GRADED EXERCISE - SUBJECT - 3H

Lettat Samia

# Créer un namespace qui porte le nom exam-kub dédié à notre application

Définition : Un namespace est un cluster virtuelle dans un cluster physique Kubernetes. Il permet d’isoler notre application, et projet en fonction de nous besoin.

On vac créer notre ns avec yaml:

* **nano 285893.yaml**

voici le fichier yaml pour notre namespace

apiVersion: v1 kind: Namespace metadata:

k8s-client@k8s-client:~$ kubectl create -f285893.yaml namespace/285893 created

Lister les namespaces au sein de notre cluster : **k8s-client@k8s-client:~$ kubectl get ns NAME STATUS AGE**

298366 Active 58m

default Active 14h kube-node-lease Active 14h kube-public Active 14h kube-system Active 14h

changer le namespace par défaut par cette commande :

kubectl config set-context $(kubectl config current-context) --namespace 298366

1. volumes

Avant de déployer prestashop on va créer d’abord les volumes

Pvc , pv données doivent être persistées, vous utiliserez les objets de type pv, pvc

nano pv\_pvc.yaml :

apiVersion: v1

kind: PersistentVolume metadata:

name: prestashop-pv labels:

prestashop\_storage: 'true' spec:

capacity:

storage: 5Gi volumeMode: Filesystem accessModes:

- ReadWriteOnce persistentVolumeReclaimPolicy: Retain storageClassName: ''

hostPath:

path: /data/prestashop type: DirectoryOrCreate

---

apiVersion: v1

kind: PersistentVolume metadata:

name: mariadb-pv labels:

mariadb\_storage: 'true' spec:

capacity:

storage: 5Gi volumeMode: Filesystem accessModes:

- ReadWriteOnce persistentVolumeReclaimPolicy: Retain storageClassName: ''

hostPath:

path: /data/mariadb type: DirectoryOrCreate

---

apiVersion: v1

kind: PersistentVolumeClaim metadata:

namespace: '285893' name: prestashop-pvc spec:

accessModes:

- ReadWriteOnce volumeMode: Filesystem resources:

requests:

storage: 5Gi storageClassName: '' selector: matchLabels:

prestashop\_storage: 'true'

---

apiVersion: v1

kind: PersistentVolumeClaim metadata:

namespace: '285893' name: mariadb-pvc spec:

accessModes:

- ReadWriteOnce volumeMode: Filesystem resources:

requests: storage: 5Gi

storageClassName: '' selector: matchLabels:

mariadb\_storage: 'true'

k8s-client@k8s-client:~$ kubectl create -f pv\_pvc.yaml persistentvolume/prestashop-pv created persistentvolume/mariadb-pv created persistentvolumeclaim/prestashop-pvc created persistentvolumeclaim/mariadb-pvc created

# secret :

maintenant on va créer le fichier yaml pour Secret ou se trouveront les informations confidentielles ( ex : PRESTASHOP\_DATABASE\_NAME , PRESTASHOP\_DATABASE\_USER, PRESTASHOP\_DATABASE\_PASSWORD )

k8s-client@k8s-client:~$ nano secret\_prestashop.yaml

apiVersion: v1 kind: Secret metadata:

namespace: '285893' name: prestashop-secret type: Opaque stringData:

prestashop\_password: 'SUPINFOmsc1' mariadb\_host: mariadb-service prestashop\_database\_name: prestashop\_name prestashop\_database\_user: prestashop\_user prestashop\_database\_password: 'SUPINFOmsc1'

mariadb\_root\_password: 'SUPINFOmsc1'

k8s-client@k8s-client:~$ kubectl create -f secret\_prestashop.yaml secret/prestashop-secret created

1. **configMap**

ConfigMap ou se trouveront les informations concernant le site ( ex : PRESTASHOP\_FIRST\_NAME , PRESTASHOP\_LAST\_NAME )

k8s-client@k8s-client:~$ nano configMap\_prestashop.yaml

apiVersion: v1 kind: ConfigMap metadata:

namespace: '285893'

name: prestashop-configmap

k8s-client@k8s-client:~$ kubectl create -f configMap\_prestashop.yaml configmap/prestashop-configmap created

# deployment prestashop

Maintenent on peut créer notre deployment

k8s-client@k8s-client:~$ nano deployment\_prestashop.yaml

apiVersion: apps/v1 kind: Deployment metadata: namespace: '285893'

name: prestashop-deployment labels:

app: prestashop spec:

replicas: 1 selector: matchLabels:

app: prestashop template: metadata:

labels:

app: prestashop spec: initContainers:

* **name: init-prestashop image: busybox:1.28 command:**
  + **chmod**

- '777'

