RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
École nationale supérieure d'informatique (ESI ex. INI)

Rapport de TP VCL

2ème Année Cycle Supérieur (2CS)
2022-2023

Option : Systèmes Informatiques et Logiciels (SIL)

TP6

ORCHESTRATION AVEC
KUBERNETE

Réalisé par :

- Gouasmia Malak
- Touhar Afnane

Table de matiere

Chapitre 1

Introduction generale

Kubernetes est un système open-source de gestion de conteneurs qui permet d'orchestrer et de déployer des applications à grande échelle. Il est devenu l'un des outils les plus populaires pour la gestion de clusters de conteneurs dans les entreprises modernes. Dans ce TP, nous allons découvrir les concepts fondamentaux de Kubernetes et apprendre à utiliser ses fonctionnalités pour déployer une application simple sur un cluster. Nous verrons également comment résoudre les problèmes courants rencontrés lors de l'utilisation de Kubernetes en production.

Chapitre 2

1. PROCESSUS KUBERNETES

1.1. Créer un cluster Kubernetes

1.1.1. Installation de Minikube

```
$wget
https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
$sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

```
malak@malak-virtual-machine:~$  wget https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
--2023-01-13 18:02:05--  https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
Résolution de storage.googleapis.com (storage.googleapis.com)… 142.251.37.176, 142.251.37.208, 142.251.37.240, ...
Connexion à storage.googleapis.com (storage.googleapis.com)|142.251.37.176|:443… connecté.
requête HTTP transmise, en attente de la réponse… 200 OK
Taille : 76750347 (73M) [application/octet-stream]
Enregistre : ‘minikube-linux-amd64’

minikube-linux-amd6 100%[===================>]  73.19M  1.55MB/s    ds 56s

2023-01-13 18:03:02 (1.30 MB/s) - ‘minikube-linux-amd64’ enregistré [76750347/76750347]

malak@malak-virtual-machine:~$ sudo cp minikube-linux-amd64 /usr/local/bin/minikube
malak@malak-virtual-machine:~$ sudo chmod +x /usr/local/bin/minikube
malak@malak-virtual-machine:~$ minikube version
minikube version: v1.28.0
commit: 986b1ebd987211ed16f8cc10aed7d2c42fc8392f
```

1.1.2. Installation de kubectl

Après l’installation de minikube , nous allons installer kubectl, ce dernier est un outil qui permet à l'utilisateur de communiquer avec le contrôleur du cluster en utilisant l'API de Kubernetes via une interface de ligne de commande.

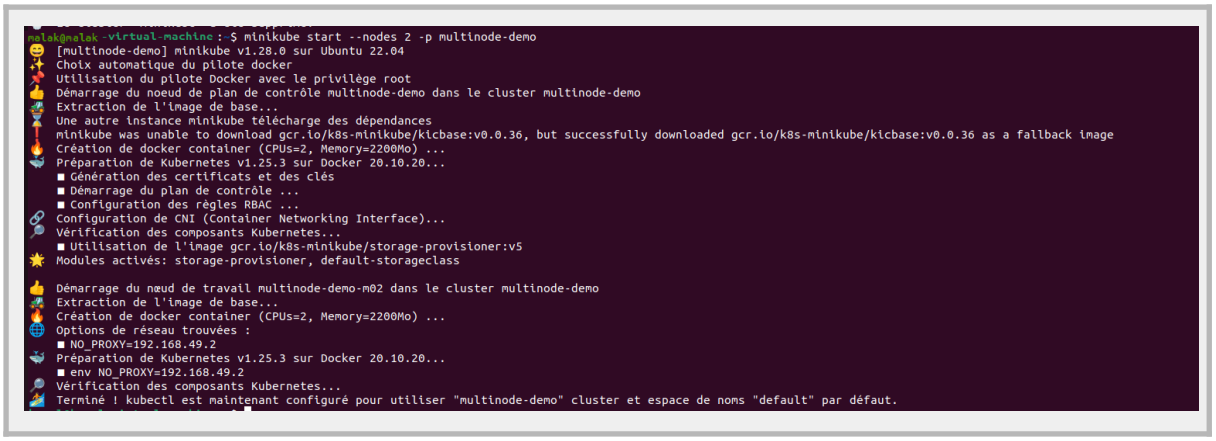
```
malak@malak-virtual-machine:~$ curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s https://storage.googleapis.com/kubernetes-release/re
xt`/bin/linux/amd64/kubectl
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 45.7M  100 45.7M    0     0  1558k      0  0:00:30  0:00:30 --:--:-- 1574k
malak@malak-virtual-machine:~$ chmod +x kubectl
malak@malak-virtual-machine:~$ sudo mv kubectl /usr/local/bin/
malak@malak-virtual-machine:~$ kubectl version -o yaml
clientVersion:
  buildDate: "2022-12-08T19:58:30Z"
  compiler: gc
  gitCommit: b46a3f887ca979b1a5d14fd39cb1af43e7e5d12d
  gitTreeState: clean
  gitVersion: v1.26.0
  goVersion: go1.19.4
  major: "1"
  minor: "26"
  platform: linux/amd64
kustomizeVersion: v4.5.7

