

# CLI kubectl

kubectl CLI description

- 1: [Aperçu de kubectl](#)
- 2: [Support de JSONPath](#)
- 3: [Aide-mémoire kubectl](#)
- 4: [Commandes kubectl](#)
- 5: [Conventions d'utilisation de kubectl](#)
- 6: [kubectl](#)

## 1 - 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<#>` .  
Exemple: `$ 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     | <code>kubectl alpha SOUS-COMMANDE [flags]</code>   | Liste les commandes disponibles qui correspondent à des fonctionnalités alpha, qui ne sont pas activées par défaut dans les clusters Kubernetes. |
| annotate  | <code>kubectl annotate (-f FICHER   TYPE NOM   TYPE/NOM) CLE_1=VAL_1 ... CLE_N=VAL_N [--overwrite] [--all] [--resource-version=version] [flags]</code> | Ajoute ou modifie les annotations d'une ou plusieurs ressources.   |

| Opération     | Syntaxe  | Description  |
|---------------|--|--|
| 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.            |

| Opération | Syntaxe   | Description   |
|-----------|---|---|
| cordon    | <code>kubectl cordon NOEUD [options]</code>   | Marque un nœud comme non programmable.  |
| cp        | <code>kubectl cp &lt;fichier-src&gt; &lt;fichier-dest&gt; [options]</code>                | Copie des fichiers et des répertoires vers et depuis des conteneurs.  |
| create    | <code>kubectl create -f FICHIER [flags]</code>  | Crée une ou plusieurs ressources depuis un fichier ou stdin.  |
| delete    | <code>kubectl delete (-f FICHIER   TYPE [NOM   /NOM   -l label   --all]) [flags]</code>   | 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  | <code>kubectl describe (-f FICHIER   TYPE [PREFIXE_NOM   /NOM   -l label]) [flags]</code> | Affiche l'état détaillé d'une ou plusieurs ressources.  |
| diff      | <code>kubectl diff -f FICHIER [flags]</code>  | Diff un fichier ou stdin par rapport à la configuration en cours  |
| drain     | <code>kubectl drain NOEUD [options]</code>  | Vide un nœud en préparation de sa mise en maintenance.  |
| edit      | <code>kubectl edit (-f FICHIER   TYPE NOM   TYPE/NOM) [flags]</code>                      | Édite et met à jour la définition d'une ou plusieurs ressources sur le serveur en utilisant l'éditeur par défaut.   |
| exec      | <code>kubectl exec POD [-c CONTENEUR] [-i] [-t] [flags] [-- COMMANDE [args...]]</code>    | Exécute une commande à l'intérieur d'un conteneur dans un pod.  |
| explain   | <code>kubectl explain [--recursive=false] [flags]</code>                                  | Obtient des informations sur différentes ressources. Par exemple pods, nœuds, services, etc.  |

| Opération    | Syntaxe  | Description  |
|--------------|--|--|
| expose       | <code>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]</code> | Expose un replication controller, service ou pod comme un nouveau service Kubernetes.  |
| get          | <code>kubectl get (-f FICHIER   TYPE [NOM   /NOM   -l label]) [--watch] [--sort-by=CHAMP] [[-o   --output]=FORMAT_AFFICHAGE] [flags]</code>  | Liste une ou plusieurs ressources.   |
| kustomize    | <code>kubectl kustomize &lt;répertoire&gt; [flags] [options]</code>  | 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        | <code>kubectl label (-f FICHIER   TYPE NOM   TYPE/NOM) CLE_1=VAL_1 ... CLE_N=VAL_N [--overwrite] [--all] [--resource-version=version] [flags]</code>   | Ajoute ou met à jour les labels d'une ou plusieurs ressources.   |
| logs         | <code>kubectl logs POD [-c CONTENEUR] [--follow] [flags]</code>  | Affiche les logs d'un conteneur dans un pod.   |
| options      | <code>kubectl options</code>   | Liste des options globales, s'appliquant à toutes commandes.   |
| patch        | <code>kubectl patch (-f FICHIER   TYPE NOM   TYPE/NOM) --patch PATCH [flags]</code>  | Met à jour un ou plusieurs champs d'une ressource en utilisant le processus de merge patch stratégique.  |
| plugin       | <code>kubectl plugin [flags] [options]</code>  | Fournit des utilitaires pour interagir avec des plugins.   |
| port-forward | <code>kubectl port-forward POD [PORT_LOCAL:]PORT_DISTANT [...] [PORT_LOCAL_N:]PORT_DISTANT_N] [flags]</code>   | Transfère un ou plusieurs ports locaux vers un pod.  |

