

INSTITUT BURKINABÉ DES ARTS ET DES MÉTIERS



PROJET DE VIRTUALISATION ET CLOUD COMPUTING

Présenté par : TRAORE Teli Bafima Martin Daouda
BAZIE Alex Damien
KONATE Adam Salim
KONATE Karim
NIKIEMA Guy Arthur

Professeur : M. Flavien SOMDA

PLAN DE PRESENTATION DU PROJET

PLAN DE PRESENTATION DU PROJET	2
Introduction.....	3
Phase 1 : Préparation et Planification	4
1. Formation des groupes et distribution des rôles	4
2. Conception de l'architecture	4
3. Choix des technologies et outils	5
4. Choisissez les outils de provisionnement et de gestion des versions	6
Phase 2 : Configuration des machines virtuels	8
1. Création des vagrantfiles	8
2. Provisionnement automatique	14
3. Exécution de notre fichier vagrantfiles.....	17
4. Connexions aux machines virtuelles et vérifications des instances	19
Phase 3 : Configuration des machines virtuels	22
1. Développement de l'application web	22
2. Déploiement de l'application.....	22
3. Test de l'application à travers le client	26
Phase 4 : Présentation des problèmes rencontrés et les solutions apportées.....	29
1. Problème de connexion	29
2. Problème de temps d'exécution du Vagrantfile trop long	29
3. Problème d'accès aux machines virtuelles après l'exécution du Vagrantfile.....	30
4. Problème : Surcharge et désorganisation du Vagrantfile	30
Conclusion	30

Introduction

Dans le cadre de notre formation à l'Institut Burkinabé des Arts et des Métiers, nous avons entrepris un projet ambitieux de virtualisation et cloud computing. Ce projet vise à nous familiariser avec les technologies modernes de gestion d'infrastructures informatiques, tout en développant des compétences pratiques essentielles pour notre future carrière dans le domaine des technologies de l'information.

La virtualisation est une technologie qui permet de créer plusieurs environnements simulés ou virtuels à partir d'un seul système physique. Elle est devenue incontournable dans le monde de l'informatique pour sa capacité à optimiser l'utilisation des ressources matérielles, réduire les coûts et améliorer la flexibilité et l'agilité des environnements IT. Le cloud computing, quant à lui, est l'évolution naturelle de la virtualisation, offrant des ressources informatiques à la demande via Internet.

Ce projet se déroule en plusieurs phases, allant de la préparation et la planification, à la configuration des machines virtuelles, en passant par le développement et le déploiement d'une application web. Nous avons choisi de travailler avec des outils et technologies de pointe tels que Vagrant, Nginx, Django, et PostgreSQL, pour concevoir une architecture multi-tiers robuste et évolutive.

L'objectif principal de ce projet est de nous permettre de maîtriser les concepts de base de la virtualisation et du cloud computing, tout en nous confrontant aux défis pratiques de la mise en œuvre de ces technologies. En travaillant en équipe, nous avons également développé des compétences en gestion de projet, en collaboration et en résolution de problèmes.

En somme, ce projet de virtualisation et cloud computing est une opportunité précieuse pour renforcer nos connaissances théoriques par la pratique et nous préparer aux exigences du marché du travail dans un domaine en constante évolution.

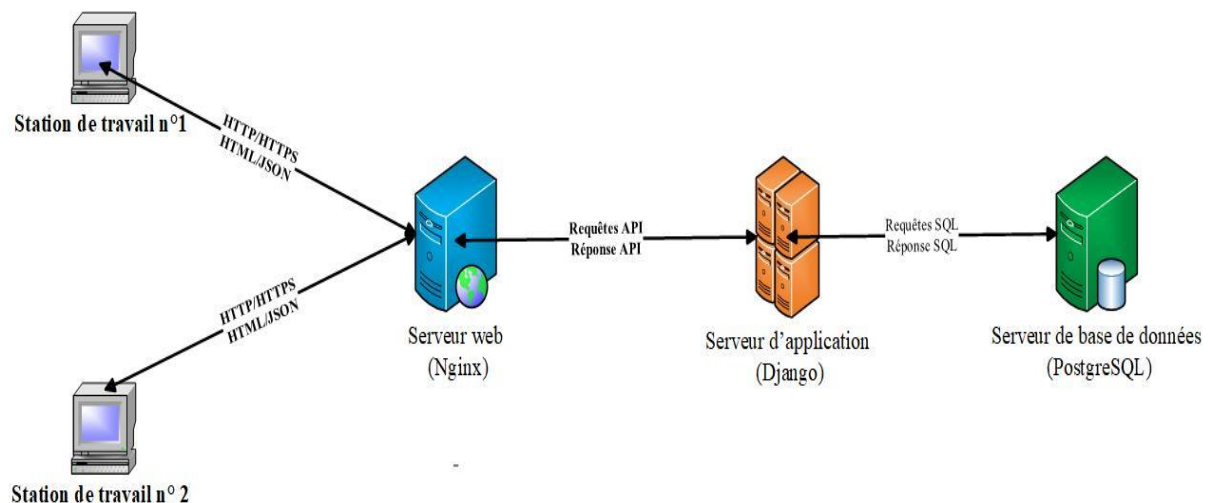
Phase 1 : Préparation et Planification

1. Formation des groupes et distribution des rôles

Pour la bonne marche de nos travaux de recherche, nous sommes constitués de 5 membres à savoir :

Nom & prénom (s)	Filières	Rôles spécifiques
TRAORE Bafima Teli Martin Daouda	MISIE	Chef de projet
KONATE Salim Adam	MISIE	Analyste
KONATE Karim	MISI	Testeur
BAZIE Alex Damien	MISI	Administrateur système
NIKIEMA Guy Arthur	MISIE	Développeur

2. Conception de l'architecture



Architecture de l'Application Web Multi-Tiers

Nous allons faire une description de notre architecture :

- Description du Flux de Requêtes : les clients envoient des requêtes HTTP/HTTPS au serveur web Django, qui les traite et les transmet au serveur d'application pour le traitement des données.
- Interaction Serveur d'Application - Base de Données : le serveur d'application Django communique avec la base de données MySQL/PostgreSQL via des requêtes SQL pour récupérer, mettre à jour ou supprimer des données selon les demandes des utilisateurs.

- Retour des Résultats aux Clients : une fois que le serveur d'application Django a traité les données, il renvoie les réponses sous forme de pages HTML ou de données JSON aux clients via le serveur web, assurant ainsi une expérience utilisateur fluide.
- Sécurité et Gestion des Sessions : le serveur web Django gère la sécurité et les sessions des utilisateurs, assurant l'authentification et l'autorisation appropriées avant de permettre l'accès aux données sensibles stockées dans la base de données.
- Redondance et Scalabilité : pour garantir la redondance et la scalabilité, le déploiement utilise des configurations de serveur web et d'application Django réparties sur plusieurs instances, connectées à une base de données centralisée pour assurer une performance optimale et une disponibilité continue.

3. Choix des technologies et outils

L'architecture multi-tiers de l'application web présentée dans le diagramme utilise trois principaux composants : un serveur web Nginx, un serveur d'application Django, et une base de données PostgreSQL. Cette séparation des préoccupations permet une gestion et une maintenance plus faciles de chaque composant. Le serveur web Nginx gère les requêtes HTTP/HTTPS des clients, sert les contenus statiques et agit comme un reverse proxy pour le serveur d'application. Le serveur d'application Django contient la logique métier de l'application, traite les requêtes entrantes, exécute la logique de l'application et interagit avec la base de données. Le serveur de base de données PostgreSQL stocke les données de manière persistante et fournit des capacités de gestion de base de données relationnelle. Cette séparation permet une meilleure évolutivité et flexibilité, car chaque composant peut être mis à jour ou remplacé indépendamment des autres.

En utilisant Nginx comme serveur web et reverse proxy, l'application bénéficie de meilleures performances et de scalabilité. Nginx est réputé pour sa capacité à gérer un grand nombre de connexions simultanées avec une utilisation minimale des ressources. Il peut également équilibrer la charge entre plusieurs instances de serveur d'application Django, permettant ainsi de gérer une augmentation du nombre de requêtes sans compromettre les performances.

La séparation des différentes couches de l'application améliore également la sécurité globale. Le serveur web Nginx peut être configuré pour gérer les certificats SSL/TLS, offrant ainsi des communications sécurisées entre les clients et le serveur. Django inclut des fonctionnalités de sécurité intégrées, telles que la protection contre les attaques CSRF (Cross-Site Request

Forgery) et XSS (Cross-Site Scripting). Les accès à la base de données PostgreSQL peuvent être restreints au serveur d'application uniquement, réduisant ainsi la surface d'attaque.

Enfin, en utilisant PostgreSQL comme base de données, l'application profite des fonctionnalités avancées de gestion des transactions et de la concurrence offertes par PostgreSQL. Cela assure l'intégrité des données même en cas d'accès simultané. Cette architecture multi-tiers utilisant Nginx, Django et PostgreSQL offre ainsi de nombreux avantages en termes de séparation des préoccupations, performances, scalabilité, sécurité et gestion des données. Elle permet de créer une application robuste, évolutive et maintenable, capable de répondre aux besoins des utilisateurs tout en offrant une base solide pour le développement futur.

4. Choisissez les outils de provisionnement et de gestion des versions

Les exemples d'outils de provisionnement :

Bash

- **Description** : Un interpréteur de commandes utilisé pour écrire des scripts shell. Les scripts Bash sont souvent utilisés pour automatiser des tâches simples de configuration et de provisionnement.
- **Utilisation** : Automatisation de la configuration des systèmes, exécution de commandes en série pour déployer et configurer des services.

Ansible

- **Description** : Un outil de gestion de configuration, de déploiement d'applications, et d'orchestration d'infrastructure qui est agentless et utilise SSH pour communiquer avec les serveurs. Il est simple à utiliser grâce à sa syntaxe basée sur YAML.
- **Utilisation** : Provisionnement de serveurs, automatisation des tâches d'infrastructure, gestion de configuration.

Terraform

- **Description** : Un outil d'infrastructure en tant que code (IaC) qui permet de créer, mettre à jour et versionner des configurations d'infrastructure pour divers fournisseurs de cloud.

- **Utilisation** : Provisionnement d'infrastructures cloud, gestion multi-cloud, automatisation de la gestion des ressources.

Puppet

- **Description** : Un outil de gestion de configuration qui utilise un modèle déclaratif pour gérer l'état des systèmes. Puppet fonctionne en mode client-serveur et peut également être utilisé en mode autonome.
- **Utilisation** : Gestion de configurations complexes, automatisation de tâches répétitives.

Chef

- **Description** : Un autre outil de gestion de configuration qui utilise des scripts Ruby (recettes) pour définir l'état de l'infrastructure. Chef peut fonctionner en mode client-serveur ou standalone (Chef Solo).
- **Utilisation** : Automatisation des configurations, gestion des serveurs.

SaltStack

- **Description** : Un outil qui offre des capacités de gestion de configuration, de provisionnement, et de gestion d'infrastructure. Il supporte le modèle maître/minion ainsi que les commandes distantes.
- **Utilisation** : Gestion de configurations à grande échelle, automatisation de l'infrastructure.

Les exemples d'outils de gestion des versions :

Git

- **Description** : Le système de gestion de versions décentralisé le plus populaire, utilisé pour suivre les modifications dans le code source tout au long du cycle de développement.
- **Utilisation** : Gestion des versions du code source, collaboration entre développeurs, maintien de l'historique des modifications.

