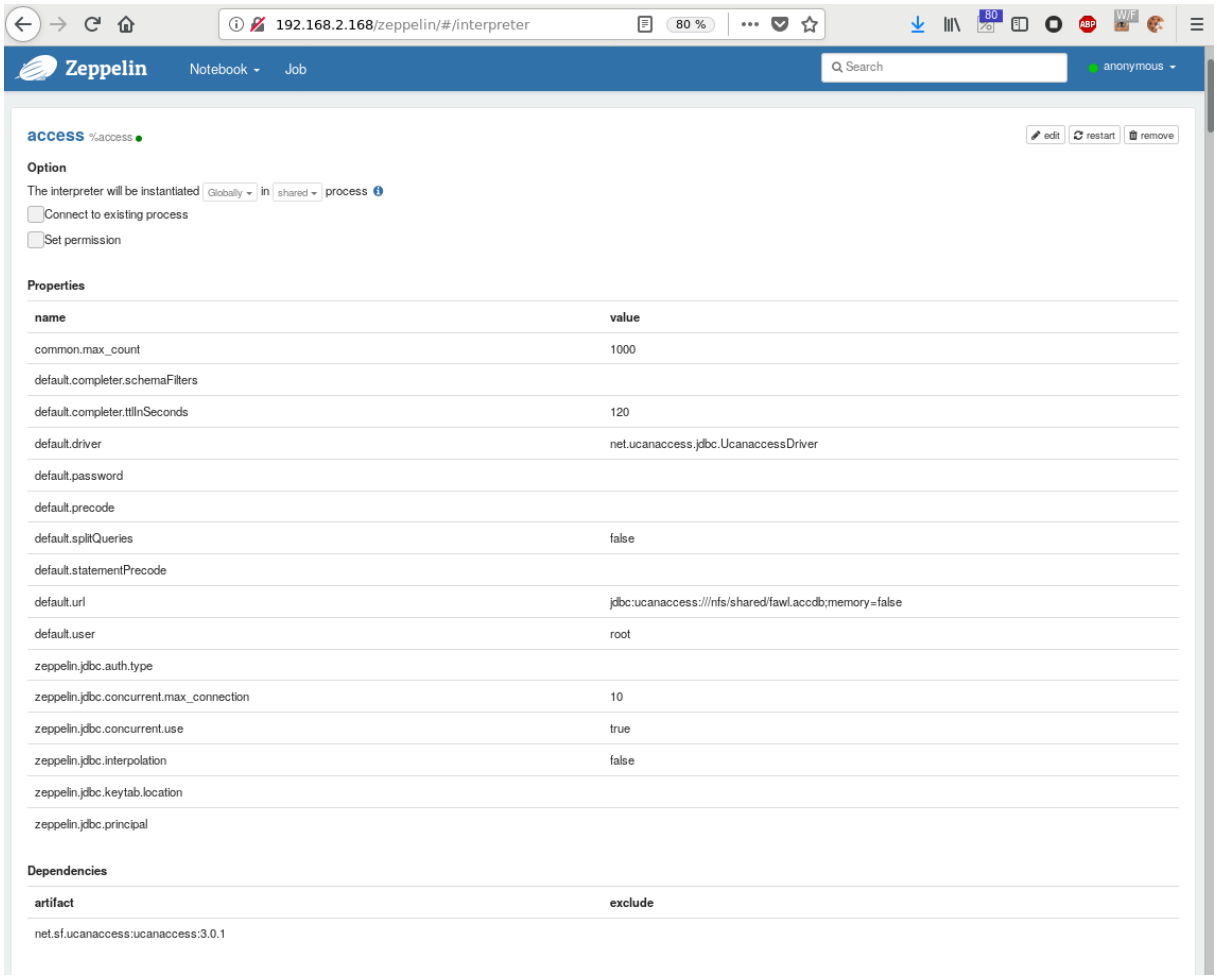

Figure 15: Zeppelin, interpréteur `Access`.

