

BIARNEIX Paul

BOIS Axel

SOLEILHAVOUP Mathis

DIALLO Salimou

Documentation d'installation

Micro-services

Déploiement d'une application



Table des matières

Introduction..... 3

Prérequis et architecture..... 3

Phase de publication de l’image 5

Déploiement sur Windows – étapes spécifiques 6

Déploiement sur Linux – étapes spécifiques..... 7

Déploiement – étapes générales..... 8




Introduction

Cette documentation d'information comporte d'une part les spécificités concernant l'installation et le déploiement d'une application web sur les environnements Windows et Linux. La version de Windows est la 11 Famille (24H2) et la distribution de Linux est la Debian 12 Bookworm. Il y aura donc des étapes à suivre communes aux deux OS, mais également des étapes divergentes. Dans le cadre de notre installation, le pseudonyme utilisé pour le profil DockerHub est mrfreeze36, tandis que l'image se prénomme droite-gauche, disponible à ce lien : <https://hub.docker.com/r/mrfreeze36/droite-gauche>

Prérequis et architecture

Le projet présenté ici, qui va nous servir d'exemple tout le long de notre guide est un projet simple utilisant le framework NodeJS en guise d'environnement d'exécution. La stack utilisée comporte une base de données mySQL et un serveur Express à minima.

Pour assurer un bon suivi des instructions, il y a des prérequis en termes d'outils, et ce pour chaque environnement :

 docker	Docker installé, à minima le daemon et le client, et Docker Desktop si l'hyperviseur le permet
 kubernetes	Kubernetes activé via Minikube (Linux) ou Docker Desktop (Windows)
 kubectl	L'utilitaire kubectl d'installé

De plus, le code source comporte une architecture particulière, que voici :

```
gauche-droite-app/  
├─ backend/  
│   ├── Dockerfile  
│   ├── server.js  
│   ├── package.json  
│   ├── db_init.sql  
│   └─ public/  
│       ├── index.html  
│       └─ script.js  
└─ k8s/  
    ├── secret.yaml  
    ├── mysql-initdb-configmap.yaml  
    ├── mysql-pvc.yaml  
    ├── mysql-deployment.yaml  
    ├── mysql-service.yaml  
    ├── phpmyadmin-deployment.yaml  
    ├── phpmyadmin-service.yaml  
    ├── nodeapp-deployment.yaml  
    └─ nodeapp-service.yaml
```

On y retrouve la partie « application », avec les fichiers d’initialisation de la base de données, de définition du serveur, et de la liste des dépendances Node. On y retrouve également le Dockerfile qui va nous permettre de construire notre image avec l’application, ainsi que la partie frontend.

Le dossier k8s recueille les fichiers Kubernetes de déploiement en format .yaml qui nous seront utiles.

Phase de publication de l'image

Pour retrouver notre image plus facilement en ayant juste à la pull et ne pas la construire à chaque déploiement, nous allons la publier sur le Docker Hub. On se dirige dans le dépôt, dans le dossier où est situé le Dockerfile :

```
cd gauche-droite-app/backend
```

On construit ensuite l'image à partir du Dockerfile de l'application présent dans le dossier :

```
docker build -t <nom utilisateur>/<nom à donner à l'image> .
```

On effectue la commande de vérification des images docker images :

```
root@vbox: /home/pol/Documents/droite-gauche-main/backend# docker images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
mrfreeze36/droite-gauche  latest      b02668fece04     2 hours ago     1.1GB
```

On vient ensuite se connecter à Docker en console, et publier l'image avec ces commandes :

```
docker login
docker push <nom utilisateur>/<nom donné à l'image>
```

Il ne faut néanmoins pas avoir oublié de créer un répertoire au même nom sur Docker Hub, en visibilité publique :

Repositories

All repositories within the mrfreeze36 namespace.

