**Kubernetes Fundaments using Imperative Approach using kubectl**

S.No Topic Name

01. Pods

02. ReplicaSets

03. Deployments

04. Services

# Kubernetes - PODs

## Step-01: PODs Introduction

* What is a POD ?
* What is a Multi-Container POD?

## Step-02: PODs Demo

### Get Worker Nodes Status

* Verify if kubernetes worker nodes are ready.

# Configure Cluster Creds (kube config) for Azure AKS Clusters

az aks get-credentials --resource-group aks-rg1 --name aksdemo1

# Get Worker Node Status

kubectl get nodes

# Get Worker Node Status with wide option

kubectl get nodes -o wide

### Create a Pod

* Create a Pod

# Template

kubectl run <desired-pod-name> --image <Container-Image>

# Replace Pod Name, Container Image

kubectl run my-first-pod --image vikranthdevops18/9to10batch

### List Pods

* Get the list of pods

# List Pods

kubectl get pods

# Alias name for pods is po

kubectl get po

### List Pods with wide option

* List pods with wide option which also provide Node information on which Pod is running

kubectl get pods -o wide

### What happened in the backgroup when above command is run?

1. Kubernetes created a pod
2. Pulled the docker image from docker hub
3. Created the container in the pod
4. Started the container present in the pod

### Describe Pod

* Describe the POD, primarily required during troubleshooting.
* Events shown will be of a great help during troubleshooting.

# To get list of pod names

kubectl get pods

# Describe the Pod

kubectl describe pod <Pod-Name>

kubectl describe pod my-first-pod

### Access Application

* Currently we can access this application only inside worker nodes.
* To access it externally, we need to create a **NodePort or Load Balancer Service**.
* **Services** is one very very important concept in Kubernetes.

### Delete Pod

# To get list of pod names

kubectl get pods

# Delete Pod

kubectl delete pod <Pod-Name>

kubectl delete pod my-first-pod

## Step-03: Load Balancer Service Introduction

* What are Services in k8s?
* What is a Load Balancer Service?
* How it works?

## Step-04: Demo - Expose Pod with a Service

* Expose pod with a service (Load Balancer Service) to access the application externally (from internet)
* **Ports**
  + **port:** Port on which node port service listens in Kubernetes cluster internally
  + **targetPort:** We define container port here on which our application is running.
* Verify the following before LB Service creation
  + Azure Standard Load Balancer created for Azure AKS Cluster
    - Frontend IP Configuration
    - Load Balancing Rules
  + Azure Public IP

# Create a Pod

kubectl run <desired-pod-name> --image <Container-Image>

kubectl run my-first-pod --image vikranthdevops18/9to10batch

# Expose Pod as a Service

kubectl expose pod <Pod-Name> --type=LoadBalancer --port=80 --name=<Service-Name>

kubectl expose pod my-first-pod --type=LoadBalancer --port=80 --name=my-first-service

# Get Service Info

kubectl get service

kubectl get svc

# Describe Service

kubectl describe service my-first-service

# Access Application

http://<External-IP-from-get-service-output>

* Verify the following after LB Service creation
  + Azure Standard Load Balancer created for Azure AKS Cluster
    - Frontend IP Configuration
    - Load Balancing Rules
  + Azure Public IP
* View the resources in Azure AKS Cluster - Resources section from Azure Portal Management Console

## Step-05: Interact with a Pod

### Verify Pod Logs

# Get Pod Name

kubectl get po

# Dump Pod logs

kubectl logs <pod-name>

kubectl logs my-first-pod

# Stream pod logs with -f option and access application to see logs

kubectl logs <pod-name>

kubectl logs -f my-first-pod

* **Important Notes**
  + Refer below link and search for **Interacting with running Pods** for additional log options
  + Troubleshooting skills are very important. So please go through all logging options available and master them.
  + **Reference:** <https://kubernetes.io/docs/reference/kubectl/cheatsheet/>

### Connect to Container in a POD

* **Connect to a Container in POD and execute commands**

# Connect to Nginx Container in a POD

kubectl exec -it <pod-name> -- /bin/bash

kubectl exec -it my-first-pod -- /bin/bash

# Execute some commands in Nginx container

ls

cd /usr/share/nginx/html

cat index.html

exit

* **Running individual commands in a Container**

kubectl exec -it <pod-name> -- env

# Sample Commands

kubectl exec -it my-first-pod -- env

kubectl exec -it my-first-pod -- ls

kubectl exec -it my-first-pod -- cat /usr/share/nginx/html/index.html

## Step-06: Get YAML Output of Pod & Service

### Get YAML Output

# Get pod definition YAML output

kubectl get pod my-first-pod -o yaml

# Get service definition YAML output

kubectl get service my-first-service -o yaml

## Step-07: Clean-Up

# Get all Objects in default namespace

kubectl get all

# Delete Services

kubectl delete svc my-first-service

# Delete Pod

kubectl delete pod my-first-pod

# Get all Objects in default namespace

kubectl get all

# Kubernetes - ReplicaSets

## Step-01: Introduction to ReplicaSets

* What are ReplicaSets?
* What is the advantage of using ReplicaSets?