Subversion (SVN)

- **Description** : Un système de gestion de versions centralisé qui était largement utilisé avant l'adoption massive de Git.
- **Utilisation** : Suivi des versions et des modifications du code source dans un dépôt central.

1. Mercurial

- **Description** : Un autre système de gestion de versions décentralisé, similaire à Git, mais avec une approche différente de la gestion des branches et des fusions.
- **Utilisation** : Gestion des versions et des modifications du code source.

Pour notre travail, nous avons choisi :

- **Bash** pour le provisionnement, car il permet d'automatiser efficacement les tâches de configuration et de déploiement simples.
- **Git** pour la gestion des versions, en raison de sa popularité, de sa flexibilité et de ses fonctionnalités avancées pour le suivi des modifications et la collaboration.

Phase 2 : Configuration des machines virtuels

1. Création des vagrantfiles

Pour notre projet, nous avons créé un dossier sur le bureau nommé « PROJET_VAGRANT » où nous allons initialiser tous les éléments nécessaires pour assurer le bon fonctionnement de notre système. Mais avant, nous allons faire des installations par étape :

Etape 1 : installation des plugins nécessaire pour notre projet

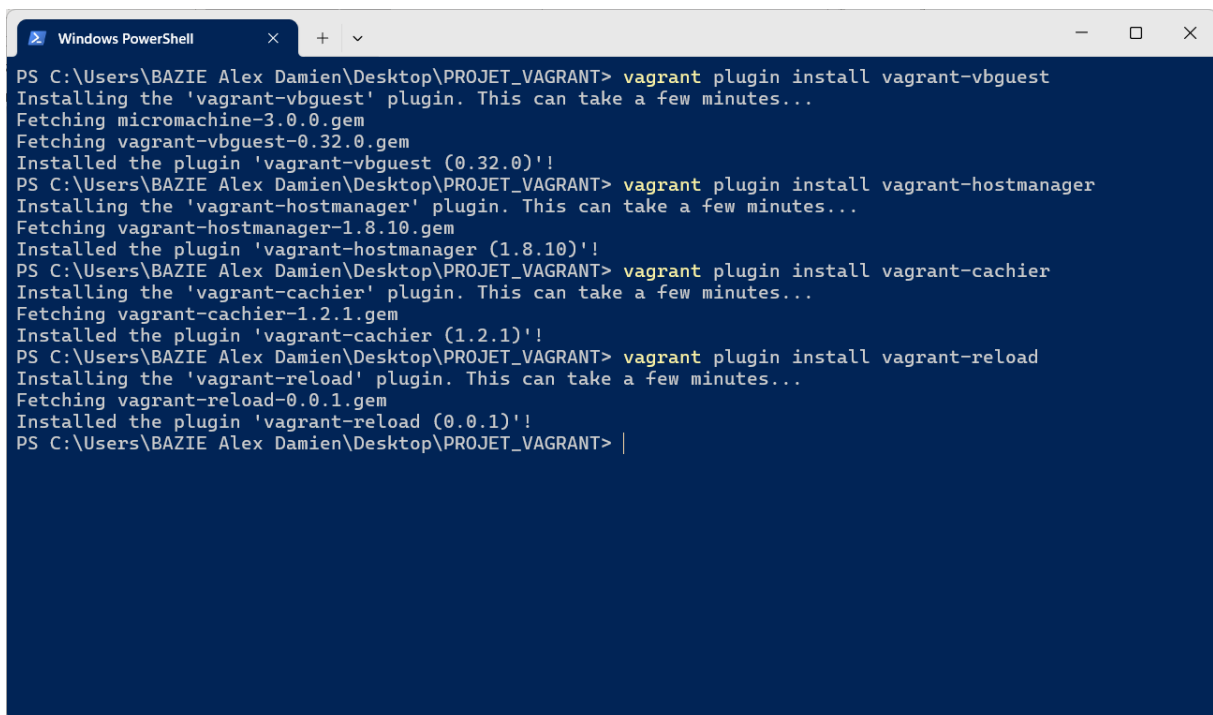
Au cours de nos recherche nous avons énumérer un certain de plugins nécessaires pour notre projet. Ils sont :

- **vagrant-vbguest** : Ce plugin assure que les "Guest Additions" de VirtualBox sont à jour, ce qui est essentiel pour le bon fonctionnement des dossiers partagés et d'autres fonctionnalités.
- **vagrant-hostmanager** : Ce plugin gère le fichier /etc/hosts des hôtes et des invités, en ajoutant des entrées pour toutes vos machines définies dans Vagrant. Cela permet une résolution DNS plus facile entre vos machines.

- **vagrant-cachier** : Ce plugin met en cache les paquets et dépendances téléchargés pour réduire les temps de téléchargement et économiser de la bande passante.
- **vagrant-reload** : Ce plugin permet de recharger la machine après le provisionnement, ce qui peut être utile si vous devez redémarrer la machine pour appliquer certaines configurations.

Procédons à l'installation de ses plugins en exécutant la commande « `vagrant plugin install` » suivie du nom du plugin :

```
vagrant plugin install vagrant-vbguest  
vagrant plugin install vagrant-hostmanager  
vagrant plugin install vagrant-cachier  
vagrant plugin install vagrant-reload
```



```
Windows PowerShell
PS C:\Users\BAZIE Alex Damien\Desktop\PROJET_VAGRANT> vagrant plugin install vagrant-vbguest
Installing the 'vagrant-vbguest' plugin. This can take a few minutes...
Fetching micromachine-3.0.0.gem
Fetching vagrant-vbguest-0.32.0.gem
Installed the plugin 'vagrant-vbguest (0.32.0)'!
PS C:\Users\BAZIE Alex Damien\Desktop\PROJET_VAGRANT> vagrant plugin install vagrant-hostmanager
Installing the 'vagrant-hostmanager' plugin. This can take a few minutes...
Fetching vagrant-hostmanager-1.8.10.gem
Installed the plugin 'vagrant-hostmanager (1.8.10)'!
PS C:\Users\BAZIE Alex Damien\Desktop\PROJET_VAGRANT> vagrant plugin install vagrant-cachier
Installing the 'vagrant-cachier' plugin. This can take a few minutes...
Fetching vagrant-cachier-1.2.1.gem
Installed the plugin 'vagrant-cachier (1.2.1)'!
PS C:\Users\BAZIE Alex Damien\Desktop\PROJET_VAGRANT> vagrant plugin install vagrant-reload
Installing the 'vagrant-reload' plugin. This can take a few minutes...
Fetching vagrant-reload-0.0.1.gem
Installed the plugin 'vagrant-reload (0.0.1)'!
PS C:\Users\BAZIE Alex Damien\Desktop\PROJET_VAGRANT> |
```

Nous allons vérifier la liste des plugins pour être sûr d'avoir installés les plugins nécessaires avec la commande :

```
vagrant plugin list
```

```
Windows PowerShell
PS C:\Users\BAZIE Alex Damien\Desktop\PROJET_VAGRANT> vagrant plugin list
vagrant-cachier (1.2.1, global)
vagrant-hostmanager (1.8.10, global)
vagrant-reload (0.0.1, global)
vagrant-vbguest (0.32.0, global)
PS C:\Users\BAZIE Alex Damien\Desktop\PROJET_VAGRANT>
```

Nous constatons la présence effective de nos plugins.

Etape 2 : installation du box vagrant pour le projet

Dans le cadre de ce projet, le box vagrant qui a été recommandé est « ubuntu/bionic64 ». Nous allons l'installer et vérifier que le box est bien n'installer avant le début des travaux a travers les commandes suivantes :

```
vagrant box add ubuntu/bionic64
vagrant box list
```

```
Windows PowerShell
PS C:\Users\BAZIE Alex Damien\Desktop\PROJET_VAGRANT> vagrant box add ubuntu/bionic64
==> box: Loading metadata for box 'ubuntu/bionic64'
    box: URL: https://vagrantcloud.com/api/v2/vagrant/ubuntu/bionic64
==> box: Adding box 'ubuntu/bionic64' (v20230607.0.1) for provider: virtualbox
The box you're attempting to add already exists. Remove it before
adding it again or add it with the '--force' flag.

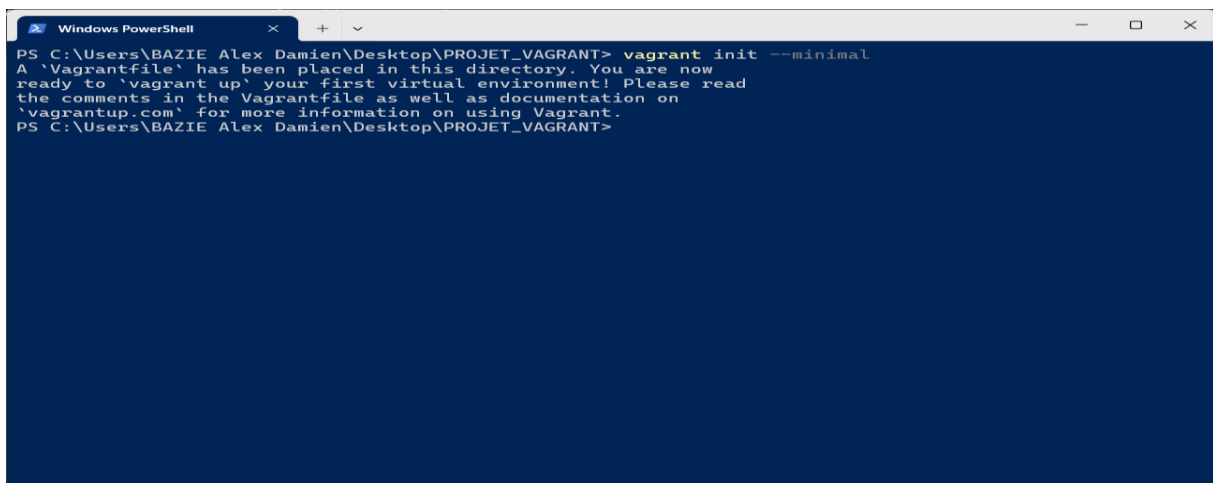
Name: ubuntu/bionic64
Provider: virtualbox
Version: 20230607.0.1
PS C:\Users\BAZIE Alex Damien\Desktop\PROJET_VAGRANT> vagrant box list
hashicorp/precise32 (virtualbox, 1.0.0)
ubuntu/bionic64    (virtualbox, 20230607.0.1)
ubuntu/focal64     (virtualbox, 20240612.0.1)
PS C:\Users\BAZIE Alex Damien\Desktop\PROJET_VAGRANT>
```

NB : le résultat de la commande pour l'installation nous notifie que nous avons le box installer et en vérifiant nous remarquons effectivement la présence du box dans la liste des box déjà installée.

