

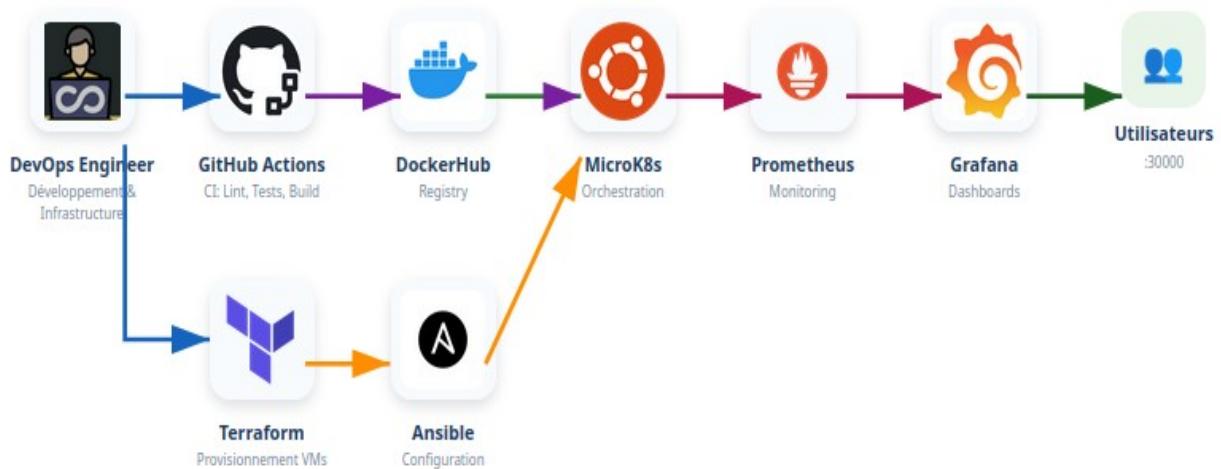
## **Vote App - Rapport Projet Fil Rouge DevOps**

Mbainaissem Narcisse MEDION

## Introduction

Ce projet illustre une chaîne DevOps complète: développement, déploiement, supervision, sécurité. L'objectif principal est de prouver la maîtrise de chaque brique moderne: CI/CD, IaC, orchestration, monitoring, et sécurité.

## Architecture du Projet



### Découpage et choix des technologies

- **Docker**: Containerisation universelle pour le backend, frontend, et Redis (portabilité, reproductibilité).
- **GitHub Actions**: Plateforme CI solide, native GitHub, facile à intégrer pour tests/lints/builds automatiques.
- **Terraform**: Provisionnement IaC, création dynamique des VMs (Proxmox), documenté, versionnable.
- **Ansible**: Configuration automatique, installation des systèmes/tools (Docker, MicroK8s), idéal pour clusters.
- **MicroK8s**: Orchestration Kubernetes locale, légère, facile à script Ansible.
- **Prometheus & Grafana**: Monitoring temps réel, dashboards, alertes, visualisation de tout le cluster.

- **Scripts Bash**: permettent un CD manuel, sécurisé (sudo local, jamais ouvert au public).

## Difficultés rencontrées

- **Restrictions réseau**: Terraform et Ansible impossibles à exécuter en CI GitHub car Proxmox & K8s sont sur un réseau local non exposé.
- **Sécurité des accès**: impossibilité de tout automatiser sans prise de risque (mot de passe sudo, non-exposition des VM), nécessité de scripts manuels.
- **Interopérabilité**: Compilation et tests sur plusieurs environnements, gestion des variables sensibles.
- **Monitoring**: Configuration avancée Prometheus/Grafana, tuning des alertes pour cluster homogène.