The connection to the server localhost:8080 was refused - did you specify the right host or port?
```

1.1.3. Démarrage de cluster

Nous allons maintenant démarrer un cluster Minikube avec 2 nœuds.

```
$minikube start –nodes 2 -p multinode-demo
```



1.1.4. Lancement de proxy

Pour pouvoir communiquer avec le contrôleur du cluster, nous allons lancer un proxy kubectl dans un autre terminal.

```
$kubectl proxy
```

Nous allons maintenant utiliser kubectl pour afficher les informations relatives au cluster.

```
$kubectl cluster-info
```

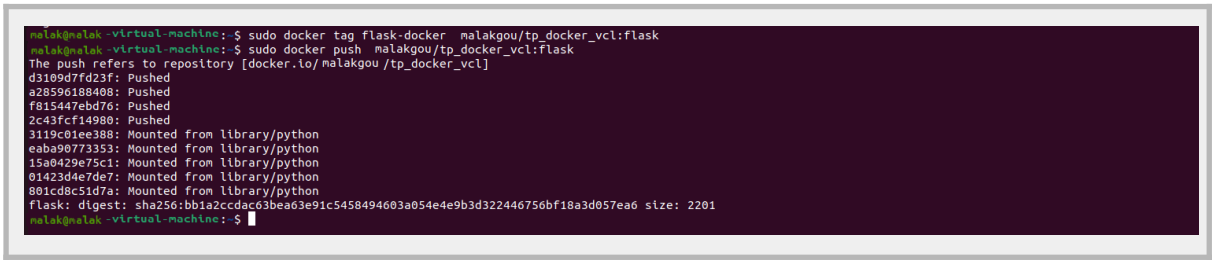
La commande kubectl nous informe que le cluster est en fonctionnement et nous fournit son adresse IP, le port sur lequel le contrôleur du cluster s'exécute, ainsi que l'URL pour accéder à celui-ci.

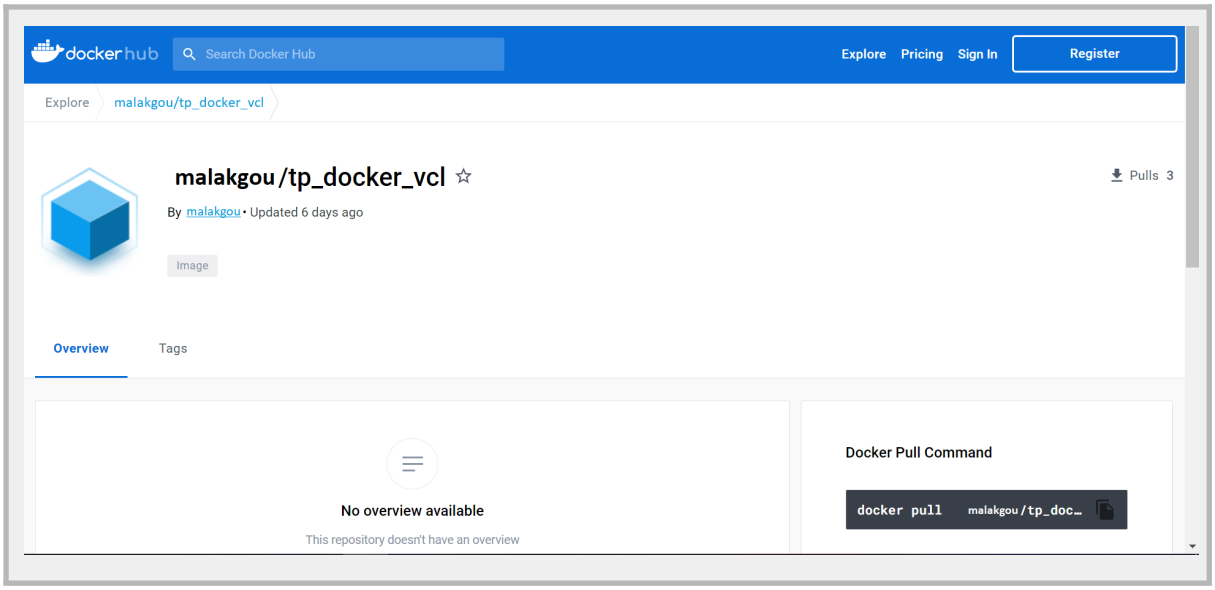
1.2. Publication d'une application

Pour publier notre image Flask sur Docker Hub, on doit d'abord se connecter à un compte Docker Hub à l'aide de la commande docker login. Ensuite taguer l' image avec le nom de du compte Docker Hub suivi du nom de l'image, en utilisant la commande docker tag. Enfin, on peut utiliser la commande docker push pour envoyer l'image sur Docker Hub.

Une fois que l'image est publiée sur Docker Hub, vous allez l'exploiter pour créer un conteneur en utilisant la commande docker run en spécifiant le nom de l'image.

