



About Kubectl

Kubectl is a command line interface for running commands against Kubernetes clusters.

CLI config

Kubectl config contains cluster's API endpoint, credentials and can be configured to use several contexts.

`kubectl version` shows a kubectl and a kubernetes cluster components version

`kubectl config view` shows a kubectl config

`kubectl config current-context` shows a current context

`kubectl config use-context my-k8s` uses a particular context

`kubectl config set-credentials \ kubeuser/foo.kubernetes.com \ --username=kubeuser \ --password=kubepassword` adds a new cluster and user credentials to your kubectl config

Namespaces

Kubernetes namespaces can be presented as directories, that help to group resources logically. The default namespace is used by default. The kube-system namespace is typically used for cluster resources.

`kubectl get ns` gets a list of all namespaces

`kubectl create ns jenkins` creates a namespace named jenkins

The default namespace will be used in every command by default. To change this behaviour, use `--namespace=<name>` and `--all-namespaces` flags.

Manage cluster

`kubectl cordon worker-1` marks a node as unschedulable

`kubectl drain worker-1` prepares worker-1 for maintenance, removes all resources from a node

`kubectl cluster-info` gets cluster information

`kubectl top node kubernetes-minion-group` gets system statistics from a node kubernetes-minion-group

`kubectl label worker-1 disk=ssd` adds a label to a node instance. Labels allow to manage resources in a more flexible way

Collect information from your cluster

Types of objects: pods/services/deployments/persistentVolumes/replicaSets/statefulSets/etc.

`kubectl get <object>` gets general info about cluster resource(s)

`kubectl get <object> -o wide` shows resource information with some additional parameters

`kubectl get <object> -o [json, yaml]` gets general information in a json or a yaml output format

`kubectl describe <object>` gets general information about cluster resource(s) in details

`kubectl get pods \ --namespace=kube-system` gets info about pods in a particular namespace

`kubectl describe nodes worker-1` gets verbose description of a node named worker-1

`kubectl get pods --field-selector=status.phase=Failed --all-namespaces` gets all pods in a failed state from the whole cluster

`kubectl describe all \ --all-namespaces` describes all cluster resources

Create resources in your cluster

Do not mix create and apply techniques when creating objects. The create command does not retain `kubectl.kubernetes.io/last-applied-configuration` annotation, which is used by the apply command. Apply is imperative and can accumulate changes made to an object (e.g by scale command).

`kubectl create -f ./manifest.yaml` creates a resource described in a manifest

`kubectl apply -f ./dir` creates resources from all files in a directory
`kubectl run dev-nginx --image=nginx` runs a single nginx instance

Update resources

Kubernetes allows you to easily scale your resources.

`kubectl scale deployment \ --replicas=3 -l run=nginx-a` scales nginx to 3 replicas

You can easily make rolling updates with zero downtime.

`kubectl rolling-update frontend-v1 \ -f frontend-v2.json` updates pods of frontend with zero downtime

`kubectl rollout undo \ deployment/nginx-deployment \ --to-revision=2` rollsback a nginx deployment to a specified revision

`kubectl autoscale deployment \ nginx-deployment --min=10 \ --max=15 --cpu-percent=80` autoscales a nginx deployment based on CPU load
`kubectl replace --force -f \ ./jenkins.json` replaces and updates resources described in a jenkins.json with downtime

`kubectl label pods jenkins \ new-label=devqa` creates a label on a pod jenkins

`kubectl edit pod \ kube-dns-565cd5b8c9-j6zmd \ --namespace=kube-system` edits a resource manifest with your text editor

Delete resources

`kubectl delete -f ./pod.json` deletes resources described in a manifest
`kubectl delete pods,services -l \ name=myLabel --all-namespaces` deletes pods and services with the label myLabel from all namespaces

Pod debugging tools

`kubectl logs nginx-8586cf59-nj55x` collects logs from a pod

`kubectl top pod nginx` shows pod's metrics

`kubectl exec -it nginx -- /bin/bash` creates or starts an interactive shell into pod

`kubectl port-forward nginx 8080:80` forwards a container port 80 to a local port 8080 so that you can access your containerized app for debugging

`kubectl cp hotfix.yaml \ web1:/config/hotfix.yaml` copies a file to or from a container file system

NOTE: Using `kubectl cp` for any purposes other than debugging or hotfixing is considered to be an antipattern.

Configmaps and Secrets

Secret is a primitive to store sensitive data (passwords, keys, certificates, and etc.) in a container. **Configmap** is a primitive to store pod's configuration.

`kubectl create configmap back-config \ --from-file=my-config.txt \ --from-literal=type=binary \ --from-literal=ext_port=12803`

creates config map from both separate vars and my-config.txt file

`kubectl describe configmaps \ back-1-config` gets configmap configuration values

`kubectl create secret generic web-tls \ --from-file=web.crt \ --from-file=web.key`

creates a secret object to store and use TLS certificates

`kubectl delete secrets \ dev-concourse-postgresql` deletes secrets from a stated object

Helm tool for Kubernetes

Helm is a tool, which helps with complex solutions (like db clusters, or CI tools) deployment to Kubernetes. It is used to install sets of resources called charts, that can be found in a helm repository

`helm init` Helm gets a cluster location and credentials from kubectl config and installs a container with a tiller - a helm server part

`helm repo update` makes sure that helm charts are in actual state

`helm install --name dev-concourse \ stable/concourse` installs a Concourse helm chart (creates a deployment and a corresponding service)

`helm delete dev-concourse` deletes dev-concourse chart resources