CLI kubectl

kubectl CLI description

1 - Aide-mémoire de kubectl

Cette page contient une liste des commandes et options fréquemment utilisées de `kubectl`.

> **Note:**
> Ces instructions concernent Kubernetes v1.32. Pour vérifier la version, utilisez la commande `kubectl version`.

Autocomplétion de Kubectl

BASH

```
source <(kubectl completion bash) # set up autocomplete in bash into the current shell, bash-completion package sho
echo "source <(kubectl completion bash)" >> ~/.bashrc # add autocomplete permanently to your bash shell.
```

Vous pouvez également utiliser un alias pour kubectl qui fonctionne aussi avec l'autocomplétion.

```
alias k=kubectl
complete -o default -F __start_kubectl k
```

ZSH

```
source <(kubectl completion zsh)  # set up autocomplete in zsh into the current shell
echo '[[ $commands[kubectl] ]] && source <(kubectl completion zsh)' >> ~/.zshrc # add autocomplete permanently to y
```

FISH

> **Note:**
> Requires kubectl version 1.23 or above.

```
echo 'kubectl completion fish | source' > ~/.config/fish/completions/kubectl.fish && source ~/.config/fish/completi
```

Remarque concernant `--all-namespaces`

L'utilisation de `--all-namespaces` (tous les espaces de nommage) est assez fréquente, alors sachez qu'il existe un raccourci pour cela :

`kubectl -A`

Contexte et configuration de Kubectl

Définissez quel cluster Kubernetes doit être utilisé avec `kubectl`, et modifiez les paramètres de configuration. Pour plus de détails sur le fichier de configuration, consultez la documentation Configurer l'accès à plusieurs clusters.

```
kubectl config view # Show Merged kubeconfig settings.

# use multiple kubeconfig files at the same time and view merged config
KUBECONFIG=~/.kube/config:~/.kube/kubconfig2

kubectl config view

# Show merged kubeconfig settings and raw certificate data and exposed secrets
kubectl config view --raw

# get the password for the e2e user
kubectl config view -o jsonpath='{.users[?(@.name == "e2e")].user.password}'

# get the certificate for the e2e user
kubectl config view --raw -o jsonpath='{.users[?(.name == "e2e")].user.client-certificate-data}' | base64 -d

kubectl config view -o jsonpath='{.users[].name}'    # display the first user
kubectl config view -o jsonpath='{.users[*].name}'   # get a list of users
kubectl config get-contexts                          # display list of contexts
kubectl config get-contexts -o name                  # get all context names
kubectl config current-context                       # display the current-context
kubectl config use-context my-cluster-name           # set the default context to my-cluster-name

kubectl config set-cluster my-cluster-name           # set a cluster entry in the kubeconfig

# configure the URL to a proxy server to use for requests made by this client in the kubeconfig
kubectl config set-cluster my-cluster-name --proxy-url=my-proxy-url

# add a new user to your kubeconf that supports basic auth
kubectl config set-credentials kubeuser/foo.kubernetes.com --username=kubeuser --password=kubepassword

# permanently save the namespace for all subsequent kubectl commands in that context.
kubectl config set-context --current --namespace=ggckad-s2

# set a context utilizing a specific username and namespace.
kubectl config set-context gce --user=cluster-admin --namespace=foo \
  && kubectl config use-context gce

kubectl config unset users.foo                       # delete user foo

# short alias to set/show context/namespace (only works for bash and bash-compatible shells, current context to be s
alias kx='f() { [ "$1" ] && kubectl config use-context $1 || kubectl config current-context ; } ; f'
alias kn='f() { [ "$1" ] && kubectl config set-context --current --namespace $1 || kubectl config view --minify | g
```

Kubectl apply

`apply` gère les applications à travers des fichiers définissant les ressources Kubernetes. Il crée et met à jour les ressources dans un cluster en exécutant `kubectl apply`. C'est la méthode recommandée pour gérer les applications Kubernetes en production. Consultez Kubectl Book.

Création d'objets

Les manifestes Kubernetes peuvent être définis en YAML ou en JSON. Les extensions de fichier `.yaml`, `.yml`, et `.json` peuvent être utilisées.

```
kubectl apply -f ./my-manifest.yaml                # create resource(s)
kubectl apply -f ./my1.yaml -f ./my2.yaml          # create from multiple files
kubectl apply -f ./dir                             # create resource(s) in all manifest files in dir
kubectl apply -f https://example.com/manifest.yaml  # create resource(s) from url (Note: this is an example domain a
kubectl create deployment nginx --image=nginx      # start a single instance of nginx

# créer un Job qui imprime « Hello World » (bonjour le monde)
kubectl create job hello --image=busybox:1.28 -- echo "Hello World"

# create a CronJob that prints "Hello World" every minute
kubectl create cronjob hello --image=busybox:1.28   --schedule="*/1 * * * *" -- echo "Hello World"

kubectl explain pods                           # get the documentation for pod manifests

# Create multiple YAML objects from stdin
kubectl apply -f - <<EOF
apiVersion: v1
kind: Pod
metadata:
  name: busybox-sleep
spec:
  containers:
  - name: busybox
    image: busybox:1.28
    args:
    - sleep
    - "1000000"
---
apiVersion: v1
kind: Pod
metadata:
  name: busybox-sleep-less
spec:
  containers:
  - name: busybox
    image: busybox:1.28
    args:
    - sleep
    - "1000"
EOF

# Create a secret with several keys
kubectl apply -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  password: $(echo -n "s33msi4" | base64 -w0)
  username: $(echo -n "jane" | base64 -w0)
EOF
```

Consultez et trouvez les ressources

```
# Get commands with basic output
kubectl get services                          # List all services in the namespace
kubectl get pods --all-namespaces             # List all pods in all namespaces
kubectl get pods -o wide                      # List all pods in the current namespace, with more details
kubectl get deployment my-dep                 # List a particular deployment
kubectl get pods                              # List all pods in the namespace
kubectl get pod my-pod -o yaml                # Get a pod's YAML

# Describe commands with verbose output
kubectl describe nodes my-node
kubectl describe pods my-pod

# List Services Sorted by Name
kubectl get services --sort-by=.metadata.name

# List pods Sorted by Restart Count
kubectl get pods --sort-by='.status.containerStatuses[0].restartCount'

# List PersistentVolumes sorted by capacity
kubectl get pv --sort-by=.spec.capacity.storage

# Get the version label of all pods with label app=cassandra
kubectl get pods --selector=app=cassandra -o \
  jsonpath='{.items[*].metadata.labels.version}'

# Retrieve the value of a key with dots, e.g. 'ca.crt'
kubectl get configmap myconfig \
  -o jsonpath='{.data.ca\.crt}'

# Retrieve a base64 encoded value with dashes instead of underscores.
kubectl get secret my-secret --template='{{index .data "key-name-with-dashes"}}'

# Get all worker nodes (use a selector to exclude results that have a label
# named 'node-role.kubernetes.io/control-plane')
kubectl get node --selector='!node-role.kubernetes.io/control-plane'

# Get all running pods in the namespace
kubectl get pods --field-selector=status.phase=Running

# Get ExternalIPs of all nodes
kubectl get nodes -o jsonpath='{.items[*].status.addresses[?(@.type=="ExternalIP")].address}'

# List Names of Pods that belong to Particular RC
# "jq" command useful for transformations that are too complex for jsonpath, it can be found at https://jqlang.githu
sel=${$(kubectl get rc my-rc --output=json | jq -j '.spec.selector | to_entries | .[] | "\(.key)=\(.value),"')%?}
echo $(kubectl get pods --selector=$sel --output=jsonpath={.items..metadata.name})

# Show labels for all pods (or any other Kubernetes object that supports labelling)
kubectl get pods --show-labels

# Check which nodes are ready
JSONPATH='{range .items[*]}{@.metadata.name}:{range @.status.conditions[*]}{@.type}={@.status};{end}{end}' \
 && kubectl get nodes -o jsonpath="$JSONPATH" | grep "Ready=True"

# Check which nodes are ready with custom-columns
kubectl get node -o custom-columns='NODE_NAME:.metadata.name,STATUS:.status.conditions[?(@.type=="Ready")].status'

# Output decoded secrets without external tools
kubectl get secret my-secret -o go-template='{{range $k,$v := .data}}{{"### "}}{{$k}}{{"\n"}}{{$v|base64decode}}{{"

# List all Secrets currently in use by a pod
kubectl get pods -o json | jq '.items[].spec.containers[].env[]?.valueFrom.secretKeyRef.name' | grep -v null | sort

# List all containerIDs of initContainer of all pods
# Helpful when cleaning up stopped containers, while avoiding removal of initContainers.
kubectl get pods --all-namespaces -o jsonpath='{range .items[*].status.initContainerStatuses[*]}{.containerID}{"\n"}

# List Events sorted by timestamp
kubectl get events --sort-by=.metadata.creationTimestamp

# List all warning events
```

```
kubectl events --types=Warning

# Compares the current state of the cluster against the state that the cluster would be in if the manifest was appl
kubectl diff -f ./my-manifest.yaml

# Produce a period-delimited tree of all keys returned for nodes
# Helpful when locating a key within a complex nested JSON structure
kubectl get nodes -o json | jq -c 'paths|join(".")'

# Produce a period-delimited tree of all keys returned for pods, etc
kubectl get pods -o json | jq -c 'paths|join(".")'

# Produce ENV for all pods, assuming you have a default container for the pods, default namespace and the `env` com
# Helpful when running any supported command across all pods, not just `env`
for pod in $(kubectl get po --output=jsonpath={.items..metadata.name}); do echo $pod && kubectl exec -it $pod -- en

# Get a deployment's status subresource
kubectl get deployment nginx-deployment --subresource=status
```

Mise à jour des ressources

```
kubectl set image deployment/frontend www=image:v2               # Rolling update "www" containers of "frontend" de
kubectl rollout history deployment/frontend                      # Check the history of deployments including the r
kubectl rollout undo deployment/frontend                         # Rollback to the previous deployment
kubectl rollout undo deployment/frontend --to-revision=2         # Rollback to a specific revision
kubectl rollout status -w deployment/frontend                    # Watch rolling update status of "frontend" deploy
kubectl rollout restart deployment/frontend                      # Rolling restart of the "frontend" deployment


cat pod.json | kubectl replace -f -                              # Replace a pod based on the JSON passed into stdi

# Force replace, delete and then re-create the resource. Will cause a service outage.
kubectl replace --force -f ./pod.json

# Create a service for a replicated nginx, which serves on port 80 and connects to the containers on port 8000
kubectl expose rc nginx --port=80 --target-port=8000

# Update a single-container pod's image version (tag) to v4
kubectl get pod mypod -o yaml | sed 's/\(image: myimage\):.*$/\1:v4/' | kubectl replace -f -

kubectl label pods my-pod new-label=awesome                      # Add a Label
kubectl label pods my-pod new-label-                             # Remove a label
kubectl label pods my-pod new-label=new-value --overwrite        # Overwrite an existing value
kubectl annotate pods my-pod icon-url=http://goo.gl/XXBTWq       # Add an annotation
kubectl annotate pods my-pod icon-url-                           # Remove annotation
kubectl autoscale deployment foo --min=2 --max=10                # Auto scale a deployment "foo"
```