All content




Create a repository

[Repositories](#) / [droite-gauche](#) / [General](#)

Using 1 of 1 private repositories. [Get more](#)

mrfreeze36/droite-gauche 

Last pushed about 3 hours ago • Repository size: 380.5 MB

Vrai cette fois ci 

Add a category  

Docker commands

[Public view](#)

To push a new tag to this repository:

```
docker push mrfreeze36/droite-gauc  
he:tagname
```

General

Tags

Image Management **BETA**

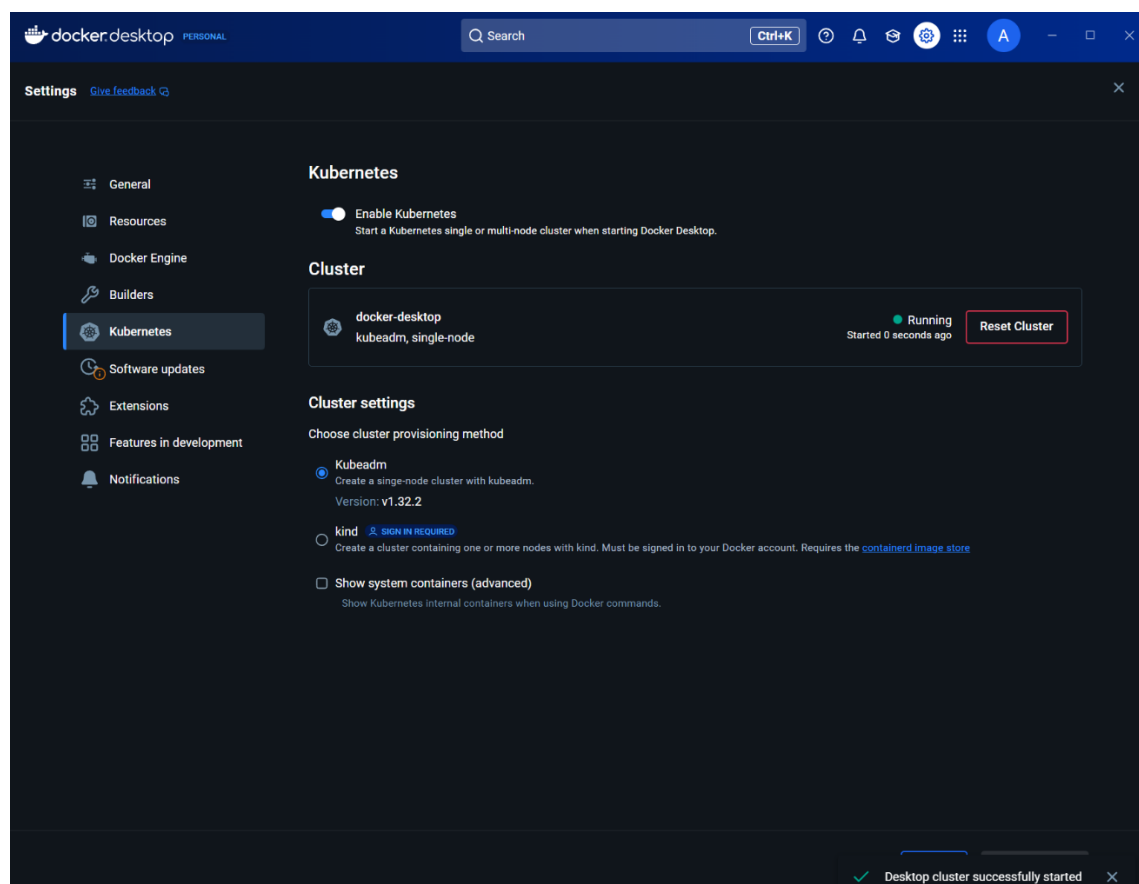
Collaborators

Webhooks

Settings 

Déploiement sur Windows – étapes spécifiques

Allons démarrer Docker Desktop. Une fois le daemon activé, on se dirige dans la section Settings, puis Kubernetes. On vient ainsi cocher la case « Enable Kubernetes », puis on presse le bouton « Apply & Restart ». Après un certain temps d’attente, nous devons retrouver un affichage comme celui-ci :



Les étapes à suivre ensuite sont communes aux autres environnements.

