KUBERNETES PARTIE II - PROJET - DEVOPS - Orchestration - SRITHARAN Vaksalan 5SRC2

Objectif

Ce projet à pour objectif de déployer et orchestrer trois applications conteneurisées dans un cluster Kubernetes Minikube pour l'entreprise fictive **IC GROUP** :

- Un site vitrine Flask personnalisable via des variables d'environnement
- L'ERP Odoo 13 connecté à une base PostgreSQL
- pgAdmin pour l'administration graphique de la base de données

Étapes de déploiement

1. Préparation de l'environnement

Dans un premier temps, j'ai préparé l'environnement en installant les outils nécessaire pour le bon fonctionnement du projet

Outils essentiels et commandes d'installation

Outil	Commande d'installation
curl	sudo apt install curl -y`
Git	`sudo apt install git -y`
Docker	`sudo apt install -y docker.io``sudo systemctl enable docker``sudo systemctl start docker``sudo usermod -aG docker $USER`
kubectl	`curl -LO "https://dl.k8s.io/release/$(curl -Ls https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"``sudo install -o r root -m 0755 kubectl /usr/local/bin/kubectl`

Outil	Commande d’installation
Minikube	`curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube_latest_amd64.deb` `dpkg -i minikube_latest_amd64.deb`

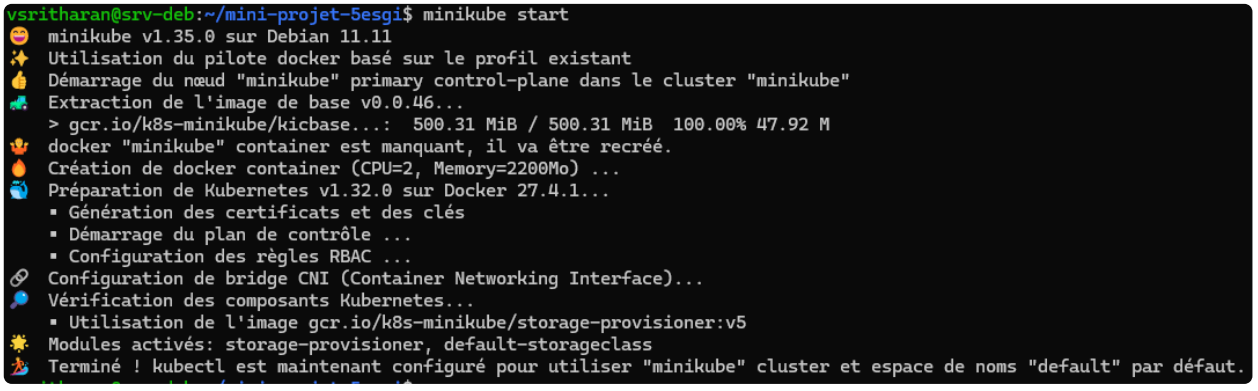
Vérification après installation

Outil	Commande
Minikube	`minikube version`
kubectl	`kubectl version --client`
Docker	`docker --version`
Git	`git --version`

Lancement de Minikube

Puis pour lancer minikube j'ai utilisé la commande :

```
minikube start
```



**2. Clonage du repository GitHub

J’ai récupéré le projet en faisant un fork du repo : https://github.com/OlivierKouokam/mini-projet-5esgi et je me suis placé dans le répertoire Kubernetes :

```
git clone https://github.com/Vaksalan/mini-projet-5esgi.git
```

```
cd mini-projet-5esgi/kubernetes
```

```
vsritharan@srv-deb:~$ git clone https://github.com/Vaksalan/mini-projet-5esgi
Clonage dans 'mini-projet-5esgi'...
remote: Enumerating objects: 46, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 46 (delta 5), reused 5 (delta 5), pack-reused 39 (from 2)
Réception d'objets: 100% (46/46), 906.76 Kio | 9.07 Mio/s, fait.
Résolution des deltas: 100% (11/11), fait.
vsritharan@srv-deb:~$ ls
mini-projet-5esgi
vsritharan@srv-deb:~$ cd mini-projet-5esgi/
vsritharan@srv-deb:~/mini-projet-5esgi$ ls
app.py  Dockerfile  images  mini_projet_5esgi.pdf  Readme.md  releases.txt  static  templates
vsritharan@srv-deb:~/mini-projet-5esgi$
```