| Opération | Syntaxe   | Description  |
|-----------|---|--|
| 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 [--label 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<br>abrégés | Groupe API                   | P<br>n |
|---------------------------------|-----------------|------------------------------|--------|
| bindings                        |                 |                              | tr     |
| componentstatuses               | cs              |                              | fa     |
| configmaps                      | cm              |                              | tr     |
| endpoints                       | ep              |                              | tr     |
| limitranges                     | limits          |                              | tr     |
| namespaces                      | ns              |                              | fa     |
| nodes                           | no              |                              | fa     |
| persistentvolumeclaims          | pvc             |                              | tr     |
| persistentvolumes               | pv              |                              | fa     |
| Pods                            | po              |                              | tr     |
| podtemplates                    |                 |                              | tr     |
| replicationcontrollers          | rc              |                              | tr     |
| resourcequotas                  | quota           |                              | tr     |
| secrets                         |                 |                              | tr     |
| serviceaccounts                 | sa              |                              | tr     |
| services                        | svc             |                              | tr     |
| mutatingwebhookconfigurations   |                 | admissionregistration.k8s.io | fa     |
| validatingwebhookconfigurations |                 | admissionregistration.k8s.io | fa     |
| customresourcedefinitions       | crd ,<br>crds   | apiextensions.k8s.io         | fa     |
| apiservices                     |                 | apiregistration.k8s.io       | fa     |
| controllerrevisions             |                 | apps                         | tr     |
| daemonsets                      | ds              | apps                         | tr     |
| deployments                     | deploy          | apps                         | tr     |
| replicasets                     | rs              | apps                         | tr     |

| Nom de la ressource        | Noms<br>abrégés | Groupe API                | P<br>n |
|----------------------------|-----------------|---------------------------|--------|
| statefulsets               | sts             | apps                      | tr     |
| tokenreviews               |                 | authentication.k8s.io     | f      |
| localsubjectaccessreviews  |                 | authorization.k8s.io      | tr     |
| selfsubjectaccessreviews   |                 | authorization.k8s.io      | f      |
| selfsubjectrulesreviews    |                 | authorization.k8s.io      | f      |
| subjectaccessreviews       |                 | authorization.k8s.io      | f      |
| horizontalpodautoscalers   | hpa             | autoscaling               | tr     |
| cronjobs                   | cj              | batch                     | tr     |
| jobs                       |                 | batch                     | tr     |
| certificatesigningrequests | csr             | certificates.k8s.io       | f      |
| leases                     |                 | coordination.k8s.io       | tr     |
| events                     | ev              | events.k8s.io             | tr     |
| ingresses                  | ing             | extensions                | tr     |
| networkpolicies            | netpol          | networking.k8s.io         | tr     |
| poddisruptionbudgets       | pdb             | policy                    | tr     |
| podsecuritypolicies        | psp             | policy                    | f      |
| clusterrolebindings        |                 | rbac.authorization.k8s.io | f      |
| clusterroles               |                 | rbac.authorization.k8s.io | f      |
| rolebindings               |                 | rbac.authorization.k8s.io | tr     |
| roles                      |                 | rbac.authorization.k8s.io | tr     |
| priorityclasses            | pc              | scheduling.k8s.io         | f      |
| csidrivers                 |                 | storage.k8s.io            | f      |
| csinodes                   |                 | storage.k8s.io            | f      |
| storageclasses             | sc              | storage.k8s.io            | f      |
| volumeattachments          |                 | storage.k8s.io            | f      |

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

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
$ 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 les colonnes de statut.
$ 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 par 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
# Rappelez-vous : les noms des pods étant créés par un replicat
$ 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
$ kubectl delete -f pod.yaml

# Supprime tous les pods et services ayant le label <clé-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>
$ kubectl exec <nom-pod> -- date

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

# Obtient un TTY interactif et exécute /bin/bash depuis le pod
$ 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 '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
# 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,
/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 .
```

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](#).

## 2 - 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 + longueurListe >= 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  |
| @                | l'objet courant            | {@}  | identifie l'entrée   |
| . ou []          | opérateur fils             | {kind}, {[kind]} ou {[name.type]}                              | Liste  |
| ..               | descente récursive         | {..name}   | 127.0.0.1<br>127.0.0.1<br>myserver                         |
| *                | joker. Tous les objets     | {items[*].metadata.name}                                       | [127.0.0.1, 127.0.0.1]                                     |
| [start:end:step] | opérateur d'indice         | {users[0].name}  | myserver   |
| [,]              | opérateur d'union          | {items[*]['metadata.name', 'status.capacity']}                 | 127.0.0.1<br>127.0.0.1<br>map[127.0.0.1]<br>map[127.0.0.1] |
| ?()              | filtre                     | {users[?(@.name=="e2e")].user.password}                        | secret   |
| range , end      | itération de liste         | {range items[*]}<br>[{metadata.name}, {status.capacity}] {end} | [127.0.0.1, 127.0.0.1]<br>map[127.0.0.1]<br>map[127.0.0.1] |
| ' '              | protège chaîne interprétée | {range items[*]}<br>{metadata.name}{'\t'}{end}                 | 127.0.0.1<br>127.0.0.1                                     |

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}
```



