How to make the best use of Live Sessions

- Please login on time
- Please do a check on your network connection and audio before the class to have a smooth session
- All participants will be on mute, by default. You will be unmuted when requested or as needed
- Please use the “Questions” panel on your webinar tool to interact with the instructor at any point during the class
- Ask and answer questions to make your learning interactive
- Please have the support phone number (US : 1855 818 0063 (toll free), India : +91 90191 17772) and raise tickets from LMS in case of any issues with the tool
- Most often logging off or rejoining will help solve the tool related issues

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

COURSE OUTLINE

Module 05

Introduction to Kubernetes

Kubernetes Architecture

Deploy app to Kubernetes Cluster

Expose App, Scale App And Update App

Managing State with Deployments

Federations, Auditing and Debugging Kubernetes, Security best practices

edureka!

Managing State with Deployments

Objectives

After completing this module, you should be able to understand:

- Taints and Tolerations
- Daemon Sets
- Persistent Volumes
- Config Maps
- Headless Services
- Stateful Sets

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Taints and Tolerations

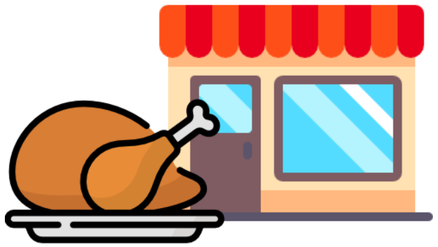
edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Taints and Tolerations

Taints allow nodes to repel a set of pods.

Toleration defines whether a pod can exist on the node or not



Non Veg Restaurant

A vegan cannot tolerate a Non-Veg Restaurant

Veg Restaurant

A vegan can only tolerate a Veg Restaurant

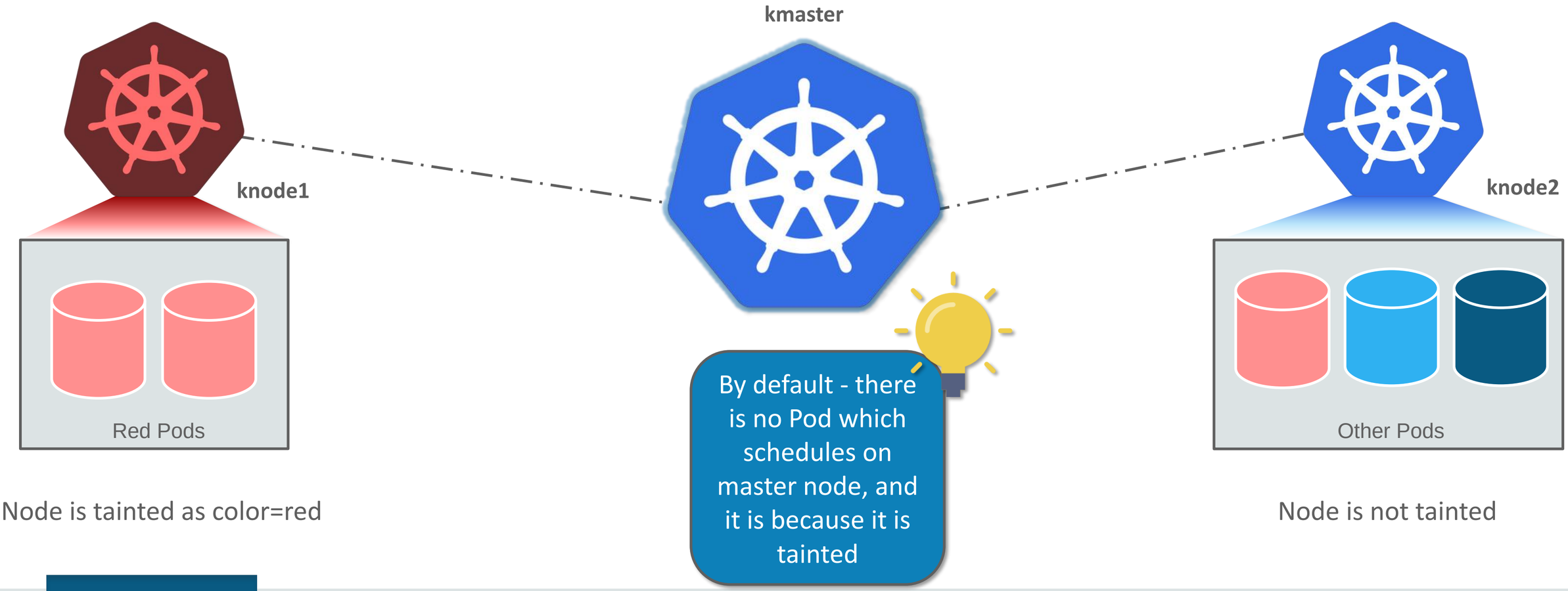
edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Taints and Tolerations

With the same approach, consider that nodes which are tainted, then **not all pods would be able to run on it**.

Only the pods which are ‘tolerant’, would be able to run on respective particular tainted nodes.



Node is tainted as color=red

Node is not tainted

edureka!

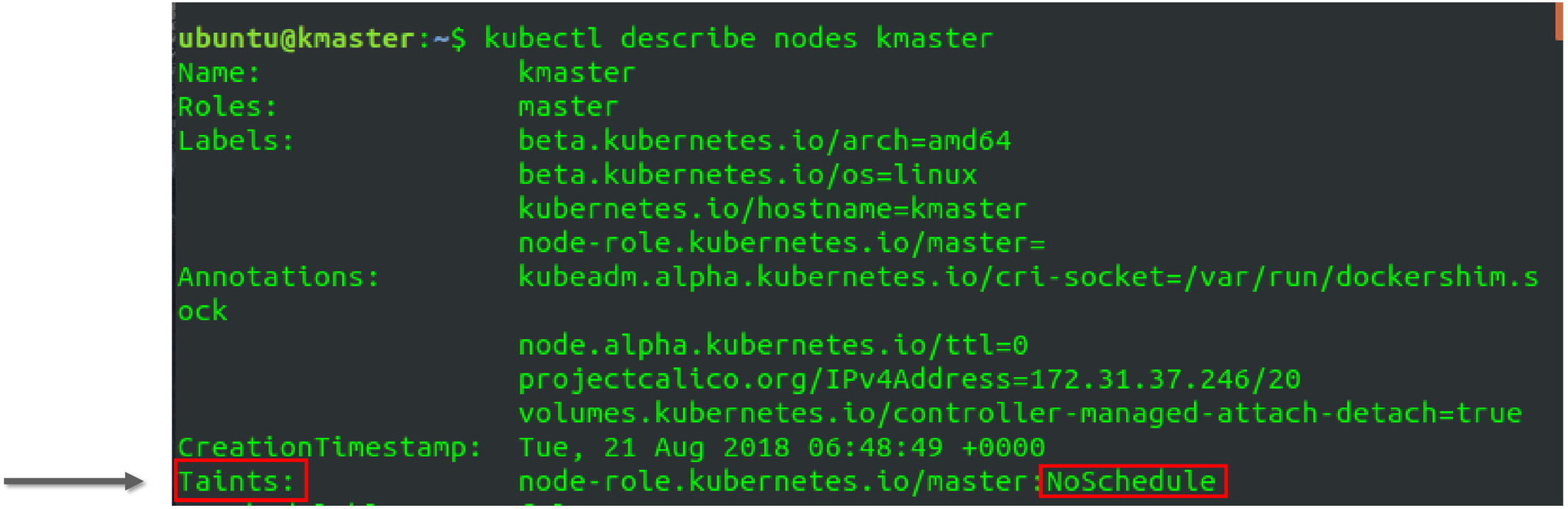
Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Taints and Tolerations