Etape 3 : initialisation du fichier vagrantfiles

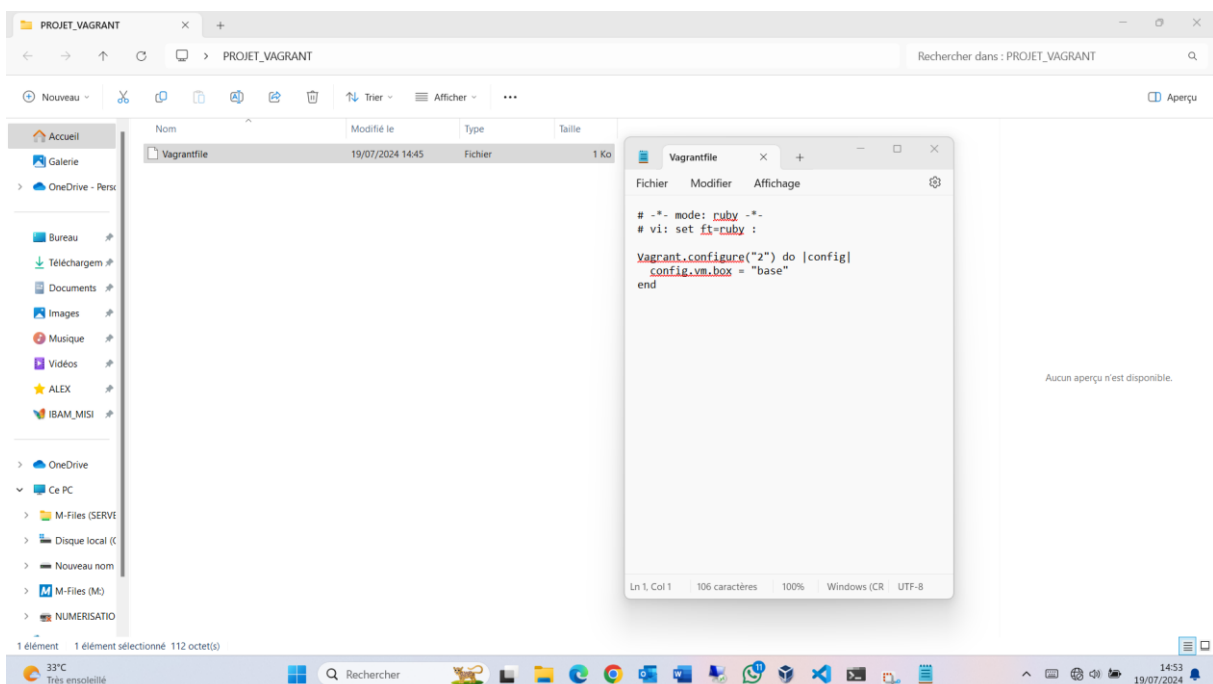
Nous allons initialiser le fichier vagrantfiles de façon minimale à travers la commande suivante :

vagrant init --minimal



```
PS C:\Users\BAZIE Alex Damien\Desktop\PROJET_VAGRANT> vagrant init --minimal
A 'Vagrantfile' has been placed in this directory. You are now
ready to 'vagrant up' your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
'vagrantup.com' for more information on using Vagrant.
PS C:\Users\BAZIE Alex Damien\Desktop\PROJET_VAGRANT>
```

Après initialisation, nous constatons la présence effective de notre fichier « vagrantfiles » avec le contenu minimal :



Etape 4 : Ecriture des scripts pour la créer et configurer (réseaux et partage de fichiers) les machines virtuelles nécessaires pour chaque composant de l'architecture :

Composant 1 : le serveur web nommé « web_server »

Pour le serveur web, nous aurons le script suivant :

```
Vagrant.configure("2") do |config|
  config.vm.boot_timeout = 600

  # Configuration du serveur web
  config.vm.define "web_server" do |web|
    web.vm.box = "ubuntu/bionic64"
    web.vm.network "private_network", ip: "192.168.33.10"
    web.vm.synced_folder ".", "/vagrant", disabled: true
    web.vm.synced_folder "./projetAlumni22/AgenceLuxe/static",
"/home/vagrant/projetAlumni22/static"
    web.vm.provider "virtualbox" do |vb|
      vb.memory = "1024"
    end
    web.vm.provision "shell", path: "provision_web_server.sh"
  end
end
```

Ce script vagrant configure une machine virtuelle pour un serveur web sous Ubuntu 18.04. Il définit un serveur appelé « web_server » avec une adresse IP privée de "192.168.33.12". La synchronisation des dossiers est activée pour rendre le répertoire de travail actuel disponible dans le dossier « /vagrant » de la machine virtuelle, et un dossier spécifique « ./projetAlumni22/AgenceLuxe/static » est synchronisé avec « /home/vagrant/projetAlumni22/static ». La machine virtuelle utilise VirtualBox avec 1024 Mo de mémoire. Enfin, un script de provisionnement (provision_web_server.sh) est exécuté pour configurer le serveur web une fois la machine virtuelle démarrée.

Composant 2 : le serveur d'application nommé « app_server »

Pour le serveur d'application, nous aurons le script suivant :

```
Vagrant.configure("2") do |config|
  config.vm.boot_timeout = 600

  # Configuration du serveur d'application
  config.vm.define "app1_server" do |app|
    app.vm.box = "ubuntu/bionic64"
```

```

app.vm.network "private_network", ip: "192.168.33.11"
app.vm.synced_folder ".", "/vagrant", disabled: true
app.vm.provider "virtualbox" do |vb|
  vb.memory = "2048"
end
app.vm.provision "shell", path: "provision_app_server.sh"
app.vm.synced_folder "./projetAlumni22", "/home/vagrant/projetAlumni22"
end
end

```

Ce script vagrant configure une machine virtuelle pour un serveur d'application sous Ubuntu 18.04. Il définit un serveur appelé « app1_server » avec une adresse IP privée de "192.168.33.10". La synchronisation des dossiers est activée pour rendre le répertoire local « ./projetAlumni22 » disponible dans le dossier « /home/vagrant/projetAlumni22 » de la machine virtuelle, tandis que la synchronisation du dossier par défaut est désactivée. La machine virtuelle utilise VirtualBox avec 2048 Mo de mémoire. Enfin, des scripts de provisionnement (provision_app_server.sh et provision_lb.sh) est exécuté pour configurer le serveur d'application une fois la machine virtuelle démarrée.

Composant 3 : le serveur de base de données nommé « db_server »

Pour le serveur de base de données, nous aurons le script suivant :

```

Vagrant.configure("2") do |config|
config.vm.boot_timeout = 600

# Configuration du serveur de base de données
config.vm.define "db_server" do |db|
  db.vm.box = "ubuntu/bionic64"
  db.vm.network "private_network", ip: "192.168.33.12"
  db.vm.synced_folder ".", "/vagrant", disabled: true
  db.vm.provider "virtualbox" do |vb|
    vb.memory = "4096"
  end
  db.vm.provision "shell", path: "provision_db_server.sh"
end
end
end

```

Ce script vagrant configure une machine virtuelle pour un serveur de base de données sous Ubuntu 18.04. Il définit un serveur appelé « db_server » avec une adresse IP privée de "192.168.33.11". La synchronisation du dossier par défaut est désactivée. La machine virtuelle utilise VirtualBox avec 4096 Mo de mémoire. Enfin, un script de provisionnement

(provision_db_server.sh) est exécuté pour configurer le serveur de base de données une fois la machine virtuelle démarrée.

2. Provisionnement automatique

Composant 1 : le fichier de provisionnement nommé « provision_web_server.sh »

Pour ce fichier, nous aurons le script suivant :

```
#!/bin/bash

# Mettre à jour les paquets et installer nginx
sudo apt-get update
sudo apt-get install -y nginx

# Configurer NGINX pour l'équilibrage de charge
sudo bash -c 'cat > /etc/nginx/sites-available/default <<EOF
upstream django_servers {
    server 192.168.33.11:8000;
}

server {
    listen 80;

    location / {
        proxy_pass http://django_servers;
        proxy_set_header Host \$host;
        proxy_set_header X-Real-IP \$remote_addr;
        proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto \$scheme;
    }
}
EOF'

# Configurer Nginx
sudo rm /etc/nginx/sites-enabled/default

cat <<EOL | sudo tee /etc/nginx/sites-available/projetAlumni22
server {
    listen 80;
    server_name 192.168.33.10;

    location / {
        proxy_pass http://192.168.33.11:8000; # Adresse et port de votre
application Django
        proxy_set_header Host \$host;
        proxy_set_header X-Real-IP \$remote_addr;
```

```

        proxy_set_header X-Forwarded-For ${proxy_add_x_forwarded_for};
        proxy_set_header X-Forwarded-Proto ${scheme};
    }

    location /static/ {
        alias /home/vagrant/projetAlumni22/static/; # Chemin vers vos
fichiers statiques Django
    }

    location /media/ {
        alias /home/vagrant/projetAlumni22/media/; # Chemin vers vos fichiers
media Django
    }
}
EOL

sudo ln -s /etc/nginx/sites-available/projetAlumni22 /etc/nginx/sites-
enabled/projetAlumni22
sudo systemctl restart nginx

```

Ce script Bash met à jour les paquets du système et installe NGINX. Ensuite, il configure NGINX pour équilibrer la charge entre les serveurs d'application. Le fichier de configuration par défaut d'NGINX est remplacé par une nouvelle configuration qui définit un pool de serveurs d'application en amont. Le script configure également un nouveau fichier de site pour le projet "projetAlumni22", définissant les règles de proxy pour acheminer les requêtes vers un serveur Django fonctionnant à l'adresse 192.168.33.10:8000. Il configure aussi des alias pour les fichiers statiques et media de Django. Enfin, le script active la nouvelle configuration du site et redémarre NGINX pour appliquer les modifications.

Composant 2 : le fichier de provisionnement nommé « provision_app_server.sh »

Pour ce fichier, nous aurons le script suivant :

```

#!/bin/bash

# Mettre à jour la liste des paquets
sudo apt-get update

# Installer pip3 s'il n'est pas déjà installé
sudo apt-get install -y python3-pip

# Installer les dépendances nécessaires
sudo apt-get install -y python3-pip libpq-dev

# Désinstaller Django

```

```

sudo pip3 uninstall -y Django

# Installer les paquets Python requis
sudo pip3 install django==2.2.28
sudo pip3 install djangorestframework
sudo pip3 install django-phonenumbers-field
sudo pip3 install psycopg2

# Installer les dépendances supplémentaires pour Pillow
sudo apt-get install -y zlib1g-dev libjpeg-dev libpng-dev libfreetype6-dev
liblcms2-dev libopenjp2-7-dev libtiff5-dev tk-dev tcl-dev

# Installer Pillow
python3 -m pip install Pillow

```

Ce script Bash commence par mettre à jour la liste des paquets disponibles du système. Il installe ensuite « pip3 », le gestionnaire de paquets pour Python 3, s'il n'est pas déjà installé, ainsi que « libpq-dev », une bibliothèque de développement pour PostgreSQL. Le script désinstalle toute version existante de Django pour éviter les conflits, puis installe une version spécifique de Django (2.2.28) et d'autres paquets Python nécessaires au projet, tels que « djangorestframework » pour la création d'API REST, « django-phonenumbers-field » pour la gestion des numéros de téléphone, et « psycopg2 » pour l'interaction avec les bases de données PostgreSQL. Enfin, le script installe les dépendances supplémentaires nécessaires pour utiliser la bibliothèque « Pillow », qui est utilisée pour la gestion des images dans Python, et installe la bibliothèque elle-même.

