KUBERNETES PARTIE II - CC - REVERSE ENGINEERING - Voting App

Objectif

Le but de ce projet était de déployer l'application open-source **Example Voting App** sur un cluster Kubernetes local via Minikube. J'ai ensuite vérifié que le système de vote fonctionne correctement et que les résultats apparaissent en temps réel.

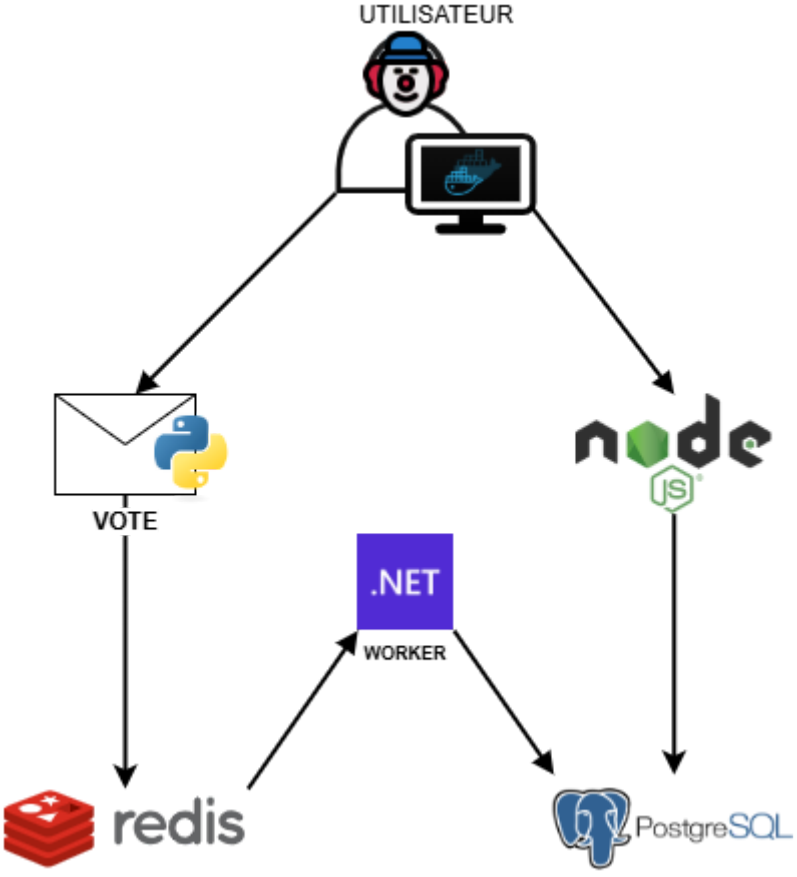
Ce travail m’a permis de :

- Comprendre le fonctionnement d’une application multi-conteneurs
- Tester la communication entre les différents services
- Documenter chaque étape avec des captures d’écran
- Proposer un schéma d’architecture pour mieux visualiser les interactions

---

Architecture de l'application

Voici le schéma que j’ai réalisé pour illustrer les flux entre les composants, pour mieux comprendre le fonctionnement, j'ai analysé les rôles de chaque composant de l'application



Voici ce que chaque service fait :

- Une **application web en Python** (vote) permet de choisir une option
- Le service **Redis** récupère les votes temporairement
- Un **worker .NET** lit Redis et enregistre les votes dans **PostgreSQL**
- Enfin, l'interface **result** (Node.js) affiche en direct les résultats

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`
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`