**3. Conteneurisation de l'application web

3.1 - Pour la conteneurisation, j'ai en premier lieu créé un Dockerfile :

```
FROM python:3.6-alpine
WORKDIR /opt
ENV ODOO_URL=http://odoo.local PGADMIN_URL=http://pgadmin.local
COPY . .
RUN pip install flask==1.1.2
EXPOSE 8080
ENTRYPOINT ["python", "app.py"]
```

3.2 - Puis, j'ai exécuté les commandes suivante pour build l'image

```
docker build -t ic-webapp:1.0 .
docker run -d -p 8080:8080 \
    -e ODOO_URL=https://www.odoo.com \
    -e PGADMIN_URL=https://www.pgadmin.org \
    --name test-ic-webapp ic-webapp
```

```
vsritharan@srv-deb:~/mini-projet-5esgi$ docker build -t ic-webapp:1.0 .
docker run -d -p 8080:8080 \
 -e ODOO_URL=https://www.odoo.com \
 -e PGADMIN_URL=https://www.pgadmin.org \
 --name test-ic-webapp ic-webapp:1.0
Sending build context to Docker daemon  2.331MB
Step 1/7 : FROM python:3.6-alpine
3.6-alpine: Pulling from library/python
59bf1c3509f3: Pull complete
8786870f2876: Pull complete
acb0e804800e: Pull complete
52bedcb3e853: Pull complete
b064415ed3d7: Pull complete
Digest: sha256:579978dec4602646fe1262f02b96371779bfb0294e92c91392707fa999c0c989
Status: Downloaded newer image for python:3.6-alpine
 ---> 3a9e80fa4606
Step 2/7 : WORKDIR /opt
 ---> Running in a5abdb2cf38f
Removing intermediate container a5abdb2cf38f
 ---> bfe9ad0aa13c
Step 3/7 : ENV ODOO_URL=http://odoo.local PGADMIN_URL=http://pgadmin.local
 ---> Running in 2ce0abc149b6
Removing intermediate container 2ce0abc149b6
 ---> 64eaee7dbef5
Step 4/7 : COPY . .
 ---> 1e07a3efca1d
Step 5/7 : RUN pip install flask==1.1.2
 ---> Running in 0614aed799a1
```

3.3 - Et enfin, j'ai push l'image sur DockerHub à l'aide des commandes suivantes :

