

**Institut Supérieur d'Informatique et de Techniques**  
**de Communication H-Sousse**

# **Administration et Sécurité des Réseaux**

## **TP 2**

**Introduction à Kubernetes**

**Réalisé Par**

**ELagas Amel**

**Classe : 2DNI G1**

**Enseignant : M. Salah Gontara**

**A.U :2020-2021**

# **Plan**

## **Introduction**

### **A. Partie théorique**

- 1. Qu'est-ce-que Kubernetes ?**
- 2. Pourquoi nous avons besoin de Kubernetes et que peut-il faire ?**
- 3. Comment Kubernetes est-il une plate-forme ?**
- 4. Pourquoi les conteneurs ?**
- 5. Que se passe-t-il au sein d'un nœud Kubernetes ?**

### **B. Partie pratique**

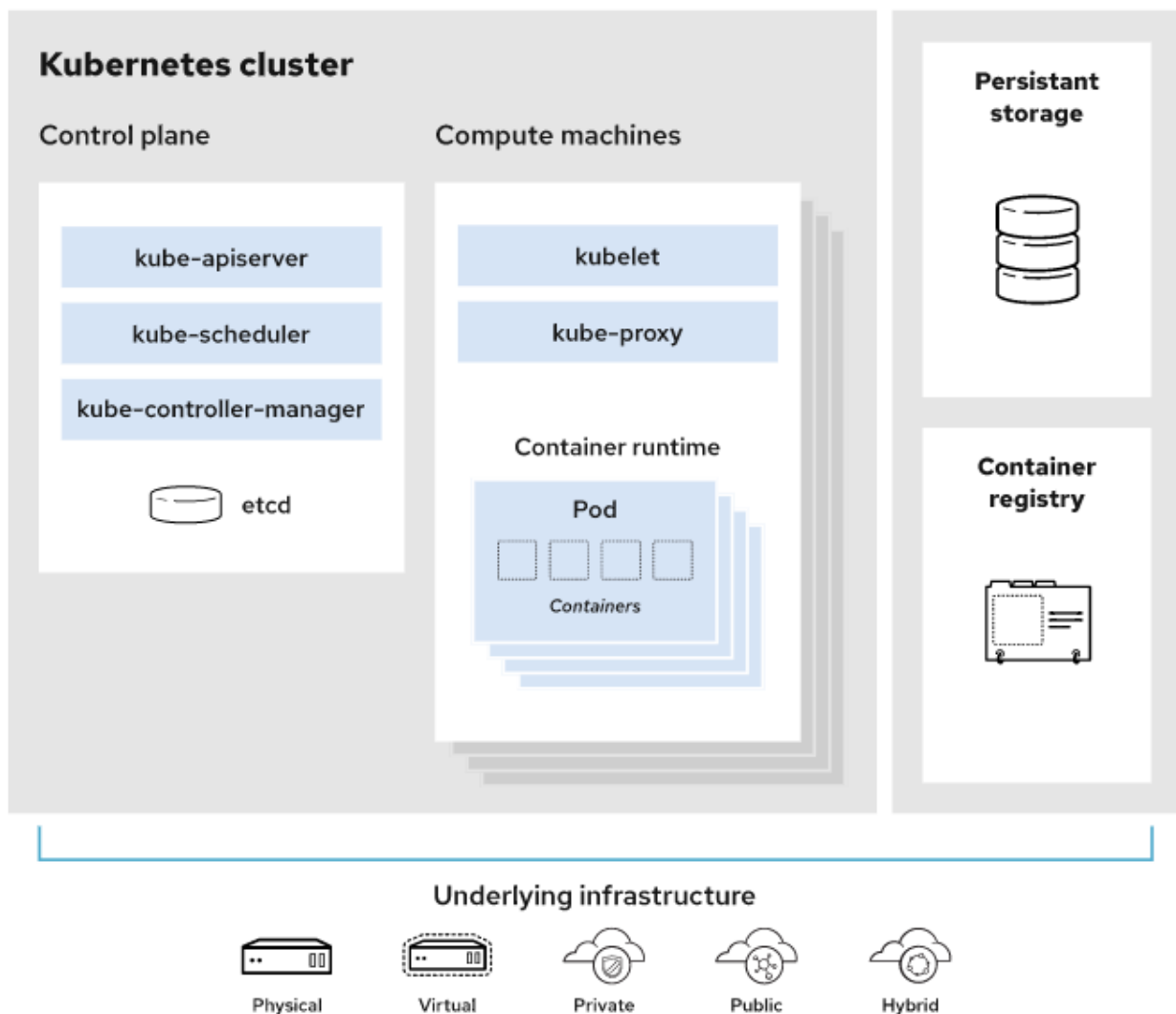
- 1. Les commandes de base**
- 2. Les déploiements simples**
- 3. Les déploiements configurés**

