257) Introduction

258) Minikube for K8s setup

259) Kops for K8s setup

260) K8s Objects & K8s Documentation

261) Kubeconfig

Kubeconfig file

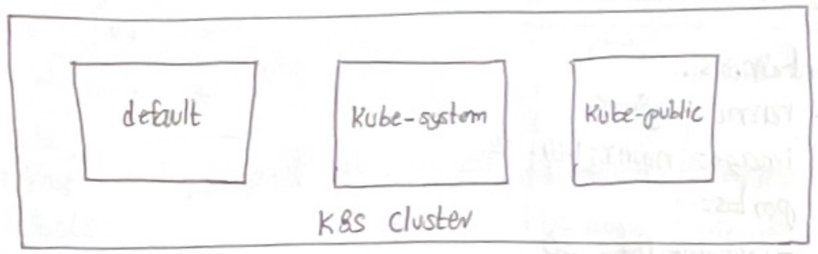
Use Kubeconfig file to organize information about

1) Clusters
2) Users
3) Namespaces
4) Authentication mechanisms

cat .kube/config

cat ~/.kube/config

262) Namespace



kubectl get ns (or) namespaces

We can isolate our environments using namespaces

=> Namespaces should be unique.

IMP Note:- We can gather resources in namespace. If we delete the namespace then our entire resources will delete with a single command.

K get ns (or) namespace

K get all (By default it will show all in default)

K get all --all-namespaces

K get svc -n mynamespace

K create ns kubekart

K run nginx --image=nginx -n kubekart

Create a Pod in particular namespace

pod.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  namespace: kubekart
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

K create -f pod.yml

K get pod -n kubekart

If we delete namespace then entire things will delete

26(s) Pods:-

A Pod is the basic execution unit of K8s application - the smallest and simplest unit in the k8s object model that you create or deploy. A Pod represents process running on your cluster

Pods that ran a single Container:

⇒ The one-container-per-pod model is the most common K8s usecase

⇒ Pod as a wrapper around a single container.

⇒ K8s manages the Pods rather than the containers directly.

Multi Containe POD

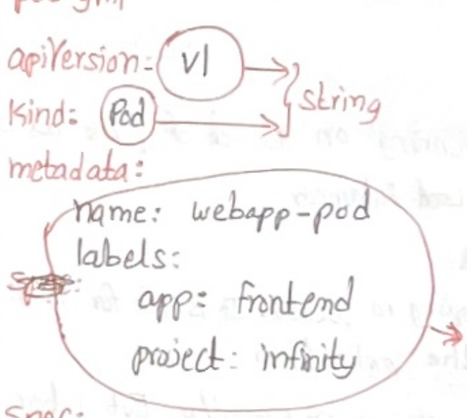
⇒ Tightly coupled and need to share resources

⇒ One main container and others as a sidecar or init container

⇒ Each Pod is meant to run a single instance of a given application

⇒ Should use multiple pods to scale horizontally.

pod.yml



Kind	Version
POD	v1
Service	v1
Deployment	apps/v1
Ingress	networking/v1beta1

GET & EDIT POD

K get pod my-pod -o yaml

K get pod my-pod -o yaml > my-pod.yml

K edit pod my-pod

264) Different Levels of Logging

K get pods

Running → Fine

ImagepullBackoff → Image doesn't exist (or) Credentials issue

CrashloopBackoff → Pulling successfully, restart failed

↓

To get detailed description

K get pod mypod -o yaml

K logs podname

Delete Pod and create again

265) Service

Way to expose an application running on a set of pods as a network service. It is similar to Load Balancer.

There are three types of Services.

1) NodePort → Similar to port mapping in Docker. It is not for Prod. It is only for exposing the application

2) Cluster IP

3) ↳ If we don't want to expose externally. But internally It is for internal communication.