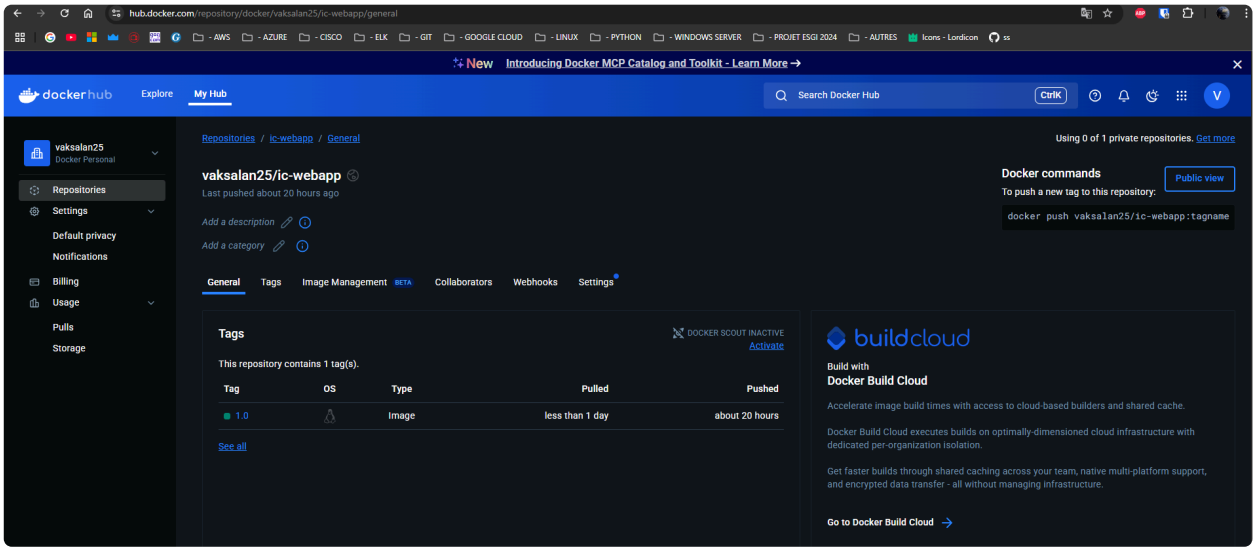
```
docker login
docker tag ic-webapp:1.0 vsritharan/ic-webapp:1.0
docker push vaksalan25/ic-webapp:1.0
```

```
vsritharan@srv-deb:~/mini-projet-5esgi$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://h
ub.docker.com to create one.
Username: vaksalan25
Password:
WARNING! Your password will be stored unencrypted in /home/vsritharan/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

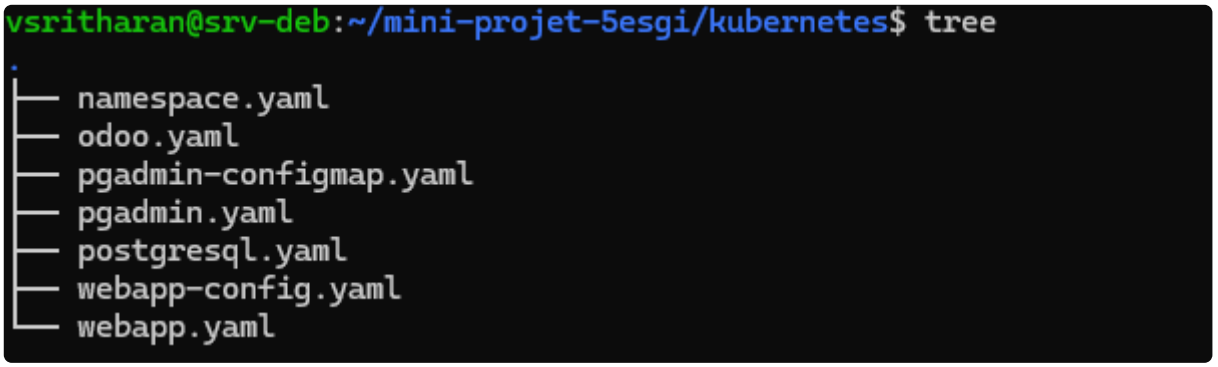
```
vsritharan@srv-deb:~/mini-projet-5esgi$ docker tag ic-webapp:1.0 vaksalan25/ic-webapp:1.0
vsritharan@srv-deb:~/mini-projet-5esgi$ docker push vaksalan25/ic-webapp:1.0
The push refers to repository [docker.io/vaksalan25/ic-webapp]
65c3d0d41938: Pushed
c488f95c8492: Pushed
3156423bd38f: Mounted from library/python
efa76becf38b: Mounted from library/python
671e3248113c: Mounted from library/python
1965cfbef2ab: Mounted from library/python
8d3ac3489996: Mounted from library/python
1.0: digest: sha256:e5c1d5cdae1ec32a6f975b9290824e11dcd8c00855684 1dd84a0671bd315616e size: 1790
```

https://hub.docker.com/repository/docker/vaksalan25/ic-webapp/general



**4. Déploiement des ressources Kubernetes

Voici les fichiers que j'ai utilisés pour construire cette infrastructure



Fichier	Rôle
`namespace.yaml`	Création du namespace `icgroup`
`odoo.yaml`	Déploiement et NodePort Odoo
`pgadmin-configmap.yaml`	Injection de `servers.json` dans pgAdmin
`pgadmin.yaml`	Déploiement de pgAdmin
`postgresql.yaml`	Déploiement et service PostgreSQL
`webapp-config.yaml`	Variables d'environnement de la webapp
`webapp.yaml`	Déploiement de l'application Flask

Vous pouvez retrouvez le contenu des fichiers `.yaml` sur : https://github.com/Vaksalan/mini-projet-5esgi/tree/main/kubernetes

4.1 - Ensuite, j'ai appliqué tous les fichiers YAML avec `kubectl apply -f` :

