Kubernetes - Kubectl Commands

Kubectl controls the Kubernetes Cluster. It is one of the key components of Kubernetes which runs on the workstation on any machine when the setup is done. It has the capability to manage the nodes in the cluster.

Kubectl commands are used to interact and manage Kubernetes objects and the cluster. In this chapter, we will discuss a few commands used in Kubernetes via kubectl.

kubectl annotate − It updates the annotation on a resource.

```
$kubectl annotate [--overwrite] (-f FILENAME | TYPE NAME) KEY_1=VAL_1 ...
KEY_N = VAL_N [--resource-version = version]
```

For example,

```
kubectl annotate pods tomcat description = 'my frontend'
```

kubectl api-versions − It prints the supported versions of API on the cluster.

```
$ kubectl api-version;
```

kubectl apply − It has the capability to configure a resource by file or stdin.

```
$ kubectl apply –f <filename>
```

kubectl attach − This attaches things to the running container.

```
$ kubectl attach <pod> –c <container>
$ kubectl attach 123456-7890 -c tomcat-conatiner
```

kubectl autoscale − This is used to auto scale pods which are defined such as Deployment, replica set, Replication Controller.

```
$ kubectl autoscale (-f FILENAME | TYPE NAME | TYPE/NAME) [--min = MINPODS] --
max = MAXPODS [--cpu-percent = CPU] [flags]
$ kubectl autoscale deployment foo --min = 2 --max = 10
```

kubectl cluster-info − It displays the cluster Info.

```
$ kubectl cluster-info
```

kubectl cluster-info dump − It dumps relevant information regarding cluster for debugging and diagnosis.

```
$ kubectl cluster-info dump
$ kubectl cluster-info dump --output-directory = /path/to/cluster-state
```

kubectl config − Modifies the kubeconfig file.

```
$ kubectl config <SUBCOMMAD>
$ kubectl config –-kubeconfig <String of File name>
```

kubectl config current-context − It displays the current context.

```
$ kubectl config current-context
#deploys the current context
```

kubectl config delete-cluster − Deletes the specified cluster from kubeconfig.

```
$ kubectl config delete-cluster <Cluster Name>
```

kubectl config delete-context − Deletes a specified context from kubeconfig.

```
$ kubectl config delete-context <Context Name>
```

kubectl config get-clusters − Displays cluster defined in the kubeconfig.

```
$ kubectl config get-cluster
$ kubectl config get-cluster <Cluser Name>
```

kubectl config get-contexts − Describes one or many contexts.

```
$ kubectl config get-context <Context Name>
```

kubectl config set-cluster − Sets the cluster entry in Kubernetes.

```
$ kubectl config set-cluster NAME [--server = server] [--certificateauthority =
path/to/certificate/authority] [--insecure-skip-tls-verify = true]
```

kubectl config set-context − Sets a context entry in kubernetes entrypoint.

```
$ kubectl config set-context NAME [--cluster = cluster_nickname] [--
user = user_nickname] [--namespace = namespace]
$ kubectl config set-context prod –user = vipin-mishra
```

kubectl config set-credentials − Sets a user entry in kubeconfig.

```
$ kubectl config set-credentials cluster-admin --username = vipin --
password = uXFGweU9l35qcif
```

kubectl config set − Sets an individual value in kubeconfig file.

```
$ kubectl config set PROPERTY_NAME PROPERTY_VALUE
```

kubectl config unset − It unsets a specific component in kubectl.

```
$ kubectl config unset PROPERTY_NAME PROPERTY_VALUE
```

kubectl config use-context − Sets the current context in kubectl file.

```
$ kubectl config use-context <Context Name>
```

kubectl config view

```
$ kubectl config view
$ kubectl config view –o jsonpath='{.users[?(@.name == "e2e")].user.password}'
```

kubectl cp − Copy files and directories to and from containers.

```
$ kubectl cp <Files from source> <Files to Destinatiion>
$ kubectl cp /tmp/foo <some-pod>:/tmp/bar -c <specific-container>
```

kubectl create − To create resource by filename of or stdin. To do this, JSON or YAML formats are accepted.

```
$ kubectl create –f <File Name>
$ cat <file name> | kubectl create –f -
```

In the same way, we can create multiple things as listed using the **create** command along with **kubectl**.