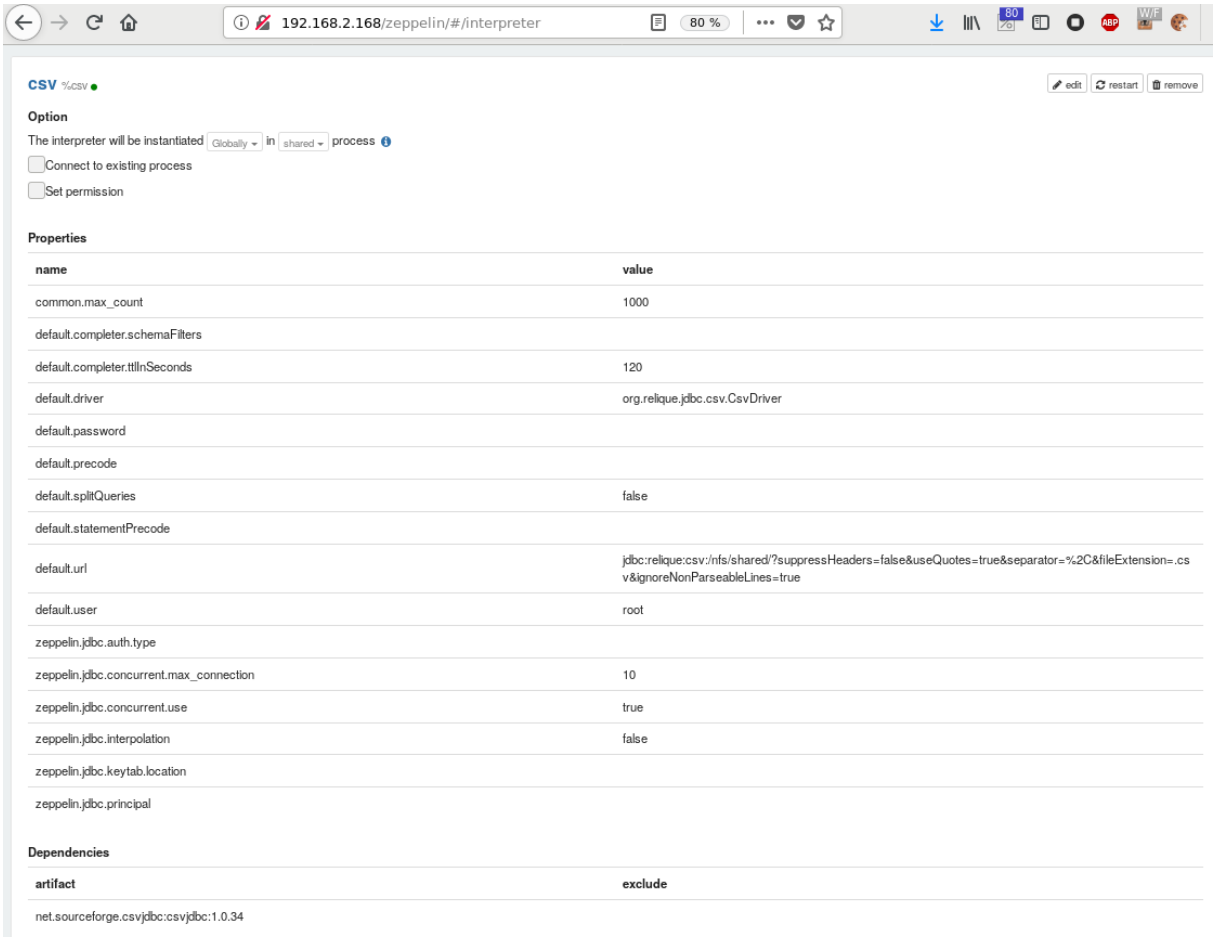

Figure 16: Zeppelin, interpréteur `CSV`.

