# Kube config

Kubernetes Clusters

Development [Master & Worker 1-2]

Staging [Master & Worker 1-4]

Production [Master & Worker 1-10]

Swami (DevOps) (CKA)

Admin Access to Clusters

**Config File:**

———————

apiVersion: v1

clusters:

- cluster:

name: dev

- cluster:

name: staging

- cluster:

name: prod

users:

- name: swami-admin

certificate:

contexts:

- context:

cluster: dev

user: swami-admin

name: swami-admin@dev

- context:

cluster: staging

user: swami-admin

name: swami-admin@staging

- context:

cluster: prod

user: swami-admin

name: swami-admin@prod

current-context: swami-admin@dev

kind: Config

preferences: {}

**Scenario:**

kubectl config view

Dev, Staging, Prod - Contexts

Connect to Dev Cluster

kubectl config use-context swami-admin@dev

kubectl get nodes

Connect to Prod

kubectl config use-context swami-admin@prod

kubectl get nodes

# Deployment

Pod 01

Container 01 -> app:0.2

Container 02

Pod 02

Container 01 -> app:0.2

Container 02

Pod 03

Container 01 -> app:0.2

Container 02

History:

1 -> app:0.1

2 -> app:0.2

CANARY / Blue Green

CI: Code -> Cont Image

cm1 -> app:0.1

cm2 -> app:0.2 = Success

cm3 -> app:0.3 = Failure

CD: Cont Image -> New pods

app:0.1 -> pods/containers

app:0.2 -> pods/containers

v2rs -> 321

v3rs -> 1

v1rs ->

v2rs -> 123

Scale up -> 5

Scale down -> 1

Uber Stack: App(Java) - DB(mysql)

App - Uber

**feature01**(Book-a-ride) - uber:1.0(app) - 10 Tables(DB Schema)

k8s - deployment(uber) - uber:1.0

**feature02**(share-a-ride) - uber:1.5(app) - 18 Tables(DB Schema)

k8s - deployment(uber) - uber:1.5

k set image deploy/uber uber=uber:1.5

ArgoCD:

Step1) Add new 8Tables[job]

Step2) uber:1.5 [update deployments]

# CronJob

Runs ev sunday

Job01 - 1st Wk sun

Pod01

Job02 - 2nd Wk sun

Pod01

Job03 - 3rd Wk sun

Pod01

Dev develops code

Local

Dev

Staging

Production

App

Code -> cont image

Config

Database: application-dev.properties

DB Name: databse-dev.example.com

Username: simplilearn

Password: swaminathan

Database: application-stag.properties

DB Name: databse-stag.example.com

Username: simplilearn

Password: 27642387453497y

Database: application-prod.properties

DB Name: databse-prod.example.com

Username: simplilearn

Password: &\*@^$&\*#$(\*

Code: -> Imag

Hello $DEMO\_GREETING, Welcome to simplilearn

app.py Swami

Hello Swami, Welc……..

app.py Nathan

Hello Nathan, Welc……..

DevOps -> usr/pwd -> secret(k8s) - Encrypted

Deploy -> Pod -> Container - Decrypted

Frontend deployment(3) -> Redis Master deployment(1)

Redis slave deployment(2)

k8s pod AutoScaling

replica:3 max:5

Threshold

CPU - min:10% and Max:80%

Pod

ContainerA

ContainerB(app) - Running

Probe: - Liveness(container) and Readiness(app)

URL: api/health.html

period: 5sec

ContainerC

# Scheduler

Affinity

if condition is TRUE -> Pod get created

condition(node selectors) -> Node Affinity

condition(pods selectors) -> Pod Affinity

Anti-Affinity

if condition is TRUE -> Pod will **NOT** get created

condition(pod selectors) -> Pod Anti-Affinity

Condition

requiredDuringScheduling = Must have

preferredDuringScheduling = May have

Flight seat map - <https://www.flyporter.com/Content/Images/travel-information/Seating/E195-seat-map.svg?version=9.3.0.1>

## Node Affinity

Passenger A

must:

Leg room

may:

Window

16D - False

14D - True

14C - True

Passenger B

must:

Leg room & Window

may:

[100] Near wings

[50] Close to restroom

17C - False

13D - True

affinity-pod

must:

network: fast

may:

disktype: ssd

Flow 01- N2

N1- disktype: ssd

N2 - network: fast & disktype: ssd

Flow 02- N2

N1- disktype: ssd

N2 - network: fast

## 

## Pod Affinity & Anti-Affinity

Region - Mumbai

AZ 1

Node 01

Pod 01

AZ 2

Node 02

Pod 03

AZ 3

Node 03

Pod 02

App Team Condition:

Redis

- 2pods on same node

if cond is True - Don’t create Pod

Anti-Aff

Webserver

- 2pods on same node

if cond is True - Don’t create Pod

Anti-Aff

- only create, when redis is there

Aff

Req: 3 pods + Anti-Aff

Node1

app=store

Node2

app=store

## Pod Priority

Metaverse - Ws, Insta, FB

Node 01

Node 02

Node 03

Node 04

Node 05

Node 06

Node 07

10th June 10:05:20 PM

Scheduler’s queue:

insta - 200 pods [p-p]