---

1. **Lancer Minikube**

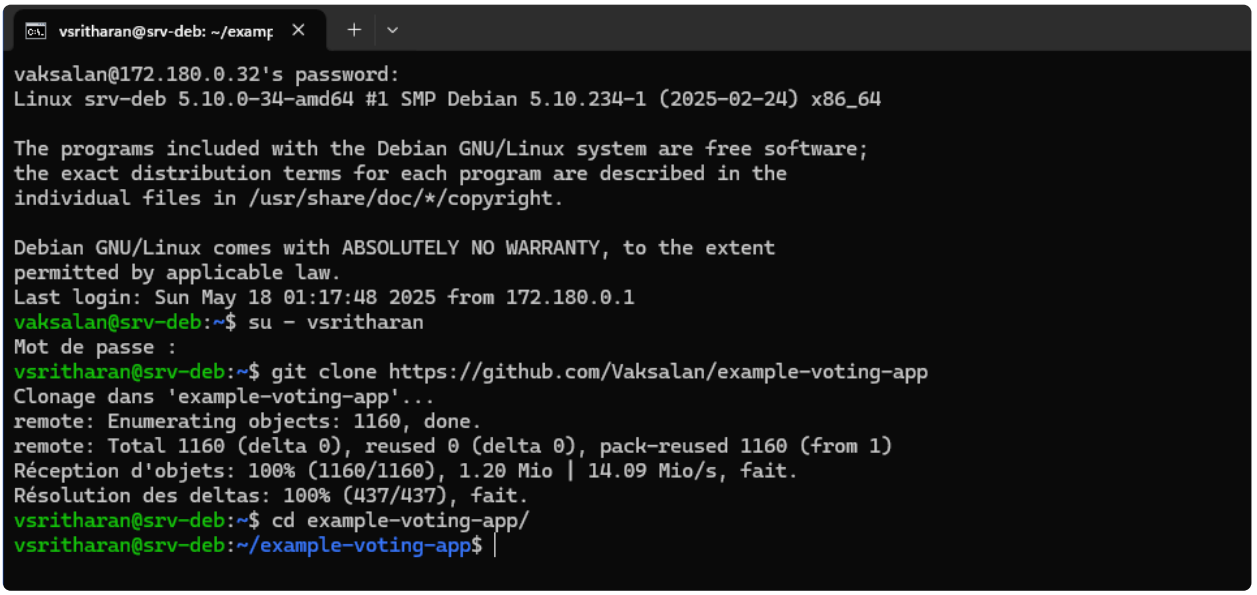
J’ai commencé par démarrer Minikube avec la commande suivante :

```
minikube start
```

2. Cloner le projet Voting App

Ensuite, j’ai forké le dépôt contenant les fichiers de déploiement YAML et je l'ai cloné sur ma machine :

```
git clone https://github.com/Vaksalan/example-voting-app
cd example-voting-app/k8s-specifications
```



---

3. Appliquer les manifests Kubernetes

J’ai déployé l’ensemble des services avec cette commande :

```
kubectl create -f .
```

---

4. Vérifier les objets déployés

Pour m’assurer que tout fonctionne bien, j’ai vérifié les pods et services :

```
kubectl get all
```

```
vsritharan@srv-deb:~/example-voting-app/k8s-specifications$ kubectl get all
NAME                          READY   STATUS    RESTARTS   AGE
pod/db-74574d66dd-jp2x5       1/1     Running   0          41s
pod/redis-6c5fb9c4b7-77xs7    1/1     Running   0          41s
pod/result-5f99548f7c-wmqhg   1/1     Running   0          41s
pod/vote-5d74dcd7c7-5rj8d     1/1     Running   0          41s
pod/worker-6f5f6cdd56-gx55f   1/1     Running   0          41s

NAME                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
service/db           ClusterIP   10.97.11.161     <none>        5432/TCP         41s
service/kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP          18h
service/redis        ClusterIP   10.103.237.185   <none>        6379/TCP         41s
service/result       NodePort    10.107.42.131    <none>        8081:31001/TCP   41s
service/vote         NodePort    10.98.232.250    <none>        8080:31000/TCP   41s

NAME                     READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/db       1/1     1            1           41s
deployment.apps/redis    1/1     1            1           41s
deployment.apps/result   1/1     1            1           41s
deployment.apps/vote     1/1     1            1           41s
deployment.apps/worker   1/1     1            1           41s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/db-74574d66dd       1         1         1       41s
replicaset.apps/redis-6c5fb9c4b7    1         1         1       41s
replicaset.apps/result-5f99548f7c   1         1         1       41s
replicaset.apps/vote-5d74dcd7c7     1         1         1       41s
replicaset.apps/worker-6f5f6cdd56   1         1         1       41s
```

---

5. Récupérer l'adresse IP du cluster Minikube

J'ai utilisé cette commande pour connaître l'adresse IP de Minikube :

```
minikube ip
```

```
vsritharan@srv-deb:~/example-voting-app/k8s-specifications$ minikube ip
192.168.58.2
```