## **Conclusion**




## Introduction

Le terme « cluster » désigne un déploiement fonctionnel de Kubernetes. Un cluster Kubernetes comprend deux principaux composants : le plan de contrôle et les machines de calcul ou nœuds. Chaque nœud est son propre environnement Linux®. Il peut s'agir d'une machine physique ou virtuelle. Chaque nœud exécute des pods, constitués de conteneurs.

Le schéma suivant représente les liens entre les différents composants d'un cluster Kubernetes :



## **OBJECTIF(S):**

-  Comprendre le fonctionnement de base de Kubernetes et Minikube.
-  Interagir efficacement (en mode debug) avec les pods.
-  Créer des déploiements avec et sans fichiers de configuration YAML.

## **OUTILS UTILISÉS:**

Kubernetes (Minikube), Nginx, Mongo

## A. Partie théorique

### 1. Qu'est-ce que Kubernetes ?

Kubernetes est une plate-forme open-source extensible et portable pour la gestion de charges de travail (workloads) et de services conteneurisés. Elle favorise à la fois l'écriture de configuration déclarative (declarative configuration) et l'automatisation. C'est un large écosystème en rapide expansion. Les services, le support et les outils Kubernetes sont largement disponibles.

Google a rendu open-source le projet Kubernetes en 2014. Le développement de Kubernetes est basé sur une décennie et demie d'expérience de Google avec la gestion de la charge et de la mise à l'échelle (scale) en production, associée aux meilleures idées et pratiques de la communauté.

### 2. Pourquoi nous avons besoin de Kubernetes et que peut-il faire ?

Kubernetes a un certain nombre de fonctionnalités. Il peut être considéré comme :

- une plate-forme de conteneurs
- une plate-forme de microservices
- une plate-forme cloud portable et beaucoup plus.

Kubernetes fournit un environnement de gestion focalisé sur le conteneur (container-centric). Il **orchestre** les ressources machines (computing), la mise en réseau et l'infrastructure de stockage sur les workloads des utilisateurs. Cela permet de se rapprocher de la simplicité des Platform as a Service (PaaS) avec la flexibilité des solutions d'Infrastructure as a Service (IaaS), tout en gardant de la portabilité entre les différents fournisseurs d'infrastructures (providers).

### 3. Comment Kubernetes est-il une plate-forme ?

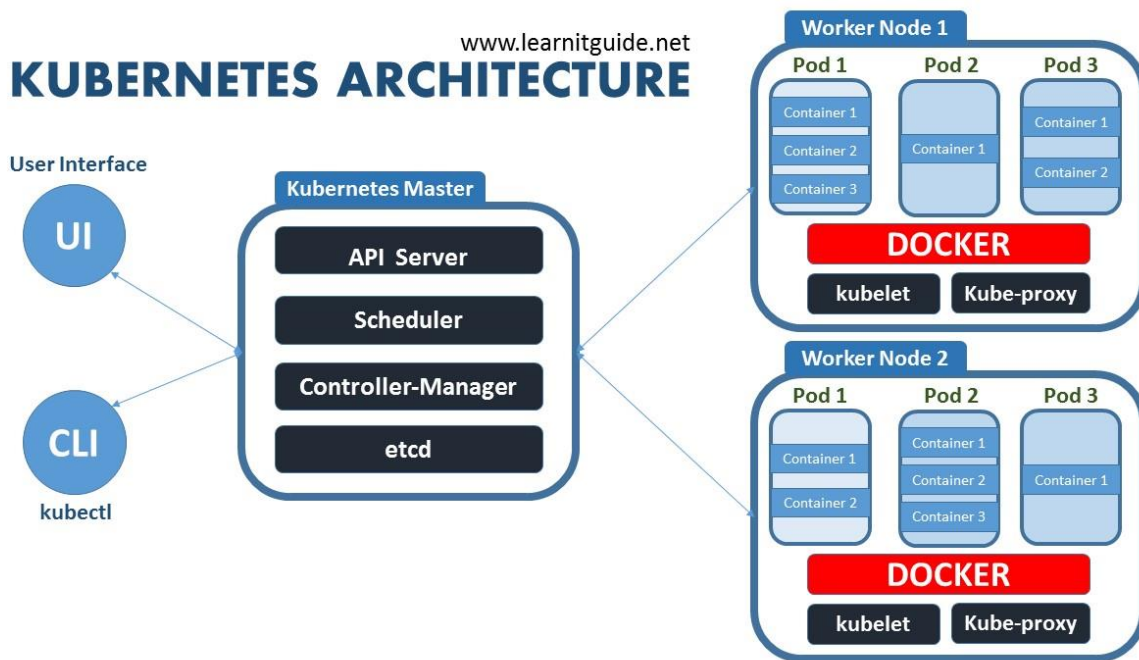
Même si Kubernetes fournit de nombreuses fonctionnalités, il existe toujours de nouveaux scénarios qui bénéficieraient de fonctionnalités complémentaires. Ces workflows spécifiques à une application permettent d'accélérer la vitesse de développement. Si l'orchestration fournie de base est acceptable pour commencer, il est souvent nécessaire d'avoir une automatisation robuste lorsque l'on doit la faire

