

# REVERSE ENGINEERING – Déploiement & Design

Kabli Mehdi

5SRC2

Pour commencer proprement notre projet nous procédons à l'installation de minikube ensuite nous clonons notre projet puis ensuite nous le démarrons

```
mkabli@ubuntuserv:~/ProjetCC/example-voting-app/k8s-specifications$ minikube start --driver=docker
* minikube v1.35.0 sur Ubuntu 24.04
* Utilisation du pilote docker basé sur le profil existant
* Démarrage du nœud "minikube" primary control-plane dans le cluster "minikube"
* Extraction de l'image de base v0.0.46...
* Redémarrage du docker container existant pour "minikube" ...
* Préparation de Kubernetes v1.32.0 sur Docker 27.4.1...
* Vérification des composants Kubernetes...
  - Utilisation de l'image gcr.io/k8s-minikube/storage-provisioner:v5
* Modules activés: default-storageclass, storage-provisioner
* Terminé ! kubectl est maintenant configuré pour utiliser "minikube" cluster et espace de noms "default" par défaut.
mkabli@ubuntuserv:~/ProjetCC/example-voting-app/k8s-specifications$
```

Nous avons récupéré le code de l'application voting-app depuis GitHub afin de déployer l'architecture via Kubernetes

```
git clone https://github.com/docker-samples/example-voting-app.git
```

```
cd example-voting-app/k8s-specifications
```

Notre espace de travail est maintenant prêt

Une fois Minikube démarré et les fichiers YAML disponibles, nous avons utilisé la commande suivante pour déployer l'ensemble des composants définis dans le dossier k8s-specifications

```
mkabli@ubuntuserv:~/ProjetCC/example-voting-app/k8s-specifications$ kubectl apply -f .
deployment.apps/db unchanged
service/db unchanged
deployment.apps/redis unchanged
service/redis unchanged
deployment.apps/result unchanged
service/result unchanged
deployment.apps/vote unchanged
service/vote unchanged
deployment.apps/worker unchanged
mkabli@ubuntuserv:~/ProjetCC/example-voting-app/k8s-specifications$
```

Cette commande applique l'ensemble des manifestes Kubernetes, incluant les *Deployments* et *Services* pour chaque composant de l'application (Vote, Result, Redis, PostgreSQL, Worker).

Dans notre cas, les ressources étaient déjà en place, ce qui explique l'état unchanged affiché pour chaque élément.

## Vérification du bon déploiement des pods et des services

La commande suivante permet de lister l'ensemble des pods en cours d'exécution :

```
mkabli@ubuntuser:~/ProjetCC/example-voting-app/k8s-specifications$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
db-74574d66dd-qxvdb                1/1     Running   3 (20m ago) 3d6h
redis-6c5fb9c4b7-2hk9s             1/1     Running   3 (20m ago) 3d6h
result-5f99548f7c-qlpzb           1/1     Running   4 (20m ago) 3d6h
vote-5d74dcd7c7-m29mb              1/1     Running   2 (20m ago) 3d5h
worker-6f5f6cdd56-9f9fz            1/1     Running   9 (20m ago) 3d6h
mkabli@ubuntuser:~/ProjetCC/example-voting-app/k8s-specifications$
```

Tous les pods attendus sont bien présents (db, redis, vote, result, worker) et ont le statut Running, ce qui confirme que les containers ont été correctement lancés.

Ensuite, nous avons utilisé :

```
mkabli@ubuntuser:~/ProjetCC/example-voting-app/k8s-specifications$ kubectl get svc
NAME            TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
db              ClusterIP     10.105.40.234   <none>           5432/TCP         3d6h
kubernetes      ClusterIP     10.96.0.1       <none>           443/TCP          61d
redis           ClusterIP     10.99.87.44     <none>           6379/TCP         3d6h
result          NodePort      10.103.218.229  <none>           8081:31001/TCP   3d6h
vote            NodePort      10.110.75.106   <none>           8080:31000/TCP   3d6h
mkabli@ubuntuser:~/ProjetCC/example-voting-app/k8s-specifications$
```

Cette commande affiche la configuration des services associés. On observe :

- Des services de type ClusterIP pour db et redis, car ils sont utilisés **uniquement en interne**.
- Des services de type NodePort pour vote et result, permettant d'y accéder **depuis l'extérieur** sur les ports 31000 et 31001 respectivement.

