# Kubectl Command Cheatsheet

Kubectl is the command line configuration tool for Kubernetes that communicates with a Kubernetes API server. Using Kubectl allows you to create, inspect, update, and delete Kubernetes objects. This cheatsheet will serve as a quick reference to make commands on many common Kubernetes components and resources. You can use the full command for an object on things like pod(s) or the shortcode variation mentioned in parantheses in the heading of each section. They will all generate the same outcome. You'll also want to be sure to follow up most of the commands with the specific <name> of the resource you are managing.

## Cluster Management

Display endpoint information about the master and services in the cluster

```
kubectl cluster-info
```

Display the Kubernetes version running on the client and server

```
kubectl version
```

Get the configuration of the cluster

```
kubectl api-resources
```

List the API versions that are available

```
kubectl api-versions
```

List everything

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

# Daemonsets

**Shortcode = ds**

List one or more daemonsets

```
kubectl get daemonset
```

Edit and update the definition of one or more daemonset

```
kubectl edit daemonset <daemonset_name>
```

Delete a daemonset

```
kubectl create daemonset <daemonset_name>
```

Manage the rollout of a daemonset

```
kubectl rollout daemonset
```

Display the detailed state of daemonsets within a namespace

```
kubectl describe ds <daemonset_name> -n <namespace_name>
```

# Deployments

**Shortcode = deploy**

List one or more deployments

```
kubectl get deployment
```

Display the detailed state of one or more deployments

```
kubectl describe deployment <deployment_name>
```

Edit and update the definition of one or more deployment on the server

```
kubectl create deployment <deployment_name>
```

Delete deployments

```
kubectl delete deployment <deployment_name>
```

See the rollout status of a deployment

```
kubectl rollout status deployment <deployment_name>
```

# Events

**Shortcode = ev**

List recent events for all resources in the system

```
kubectl get events
```

List Warnings only

```
kubectl get events --field-selector type=Warning
```

List events but exclude Pod events

```
kubectl get events --field-selector involvedObject.kind=Node, involvedObject.name=<
```

Filter out normal events from a list of events

```
kubectl get events --field-selector type!=Normal
```

## Logs

Print the logs for a pod

```
kubectl logs <pod_name>
```

Print the logs for the last hour for a pod

```
kubectl logs --since=1h <pod_name>
```

Get the most recent 20 lines of logs

```
kubectl logs --tail=20 <pod_name>
```

Get logs from a service and optionally select which container

```
kubectl logs -f <pod_name>
```

Print the logs for a container in a pod

```
kubectl logs -c <container_name> <pod_name>
```

Output the logs for a pod into a file named 'pod.log'

```
kubectl logs <pod_name> pod.log
```

View the logs for a previously failed pod

```
kubectl logs --previous <pod_name>
```

For logs we also recommend using a tool developed by Johan Haleby called Kubetail. This is a bash script that will allow you to get logs from multiple pods simultaneously. You can learn more about it at its Github repository. Here are some sample commands using Kubetail.

Get logs for all pods named with pod_prefix

```
kubetail <pod_prefix>
```

## Manifest Files

Another option for modifying objects is through Manifest Files. We highly recommend using this method. It is done by using yaml files with all the necessary options for objects configured. We have our yaml files stored in a git repository, so we can track changes and streamline changes.

Apply a configuration to an object by filename or stdin. Overrides the existing configuration.

```
kubectl apply -f manifest_file.yaml
```

Create objects

```
kubectl create -f manifest_file.yaml
```

Create objects in all manifest files in a directory

```
kubectl create -f ./dir
```

Create objects from a URL

```
kubectl create -f ‘url’
```

Delete an object

**Shortcode = ns**

Create namespace <name>

```
kubectl create namespace <namespace_name>
```

List one or more namespaces

```
kubectl get namespace <namespace_name>
```

Display the detailed state of one or more namespace

```
kubectl describe namespace <namespace_name>
```

Delete a namespace

```
kubectl delete namespace <namespace_name>
```

Edit and update the definition of a namespace

```
kubectl edit namespace <namespace_name>
```

Display Resource (CPU/Memory/Storage) usage for a namespace

**Shortcode = no**

Update the taints on one or more nodes

```
kubectl taint node <node_name>
```

List one or more nodes

```
kubectl get node
```

Delete a node or multiple nodes

```
kubectl delete node <node_name>
```

Display Resource usage (CPU/Memory/Storage) for nodes

```
kubectl top node
```

Resource allocation per node

```
kubectl describe nodes | grep Allocated -A 5
```

Pods running on a node

```
kubectl annotate node <node_name>
```

Mark a node as unschedulable

```
kubectl cordon node <node_name>
```

Mark node as schedulable

```
kubectl uncordon node <node_name>
```

Drain a node in preparation for maintenance

```
kubectl drain node <node_name>
```

Add or update the labels of one or more nodes

```
kubectl label node
```

# Pods

**Shortcode = po**

List one or more pods

```
kubectl delete pod <pod_name>
```

Display the detailed state of a pods

```
kubectl describe pod <pod_name>
```

Create a pod

```
kubectl create pod <pod_name>
```

Execute a command against a container in a pod

```
kubectl exec <pod_name> -c <container_name> <command>
```

Get interactive shell on a a single-container pod

```
kubectl exec -it <pod_name> /bin/sh
```

Display Resource usage (CPU/Memory/Storage) for pods

```
kubectl top pod
```

Add or update the label of a pod

```
kubectl label pod <pod_name>
```

# Replication Controllers

**Shortcode = rc**

List the replication controllers

```
kubectl get rc
```

List the replication controllers by namespace

```
kubectl get rc --namespace="<namespace_name>"
```

# ReplicaSets

**Shortcode = rs**

List ReplicaSets

```
kubectl get replicasets
```

Display the detailed state of one or more ReplicaSets

```
kubectl scale --replicas=[x]
```

# Secrets

Create a secret

```
kubectl create secret
```

List secrets

```
kubectl get secrets
```

List details about secrets

```
kubectl describe secrets
```

Delete a secret

```
kubectl delete secret <secret_name>
```

# Services

**Shortcode = svc**

Display the detailed state of a service

```
kubectl describe services
```

Expose a replication controller, service, deployment or pod as a new Kubernetes service

```
kubectl expose deployment [deployment_name]
```

Edit and update the definition of one or more services

```
kubectl edit services
```

## Service Accounts

**Shortcode = sa**

List service accounts

```
kubectl get serviceaccounts
```

Display the detailed state of one or more service accounts

```
kubectl describe serviceaccounts
```

Delete a service account

```
kubectl delete serviceaccount <service_account_name>
```

## StatefulSet

**Shortcode = sts**

List StatefulSet

```
kubectl get statefulset
```

Delete StatefulSet only (not pods)

```
kubectl delete statefulset/[stateful_set_name] --cascade=false
```

## Common Options

In Kubectl you can specify optional flags with commands. Here are some of the most common and useful ones.

-o Output format. For example if you wanted to list all of the pods in ps output format with more information.

```
kubectl get pods -o wide
```

```
kubectl get pods -n=[namespace_name]
```

-f Filename, directory, or URL to files to use to create a resource. For example when creating a pod using data in a file named newpod.json.

```
kubectl create -f ./newpod.json
```

-l Selector to filter on, supports '=', '==', and '!='.

Help for kubectl

-h