**Table of content:**

# [Cluster Information Commands](#_Cluster_Information_Commands)

# [Working with Pods](#_Working_with_Pods_1)

# [Working with Deployments](#_Working_with_Deployments_1)

# [Working with ReplicaSets](#_Working_with_ReplicaSets_1)

# [Working with Services](#_Working_with_Services_1)

# [Working with Namespaces](#_Working_with_Namespaces_1)

# [Working with ConfigMaps](#_Working_with_ConfigMaps_1)

# [Working with Secrets](#_Working_with_Secrets_1)

# [Working with Ingress](#_Working_with_Ingress_1)

# [Working with StatefulSets](#_Working_with_StatefulSets_1)

# [Working with DaemonSets](#_Working_with_DaemonSets_1)

# [Working with Jobs and CronJobs](#_Working_with_Jobs_1)

# [Working with Persistent Volumes (PVs) and Persistent Volume Claims (PVCs)](#_Working_with_Persistent_1)

# [Managing Configurations](#_Managing_Configurations_1)

# [Scaling and Rolling Updates](#_Scaling_and_Rolling_1)

# [Accessing the Kubernetes Dashboard](#_Accessing_the_Kubernetes_1)

# [Other Useful Commands](#_Other_Useful_Commands_1)

# [Helm (Package Manager for Kubernetes)](#_Helm_(Package_Manager_1)

# [Cluster Information Commands](#_Cluster_Information_Commands)

1. **Get cluster info**

kubectl cluster-info

This command provides details about your Kubernetes cluster’s services, such as the master API server and core DNS.

1. **Get current context**

kubectl config current-context

Displays the current Kubernetes context. The context is used to reference a particular cluster configuration, including the cluster’s API server and user credentials.

1. **List all contexts**

kubectl config get-contexts

Lists all available contexts in your kubeconfig file, showing details about clusters and namespaces.

1. **Set a context**

kubectl config use-context <context-name>

Switch to a specified context from the list of available contexts. Useful for working with multiple clusters or environments.

1. **Get Kubernetes version**

kubectl version

Displays the Kubernetes client and server versions. This is useful for troubleshooting version mismatches or compatibility issues.

# **[Working with Pods](#_Working_with_Pods)**

1. **List all Pods**

kubectl get pods

Lists all pods in the default namespace. To view Pods in a different namespace, use -n <namespace>.

1. **List Pods in a specific namespace**

kubectl get pods -n <namespace>

Shows all Pods in a particular namespace.

1. **Get detailed information about a Pod**

kubectl describe pod <pod-name>

Displays detailed information about a specific Pod, including events, containers, resource usage, etc.

1. **Get logs from a Pod**

kubectl logs <pod-name>

Retrieve the logs from the specified Pod.

1. **Get logs from a specific container inside a Pod**

kubectl logs <pod-name> -c <container-name>

Useful when a Pod contains multiple containers, and you need to fetch logs for a specific container.

1. **Run a Pod interactively**

kubectl run -i --tty <pod-name> --image=<image-name>

Runs a Pod interactively, allowing you to interact with the container via the terminal.

1. **Delete a Pod**

kubectl delete pod <pod-name>

Deletes a Pod from the cluster.

# **[Working with Deployments](#_Working_with_Deployments)**

1. **Create a deployment**

kubectl create deployment <deployment-name> --image=<image-name>

Creates a new Deployment in the cluster with the specified name and container image.

1. **Get all deployments**

kubectl get deployments

Displays a list of all Deployments in the default namespace.

1. **Describe a deployment**

kubectl describe deployment <deployment-name>

Shows detailed information about a Deployment, including its replica count, images, and other configurations.

1. **Update a deployment (e.g., change image)**

kubectl set image deployment/<deployment-name> <container-name>=<new-image-name>

This command allows you to update the image used in a Deployment without changing the configuration.

1. **Scale a deployment**

kubectl scale deployment <deployment-name> --replicas=<number-of-replicas>

Changes the number of replicas for a given Deployment, allowing you to scale up or down based on load.

1. **Delete a deployment**

kubectl delete deployment <deployment-name>

Deletes the Deployment and all associated resources.

1. **Roll back a deployment to a previous version**

kubectl rollout undo deployment/<deployment-name>

Reverts a Deployment to its previous state, useful when a new update leads to issues.

# **[Working with ReplicaSets](#_Working_with_ReplicaSets)**

1. **Get ReplicaSets**

kubectl get replicasets

Lists all ReplicaSets in the current namespace.

1. **Describe a ReplicaSet**

kubectl describe replicasets <replicaset-name>

Shows detailed information about a ReplicaSet, including the number of Pods, available Pods, and the template used.

1. **Delete a ReplicaSet**

kubectl delete replicasets <replicaset-name>

Deletes a ReplicaSet, ensuring that all managed Pods are also removed.

# **[Working with Services](#_Working_with_Services)**

1. **Create a service**

kubectl expose pod <pod-name> --port=<port>

Exposes a Pod as a Service, making it accessible within the cluster via the specified port.