```
sudo docker tag flask-docker malakgou/tp_docker_vcl:flask
```

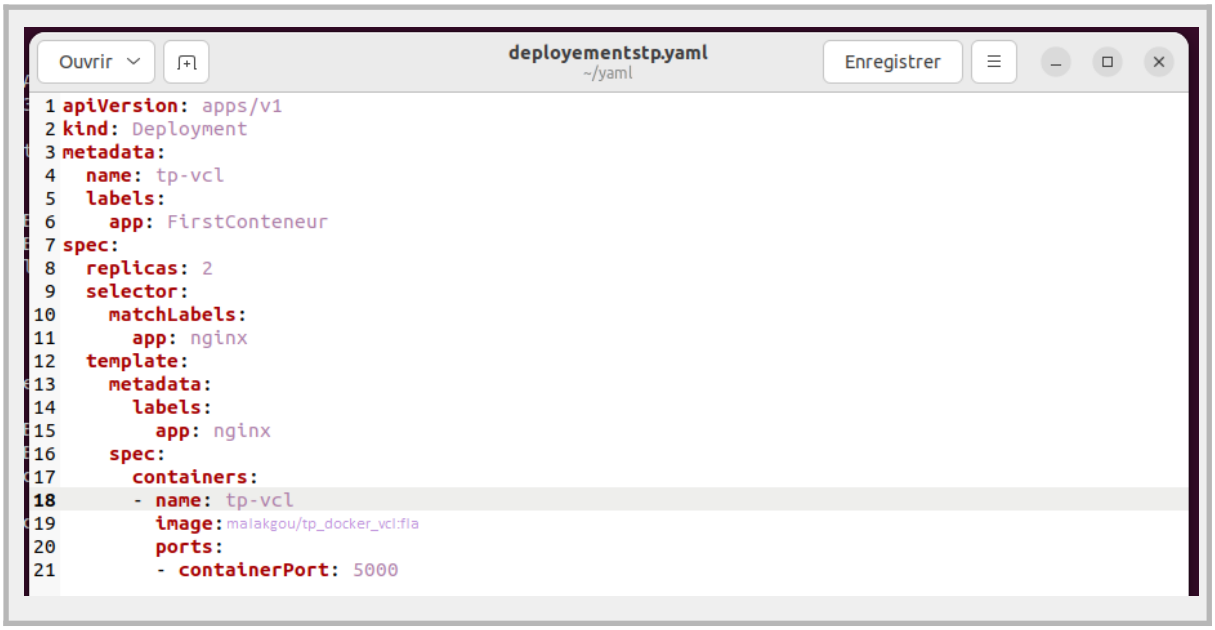




Voila, l'image docker est maintenant publiée .

1.3. Déploiement de l'application

Pour utiliser votre image Flask avec Minikube, on va créer un fichier de configuration yaml qui décrit comment Minikube doit créer et gérer le conteneur. Ce fichier doit inclure des informations telles que le nom du conteneur, l'image à utiliser, les ports à exposer, etc. Une fois ce fichier créé,on utilisera la commande kubectl create pour créer le conteneur sur Minikube, à partir de la configuration décrite dans le fichier yaml.



En utilisant Minikube,on va configurer notre conteneur pour qu'il soit répliqué sur plusieurs nœuds du cluster. Pour ce faire, il faut indiquer dans le fichier de configuration yaml le nombre de réplicas souhaitées.
En spécifiant 2 réplicas, le conteneur sera déployé sur les deux nœuds disponibles dans le cluster Minikube local.

Le conteneur déployé sur Minikube sera nommé "tp-vcl" et utilisera l'image Flask que vous avez créée et publiée précédemment sur Docker Hub. Il utilisera également le port 5000 pour permettre les communications locales dans le cluster Minikube.

```
gedit yaml/deploymentstp.yaml
```

```
malak@malak-virtual-machine:~$ gedit yaml/deployementstp.yaml
malak@malak-virtual-machine:~$ kubectl apply -f yaml/deployementstp.yaml
deployment.apps/tp-vcl created
malak@malak-virtual-machine:~$ kubectl get deployments
```

Le conteneur a été bien déployé au niveau du cluster

```
kubectl get deployments
```

```
malak@malak-virtual-machine:~$ kubectl get deployments
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
tp-vcl   2/2     2            2           85s
malak@malak-virtual-machine:~$ kubectl get pods