Déploiement sur Linux – étapes spécifiques

On utilisera l'utilitaire minikube pour intégrer nos fichiers kubernetes. On lance minikube :

```
minikube start
```

```
pol@vbox:~/Documents/droite-gauche-main/backend$ minikube start
😄 minikube v1.36.0 on Debian 12.11 (vbox/amd64)
🌟 Automatically selected the docker driver
🔧 Using Docker driver with root privileges
👉 Starting "minikube" primary control-plane node in "minikube" cluster
🚢 Pulling base image v0.0.47 ...
🔥 Creating docker container (CPUs=2, Memory=2200MB) ...
❗ Failing to connect to https://registry.k8s.io/ from inside the minikube container
💡 To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
🔧 Preparing Kubernetes v1.33.1 on Docker 28.1.1 ...
  ▪ Generating certificates and keys ...
  ▪ Booting up control plane ...
  ▪ Configuring RBAC rules ...
🔧 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
```

Par la suite, on veut faire pointer Minikube sur l'environnement Docker :

```
eval $(minikube docker-env)
```

Les étapes à suivre ensuite sont communes aux autres environnements.

Déploiement – étapes générales

On comment par pull l'image que l'on a push sur Docker Hub précédemment

```
docker pull <nom_utilisateur>/<nom_image>:<tag>
```

```
pol@vbox:~/Documents/droite-gauche-main/k8s$ docker pull mrfreeze36/droite-gauche:latest
latest: Pulling from mrfreeze36/droite-gauche
3e6b9d1a9511: Pull complete
37927ed901b1: Pull complete
79b2f47ad444: Pull complete
e23f099911d6: Pull complete
cda7f44f2bdd: Pull complete
c6b30c3f1696: Pull complete
3697be50c98b: Pull complete
461077a72fb7: Pull complete
80a29ebc75d7: Pull complete
36955cdb1657: Pull complete
90af7d43eeec: Pull complete
8d121ca0a53b: Pull complete
Digest: sha256:67dca63d9db44b2d2ec4421a5f86f0c47dddeb4588015600dbbd8bb633162275
Status: Downloaded newer image for mrfreeze36/droite-gauche:latest
docker.io/mrfreeze36/droite-gauche:latest
```

Avant de passer à la phase d'application de nos fichiers kubernetes au sein du cluster, il faut vérifier nos fichiers, et plus particulièrement le fichier de déploiement pour notre application (au sein de notre architecture, nodeapp-deployment.yml). On vient regarder en effet si le fichier contient le nom de la bonne image, en se déplaçant au préalable dans le dossier contenant les fichier YAML de configuration :

```
cd ../k8s
```

```
template:
  metadata:
    labels:
      app: nodeapp
  spec:
    containers:
      - name: nodeapp
        image: mrfreeze36/droite-gauche:latest
        imagePullPolicy: IfNotPresent
```


La ligne « imagePullPolicy » est également importante, car définit si, à chaque déploiement, kubernetes doit pull l'image référencée systématiquement, et ce en local ou non. On définit ce paramètre à IfNotPresent.