évoluer. C'est pourquoi Kubernetes a également été conçu pour servir de plate-forme et favoriser la construction d'un écosystème de composants et d'outils facilitant le déploiement, la mise à l'échelle et la gestion des applications.

Les Labels permettent aux utilisateurs d'organiser leurs ressources comme ils/elles le souhaitent. Les Annotations autorisent les utilisateurs à définir des informations personnalisées sur les ressources pour faciliter leurs workflows et fournissent un moyen simple aux outils de gérer la vérification d'un état (checkpoint state).

De plus, le plan de contrôle Kubernetes (control plane) est construit sur les mêmes APIs que celles accessibles aux développeurs et utilisateurs. Les utilisateurs peuvent écrire leurs propres contrôleurs (controllers), tels que les ordonnanceurs (schedulers), avec leurs propres APIs qui peuvent être utilisés par un outil en ligne de commande.

Ce choix de conception a permis de construire un ensemble d'autres systèmes par-dessus Kubernetes.



#### 4. Pourquoi les conteneurs ?

- **Création et déploiement agile d'applications** : Augmente la simplicité et l'efficacité de la création d'images par rapport à l'utilisation d'images de VM.

- **Développement, intégration et déploiement Continu** : Fournit un processus pour construire et déployer fréquemment et de façon fiable avec la capacité de faire des rollbacks rapides et simples (grâce à l'immuabilité de l'image).
- **Séparation des besoins entre Dev et Ops** : Création d'images applicatives au moment du build plutôt qu'au déploiement, tout en séparant l'application de l'infrastructure.
- **Observabilité** Informations venant non seulement du système d'exploitation sous-jacent mais aussi des signaux propres de l'application.
- **Consistance entre les environnements de développement, tests et production** : Fonctionne de la même manière que ce soit sur un poste local que chez un fournisseur d'hébergement / dans le Cloud.
- **Portabilité entre Cloud et distribution système** : Fonctionne sur Ubuntu, RHEL, CoreOS, on-prem, Google Kubernetes Engine, et n'importe où.
- **Gestion centrée Application** : Bascule le niveau d'abstraction d'une virtualisation hardware liée à l'OS à une logique de ressources orientée application.
- **Micro-services faiblement couplés, distribués, élastiques** : Les applications sont séparées en petits morceaux indépendants et peuvent être déployées et gérées dynamiquement pas une stack monolithique dans une seule machine à tout faire.
- **Isolation des ressources** : Performances de l'application prédictibles.
- **Utilisation des ressources** : Haute efficacité et densité.

## 5. Que se passe-t-il au sein d'un nœud Kubernetes ?

### Nœuds

Un cluster Kubernetes requiert au moins un nœud de calcul, mais en général il en contient un grand nombre. Les pods sont planifiés et orchestrés pour être exécutés sur des nœuds. Vous avez besoin de faire évoluer la capacité de votre cluster ? Ajoutez des nœuds.