- To check the taints on a node, type in the following command:

```
kubectl describe nodes <node-name>
```

- Notice the line with 'Taints', says 'Noschedule', which ensures that no Pods schedule on the master



edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Adding Taint to a Node

- To add taint on a node, type in the following command:

```
kubectl taint nodes <node-name> <label-key>=<label-value>:Effect
```

- The effect can be the following:
  - **NoSchedule** – This does not schedule any future execution of pods which do not match the taint on the node
  - **PreferNoSchedule** – This is a “soft” rule, which may or may not schedule the pod on specified node
  - **NoExecute** – Evicts any pod running on the node with no matching toleration. Also, repels in-tolerant pods for future

```
ubuntu@kmaster:~$ kubectl taint nodes knode color=red:NoSchedule
node/knode tainted
ubuntu@kmaster:~$
```


Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Adding Toleration to a Pod

- To define toleration in pod, type in the following command:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  tolerations:
  - key: "color"
    operator: "Equal"
    value: “red"
  containers:
  - name: nginx
    image: nginx:1.7.9
    ports:
      - containerPort: 80
```

```
ubuntu@kmaster:~$ kubectl get po -o wide -l app=nginx
NAME      READY     STATUS    RESTARTS   AGE       IP              NODE
nginx     1/1       Running   0          7m        192.168.1.34    knode
```

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Demo: Taints and Tolerations

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Demo: Taints and Tolerations

- Remove the taint from the **kmaster**
- Add a new taint to the master, which says *colour=red*
- Add *colour=blue* taint to the **worker node**
- Deploy two pods, with **tolerations of colour red and blue respectively**

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

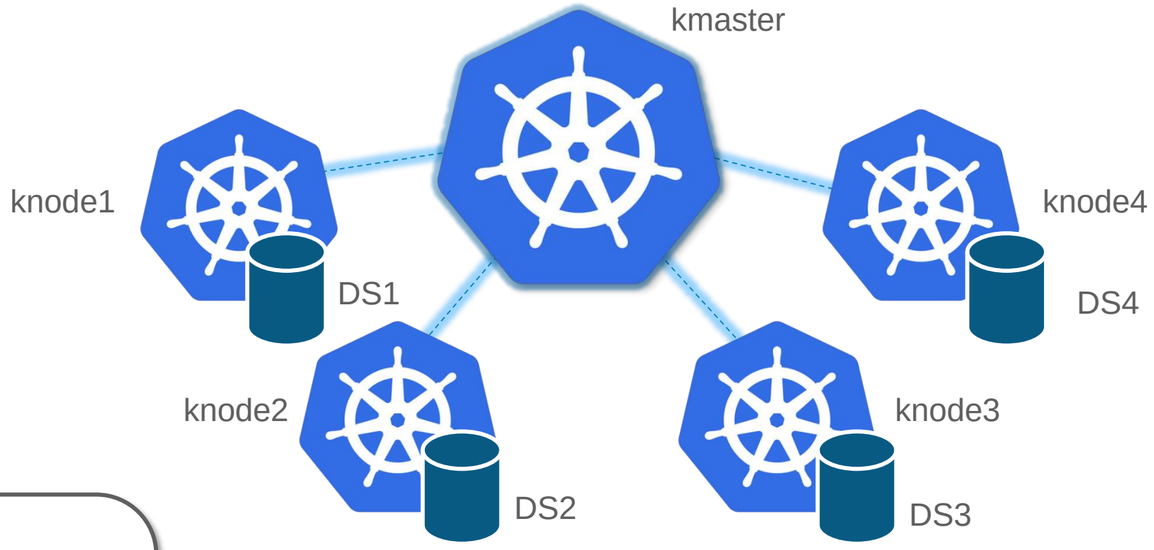
Daemon Sets

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Daemon Sets

Daemon Sets are pods or applications that we want to run on every or selected node of the cluster



Use-cases :

- Collection of logs
- Monitoring of resources

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Syntax for Daemon Set vs Deployment

```
apiVersion:  extensions/v1beta1
kind: DaemonSet
metadata:
  name: nginx
spec:
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
          ports:
            - containerPort: 80
```

Daemon Set Syntax