3) LoadBalancer: It is used to expose applications externally for production environments

Please see Mumshad Mannambeth Diagram.

service-def.yml

```
apiversion: V1
Kind: Service
metadata:
    name: webapp-service
spec:
    type: NodePort
    ports:
    - targetPort: 80
      port: 80
      nodePort: 30005
      protocol: TCP
    selector:
      app: frontend
```

K create -f service-def.yml

K get svc

K describe service Servicename

K get svc

Ports

8090:30001/TCP

↓ Internal Port

↓ External Port

To access application

⇒ Public IP of master → 3.139.103.219:30001

LOAD BALANCER:-

```
# service-lb.yml
apiVersion: v1
Kind: Service
metadata:
   name: helloworld-service
spec:
  ports:
  - port: 80
    targetPort: vproapp-port
    protocol: TCP
  selector:
     app: vproapp
  type: LoadBalancer
```

K create -f service-lb.yml

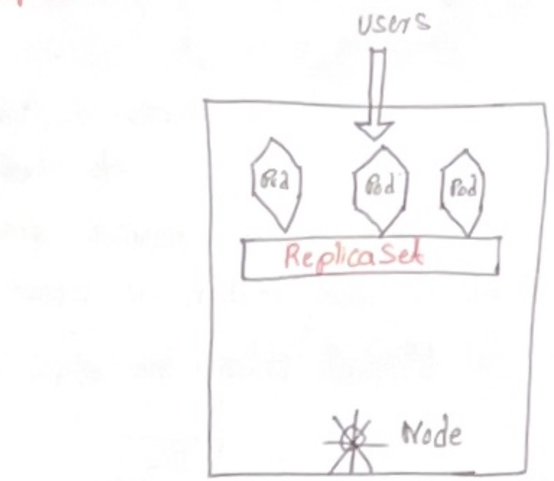
K get svc

We will get LB url in service External IP and the same is created in AWS Console.

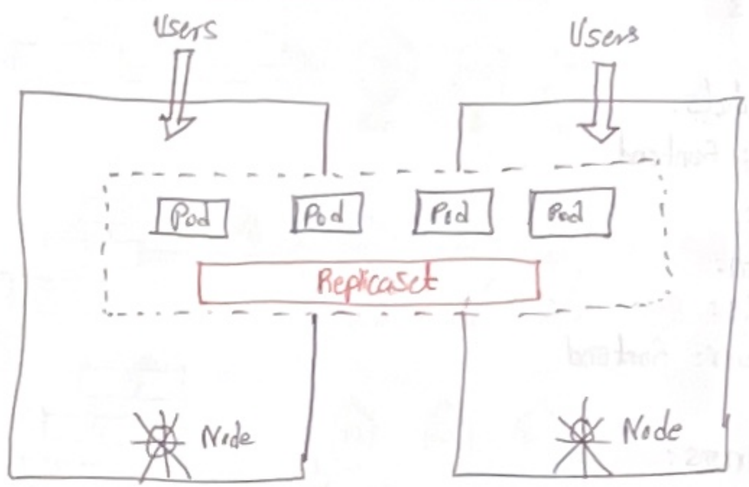
CLUSTER-IP

```
# tom-svc-clusterip.yml
apiVersion: v1
kind: Service
metadata:
   name: app-service
spec:
   type: ClusterIP
   ports:
   - targetPort: 8080
     port: 8080
     protocol: TCP
   selector:
     app: backend
```

266) ReplicaSet



If Pod goes down the Replica will spinup the Pods



Replica Set:-

A Replicaset's purpose is to maintain a stable set of replica Pods running at any given time. As such, it is often used to guarantee the availability of a specified number of identical pods.

replicaset-yml

```
apiVersion: apps/v1
Kind: ReplicaSet
metadata:
   name: frontend
   labels:
     app: guestbook
     tier: frontend
Spec:
   replicas: 3
   selector:
     matchLabels:
        tier: frontend
   template:
     metadata:
       labels:
          tier: frontend
     spec:
       containers:
       - name: php-redis
         image: grc.io/google_samples/gb-frontend:v3
```

k create -f replicaset.yml

k get rs

k get pods

(v2)

k scale --replicas=1 rs/frontend } This is not recomended in Production

(v2)

k edit rs frontend

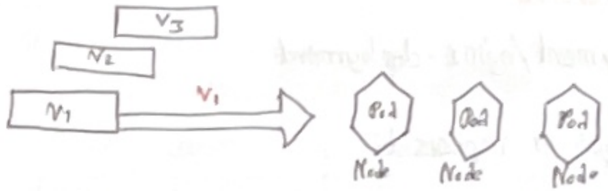
267) Deployment

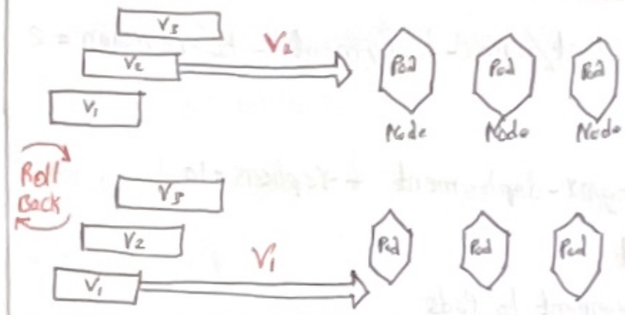
Update, Rollback, changes Gracefully

⇒ A Deployment controller provides declarative updates for Pods and ReplicaSets.