Application des correctifs aux ressources

```
# Partially update a node
kubectl patch node k8s-node-1 -p '{"spec":{"unschedulable":true}}'

# Update a container's image; spec.containers[*].name is required because it's a merge key
kubectl patch pod valid-pod -p '{"spec":{"containers":[{"name":"kubernetes-serve-hostname","image":"new image"}]}}'

# Update a container's image using a json patch with positional arrays
kubectl patch pod valid-pod --type='json' -p='[{"op": "replace", "path": "/spec/containers/0/image", "value":"new i

# Disable a deployment livenessProbe using a json patch with positional arrays
kubectl patch deployment valid-deployment  --type json   -p='[{"op": "remove", "path": "/spec/template/spec/contain

# Add a new element to a positional array
kubectl patch sa default --type='json' -p='[{"op": "add", "path": "/secrets/1", "value": {"name": "whatever" } }]'

# Update a deployment's replica count by patching its scale subresource
kubectl patch deployment nginx-deployment --subresource='scale' --type='merge' -p '{"spec":{"replicas":2}}'
```

Modification des ressources

Modifiez toutes ressources API de votre choix avec votre éditeur préféré

```
kubectl edit svc/docker-registry                      # Edit the service named docker-registry
KUBE_EDITOR="nano" kubectl edit svc/docker-registry   # Use an alternative editor
```

Mise à l'échelle des ressources

```
kubectl scale --replicas=3 rs/foo                                 # Scale a replicaset named 'foo' to 3
kubectl scale --replicas=3 -f foo.yaml                            # Scale a resource specified in "foo.yaml" to 3
kubectl scale --current-replicas=2 --replicas=3 deployment/mysql  # If the deployment named mysql's current size is
kubectl scale --replicas=5 rc/foo rc/bar rc/baz                   # Scale multiple replication controllers
```

Suppression des ressources

```
kubectl delete -f ./pod.json                                      # Delete a pod using the type and name specified
kubectl delete pod unwanted --now                                 # Delete a pod with no grace period
kubectl delete pod,service baz foo                                # Delete pods and services with same names "baz"
kubectl delete pods,services -l name=myLabel                      # Delete pods and services with label name=myLabe
kubectl -n my-ns delete pod,svc --all                             # Delete all pods and services in namespace my-ns
# Delete all pods matching the awk pattern1 or pattern2
kubectl get pods  -n mynamespace --no-headers=true | awk '/pattern1|pattern2/{print $1}' | xargs  kubectl delete -n
```

Interaction avec les Pods en cours d'exécution

```
kubectl logs my-pod                                 # dump pod logs (stdout)
kubectl logs -l name=myLabel                        # dump pod logs, with label name=myLabel (stdout)
kubectl logs my-pod --previous                      # dump pod logs (stdout) for a previous instantiation of a conta
kubectl logs my-pod -c my-container                 # dump pod container logs (stdout, multi-container case)
kubectl logs -l name=myLabel -c my-container        # dump pod container logs, with label name=myLabel (stdout)
kubectl logs my-pod -c my-container --previous      # dump pod container logs (stdout, multi-container case) for a p
kubectl logs -f my-pod                              # stream pod logs (stdout)
kubectl logs -f my-pod -c my-container              # stream pod container logs (stdout, multi-container case)
kubectl logs -f -l name=myLabel --all-containers    # stream all pods logs with label name=myLabel (stdout)
kubectl run -i --tty busybox --image=busybox:1.28 -- sh  # Run pod as interactive shell
kubectl run nginx --image=nginx -n mynamespace      # Start a single instance of nginx pod in the namespace of mynam
kubectl run nginx --image=nginx --dry-run=client -o yaml > pod.yaml
                                                    # Generate spec for running pod nginx and write it into a file c
kubectl attach my-pod -i                            # Attach to Running Container
kubectl port-forward my-pod 5000:6000               # Listen on port 5000 on the local machine and forward to port 6
kubectl exec my-pod -- ls /                         # Run command in existing pod (1 container case)
kubectl exec --stdin --tty my-pod -- /bin/sh        # Interactive shell access to a running pod (1 container case)
kubectl exec my-pod -c my-container -- ls /         # Run command in existing pod (multi-container case)
kubectl debug my-pod -it --image=busybox:1.28       # Create an interactive debugging session within existing pod an
kubectl debug node/my-node -it --image=busybox:1.28 # Create an interactive debugging session on a node and immedia
kubectl top pod                                     # Show metrics for all pods in the default namespace
kubectl top pod POD_NAME --containers               # Show metrics for a given pod and its containers
kubectl top pod POD_NAME --sort-by=cpu              # Show metrics for a given pod and sort it by 'cpu' or 'memory'
```

Copie de fichiers et de répertoires vers et depuis des conteneurs

```
kubectl cp /tmp/foo_dir my-pod:/tmp/bar_dir            # Copy /tmp/foo_dir local directory to /tmp/bar_dir in a remo
kubectl cp /tmp/foo my-pod:/tmp/bar -c my-container    # Copy /tmp/foo local file to /tmp/bar in a remote pod in a s
kubectl cp /tmp/foo my-namespace/my-pod:/tmp/bar       # Copy /tmp/foo local file to /tmp/bar in a remote pod in nam
kubectl cp my-namespace/my-pod:/tmp/foo /tmp/bar       # Copy /tmp/foo from a remote pod to /tmp/bar locally
```

> **Note:**
> `kubectl cp` nécessite que le binaire 'tar' soit présent dans l'image de votre conteneur. Si 'tar' n'est pas disponible, `kubectl cp` échouera. Pour les cas d'utilisation plus avancés, comme les liens symboliques, widlcard ou la préservation des modes d'accès aux fichiers, envisagez d'utiliser `kubectl exec`

```
tar cf - /tmp/foo | kubectl exec -i -n my-namespace my-pod -- tar xf - -C /tmp/bar           # Copy /tmp/foo local t
kubectl exec -n my-namespace my-pod -- tar cf - /tmp/foo | tar xf - -C /tmp/bar    # Copy /tmp/foo from a remote pod
```

Interaction avec les Deployments et les Services

```
kubectl logs deploy/my-deployment                          # dump Pod logs for a Deployment (single-container case)
kubectl logs deploy/my-deployment -c my-container          # dump Pod logs for a Deployment (multi-container case)

kubectl port-forward svc/my-service 5000                   # listen on local port 5000 and forward to port 5000 on Se
kubectl port-forward svc/my-service 5000:my-service-port   # listen on local port 5000 and forward to Service target

kubectl port-forward deploy/my-deployment 5000:6000        # listen on local port 5000 and forward to port 6000 on a
kubectl exec deploy/my-deployment -- ls                    # run command in first Pod and first container in Deploym
```

Interaction avec les Nodes et le cluster

```
kubectl cordon my-node                                                # Mark my-node as unschedulable
kubectl drain my-node                                                 # Drain my-node in preparation for maintenanc
kubectl uncordon my-node                                              # Mark my-node as schedulable
kubectl top node                                                      # Show metrics for all nodes
kubectl top node my-node                                              # Show metrics for a given node
kubectl cluster-info                                                  # Display addresses of the master and service
kubectl cluster-info dump                                             # Dump current cluster state to stdout
kubectl cluster-info dump --output-directory=/path/to/cluster-state   # Dump current cluster state to /path/to/clus

# View existing taints on which exist on current nodes.
kubectl get nodes -o='custom-columns=NodeName:.metadata.name,TaintKey:.spec.taints[*].key,TaintValue:.spec.taints[*

# If a taint with that key and effect already exists, its value is replaced as specified.
kubectl taint nodes foo dedicated=special-user:NoSchedule
```

Les types de ressources

Listez tous les types de ressources pris en charge, ainsi que leurs noms abrégés, leurs API group, s'ils sont namespaced, et leur kind:

```
kubectl api-resources
```

Autres opérations pour explorer les ressources API:

```
kubectl api-resources --namespaced=true      # All namespaced resources
kubectl api-resources --namespaced=false     # All non-namespaced resources
kubectl api-resources -o name                # All resources with simple output (only the resource name)
kubectl api-resources -o wide                # All resources with expanded (aka "wide") output
kubectl api-resources --verbs=list,get       # All resources that support the "list" and "get" request verbs
kubectl api-resources --api-group=extensions # All resources in the "extensions" API group
```

Formatage de la sortie

Pour afficher les détails dans votre terminal avec un format spécifique, ajoutez l'option `-o` (ou `--output`) à une commande kubectl prise en charge.

Format de sortie	Description
`-o=custom-columns=<spec>`	Affiche une table en utilisant une liste de colonnes personnalisées séparées par des virgules
`-o=custom-columns-file=<filename>`	Affiche une table en utilisant le modèle de colonnes personnalisées dans le fichier `<filename>`
`-o=go-template=<template>`	Affiche les champs définis dans un golang template
`-o=go-template-file=<filename>`	Affiche les champs définis par le golang template dans le fichier `<filename>`
`-o=json`	Produit un objet API formaté en JSON
`-o=jsonpath=<template>`	Affiche les champs définis dans une expression jsonpath
`-o=jsonpath-file=<filename>`	Affiche les champs définis par l'expression jsonpathdans le fichier `<filename>`

Format de sortie	Description
`-o=name`	Affiche uniquement le nom de la ressource et rien d'autre
`-o=wide`	Produit une sortie en format texte avec des informations supplémentaires, et pour les pods, le nom du noeud est inclus
`-o=yaml`	Produit un objet API formaté en YAML

Exemples en utilisant `-o=custom-columns` :

```
# All images running in a cluster
kubectl get pods -A -o=custom-columns='DATA:spec.containers[*].image'

# All images running in namespace: default, grouped by Pod
kubectl get pods --namespace default --output=custom-columns="NAME:.metadata.name,IMAGE:.spec.containers[*].image"

 # All images excluding "registry.k8s.io/coredns:1.6.2"
kubectl get pods -A -o=custom-columns='DATA:spec.containers[?(@.image!="registry.k8s.io/coredns:1.6.2")].image'

# All fields under metadata regardless of name
kubectl get pods -A -o=custom-columns='DATA:metadata.*'
```

Pour plus d'exemples, consultez la documentation de référence de kubectl .

Verbositée et débogage des sorties de Kubectl

La verbosité de Kubectl est contrôlée avec les options -v ou --v suivies d'un entier représentant le niveau de log. Les conventions générales de journalisation de Kubernetes et les niveaux de log associés sont décrits ici.