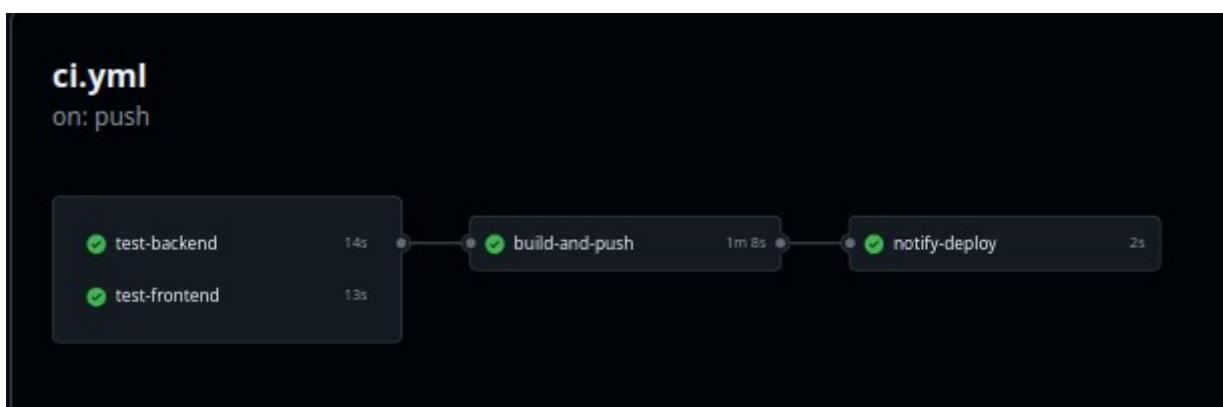
## Déroulement détaillé

### 1. Containerisation

- Dockerfile multistage, docker-compose pour toute la stack en mode dev.
- Build d'images déployables sur DockerHub.

### 2. CI (GitHub Actions)

- Lint, tests, build/push des images sur DockerHub à chaque commit.



### 3. IaC (Terraform)

- Création des VMs (1 master + 2 workers) via Proxmox.
- Inventaire dynamique pour Ansible.

### 4. Configuration (Ansible)

- Installation Docker, MicroK8s, configuration système et users, activation des addons K8s.

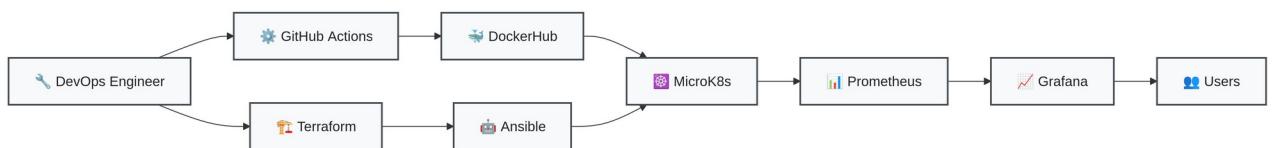
## 5. Déploiement Applicatif + Supervision

- Déploiement du backend, frontend, Redis via playbooks.
- Installation auto de Grafana & Prometheus, création du dashboard cluster.
- Activation du HPA (Horizontal Pod Autoscaler), test de charge.

## 6. CD manuel (script bash sécurisé)

- Un script local permet de déployer toute la stack avec sécurité (mot de passe sudo, accès local uniquement).

### Architecture visuelle



### Monitoring & supervision

- Grafana montre :
  - Nodes, pods, CPU/RAM, redémarrages, Health Pods
  - Alertes configurables pour toute l'infra
- Prometheus collecte et expose toutes les métriques du cluster

```
cisco@k8s-master: ~
cisco@cisco: ~/vote-app/infra/ansible      x      cisco@k8s-master: ~      x      v
Every 2,0s: microk8s kubectl get nodes      k8s-master: Sat Oct 25 23:02:20 2025
NAME        STATUS    ROLES      AGE        VERSION
k8s-master   Ready     <none>    18m       v1.32.9
k8s-worker1  Ready     <none>    7m35s    v1.32.9
k8s-worker2  Ready     <none>    5m44s    v1.32.9
```

Figure 1: Nodes du cluster

```
cisco@k8s-master: ~
cisco@cisco: ~/vote-app/infra/ansible      x      cisco@k8s-master: ~      x      v
cisco@cisco: ~/vote-app/infra/ansible      x      cisco@k8s-master: ~      x      v
cisco@cisco: ~$ microk8s kubectl get all -n vote-app
NAME          READY   STATUS    RESTARTS   AGE
pod/redis-864f65544-9klth   1/1     Running   0          2m9s
pod/vote-backend-78bf458f66-fg9sb   1/1     Running   0          2m7s
pod/vote-backend-78bf458f66-n2bw6   1/1     Running   0          2m7s
pod/vote-frontend-69f7fb4bcd-45lxc   1/1     Running   0          2m6s
pod/vote-frontend-69f7fb4bcd-qxfnk   1/1     Running   0          2m6s

NAME          TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
service/redis   ClusterIP  10.152.183.190  <none>        6379/TCP      2m9s
service/vote-backend  ClusterIP  10.152.183.124  <none>        8000/TCP      2m8s
service/vote-frontend  NodePort   10.152.183.243  <none>        3000:30001/TCP  2m6s

NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/redis   1/1     1           1           2m9s
deployment.apps/vote-backend  2/2     2           2           2m8s
deployment.apps/vote-frontend  2/2     2           2           2m6s

NAME          DESIRED  CURRENT   READY   AGE
replicaset.apps/redis-864f65544  1        1        1        2m9s
replicaset.apps/vote-backend-78bf458f66  2        2        2        2m8s
replicaset.apps/vote-frontend-69f7fb4bcd  2        2        2        2m6s
cisco@cisco: ~$ microk8s kubectl get pods -n vote-app -o wide
NAME          READY   STATUS    RESTARTS   AGE      IP          NODE      NOMINATED NODE      READINESS GATES
redis-864f65544-9klth   1/1     Running   0          3m13s  10.1.126.1  k8s-worker2  <none>        <none>
vote-backend-78bf458f66-fg9sb   1/1     Running   0          3m11s  10.1.194.65  k8s-worker1  <none>        <none>
vote-backend-78bf458f66-n2bw6   1/1     Running   0          3m11s  10.1.126.2  k8s-worker2  <none>        <none>
vote-frontend-69f7fb4bcd-45lxc   1/1     Running   0          3m10s  10.1.194.66  k8s-worker1  <none>        <none>
vote-frontend-69f7fb4bcd-qxfnk   1/1     Running   0          3m10s  10.1.126.3  k8s-worker2  <none>        <none>
```

