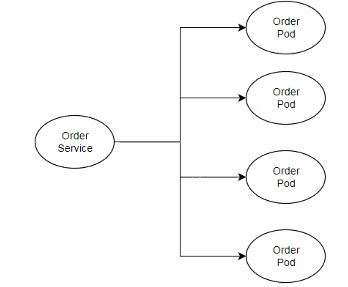
12-12-19:

Kubernetes Service:-

* We use service to expose our POD’s to the internet.
* It also servs as a load balancer



* Service also has IP
* POD’s are designed to be terminated
* Directly communicating with POD using IP is not recommended ,bcoz chances are high that POD is re-created with diff IP
* Always place a service object infront of Pod

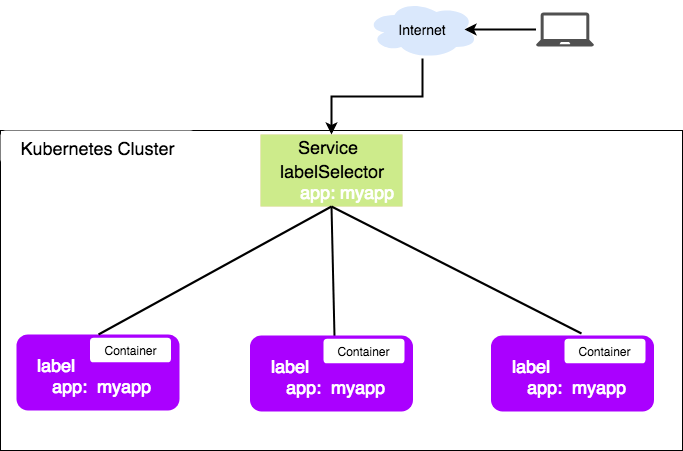


How service and it’s related POD’s are tied up:

* Kubernetes uses lables in order to map service objects with related POD’s

**Kubernetes Service Objects**

* In kubernetes service is an object that is logically mapped to pods based on labels.
* Service can be single point of contact like loadbalancer for set of pods.
* By default Pod is not expose to the internet or outside of the cluster, by using service Pod is exposed to internet.
* Any Pod create in future is added to service dynamicaly if pod label is matching with Service lable selector.

[](https://github.com/javahometech/kubernetes/blob/master/images/service.png)

# kubectl delete pod/nodeapp

# kubectl apply -f services.yml

#

Service Types:

* Node port:
* Cluster IP:it will configure our service as a Private

Ex: a service for database must be private

* Load balancer: it creates a cloud native load balancer this option works only when Kubernetes cluster is on cloud.

**Externalname:**

It maps the service to the contents of the externalname field (e.g: foo.bar)

**Replication control:**

Its job to maintain specified no.of replicas in the cluster

If no.of POD is less than desired it creates, IF POD’s are more than desired it deletes.

Refer to doc(<https://github.com/javahometech/kubernetes/tree/master/ReplicationController>)

Refer fig:replication controller

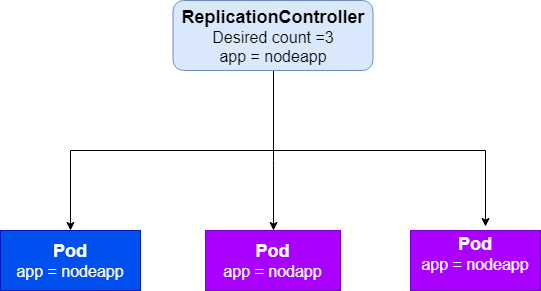
## ReplicationController

* A ReplicationController ensures that a desired number of pod replicas are running at any one time.
* A ReplicationController makes sure that a pod or a homogeneous set of pods is always up and available

### Working style of ReplicationController

* ReplicationController creates if less number of pods are available or terminates extra pods if there are too many pods
* It replaces if any one of pod is deleted or failed.
* ReplicationController supervises multiple pods across multiple nodes
* ReplicationController is often abbreviated to **"rc"** or **"rcs"** in discussion, and as a shortcut in kubectl commands.