Verbosity	Description
`--v=0`	Généralement utile pour que cela soit *toujours* visible pour un opérateur de cluster.
`--v=1`	Un niveau de log par défaut raisonnable si vous ne souhaitez pas de verbosité.
`--v=2`	Informations utiles sur l'état stable du service et messages de log importants pouvant correspondre à des changements significatifs dans le système. C'est le niveau de log par défaut recommandé pour la plupart des systèmes.
`--v=3`	Informations supplémentaires sur les changements.
`--v=4`	Verbosité de niveau débogage.
`--v=5`	Verbosité de niveau trace.
`--v=6`	Affiche les ressources demandées.
`--v=7`	Affiche les en-têtes des requêtes HTTP.
`--v=8`	Affiche le contenu des requêtes HTTP.
`--v=9`	Affiche le contenu des requêtes HTTP sans troncature.

- Lire Aperçu de kubectl et apprendre à utiliser JsonPath.

- Voir les options kubectl.
- Et lire Conventions d'utilisation de kubectl pour apprendre à utiliser kubectl dans des scripts réutilisables.
- Voir plus de kubectl cheatsheets de la communauté.

2 - Aperçu de kubectl

kubectl référence

Kubectl est un outil en ligne de commande pour contrôler des clusters Kubernetes. `kubectl` recherche un fichier appelé config dans le répertoire $HOME/.kube. Vous pouvez spécifier d'autres fichiers [kubeconfig](https://kubernetes.io/docs/concepts/configuration/organize-cluster-access-kubeconfig/) en définissant la variable d'environnement KUBECONFIG ou en utilisant le paramètre `--kubeconfig`.

Cet aperçu couvre la syntaxe `kubectl`, décrit les opérations et fournit des exemples classiques. Pour des détails sur chaque commande, incluant toutes les options et sous-commandes autorisées, voir la documentation de référence de kubectl. Pour des instructions d'installation, voir installer kubectl.

Syntaxe

Utilisez la syntaxe suivante pour exécuter des commandes `kubectl` depuis votre fenêtre de terminal :

```
kubectl [commande] [TYPE] [NOM] [flags]
```

où `commande`, `TYPE`, `NOM` et `flags` sont :

- `commande` : Indique l'opération que vous désirez exécuter sur une ou plusieurs ressources, par exemple `create`, `get`, `describe`, `delete`.
- `TYPE` : Indique le type de ressource. Les types de ressources sont insensibles à la casse et vous pouvez utiliser les formes singulier, pluriel ou abrégé. Par exemple, les commandes suivantes produisent le même résultat :

  ```
  ```shell
  $ kubectl get pod pod1
  $ kubectl get pods pod1
  $ kubectl get po pod1
  ```
  ```
- `NOM` : Indique le nom de la ressource. Les noms sont sensibles à la casse. Si le nom est omis, des détails pour toutes les ressources sont affichés, par exemple `$ kubectl get pods`.

  En effectuant une opération sur plusieurs ressources, vous pouvez soit indiquer chaque ressource par leur type et nom soit indiquer un ou plusieurs fichiers :

  - Pour indiquer des ressources par leur type et nom :
    - Pour regrouper des ressources si elles ont toutes le même type : `TYPE1 nom1 nom2 nom<#>`.
      Example: `$ kubectl get pod exemple-pod1 exemple-pod2`
    - Pour indiquer plusieurs types de ressources individuellement : `TYPE1/nom1 TYPE1/nom2 TYPE2/nom3 TYPE<#>/nom<#>`.
      Exemple: `$ kubectl get pod/exemple-pod1 replicationcontroller/exemple-rc1`
  - Pour indiquer des ressources avec un ou plusieurs fichiers : `-f fichier1 -f fichier2 -f fichier<#>`
    - Utilisez YAML plutôt que JSON, YAML tendant à être plus facile à utiliser, particulièrement pour des fichiers de configuration.
      Exemple: `$ kubectl get pod -f ./pod.yaml`
- `flags` : Indique des flags optionnels. Par exemple, vous pouvez utiliser les flags `-s` ou `--server` pour indiquer l'adresse et le port de l'API server Kubernetes.

> **Avertissement:**
> Les flags indiqués en ligne de commande écrasent les valeurs par défaut et les variables d'environnement correspondantes.

Si vous avez besoin d'aide, exécutez `kubectl help` depuis la fenêtre de terminal.

# Opérations

Le tableau suivant inclut une courte description et la syntaxe générale pour chaque opération `kubectl` :

| Opération | Syntaxe | Description |
|---|---|---|
| `alpha` | `kubectl alpha SOUS-COMMANDE [flags]` | Liste les commandes disponibles qui correspondent à des fonctionnalités alpha, qui ne sont pas activées par défaut dans les clusters Kubernetes. |
| `annotate` | `kubectl annotate (-f FICHIER \| TYPE NOM \| TYPE/NOM) CLE_1=VAL_1 ... CLE_N=VAL_N [--overwrite] [--all] [--resource-version=version] [flags]` | Ajoute ou modifie les annotations d'une ou plusieurs ressources. |
| `api-resources` | `kubectl api-resources [flags]` | Liste les ressources d'API disponibles. |
| `api-versions` | `kubectl api-versions [flags]` | Liste les versions d'API disponibles. |
| `apply` | `kubectl apply -f FICHIER [flags]` | Applique un changement de configuration à une ressource depuis un fichier ou stdin. |
| `attach` | `kubectl attach POD -c CONTENEUR [-i] [-t] [flags]` | Attache à un conteneur en cours d'exécution soit pour voir la sortie standard soit pour interagir avec le conteneur (stdin). |
| `auth` | `kubectl auth [flags] [options]` | Inspecte les autorisations. |
| `autoscale` | `kubectl autoscale (-f FICHIER \| TYPE NOM \| TYPE/NOM) [--min=MINPODS] --max=MAXPODS [--cpu-percent=CPU] [flags]` | Scale automatiquement l'ensemble des pods gérés par un replication controller. |
| `certificate` | `kubectl certificate SOUS-COMMANDE [options]` | Modifie les ressources de type certificat. |
| `cluster-info` | `kubectl cluster-info [flags]` | Affiche les informations des endpoints du master et des services du cluster. |
| `completion` | `kubectl completion SHELL [options]` | Affiche le code de complétion pour le shell spécifié (bash ou zsh). |
| `config` | `kubectl config SOUS-COMMANDE [flags]` | Modifie les fichiers kubeconfig. Voir les sous-commandes individuelles pour plus de détails. |
| `convert` | `kubectl convert -f FICHIER [options]` | Convertit des fichiers de configuration entre différentes versions d'API. Les formats YAML et JSON sont acceptés. |
| `cordon` | `kubectl cordon NOEUD [options]` | Marque un nœud comme non programmable. |
| `cp` | `kubectl cp <ficher-src> <fichier-dest> [options]` | Copie des fichiers et des répertoires vers et depuis des conteneurs. |

| Opération | Syntaxe | Description |
|---|---|---|
| create | `kubectl create -f FICHIER [flags]` | Crée une ou plusieurs ressources depuis un fichier ou stdin. |
| delete | `kubectl delete (-f FICHIER \| TYPE [NOM \| /NOM \| -l label \| --all]) [flags]` | 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. |
| describe | `kubectl describe (-f FICHIER \| TYPE [PREFIXE_NOM \| /NOM \| -l label]) [flags]` | Affiche l'état détaillé d'une ou plusieurs ressources. |
| diff | `kubectl diff -f FICHIER [flags]` | Diff un fichier ou stdin par rapport à la configuration en cours |
| drain | `kubectl drain NOEUD [options]` | Vide un nœud en préparation de sa mise en maintenance. |
| edit | `kubectl edit (-f FICHIER \| TYPE NOM \| TYPE/NOM) [flags]` | Édite et met à jour la définition d'une ou plusieurs ressources sur le serveur en utilisant l'éditeur par défaut. |
| exec | `kubectl exec POD [-c CONTENEUR] [-i] [-t] [flags] [-- COMMANDE [args...]]` | Exécute une commande à l'intérieur d'un conteneur dans un pod. |
| explain | `kubectl explain [--recursive=false] [flags]` | Obtient des informations sur différentes ressources. Par exemple pods, nœuds, services, etc. |
| expose | `kubectl expose (-f FICHIER \| TYPE NOM \| TYPE/NOM) [--port=port] [--protocol=TCP\|UDP] [--target-port=nombre-ou-nom] [--name=nom] [--external-ip=ip-externe-ou-service] [--type=type] [flags]` | Expose un replication controller, service ou pod comme un nouveau service Kubernetes. |
| get | `kubectl get (-f FICHIER \| TYPE [NOM \| /NOM \| -l label]) [--watch] [--sort-by=CHAMP] [[-o \| --output]=FORMAT_AFFICHAGE] [flags]` | Liste une ou plusieurs ressources. |
| kustomize | `kubectl kustomize <répertoire> [flags] [options]` | Liste un ensemble de ressources d'API généré à partir d'instructions d'un fichier kustomization.yaml. Le paramètre doit être le chemin d'un répertoire contenant ce fichier, ou l'URL d'un dépôt git incluant un suffixe de chemin par rapport à la racine du dépôt. |
| label | `kubectl label (-f FICHIER \| TYPE NOM \| TYPE/NOM) CLE_1=VAL_1 ... CLE_N=VAL_N [--overwrite] [--all] [--resource-version=version] [flags]` | Ajoute ou met à jour les labels d'une ou plusieurs ressources. |
| logs | `kubectl logs POD [-c CONTENEUR] [--follow] [flags]` | Affiche les logs d'un conteneur dans un pod. |
| options | `kubectl options` | Liste des options globales, s'appliquant à toutes commandes. |

| Opération | Syntaxe | Description |
|---|---|---|
| patch | `kubectl patch (-f FICHIER \| TYPE NOM \| TYPE/NOM) --patch PATCH [flags]` | Met à jour un ou plusieurs champs d'une resource en utilisant le processus de merge patch stratégique. |
| plugin | `kubectl plugin [flags] [options]` | Fournit des utilitaires pour interagir avec des plugins. |
| port-forward | `kubectl port-forward POD [PORT_LOCAL:]PORT_DISTANT [... [PORT_LOCAL_N:]PORT_DISTANT_N] [flags]` | Transfère un ou plusieurs ports locaux vers un pod. |
| proxy | `kubectl proxy [--port=PORT] [--www=static-dir] [--www-prefix=prefix] [--api-prefix=prefix] [flags]` | Exécute un proxy vers un API server Kubernetes. |
| replace | `kubectl replace -f FICHIER` | Remplace une ressource depuis un fichier ou stdin. |
| rollout | `kubectl rollout SOUS-COMMANDE [options]` | Gère le rollout d'une ressource. Les types de ressources valides sont : deployments, daemonsets et statefulsets. |
| run | `kubectl run NOM --image=image [--env="cle=valeur"] [--port=port] [--replicas=replicas] [--dry-run=server&#124;client&#124;none] [--overrides=inline-json] [flags]` | Exécute dans le cluster l'image indiquée. |
| scale | `kubectl scale (-f FICHIER \| TYPE NOM \| TYPE/NOM) --replicas=QUANTITE [--resource-version=version] [--current-replicas=quantité] [flags]` | Met à jour la taille du replication controller indiqué. |
| set | `kubectl set SOUS-COMMANDE [options]` | Configure les ressources de l'application. |
| taint | `kubectl taint NOEUD NNOM CLE_1=VAL_1:EFFET_TAINT_1 ... CLE_N=VAL_N:EFFET_TAINT_N [options]` | Met à jour les marques (taints) d'un ou plusieurs nœuds. |
| top | `kubectl top [flags] [options]` | Affiche l'utilisation des ressources (CPU/Mémoire/Stockage). |
| uncordon | `kubectl uncordon NOEUD [options]` | Marque un noeud comme programmable. |
| version | `kubectl version [--client] [flags]` | Affiche la version de Kubernetes du serveur et du client. |
| wait | `kubectl wait ([-f FICHIER] \| ressource.groupe/ressource.nom \| ressource.groupe [(-l label \| --all)]) [--for=delete\|--for condition=available] [options]` | Expérimental : Attend un condition spécifique sur une ou plusieurs ressources. |

Rappelez-vous : Pour tout savoir sur les opérations, voir la documentation de référence de kubectl.

