Kubernetes Commands:

Cheat sheet link: <https://kubernetes.io/docs/reference/kubectl/cheatsheet/>

List the deployed applications in kubernetes: kubectl get deployments

Output: The DESIRED state is showing the configured number of replicas

The CURRENT state show how many replicas are running now

The UP-TO-DATE is the number of replicas that were updated to match the desired (configured) state

The AVAILABLE state shows how many replicas are actually AVAILABLE to the users

kubectl describe deployment <deployment name> : To describe about the particular application that is deployed.

Deploy Sample application : kubectl run kubernetes-bootcamp --image=docker.io/jocatalin/kubernetes-bootcamp:v1 --port=8080

List the nodes: kubectl get nodes

Get the version: kubectl version

Kubernetes created a **Pod** to host your application instance. A Pod is a Kubernetes abstraction that represents a group of one or more application containers (such as Docker or rkt), and some shared resources for those containers.

* **kubectl get** - list resources
* **kubectl describe** - show detailed information about a resource
* **kubectl logs** - print the logs from a container in a pod
* **kubectl exec** - execute a command on a container in a pod

kubectl get pods : List down the pods.

kubectl describe pods : To view what containers are inside that Pod and what images are used to build those container.

kubectl logs <POD\_NAME> : To get the logs of the pod

kubectl exec <POD\_NAME> env : execute commands directly on the container, e.g list down the environment variable.

kubectl exec -ti $POD\_NAME bash : login into the pod’s container through bash shell

curl localhost:8080 : Check whether the application is running. E.g application on local host with port number 8080

exit: To close your container connection type

kubectl get services: List the current services in the cluster

kubectl expose deployment/kubernetes-bootcamp --type="NodePort" --port 8080 : create a new service and expose it to external traffic we’ll use the expose command with NodePort as parameter

kubectl describe services/kubernetes-bootcamp : To find out what port was opened externally (by the NodePort option)

export NODE\_PORT=$(kubectl get services/kubernetes-bootcamp -o go-template=’{{(index .spec.ports 0).nodePort}}’)

echo NODE\_PORT=$NODE\_PORT : Create an environment variable called NODE\_PORT that has as value the Node port+

curl localhost:$NODE\_PORT : Get a response from the server. The Service is exposed.

kubectl describe deployment kubernetes-bootcamp: Describe the application named “kubernetes-bootcamp”. Copy the label from the output.

kubectl get pods -l run=kubernetes-bootcamp : To list down the pods that are created with the application having label “run=kubernetes-bootcamp” (this is the label name for the application kubernetes-bootcamp).

kubectl get services -l run=kubernetes-bootcamp : To list down the services that are created with the application having label “run=kubernetes-bootcamp” (this is the label name for the application kubernetes-bootcamp).

kubectl label pod kubernetes-bootcamp-6959b9c45b-wlpqk app=v1 : To change the label of the pod named “kubernetes-bootcamp-6959b9c45b-wlpqk” to app=v1. Verify it by running this command “kubectl describe pods kubernetes-bootcamp-6959b9c45b-wlpqk”. Also verify it by running this command “kubectl get pods -l app=v1”.

kubectl delete service -l run=kubernetes-bootcamp : Delete the service

curl localhost:$NODE\_PORT : Check with this command that the serice is no longer available. This proves that the app is not reachable anymore from outside of the cluster.

kubectl exec -ti kubernetes-bootcamp-6959b9c45b-wlpqk curl localhost:8080 : The app is still running with a curl inside the pod.

To scale the number of replicas of the deployed application. “kubernetes-bootcamp” is obtained from “kubectl get deployments” command.

kubectl scale deployments/kubernetes-bootcamp --replicas=4

With this the 4 pods are created having different IP. Verify it using this command “kubectl get pods -o wide”.

Let’s check that the Service is load-balancing the traffic. To find out the exposed IP and Port run this command:

kubectl describe services/kubernetes-bootcamp

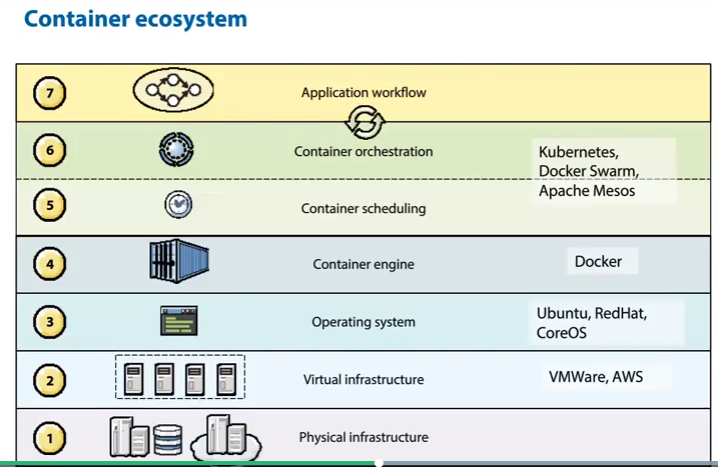
Get the Node\_PORT and run this command : curl localhost:$NODE\_PORT . The output will come from various pods.

Scale down the service to 2 replicas.

kubectl scale deployments/kubernetes-bootcamp --replicas=2

To update the deployed application, e.g update the image, check the type of image with this command “kubectl describe deployments/kubernetes-bootcamp” . It is like “jocatalin/kubernetes-bootcamp:v1” . Change it to “jocatalin/kubernetes-bootcamp:v2” with this command “kubectl set image deployments/kubernetes-bootcamp kubernetes-bootcamp=jocatalin/kubernetes-bootcamp:v2”. Verify it using the command “kubectl describe deployments/kubernetes-bootcamp”. Also, it can be verified using command “kubectl rollout status deployments/kubernetes-bootcamp”.

If something is wrong, the update can be rolled back. E.g : Let’s perform another update, and deploy image tagged as v10 with command “kubectl set image deployments/kubernetes-bootcamp kubernetes-bootcamp=jocatalin/kubernetes-bootcamp:v10” . Check the status with the command “kubectl get pods -o wide”. It will show the error. There is no image called v10 in the repository. Let’s roll back to our previously working version. We’ll use the rollout undo command “kubectl rollout undo deployments/kubernetes-bootcamp”. Verigy it using the command “kubectl get pods -o wide”.



Kubernetes creates pods to host application instances. A pod can include one of more application container and share some resources such as storage and networking information. Containers in a pod share IP address and a port space. Pods represent smallest deployable units in a Kubernetes cluster and used to group containers that are treated as single units. Kubernetes deployment controller monitors the application instances. If a pod is deleted then Kubernetes creates another pod to help running application instances. Kube-proxy directs network traffic on the node.