⇒ Define desired state in a Deployment and the Deployment controller changes the actual state to the desired state at a controlled rate

⇒ Deployment creates ReplicaSet to manage number of PODS






```
#deploy.yml
apiVersion: apps/v1
Kind: Deployment
metadata:
   name: app-controller
spec:
   template:
      metadata:
         labels:
            app: backend
      spec:
         containers:
         - name: tomcat-container
           image: tmcat
           ports:
           - name: app-port
             containerPort: 8080
   replicas: 3
   selector:
      matchLabels:
         app: backend
```

K set Image deployment deployname nginx=nginx:1.16.1
Container Image

K get deploy

K get pods

Checking Rollout History of a Deployment

K rollout history deployment/nginx-deployment

Rolling back to previous version

K rollout undo deployment/nginx-deployment

There will be changes in replicaset

Rollback to particular revision:-

K rollout undo deployment/nginx-deployment --to-revision=2

Scaling Deployment:-

K scale deployment/nginx-deployment --replicas=10

*** 268) Commands and Arguement

*** Pass Command and Arguement to Pods

From Ubuntu → docker run printer
CMD ["echo hi"] It will print echo hi

FROM Ubuntu → docker run printer hi
ENTRYPOINT ["echo"]
It will print echo hi

*** FROM Ubuntu → docker run printer → First prints echo hi
*** ENTTYPEINT [echo]
CMD ["hi"] → docker run printer hello → hello overides CMD and prints echo hello

Ex:- This Pod will prints Hostname of server and KBS Port.

pod.yml

```
apiVersion: v1
Kind: Pod
metadata:
   name: command-demo
   labels:
      purpose: demonstrate-command
spec:
   containers:
   - name: command-demo-container
     image: debian
     command: ["printenv"]
     args: ["HOSTNAME", "KUERNETES_PORT"]
   restartPolicy: OnFailure
```

K Create -f pod.yml

K get pods

K logs command-demo

command-demo
tcp://100.64.0.1:443
Hostname Port

269) Volumes:-

On disk- files in a container are ephemeral, which presents some problems for non-trival applications when running in containers One problem is the loss of files when a container crashes. The kubelet restarts the container but with a clean state. A second problem occurs when sharing files between containers running together in a Pod. The K8s volume abstraction solves both of these problems. Familarity with pod is suggested

```
# Volume-pod.yml
apiVersion: v1
kind: Pod
metadata:
   name: dbpod
spec:
   containers:
   - image: mysql:5.7
     name: mysql
     volumeMounts:
     - mountPath: /var/lib/mysql
       name: dbvol
   volumes:
   - name: dbvol
     hostPath:
       # directory location on host
       path: /data
       # this field is optional
       type: DirectoryOrCreate (If director doesn't exist it will create)
```

```
# K apply -f Volume-pod.yml
# K describe pod dbpod
```

We have some error → Modified now

Delete Pod and create. Volume will mount

270) Config Map

Environment Variables

Assign Variable values into a Pod

```
# pod.yml
apiVersion: v1
Kind: Pod
metadata:
   name: db-pod
   namespace: dev
   labels:
      app: db
      project: Infinity
spec:
   containers:
   - name: mysql-container
     image: mysql:5-7
     env:
       - name: MYSQL_DATABASE
         value: accounts
       - name: MYSQL_ROOT_PASSWORD
         value: simepasswd
```

Config Maps:-

Set & Inject variables / files in POD

Create Config Maps | Imperative

Create Config Maps | Imperative

Name of config

```
#K create configmap db-config --from-literal=MYSQL_DATABASE=accounts \
--from-literal=MYSQL_ROOT_PASSWORD=somecomplexpass ~~configmap/db-co~~

#K get cm
db-config

#K get cm db-config -o yaml
apiVersion: v1
data:
  MYSQL_DATABASE: accounts
  MYSQL_ROOT_PASSWORD: passwd
kind: ConfigMap

#K describe cm db-config
```

Create Config Maps | Declarative

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: db-config
data:
  MYSQL_ROOT_PASSWORD: somepass
  MYSQL_DATABASE: accounts