# Types de ressources

Le tableau suivant inclut la liste de tous les types de ressources pris en charge et leurs alias abrégés.

(cette sortie peut être obtenue depuis `kubectl api-resources`, et correspond à Kubernetes 1.13.3.)

| Nom de la ressource | Noms abrégés | Groupe API | Par namespace | Genre de la ressource |
|---|---|---|---|---|
| `bindings` | | | true | Binding |
| `componentstatuses` | `cs` | | false | ComponentStatus |
| `configmaps` | `cm` | | true | ConfigMap |
| `endpoints` | `ep` | | true | Endpoints |
| `limitranges` | `limits` | | true | LimitRange |
| `namespaces` | `ns` | | false | Namespace |
| `nodes` | `no` | | false | Node |
| `persistentvolumeclaims` | `pvc` | | true | PersistentVolumeClaim |
| `persistentvolumes` | `pv` | | false | PersistentVolume |
| `pods` | `po` | | true | Pod |
| `podtemplates` | | | true | PodTemplate |
| `replicationcontrollers` | `rc` | | true | ReplicationController |
| `resourcequotas` | `quota` | | true | ResourceQuota |
| `secrets` | | | true | Secret |
| `serviceaccounts` | `sa` | | true | ServiceAccount |
| `services` | `svc` | | true | Service |
| `mutatingwebhookconfigurations` | | admissionregistration.k8s.io | false | MutatingWebhookConfigur |
| `validatingwebhookconfigurations` | | admissionregistration.k8s.io | false | ValidatingWebhookConfigu |
| `customresourcedefinitions` | `crd`, `crds` | apiextensions.k8s.io | false | CustomResourceDefinition |
| `apiservices` | | apiregistration.k8s.io | false | APIService |
| `controllerrevisions` | | apps | true | ControllerRevision |
| `daemonsets` | `ds` | apps | true | DaemonSet |
| `deployments` | `deploy` | apps | true | Deployment |
| `replicasets` | `rs` | apps | true | ReplicaSet |
| `statefulsets` | `sts` | apps | true | StatefulSet |

| Nom de la ressource | Noms abrégés | Groupe API | Par namespace | Genre de la ressource |
|---|---|---|---|---|
| tokenreviews | | authentication.k8s.io | false | TokenReview |
| localsubjectaccessreviews | | authorization.k8s.io | true | LocalSubjectAccessReview |
| selfsubjectaccessreviews | | authorization.k8s.io | false | SelfSubjectAccessReview |
| selfsubjectrulesreviews | | authorization.k8s.io | false | SelfSubjectRulesReview |
| subjectaccessreviews | | authorization.k8s.io | false | SubjectAccessReview |
| horizontalpodautoscalers | hpa | autoscaling | true | HorizontalPodAutoscaler |
| cronjobs | cj | batch | true | CronJob |
| jobs | | batch | true | Job |
| certificatesigningrequests | csr | certificates.k8s.io | false | CertificateSigningRequest |
| leases | | coordination.k8s.io | true | Lease |
| events | ev | events.k8s.io | true | Event |
| ingresses | ing | extensions | true | Ingress |
| networkpolicies | netpol | networking.k8s.io | true | NetworkPolicy |
| poddisruptionbudgets | pdb | policy | true | PodDisruptionBudget |
| podsecuritypolicies | psp | policy | false | PodSecurityPolicy |
| clusterrolebindings | | rbac.authorization.k8s.io | false | ClusterRoleBinding |
| clusterroles | | rbac.authorization.k8s.io | false | ClusterRole |
| rolebindings | | rbac.authorization.k8s.io | true | RoleBinding |
| roles | | rbac.authorization.k8s.io | true | Role |
| priorityclasses | pc | scheduling.k8s.io | false | PriorityClass |
| csidrivers | | storage.k8s.io | false | CSIDriver |
| csinodes | | storage.k8s.io | false | CSINode |
| storageclasses | sc | storage.k8s.io | false | StorageClass |
| volumeattachments | | storage.k8s.io | false | VolumeAttachment |

## Options de sortie

Utilisez les sections suivantes pour savoir comment vous pouvez formater ou ordonner les sorties de certaines commandes. Pour savoir exactement quelles commandes prennent en charge quelles options de sortie, voir la documentation de référence de kubectl.

# Formater la sortie

Le format de sortie par défaut pour toutes les commandes `kubectl` est le format texte lisible par l'utilisateur. Pour afficher des détails dans votre fenêtre de terminal dans un format spécifique, vous pouvez ajouter une des options `-o` ou `--output` à une des commandes `kubectl` les prenant en charge.

## Syntaxe

```
kubectl [commande] [TYPE] [NOM] -o <format_sortie>
```

Selon l'opération `kubectl`, les formats de sortie suivants sont pris en charge :

Format de sortie	Description
`-o custom-columns=<spec>`	Affiche un tableau en utilisant une liste de colonnes personnalisées séparées par des virgules.
`-o custom-columns-file=<fichier>`	Affiche un tableau en utilisant un modèle de colonnes personnalisées dans le fichier `<fichier>`.
`-o json`	Affiche un objet de l'API formaté en JSON.
`-o jsonpath=<modèle>`	Affiche les champs définis par une expression jsonpath.
`-o jsonpath-file=<ffichier>`	Affiche les champs définis par une expression jsonpath dans le fichier `<fichier>`.
`-o name`	Affiche uniquement le nom de la ressource et rien de plus.
`-o wide`	Affiche dans le format texte avec toute information supplémentaire. Pour les pods, le nom du nœud est inclus.
`-o yaml`	Affiche un objet de l'API formaté en YAML.

Exemple

Dans cet exemple, la commande suivante affiche les détails d'un unique pod sous forme d'un objet formaté en YAML :

```
$ kubectl get pod web-pod-13je7 -o yaml
```

Souvenez-vous : Voir la documentation de référence de kubectl pour voir quels formats de sortie sont pris en charge par chaque commande.

Colonnes personnalisées

Pour définir des colonnes personnalisées et afficher uniquement les détails voulus dans un tableau, vous pouvez utiliser l'option `custom-columns`. Vous pouvez choisir de définir les colonnes personnalisées soit en ligne soit dans un fichier modèle : `-o custom-columns=<spec>` ou `-o custom-columns-file=<fichier>`.

Exemples

En ligne :

```
$ kubectl get pods <nom-pod> -o custom-columns=NOM:.metadata.name,RSRC:.metadata.resourceVersion
```

Fichier modèle :

```
$ kubectl get pods <nom-pod> -o custom-columns-file=modele.txt
```

où le fichier `modele.txt` contient :

```
NOM           RSRC
metadata.name metadata.resourceVersion
```

Le résultat de ces commandes est :

```
NOM            RSRC
submit-queue   610995
```

Colonnes côté serveur

`kubectl` est capable de recevoir des informations de colonnes spécifiques d'objets depuis le serveur. Cela veut dire que pour toute ressource donnée, le serveur va retourner les colonnes et lignes pour cette ressource, que le client pourra afficher. Cela permet un affichage de sortie lisible par l'utilisateur cohérent entre les clients utilisés sur le même cluster, le serveur encapsulant les détails d'affichage.

Cette fonctionnalité est activée par défaut dans `kubectl` version 1.11 et suivantes. Pour la désactiver, ajoutez l'option `--server-print=false` à la commande `kubectl get`.

Exemples

Pour afficher les informations sur le status d'un pod, utilisez une commande similaire à :

```
kubectl get pods <nom-pod> --server-print=false
```

La sortie ressemble à :

```
NAME       AGE
nom-pod    1m
```

Ordonner les listes d'objets

Pour afficher les objets dans une liste ordonnée dans une fenêtre de terminal, vous pouvez ajouter l'option `--sort-by` à une commande `kubectl` qui la prend en charge. Ordonnez vos objets en spécifiant n'importe quel champ numérique ou textuel avec l'option `--sort-by`. Pour spécifier un champ, utilisez une expression jsonpath.

