Kubernetes (kubectl) Cheat Sheet

Kubectl Autocomplete

BASH

```
# Configure autocomplete in current bash shell (bash-completion package must be installed first)
source <(kubectl completion bash)

# Add autocomplete permanently to bash shell
echo "source <(kubectl completion bash)" >> ~/.bashrc

# Shortcut with autocomplete
alias k=kubectl
complete -o default -F __start_kubectl k
```

ZSH

```
# Configure autocomplete in current zsh shell
source <(kubectl completion zsh)

# Add autocomplete permanently to zsh shell
echo '[[ $commands[kubectl] ]] && source <(kubectl completion zsh)' >> ~/.zshrc
```

Note: Use `-A` as a shortcut for `--all-namespaces`

Kubectl Context and Configuration

View Configuration

```
# Show merged kubeconfig settings
kubectl config view

# Use multiple kubeconfig files simultaneously
KUBECONFIG=~/.kube/config:~/.kube/kubconfig2 kubectl config view

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

# Display first user
kubectl config view -o jsonpath='{.users[].name}'

# Get list of users
kubectl config view -o jsonpath='{.users[*].name}'
```

Managing Contexts

```
# List contexts
kubectl config get-contexts

# Show current context
kubectl config current-context

# Set default context
kubectl config use-context my-cluster-name

# Set cluster entry in kubeconfig
kubectl config set-cluster my-cluster-name

# Configure proxy URL
kubectl config set-cluster my-cluster-name --proxy-url=my-proxy-url

# Add new cluster with basic authentication
kubectl config set-credentials kubeuser/foo.kubernetes.com --username=kubeuser --password=kubepassword

# Save namespace permanently for subsequent kubectl commands
kubectl config set-context --current --namespace=ggckad-s2

# Define context with username and namespace
kubectl config set-context gce --user=cluster-admin --namespace=foo \
  && kubectl config use-context gce

# Remove user
kubectl config unset users.foo
```

Context/Namespace Aliases

```
# Shortcut for setting/showing context (bash-compatible shells)
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 | grep namespace | c
```

Creating Objects

Basic Creation Methods

```
# Create resources from file
kubectl apply -f ./my-manifest.yaml
kubectl apply -f ./my1.yaml -f ./my2.yaml
kubectl apply -f ./dir
kubectl apply -f https://git.io/vPieo

# Create a deployment
kubectl create deployment nginx --image=nginx

# Create a Job
kubectl create job hello --image=busybox:1.28 -- echo "Hello World"

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

# Get pod manifest documentation
kubectl explain pods

# Create multiple objects from 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

# Create a secret
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
```

Viewing and Locating Resources

Basic Get Commands

```
# List services in namespace
kubectl get services

# List pods across all namespaces
kubectl get pods --all-namespaces

# List pods with more details
kubectl get pods -o wide

# List specific deployment
kubectl get deployment my-dep

# Get pod YAML
kubectl get pod my-pod -o yaml
```

Detailed Descriptions

```
# Describe nodes and pods
kubectl describe nodes my-node
kubectl describe pods my-pod
```

Advanced Filtering and Sorting

```
 # Sort services by name
kubectl get services --sort-by=.metadata.name

# Sort pods by restart count
kubectl get pods --sort-by='.status.containerStatuses[0].restartCount'

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

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

# Retrieve specific ConfigMap key
kubectl get configmap myconfig -o jsonpath='{.data.ca\.crt}'

# Get nodes without control plane
kubectl get node --selector='!node-role.kubernetes.io/control-plane'

# Get running pods
kubectl get pods --field-selector=status.phase=Running

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

# Show labels
kubectl get pods --show-labels

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

# Decode secrets
kubectl get secret my-secret -o go-template='{{range $k,$v := .data}}{{"### "}}{{$k}}{{"\n"}}{{$v|base64decode}}{{"\n\n"}}{{end}}'

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

# List warning events
kubectl events --types=Warning

# Compare cluster state with manifest
kubectl diff -f ./my-manifest.yaml
```

Updating Resources

Deployment Updates

```
# Update container image
kubectl set image deployment/frontend www=image:v2

# Check rollout history
kubectl rollout history deployment/frontend

# Undo rollout
kubectl rollout undo deployment/frontend

# Rollback to specific revision
kubectl rollout undo deployment/frontend --to-revision=2

# Track rollout status
kubectl rollout status -w deployment/frontend

# Restart deployment
kubectl rollout restart deployment/frontend
```