Composant 3 : le fichier de provisionnement nommé « provision_db_server.sh »

Pour ce fichier, nous aurons le script suivant :

```

#!/usr/bin/env bash

# Mettre à jour les paquets et installer PostgreSQL
sudo apt-get update
sudo apt-get install -y postgresql postgresql-contrib

# Configurer PostgreSQL
sudo -u postgres psql -c "CREATE USER vagrant WITH PASSWORD 'password';"
sudo -u postgres psql -c "ALTER ROLE vagrant SET client_encoding TO 'utf8';"
sudo -u postgres psql -c "ALTER ROLE vagrant SET default_transaction_isolation
TO 'read committed';"
sudo -u postgres psql -c "ALTER ROLE vagrant SET timezone TO 'UTC';"

```



```

sudo -u postgres psql -c "CREATE DATABASE myproject;"
sudo -u postgres psql -c "GRANT ALL PRIVILEGES ON DATABASE myproject TO
vagrant;"

# Modifier pg_hba.conf pour ajouter une nouvelle ligne
echo "host      all             all             192.168.33.11/32      md5" |
sudo tee -a /etc/postgresql/10/main/pg_hba.conf

# Modifier postgresql.conf pour ajouter listen_addresses = '*'
sudo sed -i "/^#listen_addresses = 'localhost'/c\listen_addresses = '*'"
/etc/postgresql/10/main/postgresql.conf

# Redémarrer PostgreSQL pour appliquer les modifications
sudo systemctl restart postgresql

```

Ce script Bash met à jour la liste des paquets du système et installe PostgreSQL ainsi que ses utilitaires supplémentaires. Il configure ensuite PostgreSQL en créant un utilisateur nommé « vagrant » avec le mot de passe « password ». Le script configure également ce rôle pour utiliser l'encodage UTF-8, une isolation de transaction par défaut en lecture commise, et le fuseau horaire UTC. Ensuite, il crée une base de données nommée « myproject » et accorde tous les privilèges sur cette base de données à l'utilisateur « vagrant ».

Pour permettre les connexions réseau à PostgreSQL, le script ajoute une ligne au fichier « pg_hba.conf » pour autoriser les connexions à partir de l'adresse IP « 192.168.33.10 » en utilisant l'authentification MD5. Il modifie également le fichier « postgresql.conf » pour permettre à PostgreSQL d'écouter sur toutes les adresses réseau en remplaçant « #listen_addresses = 'localhost' » par « listen_addresses = '*' ». Enfin, le script redémarre PostgreSQL pour appliquer ces modifications.

3. Exécution de notre fichier vagrantfiles

Après la création des scripts nous allons rassembler les fichiers de création des machines virtuelles dans un fichier unique vagrantfile :

```

# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|

  # Configuration du serveur web

```

```

config.vm.define "web_server" do |web|
  web.vm.box = "ubuntu/bionic64"
  web.vm.network "private_network", ip: "192.168.33.12"
  web.vm.synced_folder ".", "/vagrant", disabled: true
  web.vm.synced_folder "./projetAlumni22/AgenceLuxe/static",
"/home/vagrant/projetAlumni22/static"
  web.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
  end
  web.vm.provision "shell", path: "provision_web_server.sh"
end
end

# Configuration du serveur d'application
config.vm.define "app_server" do |app|
  app.vm.box = "ubuntu/bionic64"
  app.vm.network "private_network", ip: "192.168.33.10"
  app.vm.synced_folder ".", "/vagrant", disabled: true
  app.vm.provider "virtualbox" do |vb|
    vb.memory = "2048"
  end
  app.vm.provision "shell", path: "provision_app_server.sh"
  app.vm.provision "shell", path: "provision_db_server.sh"
  app.vm.synced_folder "./projetAlumni22", "/home/vagrant/projetAlumni22"
end

# Configuration du serveur de base de données
config.vm.define "db_server" do |db|
  db.vm.box = "ubuntu/bionic64"
  db.vm.network "private_network", ip: "192.168.33.11"
  db.vm.synced_folder ".", "/vagrant", disabled: true
  db.vm.provider "virtualbox" do |vb|
    vb.memory = "4096"
  end
  db.vm.provision "shell", path: "provision_db_server.sh"
end

```

Après la constitution du fichier unique `vagrantfile`, nous allons l'exécuter à travers la commande suivante :

vagrant up

```
Windows PowerShell
PS C:\Users\BAZIE Alex Damien\Desktop\PROJET_VAGRANT> vagrant up
Bringing machine 'app_server' up with 'virtualbox' provider...
Bringing machine 'db_server' up with 'virtualbox' provider...
Bringing machine 'web_server' up with 'virtualbox' provider...
==> app_server: Importing base box 'ubuntu/bionic64'...
==> app_server: Matching MAC address for NAT networking...
==> app_server: Checking if box 'ubuntu/bionic64' version '20230607.0.1' is up to date...
==> app_server: Setting the name of the VM: PROJET_VAGRANT_app_server_1721406004700_25477
==> app_server: Clearing any previously set network interfaces...
==> app_server: Preparing network interfaces based on configuration...
app_server: Adapter 1: nat
app_server: Adapter 2: hostonly
==> app_server: Forwarding ports...
app_server: 22 (guest) => 2222 (host) (adapter 1)
==> app_server: Running 'pre-boot' VM customizations...
==> app_server: Booting VM...
==> app_server: Waiting for machine to boot. This may take a few minutes...
app_server: SSH address: 127.0.0.1:2222
app_server: SSH username: vagrant
app_server: SSH auth method: private key
app_server:
app_server: Vagrant insecure key detected. Vagrant will automatically replace
app_server: this with a newly generated keypair for better security.
app_server:
app_server: Inserting generated public key within guest...
app_server: Removing insecure key from the guest if it's present...
app_server: Key inserted! Disconnecting and reconnecting using new SSH key...
==> app_server: Machine booted and ready!
Got different reports about installed GuestAdditions version:
Virtualbox on your host claims: 5.2.8
```

Nous remarquons dans la capture ci-dessous l'exécution sans erreur de notre fichier vagrantfile :

```
Windows PowerShell
app_server: GRANT
app_server: host    all          all          192.168.33.10/32    md5
==> db_server: Importing base box 'ubuntu/bionic64'...
==> db_server: Matching MAC address for NAT networking...
==> db_server: Checking if box 'ubuntu/bionic64' version '20230607.0.1' is up to date...
==> db_server: Setting the name of the VM: PROJET_VAGRANT_db_server_1721406327758_36091
==> db_server: Fixed port collision for 22 => 2222. Now on port 2200.
==> db_server: Clearing any previously set network interfaces...
==> db_server: Preparing network interfaces based on configuration...
db_server: Adapter 1: nat
db_server: Adapter 2: hostonly
==> db_server: Forwarding ports...
db_server: 22 (guest) => 2200 (host) (adapter 1)
==> db_server: Running 'pre-boot' VM customizations...
==> db_server: Booting VM...
==> db_server: Waiting for machine to boot. This may take a few minutes...
db_server: SSH address: 127.0.0.1:2200
db_server: SSH username: vagrant
db_server: SSH auth method: private key
db_server: Warning: Connection reset. Retrying...
db_server:
db_server: Vagrant insecure key detected. Vagrant will automatically replace
db_server: this with a newly generated keypair for better security.
db_server:
db_server: Inserting generated public key within guest...
db_server: Removing insecure key from the guest if it's present...
db_server: Key inserted! Disconnecting and reconnecting using new SSH key...
==> db_server: Machine booted and ready!
Got different reports about installed GuestAdditions version:
Virtualbox on your host claims: 5.2.8
```

4. Connexions aux machines virtuelles et vérifications des instances

Nous allons procéder à la connexion au machine virtuelle et vérifier les instances installées :

- Serveur web

Pour se connecter à cette machine, nous allons exécuter la commande :

vagrant ssh web_server

```
vagrant@ubuntu-bionic: ~  
PS C:\Users\BAZIE Alex Damien\Desktop\PROJET_VAGRANT> vagrant ssh web_server  
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.15.0-212-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
System information as of Mon Jul 22 18:58:11 UTC 2024  
  
System load:  0.0      Processes:            98  
Usage of /:   4.0% of 38.70GB   Users logged in:     0  
Memory usage: 16%      IP address for enp0s3: 10.0.2.15  
Swap usage:   0%          IP address for enp0s8: 192.168.33.10  
  
Expanded Security Maintenance for Infrastructure is not enabled.  
  
14 updates can be applied immediately.  
11 of these updates are standard security updates.  
To see these additional updates run: apt list --upgradable  
  
119 additional security updates can be applied with ESM Infra.  
Learn more about enabling ESM Infra service for Ubuntu 18.04 at  
https://ubuntu.com/18-04  
  
New release '20.04.6 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
Last login: Mon Jul 22 12:47:14 2024 from 10.0.2.2  
vagrant@ubuntu-bionic:~$
```

Nous allons vérifier les services du serveur nginx sur notre serveur web a travers la commande suivante :

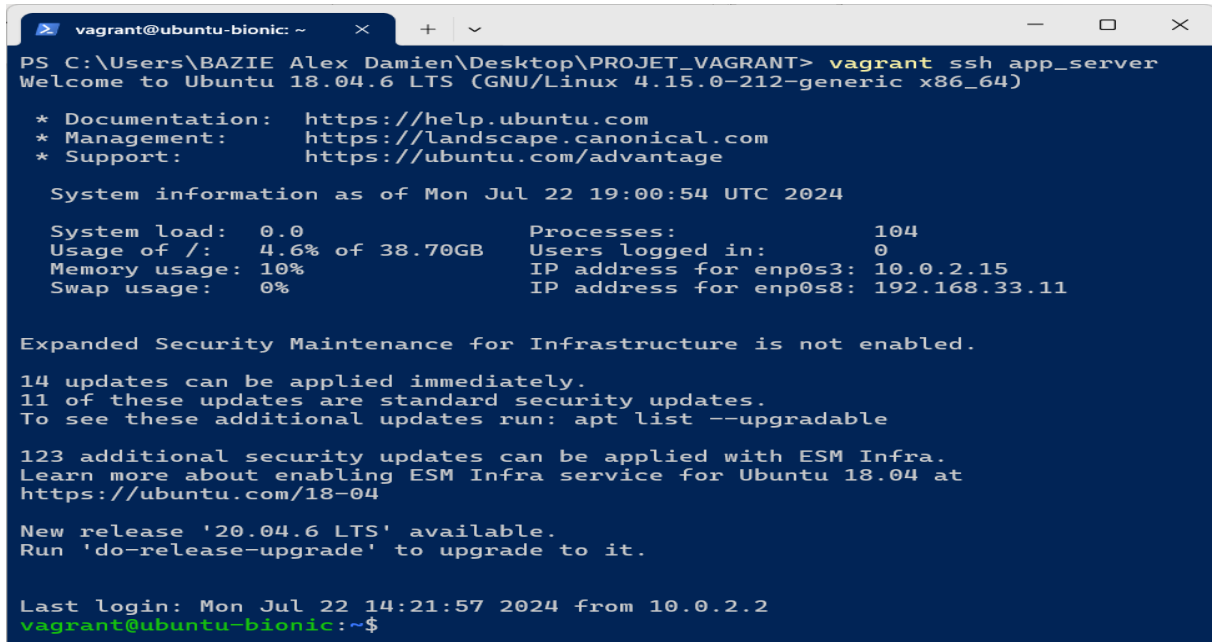
sudo systemctl status nginx

```
vagrant@ubuntu-bionic: ~  
14 updates can be applied immediately.  
11 of these updates are standard security updates.  
To see these additional updates run: apt list --upgradable  
  
119 additional security updates can be applied with ESM Infra.  
Learn more about enabling ESM Infra service for Ubuntu 18.04 at  
https://ubuntu.com/18-04  
  
New release '20.04.6 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
Last login: Mon Jul 22 12:47:14 2024 from 10.0.2.2  
vagrant@ubuntu-bionic:~$ sudo systemctl status nginx  
● nginx.service - A high performance web server and a reverse proxy server  
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: active)   
   Active: active (running) since Mon 2024-07-22 11:09:33 UTC; 7h ago  
     Docs: man:nginx(8)  
  Process: 18408 ExecStop=/sbin/start-stop-daemon --quiet --stop --retry QUIT  
  Process: 18419 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (C  
  Process: 18409 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_proc  
 Main PID: 18420 (nginx)  
    Tasks: 3 (limit: 1151)  
   CGroup: /system.slice/nginx.service  
           └─18420 nginx: master process /usr/sbin/nginx -g daemon on; master  
             └─18421 nginx: worker process  
             └─18422 nginx: worker process  
  
Jul 22 11:09:33 ubuntu-bionic systemd[1]: Starting A high performance web ser  
Jul 22 11:09:33 ubuntu-bionic systemd[1]: Started A high performance web serv  
lines 1-16/16 (END)
```