**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 littérales dans le modèle. Par exemple :

```
kubectl get pods -o=jsonpath="{range .items[*]}{.metadata.n  
kubectl get pods -o=jsonpath="{range .items[*]}{.metadata.n
```

# 3 - 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
echo "source <(kubectl completion bash)" >> ~/.bashrc # ajoute
```

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
echo "[[ \$commands[kubectl] ]] && source <(kubectl completion zsh)" >> ~/.zshrc
```

### 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 les paramètres
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 nom de l'utilisateur
kubectl config view -o jsonpath='{.users[*].name}' # Affiche une liste de noms d'utilisateurs
kubectl config get-contexts # Affiche la liste des contextes
kubectl config current-context # Affiche le contexte actuel
kubectl config use-context my-cluster-name # Définit my-cluster-name comme contexte actuel

# Ajoute un nouveau cluster à votre kubeconf, prenant en charge les certificats
kubectl config set-credentials kubeuser/foo.kubernetes.com --user=kubeuser --password=kubeuser

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

# Définit et utilise un contexte qui utilise un nom d'utilisateur et un namespace
kubectl config set-context gce --user=cluster-admin --namespace=default
&& 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
kubectl apply -f https://git.io/vPieo         # Crée une ou plusieurs ressources
kubectl create deployment nginx --image=nginx # Démarre une instance de nginx
kubectl explain pods                          # Affiche la documentation pour pods

# 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
kubectl get pods --all-namespaces # Liste tous les Pods
kubectl get pods -o wide # Liste tous les Pods
kubectl get deployment my-dep # Liste un déploiement
kubectl get pods # Liste tous les Pods
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
kubectl get pods --selector=app=cassandra -o \
  jsonpath='{.items[*].metadata.labels.version}'

# Affiche tous les noeuds (en utilisant un sélecteur pour exclure
# 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 cluster
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
# "jq" est une commande utile pour des transformations non prises en charge
sel=${$(kubectl get rc my-rc --output=json | jq -j '.spec.selector')}
echo ${$(kubectl get pods --selector=$sel --output=jsonpath='{.items[*].metadata.name}')}

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

# Vérifie quels noeuds sont prêts
JSONPATH='{range .items[*]}{@.metadata.name}:{range @.status.conditions[] | select(.type=="Ready")}.message}'
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[] | select(.value=="$(cat /dev/urandom | fold -n 40 | tr -dc 'a-z0-9' | fold -w 40 | tr -d '\n' | tr -s 40 | base64))'"

# Liste les containerIDs des initContainers de tous les Pods
# Utile lors du nettoyage des conteneurs arrêtés, tout en évitant de supprimer les containers en cours d'exécution
kubectl get pods --all-namespaces -o jsonpath='{range .items[*]}{@.metadata.namespace},{@.spec.initContainers[].name},{@.spec.initContainers[].containerID}}'

# 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 est disponible
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
kubectl rollout history deployment/frontend
kubectl rollout undo deployment/frontend
kubectl rollout undo deployment/frontend --to-revision=2
kubectl rollout status -w deployment/frontend
kubectl rollout restart deployment/frontend

cat pod.json | kubectl replace -f -

# Remplace de manière forcée (Force replace), supprime puis re-
kubectl replace --force -f ./pod.json

# Crée un service pour un nginx repliqué, qui rend le service s
kubectl expose rc nginx --port=80 --target-port=8000

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

kubectl label pods my-pod new-label=awesome
kubectl annotate pods my-pod icon-url=http://goo.gl/XXBTWq
kubectl autoscale deployment foo --min=2 --max=10
```

## 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 l'image d'un conteneur ; spec.containers[*].name e
kubectl patch pod valid-pod -p '{"spec":{"containers":[{"name":

# Met à jour l'image d'un conteneur en utilisant un patch json
kubectl patch pod valid-pod --type='json' -p='[{"op": "replace"

# Désactive la livenessProbe d'un déploiement en utilisant un p
kubectl patch deployment valid-deployment --type json -p='[{"

# Ajoute un nouvel élément à un tableau indexé
kubectl patch sa default --type='json' -p='[{"op": "add", "path
```

## Édition de ressources

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

```
kubectl edit svc/docker-registry # Édite l
KUBE_EDITOR="nano" kubectl edit svc/docker-registry # Utilise
```

