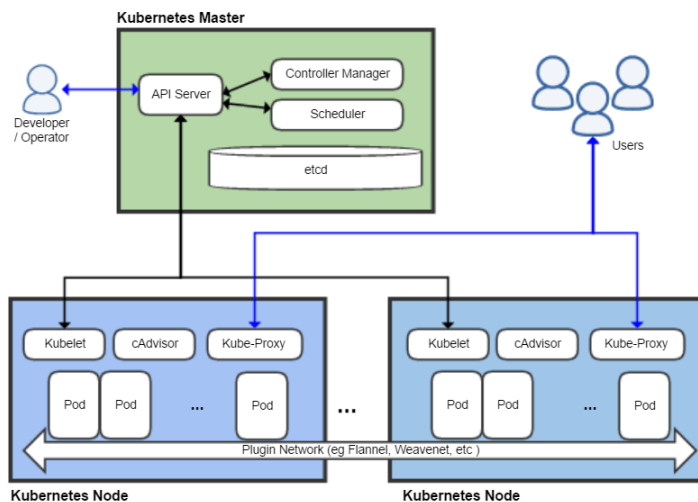


# Kubernetes

## Principes

### Architecture



### Namespaces

Objet qui permet de ranger dans différents environnements

### Labels

Affecté à un objet pour pouvoir faire différentes catégories. Peut être utilisé pour des commandes y compris delete par exemple.

## kubectl

### apply

Applique un changement de configuration à une ressource depuis un fichier ou stdin.

```
kubectl apply -f <fichier.yaml>
```

## get

Liste une ou plusieurs ressources.

```
kubectl get namespaces
kubectl -n <namespace> get nodes
kubectl -n <namespace> get pods
kubectl -n <namespace> get all
kubectl -n <namespace> get ingress
kubectl -n <namespace> get pvc
kubectl -n <namespace> get configmap
kubectl -n <namespace> get secret
kubectl -w -n <namespace> get events
```

### Interroger tous les namespaces

- --all-namespaces

### Options d'affichage

- --output=wide ou -o wide
- --output=json ou -o json
- --output=yaml ou -o yaml

### Exemple

- Interrogation des noeuds

```
kubectl get nodes
```

- Interrogation sans entête en affichant juste le nom

```
kubectl get namespaces --no-headers -o custom-columns=":metadata.name"
kubectl get pods --no-headers -o custom-columns=":metadata.name"
```

- Interrogation des pods sur un namespace

```
kubectl get pods -n <namespace>
```

- Interrogation des pods avec le node sur lequel ils tournent et adresse IP des pods

```
kubectl get pods --all-namespaces -o wide
```

## describe

Affiche l'état détaillé d'une ou plusieurs ressources.

```
kubectl describe pods <nom du pod>
```

On peut mettre le début du nom du pod, il affichera tous les pods dont le nom commence par cette chaîne de caractères.

## logs

Affiche les logs d'un container dans un pod (sortie standard et sortie d'erreur).

### Pour un pod

```
kubectl logs <nom du pod>
```

### Pour un container au sein d'un pod

```
kubectl logs -c <nom du container>
```

## create

Crée une ou plusieurs ressources depuis un fichier ou stdin.

```
kubectl create pod <nom du pod> --image <nom image>  
kubectl create job <nom du job> --image <nom image>  
kubectl create deployment <nom de deployment> --image <nom image>
```

Par défaut quand on lance un pod il considère que c'est une application à boucle d'évènements. Pour lancer un script qui se termine au bout d'un moment il faut lancer un job.

## delete

Supprime des ressources soit depuis un fichier ou stdin, ou en indiquant des sélecteurs de label, des noms, des sélecteurs de ressources ou des ressources.

```
kubectl delete namespaces <namespace>
```

Attention détruit tous les objets de ce namespace !

## rollout

### Interrogation

```
kubectl rollout status deployment.apps/lbs-maria # état du
déploiement
kubectl rollout history deployment.apps/lbs-maria #
historique des déploiements
kubectl rollout history deployment.apps/lbs-maria --revision 35 # détail
d'un déploiement
```

### Redémarrage

```
kubectl rollout restart deployment <pod>
```

### Retour arrière

```
kubectl rollout undo deployment.apps/lbs-maria # retour
à la version précédente
kubectl rollout undo deployment.apps/lbs-maria --to-revision 30 # retour
à la version 30
```

### Source

## exec

Exécute une commande à l'intérieur d'un conteneur dans un pod. La commande bash permet d'ouvrir une console.

```
kubectl -n <namespace> exec -it <pod> -- bash
```

Le caractère \* n'est pas accepté en tant que méta caractère de la commande kubectl exec. Voir palliatif ci-dessous.

```
kubectl -n <namespace> exec -it <pod> -- rm -f fichier.* # ne
fonctionne pas
kubectl -n <namespace> exec -it <pod> -- sh -c "rm -f fichier.*" # OK
```

## cp

```
kubectl -n <namespace> cp <pod>:<chemin_vers_fichier_distant>
<chemin_vers_fichier_local> -c <nom_du_module>
```

```
kubectl -n <namespace> cp <chemin_vers_fichier_local>
<pod>:<chemin_vers_fichier_distant> -c <nom_du_module>
kubectl -n project6 cp lbs-maria-5d49f758b6-drt2p:/tmp/toto toto -c lbs-
maria
```

vérifier le propriétaire du fichier une fois transféré sur le pod

Nécessite la présence de la commande tar sur le container. Palliatif ci-dessous pour les fichiers textes.

### exec: "tar": executable file not found in \$PATH: unknown

```
cat <fichier_local> | kubectl exec -i -n <namespace> <pod> -c <container> --
sh -c "cat > <fichier_distant>" # copie vers le pod
kubectl exec -n <namespace> <pod> -- cat <fichier_distant> >
<fichier_local>
```

## scale

```
kubectl -n <namespace> get deployments
kubectl -n <namespace> scale deployment <nom> --replicas=0 # on supprime
toutes les instances du pod
kubectl -n <namespace> scale deployment --all --replicas=0 # on supprime
toutes les instances de tous les pods
kubectl -n <namespace> scale deployment <nom> --replicas=1 # création d'une
instance du pod
kubectl -n <namespace> get sts
kubectl -n <namespace> scale statefulset <nom> --replicas=1 # création d'une
instance du pod (cas statefulset au lieu de deployment)
```

[Sources](#)

## Réseau

[Haut de page](#)

From:

<https://wiki.iot-acis.fr/> - **Wiki**

Permanent link:

<https://wiki.iot-acis.fr/doku.php?id=all:bibles:virtualisation:kubernetes>

Last update: **2025/06/24 15:03**