- Serveur d'application

Pour se connecter à cette machine, nous allons exécuter la commande :

vagrant ssh app_server



```
vagrant@ubuntu-bionic: ~
PS C:\Users\BAZIE Alex Damien\Desktop\PROJET_VAGRANT> vagrant ssh app_server
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.15.0-212-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Mon Jul 22 19:00:54 UTC 2024

System load:  0.0               Processes:           104
Usage of /:   4.6% of 38.70GB   Users logged in:    0
Memory usage: 10%              IP address for enp0s3: 10.0.2.15
Swap usage:   0%               IP address for enp0s8: 192.168.33.11

Expanded Security Maintenance for Infrastructure is not enabled.

14 updates can be applied immediately.
11 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

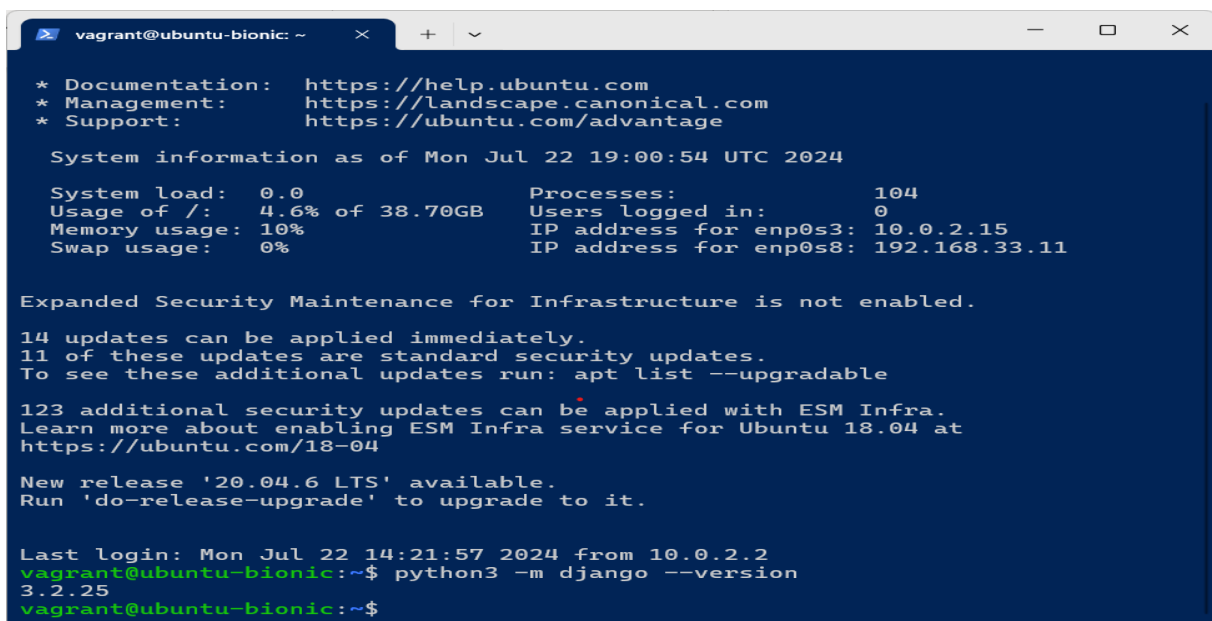
123 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 18.04 at
https://ubuntu.com/18-04

New release '20.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Jul 22 14:21:57 2024 from 10.0.2.2
vagrant@ubuntu-bionic:~$
```

Nous allons vérifier si le framework django est installer sur notre serveur d'application à travers la commande suivante :

python3 -m django --version



```
vagrant@ubuntu-bionic: ~
 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Mon Jul 22 19:00:54 UTC 2024

System load:  0.0               Processes:           104
Usage of /:   4.6% of 38.70GB   Users logged in:    0
Memory usage: 10%              IP address for enp0s3: 10.0.2.15
Swap usage:   0%               IP address for enp0s8: 192.168.33.11

Expanded Security Maintenance for Infrastructure is not enabled.

14 updates can be applied immediately.
11 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

123 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 18.04 at
https://ubuntu.com/18-04

New release '20.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Jul 22 14:21:57 2024 from 10.0.2.2
vagrant@ubuntu-bionic:~$ python3 -m django --version
3.2.25
vagrant@ubuntu-bionic:~$
```

Phase 3 : Configuration des machines virtuels

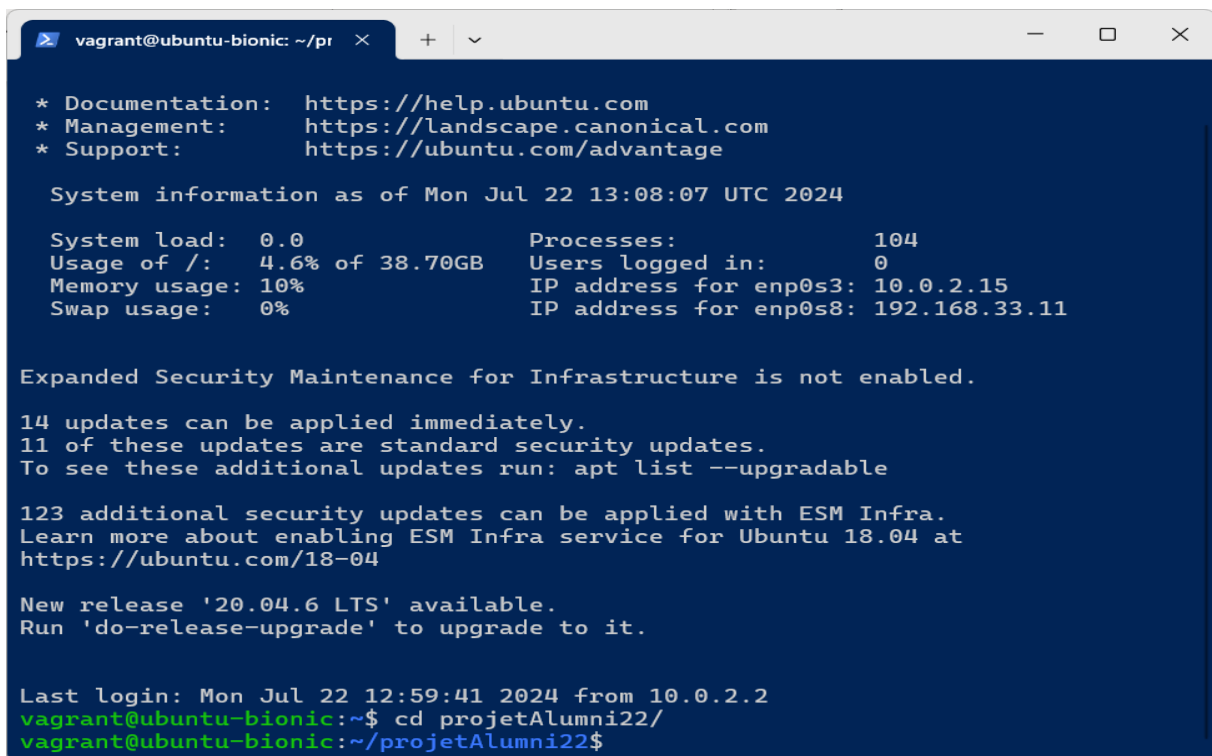
1. Développement de l'application web

Notre application est une application CRUD (Create, Read, Update, Delete) de système de gestion des employés permettant aux employés de mettre à jour leur informations sur les curriculums vitae. Elle est développée en suivant une architecture n-tiers, composée des clients, d'un serveur web, d'un serveur d'application et d'un serveur de base de données. Nous avons utilisé les dossiers partagés pour tester l'application en temps réel sur les machines virtuelles. Le déploiement de l'application s'effectue sur les machines virtuelles à l'aide de scripts de provisionnement, et nous effectuons des tests pour nous assurer que tous les composants fonctionnent correctement et que l'application est accessible via le serveur web.

2. Déploiement de l'application

D'abord, nous allons nous connecter sur le serveur d'application et nous mettre sur notre dossier partagé avec la commande :

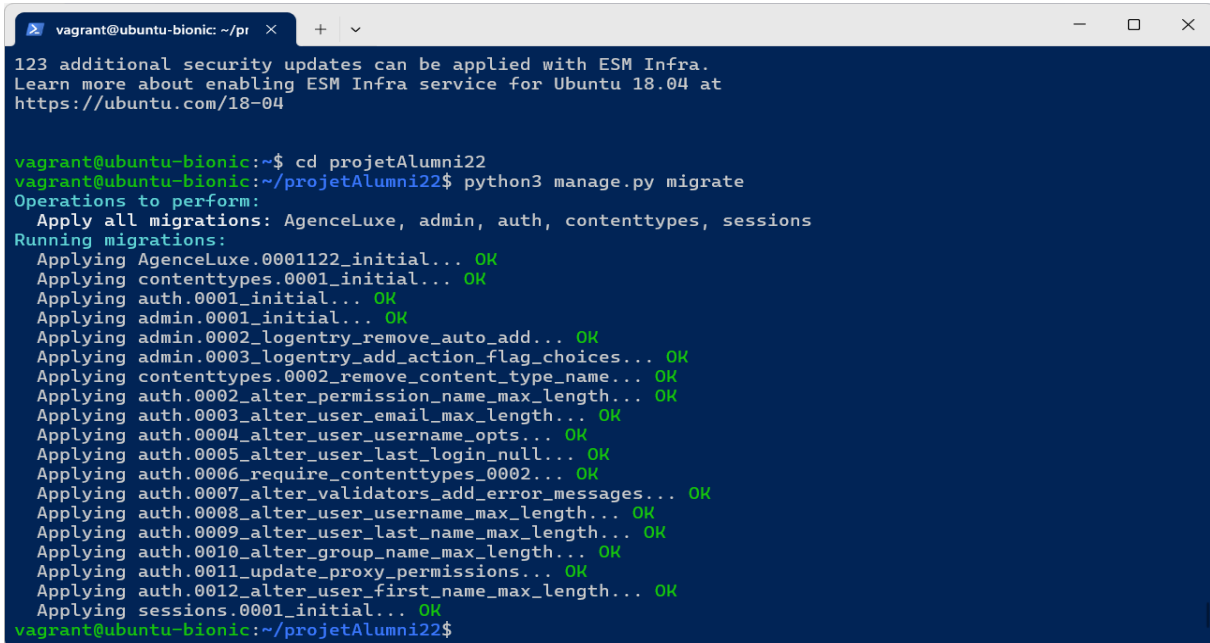
```
vagrant ssh app_server  
cd projetAlumni22
```



```
vagrant@ubuntu-bionic: ~/pr  ×  +  ▾  -  □  ×  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
System information as of Mon Jul 22 13:08:07 UTC 2024  
  
System load:  0.0          Processes:            104  
Usage of /:   4.6% of 38.70GB  Users logged in:      0  
Memory usage: 10%          IP address for enp0s3: 10.0.2.15  
Swap usage:   0%           IP address for enp0s8: 192.168.33.11  
  
Expanded Security Maintenance for Infrastructure is not enabled.  
  
14 updates can be applied immediately.  
11 of these updates are standard security updates.  
To see these additional updates run: apt list --upgradable  
  
123 additional security updates can be applied with ESM Infra.  
Learn more about enabling ESM Infra service for Ubuntu 18.04 at  
https://ubuntu.com/18-04  
  
New release '20.04.6 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
Last login: Mon Jul 22 12:59:41 2024 from 10.0.2.2  
vagrant@ubuntu-bionic:~$ cd projetAlumni22/  
vagrant@ubuntu-bionic:~/projetAlumni22$
```

