SESSION - 66

**Helm Charts**

**=============**

Containerization is two things

1. Build the image

2. Run the image

--> if image is ready that image how to run for kubernates to manifest through inform. Image name,version,port,services these and all configuration of images.

--> redis rabbitmq --> this is open source image already they builded.

--> opensource manifes files -- this files build first itself and push to open source inside public

1. Install applications into kubernates, it is like a package manager in kubernates.

Helm -- how to run manifest files

1. templatise kubernetes manifest files

This two advantages there for helm

--> install helm

--> Helm now has an installer script that will automatically grab the latest version of Helm and [install it locally](https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3" \t "https://helm.sh/docs/intro/install/_blank).

--> You can fetch that script, and then execute it locally. It's well documented so that you can read through it and understand what it is doing before you run it.

--> $ curl -fsSL -o get\_helm.sh <https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3>

--> $ chmod 700 get\_helm.sh

--> $ ./get\_helm.sh

--> helm version

The Chart File Structure

A chart is organized as a collection of files inside of a directory. The directory name is the name of the chart (without versioning information). Thus, a chart describing WordPress would be stored in a wordpress/ directory.

Inside of this directory, Helm will expect a structure that matches this:

wordpress/ Chart.yaml # A YAML file containing information about the chart LICENSE # OPTIONAL: A plain text file containing the license for the chart README.md # OPTIONAL: A human-readable README file

values.yaml # The default configuration values for this chart

values.schema.json # OPTIONAL: A JSON Schema for imposing a structure on the values.yaml file

charts/ # A directory containing any charts upon which this chart depends. crds/ # Custom Resource Definitions

templates/ # A directory of templates that, when combined with values, # will generate valid Kubernetes manifest files.

templates/NOTES.txt # OPTIONAL: A plain text file containing short usage notes

Helm reserves use of the charts/, crds/, and templates/ directories, and of the listed file names. Other files will be left as they are.

Chart.yaml

===========

{{ .Values.deployment.replicas }}

**Helm Charts**

**------------------**

**Chart.yaml**

apiVersion: v2

name: nginx

version: 1.0.0 # chart version

description: nginx helm chart description

appVersion: 1.0.1 # application version

**templates/deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx

# deployment labels

labels:

purpose: deployment-demo

project: roboshop

app: nginx

spec:

replicas: {{ .Values.deployment.replicas }}

# These are the labels replica set use to create pod replicas, this should match pod labels

selector:

matchLabels:

purpose: deployment-demo

project: roboshop

app: nginx

# This is pod definition

template:

metadata:

labels:

purpose: deployment-demo

project: roboshop

app: nginx

spec:

containers:

- name: nginx

image: nginx:{{ .Values.deployment.imageVersion }}

**templates/service.yaml**

apiVersion: v1

kind: Service

metadata:

name: nginx-lb

labels:

purpose: service-lb-demo

spec:

type: LoadBalancer

selector:

purpose: deployment-demo

project: roboshop

app: nginx

ports:

- protocol: TCP

port: 80 # service port

targetPort: {{ .Values.service.targetPort }} # container port

**values.yaml**

deployment:

replicas: 4

imageVersion: stable-otel

service:

targetPort: 80

**--> cd**

**--> git clone <https://github.com/Shan23-hash/helm-charts.git>**

**--> cd helm-charts**

**--> helm install nginx .**

**--> helm list**

**--> kubectl get svc**

**--> kubectl get pods**

**--> helm upgrade nginx .**

**--> kubectl get pods**

**-->** removing old version and coming new version because it is changed to stable card version.

--> **helm list (**check version)

--> these all installed commands

--> you can add description also

**--> helm history**

helm install nginx . --> . represents current directory

helm upgrade nginx . --> updates to latest version

helm list --> list all helm apps running

**--> helm status nginx**

**--> echo $? (**check whether success or failure)

**--> helm histroy nginx**

**--> helm upgrade nginx --description “upgrading to stable-otel version through CICD” .**

**--> helm history nginx**

**--> helm rollback nginx (**rollback privious one)

**--> helm histroy nginx**

**--> kubectl get pods**

**--> helm rollback nginx 1**(1 means rollback to 1)

**--> helm histroy nginx**

**--> kubectl get pods**

**--> helm uninstall nginx**

**--> helm list** (Total deleted)

**--> kubectl get pods**

**--> kubectl get svc**

**-->** these all helm commands,limited.

helm status nginx

1. add repo /etc/yum.repos.d/docker.repo

dnf install docker -y

--> create a k8-roboshop-helm directory

--> add helm repo

install the application charts

1. install drivers

2. add permissions to ec2 instance

3. create storage class

**mongodb/Chart.yaml**

apiVersion: v2

name: mongodb

version: 1.0.0 # chart version

description: mongodb helm chart description

appVersion: v1 # application version

**mongodb/values.yaml**

statefulset:

imageURL: joindevops/mongodb

imageVersion: v1

**mongodb/templates/service-headless.yaml**

apiVersion: v1

kind: Service

metadata:

name: mongodb-headless

namespace: roboshop

labels:

component: mongodb

project: roboshop

tier: database

spec:

clusterIP: None

selector:

component: mongodb

project: roboshop

tier: database

ports:

- protocol: TCP

port: 27017 # service port

targetPort: 27017 # container port

**mongodb/templates/service.yaml**

apiVersion: v1

kind: Service

metadata:

name: mongodb

namespace: roboshop

labels:

component: mongodb

project: roboshop

tier: database

spec:

selector:

component: mongodb

project: roboshop

tier: database

ports:

- protocol: TCP

port: 27017 # service port

targetPort: 27017 # container port

**mongodb/templates/statefulset.yaml**

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: mongodb

namespace: roboshop

labels:

component: mongodb

project: roboshop

tier: database

spec:

selector:

matchLabels:

component: mongodb

project: roboshop

tier: database # has to match .spec.template.metadata.labels

serviceName: "mongodb-headless" # this should be headless service

replicas: 1 # by default is 1

template:

metadata:

labels:

component: mongodb

project: roboshop

tier: database

spec:

containers:

- name: mongodb

image: "{{ .Values.statefulset.imageURL}}:{{ .Values.statefulset.imageVersion }}"

volumeMounts:

- name: mongodb-data

mountPath: /data/db

# This is nothing but PVC, you can directly create here

volumeClaimTemplates:

- metadata:

name: mongodb-data

spec:

accessModes: [ "ReadWriteOnce" ]

storageClassName: "roboshop-ebs"

resources:

requests:

storage: 2Gi

**Helm**

Add the aws-ebs-csi-driver Helm repository.

**--> helm repo add aws-ebs-csi-driver https://kubernetes-sigs.github.io/aws-ebs-csi-driver**

**--> helm repo update**

Install the latest release of the driver.

**--> helm upgrade --install aws-ebs-csi-driver \**

**--> helm repo list**

**--> helm list**

**--> helm list --all**

**--> helm-list --all-namespaces**

**--> --namespace kube-system \**

**--> aws-ebs-csi-driver/aws-ebs-csi-driver**

**--> helm repo bitnami <https://charts.bitnami.com>/bitnami**

**--> helm install my-redis-cluster bitnami/redis-cluster**

**--> kubectl get pods**

**--> helm list**

**--> helm uninstall my-redis-cluster**

Full permissions - select one roboshop-dev --> open IAM role --> add permissions--> attach policies --> type ebs and select --> add permissions.

**01-namespace.yaml**

apiVersion: v1

kind: Namespace

metadata:

name: roboshop

labels:

project: roboshop

environment: dev

--> git push and pull

**ebs-sc.yaml**

apiVersion: storage.k8s.io/v1

kind: StorageClass

metadata:

name: roboshop-ebs

reclaimPolicy: Retain

provisioner: ebs.csi.aws.com

volumeBindingMode: WaitForFirstConsumer # disk will be created when pod is getting created

--> **cd k8-roboshop-helm/**

**--> kubectl apply -f 01-namespace.yaml**

**--> kubectl apply -f ebs-sc.yaml**

**--> kubens roboshop**

**--> cd mongodb/**

**--> helm upgrade --install mongodb --description “first time installation with version v1\* .**

**--> helm list**

**--> kubectl get pods**

**mysql/Chart.yaml**

apiVersion: v2

name: mysql

version: 1.0.0 # chart version

description: mysql helm chart description

appVersion: v1 # application version

**mysql/values.yaml**

statefulset:

imageURL: shan2324/mysql

imageVersion: v1

**mysql/templates/secret.yaml**

apiVersion: v1

kind: Secret

metadata:

name: mysql

labels:

component: mysql

project: roboshop

tier: database

type: Opaque

data:

MYSQL\_ROOT\_PASSWORD: "Um9ib1Nob3BAMQ=="

**mysql/templates/service-headless.yaml**

apiVersion: v1

kind: Service

metadata:

name: mysql-headless

namespace: roboshop

labels:

component: mysql

project: roboshop

tier: database

spec:

clusterIP: None

selector:

component: mysql

project: roboshop

tier: database

ports:

- protocol: TCP

port: 3306 # service port

targetPort: 3306 # container port

**mysql/templates/service.yaml**

apiVersion: v1

kind: Service

metadata:

name: mysql

namespace: roboshop

labels:

component: mysql

project: roboshop

tier: database

spec:

selector:

component: mysql

project: roboshop

tier: database

ports:

- protocol: TCP

port: 3306 # service port

targetPort: 3306 # container port

**mysql/templates/statefulset.yaml**

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: mysql

namespace: roboshop

labels:

component: mysql

project: roboshop

tier: database

spec:

selector:

matchLabels:

component: mysql

project: roboshop

tier: database # has to match .spec.template.metadata.labels

serviceName: "mysql-headless" # this should be headless service

replicas: 2 # by default is 1

template:

metadata:

labels:

component: mysql

project: roboshop

tier: database

spec:

containers:

- name: mysql

image: "{{ .Values.statefulset.imageURL}}:{{ .Values.statefulset.imageVersion }}"

envFrom:

- secretRef:

name: mysql

volumeMounts:

- name: mysql-data

mountPath: /var/lib/mysql

# This is nothing but PVC, you can directly create here

volumeClaimTemplates:

- metadata:

name: mysql-data

spec:

accessModes: [ "ReadWriteOnce" ]

storageClassName: "roboshop-ebs"

resources:

requests:

storage: 2Gi

**-->** push and pull the code

**--> cd ../mysql**

**--> helm upgrade --install mysql .**

**--> kubectl get pods**

**catalogue/Chart.yaml**

apiVersion: v2

name: catalogue

version: 1.0.0 # chart version

description: catalogue helm chart description

appVersion: v1 # application version

**catalogue/values.yaml**

deployment:

imageURL: joindevops/catalogue

imageVersion: v1

hpa:

maxReplicas: 10

utilisation: 75

**catalogue/templates/configmap.yaml**

apiVersion: v1

kind: ConfigMap

metadata:

name: catalogue

namespace: roboshop

labels:

component: catalogue

project: roboshop

tier: app

data:

MONGO: "true"

MONGO\_URL: "mongodb://mongodb:27017/catalogue"

**catalogue/templates/deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: catalogue

namespace: roboshop

# deployment labels

labels:

component: catalogue

project: roboshop

tier: app

spec:

replicas: 1

# These are the labels replica set use to create pod replicas, this should match pod labels

selector:

matchLabels:

component: catalogue

project: roboshop

tier: app

# This is pod definition

template:

metadata:

labels:

component: catalogue

project: roboshop

tier: app

spec:

containers:

- name: catalogue

image: "{{ .Values.deployment.imageURL}}:{{ .Values.deployment.imageVersion }}"

resources:

requests:

cpu: "50m"

memory: "128Mi"

limits:

cpu: "100m"

memory: "256Mi"

envFrom:

- configMapRef:

name: catalogue

**catalogue/templates/hpa.yaml**

apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

name: catalogue

namespace: roboshop

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: catalogue

minReplicas: 1

maxReplicas: {{ .Values.hpa.maxReplicas }}

metrics:

- type: Resource

resource:

name: cpu

target:

type: Utilization

averageUtilization: {{ .Values.hpa.utilisation }}

**catalogue/templates/service.yaml**

apiVersion: v1

kind: Service

metadata:

name: catalogue

namespace: roboshop

labels:

component: catalogue

project: roboshop

tier: app

spec:

selector:

component: catalogue

project: roboshop

tier: app

ports:

- protocol: TCP

port: 8080 # service port

targetPort: 8080 # container port

--> push and pull the code

**--> cd ../catalogue**

**--> helm upgrade --install catalogue .**

**--> kubectl get pods**

**What is helm charts? why we use it?**

**============================**

Helm is a package manager in kubernetes, we can install applications into kubernetes using helm. We can use helm to templatise kubernetes manifest files...