1. **Get all services**

kubectl get services

Lists all Services within the current namespace.

1. **Describe a service**

kubectl describe service <service-name>

Shows detailed information about a Service, such as its type, selector, and port configurations.

1. **Delete a service**

kubectl delete service <service-name>

Deletes a specified Service from the cluster.

1. **Expose a deployment as a service**

kubectl expose deployment <deployment-name> --port=<port> --target-port=<target-port>

Exposes a Deployment as a Service, where the target port is the port inside the containers.

# **[Working with Namespaces](#_Working_with_Namespaces)**

1. **Get all namespaces**

kubectl get namespaces

Lists all available namespaces in your cluster.

1. **Create a new namespace**

kubectl create namespace <namespace-name>

Creates a new namespace for organizing resources in the cluster.

1. **Delete a namespace**

kubectl delete namespace <namespace-name>

Deletes a namespace and all resources within it.

1. **Set namespace context**

kubectl config set-context --current --namespace=<namespace-name>

Changes the current context to operate within the specified namespace.

# **[Working with ConfigMaps](#_Working_with_ConfigMaps)**

1. **Create a ConfigMap from a file**

kubectl create configmap <configmap-name> --from-file=<file-path>

Creates a ConfigMap from a file. The file content is stored as key-value pairs within the ConfigMap.

1. **Get all ConfigMaps**

kubectl get configmaps

Lists all ConfigMaps in the current namespace.

1. **Describe a ConfigMap**

kubectl describe configmap <configmap-name>

Displays detailed information about a ConfigMap, including its key-value pairs.

1. **Delete a ConfigMap**

kubectl delete configmap <configmap-name>

Deletes a ConfigMap from the cluster.

# **[Working with Secrets](#_Working_with_Secrets)**

1. **Create a secret from a file**

kubectl create secret generic <secret-name> --from-file=<file-path>

Creates a secret from a file. The content is stored as base64-encoded data.

1. **Get all secrets**

kubectl get secrets

Lists all secrets in the current namespace.

1. **Describe a secret**

kubectl describe secret <secret-name>

Provides details about a secret, including its name and type, though the secret values themselves are not displayed in plain text.

1. **Delete a secret**

kubectl delete secret <secret-name>

Removes a secret from the cluster.

# **[Working with Ingress](#_Working_with_Ingress)**

1. **Create an Ingress**

kubectl apply -f <ingress-definition.yaml>

Creates an Ingress resource based on the definition provided in a YAML file. Ingress is used to manage external access to services in the cluster.

1. **Get all Ingress resources**

kubectl get ingress

Lists all Ingress resources.

1. **Describe an Ingress**

kubectl describe ingress <ingress-name>

Displays detailed information about an Ingress, including its backend services and rules for routing traffic.

1. **Delete an Ingress**

kubectl delete ingress <ingress-name>

Deletes an Ingress resource from the cluster.

# **[Working with StatefulSets](#_Working_with_StatefulSets)**

1. **Get all StatefulSets**

kubectl get statefulsets

Lists all StatefulSets. StatefulSets are used for managing stateful applications, ensuring that each pod in the set has a unique identity.

1. **Create a StatefulSet**

kubectl apply -f <statefulset-definition.yaml>

Deploys a StatefulSet based on the configuration provided in a YAML file.

1. **Describe a StatefulSet**

kubectl describe statefulset <statefulset-name>

Provides detailed information about a StatefulSet, including the number of pods and their individual names.

1. **Delete a StatefulSet**

kubectl delete statefulset <statefulset-name>

Deletes a StatefulSet and its associated Pods.

# **[Working with DaemonSets](#_Working_with_DaemonSets)**

1. **Get all DaemonSets**

kubectl get daemonsets

Lists all DaemonSets. DaemonSets ensure that a Pod runs on all (or some) nodes in the cluster.

1. **Create a DaemonSet**

kubectl apply -f <daemonset-definition.yaml>

Creates a DaemonSet based on the definition provided in a YAML file.

1. **Describe a DaemonSet**

kubectl describe daemonset <daemonset-name>

Displays detailed information about a DaemonSet and its associated Pods.

1. **Delete a DaemonSet**

kubectl delete daemonset <daemonset-name>

Deletes a DaemonSet and its Pods from the cluster.

# **[Working with Jobs and CronJobs](#_Working_with_Jobs)**

**Create a Job**  
kubectl create job <job-name> --image=<image-name>  
Creates a new Job that runs a specified container image.

**Get all Jobs**  
kubectl get jobs  
Lists all Jobs in the current namespace.

**Describe a Job**  
kubectl describe job <job-name>  
Shows detailed information about a Job, including its status and pods.

**Delete a Job**  
kubectl delete job <job-name>  
Deletes a specified Job from the cluster.

**Create a CronJob**  
kubectl create cronjob <cronjob-name> --image=<image-name> --schedule="<schedule>"  
Creates a CronJob that runs at the specified schedule.