## Step-02: Create ReplicaSet

### Create ReplicaSet

* Create ReplicaSet

kubectl create -f replicaset-demo.yml

* **replicaset-demo.yml**

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: my-helloworld-rs

labels:

app: my-helloworld

spec:

replicas: 3

selector:

matchLabels:

app: my-helloworld

template:

metadata:

labels:

app: my-helloworld

spec:

containers:

- name: my-helloworld-app

image: vikranthdevops18/9to10batch

### List ReplicaSets

* Get list of ReplicaSets

kubectl get replicaset

kubectl get rs

### Describe ReplicaSet

* Describe the newly created ReplicaSet

kubectl describe rs/<replicaset-name>

kubectl describe rs/my-helloworld-rs

[or]

kubectl describe rs my-helloworld-rs

### List of Pods

* Get list of Pods

#Get list of Pods

kubectl get pods

kubectl describe pod <pod-name>

# Get list of Pods with Pod IP and Node in which it is running

kubectl get pods -o wide

### Verify the Owner of the Pod

* Verify the owner reference of the pod.
* Verify under **"name"** tag under **"ownerReferences"**. We will find the replicaset name to which this pod belongs to.

kubectl get pods <pod-name> -o yaml

kubectl get pods my-helloworld-rs-c8rrj -o yaml

## Step-03: Expose ReplicaSet as a Service

* Expose ReplicaSet with a service (Load Balancer Service) to access the application externally (from internet)

# Expose ReplicaSet as a Service

kubectl expose rs <ReplicaSet-Name> --type=LoadBalancer --port=80 --target-port=8080 --name=<Service-Name-To-Be-Created>

kubectl expose rs my-helloworld-rs --type=LoadBalancer --port=80 --target-port=8080 --name=my-helloworld-rs-service

# Get Service Info

kubectl get service

kubectl get svc

* **Access the Application using External or Public IP**

http://<External-IP-from-get-service-output>/hello

## Step-04: Test Replicaset Reliability or High Availability

* Test how the high availability or reliability concept is achieved automatically in Kubernetes
* Whenever a POD is accidentally terminated due to some application issue, ReplicaSet should auto-create that Pod to maintain desired number of Replicas configured to achive High Availability.

# To get Pod Name

kubectl get pods

# Delete the Pod

kubectl delete pod <Pod-Name>

# Verify the new pod got created automatically

kubectl get pods (Verify Age and name of new pod)

## Step-05: Test ReplicaSet Scalability feature

* Test how scalability is going to seamless & quick
* Update the **replicas** field in **replicaset-demo.yml** from 3 to 6.

# Before change

spec:

replicas: 3

# After change

spec:

replicas: 6

* Update the ReplicaSet

# Apply latest changes to ReplicaSet

kubectl replace -f replicaset-demo.yml

# Verify if new pods got created

kubectl get pods -o wide

## Step-06: Delete ReplicaSet & Service

### Delete ReplicaSet

# Delete ReplicaSet

kubectl delete rs <ReplicaSet-Name>

# Sample Commands

kubectl delete rs/my-helloworld-rs

[or]

kubectl delete rs my-helloworld-rs

# Verify if ReplicaSet got deleted

kubectl get rs

### Delete Service created for ReplicaSet

# Delete Service

kubectl delete svc <service-name>

# Sample Commands

kubectl delete svc my-helloworld-rs-service

[or]

kubectl delete svc/my-helloworld-rs-service

# Verify if Service got deleted

kubectl get svc

# Kubernetes - Deployment

## Step-01: Introduction to Deployments

* What is a Deployment?
* What all we can do using Deployment?
* Create a Deployment
* Scale the Deployment
* Expose the Deployment as a Service

## Step-02: Create Deployment

* Create Deployment to rollout a ReplicaSet
* Verify Deployment, ReplicaSet & Pods