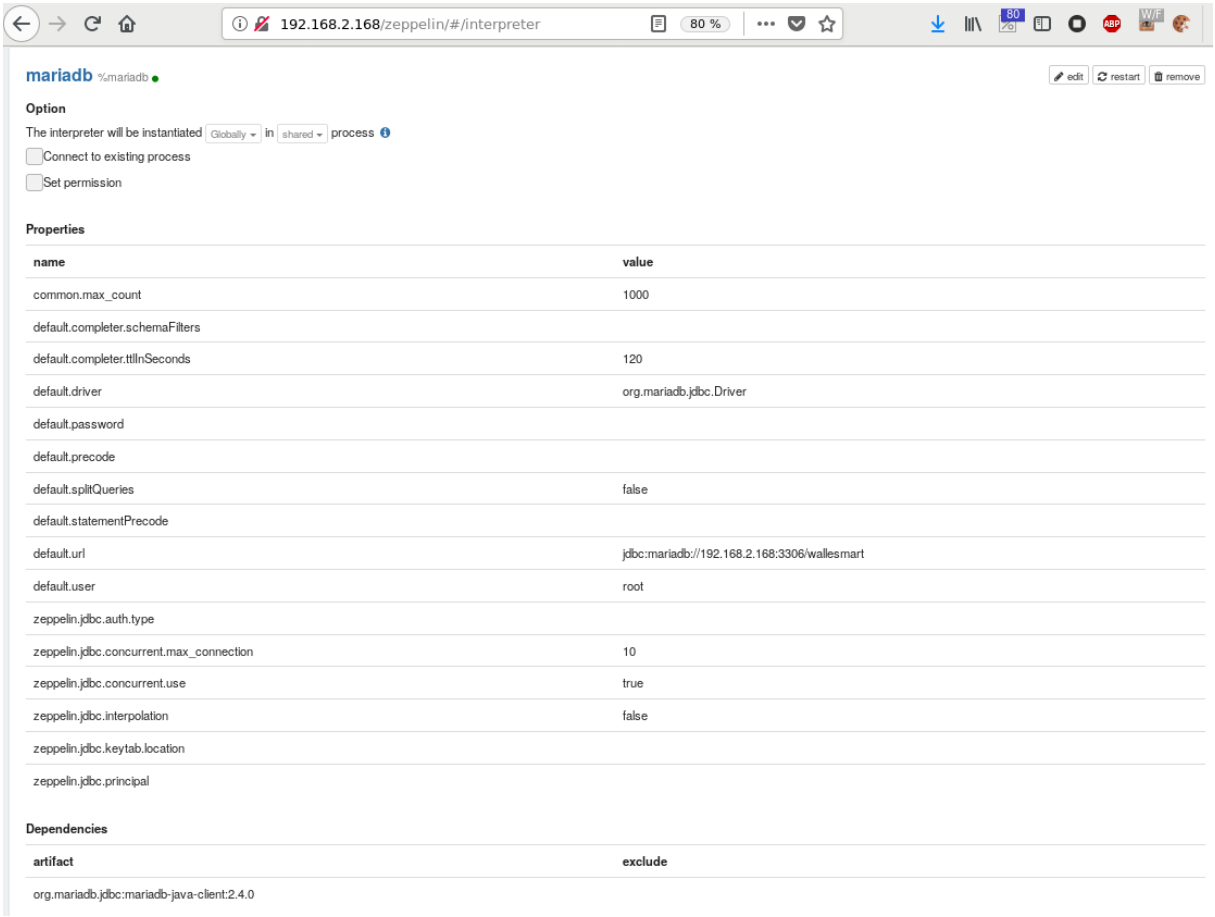

Figure 17: Zeppelin, interpréteur `MariaDB`.

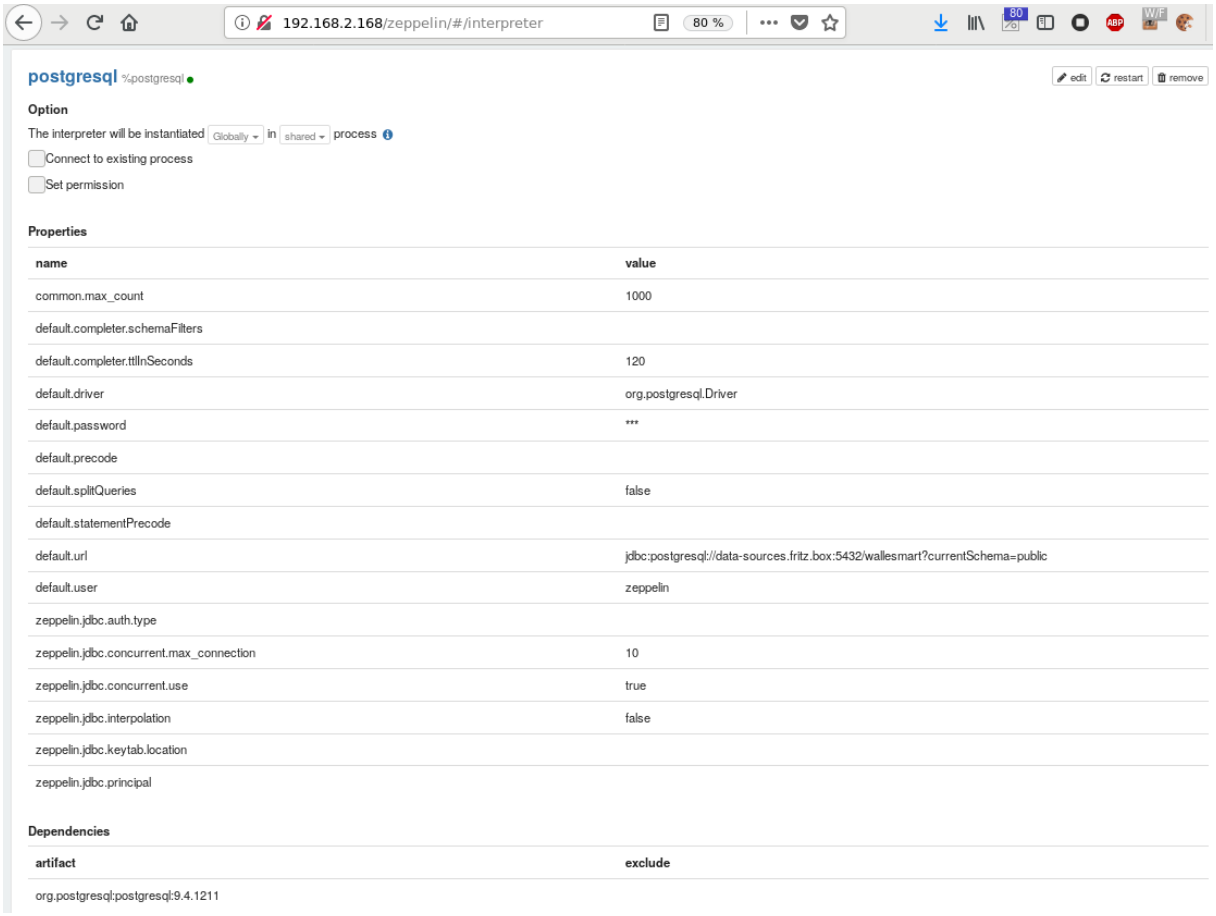

Figure 18: Zeppelin, interpréteur `PostgreSQL`.