Une fois vérifié, on déploie l'ensemble de nos fichiers YAML de configuration avec cette commande :

```
kubectl apply -f .
```

```
pol@vbox:~/Documents/droite-gauche-main/k8s$ kubectl apply -f .
deployment.apps/mysql created
configmap/mysql-initdb created
persistentvolumeclaim/mysql-pvc created
service/mysql created
deployment.apps/nodeapp created
service/nodeapp created
deployment.apps/phpmyadmin created
service/phpmyadmin created
secret/mysql-secret created
```

On vérifie alors si les pods sont bien en cours d'exécution, sans erreur particulière :

Terminal

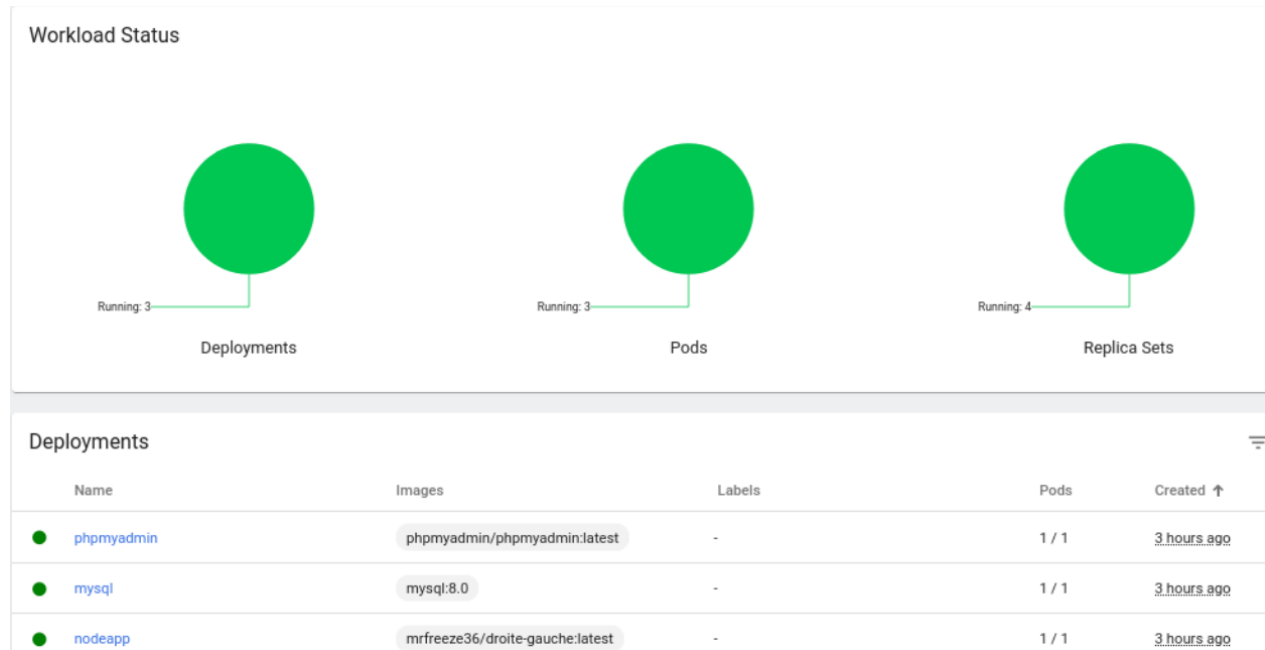
```
kubectl apply -f .
```

```
pol@vbox:~/Documents/droite-gauche-main/k8s$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
mysql-5d76c845f6-lpf69	1/1	Running	1 (2m30s ago)	19m
nodeapp-5cb7d5d6d8-p7l6j	1/1	Running	3 (73s ago)	10m
phpmyadmin-7ff5b849ff-p6nsc	1/1	Running	1 (2m31s ago)	19m

Dashboard minikube

```
minikube dashboard
```



Nos pods et déploiements ont correctement été lancés. Nous pouvons désormais accéder à la solution en effectuant la commande :

```
minikube service nodeapp
```

192.168.49.2:30002

Vote : Gauche ou Droite ?

Ajoute une chose...

Ajouter

Liste des choses

• **Iphone** | Gauche: 0 | Droite: 2

👉 Gauche

👉 Droite

• **Valorant** | Gauche: 2 | Droite: 0

👉 Gauche

👉 Droite

On peut également accéder au panel PHPMyAdmin, pour administrer la base de données :

```
minikube service phpmyadmin
```

