**Kubectl Command Cheat sheet**

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.  We've created this cheatsheet as a quick reference to make commands on many common Kubernetes components and resources. You can use the full command for an object, like pod, the plural form (pods) or the shortcode variation we mention in parantheses in the heading of each section. They will all generate the same outcome. You'll need 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 config view

 List the API resources that are available

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 delete daemonset <daemonset\_name>

 Create a new 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 edit deployment <deployment\_name>

 Create one a new deployment

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!=Pod

 Pull events for a single node with a specific name

kubectl get events --field-selector involvedObject.kind=Node, involvedObject.name=<node\_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 <service\_name> [-c <$container>]

 Print the logs for a pod and follow new logs

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.](https://github.com/johanhaleby/kubetail) Here are some sample commands using Kubetail.

Get logs for all pods named with pod\_prefix

kubetail <pod\_prefix>

 Include the most recent 5 minutes of logs

kubetail <pod\_prefix> -s 5m

**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

kubectl delete -f manifest\_file.yaml

**Namespaces**

**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

kubectl top namespace <namespace\_name>

**Nodes**

**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 get pods -o wide | grep <node\_name>

 Annotate 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 get pod

 Delete a pod

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 annotations of a pod

kubectl annotate pod <pod\_name> <annotation>

 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 describe replicasets <replicaset\_name>

Scale a ReplicaSet

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**

List one or more services

kubectl get services

 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

 Replace a service account

kubectl replace serviceaccount

 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

 -n Shorthand for --namespace. For example, if you’d like to list all the Pods in a specific Namespace you would do this command:

kubectl get pods --namespace=[namespace\_name]

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