* + **'/bitnami/prestashop' volumeMounts:**

- mountPath: "/bitnami/prestashop" name: mypd

containers:

* **name: prestashop**

image: bitnami/prestashop:1.7 volumeMounts:

* + **mountPath: "/bitnami/prestashop" name: mypd**

ports:

* + **containerPort: 80 env:**
    - **name: PRESTASHOP\_FIRST\_NAME**

valueFrom: configMapKeyRef:

name: prestashop-configmap key: prestashop\_first\_name

* + - **name: PRESTASHOP\_LAST\_NAME**

valueFrom: configMapKeyRef:

name: 'prestashop-configmap' key: prestashop\_last\_name

* + - **name: PRESTASHOP\_HOST**

valueFrom: configMapKeyRef:

name: 'prestashop-configmap' key: prestashop\_host

* + - **name: PRESTASHOP\_PASSWORD**

valueFrom:

secretKeyRef:

name: 'prestashop-secret' key: prestashop\_password

* + - **name: PRESTASHOP\_EMAIL**

valueFrom:

secretKeyRef:

name: 'prestashop-secret' key: prestashop\_email

* + - **name: MARIADB\_HOST**

valueFrom:

secretKeyRef:

name: 'prestashop-secret' key: mariadb\_host

* + - **name: PRESTASHOP\_DATABASE\_NAME**

valueFrom:

secretKeyRef:

name: 'prestashop-secret'

key: prestashop\_database\_name

* + - **name: PRESTASHOP\_DATABASE\_USER**

valueFrom:

secretKeyRef:

name: 'prestashop-secret'

key: prestashop\_database\_user

* + - **name: PRESTASHOP\_DATABASE\_PASSWORD**

valueFrom: secretKeyRef:

name: 'prestashop-secret'

key: prestashop\_database\_password volumes:

* **name: mypd persistentVolumeClaim:**

claimName: prestashop-pvc

k8s-client@k8s-client:~$ kubectl create -f deployment\_prestashop.yaml deployment.apps/prestashop-deployment created

# deployment mariadb

k8s-client@k8s-client:~$ nano deployment\_mariadb.yaml apiVersion: apps/v1

kind: Deployment metadata: namespace: '285893'

name: mariadb-deployment labels:

app: mariadb spec:

replicas: 1 selector: matchLabels:

app: mariadb template: metadata:

labels:

app: mariadb spec:

initContainers:

* **name: init-mariadb image: busybox:1.28 command:**
  + **chmod**

- '777'

* + **'/bitnami/mariadb' volumeMounts:**

- mountPath: "/bitnami/mariadb" name: mypd

containers:

* **name: mariadb**

image: bitnami/mariadb:10.1 volumeMounts:

* + **mountPath: "/bitnami/mariadb" name: mypd**

ports:

* + **containerPort: 3306 env:**

- name: MARIADB\_USER

valueFrom: secretKeyRef:

name: prestashop-secret

key: prestashop\_database\_user

- name: MARIADB\_PASSWORD

valueFrom: secretKeyRef:

name: prestashop-secret

key: prestashop\_database\_password

- name: MARIADB\_DATABASE

valueFrom:

secretKeyRef:

name: prestashop-secret

key: prestashop\_database\_name

- name: MARIADB\_ROOT\_PASSWORD

valueFrom:

secretKeyRef:

name: prestashop-secret

key: mariadb\_root\_password volumes:

* **name: mypd persistentVolumeClaim:**

claimName: mariadb-pvc

k8s-client@k8s-client:~$ kubectl create -f deployment\_mariadb.yaml deployment.apps/mariadb-deployment created

# creation des services

* **prestashop :**

k8s-client@k8s-client:~$ nano service\_prestashop.yaml apiVersion: v1

kind: Service metadata: namespace: '285893'

name: prestashop-service spec:

type: NodePort selector:

app: prestashop ports:

- port: 8080

nodePort: 30000 name: http

k8s-client@k8s-client:~$ kubectl create -f service\_prestashop.yaml service/prestashop-service created

* **service mariadb**

k8s-client@k8s-client:~$ nano service\_mariabd.yaml

apiVersion: v1 kind: Service metadata:

namespace: '285893' name: mariadb-service spec:

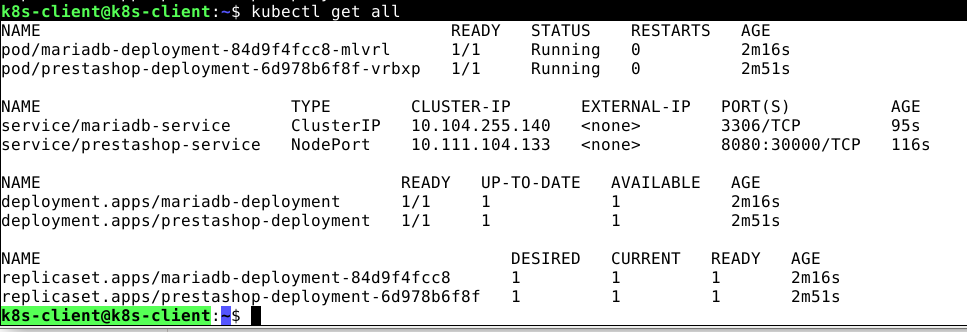
type: ClusterIP selector:

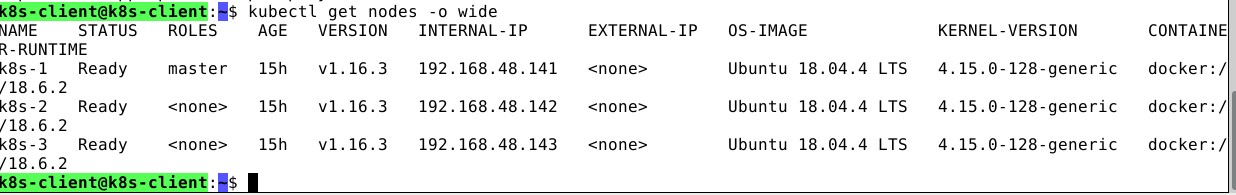
app: mariadb ports:

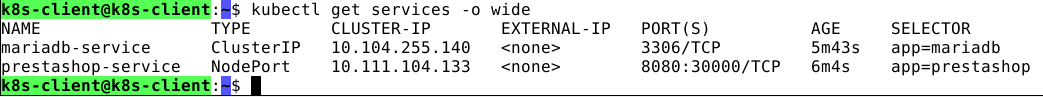
- port: 3306

k8s-client@k8s-client:~$ kubectl create -f service\_mariabd.yaml service/mariadb-service created

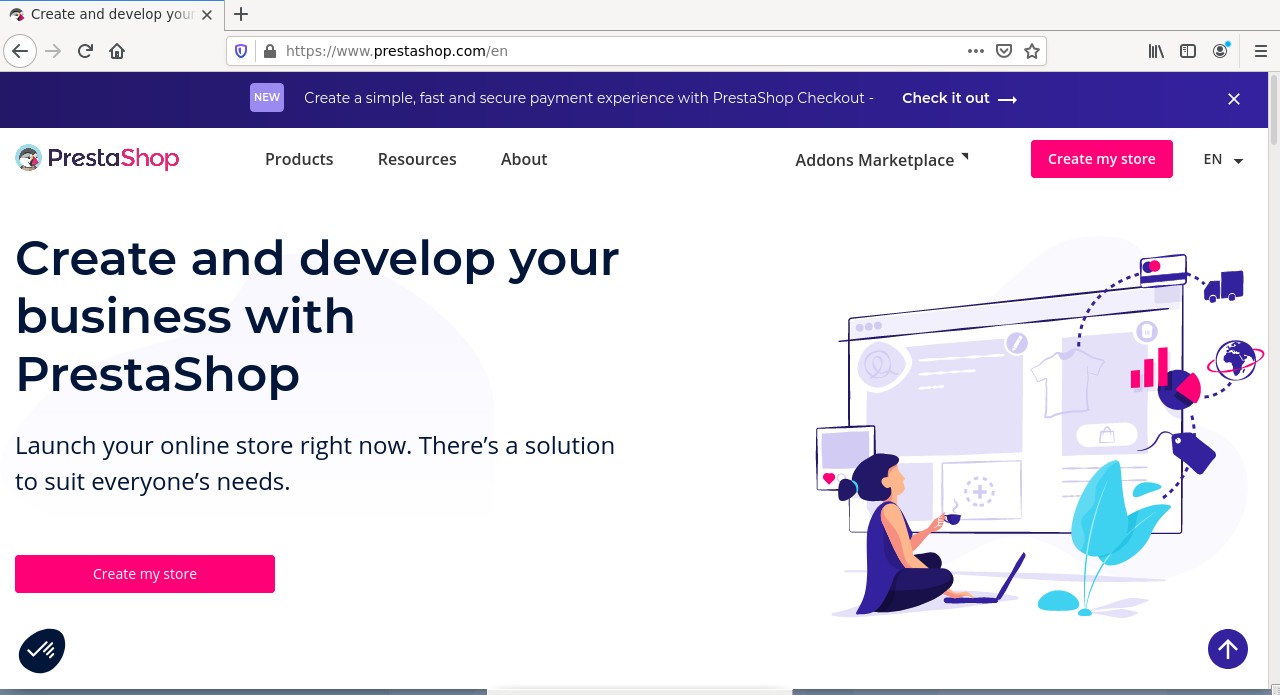
un petit get all pour voire nous ressources







sur le navigateur notre application est a l’ecoute sur 192.168.48.141:30000



# La Mise a jours des cluster

On doit tout d'abord commencer par mettre à jour notre nœud master k8s-1 , et par la suite mettre à jour les autres nœuds de travail un par un k8s-2 et k8s3.