Ws - 100 pods

FB - 50 pods

# Networking

k apply -f deploy.yaml

api -> sch

Sch:

POD 1 <labelA>

POD 2 <labelA>

POD 3 <labelA>

k get pods -l <labelA>

Svc:

<labelA>

POD 1, 2, 3

Scale up - 10

Svc:

<labelA>

POD 1, 2, 3… 10

## Ingress

simplilearn cluster

App1 30091

App2 30092

test/App1 30093

30,000 -> 32,000

Ingress - http://simplilearn:31323

accounts

cart

orders

redis<cluster-ip>

frontend<nodePort>

http://simplilearn:31323/orders -> svc:orders

http://simplilearn:31323/cart -> svc:cart

http://simplilearn:31323/frontend -> 404

ContainerPort:

orders - 8080

cart - 80

accounts - 80

Nginx Ingress Controller <Route>

service01<LB>

pod01

pod02

pod03

service02<LB>

pod01

pod02

pod03

service03<LB>

pod01

pod02

pod03

# RBAC

RBAC - Role based access control

## Simplilearn

Permissions

Start Lab

End Lab

Reset Lab

Extend Lab

Roles

Learner [Start & End Lab]

Trainer [Start & End Lab]

LSM

Support engineer

Lab engineer [Start, End, Reset, Extend Lab]

Sales

Role binding

User(Swami) + Role(Trainer)

User(Nikhil) + Role(LSM)

Group(leaner) + Role(Learner)

www.simplilearn.net -> Login -> Leaner -> Start & End Lab

—-

## Kubernetes

CRUD - Create, Read, Update, Delete

Permissions/Policies:

Create - expose, run

Read - get, describe, logs

Update - apply, set, scale, exec

Delete - delete

Roles: [Permissions + Resources] + NS

Admin

CRUD + Resources(\*) = \*.\*

Developer

RU + Resources(Pod, Deploy, ds, cm, secret, svc)

Operations

RUD + Resources(Pod, Deploy, ds, cm, secret, svc)

Viewer

R + Resources(Pod, Deploy, ds, cm, secret, svc)

RoleBinding: [User/Group + Role] + NS

Swami + Operations

Dev group + Developer

PM + Viewer

Verification / Validation?

Metaverse (Org)

Fb (ns)

Role & Rolebinding -> [Swami + Operations] + Fb

Whatsapp

Insta

kubernetes-dashboard

Role - RUD + Secrets +

Swami?

Delete ns - FB?

ClusterAdmin

NS - CRUD

Nodes

Roles

Cluster - CR & CRB

ns1

Deploy

Pod

cm, secret

Role & RB

ns2

CEP 1

**Deploy the Application Using the Kubernetes Dashboard**

**Steps to be followed:**

1. Getting started with Pods, Services, Deployments

2. Creating and Verifying the Service

1. Creating a token and working on a dashboard
2. Configure the NFS-server for MySQL and WordPress Deployment
3. Setting up the NFS Client side
4. Creating and verifying the PV

7. Creating a secret for MySQL Deployments secret data

8. Creating a configmap for WordPress Deployment to store non-sensitive information

NFS - Network File System

Master:

cd /

sudo mkdir /nfsdata

sudo mkdir /nfsdata1

sudo apt install nfs-kernel-server

sudo vi /etc/exports

#Copy and paste the below contents in the file,

/nfsdata \*(rw,sync,no\_root\_squash)

/nfsdata1 \*(rw,sync,no\_root\_squash)

sudo exportfs -rv

sudo chown nobody:nogroup /nfsdata/

sudo chown nobody:nogroup /nfsdata1/

sudo chmod 777 /nfsdata/

sudo chmod 777 /nfsdata1/

sudo systemctl restart nfs-kernel-server

Worker-Node-1

sudo apt install nfs-common

While creating persistentVolume(pv), use below nfs config

**nfs:**

**server: <master-ip>**

**# Exported path of your NFS server**

**path: "/nfsdata"**

CEP 2

**Backing up the Etcd Cluster Data**

**Steps to be followed:**

1. Backing up the etcd cluster data
2. Creating and verifying the namespaces
3. Generating a certificate and private key in the worker node
4. Upgrading the Kubernetes cluster with the latest version

Generating a certificate and private key

sudo openssl genrsa -out user4.key 2048

sudo openssl req -new -key user4.key -out user4.csr

sudo openssl x509 -req -in user4.csr -CA /etc/kubernetes/pki/ca.crt -CAkey /etc/kubernetes/pki/ca.key -CAcreateserial -out user4.crt -days 500

kubectl config set-credentials user4 --client-certificate=/home/labsuser/role/user4.crt --client-key=/home/labsuser/role/user4.key