Ensuite nous allons procéder a la migration de la base de données sur le serveur de base données avec la commande :

python3 manage.py migrate

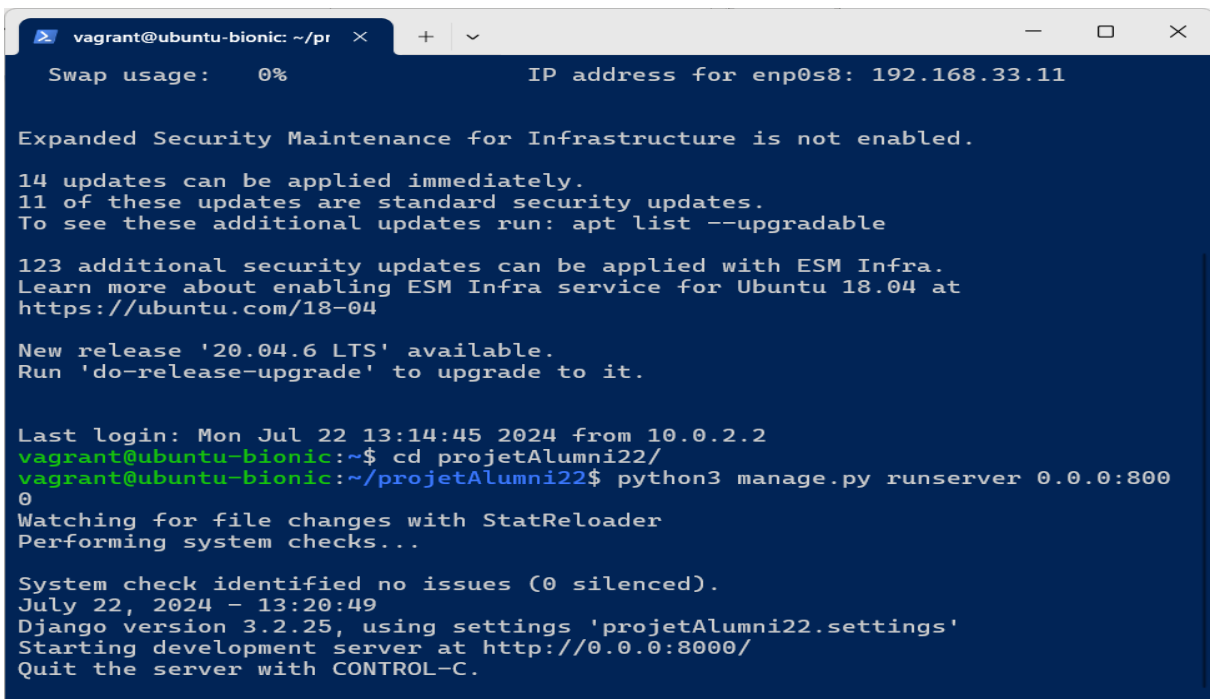
A terminal window titled 'vagrant@ubuntu-bionic: ~/pr' showing the execution of 'python3 manage.py migrate'. The output lists 123 additional security updates, the operations to perform (all migrations), and a detailed list of migrations being applied, all of which are marked as 'OK'. The terminal ends with the prompt 'vagrant@ubuntu-bionic:~/projetAlumni22\$'.

```
vagrant@ubuntu-bionic: ~/pr
123 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 18.04 at
https://ubuntu.com/18-04

vagrant@ubuntu-bionic:~$ cd projetAlumni22
vagrant@ubuntu-bionic:~/projetAlumni22$ python3 manage.py migrate
Operations to perform:
  Apply all migrations: AgenceLuxe, admin, auth, contenttypes, sessions
Running migrations:
  Applying AgenceLuxe.0001122_initial... OK
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
vagrant@ubuntu-bionic:~/projetAlumni22$
```

Enfin, nous allons lancer notre application avec la commande suivante :

python3 manage.py runserver 0.0.0:8000

A terminal window titled 'vagrant@ubuntu-bionic: ~/pr' showing the execution of 'python3 manage.py runserver 0.0.0:8000'. The output displays system information, security updates, and the Django development server starting at http://0.0.0:8000/. The terminal ends with the prompt 'vagrant@ubuntu-bionic:~/projetAlumni22\$'.

```
vagrant@ubuntu-bionic: ~/pr
Swap usage: 0% IP address for enp0s8: 192.168.33.11

Expanded Security Maintenance for Infrastructure is not enabled.

14 updates can be applied immediately.
11 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

123 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 18.04 at
https://ubuntu.com/18-04

New release '20.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Jul 22 13:14:45 2024 from 10.0.2.2
vagrant@ubuntu-bionic:~$ cd projetAlumni22/
vagrant@ubuntu-bionic:~/projetAlumni22$ python3 manage.py runserver 0.0.0:8000
0
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
July 22, 2024 - 13:20:49
Django version 3.2.25, using settings 'projetAlumni22.settings'
Starting development server at http://0.0.0:8000/
Quit the server with CONTROL-C.
```


- Serveur de base de données

Pour se connecter à cette machine, nous allons exécuter la commande :

vagrant ssh db_server

```
vagrant@ubuntu-bionic: ~
PS C:\Users\BAZIE Alex Damien\Desktop\PROJET_VAGRANT> vagrant ssh db_server
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.15.0-212-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:   https://landscape.canonical.com
 * Support:      https://ubuntu.com/advantage

System information as of Mon Jul 22 13:23:40 UTC 2024

System load:  0.0               Processes:    103
Usage of /:   4.1% of 38.70GB   Users logged in: 0
Memory usage: 4%               IP address for enp0s3: 10.0.2.15
Swap usage:  0%               IP address for enp0s8: 192.168.33.12

Expanded Security Maintenance for Infrastructure is not enabled.

14 updates can be applied immediately.
11 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

120 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 18.04 at
https://ubuntu.com/18-04

New release '20.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Jul 22 13:04:16 2024 from 10.0.2.2
vagrant@ubuntu-bionic:~$
```

Après connexion, nous allons tester le service de postgresql avec la commande suivante :

sudo systemctl status postgresql

```
vagrant@ubuntu-bionic: ~
System load:  0.0               Processes:    103
Usage of /:   4.1% of 38.70GB   Users logged in: 0
Memory usage: 4%               IP address for enp0s3: 10.0.2.15
Swap usage:  0%               IP address for enp0s8: 192.168.33.12

Expanded Security Maintenance for Infrastructure is not enabled.

14 updates can be applied immediately.
11 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

120 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 18.04 at
https://ubuntu.com/18-04

New release '20.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Jul 22 13:04:16 2024 from 10.0.2.2
vagrant@ubuntu-bionic:~$ sudo systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor pr
   Active: active (exited) since Mon 2024-07-22 11:16:34 UTC; 2h 10min ago
   Process: 20832 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 20832 (code=exited, status=0/SUCCESS)

Jul 22 11:16:34 ubuntu-bionic systemd[1]: Starting PostgreSQL RDBMS...
Jul 22 11:16:34 ubuntu-bionic systemd[1]: Started PostgreSQL RDBMS.
lines 1-8/8 (END)
```


Nous allons nous connecter a notre base de données a travers la commande suivante :

psql -U vagrant -d myproject

```
vagrant@ubuntu-bionic: ~  
* Management:      https://landscape.canonical.com  
* Support:         https://ubuntu.com/advantage  
  
System information as of Mon Jul 22 13:33:40 UTC 2024  
  
System load:  0.0      Processes:      104  
Usage of /:   4.1% of 38.70GB  Users logged in: 0  
Memory usage: 4%      IP address for enp0s3: 10.0.2.15  
Swap usage:   0%        IP address for enp0s8: 192.168.33.12  
  
Expanded Security Maintenance for Infrastructure is not enabled.  
  
14 updates can be applied immediately.  
11 of these updates are standard security updates.  
To see these additional updates run: apt list --upgradable  
  
120 additional security updates can be applied with ESM Infra.  
Learn more about enabling ESM Infra service for Ubuntu 18.04 at  
https://ubuntu.com/18-04  
  
New release '20.04.6 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
Last login: Mon Jul 22 13:23:41 2024 from 10.0.2.2  
vagrant@ubuntu-bionic:~$ psql -U vagrant -d myproject  
psql (10.23 (Ubuntu 10.23-0ubuntu0.18.04.2))  
Type "help" for help.  
  
myproject=>
```

