



KUBERNETES CHEATSHEET

A perfect companion to all
your cluster management needs

Nodes

```
| kubectl get no
```

retrieve a list of all nodes in the current cluster

```
| kubectl get no -o wide
```

retrieve a list of all nodes in the current cluster, with additional details such as the node's IP address and role

```
| kubectl describe no
```

retrieve detailed information about a specific node

```
| kubectl get no -o yaml
```

retrieve a list of all nodes in the current cluster in YAML format

```
| kubectl get node --selector=[label_name]
```

retrieve a list of all nodes with the specified label

```
| kubectl get nodes -o  
jsonpath='{.items[*].status.addresses[?  
(@.type="ExternalIP")].address}'
```

retrieve a list of all node external IP addresses

```
| kubectl top node [node_name]
```

display resource usage statistics for a specific node

Roles

```
| kubectl get roles --all-namespaces
```

display a list of all roles in all namespaces

```
| kubectl get roles --all-namespaces -o yaml
```

display the roles in all namespaces in YAML format

Pods

```
| kubectl get po
```

```
# get a list of pods
```

```
| kubectl get po -o wide
```

```
# get a wide view of pods,  
including Node name and IP
```

```
| kubectl describe po
```

```
# describe all the pods in the  
current namespace
```

```
| kubectl get po --show-labels
```

```
# get list of pods with labels
```

```
| kubectl get po -l app=[app_name]
```

```
# get list of pods with label  
"app" equal to [app_name]
```

```
| kubectl get po -o yaml
```

```
# get yaml definition of all the  
pods in the current namespace
```

```
| kubectl get pod [pod_name] -o yaml --export  
> namooffile.yaml
```

```
# save the yaml definition of the  
specific pod in the file
```

```
| kubectl get pods --field-selector  
status.phase=Running
```

```
# get list of pods with the  
status "Running"
```

Viewing Resource Information

Namespaces

```
| kubectl get ns
```

```
# display a list of all namespaces in the current cluster
```

```
| kubectl get ns -o yaml
```

```
# display the namespaces in the cluster in a yaml format
```

```
| kubectl describe ns
```

```
# display detailed information about a namespace in the cluster
```

Deployments

```
| kubectl get deploy
```

```
# display list of all deployments in the current namespace
```

```
| kubectl describe deploy
```

```
# display detailed information about deployments in the current namespace
```

```
| kubectl get deploy -o wide
```

```
# display list of deployments in the current namespace with additional details
```

```
| kubectl get deploy -o yaml
```

```
# display the deployments in the current namespace in yaml format
```


Services

```
| kubectl get svc
```

display a list of all services in the current namespace

```
| kubectl describe svc
```

display detailed information about services in the current namespace

```
| kubectl get svc -o wide
```

display a list of services in the current namespace with additional details

```
| kubectl get svc -o yaml
```

display the services in the current namespace in yaml format

```
| kubectl get svc --show-labels
```

display a list of services in the current namespace including their labels

DaemonSets

```
| kubectl get ds
```

display list of all daemon sets in the current namespace

```
| kubectl get ds --all-namespaces  
| kubectl describe ds [ds_name] -n [ns_name]
```

display detailed information about a daemon set in a specific namespace.

Viewing Resource Information

```
| kubectl get ds [ds_name] -n [ns_name] -o yaml
```

display a daemon set in a specific namespace in YAML format.

Events

```
| kubectl get events
```

display list of events in the current namespace

```
| kubectl get events -n kube-system
```

display a list of events in the kube-system namespace.

```
| kubectl get events -w
```

watch for new events in the current namespace

Logs

```
| kubectl logs [pod_name]
```

display logs of the specified pod

```
| kubectl logs --since=1h [pod_name]
```

display the logs of a pod for the past 1 hour

```
| kubectl logs --tail=20 [pod_name]
```

display the last 20 lines for the logs for a pod

```
| kubectl logs -f -c [container_name]  
[pod_name]
```

follow the logs for a specific container in the pod



Viewing Resource Information

```
| kubectl logs [pod_name] > pod.log
```

save the logs for a pod to a file

Service Accounts

```
| kubectl get sa
```

display a list of service accounts in the current namespace

```
| kubectl get sa -o yaml
```

display a list of service accounts in the current namespace in yaml format.

```
| kubectl get serviceaccounts default -o  
yaml → sa.yaml
```

save the "default" service account in the yaml format to a file

```
| kubectl replace service account default -  
f sa.yaml
```

replace the "default" service account with the contents of a yaml file

ReplicaSets

```
| kubectl get rs
```

display a list of all replica sets in the current namespace

Viewing Resource Information

```
| kubectl describe rs
```

display detailed information about replica sets in the current namespace

```
| kubectl get rs -o wide
```

display a list of replica sets in the current namespace with additional details

```
| kubectl get rs -o yaml
```

display the replica sets in the current namespace in yaml format

Multiple Resources

```
| kubectl get svc, po
```

display list of services and pods in the current namespace

```
| kubectl get deploy, no
```

display list of deployments in the current namespace

```
| kubectl get all
```

display list of all resources in the current namespace

```
| kubectl get all --all-namespaces
```

display list of all resources in all namespaces

Viewing Resource Information

Secrets

```
| kubectl get secrets
```

display a list of secrets in the current namespace

```
| kubectl get secrets --all-namespaces
```

display a list of secrets in all namespaces

```
| kubectl get secrets -o yaml
```

display the secrets in the current namespace in YAML format

ConfigMaps

```
| kubectl get cm
```

display a list of config maps in the current namespace

```
| kubectl get cm --all-namespaces
```

display a list of config maps in all namespaces

```
| kubectl get cm --all-namespaces -o yaml
```

display a list of config maps in yaml format



Viewing Resource Information

Ingress

```
| kubectl get ing
```

```
# display a list of ingresses in  
the current namespace
```

```
| kubectl get ing --all-namespaces
```

```
# display list of all ingresses  
in all namespaces
```