Syntaxe

```
kubectl [commande] [TYPE] [NOM] --sort-by=<exp_jsonpath>
```

Exemple

Pour afficher une liste de pods ordonnés par nom, exécutez :

```
$ kubectl get pods --sort-by=.metadata.name
```

Exemples : Opérations courantes

Utilisez les exemples suivants pour vous familiariser avec les opérations de `kubectl` fréquemment utilisées :

`kubectl apply` - Créer une ressource depuis un fichier ou stdin.

```
# Crée un service en utilisant la définition dans exemple-service.yaml.
$ kubectl apply -f exemple-service.yaml

# Crée un replication controller en utilisant la définition dans exemple-controller.yaml.
$ kubectl apply -f exemple-controller.yaml

# Crée les objets qui sont définis dans les fichiers .yaml, .yml ou .json du répertoire <répertoire>.
$ kubectl apply -f <répertoire>
```

`kubectl get` - Liste une ou plusieurs ressources.

```
# Liste tous les pods dans le format de sortie texte.
$ kubectl get pods

# Liste tous les pods dans le format de sortie texte et inclut des informations additionnelles (comme le nom du nœu
$ kubectl get pods -o wide

# Liste le replication controller ayant le nom donné dans le format de sortie texte.
# Astuce : Vous pouvez raccourcir et remplacer le type de ressource 'replicationcontroller' avec l'alias 'rc'.
$ kubectl get replicationcontroller <nom-rc>

# Liste ensemble tous les replication controller et les services dans le format de sortie texte.
$ kubectl get rc,services

# Liste tous les daemon sets dans le format de sortie texte.
kubectl get ds

# Liste tous les pods s'exécutant sur le nœud serveur01
$ kubectl get pods --field-selector=spec.nodeName=serveur01
```

`kubectl describe` - Affiche l'état détaillé d'une ou plusieurs ressources, en incluant par défaut les ressources non initialisées.

```
# Affiche les détails du nœud ayant le nom <nom-nœud>.
$ kubectl describe nodes <nom-nœud>

# Affiche les détails du pod ayant le nom <nom-pod>.
$ kubectl describe pods/<nom-pod>

# Affiche les détails de tous les pods gérés par le replication controller dont le nom est <nom-rc>.
# Rappelez-vous : les noms des pods étant créés par un replication controller sont préfixés par le nom du replicati
$ kubectl describe pods <nom-rc>

# Décrit tous les pods
$ kubectl describe pods
```

> **Note:**
> La commande `kubectl get` est habituellement utilisée pour afficher une ou plusieurs ressources d'un même type. Elle propose un ensemble complet d'options permettant de personnaliser le format de sortie avec les options `-o` ou `--output`, par exemple. Vous pouvez utiliser les options `-w` ou `--watch` pour initier l'écoute des modifications d'un objet particulier. La commande `kubectl describe` est elle plutôt utilisée pour décrire les divers aspects d'une ressource voulue. Elle peut invoquer plusieurs appels d'API à l'API server pour construire une vue complète pour l'utilisateur. Par exemple, la commande `kubectl`

> `describe node` retourne non seulement les informations sur les nœuds, mais aussi un résumé des pods s'exécutant dessus, les événements générés pour chaque nœud, etc.nœud

`kubectl delete` - Supprime des ressources soit depuis un fichier, stdin, ou en spécifiant des sélecteurs de labels, des noms, des sélecteurs de ressource ou des ressources.

```
# Supprime un pod en utilisant le type et le nom spécifiés dans le fichier pod.yaml.
$ kubectl delete -f pod.yaml

# Supprime tous les pods et services ayant le label <clé-label>=<valeur-label>
$ kubectl delete pods,services -l <clé-label>=<valeur-label>

# Supprime tous les pods, en incluant les non initialisés.
$ kubectl delete pods --all
```

`kubectl exec` - Exécute une commande depuis un conteneur d'un pod.

```
# Affiche la sortie de la commande 'date' depuis le pod <nom-pod>. Par défaut, la sortie se fait depuis le premier c
$ kubectl exec <nom-pod> -- date

# Affiche la sortie de la commande 'date' depuis le conteneur <nom-conteneur> du pod <nom-pod>.
$ kubectl exec <nom-pod> -c <nom-conteneur> -- date

# Obtient un TTY interactif et exécute /bin/bash depuis le pod <nom-pod>. Par défaut, la sortie se fait depuis le pi
$ kubectl exec -ti <nom-pod> -- /bin/bash
```

`kubectl logs` - Affiche les logs d'un conteneur dans un pod.

```
# Retourne un instantané des logs du pod <nom-pod>.
$ kubectl logs <nom-pod>

# Commence à streamer les logs du pod <nom-pod>. Ceci est similaire à la commande Linux 'tail -f'.
$ kubectl logs -f <nom-pod>
```

`kubectl diff` - Affiche un diff des mises à jour proposées au cluster.

```
# Diff les ressources présentes dans "pod.json".
kubectl diff -f pod.json

# Diff les ressources présentes dans le fichier lu sur l'entrée standard.
cat service.yaml | kubectl diff -f -
```

Exemples : Créer et utiliser des plugins

Utilisez les exemples suivants pour vous familiariser avec l'écriture et l'utilisation de plugins `kubectl` :

```
# créez un plugin simple dans n'importe quel langage et nommez
# l'exécutable de telle sorte qu'il commence par "kubectl-"
$ cat ./kubectl-hello
#!/bin/bash

# ce plugin affiche les mots "hello world"
echo "hello world"

# une fois votre plugin écrit, rendez-le exécutable
$ sudo chmod +x ./kubectl-hello

# et déplacez-le dans un répertoire de votre PATH
$ sudo mv ./kubectl-hello /usr/local/bin

# vous avez maintenant créé et "installé" un plugin kubectl.
# vous pouvez commencer à l'utiliser en l'invoquant depuis kubectl
# comme s'il s'agissait d'une commande ordinaire
$ kubectl hello
hello world

# vous pouvez "désinstaller" un plugin,
# simplement en le supprimant de votre PATH
$ sudo rm /usr/local/bin/kubectl-hello
```

Pour voir tous les plugins disponibles pour `kubectl`, vous pouvez utiliser la sous-commande `kubectl plugin list` :

```
$ kubectl plugin list
The following kubectl-compatible plugins are available:

/usr/local/bin/kubectl-hello
/usr/local/bin/kubectl-foo
/usr/local/bin/kubectl-bar

# cette commande peut aussi vous avertir de plugins qui ne sont pas exécutables,
# ou qui sont cachés par d'autres plugins, par exemple :
$ sudo chmod -x /usr/local/bin/kubectl-foo
$ kubectl plugin list
The following kubectl-compatible plugins are available:

/usr/local/bin/kubectl-hello
/usr/local/bin/kubectl-foo
  - warning: /usr/local/bin/kubectl-foo identified as a plugin, but it is not executable
/usr/local/bin/kubectl-bar

error: one plugin warning was found
```

Vous pouvez voir les plugins comme un moyen de construire des fonctionnalités plus complexes au dessus des commandes kubectl existantes :

```
$ cat ./kubectl-whoami
#!/bin/bash

# ce plugin utilise la commande `kubectl config` pour afficher
# l'information sur l'utilisateur courant, en se basant sur
# le contexte couramment sélectionné
kubectl config view --template='{{ range .contexts }}{{ if eq .name "'$(kubectl config current-context)'" }}Current
```

Exécuter le plugin ci-dessus vous donne une sortie contenant l'utilisateur du contexte couramment sélectionné dans votre fichier KUBECONFIG :

```
# rendre le fichier exécutable executable
$ sudo chmod +x ./kubectl-whoami

# et le déplacer dans le PATH
$ sudo mv ./kubectl-whoami /usr/local/bin

$ kubectl whoami
Current user: plugins-user
```

Pour en savoir plus sur les plugins, examinez l'exemple de plugin CLI.

A suivre

Commencez à utiliser les commandes kubectl.

3 - Support de JSONPath

JSONPath kubectl Kubernetes

Kubectl prend en charge les modèles JSONPath.

Un modèle JSONPath est composé d'expressions JSONPath entourées par des accolades {}. Kubectl utilise les expressions JSONPath pour filtrer sur des champs spécifiques de l'objet JSON et formater la sortie. En plus de la syntaxe de modèle JSONPath originale, les fonctions et syntaxes suivantes sont valides :

1. Utilisez des guillemets doubles pour marquer du texte dans les expressions JSONPath.
2. Utilisez les opérateurs `range` et `end` pour itérer sur des listes.
3. Utilisez des indices négatifs pour parcourir une liste à reculons. Les indices négatifs ne "bouclent pas" sur une liste et sont valides tant que `-index + longeurListe >= 0`.

> **Note:**
> - L'opérateur `$` est optionnel, l'expression commençant toujours, par défaut, à la racine de l'objet.
> - L'objet résultant est affiché via sa fonction String().

Étant donné l'entrée JSON :

```
{
  "kind": "List",
  "items":[
    {
      "kind":"None",
      "metadata":{"name":"127.0.0.1"},
      "status":{
        "capacity":{"cpu":"4"},
        "addresses":[{"type": "LegacyHostIP", "address":"127.0.0.1"}]
      }
    },
    {
      "kind":"None",
      "metadata":{"name":"127.0.0.2"},
      "status":{
        "capacity":{"cpu":"8"},
        "addresses":[
          {"type": "LegacyHostIP", "address":"127.0.0.2"},
          {"type": "another", "address":"127.0.0.3"}
        ]
      }
    }
  ],
  "users":[
    {
      "name": "myself",
      "user": {}
    },
    {
      "name": "e2e",
      "user": {"username": "admin", "password": "secret"}
    }
  ]
}
```

Fonction	Description	Exemple	Résultat
`text`	le texte en clair	`le type est {.kind}`	`le type est List`

Fonction	Description	Exemple	Résultat
`@`	l'objet courant	`{@}`	identique à l'entrée
`. ou []`	opérateur fils	`{.kind}, {['kind']} ou {['name\.type']}`	`List`
`..`	descente récursive	`{..name}`	`127.0.0.1 127.0.0.2 myself e2e`
`*`	joker. Tous les objets	`{.items[*].metadata.name}`	`[127.0.0.1 127.0.0.2]`
`[start:end:step]`	opérateur d'indice	`{.users[0].name}`	`myself`
`[,]`	opérateur d'union	`{.items[*]['metadata.name', 'status.capacity']}`	`127.0.0.1 127.0.0.2 map[cpu:4] map[cpu:8]`
`?()`	filtre	`{.users[?(@.name=="e2e")].user.password}`	`secret`
`range, end`	itération de liste	`{range .items[*]}[{.metadata.name}, {.status.capacity}] {end}`	`[127.0.0.1, map[cpu:4]] [127.0.0.2, map[cpu:8]]`
`''`	protège chaîne interprétée	`{range .items[*]}{.metadata.name}{'\t'}{end}`	`127.0.0.1 127.0.0.2`

Exemples utilisant `kubectl` et des expressions JSONPath :

```
kubectl get pods -o json
kubectl get pods -o=jsonpath='{@}'
kubectl get pods -o=jsonpath='{.items[0]}'
kubectl get pods -o=jsonpath='{.items[0].metadata.name}'
kubectl get pods -o=jsonpath="{.items[*]['metadata.name', 'status.capacity']}"
kubectl get pods -o=jsonpath='{range .items[*]}{.metadata.name}{"\t"}{.status.startTime}{"\n"}{end}'
```

Note:
Sous Windows, vous devez utiliser des guillemets *doubles* autour des modèles JSONPath qui contiennent des espaces (et non des guillemets simples comme ci-dessus pour bash). Ceci entraîne que vous devez utiliser un guillemet simple ou un double guillemet échappé autour des chaînes litérales dans le modèle. Par exemple :

```
kubectl get pods -o=jsonpath="{range .items[*]}{.metadata.name}{'\t'}{.status.startTime}{'\n'}{end}"
kubectl get pods -o=jsonpath="{range .items[*]}{.metadata.name}{\"\t\"}{.status.startTime}{\"\n\"}{end}"
```

4 - Aide-mémoire kubectl

Cheatsheet kubectl aide-mémoire

Voir aussi : Aperçu Kubectl et Guide JsonPath.