Patching Resources

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

# Update container image
kubectl patch pod valid-pod -p '{"spec":{"containers":[{"name":"kubernetes-serve-hostname","image":"new image"}]}}'

# Update using JSON patch
kubectl patch pod valid-pod --type='json' -p='[{"op": "replace", "path": "/spec/containers/0/image", "value":"new image"}]'

# Remove liveness probe
kubectl patch deployment valid-deployment --type json -p='[{"op": "remove", "path": "/spec/template/spec/containers/0/livenessProbe"

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

Editing Resources

```
# Edit service
kubectl edit svc/docker-registry

# Use alternative editor
KUBE_EDITOR="nano" kubectl edit svc/docker-registry
```

Scaling Resources

```
# Scale ReplicaSet
kubectl scale --replicas=3 rs/foo
kubectl scale --replicas=3 -f foo.yaml

# Conditional scaling
kubectl scale --current-replicas=2 --replicas=3 deployment/mysql

# Scale multiple controllers
kubectl scale --replicas=5 rc/foo rc/bar rc/baz
```

Deleting Resources

```
# Delete from file
kubectl delete -f ./pod.json

# Immediate pod deletion
kubectl delete pod unwanted --now

# Delete multiple resources
kubectl delete pod,service baz foo

# Delete by label
kubectl delete pods,services -l name=myLabel

# Delete all in namespace
kubectl -n my-ns delete pod,svc --all
```

Interacting with Pods

Logging

```
# Pod logs
kubectl logs my-pod
kubectl logs -l name=myLabel
kubectl logs my-pod --previous
kubectl logs my-pod -c my-container
kubectl logs -f my-pod
```

Executing Commands

```
# Interactive shell
kubectl run -i --tty busybox --image=busybox:1.28 -- sh

# Run pod in namespace
kubectl run nginx --image=nginx -n mynamespace

# Attach to container
kubectl attach my-pod -i

# Port forwarding
kubectl port-forward my-pod 5000:6000

# Execute command
kubectl exec my-pod -- ls /
kubectl exec --stdin --tty my-pod -- /bin/sh

# Show pod metrics
kubectl top pod POD_NAME --containers
```

File Operations

```
# Copy files to/from containers
kubectl cp /tmp/foo_dir my-pod:/tmp/bar_dir
kubectl cp /tmp/foo my-pod:/tmp/bar -c my-container

# Alternative tar method
tar cf - /tmp/foo | kubectl exec -i -n my-namespace my-pod -- tar xf - -C /tmp/bar
```

Cluster Interactions

```
# Node management
kubectl cordon my-node
kubectl drain my-node
kubectl uncordon my-node

# Cluster information
kubectl cluster-info
kubectl cluster-info dump
kubectl cluster-info dump --output-directory=/path/to/cluster-state

# Node taints
kubectl get nodes -o='custom-columns=NodeName:.metadata.name,TaintKey:.spec.taints[*].key,TaintValue:.spec.taints[*].value,TaintEffe
```

Resource Types and API Resources

```
# List all resource types
kubectl api-resources

# Namespaced resources
kubectl api-resources --namespaced=true

# Non-namespaced resources
kubectl api-resources --namespaced=false

# Resources by API group
kubectl api-resources --api-group=extensions
```

Output Formatting

```
# Custom columns
kubectl get pods -A -o=custom-columns='DATA:spec.containers[*].image'
kubectl get pods --namespace default --output=custom-columns="NAME:.metadata.name,IMAGE:.spec.containers[*].image"

# Output formats
-o=json            # JSON format
-o=yaml            # YAML format
-o=wide            # Detailed text
-o=name            # Resource name only
-o=custom-columns  # Custom column format
```

Verbosity Levels

- `--v=0` : Operator-level information
- `--v=1` : Default log level
- `--v=2` : Recommended system information
- `--v=3` : Extended change information
- `--v=4` : Debug level details
- `--v=5` : Trace level verbosity
- `--v=6` : Display requested resources
- `--v=7` : Display HTTP request headers
- `--v=8` : Display HTTP request content
- `--v=9` : Display full HTTP request without truncation