Figure 2 : État du déploiement

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
vote-backend-hpa	Deployment/vote-backend	cpu: 2%/50%	2	5	2	63m
vote-backend-hpa	Deployment/vote-backend	cpu: 45%/50%	2	5	2	63m
vote-backend-hpa	Deployment/vote-backend	cpu: 74%/50%	2	5	2	63m
vote-backend-hpa	Deployment/vote-backend	cpu: 72%/50%	2	5	3	63m
vote-backend-hpa	Deployment/vote-backend	cpu: 51%/50%	2	5	3	64m
vote-backend-hpa	Deployment/vote-backend	cpu: 52%/50%	2	5	3	64m
vote-backend-hpa	Deployment/vote-backend	cpu: 51%/50%	2	5	3	65m
vote-backend-hpa	Deployment/vote-backend	cpu: 52%/50%	2	5	3	66m
vote-backend-hpa	Deployment/vote-backend	cpu: 50%/50%	2	5	3	66m
vote-backend-hpa	Deployment/vote-backend	cpu: 52%/50%	2	5	3	66m
vote-backend-hpa	Deployment/vote-backend	cpu: 50%/50%	2	5	3	66m
vote-backend-hpa	Deployment/vote-backend	cpu: 51%/50%	2	5	3	67m
vote-backend-hpa	Deployment/vote-backend	cpu: 52%/50%	2	5	3	67m
vote-backend-hpa	Deployment/vote-backend	cpu: 51%/50%	2	5	3	67m
vote-backend-hpa	Deployment/vote-backend	cpu: 53%/50%	2	5	3	68m
vote-backend-hpa	Deployment/vote-backend	cpu: 24%/50%	2	5	3	68m
vote-backend-hpa	Deployment/vote-backend	cpu: 2%/50%	2	5	3	68m
vote-backend-hpa	Deployment/vote-backend	cpu: 2%/50%	2	5	3	73m
vote-backend-hpa	Deployment/vote-backend	cpu: 3%/50%	2	5	2	73m

