

Déploiement des Conteneurs avec Kubernetes

Table des matières

Configuration de Kubernetes (K3S)	2
Déploiement sur Kubernetes (K3S)	2
Script	4

Configuration de Kubernetes (K3S)

Modifier le fichier YAML pour le déploiement Kubernetes :

nano goweb-deploy.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mon-deployment
spec:
  replicas: 7
  selector:
    matchLabels:
      app: tp38 # Nom de l'application
  template:
    metadata:
      labels:
        app: tp38 # Nom de l'application
    spec:
      containers:
        - name: tp38 # Nom du conteneur
          image: tbmc93/tp38:v2 # Nom de l'image Docker
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: mon-service
spec:
  selector:
    app: tp38 # Nom de l'application
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
```

Déploiement sur Kubernetes (K3S)

Appliquer le déploiement

sudo kubectl apply -f goweb-deploy.yaml

Vérifier les déploiements :

sudo kubectl get deployments

```
ubuntu22@ubuntu22-virtual-machine:~/mysiteweb$ sudo kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
mon-deployment      7/7     7             7           16m
```

Vérifier les pods

sudo kubectl get pods

```
ubuntu22@ubuntu22-virtual-machine:~/mysiteweb$ sudo kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
mon-deployment-7d5cf77847-2l97z    1/1     Running   0           11m
mon-deployment-7d5cf77847-b7d42    1/1     Running   0           11m
mon-deployment-7d5cf77847-dj5lf     1/1     Running   0           11m
mon-deployment-7d5cf77847-hz5h2     1/1     Running   0           11m
mon-deployment-7d5cf77847-sxsv6     1/1     Running   0           11m
mon-deployment-7d5cf77847-tb2kk     1/1     Running   0           11m
mon-deployment-7d5cf77847-zbwdw     1/1     Running   0           11m
```

Vérifier les services

sudo kubectl get services

```
ubuntu22@ubuntu22-virtual-machine:~/mysiteweb$ sudo kubectl get services
NAME            TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
kubernetes      ClusterIP     10.43.0.1    <none>        443/TCP          64m
mon-service     LoadBalancer 10.43.183.74 <pending>     80:30584/TCP     42m
```

Tester l'accès à l'application

Utilisez l'adresse IP du nœud et le port NodePort pour accéder à votre application :

curl <http://<IP-du-noeud>:30584>

- Dans Kubernetes, lorsque vous utilisez un service de type LoadBalancer ou NodePort, le système attribue automatiquement un port dans la plage des NodePorts (par défaut 30000-32767) si vous ne spécifiez pas un port particulier.
- Ce port est choisi de manière aléatoire dans cette plage à chaque fois que le service est créé ou recréé.



L'épopée extraordinaire du RC Lens en Ligue des champions 1999

Introduction

La saison 1998-1999 restera gravée à jamais dans la mémoire des supporters du Racing Club de Lens. Après avoir remporté le titre de champion de France pour la première fois de son histoire, le club artésien a réalisé un parcours exceptionnel en Ligue des champions, atteignant les demi-finales de la compétition.

Cette épopée extraordinaire a été marquée par des moments inoubliables, comme la victoire 3-1 contre le Bayern Munich en phase de groupes, ou encore l'élimination du Manchester United en quarts de finale.

Le RC Lens a finalement échoué aux portes de la finale, battu par le FC Barcelone, mais son parcours a marqué l'histoire du football français et européen.

Chapitre 1 : Une phase de groupes historique et des joueurs légendaires

Le RC Lens est tombé dans un groupe relevé pour sa première participation à la Ligue des champions, avec Arsenal, l'Olympiakos et le Dynamo Kiev.

Mais les Sang et Or ont réalisé un parcours exceptionnel, ne concédant qu'une seule défaite et terminant à la première place de leur groupe.

Parmi les moments forts de cette phase de groupes, on peut citer la victoire 3-1 contre le Bayern Munich, champion d'Allemagne en titre, et le match nul 1-1 arraché à Highbury contre Arsenal.

Les héros de cette épopée

- **Tony Vairelles** : L'attaquant lensois était la star de l'équipe. Auteur de 7 buts en Ligue des champions, il a été l'un des grands artisans du parcours du RC Lens.
- **Éric Sikora** : Le milieu de terrain était le poumon de l'équipe. Infatigable et combatif, il a récupéré de nombreux ballons et a souvent initié les attaques lensoises.
- **Franck Haise** : Le défenseur central était le patron de la défense lensoise. Solide et serein, il a repoussé de nombreuses attaques adverses.
- **Jocelyn Gourvennec** : L'ailier était l'un des joueurs les plus techniques de l'équipe. Rapide et dribbleur, il a souvent pris la défense adverse à défaut.
- ****Et bien d'autres encore !**** Cette équipe du RC Lens 1999 était composée de joueurs talentueux, solidaires et combatifs. C'est grâce à eux que le club a réalisé cette épopée extraordinaire.

Chapitre 2 : Le parcours de Lens dans les compétitions européennes

Le RC Lens a participé à deux compétitions européennes avant sa grande épopée en Ligue des champions 1999 : la Coupe UEFA en 1995-1996 et la Coupe des Coupes en 1997-1998.

En Coupe UEFA 1995-1996, le RC Lens a atteint les huitièmes de finale, où il a été éliminé par le FC Kaiserslautern.

En Coupe des Coupes 1997-1998, le RC Lens a atteint les quarts de finale, où il a été éliminé par le SS Lazio.

Script

Rendre le Script Exécutable

Assurez-vous que votre script est exécutable.