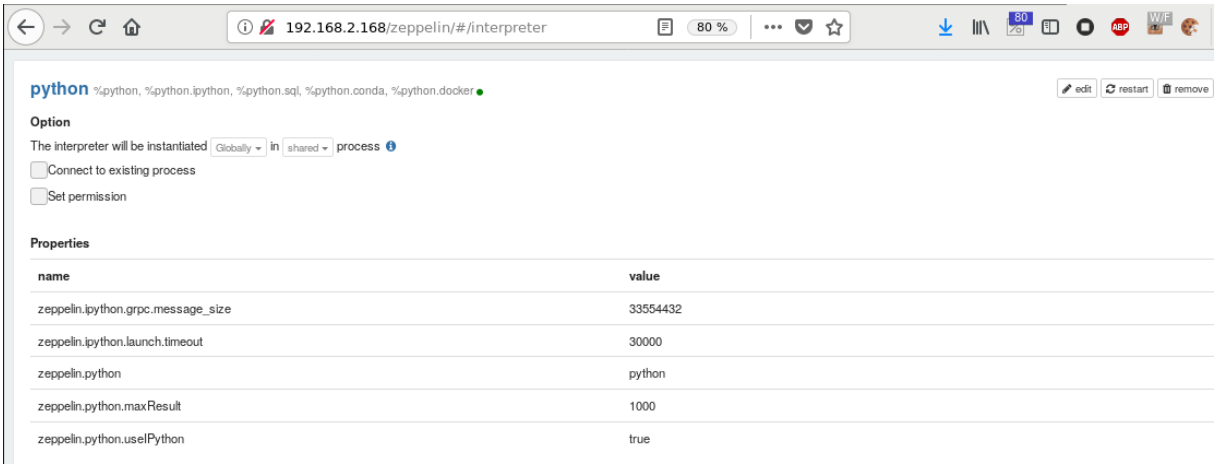

Figure 19: Zeppelin, interpréteur `Python`.

Création de sources de données d'exemple :

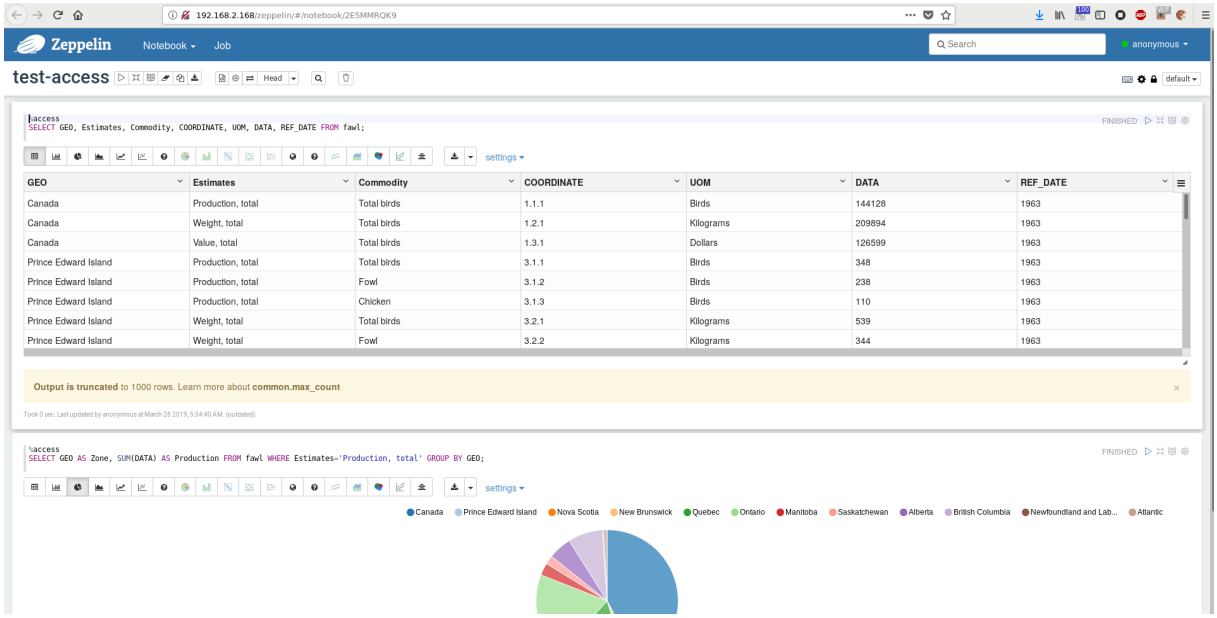

Figure 20: Zeppelin, exemple de source `Access`.

