* VM Jenkins + VM Ansible + création d’users sur avec certains droits + (2 ?) VMs hosts pour l’appli (Highavailability : 2 availability zones)+ Database backend
* Figure out how to provide database credentials to the application, through automation
* Les deux VMs hosts doivent pointer sur la même database
* Configure un webhook sur Github pour Jenkins
* Récupère l’app à partir de github avec Jenkins
* Dépose le dossier de l’app sur la VM Ansible + lance des playbooks Ansible
  + Création d’un Dockerfile à partir d’une image Apache dans lequel on va copier le dossier de l’app + installer les dépendances/packages nécessaires
  + Playbooks :
    - Build une image à partir du Dockerfile + push sur un repo DockerHub
    - Création des containers sur les VM hosts (GCP ? Azure ? AWS ?) à partir de l’image push sur le DockerHub
  + Vault : Credentials pour la database (When you perform the deployment, fetch the credentials from Vault)
* Load balancer devant les 2 VMs Hosts
* Backup de la database + cron job pour automatiser
  + Si on utilise pas Kubernetes : Use an orchestration tool like Docker Compose to ensure that the containers are automatically restarted if they are stopped.
* Configure the application to expose some metrics, to show its health. Explore which metrics the application can expose; e.g. if it’s a web server, can you expose metrics like requests-per-second, or memory usage?
* Extract or scrape the metrics into a monitoring tool
* Create a dashboard to display the metrics in real time.
* Add monitoring to your underlying infrastructure, too. So if you’re deploying onto virtual machines, then find a way to also monitor disk space, system load, etc.
* Use a load testing tool to hit your application with extra load. How does it affect the metrics in the dashboard?
* Création template déploiement de l’infrastructe avec Terraform

Réfléchir et tester: Comment est géré l’écriture dans la base de donnée quand plusieurs utilisateurs sont connectés et écrivent des notes ? Déploiement de l’infrastructure avec Terraform à la fin ? Améliroration de l’app (ajout features) ? Remplacer Jenkins par Github Actions ?

Déploiement app sur VM Apache :

* Apt update
* Apt install apache2 git postgresql python pip
* Importer le dossier de l’app dans /var/www/
* apt-get install libpq-dev (sinon problème à l’étape suivante pour l’installation de psycopg2)
* cd Dossier-App + ajout psycopg2-binary avant psycopg2 dans requirments.txt+ pip install -r requirements.txt
* Créer le fichier nom-app.wsgi
* Créer le fichier de conf pour apache (*trouver un moyen de mettre l’adresse ip automatiquement dans le fichier puisqu’elle peut changer si on éteint et rallume la VM*)
* Mettre le fichier de conf dans /etc/apache2/sites-available/
* apt-get install apache2-dev
* <https://stackoverflow.com/questions/33320889/invalid-command-wsgidaemonprocess-deploy-django-application-on-centos-6-7>
  + pip install mod\_wsgi
  + mod\_wsgi-express module-config + copier le output
  + vim /etc/apache2/mods-available/wsgi.load + coller le output précedent
  + sudo a2enmod wsgi
* Enable le nouveau site => sudo a2ensite nom-du-fichier-de-conf
* Systemctl reload apache2
* Apache est actif mais Internal Server Error quand on va sur ip :80
* cat logs/error.logs
* A corriger dans le website.wsgi : from main et pas from website
* Recharger la page + cat logs/errors.logs => partie base de donnée postgresql
* Postgresql :
  + Sudo -i -u postgres (se connecter à la DB la premiere fois avec l’user postgres)
  + Psql (lance l’invite de command sql de Postgres)
  + \l => liste les DB
  + \du => liste les rôles des users et groupes
  + Créer un user : CREATE ROLE <nom\_user> LOGIN ;
  + Donner des droits : ALTER ROLE <nom\_user> CREATEDB ;
  + Créer une DB : CREATE DATABASE <nom\_DB> OWNER <nom\_user>
  + Mdp user : ALTER ROLE <nom\_user > WITH ENCRYPTED PASSWORD ‘mdp’ ;
  + \q => exit
* Changer dans le code  mdp DB + nom DB + username dans les fichiers :
  + website/\_\_init\_\_.py
  + test\_postgre.py
  + config.py

Containerisation de l’app :

* installation docker ubuntu :
  + <https://stackoverflow.com/questions/30379381/docker-command-not-found-even-though-installed-with-apt-get>
* Dockerfile :

Postgresql :192.168.92.21

-Docker compose :

-installer vim dans le conteneur web pour changer les ‘localhost’ dans les fichiers par l’adresse

Ip du conteneur postgres (automatiser : créer une variable d’environnement ?)

Apachectl configtest

/etc/init.d/apache2 reload

cat /var/log/apache2/error.log

ajouter dans le dockerfile a2dissite 000-default.conf

changer url pour accès database

<https://github.com/carlostighe/apache-flask>

Secrets :

* Docker swarm init
* 1. Créer un fichier avec le password + docker create secret <nom\_secret> <fichier>