```
vsritharan@srv-deb:~/mini-projet-5esgi$ kubectl apply -f kubernetes/namespace.yaml
namespace/icgroup created
vsritharan@srv-deb:~/mini-projet-5esgi$ kubectl apply -f kubernetes/odoo.yaml
deployment.apps/odoo created
vsritharan@srv-deb:~/mini-projet-5esgi$ kubectl apply -f kubernetes/pgadmin-configmap.yaml
configmap/pgadmin-config created
vsritharan@srv-deb:~/mini-projet-5esgi$ kubectl apply -f kubernetes/pgadmin
error: the path "kubernetes/pgadmin" does not exist
vsritharan@srv-deb:~/mini-projet-5esgi$ kubectl apply -f kubernetes/pgadmin.yaml
deployment.apps/pgadmin created
service/pgadmin created
vsritharan@srv-deb:~/mini-projet-5esgi$ kubectl apply -f kubernetes/postgresql.yaml
persistentvolumeclaim/postgres-pvc created
service/postgres created
deployment.apps/postgres created
vsritharan@srv-deb:~/mini-projet-5esgi$ kubectl apply -f kubernetes/webapp.yaml
deployment.apps/ic-webapp created
service/ic-webapp created
vsritharan@srv-deb:~/mini-projet-5esgi$ kubectl apply -f kubernetes/webapp-config.yaml
configmap/webapp-config created
vsritharan@srv-deb:~/mini-projet-5esgi$
```

4.2 - Sécurisation des informations

Pour sécuriser les informations sensibles de mon projet Kubernetes, j'ai utilisé un fichier `secret.yaml` qui me permet de stocker de manière masquée les identifiants et mots de passe nécessaires au fonctionnement d'Odoo, PostgreSQL et PGAdmin. Ce fichier contient des données encodées en base64, comme le mot de passe administrateur d'Odoo ou encore les identifiants de connexion à PGAdmin.

secret.yaml :

`apiVersion: v1`
`kind: Secret`
`metadata:`
`name: app-secrets`
`namespace: icgroup`
`type: Opaque`
`data:`
`pgadmin-email: dnNyaXRoYXJhbkBteWdlcy5mcg== # vsritharan@myges.fr`
`pgadmin-password: YWRtaW4xMjM= # admin123`
`postgres-password: b2Rvbw== # mypostgres`
`odoo-password: b2Rvbw== # odoo`

Pour l'appliquer dans Kubernetes j'ai effectué les étapes ci-dessous :

1. Application du secret avec `kubectl`

```
kubectl apply -f secret.yaml
```

2. Vérifier que le secret est bien créé :

```
kubectl get secrets -n icgroup
```

3. Affichage du contenu encodé

```
kubectl get secret app-secrets -n icgroup -o yaml
```

```
vsritharan@srv-deb:~/mini-projet-5esgi/kubernetes$ kubectl get secrets -n icgroup
NAME          TYPE     DATA   AGE
app-secrets   Opaque   4      18h
vsritharan@srv-deb:~/mini-projet-5esgi/kubernetes$ kubectl get secret app-secrets -n icgroup -o yaml
apiVersion: v1
data:
  odoo-password: b2Rvbw==
  pgadmin-email: dnNyaXRoYXJhbkBteWdlcy5mcg==
  pgadmin-password: YWRtaW4xMjM=
  postgres-password: b2Rvbw==
kind: Secret
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","data":{"odoo-password":"b2Rvbw==","pgadmin-email":"dnNyaXRoYXJhbkBteWdlcy5mcg==","pgadmin-pass
word":"YWRtaW4xMjM=","postgres-password":"b2Rvbw=="},"kind":"Secret","metadata":{"annotations":{},"name":"app-secrets","
namespace":"icgroup"},"type":"Opaque"}
  creationTimestamp: "2025-05-18T01:45:12Z"
  name: app-secrets
  namespace: icgroup
  resourceVersion: "7339"
  uid: 0a8d3175-8fee-4eea-935f-101fcb48a404
type: Opaque
```

**5. Vérification du déploiement

Pour m’assurer que tous les pods et services sont en place, j’ai utilisé :