Cette page donne un aperçu de la commande `kubectl`.

Aide-mémoire kubectl

Auto-complétion avec Kubectl

BASH

```
source <(kubectl completion bash) # active l'auto-complétion pour bash dans le shell courant, le paquet bash-comple
echo "source <(kubectl completion bash)" >> ~/.bashrc # ajoute l'auto-complétion de manière permanente à votre shel
```

Vous pouvez de plus déclarer un alias pour `kubectl` qui fonctionne aussi avec l'auto-complétion :

```
alias k=kubectl
complete -o default -F __start_kubectl k
```

ZSH

```
source <(kubectl completion zsh)  # active l'auto-complétion pour zsh dans le shell courant
echo "[[ $commands[kubectl] ]] && source <(kubectl completion zsh)" >> ~/.zshrc # ajoute l'auto-complétion de manièr
```

Contexte et configuration de Kubectl

Indique avec quel cluster Kubernetes `kubectl` communique et modifie les informations de configuration. Voir la documentation Authentification multi-clusters avec kubeconfig pour des informations détaillées sur le fichier de configuration. Information. Voir la documentation Authentification à travers des clusters avec kubeconfig pour des informations détaillées sur le fichier de configuration.

```
kubectl config view # Affiche les paramètres fusionnés de kubeconfig

# Utilise plusieurs fichiers kubeconfig en même temps et affiche la configuration fusionnée
KUBECONFIG=~/.kube/config:~/.kube/kubconfig2

kubectl config view

# Affiche le mot de passe pour l'utilisateur e2e
kubectl config view -o jsonpath='{.users[?(@.name == "e2e")].user.password}'

kubectl config view -o jsonpath='{.users[].name}'  # Affiche le premier utilisateur
kubectl config view -o jsonpath='{.users[*].name}' # Affiche une liste d'utilisateurs
kubectl config get-contexts                        # Affiche la liste des contextes
kubectl config current-context                     # Affiche le contexte courant (current-context)
kubectl config use-context my-cluster-name         # Définit my-cluster-name comme contexte courant

# Ajoute un nouveau cluster à votre kubeconf, prenant en charge l'authentification de base (basic auth)
kubectl config set-credentials kubeuser/foo.kubernetes.com --username=kubeuser --password=kubepassword

# Enregistre de manière permanente le namespace pour toutes les commandes kubectl suivantes dans ce contexte
kubectl config set-context --current --namespace=ggckad-s2

# Définit et utilise un contexte qui utilise un nom d'utilisateur et un namespace spécifiques
kubectl config set-context gce --user=cluster-admin --namespace=foo \
  && kubectl config use-context gce

kubectl config unset users.foo                     # Supprime l'utilisateur foo
```

Apply

`apply` gère des applications en utilisant des fichiers définissant des ressources Kubernetes. Elle crée et met à jour des ressources dans un cluster en exécutant `kubectl apply`. C'est la manière recommandée de gérer des applications Kubernetes en production. Voir le Livre Kubectl.

Création d'objets

Les manifests Kubernetes peuvent être définis en YAML ou JSON. Les extensions de fichier `.yaml`, `.yml`, et `.json` peuvent être utilisés.

```
kubectl apply -f ./my-manifest.yaml            # Crée une ou plusieurs ressources
kubectl apply -f ./my1.yaml -f ./my2.yaml      # Crée depuis plusieurs fichiers
kubectl apply -f ./dir                         # Crée une ou plusieurs ressources depuis tous les manifests dans di
kubectl apply -f https://git.io/vPieo          # Crée une ou plusieurs ressources depuis une url
kubectl create deployment nginx --image=nginx  # Démarre une instance unique de nginx
kubectl explain pods                           # Affiche la documentation pour les manifests pod

# Crée plusieurs objets YAML depuis l'entrée standard (stdin)
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: busybox-sleep
spec:
  containers:
  - name: busybox
    image: busybox
    args:
    - sleep
    - "1000000"
---
apiVersion: v1
kind: Pod
metadata:
  name: busybox-sleep-less
spec:
  containers:
  - name: busybox
    image: busybox
    args:
    - sleep
    - "1000"
EOF

# Crée un Secret contenant plusieurs clés
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  password: $(echo -n "s33msi4" | base64 -w0)
  username: $(echo -n "jane" | base64 -w0)
EOF
```

Visualisation et Recherche de ressources

```
# Commandes Get avec un affichage basique
kubectl get services                          # Liste tous les services d'un namespace
kubectl get pods --all-namespaces             # Liste tous les Pods de tous les namespaces
kubectl get pods -o wide                      # Liste tous les Pods du namespace courant, avec plus de détails
kubectl get deployment my-dep                 # Liste un déploiement particulier
kubectl get pods                              # Liste tous les Pods dans un namespace
kubectl get pod my-pod -o yaml                # Affiche le YAML du Pod

# Commandes Describe avec un affichage verbeux
kubectl describe nodes my-node
kubectl describe pods my-pod

# Liste les services triés par nom
kubectl get services --sort-by=.metadata.name

# Liste les pods classés par nombre de redémarrages
kubectl get pods --sort-by='.status.containerStatuses[0].restartCount'

# Affiche les volumes persistants classés par capacité de stockage
kubectl get pv --sort-by=.spec.capacity.storage

# Affiche la version des labels de tous les pods ayant un label app=cassandra
kubectl get pods --selector=app=cassandra -o \
  jsonpath='{.items[*].metadata.labels.version}'

# Affiche tous les noeuds (en utilisant un sélecteur pour exclure ceux ayant un label
# nommé 'node-role.kubernetes.io/master')
kubectl get node --selector='!node-role.kubernetes.io/master'

# Affiche tous les pods en cours d'exécution (Running) dans le namespace
kubectl get pods --field-selector=status.phase=Running

# Affiche les IPs externes (ExternalIPs) de tous les noeuds
kubectl get nodes -o jsonpath='{.items[*].status.addresses[?(@.type=="ExternalIP")].address}'

# Liste les noms des pods appartenant à un ReplicationController particulier
# "jq" est une commande utile pour des transformations non prises en charge par jsonpath, il est disponible ici : h
sel=${$(kubectl get rc my-rc --output=json | jq -j '.spec.selector | to_entries | .[] | "\(.key)=\(.value),"')%?}
echo $(kubectl get pods --selector=$sel --output=jsonpath={.items..metadata.name})

# Affiche les labels pour tous les pods (ou tout autre objet Kubernetes prenant en charge les labels)
kubectl get pods --show-labels

# Vérifie quels noeuds sont prêts
JSONPATH='{range .items[*]}{@.metadata.name}:{range @.status.conditions[*]}{@.type}={@.status};{end}{end}' \
 && kubectl get nodes -o jsonpath="$JSONPATH" | grep "Ready=True"

# Liste tous les Secrets actuellement utilisés par un pod
kubectl get pods -o json | jq '.items[].spec.containers[].env[]?.valueFrom.secretKeyRef.name' | grep -v null | sort

# Liste les containerIDs des initContainer de tous les Pods
# Utile lors du nettoyage des conteneurs arrêtés, tout en évitant de retirer les initContainers.
kubectl get pods --all-namespaces -o jsonpath='{range .items[*].status.initContainerStatuses[*]}{.containerID}{"\n"}

# Liste les événements (Events) classés par timestamp
kubectl get events --sort-by=.metadata.creationTimestamp

# Compare l'état actuel du cluster à l'état du cluster si le manifeste était appliqué.
kubectl diff -f ./my-manifest.yaml
```

Mise à jour de ressources

Depuis la version 1.11, `rolling-update` a été déprécié (voir CHANGELOG-1.11.md), utilisez plutôt `rollout` .

```
kubectl set image deployment/frontend www=image:v2               # Rolling update du conteneur "www" du déploiement
kubectl rollout history deployment/frontend                      # Vérifie l'historique de déploiements incluant la
kubectl rollout undo deployment/frontend                         # Rollback du déploiement précédent
kubectl rollout undo deployment/frontend --to-revision=2         # Rollback à une version spécifique
kubectl rollout status -w deployment/frontend                    # Écoute (Watch) le status du rolling update du dé
kubectl rollout restart deployment/frontend                      # Rolling restart du déploiement "frontend"


cat pod.json | kubectl replace -f -                              # Remplace un pod, en utilisant un JSON passé en e

# Remplace de manière forcée (Force replace), supprime puis re-crée la ressource. Provoque une interruption de serv
kubectl replace --force -f ./pod.json

# Crée un service pour un nginx repliqué, qui rend le service sur le port 80 et se connecte aux conteneurs sur le p
kubectl expose rc nginx --port=80 --target-port=8000

# Modifie la version (tag) de l'image du conteneur unique du pod à v4
kubectl get pod mypod -o yaml | sed 's/\(image: myimage\):.*$/\1:v4/' | kubectl replace -f -

kubectl label pods my-pod new-label=awesome                      # Ajoute un Label
kubectl annotate pods my-pod icon-url=http://goo.gl/XXBTWq       # Ajoute une annotation
kubectl autoscale deployment foo --min=2 --max=10                # Mise à l'échelle automatique (Auto scale) d'un d
```

Mise à jour partielle de ressources

```
# Mise à jour partielle d'un node
kubectl patch node k8s-node-1 -p '{"spec":{"unschedulable":true}}' # Met à jour partiellement un noeud

# Met à jour l'image d'un conteneur ; spec.containers[*].name est requis car c'est une clé du merge
kubectl patch pod valid-pod -p '{"spec":{"containers":[{"name":"kubernetes-serve-hostname","image":"new image"}]}}'

# Met à jour l'image d'un conteneur en utilisant un patch json avec tableaux indexés
kubectl patch pod valid-pod --type='json' -p='[{"op": "replace", "path": "/spec/containers/0/image", "value":"new i

# Désactive la livenessProbe d'un déploiement en utilisant un patch json avec tableaux indexés
kubectl patch deployment valid-deployment  --type json   -p='[{"op": "remove", "path": "/spec/template/spec/contain

# Ajoute un nouvel élément à un tableau indexé
kubectl patch sa default --type='json' -p='[{"op": "add", "path": "/secrets/1", "value": {"name": "whatever" } }]'
```

Édition de ressources

Édite n'importe quelle ressource de l'API dans un éditeur.

```
kubectl edit svc/docker-registry                      # Édite le service nommé docker-registry
KUBE_EDITOR="nano" kubectl edit svc/docker-registry   # Utilise un autre éditeur
```

Mise à l'échelle de ressources

```
kubectl scale --replicas=3 rs/foo                                 # Scale un replicaset nommé 'foo' à 3
kubectl scale --replicas=3 -f foo.yaml                            # Scale une ressource spécifiée dans foo.yaml" à
kubectl scale --current-replicas=2 --replicas=3 deployment/mysql  # Si la taille du déploiement nommé mysql est act
kubectl scale --replicas=5 rc/foo rc/bar rc/baz                   # Scale plusieurs contrôleurs de réplication
```

