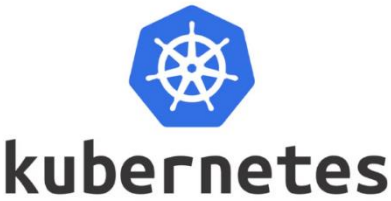

OBJECTS IMPERATIVE

❖ KUBERNETES OBJECTS:

- Kubernetes objects are persistent entities in the Kubernetes system. Kubernetes uses these entities to represent the state of your cluster.
- Specifically, they can describe:
  - What containerized applications are running (and on which nodes)
  - The resources available to those applications
  - The policies around how those applications behave, such as restart policies, upgrades, and fault-tolerance.

OBJECT SPEC AND STATUS:

- Every Kubernetes object includes **two nested object fields** that govern the object's configuration: **object spec & object status.**
- For **objects** that have **a spec,** you have to set this when you create the object, providing a description of the characteristics you want the resource to have: its desired state.
- The **status** describes the current state of the object, supplied and updated by the Kubernetes system and its components.

OBJECT MANAGEMENT:

- **"kubectl",** the command line tool of K8s is used for interacting with Kubernetes clusters.
- The tool is used to deploy the container workloads into production clusters, inspect and manage cluster resources, view logs etc.
- According to K8s it is like a "Swiss Army Knife" of container orchestration and management.
- There are fundamentally two types of K8s object management.
  - Imperative (kubectl commands)
  - Declarative (Writing Manifests)

NOTE: Each has its own advantages and disadvantages, and its highly recommended to manage a K8s object by a single method.

➢ IMPERATIVE COMMANDS:

- Imperative commands are great for learning and interactive experiments, but they don't give you full access to the Kubernetes API. They're more commonly used to create YAML manifests and objects on the go.
- When using imperative commands, a user operates directly on live objects in a cluster.

CREATE OBJECTS:

→ **Creating a new pod from the image nginx:**
$kubectl run mypod –image=nginx

→ **To List Running Pods:**
$kubectl get pods or $kubectl get po

VIEW AN OBJECT:

$kubectl get pods -o wide

$kubectl describe pod mypod

$kubectl logs mypod

→ **Connecting a container in a pod:**

$kubectl exec -it mypod -- bash

→ **Run an instance of the nginx container by creating a Deployment object:**

$kubectl create deployment nginx –image=nginx

$kubectl get deployments (or) $kubectl get deploy

→ **Create a Replicasets:**

$kubectl create deployment mynginx --image=nginx --replicas=3

$kubectl get replicasets (or) $kubectl get rs

$kubectl get deployments

UPDATE OBJECTS:

- **scale:** Horizontally scale a controller to add or remove Pods by updating the replica count of the controller.

  → **To update number of replicas:**

  $kubectl scale deployment mynginx --replicas=5

  $kubectl get deployments

  $kubectl get rs

  → **Using set commands to modify objects:**

  $kubectl set image deployment mynginx mynginx= nginx:1.16.1

  $kubectl describe mynginx

DELETING OBJECTS:

→ **You can use the delete command to delete an object from a cluster:**

$kubectl delete pod mypod

$kubectl delete deployment mynginx

$kubectl get pods

$kubectl get deployments