Rapport – Déploiement de NotesApp sur Kubernetes avec Terraform (Azure VM)

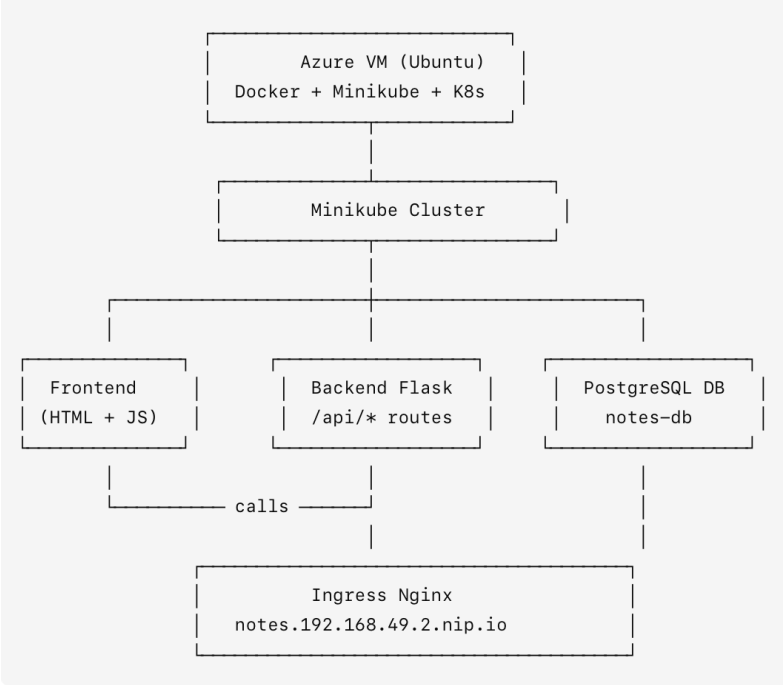
Zainab Touhami 127648

1. Introduction

Ce rapport présente le déploiement complet d'une application 3-tiers (Frontend, API Flask, Base PostgreSQL) sur un cluster Kubernetes Minikube hébergé dans une machine virtuelle Azure. L'objectif est de mettre en œuvre Docker, Kubernetes, Ingress Nginx et Terraform pour automatiser l'infrastructure.

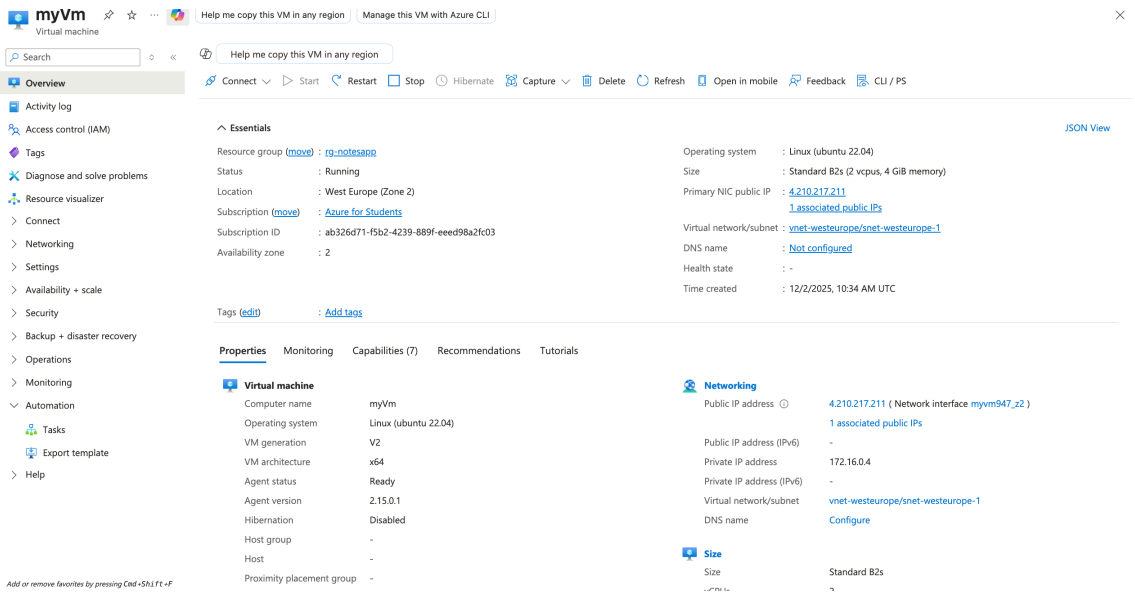
2. Architecture Globale

L'application se compose de trois éléments : un frontend HTML/JS, un backend Flask exposant des routes API, et une base PostgreSQL. Le tout est encapsulé dans des conteneurs Docker et orchestré via Kubernetes.



3. Création de la VM Azure

Une VM Ubuntu 22.04 a été créée sur Azure, utilisée comme base pour héberger Docker et Minikube.



4. Installation de l’environnement sur la VM

Les outils suivants ont été installés : Docker, Kubectl, Minikube.