- deployment
- namespace

- quota
- secret docker-registry
- secret
- secret generic
- secret tls
- serviceaccount
- service clusterip
- service loadbalancer
- service nodeport

kubectl delete − Deletes resources by file name, stdin, resource and names.

```
$ kubectl delete –f ([-f FILENAME] | TYPE [(NAME | -l label | --all)])
```

kubectl describe − Describes any particular resource in kubernetes. Shows details of resource or a group of resources.

```
$ kubectl describe <type> <type name>
$ kubectl describe pod tomcat
```

kubectl drain − This is used to drain a node for maintenance purpose. It prepares the node for maintenance. This will mark the node as unavailable so that it should not be assigned with a new container which will be created.

```
$ kubectl drain tomcat –force
```

kubectl edit − It is used to end the resources on the server. This allows to directly edit a resource which one can receive via the command line tool.

```
$ kubectl edit <Resource/Name | File Name)
Ex.
$ kubectl edit rc/tomcat
```

kubectl exec − This helps to execute a command in the container.

```
$ kubectl exec POD <-c CONTAINER > -- COMMAND < args...>
$ kubectl exec tomcat 123-5-456 date
```

kubectl expose − This is used to expose the Kubernetes objects such as pod, replication controller, and service as a new Kubernetes service. This has the capability to expose it via a running container or from a **yaml** file.

```
$ kubectl expose (-f FILENAME | TYPE NAME) [--port=port] [--protocol = TCP|UDP]
[--target-port = number-or-name] [--name = name] [--external-ip = external-ip-ofserv
[--type = type]
$ kubectl expose rc tomcat --port=80 –target-port = 30000
$ kubectl expose –f tomcat.yaml –port = 80 –target-port =
```

kubectl get − This command is capable of fetching data on the cluster about the Kubernetes resources.

```
$ kubectl get [(-o|--output=)json|yaml|wide|custom-columns=...|custom-columnsfile=..
go-template=...|go-template-file=...|jsonpath=...|jsonpath-file=...]
(TYPE [NAME | -l label] | TYPE/NAME ...) [flags]
```

For example,

```
$ kubectl get pod <pod name>
$ kubectl get service <Service name>
```

kubectl logs − They are used to get the logs of the container in a pod. Printing the logs can be defining the container name in the pod. If the POD has only one container there is no need to define its name.

```
$ kubectl logs [-f] [-p] POD [-c CONTAINER]
Example
$ kubectl logs tomcat.
$ kubectl logs –p –c tomcat.8
```

kubectl port-forward − They are used to forward one or more local port to pods.

```
$ kubectl port-forward POD [LOCAL_PORT:]REMOTE_PORT
[...[LOCAL_PORT_N:]REMOTE_PORT_N]
$ kubectl port-forward tomcat 3000 4000
$ kubectl port-forward tomcat 3000:5000
```

kubectl replace − Capable of replacing a resource by file name or **stdin**.

```
$ kubectl replace -f FILENAME
$ kubectl replace –f tomcat.yml
$ cat tomcat.yml | kubectl replace –f -
```

kubectl rolling-update − Performs a rolling update on a replication controller. Replaces the specified replication controller with a new replication controller by updating a POD at a time.

```
$ kubectl rolling-update OLD_CONTROLLER_NAME ([NEW_CONTROLLER_NAME] --
image = NEW_CONTAINER_IMAGE | -f NEW_CONTROLLER_SPEC)
$ kubectl rolling-update frontend-v1 –f freontend-v2.yaml
```

kubectl rollout − It is capable of managing the rollout of deployment.

```
$ Kubectl rollout <Sub Command>
$ kubectl rollout undo deployment/tomcat
```

Apart from the above, we can perform multiple tasks using the rollout such as −

- rollout history
- rollout pause
- rollout resume
- rollout status
- rollout undo

kubectl run − Run command has the capability to run an image on the Kubernetes cluster.

```
$ kubectl run NAME --image = image [--env = "key = value"] [--port = port] [--
replicas = replicas] [--dry-run = bool] [--overrides = inline-json] [--command] --
[COMMAND] [args...]
$ kubectl run tomcat --image = tomcat:7.0
$ kubectl run tomcat –-image = tomcat:7.0 –port = 5000
```

kubectl scale − It will scale the size of Kubernetes Deployments, ReplicaSet, Replication Controller, or job.

```
$ kubectl scale [--resource-version = version] [--current-replicas = count] --
replicas = COUNT (-f FILENAME | TYPE NAME )
$ kubectl scale –-replica = 3 rs/tomcat
$ kubectl scale –replica = 3 tomcat.yaml
```

kubectl set image − It updates the image of a pod template.

```
$ kubectl set image (-f FILENAME | TYPE NAME)
CONTAINER_NAME_1 = CONTAINER_IMAGE_1 ... CONTAINER_NAME_N = CONTAINER_IMAGE_N
$ kubectl set image deployment/tomcat busybox = busybox ngnix = ngnix:1.9.1
$ kubectl set image deployments, rc tomcat = tomcat6.0 --all
```

kubectl set resources − It is used to set the content of the resource. It updates resource/limits on object with pod template.

```
$ kubectl set resources (-f FILENAME | TYPE NAME) ([--limits = LIMITS & --
requests = REQUESTS]
$ kubectl set resources deployment tomcat -c = tomcat --
limits = cpu = 200m,memory = 512Mi
```

kubectl top node − It displays CPU/Memory/Storage usage. The top command allows you to see the resource consumption for nodes.

```
$ kubectl top node [node Name]
```

The same command can be used with a pod as well.