```
apiVersion:  extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
          ports:
            - containerPort: 80
```

Deployment or Replica Set Syntax

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Demo: Daemon Set

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Demo: Daemon Sets

- Create a Daemon Set for nginx application

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

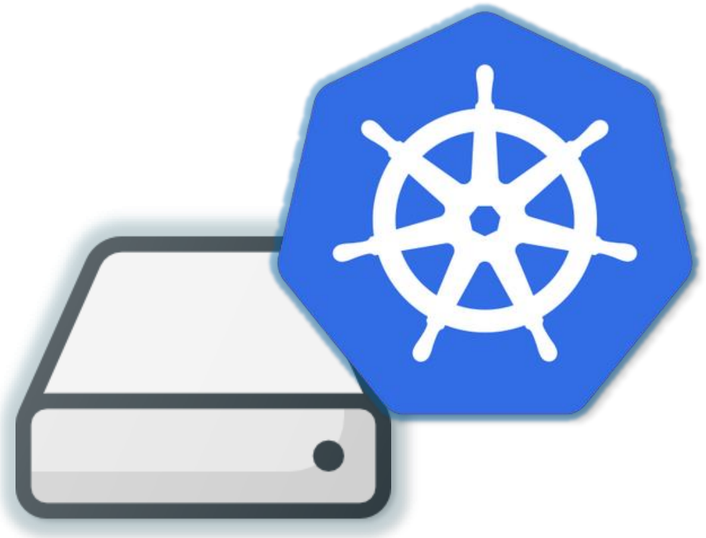
Persistent Volumes

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Persistent Volumes

- Data stored inside a container persists only till the time the container is running
- If the container crashes for any reason, the new container which will be created will not persist the data from the previous container
- To solve this, we have Persistent Volumes
- They act as an external volume to a container or pod. A Persistent Volume's lifecycle is separate from the container's lifecycle



Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Deploying a Persistent Volume in Kubernetes

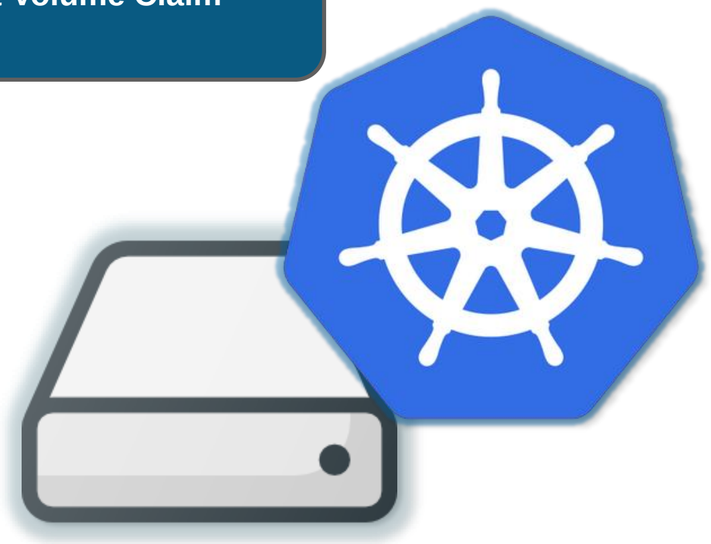
- The process of deploying a Persistent Volume is a two stage process.

Step 1: Create a Persistent Volume

Defines a resource in the the cluster

Creates a claim for that resource

Step 2: Create a Persistent Volume Claim



Let us discuss each of these in detail

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Persistent Volumes

Persistent Volumes

Persistent Volume Claim

It is the storage which is provisioned by the admin.

API captures the details of the storage which is implemented be it be ISCSI, NFS.

It is just like another resource in the cluster.

PersistentVolume is independent of any individual Pod that uses PV.

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Persistent Volume Claim (PVC)

Persistent Volumes

Persistent Volume Claim

It is the claim raised by developer for storage volume

Claims can be of specific size or/and access (read-only, read/write)

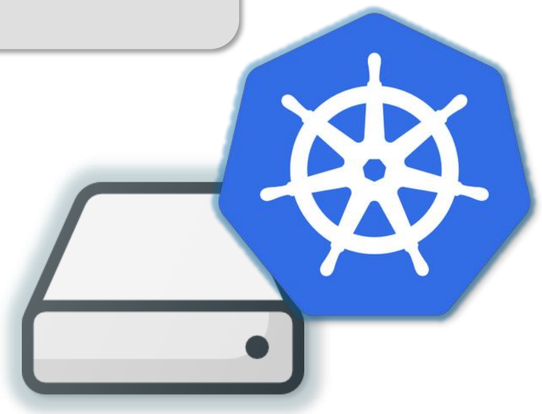
edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Access Modes for Persistent Volumes

A persistent volume can be accessed by in different ways, these are called **Access Modes.** One can choose an Access Mode based on needs, security etc.

ReadWriteOnce	ReadOnlyMany	ReadWriteMany
The volume can be mounted as read write by a single node	the volume can be mounted read-only by many nodes	the volume can be mounted as read-write by many nodes



Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Life Cycle of Persistent Volume - Binding

With respect to the lifecycle of Persistent Volume there are some more concepts which you should know.

Lets get some details about it.

- **Binding** :
  - When a PersistentVolumeClaim is been raised for required access permission and storage capacity, there a control loop runs on the master to find the matching PersistentVolume. And when PV is matched to the required capacity and access permission, it gets bind to it.
  - Please note that matched PV should meet the requirement or provide more than that but in any condition it won’t provide or compromise on the requirement.

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Life Cycle of Persistent Volume

- Ex. if there is a PVC for 50GB of storage capacity then controller will make sure to provide minimum of 50GB capacity or more than that.
- If required capacity is not available then claim will remain unbounded.
- Please note, PVC to PV binding is a one-to-one mapping.

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Life Cycle of Persistent Volume - Reclaim

- **Reclaim:**
  - When admin/user completes the desired activity then can opt to release the PVC.
  - They triggers the process of reclamation of the resource.
  - Policy defined for PersistentVolume, decides what needs to be done after the PV is been released.
  - There are three options, which can be chosen:
    - Retained
    - Recycled
    - Deleted

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Pod Scheduling and Resource Limitation

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Pod Scheduling and Resource Limitation

01 Scheduling Pod manually is sometime necessary in production environment.

02 Most primary use-case is of scheduling pods on selected hardware / nodes.

03 Kubernetes comes with a default scheduler which is aware of topology, resource availability etc.

04 You can even customize or write your own scheduler depending upon the requirement, however the default scheduler holds good for most of the use-cases.

05 If scheduler doesn't find the sufficient resource for scheduling the pod then it will put pod in 'pending state', untill a node become sufficient resource available.

06 CPU and Memory are the two main resources. CPU is measured by 'cores' and Memory is measured by 'bytes'

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Pod Scheduling and Resource limitation (Cont..)

- To have better utilization of resources, it becomes important to define the resource requirement.
- Kubernetes specifies pre-defined spec for it.
    - spec.containers[].resources.limits.cpu
    - spec.containers[].resources.limits.memory
    - spec.containers[].resources.requests.cpu
    - spec.containers[].resources.requests.memory
- Once, it is specified - scheduler check for the resource availability and ensure that Pods does not get terminated.

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Application Environment Configuration : ConfigMap

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

ConfigMap

- We wlll see how kubernetes setup applications through environment variables, but we don’t setup environment variable for individual pods in every single YAML file which we create or even using command line.
- We do this using **ConfigMap**.
    - This helps in maintaining the containerized application portability.

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

ConfigMap

- You can create ConfigMap by using simple command like :

```
 $ kubectl create configmap my-config --from-
literal=key1=config1 --from-literal=key2=config2
```

- Create a new configmap named my-config with key1=config1 and key2=config2

```
learnkarts@master:~$
learnkarts@master:~$ kubectl create configmap example-map --from-literal=school=edureka
configmap/example-map created
learnkarts@master:~$
```


Copyright © 2018, edureka and/or its affiliates. All rights reserved.

ConfigMap

- As an example, we will create a sample config map and ensure that it set up the environment variable for the application.
- In this file, we are setting up the environment variable by using the configMap.

```
learnkarts@master:~$ kubectl describe configmap example-map
Name:         example-map
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
school:
----
edureka
Events:  <none>
learnkarts@master:~$
```

```
$cat example-map.yaml
apiVersion: v1
kind: Pod
metadata:
  name: env-output
spec:
  containers:
  - name: nginx
    image: nginx
    command:
      - env
    env:
    - name: SCHOOL
      value: "Edureka"
    - name: Edureka
      value: "is awesome"
```

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

ConfigMap

- Create the pod with the sample file and log the output to see the environment variables.

```
$kubectl create -f <file-name>.yaml
$kubectl get pods
$kubectl logs <pod_name>
```

- Using this approach, containers will remain portable and can be moved easily from one environment to another.

