



Best Monitoring for Hybrid IT

[Checkmk](#)[Download](#)

Cheatsheets - Kubernetes

[kubect!](#)[k9s](#)[crictl](#)[containerd](#)[kind](#)[helm](#)[harbor](#)[cilium](#)[docker](#)[containers](#)

Cheatsheets - Tools

[tmux](#)[jq](#)[yq](#)[Git](#)[Bazel](#)[Maven](#)[openssl](#)[ceph](#)[brew](#)[cron](#)[terraform](#)[ansible](#)[awk](#)[sed](#)[grep](#)[Networking](#)[ssh](#)[🏠](#) / [en](#) / [cheatsheets](#) / [kubect!](#)

kubect! Cheatsheet

Last Updated: 2024-07-23

Clusters

Get Clusters.

\$ kubect! config get-clusters

Get Cluster Info

\$ kubect! cluster-info

Kubernetes control plane is running at https://127.0.0.1:36397

CoreDNS is running at https://127.0.0.1:36397/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

\$ kubect! cluster-info dump

Specify output columns

\$ kubect! get services -A -o=custom-columns=NAME:.metadata.name,Namespace:.metadata.namespace

API Resources

To see which Kubernetes resources are and aren't in a namespace:

In a namespace

\$ kubect! api-resources --namespaced=true

Not in a namespace

\$ kubect! api-resources --namespaced=false

Check resources

Get a list of Services:

\$ kubect! get services

Check the service accounts:

\$ kubect! -n kube-system get sa

Get pods on a specific node.

\$ kubect! get pods --all-namespaces -o wide --field-selector spec.nodeName=\$NODE

curl

regex

base64

signals

Cheatsheets - Languages

Go

SQL

html

css

Cheatsheets - Formats

markdown

csv

json

xml

yaml

Cheatsheets - Editors

Vim

vscode

Intellij

Cheatsheets - Databases

MySQL

PostgreSQL

Cheatsheets - Shell

Shell Shortcuts

Shell - Tips

Shell - Scripts

Shell - Commands

bash

zsh

```
# Get num of running pods.
$ kubectl get pods -A --field-selector status.phase=Running | wc -l
```

If there are multiple resources with the same name (e.g. **cluster**), add the apigroup to it:

```
$ kubectl get clusters.cluster.x-k8s.io
```

Check resource consumption

```
$ kubectl top node
$ kubectl top pod -A
```

Delete multiple pods

Delete multiple pods by label:

```
$ kubectl delete pods -l app=my-app -n default
```

Delete multiple pods by name:

```
$ kubectl get pods -n $NAMESPACE --no-headers=true | awk '/pattern/{print $1}' | xargs kubectl delete -n $NAMESPACE/
$ kubectl get pods -n $NAMESPACE | grep $PATTERN | awk '{print $2}' | xargs kubectl delete pod -n $NAMESPACE
```

Delete all completed / failed pods

```
$ kubectl --kubeconfig <kubeconfig> delete pods -A --field-selector status.phase=Succeeded
$ kubectl --kubeconfig <kubeconfig> delete pods -A --field-selector status.phase=Failed
```

Force delete all pods in a namespace:

```
$ kubectl delete pod --all --grace-period=0 --force --namespace ui-system
```

Force delete all terminating pods

```
$ kubectl get pods -A | grep Terminating | awk '{print $2 " -n=" $1}' | xargs kubectl delete pod --grace-period=0
```

Storage

Check capacities:

```
$ kubectl describe pv
$ kubectl describe pvc
```

The PV's **status** should be **"Bound"** if it has been successfully allocated to the application.

Check remaining disk space:

```
$ kubectl -n <namespace> exec <pod-name> -- df -ah
```

Plugins

Add the tree plugin to visualize

```
$ kubectl krew install tree
```

How to force restart a pod

```
$ kubectl get pod PODNAME -n NAMESPACE -o yaml | kubectl replace --force -f -
```

Check status

```
$ kubectl get --raw='/readyz?verbose'
```

Who Am I and What Can I Do?

Who Am I?

```
# Show current-context
$ kubectl config current-context

# Check details of the Config
$ kubectl config view

# use a different context
$ kubectl config use-context <context-name>
```

What can i do?

```
# List all
$ kubectl auth can-i --list

# Check to see if I can do everything in my current namespace ("*" means all)
$ kubectl auth can-i '*' '*'

# Check to see if I can create pods in any namespace
$ kubectl auth can-i create pods --all-namespaces

# Check to see if I can list deployments in my current namespace
$ kubectl auth can-i list deployments.extensions
```

Patch

```
$ kubectl patch serviceaccount NAME -n NAMESPACE -p '{"imagePullSecrets": [{"name": "IMAGE_PULL_SECRET_NAME"}]}'
```

```
exec.Command("kubectl", "patch", "serviceaccount",
    "NAME",
    "-n", "NAMESPACE",
    "-p", `{"imagePullSecrets": [{"name": "IMAGE_PULL_SECRET_NAME"}]}`).Run()
```

Search string in resources

```
# use grep, but hard to see which pod it is.
$ kubectl get pod -A -o yaml | grep "something"

# use jq, get pod name.
$ kubectl get pod -A -o json | jq -r '.items[] | select(tostring | contains("something")) | .metadata.name'
```