### Pods

Le pod est l'unité la plus petite et la plus simple dans le modèle d'objets de Kubernetes. Il représente une instance unique d'une application. Chaque pod est constitué d'un conteneur ou d'une série de conteneurs étroitement couplés, ainsi que des options

permettant de contrôler l'exécution de ces conteneurs. Il est possible de connecter les pods à un système de stockage persistant afin d'exécuter des applications avec état.

### **Moteur d'exécution de conteneurs**

Chaque nœud de calcul dispose d'un moteur qui permet d'exécuter les conteneurs. Docker en est un exemple, mais Kubernetes prend en charge d'autres environnements conformes aux normes OCI (Open Container Initiative), tels que rkt et CRI-O.

### **kubelet**

Chaque nœud de calcul contient un kubelet, une petite application qui communique avec le plan de contrôle. Le kubelet s'assure que les conteneurs sont exécutés dans un pod. Lorsque le plan de contrôle envoie une requête vers un nœud, le kubelet exécute l'action.

### **kube-proxy**

Chaque nœud de calcul contient également un proxy réseau appelé « kube-proxy » qui facilite la mise en œuvre des services de mise en réseau de Kubernetes. Le composant kube-proxy gère les communications réseau dans et en dehors du cluster. Il utilise la couche de filtrage de paquets du système d'exploitation si elle est disponible, sinon il transmet le trafic lui-même.

## **B. Partie pratique**

### **Partie 1 : Les commandes de base**

1. Démarrez le cluster Minikube :



```

ubuntu@ubuntuuserver:~$ sudo minikube start --driver=none
[sudo] password for ubuntu:
* minikube v1.17.1 on Ubuntu 20.04 (vbox/amd64)
* Using the none driver based on existing profile

X The requested memory allocation of 1987MiB does not leave room for system over
head (total system memory: 1987MiB). You may face stability issues.
* Suggestion: Start minikube with less memory allocated: 'minikube start --memor
y=1987mb'

* Starting control plane node minikube in cluster minikube
* Restarting existing none bare metal machine for "minikube" ...
* OS release is Ubuntu 20.04.2 LTS
! This bare metal machine is having trouble accessing https://k8s.gcr.io
* To pull new external images, you may need to configure a proxy: https://miniku
be.sigs.k8s.io/docs/reference/networking/proxy/
* Preparing Kubernetes v1.20.2 on Docker 20.10.3 ...
- kubelet.resolv-conf=/run/systemd/resolve/resolv.conf
* minikube 1.19.0 is available! Download it: https://github.com/kubernetes/minik
ube/releases/tag/v1.19.0
* To disable this notice, run: 'minikube config set WantUpdateNotification false
'

```

```

* Done! kubectl is now configured to use "minikube" cluster and "default" namesp
ace by default
ubuntu@ubuntuuserver:~$ █

```

## 2. Affichez la version de Minikube :

```

ubuntu@ubuntuuserver:~$ sudo kubectl version
[sudo] password for ubuntu:
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.2", GitCom
mit:"faecb196815e248d3ecfb03c680a4507229c2a56", GitTreeState:"clean", BuildDate:
"2021-01-13T13:28:09Z", GoVersion:"go1.15.5", Compiler:"gc", Platform:"linux/amd
64"}
Server Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.2", GitCom
mit:"faecb196815e248d3ecfb03c680a4507229c2a56", GitTreeState:"clean", BuildDate:
"2021-01-13T13:20:00Z", GoVersion:"go1.15.5", Compiler:"gc", Platform:"linux/amd
64"}

```

## 3. Listez les nœuds de votre cluster :

```

ubuntu@ubuntuuserver:~$ sudo kubectl get nodes
NAME                STATUS    ROLES                  AGE     VERSION
ubuntuuserver       Ready     control-plane,master   60d     v1.20.2
ubuntu@ubuntuuserver:~$ █

```

## 4. Affichez le status de votre cluster :

```

ubuntu@ubuntu-server:~$ sudo kubectl get nodes
NAME          STATUS    ROLES          AGE   VERSION
ubuntu-server Ready    control-plane,master 60d   v1.20.2
ubuntu@ubuntu-server:~$ sudo minikube status
minikube
type: Control Plane
host: InsufficientStorage
kubelet: Running
apiserver: Running
kubeconfig: Configured
timeToStop: Nonexistent
ubuntu@ubuntu-server:~$ █