```
learnkarts@master:~$ kubectl get pods
NAME          READY     STATUS             RESTARTS   AGE
env-output    0/1       CrashLoopBackOff   3          1m
learnkarts@master:~$ kubectl logs env-output
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=env-output
SCHOOL=Edureka
Edureka=is awesome
KUBERNETES_SERVICE_HOST=10.96.0.1
```


Copyright © 2018, edureka and/or its affiliates. All rights reserved.

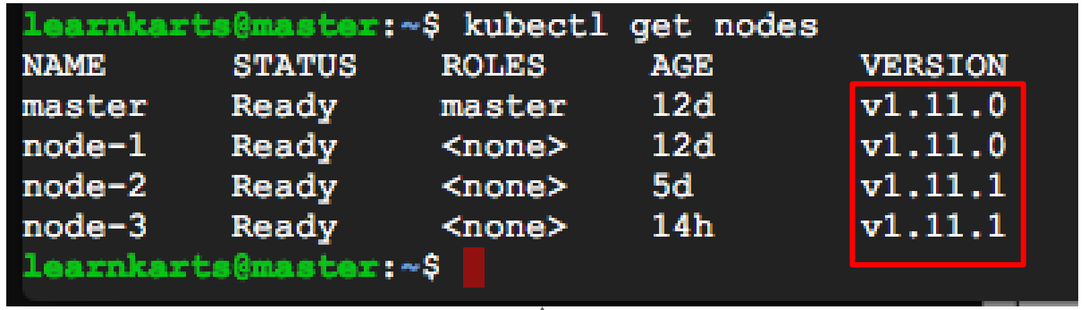
Maintenance of a Node in a Kubernetes Cluster

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Maintenance of a Node in a Kubernetes Cluster

- There is no system or infrastructure in the IT world which can survive without maintenance.
- In this section, we will see how to upgrade kubernetes environment without taking down the cluster.
- In our environment, we see that master node and node-1 is running on v1.11.0 and rest all the nodes are running on v1.11.1.
- Now we will see how we can upgrade the remaining nodes to the latest release.



Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Maintenance of a Node in a Kubernetes Cluster

- **Step 1**: Upgrade the control plane
  - This we will do by upgrading the latest version of kubeadm on master node.
  - Below is the current version of kubeadm.

```
learnkarts@master:~$ kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"11", GitVersion:"v1.11.0", GitCommit:"91e7b4fd31fcd3d5f436da26c980
becec37ceefe", GitTreeState:"clean", BuildDate:"2018-06-27T20:14:41Z", GoVersion:"go1.10.2", Compiler:"gc", Platfor
m:"linux/amd64"}
learnkarts@master:~$
```

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Maintenance of a Node in a Kubernetes Cluster

- Please note that - version will depend upon the repository which you are using.
- Use the command

```
$ sudo apt  upgrade kubeadm
```

- This will upgrade the kubeadm to the version available in your repository. Here, I am fetching from upstream repo.

```
learnkarts@master:~$ kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"11", GitVersion:"v1.11.1", GitCommit:"b1b29978270dc22fecc592ac55d9
03350454310a", GitTreeState:"clean", BuildDate:"2018-07-17T18:50:16Z", GoVersion:"go1.10.3", Compiler:"gc", Platfor
m:"linux/amd64"}
learnkarts@master:~$
```

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Maintenance of a Node in a Kubernetes Cluster

Now, this will help us in upgrading rest of the control plane.

Step 2: Prepare the upgrade plan

```
$sudo kubeadm upgrade plan
```

```
learnkarts@master:~$ sudo kubeadm upgrade plan
[preflight] Running pre-flight checks.
[upgrade] Making sure the cluster is healthy:
[upgrade/config] Making sure the configuration is correct:
[upgrade/config] Reading configuration from the cluster...
[upgrade/config] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -oyaml'
[upgrade] Fetching available versions to upgrade to
[upgrade/versions] Cluster version: v1.11.0
[upgrade/versions] kubeadm version: v1.11.1
[upgrade/versions] Latest stable version: v1.11.1
[upgrade/versions] Latest version in the v1.11 series: v1.11.1

Components that must be upgraded manually after you have upgraded the control plane with 'kubeadm upgrade apply':
COMPONENT   CURRENT       AVAILABLE
Kubelet     1 x v1.11.0   v1.11.1
            3 x v1.11.1   v1.11.1

Upgrade to the latest version in the v1.11 series:

COMPONENT             CURRENT   AVAILABLE
API Server            v1.11.0   v1.11.1
Controller Manager    v1.11.0   v1.11.1
Scheduler             v1.11.0   v1.11.1
Kube Proxy            v1.11.0   v1.11.1
CoreDNS               1.1.3     1.1.3
Etcd                  3.2.18    3.2.18

You can now apply the upgrade by executing the following command:

        kubeadm upgrade apply v1.11.1

_____________________________________________________________________

learnkarts@master:~$
```

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

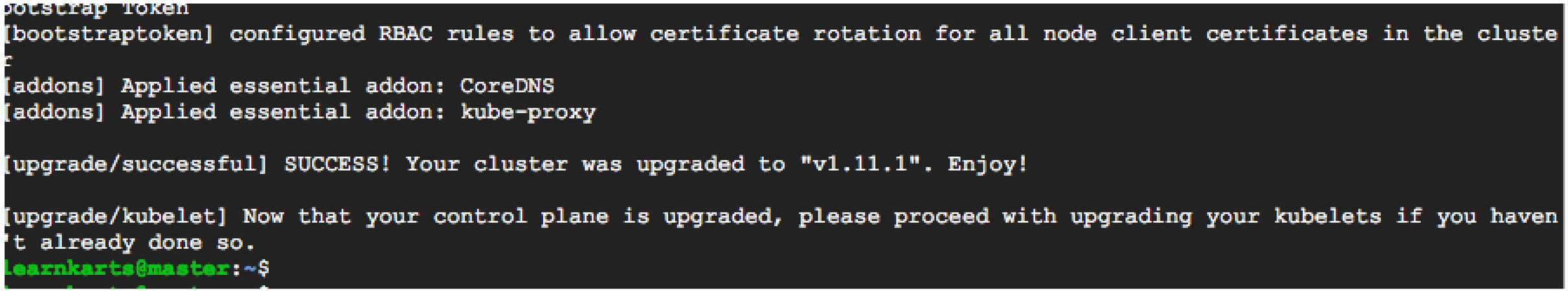
Maintenance of a node in a kubernetes cluster

- Upgrade the control plane using the command as mentioned on the screen

```
$ sudo kubeadm upgrade apply v1.11.1
```

- This command will upgrade all the necessary components of kubernetes control plane as per the plan prepared in previous step.

- Once done, output will show something like below screen capture:



edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Maintenance of a node in a kubernetes cluster

- **Step 3**: Upgrade the CNI.
- Presently we are running with latest version of calico. So, I am skipping this step, but if it is there then you need to follow the official procedure of calico upgrade. You need to use the command :

```
$kubectl apply -f <http_path_to_calico_upgrade>.yaml
```


Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Maintenance of a node in a kubernetes cluster

- **Step 4**: Upgrade the individual nodes (minion / worker node) to the latest release.
- Do it one at a time.
- To make it more real time, let's have a deployment running on it with 3 replicas.

```
learnkarts@master:~$ kubectl get pods -o wide
NAME                            READY     STATUS    RESTARTS   AGE       IP                NODE
redis-master-76bc966444-2kg8n   1/1       Running   0          35s       192.168.247.15    node-2
redis-master-76bc966444-p6lt7   1/1       Running   0          35s       192.168.139.69    node-3
redis-master-76bc966444-q84tz   1/1       Running   0          35s       192.168.84.150    node-1
learnkarts@master:~$ kubectl get deployment
NAME           DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
redis-master   3         3         3            3           49s
learnkarts@master:~$
```

We see that there are 3 replicas and each one of them are running on node-1, node-2, node-3 respectively.
So, we need to upgrade the kubelet on each of these nodes, but before doing that we need to make sure that none of these pods are running on the node on which we are going to upgrade the kubelet.

So, we need to drain all the running pods (exclude the daemonSets like network container).

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Maintenance of a node in a kubernetes cluster

- **Step 5** : Upgrade the individual nodes (minion / worker node) to the latest release.
- Do it one at a time.
- To make it more real time, let’s have a deployment running on it with 3 replicas.
- But before doing it on worker nodes, upgrade the master first with kubelet.
- You need to drain the pods running on it.