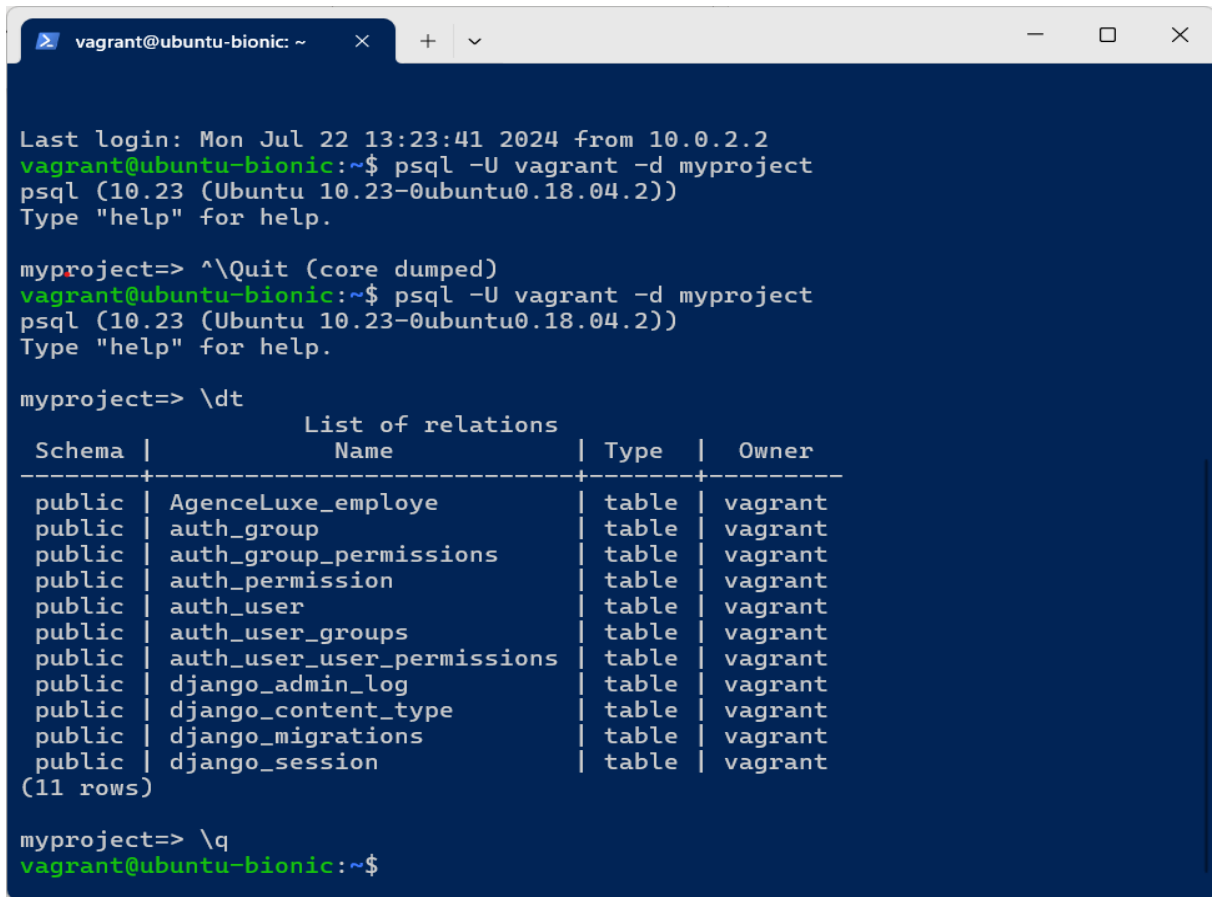
Nous venons de nous connecter à la base de données du projet, nous allons maintenant verifier la liste des tables :

\dt

```
vagrant@ubuntu-bionic: ~  
Run 'do-release-upgrade' to upgrade to it.  
  
Last login: Mon Jul 22 13:23:41 2024 from 10.0.2.2  
vagrant@ubuntu-bionic:~$ psql -U vagrant -d myproject  
psql (10.23 (Ubuntu 10.23-0ubuntu0.18.04.2))  
Type "help" for help.  
  
myproject=> ^\Quit (core dumped)  
vagrant@ubuntu-bionic:~$ psql -U vagrant -d myproject  
psql (10.23 (Ubuntu 10.23-0ubuntu0.18.04.2))  
Type "help" for help.  
  
myproject=> \dt  
  
List of relations  
Schema | Name | Type | Owner  
-----+-----+-----+-----  
public | AgenceLuxe_employe | table | vagrant  
public | auth_group | table | vagrant  
public | auth_group_permissions | table | vagrant  
public | auth_permission | table | vagrant  
public | auth_user | table | vagrant  
public | auth_user_groups | table | vagrant  
public | auth_user_user_permissions | table | vagrant  
public | django_admin_log | table | vagrant  
public | django_content_type | table | vagrant  
public | django_migrations | table | vagrant  
public | django_session | table | vagrant  
(11 rows)  
  
myproject=>
```

Pour quitter la base de données nous allons exécuter la commande suivante :

`\q`



```
vagrant@ubuntu-bionic: ~  
Last login: Mon Jul 22 13:23:41 2024 from 10.0.2.2  
vagrant@ubuntu-bionic:~$ psql -U vagrant -d myproject  
psql (10.23 (Ubuntu 10.23-0ubuntu0.18.04.2))  
Type "help" for help.  
  
myproject=> ^\Quit (core dumped)  
vagrant@ubuntu-bionic:~$ psql -U vagrant -d myproject  
psql (10.23 (Ubuntu 10.23-0ubuntu0.18.04.2))  
Type "help" for help.  
  
myproject=> \dt  
List of relations  
Schema | Name | Type | Owner  
-----+-----+-----+-----  
public | AgenceLuxe_employe | table | vagrant  
public | auth_group | table | vagrant  
public | auth_group_permissions | table | vagrant  
public | auth_permission | table | vagrant  
public | auth_user | table | vagrant  
public | auth_user_groups | table | vagrant  
public | auth_user_user_permissions | table | vagrant  
public | django_admin_log | table | vagrant  
public | django_content_type | table | vagrant  
public | django_migrations | table | vagrant  
public | django_session | table | vagrant  
(11 rows)  
  
myproject=> \q  
vagrant@ubuntu-bionic:~$
```

3. Test de l'application à travers le client

Notre machine sera utilisée comme client pour nous connecter à notre application sur le lien <http://192.168.33.10> sur le serveur web ou <http://192.168.33.11:8000> sur le serveur d'application. Mais avant, nous allons nous connecter à notre serveur d'application et nous positionner sur le dossier partagé pour lancer l'application à travers les commandes suivantes :

```
vagrant ssh app_server  
cd projetAlumni22/  
python3 manage.py runserver 0.0.0:8000
```

```
vagrant@ubuntu-bionic: ~/pr
Memory usage: 10%
Swap usage: 0%
IP address for enp0s3: 10.0.2.15
IP address for enp0s8: 192.168.33.11

Expanded Security Maintenance for Infrastructure is not enabled.

14 updates can be applied immediately.
11 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

123 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 18.04 at
https://ubuntu.com/18-04

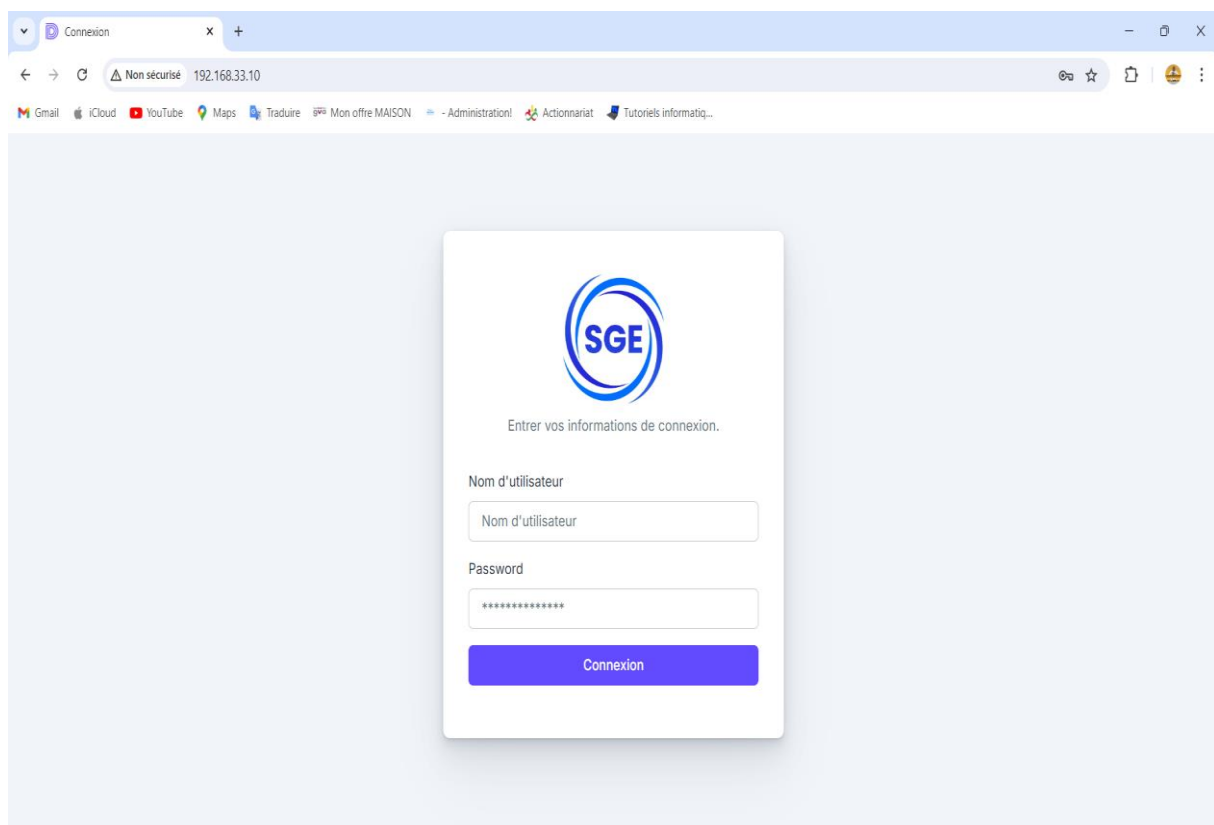
New release '20.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Jul 22 19:00:54 2024 from 10.0.2.2
vagrant@ubuntu-bionic:~$ cd projetAlumni22/
vagrant@ubuntu-bionic:~/projetAlumni22$ python3 manage.py runserver 0.0.0:8000
0
Watching for file changes with StatReloader
Performing system checks...

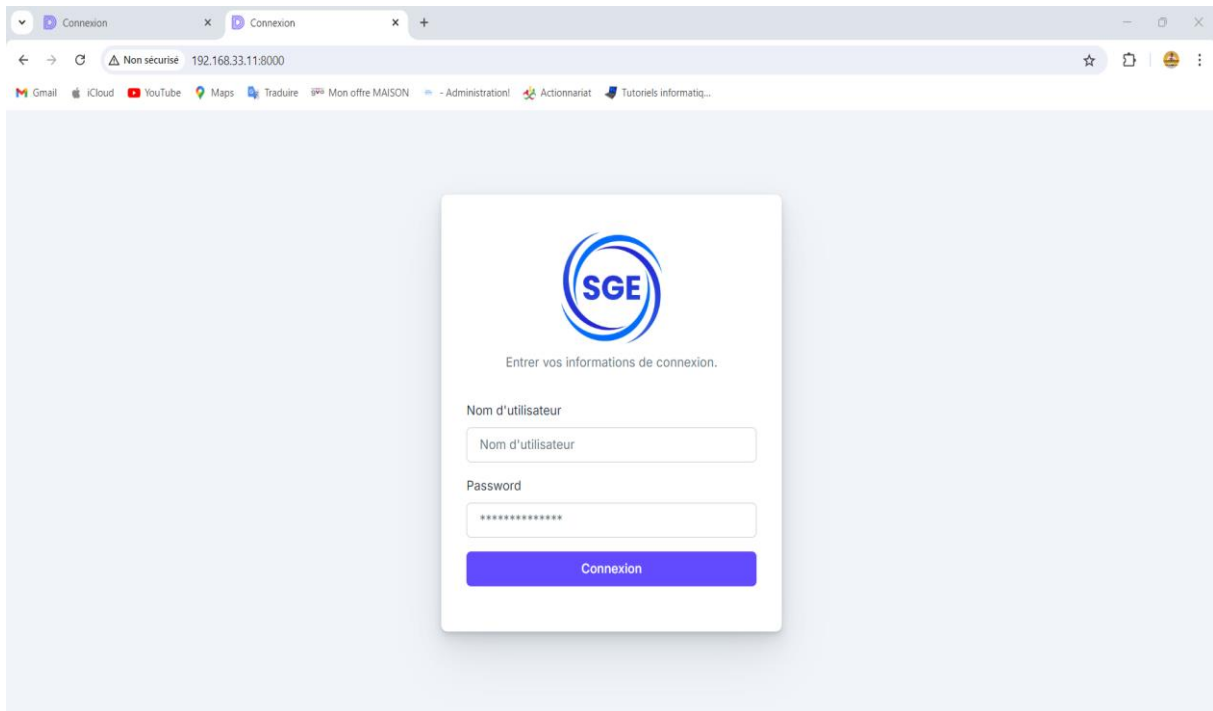
System check identified no issues (0 silenced).
July 22, 2024 - 21:47:31
Django version 3.2.25, using settings 'projetAlumni22.settings'
Starting development server at http://0.0.0:8000/
Quit the server with CONTROL-C.
```

Nous allons aller sur le navigateur pour tester notre application :

- Sur l'adresse du serveur web



- Sur l'adresse du server d'application



Connexion

Non sécurisé 192.168.33.11:8000

Gmail iCloud YouTube Maps Traduire Mon offre MAISON Administration! Actionnariat Tutoriels informatiq...