```

5. Effacez votre cluster et redémarrez-le en mode debugging :

```
sudo minikube delete
```

```

ubuntu@ubuntu-server:~$ sudo kubectl get nodes
The connection to the server localhost:8080 was refused - did you specify the right host or port?
ubuntu@ubuntu-server:~$ sudo minikube delete
* Uninstalling Kubernetes v1.20.2 using kubeadm ...
* Deleting "minikube" in none ...
* Removed all traces of the "minikube" cluster.
ubuntu@ubuntu-server:~$ █

```

```
sudo minikube start --driver=none --alsologtostderr
```

```
ubuntu@ubuntuuser:~$ sudo minikube start --driver=none --alsologtostderr
[sudo] password for ubuntu:
I0417 21:46:46.631397 5828 out.go:229] Setting OutFile to fd 1 ...
I0417 21:46:46.633524 5828 out.go:276] TERM=xterm,COLORTERM=, which probably
does not support color
I0417 21:46:46.633648 5828 out.go:242] Setting ErrFile to fd 2...
I0417 21:46:46.633734 5828 out.go:276] TERM=xterm,COLORTERM=, which probably
does not support color
I0417 21:46:46.634134 5828 root.go:291] Updating PATH: /root/.minikube/bin
I0417 21:46:46.634589 5828 out.go:236] Setting JSON to false
I0417 21:46:46.635472 5828 start.go:106] hostinfo: {"hostname":"ubuntuuser",
,"uptime":1836,"bootTime":1618694171,"procs":114,"os":"linux","platform":"ubuntu",
,"platformFamily":"debian","platformVersion":"20.04","kernelVersion":"5.4.0-65-
generic","kernelArch":"x86_64","virtualizationSystem":"vbox","virtualizationRole":
"guest","hostId":"ab664978-487a-304c-b04e-3998d0f3d51d"}
I0417 21:46:46.635599 5828 start.go:116] virtualization: vbox guest
I0417 21:46:46.668748 5828 out.go:119] * minikube v1.17.1 on Ubuntu 20.04 (vb
ox/amd64)
```

### sudo minikube status

```
ubuntu@ubuntuuser:~$ sudo minikube status
* Profile "minikube" not found. Run "minikube profile list" to view all profiles.
- To start a cluster, run: "minikube start"
ubuntu@ubuntuuser:~$
```

## Partie 2 : Les déploiements simples

1. Listez les images importées avec la commande :

```
ubuntu@ubuntuuser:~$ sudo kubectl get pods
No resources found in default namespace.
ubuntu@ubuntuuser:~$
```

2. Listez les services de votre cluster :

```
ubuntu@ubuntuuser:~$ sudo kubectl get services
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes    ClusterIP     10.96.0.1     <none>         443/TCP    5m15s
ubuntu@ubuntuuser:~$
```

### 3. Créez un déploiement simple de Nginx :

```
ubuntu@ubuntu-server:~$ sudo kubectl create deployment nginx-depl --image=nginx
deployment.apps/nginx-depl created
```

### 4. Confirmez la création de ce déploiement :

```
ubuntu@ubuntu-server:~$ sudo kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
nginx-depl    0/1     1             0           14s
```

### 5. Affichez les répliquions de vos pods déployés, s'ils existent :

```
ubuntu@ubuntu-server:~$ sudo kubectl get replicaset
NAME                                DESIRED   CURRENT   READY   AGE
nginx-depl-5c8bf76b5b              1         1         0       43s
```

### 6. Créez un déploiement simple de MongoDB

```
ubuntu@ubuntu-server:~$ sudo kubectl create deployment mongo-depl --image=mongo
deployment.apps/mongo-depl created
```

### 7. Affichez les logs de vos pods déployés :

#### ▪ Les noms des pods sont dégagés :

```
ubuntu@ubuntu-server:~$ sudo kubectl get pods
NAME                                READY   STATUS             RESTARTS   AGE
mongo-depl-5fd6b7d4b4-x2fbh        0/1     ContainerCreating   0           117s
nginx-depl-5c8bf76b5b-kvfg5        0/1     ContainerCreating   0           2m53s
```

#### ▪ Les logs de pods déployés :

```

ubuntu@ubuntuuser:~$ sudo kubectl logs nginx-depl-5c8bf76b5b-kvfg5
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
ubuntu@ubuntuuser:~$

