

## Examen de 4KUBE

### Introduction

L'objectif de ce TP est de pouvoir mettre en œuvre de l'orchestration de conteneurs à savoir le déploiement de deux services : Prestashop et MySQL en spécifiant les volumes où les données seront persistées, les secrets à stocker, leur configmap et enfin un Nodeport afin de pouvoir exposer ces services de façon externe.

Il s'ensuivra ensuite une partie de mise à jour du cluster ainsi que la réalisation d'une pipeline CD.

### Déploiement de services YAML

#### 1/ Création d'un namespace :

Il est possible de créer un namespace à partir de la commande « *kubectl create namespace* » ou bien de tout simplement créer un fichier où on remplira manuellement son fichier YAML.

```
namespace.yaml
1 # Namespace
2 apiVersion: v1
3 kind: Namespace
4 metadata:
5   name: 4kube
6
```

- Le champ `apiVersion` spécifie la version de l'API Kubernetes à utiliser pour cet objet `Namespace`.
- Le champ `kind` indique le type d'objet que vous êtes en train de créer, dans ce cas, un `Namespace`.
- Le champ `metadata` contient des informations supplémentaires sur l'objet, telles que son nom. Dans ce cas, le champ `name` est défini sur "4kube".

2/ Avant d'entamer la création du déploiement d'un service, nous tacherons de définir un fichier YAML où seront persistées les secrets de notre service MySQL :

```
namespace.yaml secrets.yaml
1 apiVersion: v1
2 kind: Secret
3 metadata:
4   name: secret
5   namespace: 4kube
6 type: Opaque
7 data:
8   DATABASE_PASSWORD: MTI1NHJvb3RtZHA=
9   DATABASE_USER: dGVzdf9yb290
10  DATABASE_NAME: bWRwX3Jvb3QxMjU0
11
```

- Le champ `apiVersion` spécifie la version de l'API Kubernetes à utiliser pour cet objet `Secret`.
- Le champ `kind` indique le type d'objet que vous êtes en train de créer, dans ce cas, un `Secret`.

**Réalisé par :** Othmane BENCHEKROUN

**Année :** M.Eng 1

- Le champ metadata contient des informations supplémentaires sur l'objet, telles que son nom et son Namespace. Dans ce cas, le champ name est défini sur "secret" et le champ namespace est défini sur "4kube".
- Le champ type indique le type de Secret à créer. Dans ce cas, il est défini sur "Opaque", ce qui signifie que le Secret ne contient que des données arbitraires et opaques.
- Le champ data contient les données à stocker dans le Secret, en encodage base64. Dans cet exemple, le Secret contient trois clés : DATABASE\_PASSWORD, DATABASE\_USER et DATABASE\_NAME.
- Les valeurs correspondantes sont stockées en encodage base64. Kubernetes n'acceptant uniquement des valeurs sous un tel encodage, nous procédons donc à la commande suivante :

```
[root@master-server 4KUBE-BENCHEKROUN-Othmane-MEng]# echo -n "test_root" | base64
dGVzdF9yb290
[root@master-server 4KUBE-BENCHEKROUN-Othmane-MEng]# echo -n "mdp_root" | base64
bWRwX3Jvb3Q=
[root@master-server 4KUBE-BENCHEKROUN-Othmane-MEng]# echo -n "mdp_root1254" | base64
bWRwX3Jvb3QxMjU0
[root@master-server 4KUBE-BENCHEKROUN-Othmane-MEng]# echo -n "1254rootmdp" | base64
MTI1NHJvb3RtZHA=
```