Check Node Status

e.g. check ephemeral storage

```
$ kubectl get --raw "/api/v1/nodes/${NODE_NAME}/proxy/stats/summary"

# equivalent to
$ curl http://${HOST}:${PORT}/api/v1/nodes/${NODE_NAME}/proxy/stats/summary

# and
$ kubectl get --raw "/api/v1/nodes/${NODE_NAME}/proxy/metrics/resource"
$ kubectl get --raw "/api/v1/nodes/${NODE_NAME}/proxy/metrics/cadvisor"
```

More Examples

```
# get PVs of a namespace
$ kubectl get pv -o json | jq -r '.items[] | select(.spec.claimRef.namespace == "NAMESPACE") | .metadata.name'

# Change the reclaim policies of the persistent volumes to Retain.
$ kubectl patch pv/${NAME} -p "{\"spec\":{\"persistentVolumeReclaimPolicy\":\"Retain\"}}"

# remove a field
$ kubectl patch pv/${NAME} --type json -p '[{"op":"remove","path":"/spec/claimRef"}]';

# Get and decode secret
$ kubectl get secret SECRET_NAME -n NAMESPACE --template="{index .data \"ca.crt\" | base64decode}" > https.crt

# cert is stored in certificate-authority-data in kubeconfig
$ curl $(kubectl config view --minify --output 'jsonpath={..cluster.server}')
# curl: (60) SSL certificate problem: unable to get local issuer certificate

# get cert
$ kubectl config view --minify --raw --output 'jsonpath={..cluster.certificate-authority-data}' | base64 -d > /tr

$ curl --cacert /tmp/kubectl-cacert $(kubectl config view --minify --output 'jsonpath={..cluster.server}')
# should get 403

# Show init containers and normal containers.
$ kubectl get -A pod -o="custom-columns=NAME:.metadata.name,INIT-CONTAINERS:.spec.initContainers[*].name,CONTAINERS:.spec.containers[*].name"

# Get ClusterRoleBinding of a specific subject kind / name.
$ kubectl get clusterrolebindings -o json | jq -r '.items[] | select(.subjects[0].kind=="Group") | select(.subjects[0].name=="group1")'

# get a list of pending pods
$ kubectl get pods --field-selector=status.phase=Pending
```

How to create pods in the cluster for debugging?

Start a pod with alpine Linux:

```
$ kubectl run -i --tty --rm debug --image=alpine --restart=Never -- sh
```

Start an interactive shell in busybox pod in your namespace; dies on exit.

```
$ kubectl run -i --tty busybox --image=busybox --restart=Never -- sh
```

Start a pod with nginx:

```
$ kubectl run nginx --image=nginx --port=80
$ kubectl expose pod nginx --port=80 --type=LoadBalancer
$ kubectl expose pod nginx --port=80 --type=NodePort
```

Start a pod Ubuntu and use **curl** from the pod. (Useful for testing network connectivity from within the cluster.)

```
$ kubectl run -it ubuntu --image=ubuntu -- /bin/bash

# inside the pod:
> apt update && apt install curl
> curl xx.xx.xx.xx
```

How to create a namespace?

```
$ kubectl apply -f - <<EOF
apiVersion: v1
kind: Namespace
metadata:
  name: example-namespace
EOF
```

How to check the x509 certificate?

```
# Check the cert in a Secret
$ kubectl get secret -n foo-system foo-serving-cert -o json | jq -r '.data."ca.crt"' | base64 -d | openssl x509 -text

# Check the cert in a CertificateRequest
$ kubectl get certificaterequest -n foo-system foo-serving-cert-p8795 -o json | jq -r '.status.ca' | base64 -d | openssl x509 -text
```

How to Renew a Certificate?

Certificates are stored by **cert-manager** inside a **Secret**, deleting this **Secret** triggers a certificate renewal.

Note: Delete the **Secret** holding the certificate, not the **Certificate** itself.

```
# Get the name of the Secret:
SECRET_NAME=$(kubectl -n foo-system get Certificate foo-serving-cert -o jsonpath='{.spec.secretName}')

# Delete the Secret to trigger certificate renewal.
$ kubectl --kubeconfig ${KUBECONFIG:?} -n gpc-system delete Secret ${SECRET_NAME}
```

Check all possible clusters

Your **KUBECONFIG** may have multiple contexts:

```
$ kubectl config view -o jsonpath='{ "Cluster name\tServer\n"}{range .clusters[*]}{.name}{ "\t"}{.cluster.server}{ "\n"}'
```

Update configmap

```
# see what changes would be made, returns nonzero returncode if different
$ kubectl get configmap kube-proxy -n kube-system -o yaml | \
sed -e "s/strictARP: false/strictARP: true/" | \
kubectl diff -f - -n kube-system

# actually apply the changes, returns nonzero returncode on errors only
$ kubectl get configmap kube-proxy -n kube-system -o yaml | \
```

```
sed -e "s/strictARP: false/strictARP: true/" | \
kubectl apply -f - -n kube-system
```

Restrict pods to only run on the control-plane nodes

```
$ kubectl patch -n kubevirt kubevirt kubevirt --type merge --patch '{"spec": {"infra": {"nodePlacement": {"nodeSe
```



To restrict the virt-handler pods to only run on nodes with the "region=primary" label:

```
$ kubectl patch -n kubevirt kubevirt kubevirt --type merge --patch '{"spec": {"workloads": {"nodePlacement": {"nc
```



Troubleshooting

Unable to use a TTY - input is not a terminal or the right kind of file

If you see this error when running `kubectl exec -it`, try to remove `-t`.



HackingNote

[PRIVACY POLICY](#) [ABOUT](#) [中文](#)

[Privacy and cookie settings](#)

Managed by Google. Complies with IAB TCF. CMP ID: 300