Dans mon cas, je souhaite upgrade mon master de la version 1.16 vers la version 1.17. Pour ce faire, nous entamerons une phase de recherche afin de trouver la dernière version stable de kubeadm et kubelet en v1.17. je vais utiliser l'outil (apt) afin de rechercher mes nouveaux paquets

**apt-get update && \**

**apt-cache policy kubeadmCopier**

**Résultat :**

kubeadm:

Installed: 1.16.0-00

Candidate: 1.17.0-00

...

apt-cache policy kubelet

**Résultat :**

kubelet:

Installed: 1.16.0-00

Candidate: 1.17.0-00

...

d'abord nous attaquer l'installation du nouveau paquet kubeadm :

**apt-get upgrade -y kubeadm=1.17.0-00**

Ensuite, nous vérifions que le téléchargement fonctionne et possède la version attendue

**kubeadm version Résultat :**

kubeadm version: &version.Info{Major:"1", mineur:"17", GitVersion:"v1.17.0" ...}

À présent, nous allons interroger l'outil kubeadm

**kubeadm upgrade plan**

**Résultat :**

Components that must be upgraded manually after you have upgraded the control plane with 'kubeadm upgrade apply':

COMPONENT CURRENT AVAILABLE

Kubelet 4 x v1.16.0 v1.17.0

Upgrade to the latest stable version:

COMPONENT CURRENT AVAILABLE

API Server v1.16.0 v1.17.0

Controller Manager v1.16.0 v1.17.0

Scheduler v1.16.0 v1.17.0

Kube Proxy v1.16.0 v1.17.0

CoreDNS 1.6.2 1.6.5

Etcd 3.3.15 3.4.3-0

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

kubeadm upgrade apply v1.17.0

L'étape suivante consiste à **rendre notre nœud master unschedulable**: **kubectl drain k8s-1 --ignore-daemonsets**

Enfin, nous allons appliquer les nouveaux changement grâce la commande suivante :

**kubeadm upgrade apply v1.17.0 Résultat :**

[upgrade/successful] SUCCESS! Your cluster was upgraded to "v1.17.0". Enjoy!

Nous devons par la suite installer la version stable de kubelet en v1.17.0 et redémarrer le service de manière à prendre en compte sa nouvelle version :

**apt-get upgrade -y kubelet=1.17.0-00 && \ systemctl restart kubelet**

Enfin, on n'oublie pas de rendre notre nœud à nouveau schedulable :

**kubectl uncordon master**

En lançant la commande ci-dessous, on peut remarquer que nos nœuds workers ne possèdent pas la même version que notre nœud master:

kubectl get nodes

**Résultat :**

**NAME STATUS ROLES AGE VERSION**

**K8s-1 Ready master 33m v1.17.0 K8s-2 Ready <none> 32m v1.16.0 K8s-3 Ready <none> 31m v1.16.0**

Upgrade des worker k8s-2 et k8s-3

Premièrement, nous allons rendre notre nœud de travail unschedulable depuis notre nœud master:

kubectl drain k8s-2 --ignore-daemonsets

Ensuite il faut être connecté sur un worker et installer les dernières versions stables de kubeadm et kubelet depuis notre nœud de travail :

ssh root@k8s2 "apt-get update && apt-get upgrade -y kubeadm=1.17.0-00 kubelet=1.17.0-00"

Depuis notre master nous allons mettre à niveau notre worker :

kubeadm upgrade k8s-2 config --kubelet-version v1.17.0

Ensuite, il faut redémarrer le service kubelet :

ssh root@ "systemctl restart kubelet"

Enfin, on n'oublie pas de rendre notre nœud à nouveau schedulable :

kubectl uncordon k8s-2

on fait la meme chose pour le node k8s-3

En vérifiant les nœuds disponibles de note cluster

kubectl get nodes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Résultat : |  | | | |
| NAME | STATUS | ROLES | AGE | VERSION |
| K8s-1 | Ready | master | 55m | v1.17.0 |
| K8s-2 | Ready | <none> | 53m | v1.17.0 |
| K8s-3 | Ready | <none> | 51m | v1.17.0 |

# Le lien github

<https://github.com/batachelydia/examen>