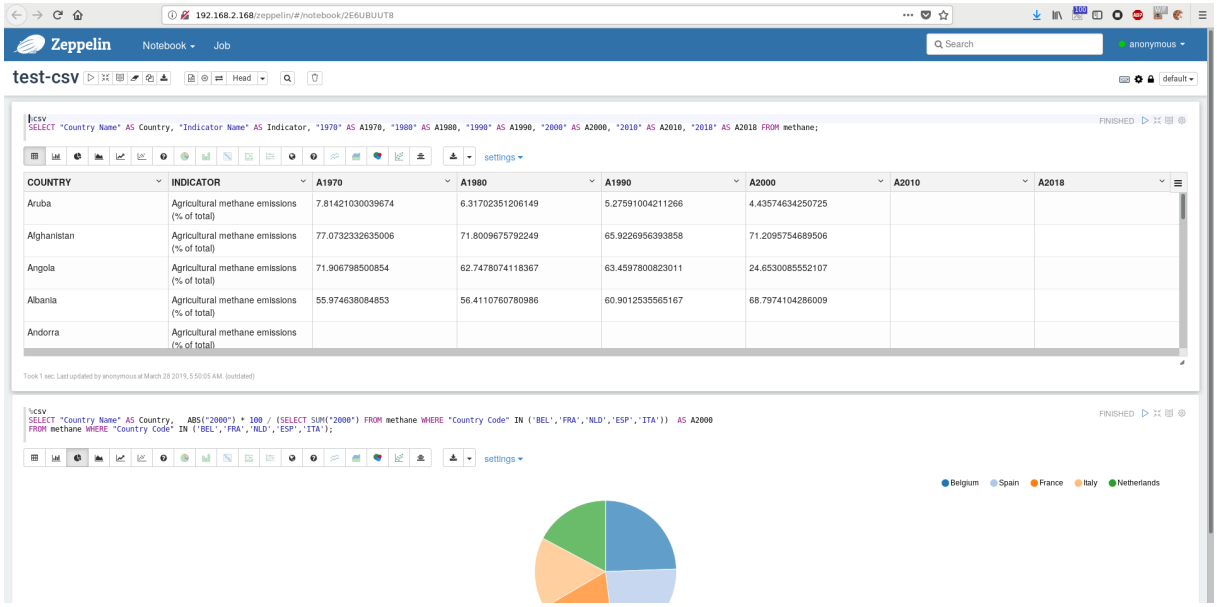

Figure 21: Zeppelin, exemple de source `CSV`.

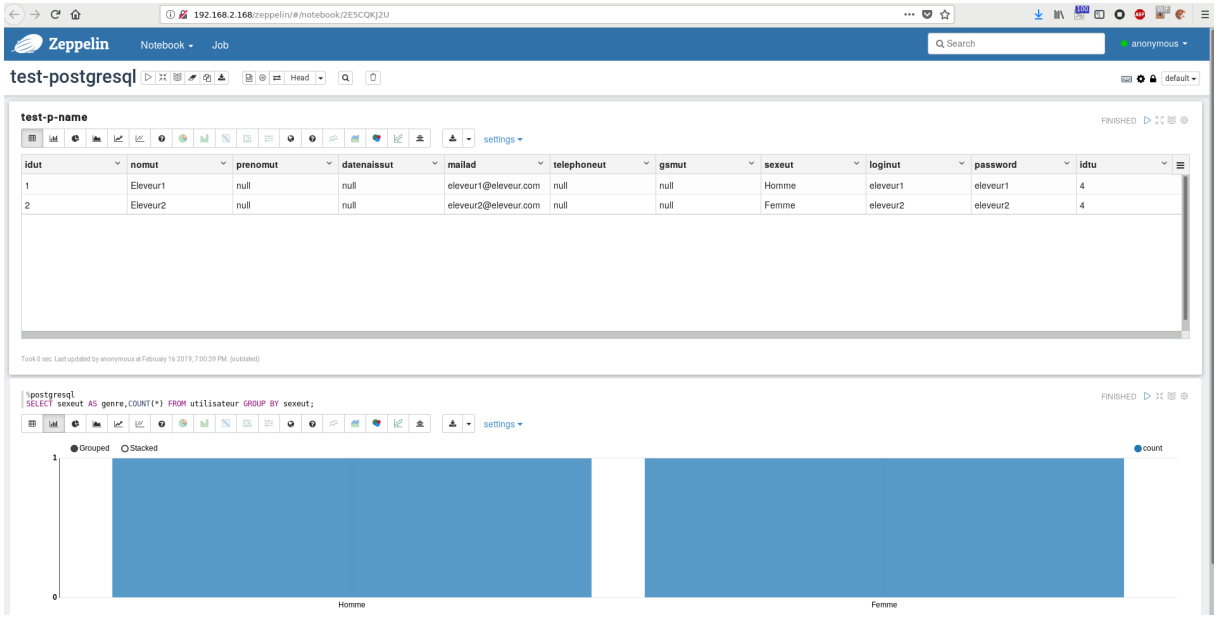

Figure 22: Zeppelin, exemple de source `PostgreSQL`.