### 3/ Déploiement du service Prestashop :

Nous définissons un fichier YAML contenant à la fois le déploiement de Prestashop, son volume et son service. Ce fichier se fera en trois temps.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: 4kube
  name: deployment-prestashop
  labels:
    app: prestashop
spec:
  replicas: 1
  selector:
    matchLabels:
      app: prestashop
  template:
    metadata:
      labels:
        app: prestashop
    spec:
      containers:
        - name: prestashop
          image: bitnami/prestashop:1.7
          ports:
            - containerPort: 8080
          env:
            - name: PRESTASHOP_LAST_NAME
              valueFrom:
                configMapKeyRef:
                  name: configmap
                  key: PRESTASHOP_LAST_NAME
            - name: PRESTASHOP_FIRST_NAME
              valueFrom:
                configMapKeyRef:
                  name: configmap
                  key: PRESTASHOP_FIRST_NAME
            - name: PRESTASHOP_DATABASE_USER
              valueFrom:
                secretKeyRef:
                  name: secret
                  key: DATABASE_USER
            - name: PRESTASHOP_DATABASE_NAME
              valueFrom:
                secretKeyRef:
                  name: secret
                  key: DATABASE_NAME
            - name: PRESTASHOP_DATABASE_HOST
              value: service-mysql
            - name: PRESTASHOP_DATABASE_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: secret
                  key: DATABASE_PASSWORD
```

```
key: DATABASE_PASSWORD
volumeMounts:
  - name: prestashop-vol
    mountPath: /bitnami/prestashop
volumes:
  - name: prestashop-vol
    persistentVolumeClaim:
      claimName: prestapvc
```

- Le champ `apiVersion` spécifie la version de l'API Kubernetes à utiliser pour cet objet Deployment.
- Le champ `kind` indique le type d'objet que vous êtes en train de créer, dans ce cas, un Deployment.
- Le champ `metadata` contient des informations supplémentaires sur l'objet, telles que son nom et son Namespace.
- Le champ `spec` décrit la spécification du déploiement, qui inclut :
  - Le nombre de répliques de l'application PrestaShop à créer, dans ce cas, une seule réplique.
  - Le sélecteur pour déterminer quelles répliques doivent être gérées par le déploiement, dans ce cas, toutes les répliques avec l'étiquette `app: prestashop`.
  - Les conteneurs à déployer, dans ce cas, un seul conteneur nommé "prestashop", basé sur l'image Docker "bitnami/prestashop:1.7".
  - Les ports à exposer pour le conteneur PrestaShop, dans ce cas, le port 8080.
  - Les variables d'environnement à passer au conteneur, qui contiennent des informations sensibles telles que les noms d'utilisateur et les mots de passe pour la base de données. Les valeurs de ces variables sont obtenues à partir d'un Secret Kubernetes nommé "secret" et d'une ConfigMap nommée "configmap".
  - Les points de montage des volumes à utiliser pour le conteneur PrestaShop, dans ce cas, un volume nommé "prestashop-vol" est monté dans le conteneur PrestaShop à partir d'un PersistentVolumeClaim nommé "prestapvc".

#### 4/ Mise en place du volume de Prestashop :

Il s'agira de définir à la fois le PV et le PVC (Persistent Volume et Persistent Volume Claim)

- PV :

```
# Persistent Volume
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-prestashop
  labels:
    storage: prestashop
spec:
  capacity:
    storage: 7Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  hostPath:
    path: /tmp/pv-prestashop
    type: Directory
    volumeMode: Filesystem
```

- PVC :

```
# Persistent Volume Claim
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  namespace: 4kube
  name: presta-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 7Gi
  selector:
    matchLabels:
      storage: prestashop
  volumeMode: Filesystem
```

- Le champ apiVersion spécifie la version de l'API Kubernetes à utiliser pour cet objet PersistentVolume.
- Le champ kind indique le type d'objet que vous êtes en train de créer, dans ce cas, un PersistentVolume.
- Le champ metadata contient des informations supplémentaires sur l'objet, telles que son nom et des étiquettes pour l'identifier.
- Le champ spec décrit la spécification du PersistentVolume, qui inclut :
  - La capacité de stockage à allouer au PersistentVolume, dans ce cas, 7 gigaoctets.
  - Les modes d'accès pour le PersistentVolume, dans ce cas, ReadWriteOnce, ce qui signifie que le volume ne peut être monté que sur un seul nœud à la fois en lecture-écriture.
  - La politique de récupération du volume en cas de suppression du PersistentVolume, dans ce cas, Retain, ce qui signifie que les données du volume doivent être conservées même si le PersistentVolume est supprimé.
  - Le type de stockage utilisé pour le PersistentVolume, dans ce cas, un volume de type hostPath qui utilise un chemin d'accès sur le système de fichiers de l'hôte pour stocker les données.

L'ensemble des instructions du PVC est similaire à celui du PV.

5/ Le service :