Figure 3: Test de HPA

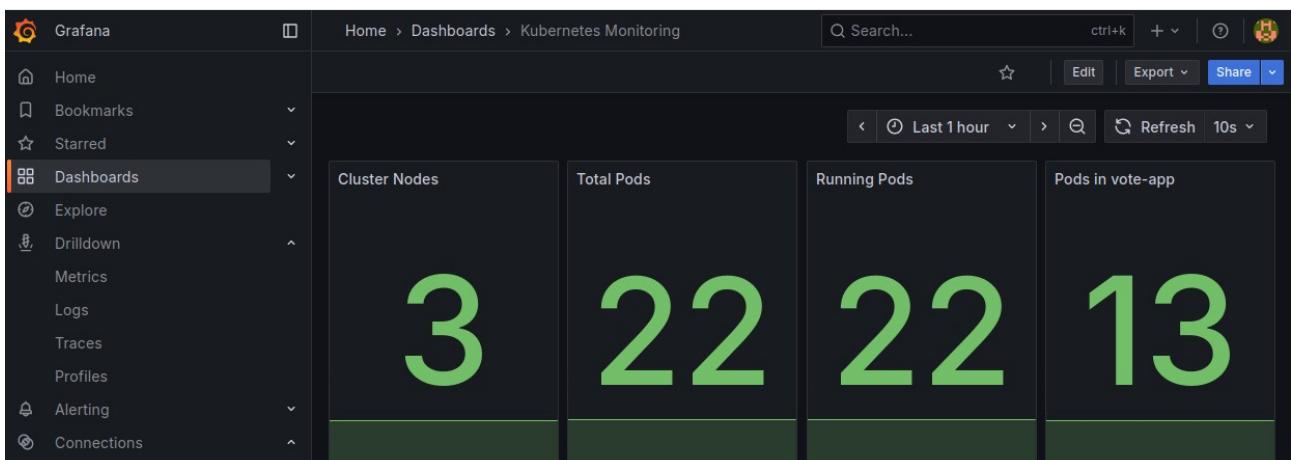


Figure 4: Nombre de pods

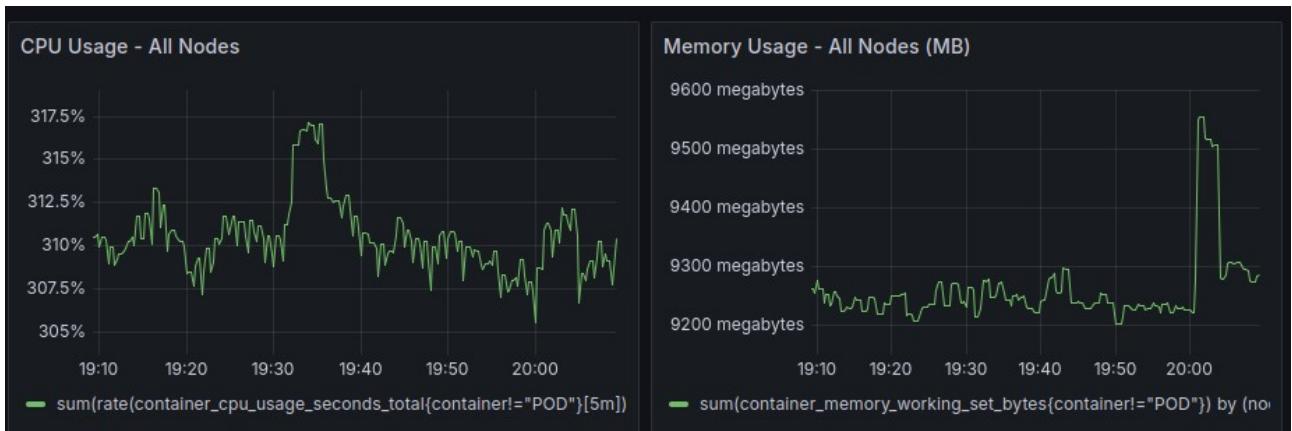


Figure 5: Taux d'utilisations du CPU et mémoires des Nodes

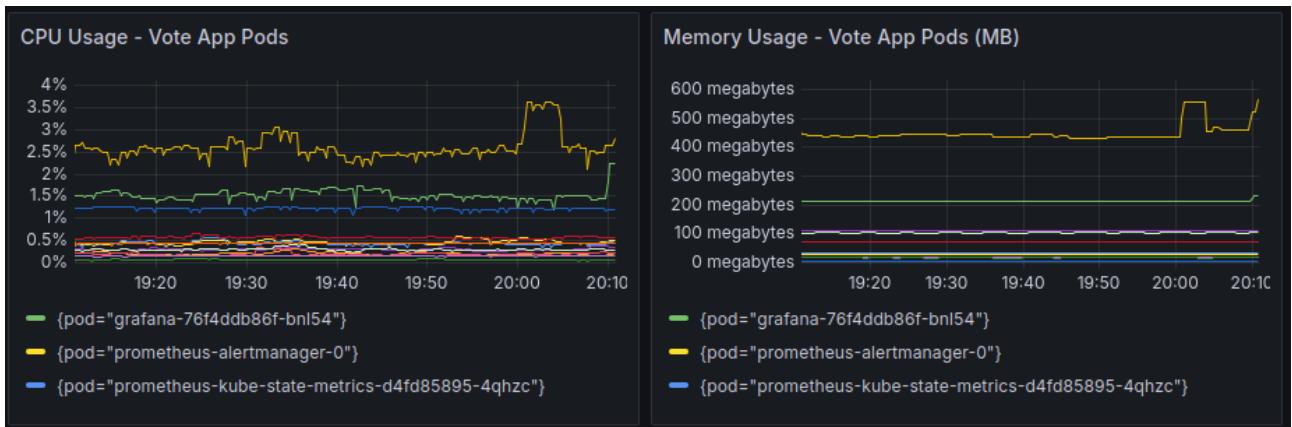


Figure 6: Taux d'utilisations du CPU et mémoires des Pods

## Sécurité et évolutivité

- Jamais d'accès root ou sudo automatisés sans prompt; tout reste local pour la partie critique.
- Extension possible vers cluster cloud (AWS) ou runner GitHub Action auto-hébergé pour une CI/CD full auto.
- Ajout possible de scans Trivy, SonarQube, Kube-bench, Vault à court terme.