KUBERNETES COMMANDS CHEAT SHEET

Unlock the Power of Kubernetes with our Comprehensive Commands Cheat Sheet!

KUBERNETES COMMANDS CHEAT SHEET

IMPERATIVE COMMANDS

Imperative commands are instructions a user gives to the Kubernetes cluster to perform tasks. For example, you can use imperative commands to create or delete a pod or to start or stop a container.

PODS AND CONTAINER INTROSPECTION COMMANDS

Pods are the basic unit of scheduling in Kubernetes, which is the process of spreading your application across multiple machines. A pod consists of one or more containers that share network namespaces, storage volumes, and other configuration details. The containers inside a pod are isolated from each other concerning resource usage (CPU/memory), but they can communicate through localhost.

DEBUGGING COMMANDS

You can use debugging commands in Kubernetes to get information about various aspects of your cluster's behavior. For example, you can view the logs for an individual container or pod; list all running pods; check whether a particular service is healthy; find out details about a separate node, and so forth.

The commands to view the logs are listed below:

Command	Description
kubectl logs <pod_name>	To display the logs for a Pod with the given name
kubectl logs --since=1h <pod_name>	To display the logs of last 1 hour for the pod with the given name
kubectl logs --tail-10 <pod_name>	To display the most recent 10 lines of logs
kubectl logs -c <container_name> <pod_name>	To display the logs for a container in a pod with the given names
kubectl logs <pod_name> pod.log	To save the logs into a file named as pod.log

CLUSTER INTROSPECTION COMMANDS

The Kubernetes cluster introspection commands, kubectl, and Kube-API server provide a set of tools for inspecting the state of a Kubernetes cluster. You can use these commands to query for information about objects in your cluster and monitor their condition and that of your running applications.

QUICK COMMANDS

Quick commands are single-line commands that return values from the Kubernetes API server. These are useful for getting quick access to information about your resources without having to write an entire script or shell command.

CHANGING RESOURCE ATTRIBUTES

There are two types of commands for changing resource attributes. They are:

Taints: They ensure that pods are not placed on inappropriate nodes.

Labels: They are used to identify pods.

You can refer to the below commands for changing resource attributes:

Command	Description
kubectl taint <node_name><taint_name>	To update the taints on one or more nodes.
kubectl label pod <pod_name>	To add or update the label of a pod

FORCLUSTER INTROSPECTION

The cluster introspection tool is used to get information about the current state of a Kubernetes cluster. This information includes the nodes' number, status, and storage type.

Command	Description
kubectl version	To get the information related to the version.
kubectl cluster-info	To get the information related to the cluster
kubectl config g view	To get the configuration details
kubectl describe node <node_name>	To get the information about a node

INTERACTING WITH DEPLOYMENTS AND SERVICES

A deployment is a collection of pods that share an identical configuration. You can use a deployment to scale up or down your application by adding or removing pods from the cluster. You create a new deployment version whenever you change your application's configuration.

A service is used to expose one or more Pods as a single endpoint to clients. You can use services to load balance traffic across multiple Pods in your cluster and provide external access to pods that don't have public IP addresses. Services are also used for DNS resolution and name discovery.

Command	Description
kubectl logs deploy/my-deployment	To dump Pod logs for a Deployment (single-container case)
kubectl logs deploy/my-deployment -c my-container	To dump Pod logs for a Deployment (multi-container case)
kubectl port-forward svc/my-service 5000	To listen on local port 5000 and forward to port 5000 on Service backend
kubectl port-forward svc/my-service 5000:my-service-port	To listen on local port 5000 and forward to Service target port with name <my-service-port>
kubectl port-forward deploy/my-deployment 5000:6000	To listen on local port 5000 and forward to port 6000 on a Pod created by <my-deployment>
kubectl exec deploy/my-deployment – ls	To run command in first Pod and first container in Deployment (single- or multi-container cases)

COPY FILES AND DIRECTORIES TO AND FROM CONTAINERS

The kubectl cp command can copy files and directories from one location to another. This command can be used to copy files and directories both outside of a container and inside a container.

Command	Description
kubectl cp /tmp/foo_dir my-pod:/tmp/bar_dir	To copy /tmp/foo_dir local directory to /tmp/bar_dir in a remote pod in the current namespace
kubectl cp /tmp/foo my-pod:/tmp/bar -c my-container	To copy /tmp/foo local file to /tmp/bar in a remote pod in a specific container
kubectl cp /tmp/foo my-namespace/my-pod:/tmp/bar	To copy /tmp/foo local file to /tmp/bar in a remote pod in namespace my-namespace
kubectl cp my-namespace/my-pod:/tmp/foo /tmp/bar	To copy /tmp/foo from a remote pod to /tmp/bar locally

THANK YOU FOR EXPLORING THE KUBERNETES COMMANDS CHEAT SHEET!

May your Kubernetes journey be smooth and your clusters always thrive. Happy orchestrating!