```
---
# Service
apiVersion: v1
kind: Service
metadata:
  name: service-prestashop
  namespace: 4kube
spec:
  type: NodePort
  ports:
    - port: 80
      name: http
      nodePort: 30080
  selector:
    app: prestashop
```

**Réalisé par :** Othmane BENCHEKROUN

**Année :** M.Eng 1

- Le champ `apiVersion` spécifie la version de l'API Kubernetes à utiliser pour cet objet Service.
- Le champ `kind` indique le type d'objet que vous êtes en train de créer, dans ce cas, un Service.
- Le champ `metadata` contient des informations supplémentaires sur l'objet, telles que son nom et son Namespace.
- Le champ `spec` décrit la spécification du Service, qui inclut :
  - Le type de service à créer, dans ce cas, un service de type "NodePort", qui expose le service sur un port fixe au niveau de chaque nœud du cluster.
  - Les ports à exposer pour le service, dans ce cas, le port 80 est exposé sous le nom "http" avec un nodePort de 30080.
  - Les sélecteurs pour déterminer à quels pods ou répliques l'objet Service doit acheminer le trafic, dans ce cas, tous les pods avec l'étiquette `app: prestashop`.

## 6/ Déploiement de MySQL, de ses volumes et de son service :

Dans la même logique que le déploiement Prestashop :

```
# Deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: 4kube
  name: deployment-mysql
  labels:
    app: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - name: mysql
          image: bitnami/mysql:8.0.32
          ports:
            - containerPort: 3306
          env:
            - name: MYSQL_DATABASE
              valueFrom:
                secretKeyRef:
                  name: secret
                  key: DATABASE_NAME
            - name: ALLOW_EMPTY_PASSWORD
              value: "yes"
            - name: MYSQL_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: secret
                  key: DATABASE_PASSWORD
            - name: MYSQL_USER
              valueFrom:
                secretKeyRef:
                  name: secret
                  key: DATABASE_USER
          volumeMounts:
            - name: mysqlpvc
              mountPath: /bitnami
      volumes:
        - name: mysqlpvc
          persistentVolumeClaim:
            claimName: mysqlpvc
```

Ses volumes (PV/PVC) et son service sont définis également comme suit :

```
# Persistent Volume
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-mysql
  labels:
    storage: mysql
spec:
  capacity:
    storage: 7Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  hostPath:
    path: /tmp/pv-mysql
    type: Directory
  volumeMode: Filesystem
---
# Persistent Volume Claim
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  namespace: 4kub
  name: mysqlpvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 7Gi
  selector:
    matchLabels:
      storage: mysql
  volumeMode: Filesystem
---
# Service
apiVersion: v1
kind: Service
metadata:
  name: service-mysql
  namespace: 4kub
spec:
  type: ClusterIP
  ports:
    - port: 3306
  selector:
    app: mysql
```

## 7/ Mise en place d'un configmap :