Suppression de ressources

```
kubectl delete -f ./pod.json                                              # Supprime un pod en utilisant le type et
kubectl delete pod,service baz foo                                        # Supprime les pods et services ayant les
kubectl delete pods,services -l name=myLabel                              # Supprime les pods et services ayant le
kubectl -n my-ns delete pod,svc --all                                     # Supprime tous les pods et services dans
# Supprime tous les pods correspondants à pattern1 ou pattern2 avec awk
kubectl get pods  -n mynamespace --no-headers=true | awk '/pattern1|pattern2/{print $1}' | xargs  kubectl delete -n
```

Interaction avec des Pods en cours d'exécution

```
kubectl logs my-pod                                 # Affiche les logs du pod (stdout)
kubectl logs -l name=myLabel                        # Affiche les logs des pods ayant le label name=myLabel (stdout
kubectl logs my-pod --previous                      # Affiche les logs du pod (stdout) pour une instance précédente
kubectl logs my-pod -c my-container                 # Affiche les logs d'un conteneur particulier du pod (stdout, ca
kubectl logs -l name=myLabel -c my-container        # Affiche les logs des pods avec le label name=myLabel (stdout,
kubectl logs my-pod -c my-container --previous      # Affiche les logs d'un conteneur particulier du pod (stdout, ca
kubectl logs -f my-pod                              # Fait défiler (stream) les logs du pod (stdout)
kubectl logs -f my-pod -c my-container              # Fait défiler (stream) les logs d'un conteneur particulier du p
kubectl logs -f -l name=myLabel --all-containers    # Fait défiler (stream) les logs de tous les pods ayant le labe
kubectl run -i --tty busybox --image=busybox -- sh  # Exécute un pod comme un shell interactif
kubectl run nginx --image=nginx --restart=Never -n
mynamespace                                         # Exécute le pod nginx dans un namespace spécifique
kubectl run nginx --image=nginx --restart=Never     # Simule l'exécution du pod nginx et écrit sa spécification dans
--dry-run -o yaml > pod.yaml

kubectl attach my-pod -i                            # Attache à un conteneur en cours d'exécution
kubectl port-forward my-pod 5000:6000               # Écoute le port 5000 de la machine locale et forwarde vers le p
kubectl exec my-pod -- ls /                         # Exécute une commande dans un pod existant (cas d'un seul cont
kubectl exec my-pod -c my-container -- ls /         # Exécute une commande dans un pod existant (cas multi-conteneu
kubectl top pod POD_NAME --containers               # Affiche les métriques pour un pod donné et ses conteneurs
```

Interaction avec des Noeuds et Clusters

```
kubectl cordon mon-noeud                                              # Marque mon-noeud comme non assignable (unsc
kubectl drain mon-noeud                                               # Draine mon-noeud en préparation d'une mise
kubectl uncordon mon-noeud                                            # Marque mon-noeud comme assignable
kubectl top node mon-noeud                                            # Affiche les métriques pour un noeud donné
kubectl cluster-info                                                  # Affiche les adresses du master et des servi
kubectl cluster-info dump                                             # Affiche l'état courant du cluster sur stdou
kubectl cluster-info dump --output-directory=/path/to/cluster-state   # Affiche l'état courant du cluster sur /path/

# Si une teinte avec cette clé et cet effet existe déjà, sa valeur est remplacée comme spécifié.
kubectl taint nodes foo dedicated=special-user:NoSchedule
```

Types de ressources

Liste tous les types de ressources pris en charge avec leurs noms courts (shortnames), groupe d'API (API group), si elles sont cantonnées à un namespace (namespaced), et leur Genre (Kind):

```
kubectl api-resources
```

Autres opérations pour explorer les ressources de l'API :

```
kubectl api-resources --namespaced=true      # Toutes les ressources cantonnées à un namespace
kubectl api-resources --namespaced=false     # Toutes les ressources non cantonnées à un namespace
kubectl api-resources -o name                # Toutes les ressources avec un affichage simple (uniquement le nom de
kubectl api-resources -o wide                # Toutes les ressources avec un affichage étendu (alias "wide")
kubectl api-resources --verbs=list,get       # Toutes les ressources prenant en charge les verbes de requête "list"
kubectl api-resources --api-group=extensions # Toutes les ressources dans le groupe d'API "extensions"
```

Formattage de l'affichage

Pour afficher les détails sur votre terminal dans un format spécifique, utilisez l'option `-o` (ou `--output`) avec les commandes `kubectl` qui la prend en charge.

Format d'affichage	Description
`-o=custom-columns=<spec>`	Affiche un tableau en spécifiant une liste de colonnes séparées par des virgules
`-o=custom-columns-file=<filename>`	Affiche un tableau en utilisant les colonnes spécifiées dans le fichier `<filename>`
`-o=json`	Affiche un objet de l'API formaté en JSON
`-o=jsonpath=<template>`	Affiche les champs définis par une expression jsonpath
`-o=jsonpath-file=<filename>`	Affiche les champs définis par l'expression jsonpath dans le fichier `<filename>`
`-o=name`	Affiche seulement le nom de la ressource et rien de plus
`-o=wide`	Affiche dans le format texte avec toute information supplémentaire, et pour des pods, le nom du noeud est inclus
`-o=yaml`	Affiche un objet de l'API formaté en YAML

Exemples utilisant `-o=custom-columns` :

```
# Toutes les images s'exécutant dans un cluster
kubectl get pods -A -o=custom-columns='DATA:spec.containers[*].image'

 # Toutes les images excepté "registry.k8s.io/coredns:1.6.2"
kubectl get pods -A -o=custom-columns='DATA:spec.containers[?(@.image!="registry.k8s.io/coredns:1.6.2")].image'

# Tous les champs dans metadata quel que soit leur nom
kubectl get pods -A -o=custom-columns='DATA:metadata.*'
```

Plus d'exemples dans la documentation de référence de kubectl.

Verbosité de l'affichage de Kubectl et débogage

La verbosité de Kubectl est contrôlée par une des options `-v` ou `--v` suivie d'un entier représentant le niveau de log. Les conventions générales de logging de Kubernetes et les niveaux de log associés sont décrits ici.

Verbosité	Description
`--v=0`	Le minimum qui doit *toujours* être affiché à un opérateur.
`--v=1`	Un niveau de log par défaut raisonnable si vous n'avez pas besoin de verbosité.

Verbosité	Description
`--v=2`	Informations utiles sur l'état stable du service et messages de logs importants qui peuvent être corrélés à des changements significatifs dans le système. C'est le niveau de log par défaut recommandé pour la plupart des systèmes.
`--v=3`	Informations étendues sur les changements.
`--v=4`	Verbosité de Debug.
`--v=6`	Affiche les ressources requêtées.
`--v=7`	Affiche les entêtes des requêtes HTTP.
`--v=8`	Affiche les contenus des requêtes HTTP.
`--v=9`	Affiche les contenus des requêtes HTTP sans les tronquer.

A suivre

- En savoir plus sur l'Aperçu de kubectl.
- Voir les options kubectl.
- Voir aussi les Conventions d'usage de kubectl pour comprendre comment l'utiliser dans des scripts réutilisables.
- Voir plus d'aides-mémoire kubectl.

5 - Commandes kubectl

Commandes kubectl

[Référence des commandes kubectl]

6 - Conventions d'utilisation de kubectl

kubectl conventions

Conventions d'utilisation recommandées pour `kubectl`.

Utiliser `kubectl` dans des scripts réutilisables

Pour une sortie stable dans un script :

- Demandez un des formats de sortie orienté machine, comme `-o name`, `-o json`, `-o yaml`, `-o go-template` ou `-o jsonpath`.
- Spécifiez complètement la version. Par exemple, `jobs.v1.batch/monjob`. Cela va assurer que kubectl n'utilise pas sa version par défaut, qui risque d'évoluer avec le temps.
- Ne vous basez pas sur un contexte, des préférences ou tout autre état implicite.

Bonnes pratiques

`kubectl run`

Pour que `kubectl run` satisfasse l'infrastructure as code :

- Taggez les images avec un tag spécifique à une version et n'utilisez pas ce tag pour une nouvelle version. Par exemple, utilisez `:v1234`, `v1.2.3`, `r03062016-1-4`, plutôt que `:latest` (Pour plus d'informations, voir Bonnes pratiques pour la configuration).
- Capturez le script pour une image fortement paramétrée.
- Passez à des fichiers de configuration enregistrés dans un système de contrôle de source pour des fonctionnalités désirées mais non exprimables avec des flags de `kubectl run`.

Vous pouvez utiliser l'option `--dry-run` pour prévisualiser l'objet qui serait envoyé à votre cluster, sans réellement l'envoyer.

> **Note:**
> Tous les générateurs `kubectl` sont dépréciés. Voir la documentation de Kubernetes v1.17 pour une liste de générateurs et comment ils étaient utilisés.

Générateurs

Vous pouvez générer les ressources suivantes avec une commande kubectl, `kubectl create --dry-run -o yaml` :

```
clusterrole         Crée un ClusterRole.
clusterrolebinding  Crée un ClusterRoleBinding pour un ClusterRole particulier.
configmap           Crée une configmap à partir d'un fichier local, un répertoire ou une valeur litérale.
cronjob             Crée un cronjob avec le nom spécifié.
deployment          Crée un deployment avec le nom spécifié.
job                 Crée un job avec le nom spécifié.
namespace           Crée un namespace avec le nom spécifié.
poddisruptionbudget Crée un pod disruption budget avec le nom spécifié.
priorityclass       Crée une priorityclass avec le nom spécifié.
quota               Crée un quota avec le nom spécifié.
role                Crée un role avec une unique règle.
rolebinding         Crée un RoleBinding pour un Role ou ClusterRole particulier.
secret              Crée un secret en utilisant la sous-commande spécifiée.
service             Crée un service en utilisant la sous-commande spécifiée.
serviceaccount      Crée un service account avec le nom spécifié.
```

`kubectl apply`

- Vous pouvez utiliser `kubectl apply` pour créer ou mettre à jour des ressources. Pour plus d'informations sur l'utilisation de `kubectl apply` pour la mise à jour de ressources, voir le livre Kubectl.