```
kubectl get all -n icgroup
```

```
vsritharan@srv-deb:~/mini-projet-5esgi/kubernetes$ kubectl get all -n icgroup
NAME                                 READY   STATUS    RESTARTS       AGE
pod/ic-webapp-84b4f8785b-56xvf       1/1     Running   1 (166m ago)   20h
pod/odoo-655fd48dff-rw5hq            1/1     Running   1 (18h ago)    18h
pod/pgadmin-774674bd89-8ckkp         1/1     Running   1 (18h ago)    19h
pod/postgres-86496cf75f-4gwbl        1/1     Running   1 (18h ago)    18h

NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/ic-webapp     NodePort    10.106.19.123   <none>        8080:30080/TCP   20h
service/odoo          NodePort    10.105.47.156   <none>        8069:30069/TCP   19h
service/pgadmin       NodePort    10.96.70.71     <none>        80:30050/TCP     20h
service/postgres      ClusterIP   10.111.208.10   <none>        5432/TCP         20h

NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ic-webapp     1/1     1            1           20h
deployment.apps/odoo          1/1     1            1           18h
deployment.apps/pgadmin       1/1     1            1           20h
deployment.apps/postgres      1/1     1            1           18h

NAME                                     DESIRED   CURRENT   READY   AGE
replicaset.apps/ic-webapp-84b4f8785b     1         1         1       20h
replicaset.apps/odoo-655fd48dff          1         1         1       18h
replicaset.apps/pgadmin-774674bd89       1         1         1       19h