```
namespace.yaml x secrets.yaml x * prestashop-deploy.... x mysql-deploy.yaml x configmap.yaml x
1 # ConfigMap
2 apiVersion: v1
3 kind: ConfigMap
4 metadata:
5   namespace: 4kub
6   name: configmap
7 data:
8   PRESTASHOP_FIRST_NAME: Othmane
9   PRESTASHOP_LAST_NAME: BENCHEKROUN
10 |
```

- Le champ apiVersion spécifie la version de l'API Kubernetes à utiliser pour cet objet ConfigMap.
- Le champ kind indique le type d'objet que vous êtes en train de créer, dans ce cas, un ConfigMap.

**Réalisé par :** Othmane BENCHEKROUN

**Année :** M.Eng 1

- Le champ metadata contient des informations supplémentaires sur l'objet, telles que son nom et son Namespace.
- Le champ data contient les données de configuration stockées dans la ConfigMap, qui sont des paires clé-valeur. Dans cet exemple, la ConfigMap contient deux clés : PRESTASHOP\_FIRST\_NAME et PRESTASHOP\_LAST\_NAME, avec les valeurs "Othmane" et "BENCHEKROUN" respectivement.

8/ Ensuite pour créer ces fichiers YAML et les déployer, nous avons défini un fichier YAML faisant appel à tous les fichiers définis ci-haut afin d'éviter la redondance d'appel de la meme commande :

```
resources:  
- secrets.yaml  
- configmap.yaml  
- prestashop-deploy.yaml  
- mysql-deploy.yaml  
- namespace.yaml
```

Enfin nous créons les ressources suivantes à la fois :

```
[root@master-server 4KUBE-BENCHEKROUN-Othmane-MEng]# kubectl apply -k .  
namespace/4kube unchanged  
configmap/configmap unchanged  
secret/secret unchanged  
service/service-mysql unchanged  
service/service-prestashop created  
persistentvolume/pv-mysql unchanged  
persistentvolume/pv-prestashop unchanged  
persistentvolumeclaim/mysqlpvc unchanged  
persistentvolumeclaim/prestapvc unchanged  
deployment.apps/deployment-mysql unchanged  
deployment.apps/deployment-prestashop unchanged
```

On vérifie si les services sont bien déployés en faisant attention à préciser le bon namespace :

```
[root@master-server 4KUBE-BENCHEKROUN-Othmane-MEng]# kubectl get svc -n 4kube  
NAME                TYPE          CLUSTER-IP      EXTERNAL-IP  PORT(S)          AGE  
service-mysql        ClusterIP      10.101.142.216  <none>       3306/TCP         18m  
service-prestashop  NodePort      10.108.48.228   <none>       80:30080/TCP     15m
```

Puis nous exposerons comme prévu le webservice Prestashop dans une forward port. Une fois faite, le service est bien exposé.

### Mise à jour du cluster

Il est important d'appliquer une mise à jour de cluster afin d'apporter des correctifs aux éventuels bugs et d'améliorer la sécurité du cluster.

Pour ce faire, le but du jeu sera de mettre à jour les nodes et les commandes kubeadm, kubelet et kubectl.

Voici les étapes à suivre pour la mise à jour du cluster :

- Nous vérifions tout d'abord les noms de nos nodes (par souci de performances, je ne possède qu'un seul master et un seul worker) :

```
[root@master-server 4KUBE-BENCHEKROUN-0thmane-MEng]# kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
master-server       Ready    control-plane   21d   v1.26.1
worker1             Ready    <none>         21d   v1.26.1
```

- Vérifier la version actuelle de kubeadm :

```
[root@master-server 4KUBE-BENCHEKROUN-0thmane-MEng]# kubeadm version -o json
{
  "clientVersion": {
    "major": "1",
    "minor": "26",
    "gitVersion": "v1.26.1",
    "gitCommit": "8f94681cd294aa8cfd3407b8191f6c70214973a4",
    "gitTreeState": "clean",
    "buildDate": "2023-01-18T15:56:50Z",
    "goVersion": "go1.19.5",
    "compiler": "gc",
    "platform": "linux/amd64"
  }
}
```

- Nous mettons à jour tout le système (la liste des packages MAJ est longue) :

```
[root@master-server 4KUBE-BENCHEKROUN-0thmane-MEng]# sudo yum update
Modules complémentaires chargés : fastestmirror, product-id, search-disabled-repos, subscription-manager

This system is not registered with an entitlement server. You can use subscription-manager to register.