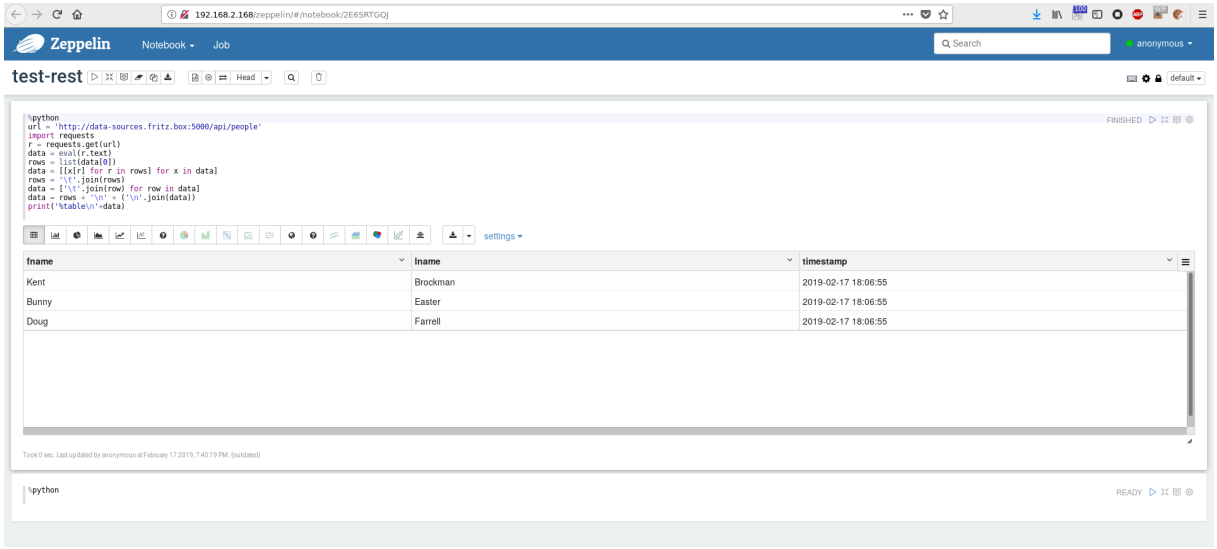

Figure 23: Zeppelin, exemple de source `REST`.

2.3 NFS

En tant qu'administrateur, il faut créer un espace de stockage sur la machine `data-sources` :

```
mkdir /shared
```

Et créer le point de montage réseau sur ce dossier :

```
echo "/shared    *(rw,sync,insecure,no_root_squash,no_subtree_check)" >>
   /etc/exports
```

Ensuite, il faut propager cette modification dans le système de fichiers via la commande :

```
exportfs -a
```

La machine `data-sources` est prête, il faut à présent monter automatiquement ce partage sur la machine `apache-zeppelin` :

```
echo "data-sources.umons-ig.lan:/   /nfs   nfs  auto 0 0" >> /etc/fstab
service autofs restart
```

2.4 Jenkins

En tant qu'administrateur, il faut ajouter le dépôt de Jenkins sur la machine `data-sources` via :

```
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key |  apt-key add -
sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
apt update
```

Il est à présent possible d'installer Jenkins :

```
sudo apt install jenkins
```

et de l'ajouter en tant que service :

```
sudo systemctl enable jenkins
sudo systemctl start jenkins
```

L'administration et configuration de Jenkins s'effectue à l'adresse : `http://192.168.2.169:8080` avec un mot de passe administrateur récupérable via la commande :

```
cat /var/lib/jenkins/secrets/initialAdminPassword
```

2.5 Apache

Sur la machine `data-sources`, exécutez la commande :

```
apt install apache2 php libapache2-mod-php php-mysql php-cli
```

en tant qu'administrateur.

2.6 Sources de données

Cette section se concentre sur l'installation de serveur de sources de données utilisées dans la *proof of concept* du projet WalleSmart. Les commandes sont à exécuter en tant qu'administrateur sur la machine `data-sources`.

2.6.1 PostgreSQL

L'installation de PostgreSQL et de phppgadmin se fait via la commande

```
apt -y install postgresql postgresql-contrib phppgadmin
```

A l'aide de l'utilitaire `nano`, il faut s'assurer que la directive d'accès *Require all granted* ne soit pas commentée :

```
nano /etc/apache2/conf-available/phppgadmin.conf
```

A l'aide de l'utilitaire `nano`, il faut s'assurer que la directive de restriction d'accès pour le compte root de `phppgadmin` soit désactivée :

```
nano /etc/phppgadmin/config.inc.php
```

Ce qui doit donner la valeur

```
$conf['extra_login_security'] = false;
```

Les services peuvent ensuite être activés et redémarrés :

```
systemctl enable postgresql
systemctl restart postgresql
systemctl restart apache2
```

2.6.2 MariaDB

Tout comme pour postgresql, MariaDB s'installa via la commande :

```
apt-get install mariadb-server mariadb-client phpmyadmin
```

A l'aide de l'utilitaire `nano`, il faut s'assurer que l'authentification utilisée soit bien *cookie*, que l'extention PHP employée soit `mysqli` et que la connexion sans mot de passe soit autorisée :

```
nano /etc/phpmyadmin/config.inc.php
```

Ce qui donne les valeurs :

```
$cfg['Servers'][$i]['auth_type'] = 'cookie';
$cfg['Servers'][$i]['extension'] = 'mysqli';
$cfg['Servers'][$i]['AllowNoPassword'] = TRUE;
```

Les services peuvent ensuite être activés et redémarrés :

```
sudo systemctl enable mariadb
sudo systemctl restart mariadb
systemctl restart apache2
```