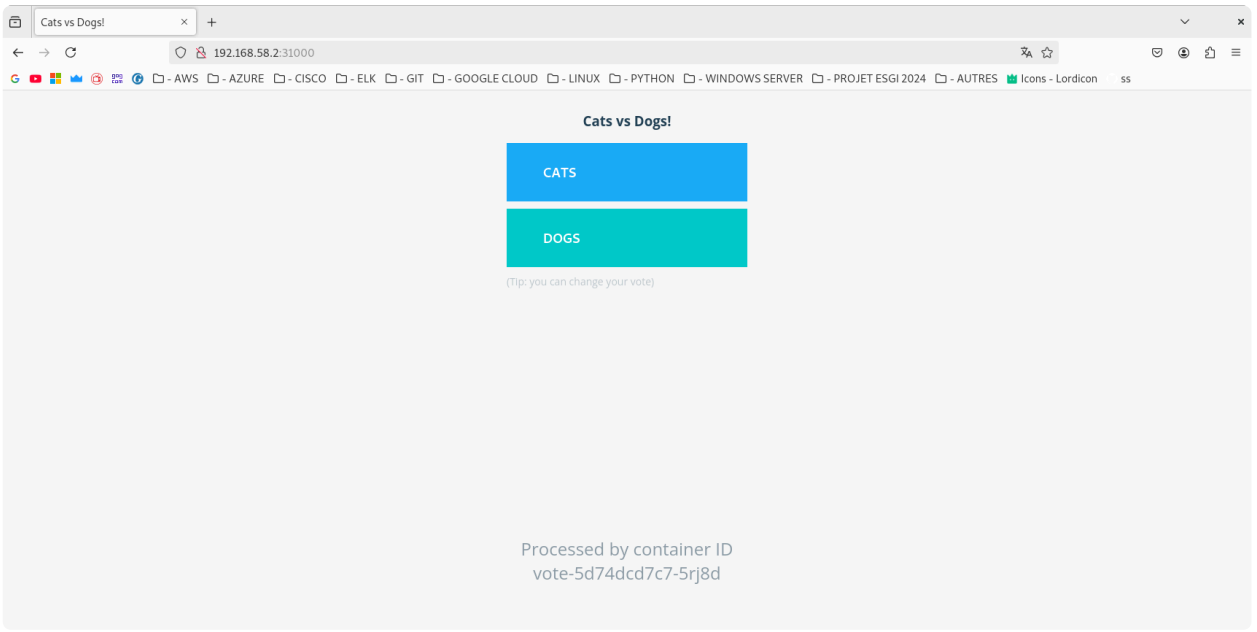
6. Accéder aux interfaces vote/result

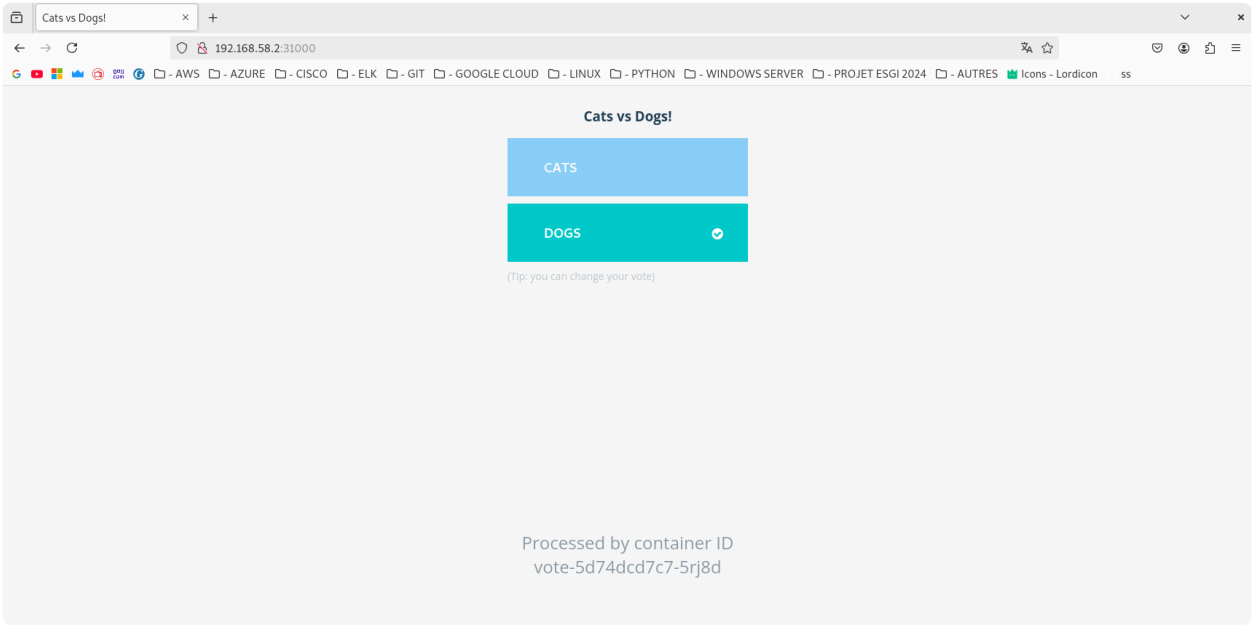
En utilisant l'IP récupérée précédemment et les NodePort associés aux services, j'ai pu accéder aux interfaces suivantes :