```
learnkarts@master:~$ kubectl drain master --ignore-daemonsets
node/master cordoned
WARNING: Ignoring DaemonSet-managed pods: calico-etcd-785jk, calico-node-tm95p, kube-proxy-zhmwm
pod/coredns-78fcdf6894-5kb4s evicted
pod/calico-kube-controllers-84fd4db7cd-wghv8 evicted
pod/coredns-78fcdf6894-xxh44 evicted
learnkarts@master:~$
learnkarts@master:~$
```

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Maintenance of a node in a kubernetes cluster

- Once it is done. Run the command to upgrade kubelet.

```
$ sudo apt upgrade kubelet
```

- This will restart the kubelet. Make sure service is up and running after the upgrade.

```
$ systemctl status kubelet
```


Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Maintenance of a node in a kubernetes cluster

- Now, we will see the version of our master node

```
learnkarts@master:~$ kubectl get nodes
NAME      STATUS                        ROLES     AGE       VERSION
master    Ready,SchedulingDisabled      master    12d       v1.11.1
node-1    Ready                         <none>    12d       v1.11.0
node-2    Ready                         <none>    5d        v1.11.1
node-3    Ready                         <none>    16h       v1.11.1
learnkarts@master:~$
```

- Here, we see that master is upgraded to now v1.11.1
- Only node-1 is running on the lower version but before doing that, lets mark master back to ‘Ready’ status
-

```
$ kubectl uncordon <node-name>
```

```
learnkarts@master:~$ kubectl get nodes
NAME      STATUS                        ROLES     AGE       VERSION
master    Ready,SchedulingDisabled      master    12d       v1.11.1
node-1    Ready                         <none>    12d       v1.11.0
node-2    Ready                         <none>    5d        v1.11.1
node-3    Ready                         <none>    16h       v1.11.1
learnkarts@master:~$ kubectl uncordon master
node/master uncordoned
learnkarts@master:~$ kubectl get nodes
NAME      STATUS    ROLES     AGE       VERSION
master    Ready     master    12d       v1.11.1
node-1    Ready     <none>    12d       v1.11.0
node-2    Ready     <none>    5d        v1.11.1
node-3    Ready     <none>    16h       v1.11.1
learnkarts@master:~$ $
```

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Maintenance of a node in a kubernetes cluster

- Now, remember that we have deployment with 3 replicas running with each replica on individual node.

```
learnkarts@master:~$ kubectl get pods -o wide
NAME                            READY     STATUS    RESTARTS   AGE       IP                NODE
redis-master-76bc966444-2kg8n   1/1       Running   0          24m       192.168.247.15    node-2
redis-master-76bc966444-p6lt7   1/1       Running   0          24m       192.168.139.69    node-3
redis-master-76bc966444-q84tz   1/1       Running   0          24m       192.168.84.150    node-1
learnkarts@master:~$
```

- Let’s drain the pod running on node-2 and make it ready for upgrade.

```
learnkarts@master:~$ kubectl drain node-1 --ignore-daemonsets
node/node-1 cordoned
WARNING: Ignoring DaemonSet-managed pods: calico-node-cswcs, kube-proxy-klxqf
pod/coredns-78fcdf6894-ml8nq evicted
pod/redis-master-76bc966444-q84tz evicted
learnkarts@master:~$ kubectl get pods -o wide
NAME                            READY     STATUS    RESTARTS   AGE       IP                NODE
redis-master-76bc966444-2kg8n   1/1       Running   0          29m       192.168.247.15    node-2
redis-master-76bc966444-9zhb7   1/1       Running   0          7s        192.168.139.71    node-3
redis-master-76bc966444-p6lt7   1/1       Running   0          29m       192.168.139.69    node-3
learnkarts@master:~$
```

- Here, we see that pod has been evicted to node-3.

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Maintenance of a node in a kubernetes cluster

- Also, check the node status

```
learnkarts@master:~$ kubectl get nodes
NAME      STATUS                        ROLES     AGE       VERSION
master    Ready                         master    12d       v1.11.1
node-1    Ready,SchedulingDisabled      <none>    12d       v1.11.0
node-2    Ready                         <none>    5d        v1.11.1
node-3    Ready                         <none>    16h       v1.11.1
learnkarts@master:~$
```

- Now, directly ssh into the node-1 and perform the upgrade procedure.

```
$sudo apt-get update
$sudo apt upgrade kubelet
```

- Once done, move back to master node and check the status of node-1

```
learnkarts@master:~$ kubectl get nodes
NAME      STATUS                        ROLES     AGE       VERSION
master    Ready                         master    12d       v1.11.1
node-1    Ready,SchedulingDisabled      <none>    12d       v1.11.1
node-2    Ready                         <none>    5d        v1.11.1
node-3    Ready                         <none>    16h       v1.11.1
learnkarts@master:~$
```

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Maintenance of a node in a kubernetes cluster

- Now, uncordon the node-1 and make it ready for scheduling the pods.