#K create -f db-cm.yml
```

POD Reading Config Maps :-

How to inject our config maps to the PODs

```
apiVersion:
kind:
metadata:
spec:
  containers:
    - name:
      image:
      envFrom:
        - configMapRef:
            name: db-config
      ports:
```

Example :- (See later)

```
spec:
  containers:
    - name:
      image:
      env:
        - name: DB-HOST
          valueFrom:
            configMapKeyRef:
              name: db-config
              key: DB_HOST
```

27) Secrets

Store encoded / encrypted variables to POD

Store and manage sensitive information, such as passwords

Create Secrets | Imperative

K create secret generic db-secret --from-literal=MYSQL-ROOT_PASSWORD
=passwd

secret/db-secret created

#Create files needed for rest of example

echo -n 'admin' > ./username.txt

echo -n '1f2d1e2e67df' > ./password.txt

K create secret generic db-user-pass --from-file=./username.txt
--from-file=./password.txt

K get secret db-secret -o yaml

apiVersion: v1

data:

  MYSQL_ROOT_PASSWORD: abc23l9h2r7489ch== → Encoded

kind: Secret

metadata:

How to encode data:-

echo -n "Secretpass" | base64

Decode:-

echo 'abcd1245891208' | base64 --decode

secretpass

It will be stored in k8s control plane. So it is safe

Create Secrets | Declarative:-

```
# echo -n "somepassword" | base64
c2afdzaklmYop==
```

```
# db-secret.yml
apiVersion: v1
Kind: secret
metadata:
  name: mysecret
type: Opaque
data:
  my_root_pass: c2afdzaklmYop==
```

```
# K create -f db-secret.yml
```

POD Reading Secret:-

```
apiVersion:
Kind:
metadata:

spec:
  containers:
  - name:
    Image:
    envFrom:
      - secretRef:
          name: mysecret
```

```
spec:
  containers:
  - name:
    image:
    env:
      - name: MYSQL_ROOT_PASSWORD
        valueFrom:
          secretKeyRef:
            name: mysecret
            key: my_root_pass
```

* *

To pull Image from your private registry

```
# echo -n "admin" | base64
YuRtaW4
# echo -n "mysecretpass" | base64
bXlzmnpad14789Nz
```

```
# secret.yml
apiVersion: v1
Kind: Secret
metadata:
   name: mysecret
data:
   username: YuRtaW4
   password: bXlzmnpad14789Nz
type: Opaque
```

```
# K create -f secret.yml
```

Now Inject into Pod, Pod.yml

```
apiVersion:
Kind: Pod
metadata
spec:
  Containers:
  - name:
    image:
    env:
      - name: SECRET_USERNAME
        valueFrom:
          secretKeyRef:
            name: mysecret
            Key: username
```

Continuation

```
- name: SECRET_PASSWORD
  valueFrom:
    secretKeyRef:
      name: mysecret
      key: password
      optional: false
restartPolicy: Never
```