## Mise à l'échelle de ressources

```
kubectl scale --replicas=3 rs/foo
kubectl scale --replicas=3 -f foo.yaml
kubectl scale --current-replicas=2 --replicas=3 deployment/mysc
kubectl scale --replicas=5 rc/foo rc/bar rc/baz
```

## Suppression de ressources

```
kubectl delete -f ./pod.json
kubectl delete pod,service baz foo
kubectl delete pods,services -l name=myLabel
kubectl -n my-ns delete pod,svc --all
# Supprime tous les pods correspondants à pattern1 ou pattern2
kubectl get pods -n mynamespace --no-headers=true | awk '/patt
```

## Interaction avec des Pods en cours d'exécution

```
kubectl logs my-pod # Affiche l'
kubectl logs -l name=myLabel # Affiche l'
kubectl logs my-pod --previous # Affiche l'
kubectl logs my-pod -c my-container # Affiche l'
kubectl logs -l name=myLabel -c my-container # Affiche l'
kubectl logs my-pod -c my-container --previous # Affiche l'
kubectl logs -f my-pod # Fait défi
kubectl logs -f my-pod -c my-container # Fait défi
kubectl logs -f -l name=myLabel --all-containers # Fait défi
kubectl run -i --tty busybox --image=busybox -- sh # Exécute u
kubectl run nginx --image=nginx --restart=Never -n # Exécute l'
mysamespace # Exécute l'
kubectl run nginx --image=nginx --restart=Never # Simule l'
--dry-run -o yaml > pod.yaml

kubectl attach my-pod -i # Attache à
kubectl port-forward my-pod 5000:6000 # Écoute le
kubectl exec my-pod -- ls / # Exécute u
kubectl exec my-pod -c my-container -- ls / # Exécute u
kubectl top pod POD_NAME --containers # Affiche l'
```

## Interaction avec des Noeuds et Clusters

```
kubectl cordon mon-noeud
kubectl drain mon-noeud
kubectl uncordon mon-noeud
kubectl top node mon-noeud
kubectl cluster-info
kubectl cluster-info dump
kubectl cluster-info dump --output-directory=/path/to/cluster-s

# Si une teinte avec cette clé et cet effet existe déjà, sa val
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 ressource
kubectl api-resources --namespaced=false     # Toutes les ressource
kubectl api-resources -o name                 # Toutes les ressource
kubectl api-resources -o wide                 # Toutes les ressource
kubectl api-resources --verbs=list,get       # Toutes les ressource
kubectl api-resources --api-group=extensions # Toutes les ressource
```

## 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  |
|--|--|
| <code>-o=custom-columns=&lt;spec&gt;</code>          | Affiche un tableau en spécifiant une liste de colonnes séparées par des virgules                                   |
| <code>-o=custom-columns-file=&lt;filename&gt;</code> | Affiche un tableau en utilisant les colonnes spécifiées dans le fichier <code>&lt;filename&gt;</code>              |
| <code>-o=json</code>                                 | Affiche un objet de l'API formaté en JSON  |
| <code>-o=jsonpath=&lt;template&gt;</code>            | Affiche les champs définis par une expression <a href="#">jsonpath</a>   |
| <code>-o=jsonpath-file=&lt;filename&gt;</code>       | Affiche les champs définis par l'expression <a href="#">jsonpath</a> dans le fichier <code>&lt;filename&gt;</code> |
| <code>-o=name</code>                                 | Affiche seulement le nom de la ressource et rien de plus   |
| <code>-o=wide</code>                                 | Affiche dans le format texte avec toute information supplémentaire, et pour des pods, le nom du noeud est inclus   |
| <code>-o=yaml</code>                                 | 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 <i>toujours</i> être affiché à un opérateur.  |
| --v=1     | Un niveau de log par défaut raisonnable si vous n'avez pas besoin de verbosité.   |
| --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](#).

## 4 - Commandes kubectrl

Commandes kubectrl

[Référence des commandes kubectrl](#)

# 5 - 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 Cluste |
| configmap           | Crée une configmap à partir d'un fichier  |
| 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 |
| priorityclass       | Crée une priorityclass avec le nom spécif |
| quota               | Crée un quota avec le nom spécifié.       |
| role                | Crée un role avec une unique règle.       |
| rolebinding         | Crée un RoleBinding pour un Role ou Clust |
| secret              | Crée un secret en utilisant la sous-comm  |
| service             | Crée un service en utilisant la sous-comm |
| serviceaccount      | Crée un service account avec le nom spéci |

## 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](#).

# 6 - 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:<br>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   |
| Intervalle 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   |



|  |                |   |  |
|--|----------------|---|--|
| LI kubectl   Kubernetes  |                | https://kubernetes.io/fr/docs/reference/kubectl/_print/ |  |
| --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