Reference Documentation

DOCUMENTATION SETUP CONCEPTS TASKS TUTORIALS **REFERENCE**

Search

kubectl Cheat Sheet

See also: Kubectl Overview and JsonPath Guide.

This page is an overview of the `kubectl` command.

kubectl - Cheat Sheet

Kubectl Autocomplete

BASH

```
source <(kubectl completion bash) # setup autocomplete in bash into
echo "source <(kubectl completion bash)" >> ~/.bashrc # add autocomp
```

ZSH

```
source <(kubectl completion zsh)  # setup autocomplete in zsh into t
echo "if [ $commands[kubectl] ]; then source <(kubectl completion zs
```

Kubectl Context and Configuration

Set which Kubernetes cluster **`kubectl`** communicates with and modifies configuration information. See Authenticating Across Clusters with kubeconfig documentation for detailed config file information.

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

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

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

kubectl config current-context                    # Display the current-co
kubectl config use-context my-cluster-name    # set the default contex

# add a new cluster to your kubeconf that supports basic auth
kubectl config set-credentials kubeuser/foo.kubernetes.com --usernam

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

Creating Objects

Kubernetes manifests can be defined in json or yaml. The file extension `.yaml`, `.yml`, and `.json` can be used.

```
kubectl create -f ./my-manifest.yaml           # create resource(s)
kubectl create -f ./my1.yaml -f ./my2.yaml     # create from multipl
kubectl create -f ./dir                        # create resource(s) 
kubectl create -f https://git.io/vPieo         # create resource(s) 
kubectl run nginx --image=nginx                # start a single inst
kubectl explain pods,svc                       # get the documentati

# Create multiple YAML objects from stdin
cat <<EOF | kubectl create -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

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

Viewing, Finding Resources

```
# Get commands with basic output
kubectl get services                          # List all services in
kubectl get pods --all-namespaces             # List all pods in all
kubectl get pods -o wide                      # List all pods in the
kubectl get deployment my-dep                 # List a particular dep
kubectl get pods --include-uninitialized      # List all pods in the

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

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

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

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

# 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==

# List Names of Pods that belong to Particular RC
# "jq" command useful for transformations that are too complex for js
sel=${$(kubectl get rc my-rc --output=json | jq -j '.spec.selector |
echo $(kubectl get pods --selector=$sel --output=jsonpath={.items..me

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

# List all Secrets currently in use by a pod
kubectl get pods -o json | jq '.items[].spec.containers[].env[]?.valu

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

Updating Resources

```
kubectl rolling-update frontend-v1 -f frontend-v2.json           # Re
kubectl rolling-update frontend-v1 frontend-v2 --image=image:v2  # Cl
kubectl rolling-update frontend --image=image:v2                 # Up
kubectl rolling-update frontend-v1 frontend-v2 --rollback        # Al
cat pod.json | kubectl replace -f -                              # Re

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

# Create a service for a replicated nginx, which serves on port 80 an
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 label pods my-pod new-label=awesome                      # Ad
kubectl annotate pods my-pod icon-url=http://goo.gl/XXBTWq       # Ad
kubectl autoscale deployment foo --min=2 --max=10                # Au
```

Patching Resources

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

# Update a container's image; spec.containers[*].name is required bec
kubectl patch pod valid-pod -p '{"spec":{"containers":[{"name":"kuber

# Update a container's image using a json patch with positional array
kubectl patch pod valid-pod --type='json' -p='[{"op": "replace", "pat

# Disable a deployment livenessProbe using a json patch with position
kubectl patch deployment valid-deployment  --type json   -p='[{"op":

# Add a new element to a positional array
kubectl patch sa default --type='json' -p='[{"op": "add", "path": "/s
```

Editing Resources

The edit any API resource in an editor.

```
kubectl edit svc/docker-registry                      # Edit the serv
KUBE_EDITOR="nano" kubectl edit svc/docker-registry   # Use an altern
```

Scaling Resources

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

cluster that runs 'o World" for ode.js.

samples, and reference documentation. You can even help

users and the Kubernetes authors, attend community events, and watch

in general, and get technical how-tos hot off the presses.

```
kubectl delete -f ./pod.json
kubectl delete pod,service baz foo
kubectl delete pods,services -l name=myLabel
kubectl delete pods,services -l name=myLabel --include-uninitialized
kubectl -n my-ns delete po,svc --all
```

Interacting with running Pods

```
kubectl logs my-pod                                  # dump pod logs (
kubectl logs my-pod -c my-container                  # dump pod contai
kubectl logs -f my-pod                               # stream pod logs
kubectl logs -f my-pod -c my-container               # stream pod cont
kubectl run -i --tty busybox --image=busybox -- sh   # Run pod as inte
kubectl attach my-pod -i                             # Attach to Runni
kubectl port-forward my-pod 5000:6000                # Listen on port
kubectl exec my-pod -- ls /                          # Run command in
kubectl exec my-pod -c my-container -- ls /          # Run command in
kubectl top pod POD_NAME --containers                # Show metrics fo
```

Interacting with Nodes and Cluster

```
kubectl cordon my-node
kubectl drain my-node
kubectl uncordon my-node
kubectl top node my-node
kubectl cluster-info
kubectl cluster-info dump
kubectl cluster-info dump --output-directory=/path/to/cluster-state

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

Resource types

List all supported resource types along with their shortnames, API group, whether they are namespaced, and Kind:

```
kubectl api-resources
```

Other operations for exploring API resources:

```
kubectl api-resources --namespaced=true      # All namespaced resour
kubectl api-resources --namespaced=false     # All non-namespaced re
kubectl api-resources -o name                # All resources with si
kubectl api-resources -o wide                # All resources with ex
kubectl api-resources --verbs=list,get       # All resources that su
kubectl api-resources --api-group=extensions # All resources in the 
```

Formatting output

To output details to your terminal window in a specific format, you can add either the `-o` or `-output` flags to a supported `kubectl` command.

Output format	Description

`-o=custom-columns=<spec>`	Print a table using a comma separated list of custom columns
`-o=custom-columns-file=<filename>`	Print a table using the custom columns template in the `<filename>` file
`-o=json`	Output a JSON formatted API object
`-o=jsonpath=<template>`	Print the fields defined in a jsonpath expression
`-o=jsonpath-file=<filename>`	Print the fields defined by the jsonpath expression in the `<filename>` file
`-o=name`	Print only the resource name and nothing else
`-o=wide`	Output in the plain-text format with any additional information, and for pods, the node name is included
`-o=yaml`	Output a YAML formatted API object

Kubectl output verbosity and debugging

Kubectl verbosity is controlled with the `-v` or `--v` flags followed by an integer representing the log level. General Kubernetes logging conventions and the associated log levels are described here.

Verbosity	Description
`--v=0`	Generally useful for this to ALWAYS be visible to an operator.
`--v=1`	A reasonable default log level if you don't want verbosity.
`--v=2`	Useful steady state information about the service and important log messages that may correlate to significant changes in the system. This is the recommended default log level for most systems.
`--v=3`	Extended information about changes.
`--v=4`	Debug level verbosity.
`--v=6`	Display requested resources.
`--v=7`	Display HTTP request headers.
`--v=8`	Display HTTP request contents.

`--v=9`	Display HTTP request contents without truncation of contents.

What's next

- Learn more about Overview of kubectl.
- See kubectl options.
- Also kubectl Usage Conventions to understand how to use it in reusable scripts.