```
chmod +x myscript.sh
```

Exécuter le script

```
./deploy-script.sh
```

Menu

```
ubuntu22@ubuntu22-virtual-machine:~/mysiteweb$ sudo ./myscript.sh
1) Build Docker image
2) Tag and push Docker image to Docker Hub
3) Open Kubernetes deployment
4) Apply Kubernetes deployment
5) Check deployment status
6) Get services and pods
7) Exit
Choose an option: _
```

Option 1 & 2

```
Choose an option: 1
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
             Install the buildx component to build images with BuildKit:
             https://docs.docker.com/go/buildx/

Sending build context to Docker daemon 10.75kB
Step 1/2 : FROM customweb
----> a72860cb95fd
Step 2/2 : COPY index.html /usr/share/nginx/html
----> Using cache
----> b13e73db29c9
Successfully built b13e73db29c9
Successfully tagged tbmc93/tp38:v2
1) Build Docker image
2) Tag and push Docker image to Docker Hub
3) Open Kubernetes deployment
4) Apply Kubernetes deployment
5) Check deployment status
6) Get services and pods
7) Exit
Choose an option: 2
The push refers to repository [docker.io/tbmc93/tp38]
cc5ae9b936b8: Layer already exists
60e72fbb314e: Layer already exists
599e8de62018: Layer already exists
09581b9299a2: Layer already exists
a39383416a22: Layer already exists
a6355e7844d5: Layer already exists
fcfa12460e7d: Layer already exists
e0781bc8667f: Layer already exists
v2: digest: sha256:30438e6e80c70cba5f44ea252070d66ebffbcfdec0242f3ca5a0e238fd21cffa size: 1986
```

Option 4, 5, 6 & 7

```
Choose an option: 4
deployment.apps/mon-deployment unchanged
service/mon-service unchanged
1) Build Docker image
2) Tag and push Docker image to Docker Hub
3) Open Kubernetes deployment
4) Apply Kubernetes deployment
5) Check deployment status
6) Get services and pods
7) Exit
Choose an option: 5
deployment "mon-deployment" successfully rolled out
1) Build Docker image
2) Tag and push Docker image to Docker Hub
3) Open Kubernetes deployment
4) Apply Kubernetes deployment
5) Check deployment status
6) Get services and pods
7) Exit
Choose an option: 6
NAME                 TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
kubernetes            ClusterIP           10.43.0.1       <none>           443/TCP          19h
mon-service           LoadBalancer       10.43.183.74    <pending>        80:30584/TCP     19h
```

NAME	READY	STATUS	RESTARTS	AGE
mon-deployment-7d5cf77847-2197z	1/1	Running	1 (23m ago)	18h
mon-deployment-7d5cf77847-b7d42	1/1	Running	1 (23m ago)	18h
mon-deployment-7d5cf77847-dj51f	1/1	Running	1 (23m ago)	18h
mon-deployment-7d5cf77847-hz5h2	1/1	Running	1 (23m ago)	18h
mon-deployment-7d5cf77847-sxsv6	1/1	Running	1 (23m ago)	18h
mon-deployment-7d5cf77847-tb2kk	1/1	Running	1 (23m ago)	18h
mon-deployment-7d5cf77847-zbwdu	1/1	Running	1 (23m ago)	18h

```
1) Build Docker image
2) Tag and push Docker image to Docker Hub
3) Open Kubernetes deployment
4) Apply Kubernetes deployment
5) Check deployment status
6) Get services and pods
7) Exit
Choose an option: 7
Exit
```