

TP 2 – Premier pas avec Kubernetes

Objectif général

Rendre concrets les concepts vus en cours (Kubernetes) au travers d'exercices graduels. Vous serez capable de créer votre premier cluster K8S et de déployer vos premières applications.

Exercice 1 – Installation et configuration de votre premier cluster

Objectifs :

- Avoir un cluster K3S et un environnement de travail opérationnel
- Vérifier l'état de santé de son cluster

Outils conseillés : K3S

Durée indicative : 10-20 min

Consignes

1. Sur un serveur Debian, installez K3S en respectant les étapes suivantes :
 - a. Mettre à jour votre serveur et installé **curl** (si ce n'est pas déjà fait)
 - b. Installez K3S :

```
$ curl -sfL https://get.k3s.io | INSTALL_K3S_EXEC="--write-kubeconfig-mode 644" sh -s -
```

1. Vérifiez l'état de votre cluster :

```
$ kubectl cluster-info  
$ kubectl get nodes  
$ kubectl get pods --all-namespaces
```

2. Installation de HELM via apt :

```
$ sudo apt-get install curl gpg apt-transport-https --yes  
$ curl -fsSL https://packages.buildkite.com/helm-linux/helm-debian/gpgkey | gpg --dearmor |  
sudo tee /usr/share/keyrings/helm.gpg > /dev/null  
$ echo "deb [signed-by=/usr/share/keyrings/helm.gpg] https://packages.buildkite.com/helm-  
linux/helm-debian/any/ any main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list  
$ sudo apt-get update  
$ sudo apt-get install helm  
$ helm completion bash > ~/.helm-completion.bash  
$ echo "source ~/.helm-completion.bash" >> ~/.bashrc  
$ source ~/.bashrc
```

Exercice 2 – Déploiement via kubectl

Objectifs :

- Savoir écrire des manifests simples
- Etre capable de déployer/modifier/supprimer des applications sur kubernetes

Outils conseillés : kubectl

Durée indicative : 40-50 min

Consignes

1. Nous allons commencer par créer un Namespace nommé « hello-world ».
2. Créer votre premier fichier de manifest et définir dedans une ressource de type pod pour créer un conteneur basé sur l'image « hello-world:latest » et affichez ses logs. Ce conteneur doit être dans le namespace hello-world.
3. Ajouter une ressource deployment pour créer 5 réplicas d'un conteneur basé sur l'image « digitalocean/flask-helloworld:latest » et déployé le. Vérifier que nos conteneurs sont bien présents et dans le bon namespace.
4. Supprimer un pod flask et afficher la liste des pods. Que constatez-vous ?
5. On va maintenant créer une ressource service de type « nodeport » (attention : c'est le port 5000).
6. Afficher les services du namespace « hello-world » et récupérer le port ouvert. Vous pouvez vous y connecter en web depuis votre navigateur en tapant l'ip de votre serveur avec le port (>30000).
7. On va maintenant créer une ressource de type ingress avec le nom de domaine de votre choix (exemple : mon-serveur-flask.com) vers notre service flask.
On peut vérifier en modifiant notre fichier hosts local (linux / mac : /etc/hosts, windows : C:\windows\system32\drivers\etc\hosts).

Exercice 3 – Déploiement via HELM

Objectifs :

- Etre capable de déployer une application via HELM
- Savoir modifier des paramètres de base

Outils conseillés : helm, kubectl

Durée indicative : 30-40 min

Consignes

1. Créer un namespace « helm-nginx »
2. Ajouter le dépôt de charts HELM <https://charts.bitnami.com/bitnami>. Elle doit apparaître dans la liste « helm repo list ».
3. Listez les charts disponibles dans le dépôt Bitnami.
4. Installer nginx dans le bon namespace et vérifiez l'installation avec kubectl
5. On va vouloir accéder à notre application depuis l'extérieur, utilisez la commande helm pull pour récupérer la charte par défaut et récupérez le fichier values.yaml.
6. Modifier le fichier values.yaml pour passer le service de LoadBalancer à NodePort, relancer l'installation et vérifier que la modification a été apportée.
7. Faire la même chose pour activer l'autoscaling et mettez un check à 50% sur le CPU et la RAM. Vérifier le résultat
8. Activer l'ingress dans votre fichier values.yaml et mettez à jour votre application et vérifier le résultat.

Exercice 4 – Kustomize

Objectifs :

- Etre capable de déployer une application et la modifier avec Kustomize

Outils conseillés : kubectl, kustomize

Durée indicative : 20-25 min

Consignes

1. A partir du / des manifest(s) de l'exercice 2, nous allons créer une arborescence kustomize (cf. cours)
2. Créer des overlays pour de la production et pour le développement
3. Créer des namespaces séparés pour la prod et le dev
- 4.Modifier le nombre de réplicas à 2 pour le dev et 8 pour la production et déployer dans les bons namespaces.