```

8. Accédez à l'un des pods et confirmez le système :

```

ubuntu@ubuntuuser:~$ sudo kubectl exec -it nginx-depl-5c8bf76b5b-kvfg5 -- bin/bash
root@nginx-depl-5c8bf76b5b-kvfg5:/#

```

9. Affichez la description des pods déployés :

- Pod : mongo-depl-5fd6b7d4b4-x2fbh

**sudo kubectl describe pod mongo-depl-5fd6b7d4b4-x2fbh**

```
ubuntu@ubuntuuserver:~$ sudo kubectl describe pod mongo-depl-5fd6b7d4b4-x2fbh
Name:          mongo-depl-5fd6b7d4b4-x2fbh
Namespace:     default
Priority:       0
Node:          ubuntuuserver/192.168.1.103
Start Time:    Tue, 20 Apr 2021 00:00:26 +0000
Labels:        app=mongo-depl
               pod-template-hash=5fd6b7d4b4
Annotations:   <none>
Status:        Pending
IP:            172.17.0.4
IPs:
  IP:          172.17.0.4
Controlled By: ReplicaSet/mongo-depl-5fd6b7d4b4
Containers:
  mongo:
    Container ID:
    Image:        mongo
    Image ID:
    Port:         <none>
    Host Port:    <none>
    State:        Waiting
      Reason:     ErrImagePull
    Ready:        False
    Restart Count: 0
    Environment:  <none>
```

```

Mounts:
  /var/run/secrets/kubernetes.io/serviceaccount from default-token-grcnr (ro
)
Conditions:
  Type            Status
  Initialized      True
  Ready            False
  ContainersReady  False
  PodScheduled     True
Volumes:
  default-token-grcnr:
    Type:          Secret (a volume populated by a Secret)
    SecretName:    default-token-grcnr
    Optional:      false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                  node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type            Reason      Age              From           Message
  ----            -
  Normal          BackOff     53m (x122 over 4h38m)  kubelet        Back-off pulling image "mongo"
  Warning         Failed      33m (x208 over 4h38m)  kubelet        Error: ImagePullBackOff
  Warning         Failed      16m                kubelet        Failed to pull image "mongo"
: rpc error: code = Unknown desc = write /var/lib/docker/tmp/GetImageBlob7762645
96: no space left on device
  Warning         Failed      52s (x25 over 4h38m)  kubelet        Error: ErrImagePull
  Warning         Failed      52s                kubelet        Failed to pull image "mongo"
: rpc error: code = Unknown desc = write /var/lib/docker/tmp/GetImageBlob9358157
58: no space left on device
  Normal          Pulling     37s (x26 over 6h42m)  kubelet        Pulling image "mongo"
ubuntu@ubuntuserver:~$

```

- Pod : nginx-depl-5c8bf7b5b-kvfg5

```
sudo kubectl describe pod nginx-depl-5c8bf7b5b-kvfg5
```

```

ubuntu@ubuntuserver:~$ sudo kubectl describe pod nginx-depl-5c8bf76b5b-kvfg5
Name:          nginx-depl-5c8bf76b5b-kvfg5
Namespace:     default
Priority:       0
Node:          ubuntuuserver/192.168.1.103
Start Time:    Mon, 19 Apr 2021 23:59:30 +0000
Labels:        app=nginx-depl
               pod-template-hash=5c8bf76b5b
Annotations:   <none>
Status:        Running
IP:            172.17.0.3
IPs:
  IP:          172.17.0.3
Controlled By: ReplicaSet/nginx-depl-5c8bf76b5b
Containers:
  nginx:
    Container ID:  docker://7f298faf3a23252e04bb40ad06376ccb3d8fe78b59af382559e1390563759156
    Image:         nginx
    Image ID:      docker-pullable://nginx@sha256:75a55d33ecc73c2a242450a9f1cc858499d468f077ea942867e662c247b5e412
    Port:          <none>
    Host Port:     <none>
    State:         Running
      Started:     Tue, 20 Apr 2021 00:34:22 +0000
    Ready:         True
    Restart Count: 0
    Environment:   <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-grcnr (ro)
Conditions:
  Type            Status
  Initialized      True
  Ready            True
  ContainersReady  True
  PodScheduled     True
Volumes:
  default-token-grcnr:
    Type:          Secret (a volume populated by a Secret)