**EX:-** If pods are re-created on a node after disruptive maintenance such as a kernel upgrade. for this reason, use ReplicationController if if your application require only single-pod

[](https://github.com/javahometech/kubernetes/blob/master/images/ReplicationController.png)

## Creating a ReplicationController

### Create rc.yml with following content

Get the file [(rc.yml)](https://github.com/javahometech/kubernetes/blob/master/ReplicationController/rc.yml)

apiVersion: v1

kind: ReplicationController

metadata:

name: nodeapp

spec:

replicas: 3

selector:

app: nodeapp

template:

metadata:

name: nodeapp

labels:

app: nodeapp

spec:

containers:

- name: nodeapp

image: kammana/nodeapp:v1

ports:

- containerPort: 8080

### Command to create ReplicationController

**Syntax:** kubectl create -f FILE\_NAME

$ kubectl create -f rc.yml

### Command to get all ReplicationController

**Syntax:** kubectl get rc

$ kubectl get rc

### Command to describe ReplicationController

**Syntax:** kubectl describe rc REPLICATIONCONTROLLER\_NAME

$ kubectl describe rc nodeapp

### Command to Expose ReplicationController to Internet

By default ReplicationController run in a isolated environment i.e. they are reachable within kubernetes cluster, if you wanna reach your ReplicationController outside cluster, You have to expose it

**Syntax:** kubectl expose rc REPLICATIONCONTROLLER\_NAME --type="NodePort" --port CONTAINER\_PORT

$ kubectl expose rc nodeapp --type="NodePort" --port 8080

13-11-19

Replication controller :

Replica set is an extension tool app controller,replica set supports set based selector .

IQ) How do you manage secrets in Kubernetes

You have private images , right now images are public

Ans) we have to use secrets object there

Create kubernetes secrets

kubectl create secret docker-registry regcred --docker-server=https://index.docker.io/v1/ --docker-userme=kammana --docker-password=<your-password> [--docker-email=hari.kammana@gmail.com](mailto:--docker-email=hari.kammana@gmail.com))

Deployements:

Deployement object has rolling updates &roll backs.

## Deployments

It is a kubernetes object which the declarative updates for Pods and ReplicaSets

### Create Deployment

kubectl create -f https://raw.githubusercontent.com/javahometech/kubernetes/master/deployments/deployments.yml --record=true

### WE can Check status of the current deployment

kubectl rollout status deployment nodeappdeployment

### Updating deployment (created and upate my deployement

For example we want to change number of replicas, change replicas in yaml and run following command

kubectl apply -f https://raw.githubusercontent.com/javahometech/kubernetes/master/deployments/deployments.yml --record=true

### Kubernetes Deployment revisions

Kubernetes maintains deployment state of all versions

inorder to see deployment revision history

kubectl rollout history deployment nodeappdeployment

### Undo recent deployment

kubectl rollout undo deployment nodeappdeployment

### rollback to specific deployment revision

kubectl rollout undo deployment nodeappdeployment --to-revision=1

Deployement objects by default use roling update strategy

Readiness probes & LIVENESS PROBES

## Liveness and Readiness Probes

The kubelet uses liveness probes to know when to restart a Container. For example, liveness probes could catch a deadlock, where an application is running, but unable to make progress. Restarting a Container in such a state can help to make the application more available despite bugs.

apiVersion: v1

kind: Pod

metadata:

labels:

test: liveness

name: liveness-exec

spec:

containers:

- name: liveness

image: k8s.gcr.io/busybox

args:

- /bin/sh

- -c

- touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 600

livenessProbe:

exec:

command:

- cat

- /tmp/healthy

initialDelaySeconds: 5

periodSeconds: 5

kubectl create -f https://raw.githubusercontent.com/javahometech/kubernetes/master/liveness-readyness/liveness-probe.yml

---------------------------some part of the calss missed -----------------------

14-11-19

<https://github.com/javahometech/helm/blob/master/README.md>

# Helm Kubernetes

Helm is for packaging and deploying kubernetes applications

### Install

curl -LO https://git.io/get\_helm.sh

chmod 700 get\_helm.sh

./get\_helm.sh

### To intial helm local repository

helm init

### To create helm chart

helm create nodeapp-helm

### To create helm package

helm package nodeapp-helm

### To deploy helm package

helm install nodeapp-helm

### To deploy helm package with custom release name

If you do not mention name, helm chooses random names for release, if you want to have custom name for a relase, run the following command

helm install --name my-release nodeapp-helm

## NOTE - If you get error as follows when you run above command

configmaps is forbidden: User "system:serviceaccount:kube-system:default" cannot list resource "configmaps" in API group "" in the namespace "kube-system"