Determining fastest mirrors
 * base: mirror.in2p3.fr
 * extras: distrib-coffee.ipsl.jussieu.fr
 * updates: mirror.in2p3.fr
base                                     | 3.6 kB  00:00:00
docker-ce-stable                       | 3.5 kB  00:00:00
extras                                 | 2.9 kB  00:00:00
kubernetes/x86_64/signature            | 454 B  00:00:00
kubernetes/x86_64/signature            | 1.4 kB  00:00:00 !!!
updates                                | 2.9 kB  00:00:00
(1/2): kubernetes/x86_64/primary        | 126 kB  00:00:00
```

- Nouvelle vérification des paquets disponibles de kubeadm :

```
[root@master-server ~]# sudo yum --showduplicates list kubeadm
Modules complémentaires chargés : fastestmirror, product-id, search-disabled-repos, subscription-manager

This system is not registered with an entitlement server. You can use subscription-manager to register.

Loading mirror speeds from cached hostfile
 * base: mirror.in2p3.fr
 * extras: distrib-coffee.ipsl.jussieu.fr
 * updates: mirror.in2p3.fr
Paquets installés
kubeadm.x86_64 1.26.1-0 @kubernetes
```

- La version de MAJ sur kubeadm est suggérée à partir de la commande « *sudo kubeadm upgrade plan* »

```
[root@master-server ~]# sudo kubeadm upgrade plan
[upgrade/config] Making sure the configuration is correct:
[upgrade/config] Reading configuration from the cluster...
[upgrade/config] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[preflight] Running pre-flight checks.
[upgrade] Running cluster health checks
[upgrade] Fetching available versions to upgrade to
[upgrade/versions] Cluster version: v1.26.1
[upgrade/versions] kubeadm version: v1.26.1
[upgrade/versions] Target version: v1.26.3
[upgrade/versions] Latest version in the v1.26 series: v1.26.3

Components that must be upgraded manually after you have upgraded the control plane with 'kubeadm upgrade apply':
COMPONENT  CURRENT  TARGET
kubelet    2 x v1.26.1  v1.26.3

Upgrade to the latest version in the v1.26 series:
COMPONENT  CURRENT  TARGET
kube-apiserver  v1.26.1  v1.26.3
kube-controller-manager  v1.26.1  v1.26.3
kube-scheduler  v1.26.1  v1.26.3
kube-proxy      v1.26.1  v1.26.3
CoreDNS         v1.9.3   v1.9.3
etcd            3.5.6-0  3.5.6-0

You can now apply the upgrade by executing the following command:

kubeadm upgrade apply v1.26.3
```



**Réalisé par :** Othmane BENCHEKROUN

**Année :** M.Eng 1

Cependant, il s'avère qu'aucun paquet n'est disponible lors de la mise à jour.

```
[root@master-server ~]# sudo yum install -y kubeadm-1.26.3
Modules complémentaires chargés : fastestmirror, product-id, search-disabled-repos, subscription-manager

This system is not registered with an entitlement server. You can use subscription-manager to register.

Loading mirror speeds from cached hostfile
* base: mirror.in2p3.fr
* extras: distrib-coffee.ipsl.jussieu.fr
* updates: mirror.in2p3.fr
Aucun paquet kubeadm-1.26.3 disponible.
Erreur : Rien à faire
```

- Les nodes ont, en revanche, bien été mis à jour :