```
learnkarts@master:~$ kubectl uncordon node-1
node/node-1 uncordoned
learnkarts@master:~$ kubectl get nodes
NAME      STATUS    ROLES     AGE       VERSION
master    Ready     master    12d       v1.11.1
node-1    Ready     <none>    12d       v1.11.1
node-2    Ready     <none>    5d        v1.11.1
node-3    Ready     <none>    16h       v1.11.1
```

- Now, we have completely upgraded our cluster.
- Same approach would be applied to all the nodes in the cluster

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Headless services

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Headless services

- Headless Service is very import for stateful applications.
- Definition of headless service is similar to that of ‘Normal’ service, but only difference is that is does not have ‘ClusterIP’ .
- By simply defining ‘ClusterIP: none’, headless service gets configured.
- Biggest benefit of headless service that you can directly reach to the respective Pod, otherwise, first it would go to Load Balancer and you would access it through some proxy.

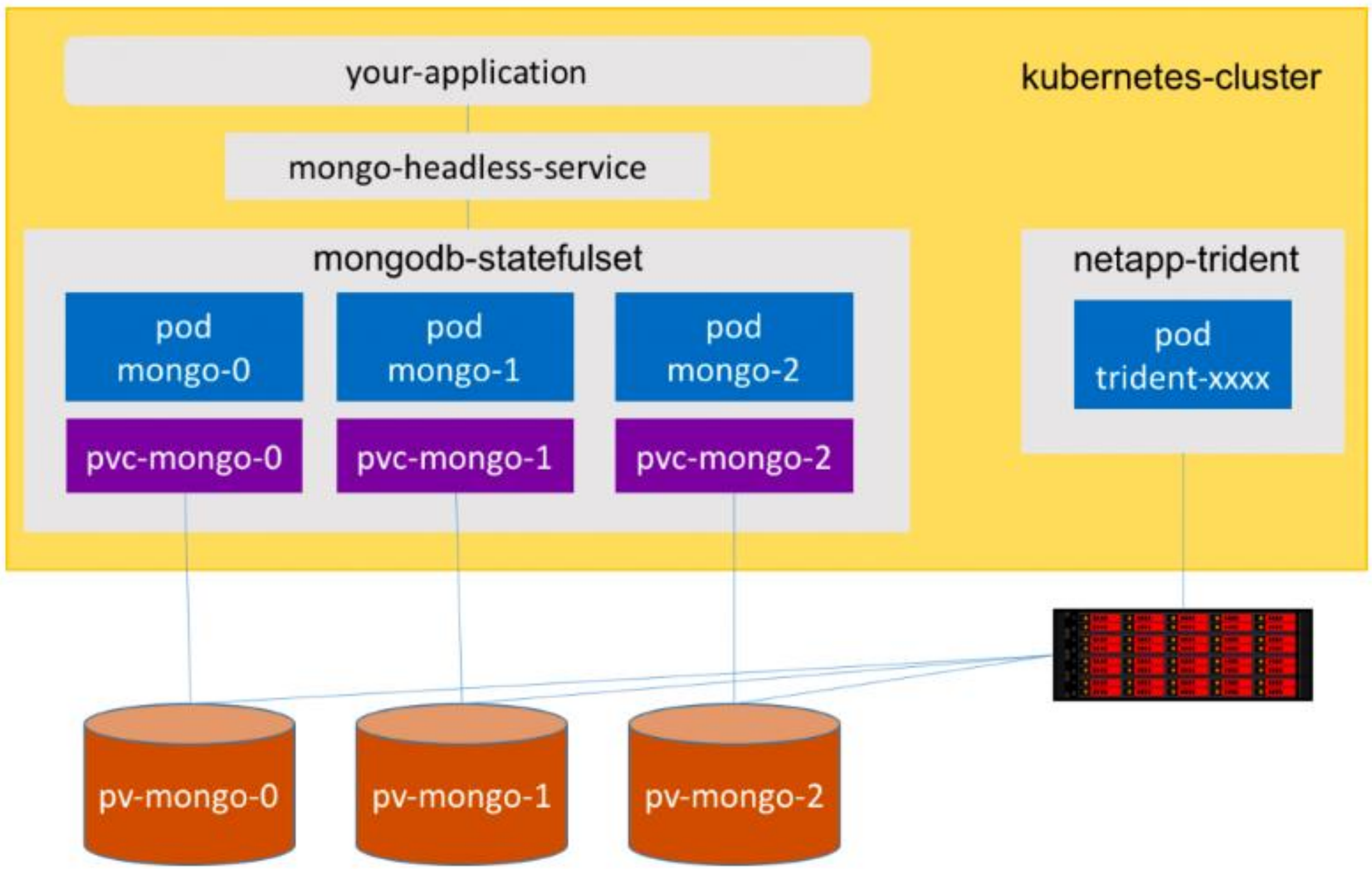
Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Headless services

- Depending upon the ‘Selector’ configuration, DNS is configured for the service.
- **With Selector:** For headless services which has the selector definition, Endpoints records are created in the API and DNS configuration is also modified to return the ‘A’ records, which is mapped to the backend Pods.
- **Without Selector:** In this, endpoint controllers don’t do anything i.e. no records are created. However, DNS system configure either of the things:
  - CNAME records for <external> service
  - ‘A’ record for any other endpoint sharing a name with the services.

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Sample Use Case for Stateful Applications



edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Stateful Set

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Stateful set

- In previous chapter, we learned about the ‘Deployment’.
- **Deployment set** :
  - Makes your the current state of the kubernetes pods to desired state at a controlled rate.
  - But deployment set does not take care of uniqueness of each pods. And that’s where statefulset comes in picture.
- **Stateful set**:
  - Like Deployment set, Stateful set also manages the pods based on similar container spec.
  - Thing it does different from Deployment set is it creates sticky identity to it .

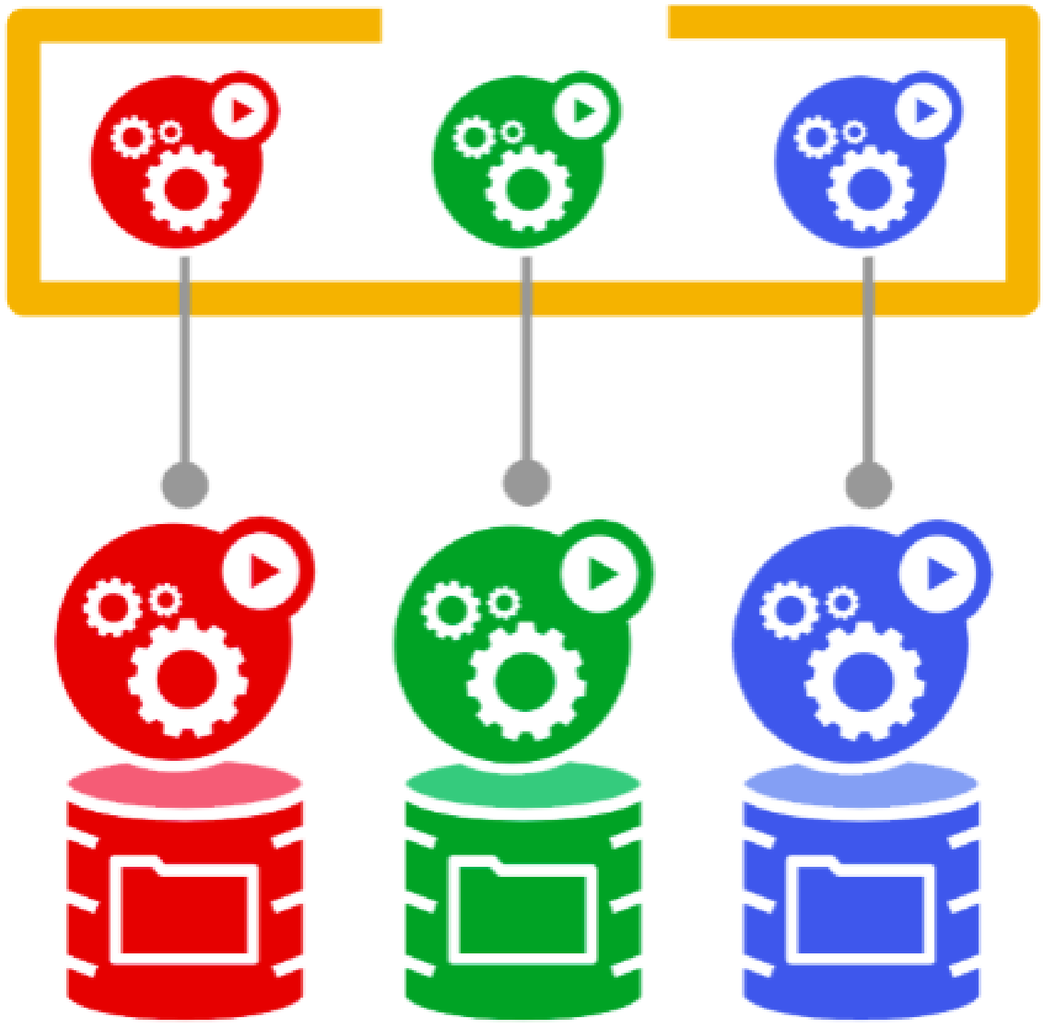
Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Stateful set (cont..)