Minikube a été lancé avec le driver Docker.

```
azureuser@myVm:~$ minikube start --driver=docker --memory=2500mb
😄  minikube v1.37.0 on Ubuntu 22.04
✨  Using the docker driver based on user configuration
📌  Using Docker driver with root privileges
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.48 ...
💾  Downloading Kubernetes v1.34.0 preload ...
    > gcr.io/k8s-minikube/kicbase...:  488.51 MiB / 488.52 MiB  100.00% 119.67
    > preloaded-images-k8s-v18-v1...:  337.07 MiB / 337.07 MiB  100.00% 25.05 M
🔥  Creating docker container (CPUs=2, Memory=2500MB) ...
🐳  Preparing Kubernetes v1.34.0 on Docker 28.4.0 ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
azureuser@myVm:~$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

```
azureuser@myVm:~/notesapp-project/terraform$ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
notes-api-6db8c47fc9-l9lhq        1/1     Running   0          36s
notes-db-6c45b97c68-8zzvs         1/1     Running   0          36s
notes-frontend-69d664d54f-s6dmh   1/1     Running   0          36s
```

5. Dockerisation de l'application

Les trois services (API, frontend, DB) ont été dockerisés.

Les images ont été chargées dans le registre Docker interne de Minikube.

```
azureuser@myVm:~/notesapp-project/app/db$ docker images | grep notes
WARNING: This output is designed for human readability. For machine-readable output, please use --forma
t.
notes-api:1.0                                23c7be6d9a5a        134MB             0B   U
notes-db:1.0                                 1440b2370f2d        441MB             0B   U
notes-frontend:1.0                           167fbd52ff62        152MB             0B   U
```

6. Déploiement Kubernetes via Terraform

Terraform a été utilisé pour déployer :

- Namespace

- Deployments

- Services

- Ingress Nginx

```
azureuser@myVm:~/notesapp-project/terraform$ terraform apply
kubernetes_service.api: Refreshing state... [id=notes/notes-api]
kubernetes_ingress_v1.notes_ingress: Refreshing state... [id=notes/notes-ingress]
kubernetes_namespace.notes: Refreshing state... [id=notes]
kubernetes_service.frontend: Refreshing state... [id=notes/notes-frontend]
kubernetes_deployment.frontend: Refreshing state... [id=notes/notes-frontend]
kubernetes_deployment.api: Refreshing state... [id=notes/notes-api]
kubernetes_deployment.db: Refreshing state... [id=notes/notes-db]

Terraform used the selected providers to generate the following execution plan. Resource actions are
indicated with the following symbols:
  ~ update in-place

Terraform will perform the following actions:
```

```
Apply complete! Resources: 0 added, 1 changed, 0 destroyed.
```

[CAPTURE : arborescence terraform/]

7. Vérification du cluster

Des commandes Kubernetes ont été exécutées pour vérifier la création des pods, services et ingress.

```
azureuser@myVm:~/notesapp-project/terraform$ kubectl get pods -n notes
kubectl get svc -n notes
kubectl get ingress -n notes
NAME                              READY   STATUS    RESTARTS        AGE
notes-api-6db8c47fc9-d79xz        1/1     Running   0               3m56s
notes-db-6c45b97c68-8zzvs         1/1     Running   1 (6h24m ago)   7h12m
notes-frontend-69d664d54f-s6dmh   1/1     Running   1 (6h24m ago)   7h12m
NAME             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
notes-api        ClusterIP   10.111.110.73   <none>        5000/TCP   7h12m
notes-frontend   ClusterIP   10.98.63.246    <none>        80/TCP     7h12m
NAME            CLASS   HOSTS                          ADDRESS        PORTS   AGE
notes-ingress   nginx   notes.192.168.49.2.nip.io   192.168.49.2   80      7h12m
```

8. Configuration de l'Ingress + nip.io

Un Ingress a été configuré pour exposer :

- '/' → frontend

- '/api' → backend Flask

L'accès se fait via : http://notes.<minikube-ip>.nip.io

```
azureuser@myVm:~/notesapp-project/terraform$ curl http://notes.192.168.49.2.nip.io
<!DOCTYPE html>
<html>
<head>
  <title>NotesApp</title>
</head>
<body>
  <h1>Notes App</h1>
  <form id="form">
    <input id="note" placeholder="Write a note" />
    <button type="submit">Add</button>
  </form>

  <ul id="notes"></ul>

<script>
async function loadNotes() {
  const res = await fetch("/api/notes");
  const data = await res.json();
  document.getElementById("notes").innerHTML =
    data.map(n => `<li>${n}</li>`).join("");
}

document.getElementById("form").addEventListener("submit", async (e) => {
  e.preventDefault();
  const note = document.getElementById("note").value;
  await fetch("/api/add", {
    method: "POST",
    headers: {"Content-Type": "application/json"},
    body: JSON.stringify({note})
  })
  loadNotes();
});

loadNotes();
</script>
</body>
</html>
```

9. Tests fonctionnels

Les tests ont été réalisés via curl : récupération et ajout de notes.

```
azureuser@myVm:~/notesapp-project/app/api$ curl http://notes.192.168.49.2.nip.io/api/
notes
[]
azureuser@myVm:~/notesapp-project/app/api$ curl -X POST http://notes.192.168.49.2.nip
.io/api/add \
  -H "Content-Type: application/json" \
  -d '{"note":"test"}'
{"message":"note added"}
azureuser@myVm:~/notesapp-project/app/api$ curl http://notes.192.168.49.2.nip.io/api/
notes
["test"]
```

10. Accès via navigateur

Le frontend a été rendu accessible sur Chrome grâce au port-forwarding Kubernetes.

Commande utilisée :

kubectl port-forward -n notes svc/notes-frontend 8080:80

Accès Chrome : http://<IP_VM>:8080

12. Conclusion

Le projet a permis de comprendre et maîtriser :

- Docker

- Kubernetes (Deployments, Services, Ingress)

- Terraform pour automatiser l'infrastructure

- Hébergement cloud sur Azure

- Communication entre services 3-tiers

L'application NotesApp fonctionne pleinement dans Kubernetes.