# Create Deployment

kubectl create deployment <Deplyment-Name> --image=<Container-Image>

kubectl create deployment my-first-deployment --image= vikranthdevops18/9to10batch

# Verify Deployment

kubectl get deployments

kubectl get deploy

# Describe Deployment

kubectl describe deployment <deployment-name>

kubectl describe deployment my-first-deployment

# Verify ReplicaSet

kubectl get rs

# Verify Pod

kubectl get po

## Step-03: Scaling a Deployment

* Scale the deployment to increase the number of replicas (pods)

# Scale Up the Deployment

kubectl scale --replicas=10 deployment/<Deployment-Name>

kubectl scale --replicas=10 deployment/my-first-deployment

# Verify Deployment

kubectl get deploy

# Verify ReplicaSet

kubectl get rs

# Verify Pods

kubectl get po

# Scale Down the Deployment

kubectl scale --replicas=2 deployment/my-first-deployment

kubectl get deploy

## Step-04: Expose Deployment as a Service

* Expose **Deployment** with a service (LoadBalancer Service) to access the application externally (from internet)

# Expose Deployment as a Service

kubectl expose deployment <Deployment-Name> --type=LoadBalancer --port=80 --target-port=80 --name=<Service-Name-To-Be-Created>

kubectl expose deployment my-first-deployment --type=LoadBalancer --port=80 --target-port=80 --name=my-first-deployment-service

# Get Service Info

kubectl get svc

* **Access the Application using Public IP**

http://<External-IP-from-get-service-output>

# Kubernetes - Services

## Step-01: Introduction to Services

* **Service Types**
  1. ClusterIp
  2. NodePort
  3. LoadBalancer
  4. ExternalName
  5. Ingress
* We are going to look in to ClusterIP and LoadBalancer Service in this section with a detailed example.
* LoadBalancer Type is primarily for cloud providers and it will differ cloud to cloud, so we will do it accordingly (per cloud basis)
* ExternalName doesn't have Imperative commands and we need to write YAML definition for the same, so we will look in to it as and when it is required in our course.

## Step-02: ClusterIP Service - Backend Application Setup

* Create a deployment for Backend Application (Spring Boot REST Application)
* Create a ClusterIP service for load balancing backend application.

# Create Deployment for Backend Rest App

kubectl create deployment my-backend-rest-app --image= vikranthdevops18/9to10batch

kubectl get deploy

# Create ClusterIp Service for Backend Rest App

kubectl expose deployment my-backend-rest-app --port=8080 --target-port=8080 --name=my-backend-service

kubectl get svc

Observation: We don't need to specify "--type=ClusterIp" because default setting is to create ClusterIp Service.

* **Important Note:** If backend application port (Container Port: 8080) and Service Port (8080) are same we don't need to use **--target-port=8080** but for avoiding the confusion i have added it. Same case applies to frontend application and service.
* **Backend HelloWorld Application Source** [kube-helloworld](https://github.com/stacksimplify/kubernetes-fundamentals/tree/master/00-Docker-Images/02-kube-backend-helloworld-springboot/kube-helloworld)

## Step-03: LoadBalancer Service - Frontend Application Setup

* We have implemented **LoadBalancer Service** multiple times so far (in pods, replicasets and deployments), even then we are going to implement one more time to get a full architectural view in relation with ClusterIp service.
* Create a deployment for Frontend Application (Nginx acting as Reverse Proxy)
* Create a LoadBalancer service for load balancing frontend application.
* **Nginx Conf File**

server {

listen 80;

server\_name localhost;

location / {

# Update your backend application Kubernetes Cluster-IP Service name and port below

# proxy\_pass http://<Backend-ClusterIp-Service-Name>:<Port>;

proxy\_pass http://my-backend-service:8080;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root /usr/share/nginx/html;

}

}

# Create Deployment for Frontend Nginx Proxy

kubectl create deployment my-frontend-nginx-app --image= vikranthdevops18/9to10batch

kubectl get deploy

# Create LoadBalancer Service for Frontend Nginx Proxy

kubectl expose deployment my-frontend-nginx-app --type=LoadBalancer --port=80 --target-port=80 --name=my-frontend-service

kubectl get svc

# Get Load Balancer IP

kubectl get svc

http://<External-IP-from-get-service-output>/hello

# Scale backend with 10 replicas

kubectl scale --replicas=10 deployment/my-backend-rest-app

# Test again to view the backend service Load Balancing

http://<External-IP-from-get-service-output>/hello