Fiche pratique Minikube & Kubernetes

Objectif

Manipuler Kubernetes sur un cluster local Minikube en utilisant : - `kubectl` direct (sans YAML) - Définition d'objets avec fichiers YAML

---

1. Lancer Minikube et vérifier le cluster

Sans YAML :

```
minikube start
kubectl cluster-info
kubectl get nodes
```

---

2. Namespace et déploiements

Sans YAML :

```
kubectl create namespace tp-kube
kubectl create deployment nginx1 --image=nginx -n tp-kube
kubectl get deployments -n tp-kube
kubectl get pods -n tp-kube
```

Avec YAML : - Créer `nginx2-deploy.yaml` :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx2
  namespace: tp-kube
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx2
  template:
    metadata:
      labels:
        app: nginx2
```

```
    spec:
      containers:
      - name: nginx2
        image: nginx
        ports:
        - containerPort: 80
```

- Déployer :

```
kubectl apply -f nginx2-deploy.yaml
kubectl get deployments -n tp-kube
kubectl get pods -n tp-kube
```

---

3. Scaling et inspection

Sans YAML :

```
kubectl scale deployment nginx2 --replicas=3 -n tp-kube
kubectl get pods -n tp-kube
kubectl describe pod <nom-du-pod> -n tp-kube
```

Avec YAML : - Modifier `nginx2-deploy.yaml` : `replicas: 3` - Appliquer :

```
kubectl apply -f nginx2-deploy.yaml
```

---

4. Exposer une application

Sans YAML :

```
kubectl expose deployment nginx2 --type=NodePort --port=80 -n tp-kube
kubectl get svc -n tp-kube
minikube service nginx2 -n tp-kube --url
```

Avec YAML : - Créer `nginx2-service.yaml` :

```
apiVersion: v1
kind: Service
metadata:
  name: nginx2-service
  namespace: tp-kube
spec:
```

```
  selector:
    app: nginx2
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 30080
  type: NodePort
```

- Déployer :

```
kubectl apply -f nginx2-service.yaml
kubectl get svc -n tp-kube
minikube service nginx2-service -n tp-kube --url
```

---

5. Vérifier les logs d'un pod

```
kubectl logs <nom-du-pod> -n tp-kube
kubectl logs -f <nom-du-pod> -n tp-kube
```

---

6. Tester l'exécution dans un conteneur

```
kubectl exec -it <nom-du-pod> -n tp-kube -- bash
```

---

7. Mettre à jour l'image d'un déploiement

```
kubectl set image deployment/nginx2 nginx2=nginx:1.25 -n tp-kube
kubectl describe deployment nginx2 -n tp-kube
```

---

8. Partie Troubleshooting – Exercices pratiques

Exercice 1 : ImagePullBackOff

- Créer `nginx3-deploy.yaml` :

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-errone
```

```
  namespace: tp-kube
spec:
  containers:
  - name: container-test
    image: nginx:red
    ports:
    - containerPort: 80
```

- Appliquer et corriger les erreurs :

```
kubectl apply -f nginx3-deploy.yaml
kubectl get pods -n tp-kube
```

Exercice 2 : CrashLoopBackOff

- Créer `crash-deploy.yaml` :

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-crash
  namespace: tp-kube
spec:
  containers:
  - name: container-bug
    image: busybox
    command: ["sh", "-c", "exit 1"]
```

- Appliquer et corriger les erreurs :

```
kubectl apply -f crash-deploy.yaml
kubectl get pods -n tp-kube
```

---

9. Nettoyage

```
kubectl delete namespace tp-kube
minikube stop
```