Persistent Volume

```
| kubectl get pvc
```

```
# display a list of persistent  
volume claims in the current  
namespace
```

```
| kubectl describe pvc
```

```
# display list of all persistent  
volume claims in all namespaces
```

StorageClass

```
| kubectl get sc
```

```
# display list of all storage  
classes in the cluster
```

```
| kubectl get sc -o yaml
```

```
# display the storage classes in  
yaml format
```



Viewing Resource Information

API Call

```
| kubectl get --raw /apis/metrics.k8s.io/
```

```
# get raw json data for the  
metrics API
```

Cluster Info

```
| kubectl config
```

```
# view and manage Kubernetes  
cluster
```

```
| kubectl cluster-info
```

```
# display information about the  
kubernetes cluster
```

```
| kubectl get component statuses
```

```
# get the status of the various  
components in the kubernetes  
cluster
```



Changing Resource Attributes

Taint

```
| kubectl taint [node_name] [taint_name]
```

```
# add taint to a node
```

Labels

```
| kubectl label [node_name] disktype=ssd
```

```
# add label to a node
```

```
| kubectl label [pod_name] env=prod
```

```
# add label to a pod
```

Cordon / Uncordon

```
| kubectl cordon [node_name]
```

```
# mark a node as unschedulable
```

```
| kubectl uncordon [node_name]
```

```
# mark a node as schedulable
```

Drain

```
| kubectl drain [node_name]
```

```
# drain a node in preparation for  
maintenance
```

Nodes

```
| kubectl delete node [node_name]
```

```
# delete a node from the cluster
```

```
| kubectl edit node [node_name]
```

```
# edit a node's configuration
```




Namespaces

```
| kubectl delete ns [namespace_name]
```

```
# delete a namespace
```

```
| kubectl edit ns [namespace_name]
```

```
# edit a namespace's  
configuration
```

Deployments

```
| kubectl delete deploy [deploy_name]
```

```
# delete a deployment
```

```
| kubectl edit deploy [deploy_name]
```

```
# edit a deployment's  
configuration
```

```
| kubectl get roles --all-namespaces  
kubectl expose deploy [deploy_name] --port=80 --  
type=NodePort
```

```
# expose a deployment as a  
NodePort service
```

```
| kubectl scale deploy [deploy_name] --  
replicas=5
```

```
# scale a deployment to 5  
replicas
```

Pods

```
| kubectl delete pod [pod_name]
```

```
# delete a pod
```

```
| kubectl edit pod [pod_name]
```

```
# edit a pod's configuration
```


Adding Resources

Creating A Pod

```
| kubectl create -f [name_of_file]
```

create resources from a file

```
| kubectl apply -f [name_of_file]
```

apply changes from a file

```
| kubectl run [pod_name] --image=nginx --  
restart=Never
```

run a single instance of an
nginx container

```
| kubectl run [pod_name] --generator=run-  
pod/v1 --image=nginx
```

run a single instance of nginx
container using "run-pod/v1"
generator

Creating A Service

```
| kubectl create svc nodeport [svc_name]  
--tcp=8080:80
```

create a NodePort service that
exposes TCP port 8080 on the nodes
and maps it to the port 80 in the
pods

Creating A Deployment

```
| kubectl create -f [name_of_file]
```

create resources from a file

```
| kubectl apply -f [name_of_file]
```

apply changes from a file

```
| kubectl create deploy [deploy_name] --  
image=nginx
```

create a deployment with nginx
container





Interactive Pod

```
| kubectl run [pod_name] --image=busybox --rm --it --restart=Never -- sh
```

run a temporary busybox container, open an interactive shell and delete it when it exits

Output YAML to a File

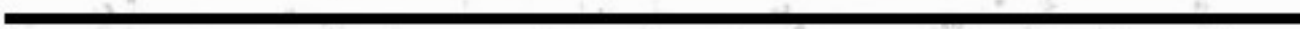
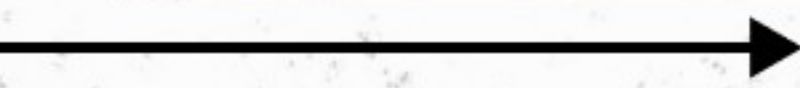
```
| kubectl create deploy [deploy_name] --image nginx --dry-run -o yaml > deploy.yaml
```

create a deployment with an nginx container and output the configuration to a file in YAML format without creating the resources.

```
| kubectl get po [pod_name] -o yaml --export > pod.yaml
```

get a pod and output the configuration to a file in YAML format, including all resource details but ignoring cluster-specific information.

XX



This Content Helpful ?



If you benefit from it, Hit the like



Share your thoughts in the comment



Share with others to learn

@jimmyramani