```
[root@master-server ~]# sudo kubeadm upgrade node
[upgrade] Reading configuration from the cluster...
[upgrade] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[upgrade] Upgrading your Static Pod-hosted control plane instance to version "v1.26.1"...
[upgrade/etcd] Upgrading to TLS for etcd
[upgrade/staticpods] Preparing for "etcd" upgrade
[upgrade/staticpods] Current and new manifests of etcd are equal, skipping upgrade
[upgrade/etcd] Waiting for etcd to become available
[upgrade/staticpods] Writing new Static Pod manifests to "/etc/kubernetes/tmp/kubeadm-upgraded-manifests1749930539"
[upgrade/staticpods] Preparing for "kube-apiserver" upgrade
[upgrade/staticpods] Current and new manifests of kube-apiserver are equal, skipping upgrade
[upgrade/staticpods] Preparing for "kube-controller-manager" upgrade
[upgrade/staticpods] Current and new manifests of kube-controller-manager are equal, skipping upgrade
[upgrade/staticpods] Preparing for "kube-scheduler" upgrade
[upgrade/staticpods] Current and new manifests of kube-scheduler are equal, skipping upgrade
[upgrade] The control plane instance for this node was successfully updated!
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[upgrade] The configuration for this node was successfully updated!
[upgrade] Now you should go ahead and upgrade the kubelet package using your package manager.
```

Pour kubelet et kubelet il ne retrouve cependant pas une possibilité d'upgrade :

```
[root@master-server ~]# sudo yum install -y kubeadm-1.26.3-0.x86_64 kubectll-1.26.3-0.x86_64 kubelet-1.26.3-0.x86_64
Modules complémentaires chargés : fastestmirror, product-id, search-disabled-repos, subscription-manager

This system is not registered with an entitlement server. You can use subscription-manager to register.

Loading mirror speeds from cached hostfile
* base: mirror.in2p3.fr
* extras: distrib-coffee.ipsl.jussieu.fr
* updates: mirror.in2p3.fr
Aucun paquet kubeadm-1.26.3-0.x86_64 disponible.
Aucun paquet kubectll-1.26.3-0.x86_64 disponible.
Aucun paquet kubelet-1.26.3-0.x86_64 disponible.
```

Il s'agit donc de la dernière version possible dans ce cas-là.

## Réalisation d'une pipeline CD

Il faudra installer flux CD à l'aide de helm. Mais pour ce faire, il faudra tout d'abord installer helm.

```
[root@master-server ~]# curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 11345 100 11345 0 0 30606 0 --:--:-- --:--:-- --:--:-- 30662
[WARNING] Could not find git. It is required for plugin installation.
Downloading https://get.helm.sh/helm-v3.11.2-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
[root@master-server ~]# helm version
version.BuildInfo{Version:"v3.11.2", GitCommit:"912ebc1cd10d38d340f048efaf0abda047c3468e", GitTreeState:"clean", GoVersion:"go1.18.10"}
[root@master-server ~]#
```

On ajoute le repo en question avec helm :

```
[root@master-server ~]# helm repo add fluxcd https://charts.fluxcd.io
"fluxcd" has been added to your repositories
```

**Réalisé par :** Othmane BENCHEKROUN

**Année :** M.Eng 1

Enfin nous avons installé flux :

```
[root@master-server 4KUBE-BENCHEKROUN-Othmane-MEng]# helm install flux fluxcd/flux --namespace 4kube --set git.url=https://github.com/obenchekro/fluxcd.  
NAME: flux  
LAST DEPLOYED: Wed Mar 22 11:54:36 2023  
NAMESPACE: 4kube  
STATUS: deployed  
REVISION: 1  
TEST SUITE: None  
NOTES:  
Get the Git deploy key by either (a) running  
  
    kubectl -n 4kube logs deployment/flux | grep identity.pub | cut -d '"' -f2  
  
or by (b) installing fluxctl through  
https://fluxcd.io/legacy/flux/references/fluxctl/#installing-fluxctl  
and running:  
  
    fluxctl identity --k8s-fwd-ns 4kube  
  
---  
  
**Flux v1 is deprecated, please upgrade to v2 as soon as possible!**  
  
New users of Flux can Get Started here:  
https://fluxcd.io/docs/get-started/  
  
Existing users can upgrade using the Migration Guide:  
https://fluxcd.io/docs/migration/
```

Nous vérifions ainsi l'état du statut du déploiement :

```
[root@master-server 4KUBE-BENCHEKROUN-Othmane-MEng]# kubectl -n 4kube rollout status deployment/flux  
Waiting for deployment "flux" rollout to finish: 0 of 1 updated replicas are available...  
deployment "flux" successfully rolled out
```