2.6.3 REST

Tout d'abord, créons un utilisateur propre au serveur REST via la commande :

```
useradd -d /rest -s /bin/false rest
```

A l'aide du code disponible sur `https://realpython.com/flask-connexion-rest-api/`, un exemple de serveur REST peut être implémenté en Python via le code :

```
#coding: utf-8

from datetime import datetime

def get_timestamp():
    return datetime.now().strftime(("%Y-%m-%d %H:%M:%S"))

# Data to serve with our API
PEOPLE = {
    "Farrell": {
        "fname": "Doug",
        "lname": "Farrell",
        "timestamp": get_timestamp()
    },
    "Brockman": {
        "fname": "Kent",
        "lname": "Brockman",
        "timestamp": get_timestamp()
    },
    "Easter": {
        "fname": "Bunny",
        "lname": "Easter",
        "timestamp": get_timestamp()
    }
}

# Create a handler for our read (GET) people
def read():
    """
    This function responds to a request for /api/people
    with the complete lists of people

    :return:        sorted list of people
    """
    # Create the list of people from our data
    return [PEOPLE[key] for key in sorted(PEOPLE.keys())]
```

Ce code est à écrire dans le fichier `/rest/server.py` à l'aide de l'utilitaire nano :

```
nano /rest/server.py
```

Ensuite, il faut créer une page HTML comme réponse par défaut pour ce serveur. Ces pages sont chargées depuis le dossier templates :

```
mkdir /rest/templates
```

A l'aide de `nano`, le fichier `/rest/templates/home.html` doit être créé :

```
nano /rest/templates/home.html
```

et contenir :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Application Home Page</title>
</head>
<body>
    <h2>
        Hello World!
    </h2>
</body>
</html>
```

A présent, il faut attribuer les droits de l'utilisateur *rest* sur l'architecture créée :

```
chown -R rest:rest /rest/templates
```

Le serveur REST est implémenté mais il n'est pas encore exécuté. Afin de l'exécuter au démarrage, il faut soit créer un service soit ajouter une tâche à exécuter au démarrage. Cette dernière solution a été choisie et est mise en place via la commande :

```
echo "@reboot         rest    /rest/server.py &" >>  /etc/crontab
```

3 Site Web

Cette section se concentre sur la mise en place du portail Web WalleSmart dans l'environnement de développement pré-installé.

3.1 Portail WalleSmart

Le portail WalleSmart peut être récupéré depuis le dépôt Github. Une fois authentifié en tant qu'utilisateur`user` :

```
git -C /home/user/git/ checkout https://github.com/umons-ig-201819/UMONS-IG-201819.git
```

Le groupe propriétaire doit être changé afin que l'utilisateur *www-data* d'Apache puisse exécuter le code PHP :

```
chown -R user:www-data /home/user/git/
```

Lors du développement, une mise à jour régulière (toutes les 5 minutes) peut être mise en place via la commande :

```
echo "*/5 *   * * *   user    git -C /home/user/git/ pull" >> /etc/crontab
```

A présent, déplaçons le site Web par défaut d'Apache en tant qu'autre dossier et créons un lien symbolique vers le portail Web de WalleSmart :

```
mv /var/www/html /var/www/default-site
ln -s /home/user/git/WebPortal/ /var/www/html
```

3.2 Création des bases de données

Une fois sur `http://192.168.2.168/phpmyadmin`, créez une base de données `wallesmart` via le compte *root* (sans mot de passe). Chargez ensuite le script du dépôt GitHub (`/home/user/git/doc/final/`) `Wallesmart.sql` depuis l'importateur de fichiers de PHPMyAdmin.

De manière similaire, avec le compte *postgres* sur `http://192.168.2.168/phppgadmin/` chargez le fichier (`/home/user/git/doc/final/`) `PostgreSQL.sql`.

3.3 Reverse Proxy

Afin d'associer le dossier /zepplin du site Web vers le serveur de Zeppelin, il faut modifier la configuration d'Apache et y activer le mode de fonctionnement en reverse proxy. Pour cela, le fichier de configuration du site Web par défaut doit être édité :

Et le contenu doit ressembler à

```
<VirtualHost *:80>
        ProxyPreserveHost On
        AllowEncodedSlashes NoDecode
        ProxyRequests Off
        ProxyVia Off

        <Proxy *>
            Order deny,allow
            Allow from all
        </Proxy>

        <Location /zeppelin/ws>
                ProxyPass http://192.168.2.169:8080/ws
                ProxyPassReverse http://192.168.2.169:8080/ws
        </Location>
        <Location /zeppelin>
                ProxyPass http://192.168.2.169:8080
                ProxyPassReverse http://192.168.2.169:8080

    RewriteEngine on
    RewriteCond %{HTTP:UPGRADE} ^WebSocket$ [NC]
    RewriteCond %{HTTP:CONNECTION} Upgrade$ [NC]
    RewriteRule /ws(.*) ws://192.168.2.169:8080/ws$1 [P]
            </Location>

   RewriteRule /ws(.*) ws://192.168.2.169:8080/ws$1 [P]
        </Location>
 ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        ErrorLog ${APACHE_LOG_DIR}/error.log
```

```
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

4 Sécurité

Cette section propose quelques recommandations essentielles de sécurisation lors de la mise en production d'une version aboutie aux termes du projet de 3 ans.