```

On a affiche les pods avec la commande suivante

```
kubectl get pods
```

```
malak@malak-virtual-machine:~$ kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
tp-vcl-c487969ff-dkq77    1/1     Running   0          88s
tp-vcl-c487969ff-pth8p    1/1     Running   0          88s
malak@malak-virtual-machine:~$
```

Pour déployer l'application sur deux instances Minikube, il nous faut r deux pods. Chacun de ces pods est situé sur un nœud différent et contient une réplique du conteneur qui utilise l'image Flask et le port 5000. Minikube assigne des adresses IP uniques à chaque pod pour permettre les communications entre les différents éléments du cluster.

```
$ kuberctl describe pods
```

Pods 1:

```
malak@malak-virtual-machine:~$ kubectl describe pods
Name:             tp-vcl-c487969ff-dkq77
Namespace:        default
Priority:         0
Service Account:  default
Node:             multinode-demo/192.168.49.2
Start Time:       Fri, 13 Jan 2023 20:25:40 +0100
Labels:           app=nginx
                  pod-template-hash=c487969ff
Annotations:      <none>
Status:           Running
IP:               10.244.0.5
IPs:
  IP:           10.244.0.5
Controlled By:  ReplicaSet/tp-vcl-c487969ff
Containers:
  tp-vcl:
    Container ID:   docker://f47c276593f34065634490ca74eab59d2e97eb9610e2b379e3e7982bba8de02f
    Image:          kamelferr/tp_docker_vcl:flask
    Image ID:       docker-pullable://malakgou/tp_docker_vcl@sha256:bb1a2ccdac63bea63e91c5458494603a054e4e9b3d322446756bf18a3d057ea6
    Port:           5000/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 13 Jan 2023 20:26:52 +0100
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-hhp6m (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
```

Pods 2:

```
Name:             tp-vcl-c487969ff-pth8p
Namespace:        default
Priority:         0
Service Account:  default
Node:             multinode-demo-m02/192.168.49.3
Start Time:       Fri, 13 Jan 2023 20:25:40 +0100
Labels:           app=nginx
                  pod-template-hash=c487969ff
Annotations:      <none>
Status:           Running
IP:               10.244.1.6
IPs:
  IP:           10.244.1.6
Controlled By:  ReplicaSet/tp-vcl-c487969ff
Containers:
  tp-vcl:
    Container ID:   docker://3f65bcb8c3ebad496f4064e028bfdf51b6850ae9806629d8baef6ced1c495914
    Image:          kamelferr/tp_docker_vcl:flask
    Image ID:       docker-pullable:// malakgou/tp_docker_vcl@sha256:bb1a2ccdac63bea63e91c5458494603a054e4e9b3d322446756bf18a3d057ea6
    Port:           5000/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 13 Jan 2023 20:26:48 +0100
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-q5hrw (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
```

Pour rendre notre application opérationnelle, on va exécuter le pod créé précédemment. en utilisant **kubectl exec** pour lancer le pod.

```
$kubectl exec tp-vcl-c487969ff-dkq 77 - -  env
```

```
malak@malak-virtual-machine:~$ kubectl exec tp-vcl-c487969ff-dkq77 -- env
PATH=/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=tp-vcl-c487969ff-dkq77
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_SERVICE_PORT=443
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
LANG=C.UTF-8
GPG_KEY=E3FF2839C048B25C084DEBE9B26995E310250568
PYTHON_VERSION=3.8.16
PYTHON_PIP_VERSION=22.0.4
PYTHON_SETUPTOOLS_VERSION=57.5.0
PYTHON_GET_PIP_URL=https://github.com/pypa/get-pip/raw/66030fa03382b4914d4c4d0896961a0bdeeeb274/public/get-pip.py
PYTHON_GET_PIP_SHA256=1e501cf004eac1b7eb1f97266d28f995ae835d30250bec7f8850562703067dc6
HOME=/root
```

Et pour accéder au pod localement on exécutera la commande suivante:

```
$kubectl exec -ti tp-vcl-c487969ff-dkq 77 - -  bash
```

```
malak@malak-virtual-machine:~$ kubectl exec -ti tp-vcl-c487969ff-dkq77 -- bash
root@tp-vcl-c487969ff-dkq77:/app# ls
Dockerfile  __pycache__  flaskapp.py  requirements.txt
root@tp-vcl-c487969ff-dkq77:/app#
```

Notre application fonctionne correctement, mais actuellement elle n'est accessible qu'à l'intérieur du cluster Minikube. Pour rendre l'application accessible à l'extérieur du cluster, on doit créer un service en utilisant la commande **kuberctl get services** qui va exposer les ports de l'application et permettre une communication avec l'extérieur. Cela permettra à l'application d'être accessible depuis l'extérieur du cluster Minikube.