### deploy tiller service account RBAC to avoid above error

kubectl --namespace kube-system create serviceaccount tiller

kubectl create clusterrolebinding tiller-cluster-rule --clusterrole=cluster-admin --serviceaccount=kube-system:tiller

kubectl --namespace kube-system patch deploy tiller-deploy -p '{"spec":{"template":{"spec":{"serviceAccount":"tiller"}}}}'

### Upgrade helm release

helm upgrade --set-string image.tag=v2

helm upgrade --set-string image.tag=v2 nodeapp nodeapp

### To list out helm releases which are in deployed state

helm ls

### To list out helm releases which are deployed and deleted state

helm ls --all

### To delete helm release

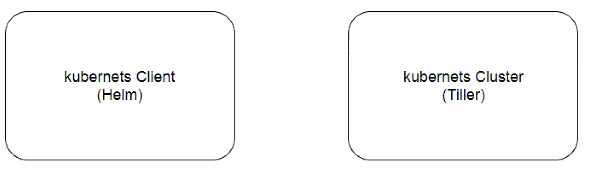
To delete release completely run following two commands

helm delete helm-release-name

helm del --purge helm-release-name

**Helm Architecture:**

Helm is an utility runs at clients side

Tiller is an POD that runs in Kubernetes cluster

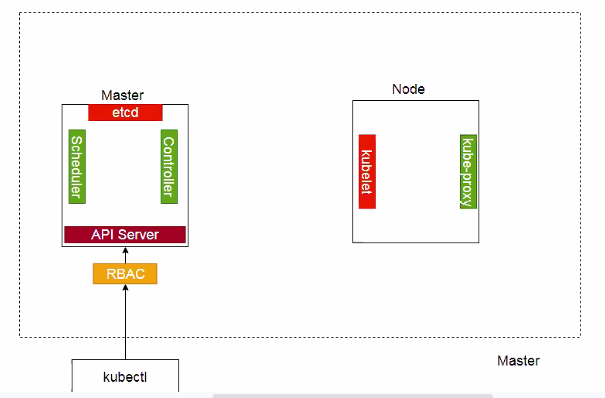
**Kubernetes Architecture:**

**(All comm** happens through Api Server

Many controllers will be available

Master will assign to node , to do a task

Yaml doc is sent to Api server -thn scheduler -then kubetel -kubtel creates pod ,if pod crash -kubetel communicates with master)

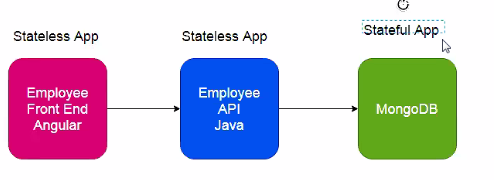


IQ) How do you secure your Kubernetes cluster?

ANS) RBAC

**Deploying 3-tier applications in to Kubernetes:**

<https://github.com/javahometech/kubernetes/tree/master/3-tier-app>



We can consider above fig as 3 micro services

Employee front end angular speaks to employee api java and this talks to Mango DB

State full maintains data

State less wont maintain data

1st deploy mongo db , then front end then API

15-11-14

IQ) In Kubernetes how can we schedule a POD only on specific nodes.

* **Node selector :** We can assign POD’s to specific nodes with help of Lables

nodeSelector:

disktype: ssd

the above POD is scheduled only on the nodes which has a label key “described” and label values “ssd”

* **Node name:**
* **Node affinity:**  you can indicate that the rule is “soft “/ “preference “ rather than a hard requirement , so if the scheduler can’t atisfy it,the prod will still be scheduled.

you can constrain against lables on other pods running on the node (or other topological domain), rather than against lables on the node itself, which allows rules about which pods can and can not be co-located.

**(** you check anti affinity and taints and tolerations)