Ces services jouent un rôle clé dans l'exposition de l'interface web et la récupération des résultats de vote.

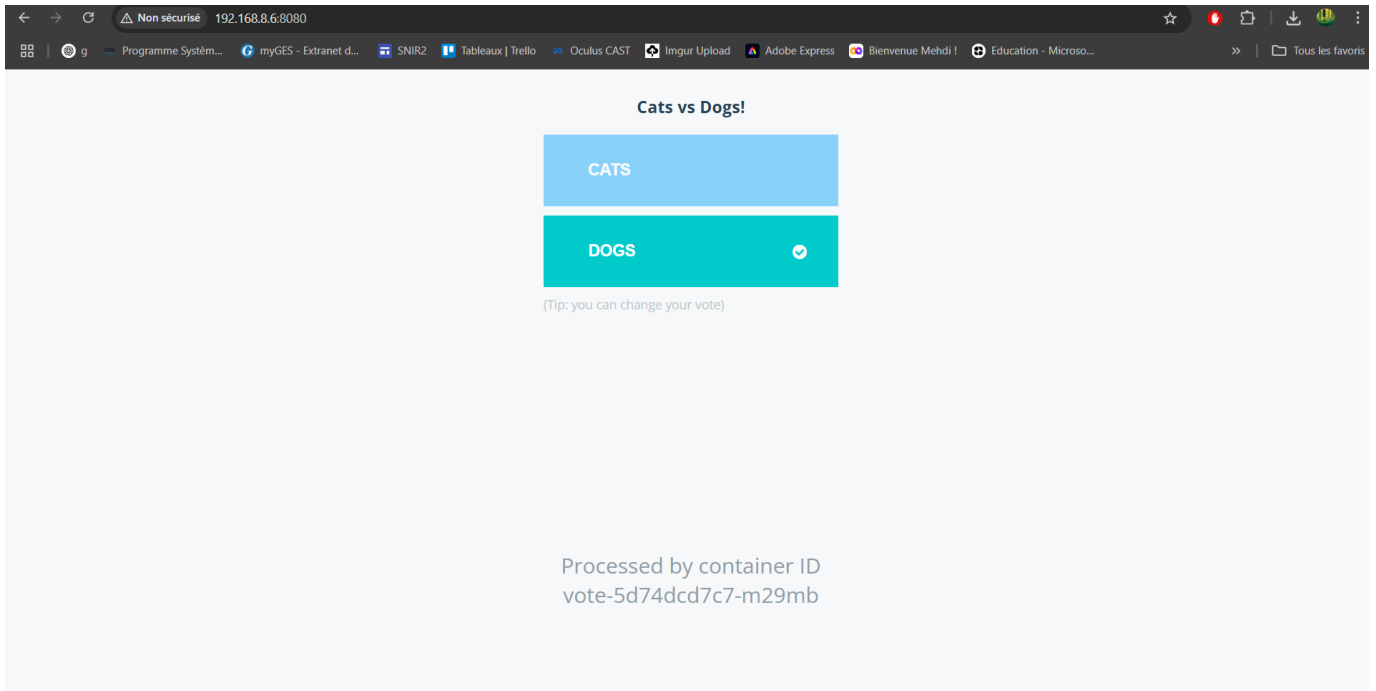
**L'architecture** est désormais en place et chaque composant dispose d'un point d'accès dédié, prêt à recevoir du trafic ou des connexions d'autres pods.

Minikube utilisant le driver Docker, il isole les services exposés en NodePort du réseau local.

```
mkabli@ubuntu:~/ProjetCC/example-voting-app/k8s-specifications$ kubectl port-forward svc/vote 8080:8080 --address 0.0.0.0
Forwarding from 0.0.0.0:8080 -> 80
Handling connection for 8080
Handling connection for 8080
```

Pour contourner cette limitation, nous avons redirigé manuellement le port 8080 du service vote vers l'extérieur via la commande `kubectl port-forward`.