- **Sticky identity** for each Pods refers to the persistent identifier that is maintained across any scheduling or re-scheduling.
- Pods are deployed in specific order and deleted in reverse order to what they get deployed.

Eg: There is an App which is defined by three different Pods Sev-A, Sev-B, Sev-C.

If there is any change in the specification then they will be rolled out in specific order as defined in the set.



Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Stateful set (Cont..)

- It is one of the most valuable solution for the applications which desired to have one or more of the following scenarios.

Ordered and automated rolling updates

Persistent storage.

Ordered deployment and scaling of pods

Unique network affinity or identifiers.

Ordered deletion and termination of pods.

If any application that doesn't require ordered deployment, scaling, termination or updates then it is better to have a stateless set like Deployment or ReplicaSet for such requirement.

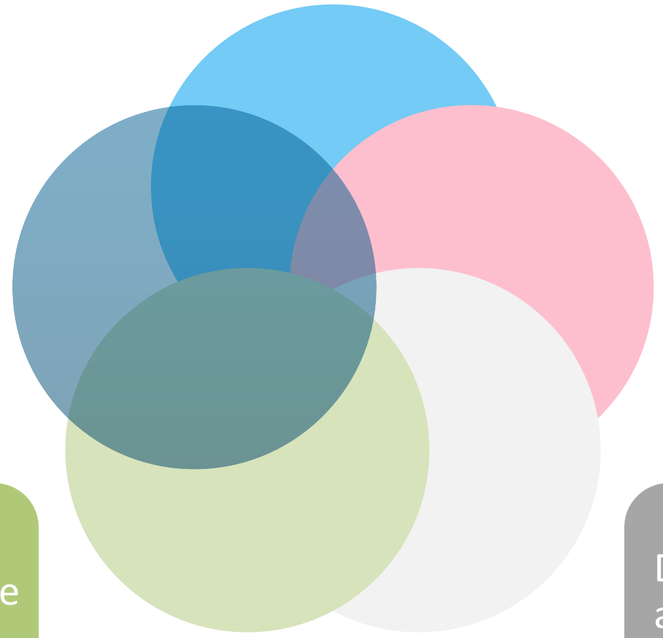
edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Limitations and Pre-requisite for statefulSet

Check the kubernetes version before using it. It is available from v1.9 as beta.

With the current release, StatefulSet requires ‘Headless Service’ for the network identity of the Pods.



Storage for a given Pod must be checked and provisioned accordingly. Admin must ensure the availability of the storage.

If required then respected storage volume should be deleted separetely.

Deletion of Pods will not delete the attached Storage Volume by default. This protects the data.

(Will talk about headless service later in the course)

edureka!

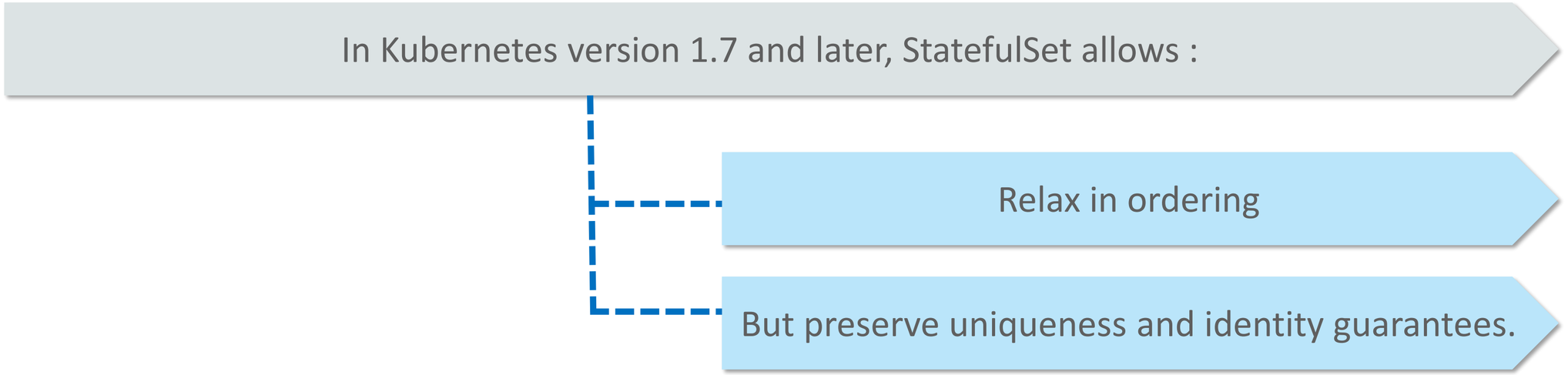
Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Pod management policies

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

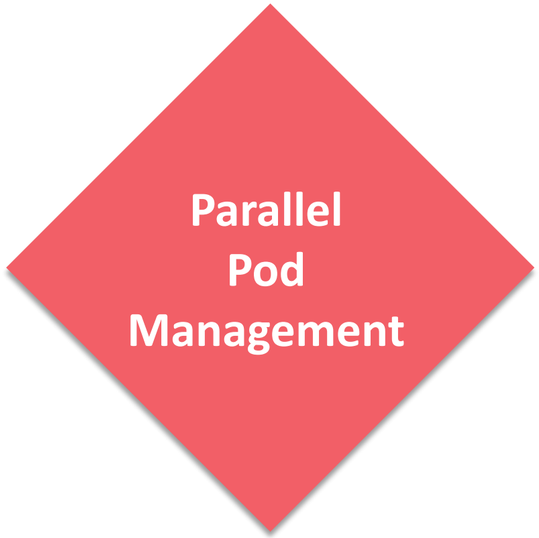
Pod management policies



Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Pod management policies

Kubernetes comes with two sets of Pod Management Policies

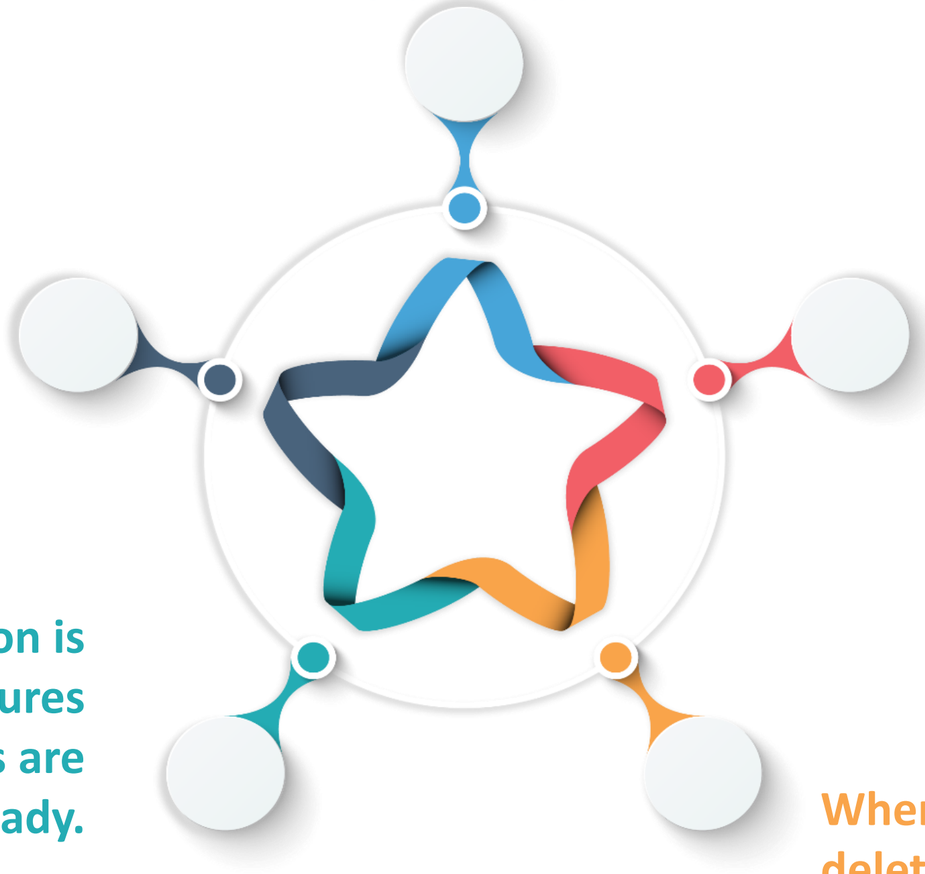


Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Pod management policies : OrderedReady

It is the default policy for StatefulSet.

Before a Pod is terminated, all of its successors must be completely shutdown.



When Pods are being created, during deployment, they are cerated sequentailly in oder from 0..N-1 (for a StatefulSet with N replicas)

Before a scaling operation is executed to a Pod, it ensures that all the predecessors are running and ready.