```

## 10.Effacez les déploiements :

```
sudo kubectl delete deployment mongo-depl
```

```

ubuntu@ubuntuserver:~$ sudo kubectl delete deployment mongo-depl
[sudo] password for ubuntu:
deployment.apps "mongo-depl" deleted
ubuntu@ubuntuserver:~$ █

```



```
sudo kubectl delete deployment nginx-depl
```

```
ubuntu@ubuntu-server:~$ sudo kubectl delete deployment nginx-depl  
deployment.apps "nginx-depl" deleted  
ubuntu@ubuntu-server:~$
```

## Partie 3 : Les déploiements configurés

1. Créez le fichier YAML de configuration suivante : vim nginx-deployment.yaml

```
ubuntu@ubuntu-server: ~  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: nginx-deployment  
  labels:  
    app: nginx  
spec:  
  replicas: 2  
  selector:  
    matchLabels:  
      app: nginx  
  template:  
    metadata:  
      labels:  
        app: nginx  
    spec:  
      containers:  
      - name: nginx  
        image: nginx:1.16  
        ports:  
        - containerPort: 8080  
~  
~  
"nginx-deployment.yaml" [noeol][dos] 21L, 384C 1,1
```

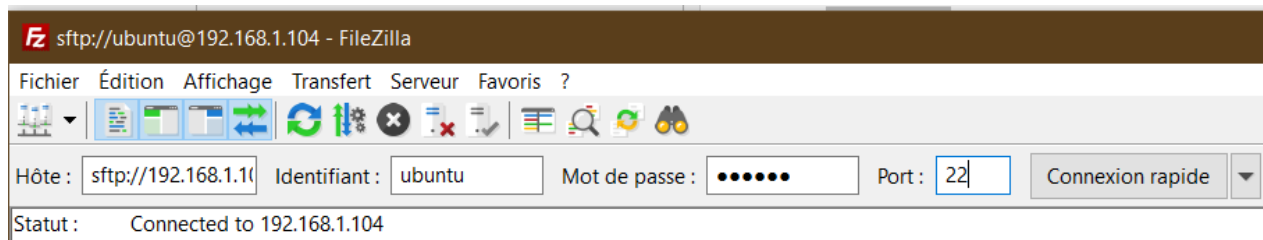
➔ On va utiliser :

**FileZilla** : c'est un client FTP (File Transfert Protocol).

Ce logiciel vous permet de charger ou télécharger les fichiers sur un serveur distant comme par exemple les éléments de votre site web chez vous ou depuis votre hébergeur.

Il possède une interface utilisateur graphique intuitive.

Rapide et fiable, Filezilla est gratuit et multi-plateforme : il fonctionne sur tout système d'exploitation et supporte plusieurs types de connexion : client FTP, FTPS et SFTP.



On utilise l'@ **IP** de notre machine virtuelle comme hôte, l'identifiant **ubuntu**, le mot de passe **ubuntu** et **22** comme numéro de port.

- l'@ **IP** de notre machine virtuelle : **192.168.1.104**

2. Démarrez le déploiement en vous basant sur le fichier de configuration :

```
sudo kubectl apply -f nginx-deployment.yaml
```



```
root@ubuntuserver:/home/ubuntu# sudo kubectl get deployment
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment    0/2     2             0           27m
root@ubuntuserver:/home/ubuntu# _
```

#### 4. Effacez le déploiement configuré :

```
sudo kubectl delete -f nginx-deployment.yaml
```

```
root@ubuntuserver:/home/ubuntu# sudo kubectl delete -f nginx-deployment.yaml
deployment.apps "nginx-deployment" deleted
root@ubuntuserver:/home/ubuntu#
```

### Conclusion

Au cours de ce TP, nous avons commencé par comprendre la notion du Kubernetes, ensuite nous avons bien interagir avec (en mode debug) avec les pods, et nous avons Créé des déploiements avec et sans fichiers de configuration YAML.

Un cluster Kubernetes doit être :

- **Sécurisé** : il doit suivre les bonnes pratiques les plus récentes en matière de sécurité.
- **Facile à utiliser** : quelques commandes simples doivent suffire à son fonctionnement.
- **Adaptable** : il ne doit pas favoriser un fournisseur et doit fournir un fichier de configuration permettant de le personnaliser.

