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) # se[illegible] autocomplete in bash into the current sh
echo "source <(kubectl completion bas[illegible]> ~/.bashrc # add autocomplete permanent
```

ZSH

```
source <(kubectl completion zsh)  # setup autocomplete in zsh into the current she
echo "if [ $commands[kubectl] ]; then source <(kubectl completion zsh); fi" >> ~/.
```

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 config
KUBECONFIG=~/.kube/config:~/.kube/kubconfig2 kubectl config view

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

kubectl config current-context                # Display the current-context
kubectl config use-context my-cluster-name    # set the default context to my-cluste

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

# 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.         # create from multiple files
kubectl create -f ./dir                        # create resource(s) in all manifes
kubectl create -f https://git.io/vPieo         # create resource(s) from url
kubectl run nginx --image=nginx                # start a single instance of nginx
kubectl explain pods,svc                       # get the documentation for pod and

# 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 the namespace
kubectl get pods --all-namespaces             # List all pods in all namespaces
kubectl get pods -o wide                      # List all pods in the namespace, wi
kubectl get deployment my-dep                 # List a particular deployment
kubectl get pods --include-uninitialized      # List all pods in the namespace, in

# 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 by Name

# 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=="ExternalIP")

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

# Check which nodes are ready
JSONPATH='{range .items[*]}{@.metadata.name}:{range @.status.conditions[*]}{@.type
 && 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[]?.valueFrom.secretK

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

Updating Resources

```
kubectl rolling-update frontend-v1 -f frontend-v2.json           # Rolling update
kubectl rolling-update frontend-v1 frontend-v2 --image=image:v2  # Change the name
kubectl rolling-update frontend --image=image:v2                 # Update the pods
kubectl rolling-update frontend-v1 frontend-v2 --rollback        # Abort existing
cat pod.json | kubectl replace -f -                              # Replace a pod b

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

# Create a service for a replicated nginx, which serves on port 80 and connects to
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 re

kubectl label pods my-pod new-label=awesome                      # Add a Label
kubectl annotate pods my-pod icon-url=http://goo.gl/XXBTWq       # Add an annotati
kubectl autoscale deployment foo --min=2 --max=10                # Auto scale a de
```

Patching Resources

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

# Update a container's image; spec.containers[*].name is required because it's a m
kubectl patch pod valid-pod -p '{"spec":{"containers":[{"name":"kubernetes-serve-h

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

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

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

Editing Resources

The edit any API resource in an editor.

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

Scaling Resources

```
kubectl scale --replicas=3 rs/foo                                 # Scale a replic
kubectl scale --replicas=3 -f foo.yaml                            # Scale a resour
kubectl scale --current-replicas=2 --replicas=3 deployment/mysql  # If the deploym
kubectl scale --replicas=5 rc/foo rc/bar rc/baz                   # Scale multiple
```

Deleting Resources

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

Interacting with running Pods

```
kubectl logs my-pod                                 # dump pod logs (stdout)
kubectl logs my-pod --previous                      # dump pod logs (stdout) for a
kubectl logs my-pod -c my-container                 # dump pod container logs (std
kubectl logs my-pod -c my-container --previous      # dump pod container logs (std
kubectl logs -f my-pod                              # stream pod logs (stdout)
kubectl logs -f my-pod -c my-container              # stream pod container logs (s
kubectl run -i --tty busybox --image=busybox -- sh  # Run pod as interactive shell
kubectl attach my-pod -i                            # Attach to Running Container
kubectl port-forward my-pod 5000:6000               # Listen on port 5000 on the l
kubectl exec my-pod -- ls /                         # Run command in existing pod
kubectl exec my-pod -c my-container -- ls /         # Run command in existing pod
kubectl top pod POD_NAME --containers               # Show metrics for a given pod
```

Interacting with Nodes and Cluster

```
kubectl cordon my-node                                                # Mark my-nc
kubectl drain my-node                                                 # Drain my-n
kubectl uncordon my-node                                              # Mark my-nc
kubectl top node my-node                                              # Show metri
kubectl cluster-info                                                  # Display ac
kubectl cluster-info dump                                             # Dump curre
kubectl cluster-info dump --output-directory=/path/to/cluster-state   # Dump curre

# If a taint with that key and effect already exists, its value is replaced as spe
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 resources
kubectl api-resources --namespaced=false     # All non-namespaced resources
kubectl api-resources -o name                # All resources with simple output (j
kubectl api-resources -o wide                # All resources with expanded (aka "w
kubectl api-resources --verbs=list,get       # All resources that support the "lis
kubectl api-resources --api-group=extensions # All resources in the "extensions" A
```

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

Output format	Description
`-o=jsonpath=<template>`	Print the fields [illegible]d 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.