Cela permet d'accéder à l'interface de vote depuis le navigateur hôte à l'adresse suivante :

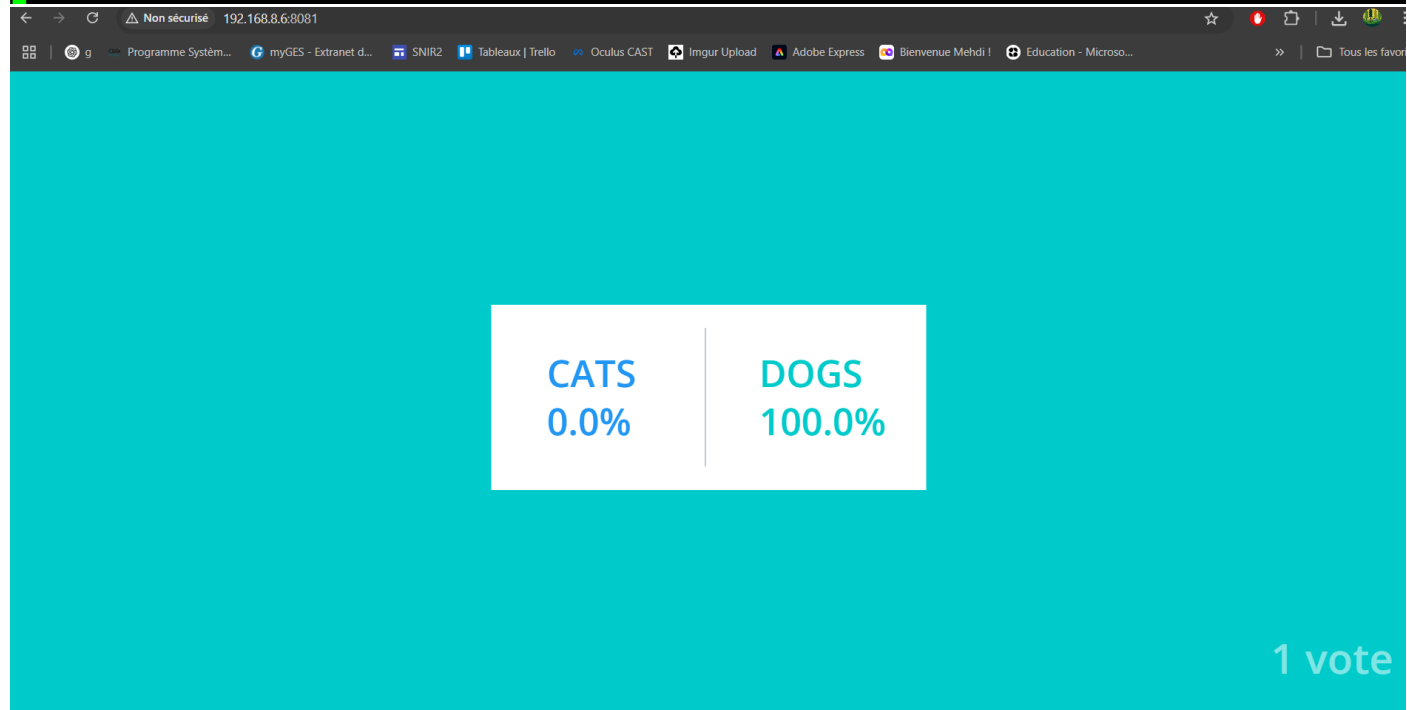


Ci-dessous, l'interface web du service vote est accessible depuis le PC hôte.

L'utilisateur peut sélectionner une option et soumettre son vote.

Le container ayant traité la requête est affiché dynamiquement en bas de page.

```
mkabli@ubuntuser:~/ProjetCC/example-voting-app/k8s-specifications$ kubectl port-forward svc/result 8081
:8081 --address 0.0.0.0
Forwarding from 0.0.0.0:8081 -> 80
Handling connection for 8081
Handling connection for 8081
Handling connection for 8081
Handling connection for 8081
Handling connection for 8081
```



Option	Percentage
CATS	0.0%
DOGS	100.0%

1 vote

L'application result permet d'afficher en temps réel les résultats du vote.

On observe la mise à jour immédiate après validation d'un vote.