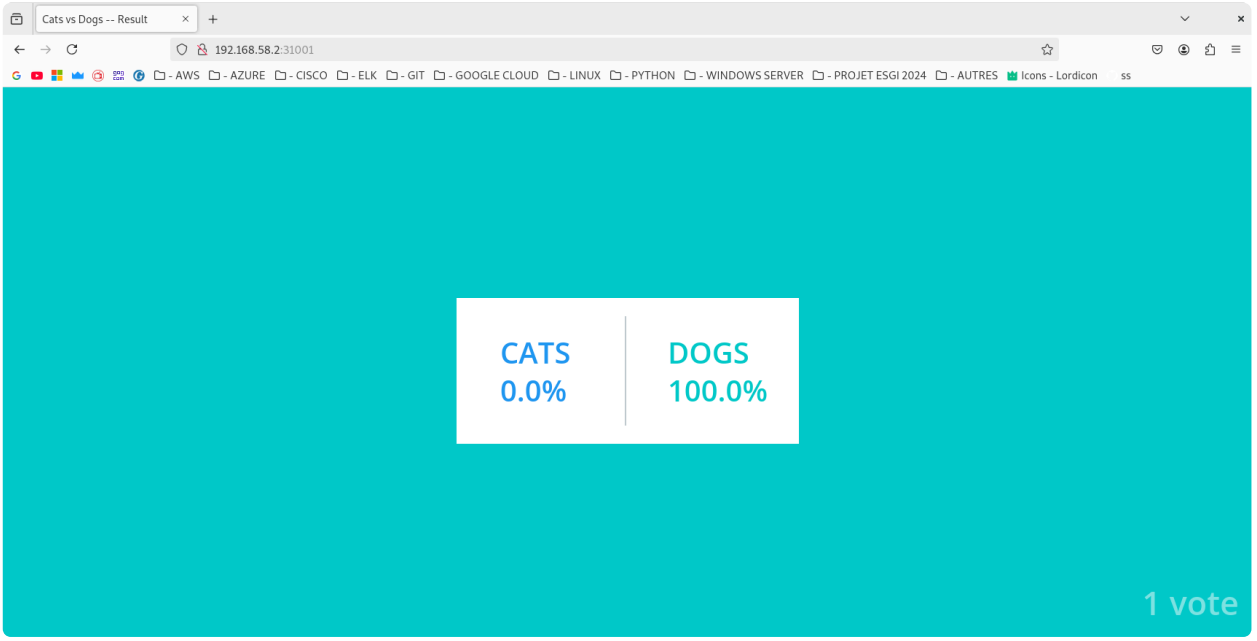
```
kubectl get svc
```

```
vsritharan@srv-deb:~/example-voting-app/k8s-specifications$ kubectl get svc
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
db           ClusterIP   10.97.11.161     <none>        5432/TCP         12m
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP          18h
redis        ClusterIP   10.103.237.185   <none>        6379/TCP         12m
result       NodePort    10.107.42.131    <none>        8081:31001/TCP   12m
vote         NodePort    10.98.232.250    <none>        8080:31000/TCP   12m
```

- vote : http://192.168.58.2:31000
- result : http://192.168.58.2:31001







Résultats obtenus

Après avoir testé, j'ai pu constater que :

- L'interface `vote` permet de voter
- Les votes passent bien par Redis et sont traités par le worker
- Les résultats apparaissent en temps réel dans `result`

Nettoyage

À la fin, j'ai pu supprimer les ressources créées avec :

```
kubectl delete -f .
minikube stop
```

```
vsritharan@srv-deb:~/example-voting-app/k8s-specifications$ kubectl delete -f .
deployment.apps "db" deleted
service "db" deleted
deployment.apps "redis" deleted
service "redis" deleted
deployment.apps "result" deleted
service "result" deleted
deployment.apps "vote" deleted
service "vote" deleted
deployment.apps "worker" deleted
```

Conclusion

Ce projet m'a permis de :

- Comprendre une architecture microservices distribuée
- Déployer des composants Docker dans Kubernetes
- Observer les échanges entre Redis, PostgreSQL, et les applications frontend/backend
- Documenter chaque étape de façon claire

Merci **M Ulrich NOUMSI** pour ce cours !