SGE

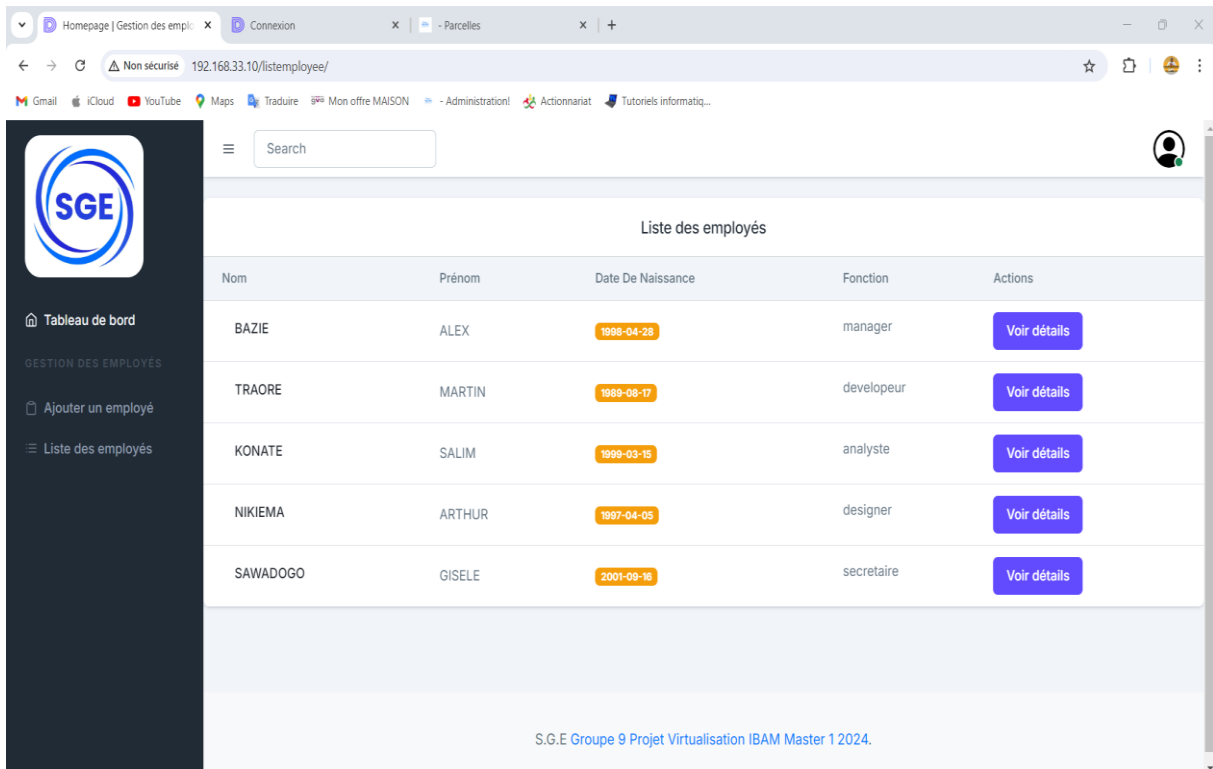
Entrer vos informations de connexion.

Nom d'utilisateur

Password

Connexion

Nous allons procéder à l'enregistrement de des employés sur notre application :



Homepage | Gestion des empl: x Connexion x Parcelles x

Non sécurisé 192.168.33.10/listemployee/

Gmail iCloud YouTube Maps Traduire Mon offre MAISON Administration! Actionnariat Tutoriels informatiq...

SGE

Tableau de bord

GESTION DES EMPLOYÉS

Ajouter un employé

Liste des employés

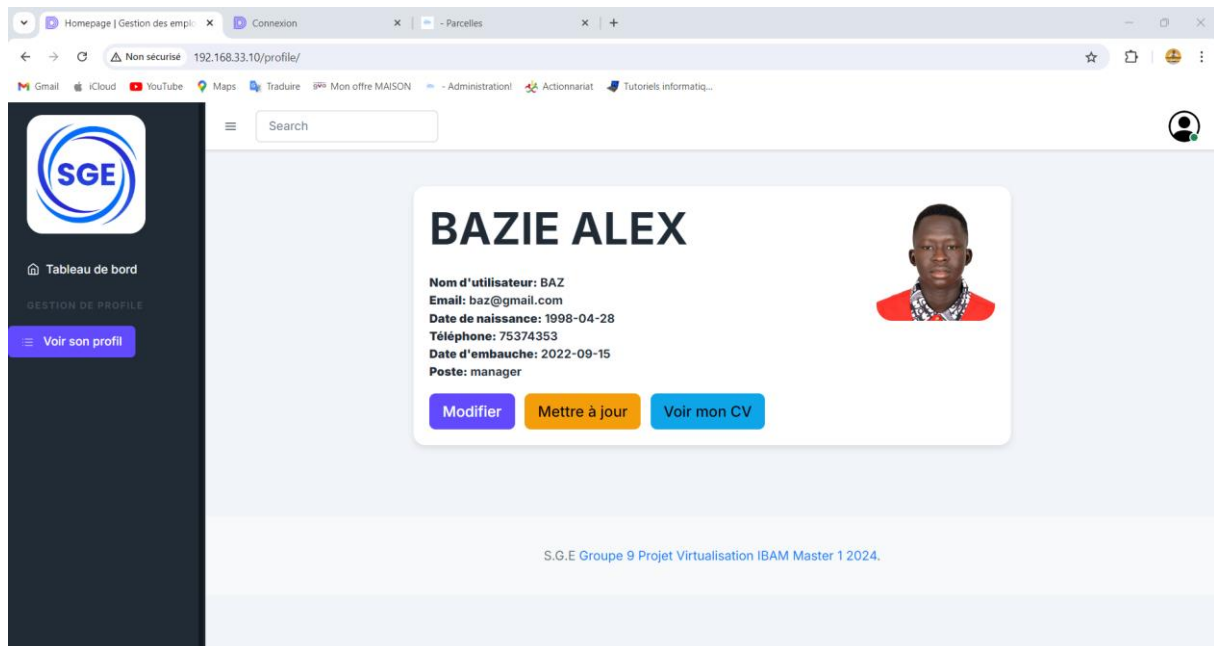
Rechercher

Liste des employés

Nom	Prénom	Date De Naissance	Fonction	Actions
BAZIE	ALEX	1998-04-28	manager	Voir détails
TRAORE	MARTIN	1989-08-17	developeur	Voir détails
KONATE	SALIM	1999-03-15	analyste	Voir détails
NIKIEMA	ARTHUR	1997-04-05	designer	Voir détails
SAWADOGO	GISELE	2001-09-18	secretaire	Voir détails

S.G.E Groupe 9 Projet Virtualisation IBAM Master 1 2024.

Nous allons, nous connecter avec un employé pour voir l'interface :



Phase 4 : Présentation des problèmes rencontrés et les solutions apportées

1. Problème de connexion

- **Description :** Il y a eu des difficultés d'accès à internet des machines virtuelles créées via Vagrant up, empêchant le téléchargement des packages nécessaires.
- **Solution :** Pour résoudre ce problème, nous avons configuré les machines en mode "pont" (bridge mode). Cela permet aux machines virtuelles de se connecter directement au réseau physique pour le téléchargement des packages nécessaires.

2. Problème de temps d'exécution du Vagrantfile trop long

- **Description :** Le temps d'exécution du script Vagrantfile était trop long, provoquant des interruptions dans le déploiement des machines virtuelles et des configurations associées.
- **Solution :** Pour résoudre ce problème, nous avons ajouté la ligne suivante dans le Vagrantfile :

```
config.vm.boot_timeout = 600
```

Cette ligne augmente le délai d'exécution autorisé à 600 secondes (10 minutes), assurant ainsi que Vagrant dispose de suffisamment de temps pour terminer toutes les tâches de configuration sans rencontrer de dépassements de temps.

3. Problème d'accès aux machines virtuelles après l'exécution du Vagrantfile

- **Description :** Après avoir correctement exécuté le Vagrantfile, il était impossible d'accéder aux machines virtuelles et à l'application depuis la machine physique considérée comme le client.

- **Solution :** Les machines virtuelles ont été mises sur un réseau privé hôte (host-only network). Cette configuration permet à la machine physique (le client) d'accéder directement aux machines virtuelles et à l'application hébergée sur celles-ci.

4. Problème : Surcharge et désorganisation du Vagrantfile

- **Description :** Le Vagrantfile devenait trop chargé et difficile à gérer en raison de la quantité croissante de configurations et de scripts de provisionnement.

- **Solution :** Pour éviter la surcharge et la confusion, nous avons décidé de déplacer les scripts de provisionnement dans des fichiers séparés pour chaque machine virtuelle. Chaque machine est ainsi liée à son propre fichier de provisionnement spécifique. Cette séparation permet une gestion plus claire et organisée des configurations.

Conclusion

Ce projet de virtualisation et cloud computing a été une expérience enrichissante et formatrice. L'utilisation de Vagrant nous a permis de comprendre en profondeur les mécanismes de création et de gestion des machines virtuelles. Grâce à cet outil, nous avons pu configurer un environnement complexe comprenant un serveur web, un serveur d'application et une base de données, le tout de manière automatisée et reproductible.

La virtualisation joue un rôle crucial dans le domaine des technologies de l'information. Elle offre de nombreux avantages tels que la réduction des coûts matériels, l'amélioration de l'efficacité des ressources, la flexibilité, et la facilité de gestion des environnements de développement et de production. En isolant les différentes applications et services, la

virtualisation permet également une meilleure sécurité et une gestion simplifiée des mises à jour et des sauvegardes.

Le travail pratique (TP) réalisé nous a permis de mettre en œuvre concrètement les concepts théoriques étudiés. Nous avons pu expérimenter les défis réels liés à la configuration des machines virtuelles, à l'intégration des services et à la résolution des problèmes techniques rencontrés. Cette approche pratique nous a offert une compréhension approfondie et une compétence accrue dans le domaine de la virtualisation et du cloud computing.

En conclusion, ce projet a été une opportunité précieuse pour développer nos compétences techniques et nous préparer aux exigences du monde professionnel. La maîtrise des outils de virtualisation et l'expérience acquise seront sans aucun doute des atouts importants pour notre future carrière dans le domaine des technologies de l'information.