```
$kuber ctl get services
```

```
malak@malak-virtual-machine:~$ kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   96m
malak@malak-virtual-machine:~$
```

Cela permettra de connecter le port 5000 utilisé à l'intérieur du conteneur à un port exposé au niveau du cluster, ici le port 8081. Cela permettra à l'application de devenir accessible depuis l'extérieur du cluster Minikube.

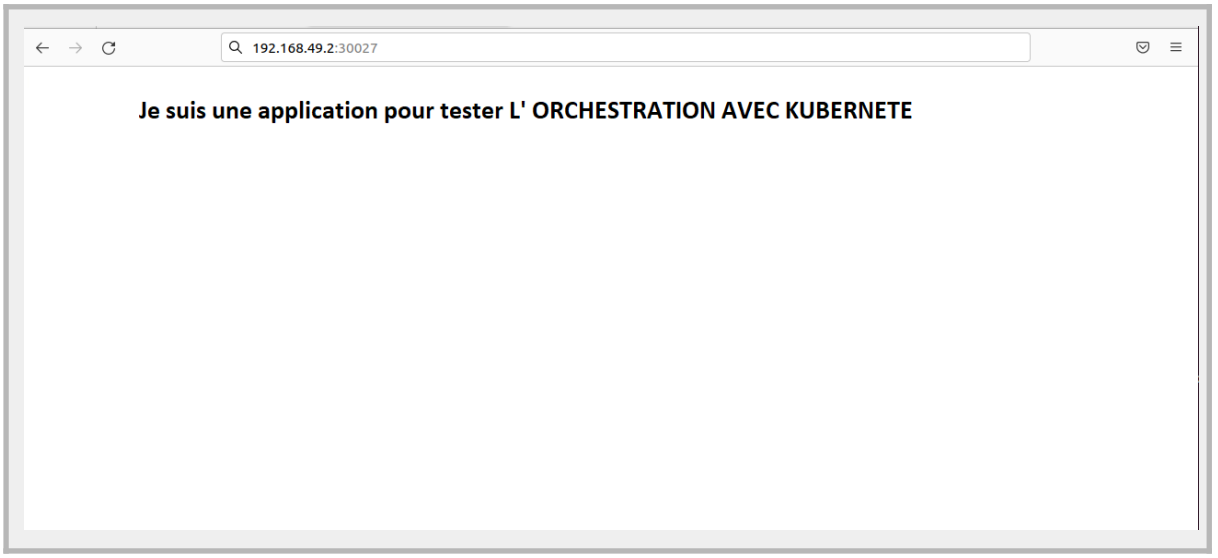
$ kubectl expose deployment/tp-vcl – type =”NodePort” - - port 8081

```
malak@malak-virtual-machine:~$ kubectl expose deployment/tp-vcl --type="NodePort" --port 8081
service/tp-vcl exposed
```

$kubectl describe services/tp-vcl

```
malak@malak-virtual-machine:~$ kubectl describe services/tp-vcl
Name:                     tp-vcl
Namespace:                default
Labels:                   app=FirstConteneur
Annotations:              <none>
Selector:                 app=nginx
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.97.226.9
IPs:                      10.97.226.9
Port:                     <unset>  8081/TCP
TargetPort:               8081/TCP
NodePort:                 <unset>  30027/TCP
Endpoints:                10.244.0.5:8081,10.244.1.6:8081
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
malak@malak-virtual-machine:~$
```

Maintenant que le service a été configuré pour exposer l'application sur le port 30027, elle est accessible pour les machines en dehors du cluster Minikube. On peut vérifier cela en accédant à l'application via un navigateur en utilisant l'adresse IP de Minikube avec le port 30027. Cela permettra de vérifier que l'application est bien accessible depuis l'extérieur du cluster.



Chapitre 3

Conclusion

En conclusion, ce TP sur la kubernétisation a permis d'acquérir les connaissances de base sur l'utilisation de Kubernetes pour déployer des applications en utilisant Docker.On a appris à créer des conteneurs, des pods et des services, ainsi qu'à utiliser les fichiers de configuration pour définir les ressources nécessaires à l'application. Le déploiement de l'application simple a été un succès et a montré la puissance de Kubernetes pour automatiser le déploiement, la scalabilité et la gestion des erreurs des applications. Ce TP a été un outil pratique pour comprendre les concepts fondamentaux de la kubernétisation.