7 - kubectl

Référence kubectl

Synopsis

kubectl contrôle le manager d'un cluster Kubernetes

Vous trouverez plus d'informations ici : https://kubernetes.io/fr/docs/reference/kubectl/overview/

```
kubectl [flags]
```

Options

--add-dir-header

Si vrai, ajoute le répertoire du fichier à l'entête

--alsologtostderr

log sur l'erreur standard en plus d'un fichier

--application-metrics-count-limit int Défaut : 100

Nombre max de métriques d'applications à stocker (par conteneur)

--as chaîne

Nom d'utilisateur à utiliser pour l'opération

--as-group tableauDeChaînes

Groupe à utiliser pour l'opération, ce flag peut être répété pour spécifier plusieurs groupes

--azure-container-registry-config chaîne

Chemin du fichier contenant les informations de configuration du registre de conteneurs Azure

--boot-id-file string Défaut : "/proc/sys/kernel/random/boot_id"

Liste séparée par des virgules de fichiers dans lesquels rechercher le boot-id. Utilise le premier trouvé.

--cache-dir chaîne Défaut: "/home/karen/.kube/http-cache"

Répertoire de cache HTTP par défaut

--certificate-authority chaîne

Chemin vers un fichier cert pour l'autorité de certification

--client-certificate chaîne

Chemin vers un fichier de certificat client pour TLS

--client-key chaîne

Chemin vers un fichier de clé client pour TLS

--cloud-provider-gce-lb-src-cidrs cidrs Défaut: 130.211.0.0/22,209.85.152.0/22,209.85.204.0/22,35.191.0.0/16

CIDRs ouverts dans le firewall GCE pour le proxy de trafic LB & health checks

--cluster chaîne

Le nom du cluster kubeconfig à utiliser

--container-hints chaîne Défaut : "/etc/cadvisor/container_hints.json"

location du fichier hints du conteneur

--containerd chaîne Défaut : "/run/containerd/containerd.sock"

Point de terminaison de containerd

--containerd-namespace chaîne Défaut : "k8s.io"

namespace de containerd

--context chaîne

Le nom du contexte kubeconfig à utiliser

--default-not-ready-toleration-seconds int Défaut: 300

Indique les tolerationSeconds de la tolérance pour notReady:NoExecute qui sont ajoutées par défaut à tous les pods qui n'ont pas défini une telle tolérance

--default-unreachable-toleration-seconds int Défaut: 300

Indique les tolerationSeconds de la tolérance pour unreachable:NoExecute qui sont ajoutées par défaut à tous les pods qui n'ont pas défini une telle tolérance

--disable-root-cgroup-stats

Désactive la collecte des stats du Cgroup racine

--docker chaîne Défaut : "unix:///var/run/docker.sock"

Point de terminaison docker

--docker-env-metadata-whitelist chaîne

une liste séparée par des virgules de variables d'environnement qui doivent être collectées pour les conteneurs docker

--docker-only

Remonte uniquement les stats Docker en plus des stats racine

--docker-root chaîne Défaut : "/var/lib/docker"

DÉPRÉCIÉ : la racine de docker est lue depuis docker info (ceci est une solution de secours, défaut : /var/lib/docker)

--docker-tls

utiliser TLS pour se connecter à docker

--docker-tls-ca chaîne Défaut : "ca.pem"

chemin vers CA de confiance

--docker-tls-cert chaîne Défaut : "cert.pem"

chemin vers le certificat client

--docker-tls-key chaîne Défaut : "key.pem"

chemin vers la clef privée

--enable-load-reader

Activer le lecteur de la charge CPU

--event-storage-age-limit chaîne Défaut : "default=0"

Durée maximale pendant laquelle stocker les événements (par type). La valeur est une liste séparée par des virgules de clefs/valeurs, où les clefs sont des types d'événements (par ex: creation, oom) ou "default" et la valeur est la durée. La valeur par défaut est appliquée à tous les types d'événements non spécifiés

--event-storage-event-limit chaîne Défaut : "default=0"

Nombre max d'événements à stocker (par type). La valeur est une liste séparée par des virgules de clefs/valeurs, où les clefs sont les types d'événements (par ex: creation, oom) ou "default" et la valeur est un entier. La valeur par défaut est appliquée à tous les types d'événements non spécifiés

--global-housekeeping-interval durée Défaut : 1m0s

Intevalle entre ménages globaux

-h, --help

aide pour kubectl

--housekeeping-interval durée Défaut : 10s

Intervalle entre ménages des conteneurs

--insecure-skip-tls-verify

Si vrai, la validité du certificat du serveur ne sera pas vérifiée. Ceci rend vos connexions HTTPS non sécurisées

--kubeconfig chaîne

Chemin du fichier kubeconfig à utiliser pour les requêtes du CLI

--log-backtrace-at traceLocation Défaut: :0

lorsque les logs arrivent à la ligne fichier:N, émet une stack trace

--log-cadvisor-usage

Activer les logs d'usage du conteneur cAdvisor

--log-dir chaîne

Si non vide, écrit les fichiers de log dans ce répertoire

--log-file chaîne

Si non vide, utilise ce fichier de log

--log-file-max-size uint Défaut : 1800

Définit la taille maximale d'un fichier de log. L'unité est le mega-octet. Si la valeur est 0, la taille de fichier maximale est illimitée.

--log-flush-frequency durée Défaut: 5s

Nombre de secondes maximum entre flushs des logs

--logtostderr Défaut: true

log sur l'erreur standard plutôt que dans un fichier

--machine-id-file chaîne Défaut : "/etc/machine-id,/var/lib/dbus/machine-id"

liste séparée par des virgules de fichiers dans lesquels rechercher le machine-id. Utiliser le premier trouvé.

--match-server-version

La version du serveur doit correspondre à la version du client

-n, --namespace chaîne

Si présent, la portée de namespace pour la requête du CLI

--password chaîne

Mot de passe pour l'authentification de base au serveur d'API

--profile chaîne Défaut: "none"

Nom du profil à capturer. Parmi (none|cpu|heap|goroutine|threadcreate|block|mutex)

--profile-output chaîne Défaut: "profile.pprof"

Nom du fichier dans lequel écrire le profil

--request-timeout chaîne Défaut: "0"

La durée à attendre avant d'abandonner une requête au serveur. Les valeurs non égales à zéro doivent contenir une unité de temps correspondante (ex 1s, 2m, 3h). Une valeur à zéro indique de ne pas abandonner les requêtes

-s, --server chaîne

L'adresse et le port de l'API server Kubernetes

--skip-headers

Si vrai, n'affiche pas les entêtes dans les messages de log

--skip-log-headers

Si vrai, évite les entêtes lors de l'ouverture des fichiers de log

--stderrthreshold sévérité Défaut: 2

logs à cette sévérité et au dessus de ce seuil vont dans stderr

--storage-driver-buffer-duration durée Défaut : 1m0s

Les écritures dans le driver de stockage seront bufferisés pour cette durée, et seront envoyés aux backends non-mémoire en une seule transaction

--storage-driver-db chaîne Défaut : "cadvisor"

nom de la base de données

--storage-driver-host chaîne Défaut : "localhost:8086"

hôte:port de la base de données

--storage-driver-password chaîne Défaut : "root"

Mot de passe de la base de données

--storage-driver-secure

utiliser une connexion sécurisée avec la base de données

--storage-driver-table chaîne Défaut : "stats"

Nom de la table dans la base de données

--storage-driver-user chaîne Défaut : "root"

nom d'utilisateur de la base de données

--token chaîne

Bearer token pour l'authentification auprès de l'API server

--update-machine-info-interval durée Défaut : 5m0s

Intevalle entre mises à jour des infos machine.

--user chaîne

Le nom de l'utilisateur kubeconfig à utiliser

--username chaîne

Nom d'utilisateur pour l'authentification de base au serveur d'API

-v, --v Niveau

Niveau de verbosité des logs

--version version[=true]

Affiche les informations de version et quitte

--vmodule moduleSpec

Liste de settings pattern=N séparés par des virgules pour le logging filtré par fichiers

See Also

- kubectl alpha - Commandes pour fonctionnalités alpha
- kubectl annotate - Met à jour les annotations d'une ressource
- kubectl api-resources - Affiche les ressources de l'API prises en charge sur le serveur
- kubectl api-versions - Affiche les versions de l'API prises en charge sur le serveur, sous la forme "groupe/version"
- kubectl apply - Applique une configuration à une ressource depuis un fichier ou stdin
- kubectl attach - Attache à un conteneur en cours d'exécution
- kubectl auth - Inspecte les autorisations
- kubectl autoscale - Auto-scale un Deployment, ReplicaSet, ou ReplicationController

- kubectl certificate - Modifie des ressources certificat
- kubectl cluster-info - Affiche les informations du cluster
- kubectl completion - Génère le code de complétion pour le shell spécifié (bash ou zsh)
- kubectl config - Modifie les fichiers kubeconfig
- kubectl convert - Convertit des fichiers de config entre différentes versions d'API
- kubectl cordon - Marque un nœud comme non assignable (unschedulable)
- kubectl cp - Copie des fichiers et répertoires depuis et vers des conteneurs
- kubectl create - Crée une ressource depuis un fichier ou stdin
- kubectl delete - Supprime des ressources par fichiers ou stdin, par ressource et nom, ou par ressource et sélecteur de label
- kubectl describe - Affiche les informations d'une ressource spécifique ou d'un groupe de ressources
- kubectl diff - Différence entre la version live et la version désirée
- kubectl drain - Draine un nœud en préparation d'une mise en maintenance
- kubectl edit - Édite une ressource du serveur
- kubectl exec - Exécute une commande dans un conteneur
- kubectl explain - Documentation sur les ressources
- kubectl expose - Prend un replication controller, service, deployment ou pod et l'expose comme un nouveau Service Kubernetes
- kubectl get - Affiche une ou plusieurs ressources
- kubectl kustomize - Construit une cible kustomization à partir d'un répertoire ou d'une URL distante.
- kubectl label - Met à jour les labels d'une ressource
- kubectl logs - Affiche les logs d'un conteneur dans un pod
- kubectl options - Affiche la liste des flags hérités par toutes les commandes
- kubectl patch - Met à jour un ou plusieurs champs d'une ressource par merge patch stratégique
- kubectl plugin - Fournit des utilitaires pour interagir avec des plugins
- kubectl port-forward - Redirige un ou plusieurs ports vers un pod
- kubectl proxy - Exécute un proxy vers l'API server Kubernetes
- kubectl replace - Remplace une ressource par fichier ou stdin
- kubectl rollout - Gère le rollout d'une ressource
- kubectl run - Exécute une image donnée dans le cluster
- kubectl scale - Définit une nouvelle taille pour un Deployment, ReplicaSet ou Replication Controller
- kubectl set - Définit des fonctionnalités spécifiques sur des objets
- kubectl taint - Met à jour les marques (taints) sur un ou plusieurs nœuds
- kubectl top - Affiche l'utilisation de ressources matérielles (CPU/Memory/Storage)
- kubectl uncordon - Marque un nœud comme assignable (schedulable)
- kubectl version - Affiche les informations de version du client et du serveur
- kubectl wait - Expérimental : Attend une condition particulière sur une ou plusieurs ressources