replicaset.apps/pgadmin-7796bb8dd6       0         0         0       20h
replicaset.apps/pgadmin-f8f8fb7          0         0         0       19h
replicaset.apps/postgres-86496cf75f      1         1         1       18h
```

Accès aux applications

Après avoir récupéré l'IP du cluster avec `minikube ip`, j'ai pu accéder aux services :

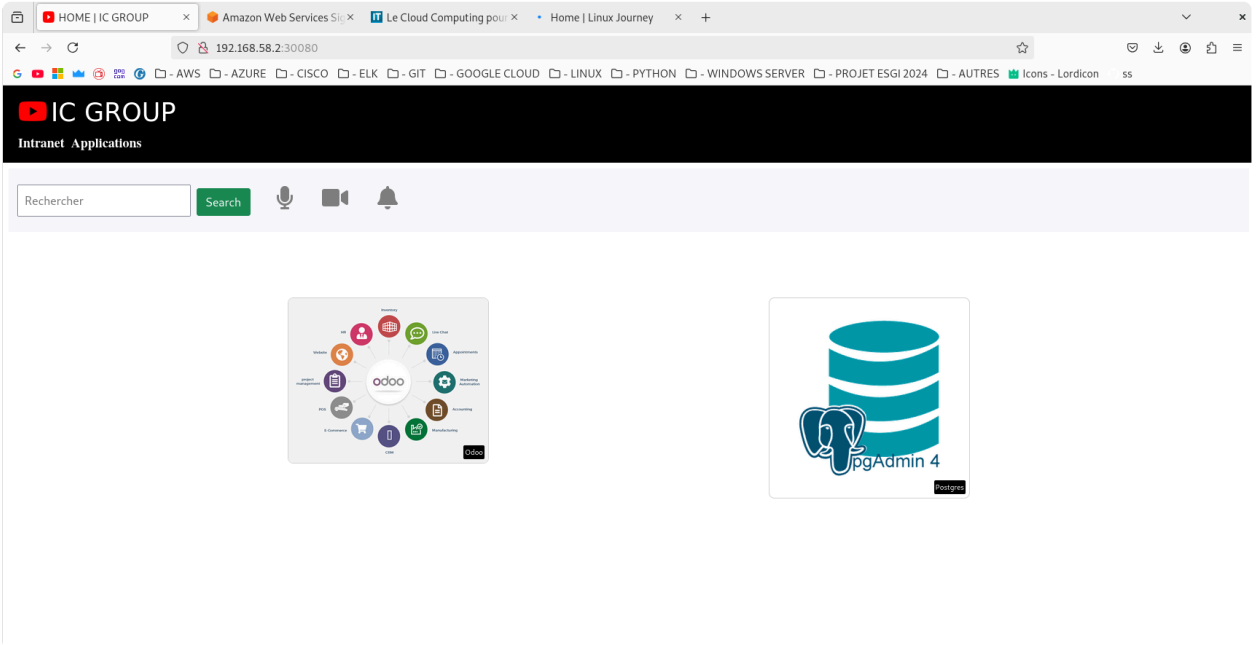
```
vsritharan@srv-deb:~/mini-projet-5esgi/kubernetes$ minikube ip
192.168.58.2
```

Application	URL
Webapp	`http://192.168.58.2:30080`
Odoo	`http://192.168.58.2:30069`
pgAdmin	`http:/192.168.58.2:30050`

**6. Test d'accès aux services

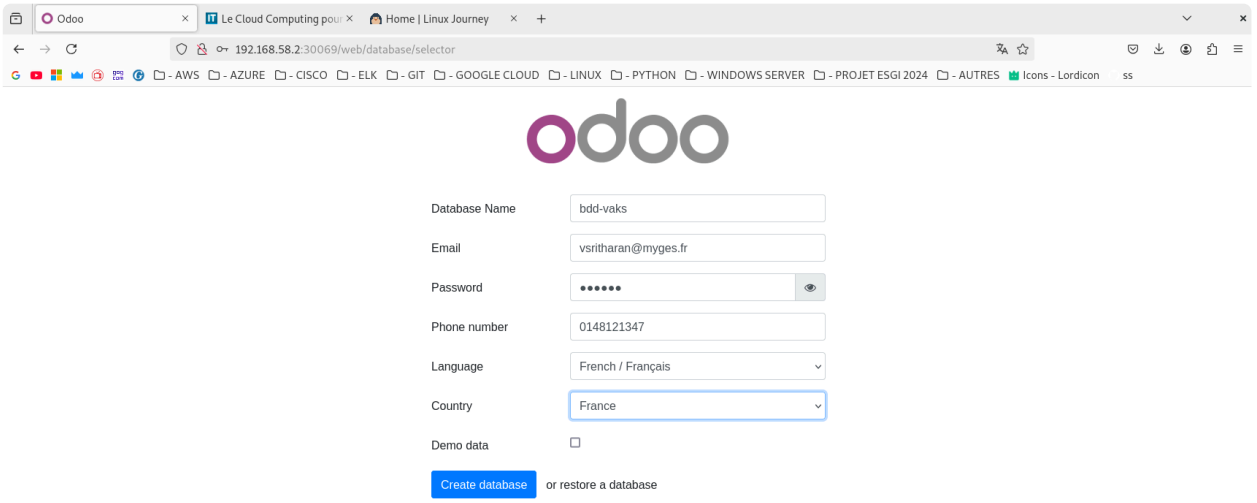
Webapp Flask :

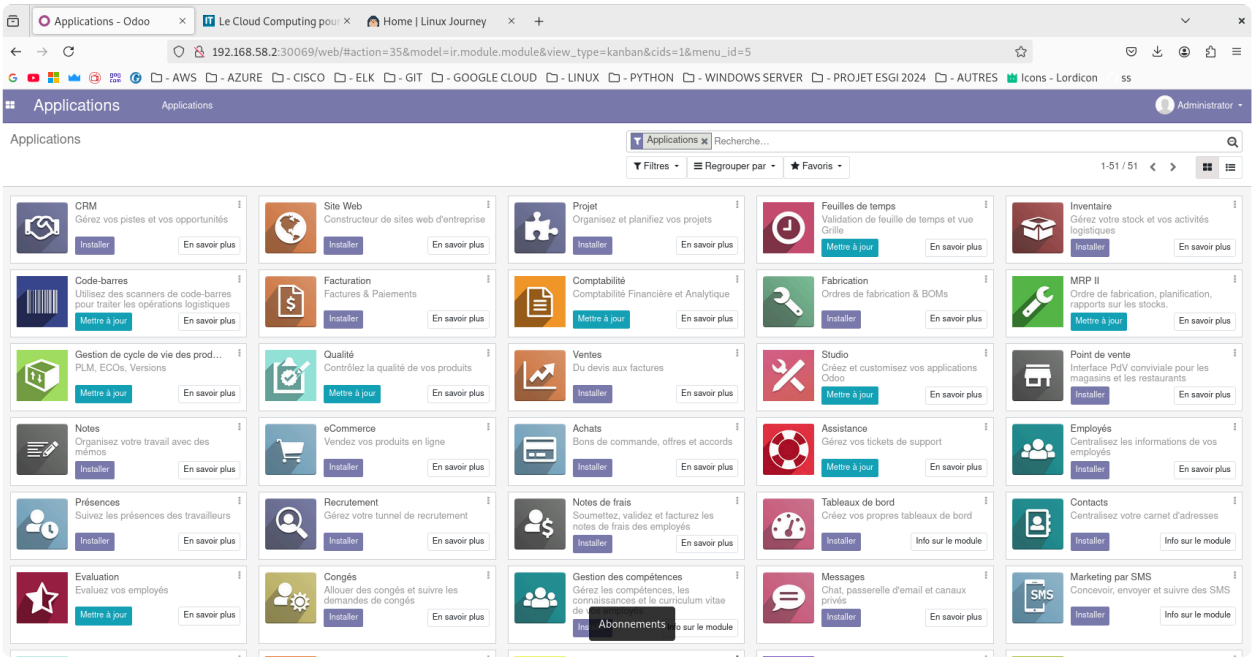