```
# K create -f pod.yml
# K get pods
# K exec --stdin --tty secret-env-pod -- /bin/bash
                # echo $username
                # echo $SECRET_USERNAME
admin
                # echo $SECRET_PASSWORD
mysecretpass
```

272) Ingress

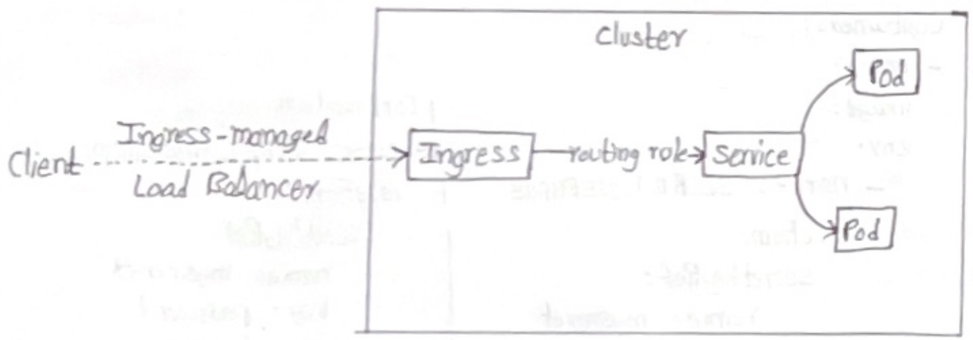
An API Object that manages external access to the service in a cluster, typically HTTP.

Ingress may provide load balancing, SSL termination and name-based virtual hosting.

What is Ingress?

Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster. Traffic routing is controlled by rules defined on the Ingress resource.

Here is a simple example where an Ingress sends all its traffic to one service.



Prerequisites:-

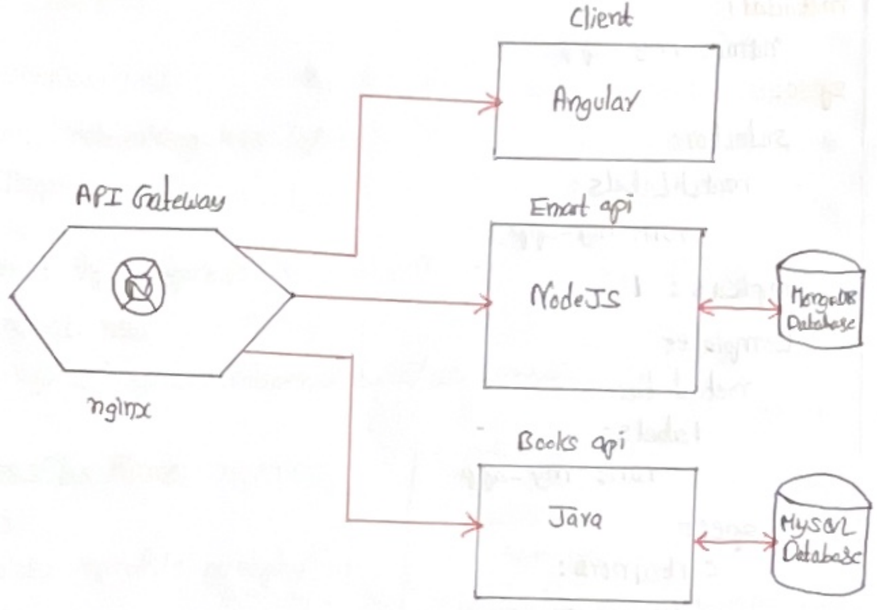
You must have an ingress controller to satisfy an Ingress. Only creating an Ingress resource has no effect.

You may need to deploy an Ingress controller such as Ingress-nginx. You can choose from a number of Ingress controllers.

Ideally, all Ingress controllers should fit the reference specification. In reality, the various Ingress controllers operate slightly differently.

We have different Ingress controllers

We will see NGINX Ingress Controller for Kubernetes



Install Ingress Controller:-

Goto Google Install nginx Ingress controller

https://kubernetes-github.io/ingress-nginx/deploy/#aws

K apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.1.3/deploy/static/provider/aws/deploy.yml

K get all -n ingress-nginx

Now we can see Load Balancer and user can access LB.

vprodep.yml

```
apiVersion: apps/v1
Kind: Deployment
metadata:
   name: my-app
spec:
   selector:
     matchLabels:
        run: my-app
   replicas: 1
   template:
     metadata:
        labels:
          run: my-app
     spec:
        containers:
        - name: my-app
          image: tomcat (or) our own image
          ports:
          - containerPort: 8080
```

vprosvc.yml

```
apiVersion: v1
Kind: Service
metadata:
   name: my-app
spec:
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 8080
  selector:
     run: my-app
  type: ClusterIP
```

vproingress.yml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
   name: vpro-ingress
   annotations:
      nginx.ingress.kubernetes.io/use-regex: "true"
Spec:
   ingressClassName: nginx
   rules:
   - host: vprofile.groophy.in
     http:
       paths:
       - path: /login
         pathType: Prefix
         backend:
           service:
             name: my-app
             port:
               number: 8080
       pathType: Prefix
       backend:
```

K create -f vprodep.yml
K create -f vprosvc.yml
Now goto LB in Aws console and copy DNS and add DNS records
With CNAME = Value and add record.
K get svc
K get describe svc my-app
K apply -f vproingress.yml

Just Overview

Controller
Deployment
Service
Created DNS Cname Record for LB
Created Ingress
K get Ingress
Copy the URL and check in Browser (or) we can check in our Domain

Path Based Rating also available in docs please check video

273) Kubectl CLI & Cheatsheet (Tricks)
Goto K8s cheatsheet

274) Extras
** Learn Taints & Tolerations from Hemshad
** Limits
Jobs [Kind: Job]
Cronjob
Daemonset

275) Lens
Lens can have contra view of K8s cluster
Goto https://k8slens.dev
Install Lens