Tout d'abord, il faut interdire les connexions avec les comptes administrateurs (*root*) et associer un mot de passe suffisamment complexe pour chacun des comptes utilisés dans l'environnement. Idéalement, la connexion en tant qu'administrateur sur les machines Ubuntu doivent être interdites depuis l'extérieur. Enfin, il faut réactiver le service firewall et autoriser uniquement le trafic adéquat (HTTP(S) sur les port 80, 443, 8080, 9000 ainsi que NFS).

4.1 Réseau

Le réseau devrait posséder une topologie où les machines serveurs sont facilement identifiables par rapport aux machines du personnels. Afin d'identifier les flux à autoriser et à bloquer, des couleurs ont été utilisées sur la figure suivante :

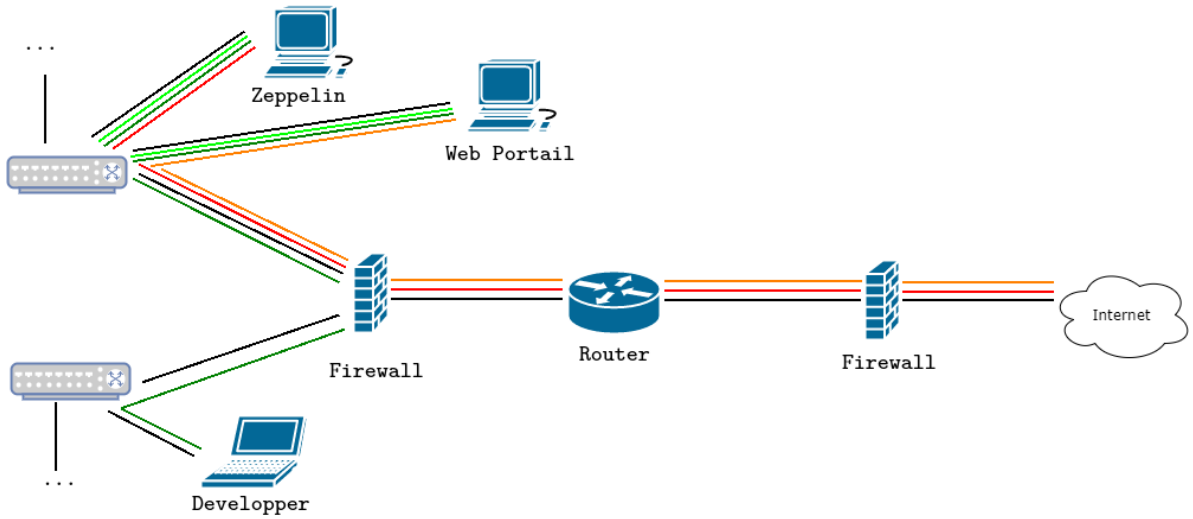

Figure 24: Sécurité, topologie proposée.

Le rouge est le trafic interdit contrairement aux autres couleurs.

4.2 Apache

La configuration d'Apache doit être adaptée afin d'activer le HTTPS et d'y inclure un certificat TLS certifié (acheté auprès d'une autorisé de certification). Le reverse proxy doit filtrer les requêtes qui sont acheminées vers Zeppelin uniquement depuis la machine source. Zeppelin recommande une configuration plus sécurisée sur `https://zeppelin.apache.org/docs/0.8.0/setup/security/authentication_nginx.html` adaptée au serveur NGinx au lieu de Apache.

En parallèle de la configuration propre à Apache, il est intéressant d'avoir un nom de domaine (configuration DNS) et de ne répondre qu'aux requêtes destinées à ce domaine. Les directives PHP doivent également être passées en revue :

- liste des modules PHP actifs
- restreindre les informations de configuration PHP (affichage)
- journaliser les erreurs mais ne pas les afficher
- limiter l'upload de fichier (taille disponible)
- désactiver l'exécution de code à distance sauf pour l'adresse de la machine Zeppelin
- limiter le nombre de ressources
- limiter la taille des données en POST
- désactiver les fonctions PHP inutilisées et qui permettent de gagner des privilèges
- limiter l'accès du système de fichiers pour les scripts PHP
- activer SELinux

4.3 Zeppelin

Zeppelin permet de gérer les accès utilisateurs. Il est donc recommandé de créer un compte utilisateur Zeppelin par utilisateur du site Web afin de cadenasser les droits de chaque utilisateur. De plus, les développeurs peuvent avoir des accès privilégiés via Zeppelin et peuvent ainsi tester de nouvelles sources de données avant de les publier sur le portail Web de WalleSmart.

5 Conclusion

Ce document a permis de retracer les étapes d'installation de la *Proof of concept* du projet WalleSmart. Ces étapes d'installation comprennent l'environnement de développement ainsi que les différents services nécessaires à son fonctionnement et la mise en place du portail Web.

Nous étendons le cadre de travail afin de proposer une sécurisation de la solution aboutie qui sera fournie aux termes des trois ans de projet.