Le site vitrine Flask est fonctionnel et affiche dynamiquement les liens vers Odoo et pgAdmin



Odoo:

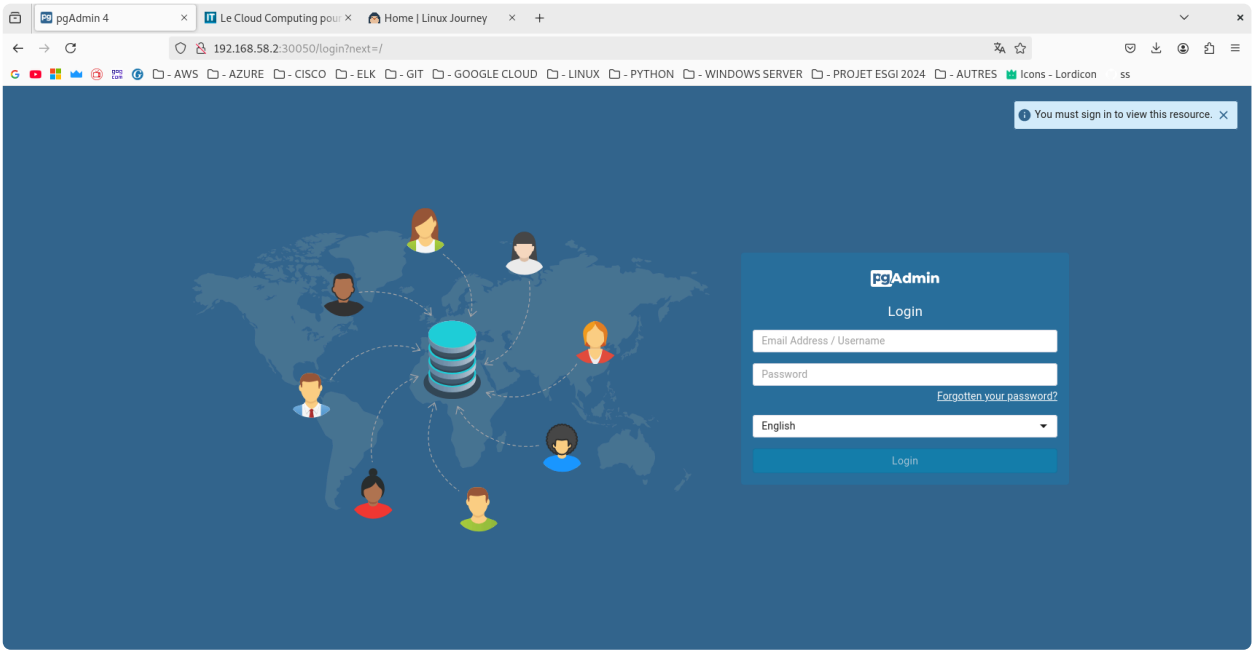
Odoo est connecté à PostgreSQL et permet de créer des bases

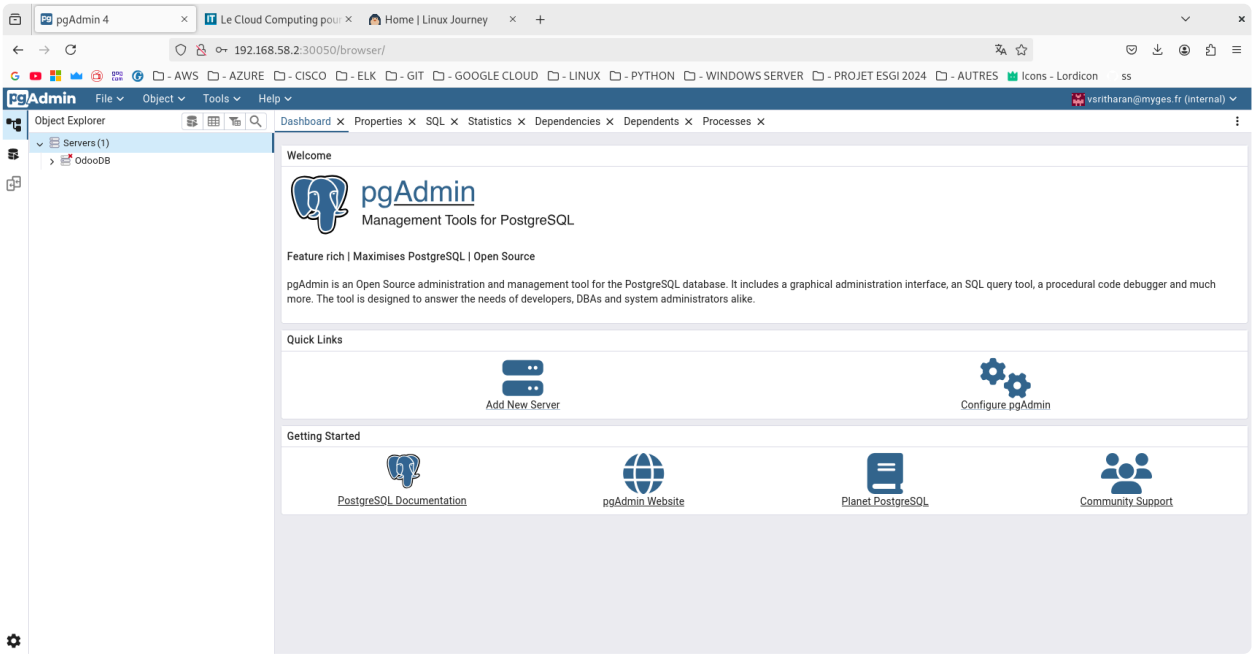




PgAdmin:

pgAdmin est préconfiguré grâce au fichier `servers.json. Pour accéder à l'interface de pgAdmin, j'ai utilisé les identifiants définis dans le fichier` secret.yaml`, qui contient l'email et le mot de passe encodés et injectés dans le déploiement.





7. Conclusion

Ce projet m’a permis de :

- Apprendre à orchestrer plusieurs conteneurs dans un environnement Kubernetes
- Utiliser les objets `Secret`, `ConfigMap`, `NodePort`, `PVC`, etc.
- Comprendre comment connecter plusieurs services entre eux dans un cluster