**Get all CronJobs**  
kubectl get cronjobs  
Lists all CronJobs in the current namespace.

**Describe a CronJob**  
kubectl describe cronjob <cronjob-name>  
Displays detailed information about a CronJob, including its schedule and associated Jobs.

**Delete a CronJob**  
kubectl delete cronjob <cronjob-name>  
Deletes a specified CronJob from the cluster.

# **[Working with Persistent Volumes (PVs) and Persistent Volume Claims (PVCs)](#_Working_with_Persistent)**

**Create a Persistent Volume (PV)**  
kubectl apply -f <pv-definition.yaml>  
Creates a Persistent Volume using the specified YAML definition.

**Get all Persistent Volumes**  
kubectl get pv  
Lists all Persistent Volumes in the cluster.

**Describe a Persistent Volume**  
kubectl describe pv <pv-name>  
Provides detailed information about a Persistent Volume, including capacity and access modes.

**Create a Persistent Volume Claim (PVC)**  
kubectl apply -f <pvc-definition.yaml>  
Creates a Persistent Volume Claim based on the given YAML definition.

**Get all Persistent Volume Claims**  
kubectl get pvc  
Lists all Persistent Volume Claims in the current namespace.

**Describe a Persistent Volume Claim**  
kubectl describe pvc <pvc-name>  
Displays detailed information about a Persistent Volume Claim, including the associated PV.

**Delete a Persistent Volume**  
kubectl delete pv <pv-name>  
Deletes a specified Persistent Volume from the cluster.

**Delete a Persistent Volume Claim**  
kubectl delete pvc <pvc-name>  
Deletes a Persistent Volume Claim from the cluster.

# **[Managing Configurations](#_Managing_Configurations)**

**View the current configuration**  
kubectl config view  
Displays the current configuration of Kubernetes contexts, clusters, and users.

**Set the current namespace**  
kubectl config set-context --current --namespace=<namespace-name>  
Changes the current context to operate within the specified namespace.

**Set a context for a cluster**  
kubectl config set-context <context-name> --cluster=<cluster-name>  
Creates or updates a context for a specified cluster.

**Rename a context**  
kubectl config rename-context <old-context-name> <new-context-name>  
Renames a Kubernetes context.

# **[Scaling and Rolling Updates](#_Scaling_and_Rolling)**

**Scale a deployment**  
kubectl scale deployment <deployment-name> --replicas=<number-of-replicas>  
Changes the number of replicas for a given Deployment, allowing you to scale up or down based on load.

**Roll out a deployment**  
kubectl rollout deploy <deployment-name>  
Deploys changes to a Deployment, such as new images or configurations.

**Roll back a deployment**  
kubectl rollout undo deployment/<deployment-name>  
Reverts a Deployment to its previous state, useful when a new update leads to issues.

**Get rollout status**  
kubectl rollout status deployment/<deployment-name>  
Displays the status of a deployment rollout.

**Perform a rolling update**  
kubectl set image deployment/<deployment-name> <container-name>=<new-image-name>  
Updates the image for a container in the Deployment and performs a rolling update.

# **[Accessing the Kubernetes Dashboard](#_Accessing_the_Kubernetes)**

**Get the Kubernetes Dashboard URL**  
kubectl proxy  
Starts a local proxy to the Kubernetes API server. You can then access the Kubernetes Dashboard at http://localhost:8001/ui.

**Access the Dashboard**  
Navigate to http://localhost:8001/ui in your web browser after starting the proxy with the kubectl proxy command.

**Create an admin user for the Dashboard**  
kubectl apply -f <admin-user-definition.yaml>  
Creates an admin user to access the Dashboard using a YAML file with the necessary RBAC roles.

# **[Other Useful Commands](#_Other_Useful_Commands)**

**Get cluster info**  
kubectl cluster-info  
Provides details about the Kubernetes cluster, including master and DNS services.

**Get all namespaces**  
kubectl get namespaces  
Lists all namespaces in the cluster.

**View logs of a pod**  
kubectl logs <pod-name>  
Retrieves the logs of a Pod.

**Get resources usage**  
kubectl top pod  
Displays the resource usage of all pods in the cluster.

**Get resource usage for nodes**  
kubectl top nodes  
Displays the resource usage of all nodes in the cluster.

# **[Helm (Package Manager for Kubernetes)](#_Helm_(Package_Manager)**

**Install Helm**  
curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash  
Installs Helm on your local machine.

**Add a Helm repository**  
helm repo add <repo-name> <repo-url>  
Adds a Helm repository to your Helm client.

**Install a chart**  
helm install <release-name> <chart-name>  
Installs a chart from a Helm repository.

**Upgrade a release**  
helm upgrade <release-name> <chart-name>  
Upgrades an existing Helm release with a new version of the chart.

**List all Helm releases**  
helm list  
Displays a list of all the releases in the current namespace.

**Uninstall a release**  
helm uninstall <release-name>  
Deletes a Helm release from the cluster.