When Pods are being deleted, they are deleted in reverse order, from (N-1..0).

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Pod management policies : OrderedReady

Let’s try to understand by a use-case.

- Assume, we have 3-tier application running with 3 pods: Web, App, DB.
- For StatefulSet OrderedReady policy, it will ensure that DB starts first and comes up properly.
  - Once DB Pod is up and running successfully then App Pod will come up.
  - Before Web Pod starts , it will ensure that Web Pod is up and running.
    - DB → App → Web (While getting deployed (
    - Failure of any of the Pod, will impact the other Pod deployment.
    - Eg. If DB is up and running but App fails to get deploy then Web will not even start or come in the process.
- Exactly reverse process is applied at the time of termination.
- First Web, once it is successfully terminated, then App and at last it would be DB.

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Pod management policies : OrderedReady

Also, understand the scenario

- At the time of termination, assume
  - Web Pod is successfully terminated
  - And DB fails in between and its status is not up / running. Then App Pod will not get terminated until DB Pod gets fix and status changes to up / running.

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Pod management policies : Parallel Pod Management

- In this, StatefulSet controller will not wait for Predecessors or Successor Pods to complete its operation. i.e.
  - It will not wait for Pod to get “Ready / running status” or get “terminated”
  - It will launch new Pods or terminate all Pods in Parallel.
- Use-case:
  - Batch processing jobs.

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

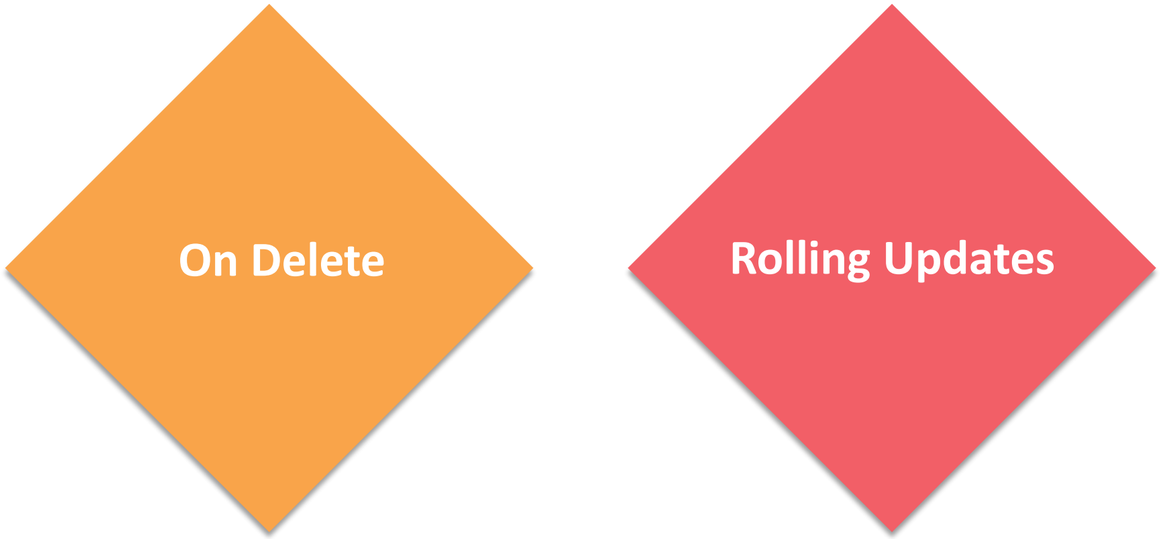
Update strategies

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Update strategies

- In continuation of Pod Management, update strategies are equally important.
- In StatefulSet's under spec , updateStrategy allows to configure and disable automated rolling updates, resource requests, put limits etc. for the Pods.
- Primarily there are two fields which are widely used :



Copyright © 2018, edureka and/or its affiliates. All rights reserved.

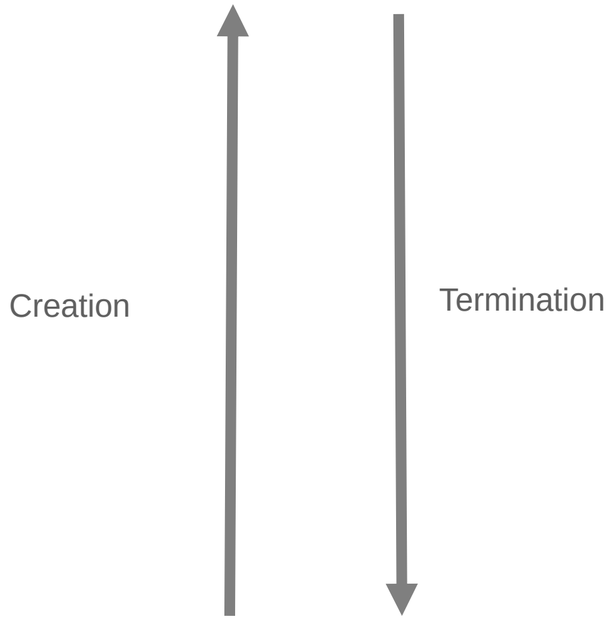
Update strategies : On Delete

1. When .spec.updateStrategy.type is set to OnDelete under the StatefulSet, then controller will not auto-update the Pods.
2. And for update to happen, admin needs to manually delete Pods so that controller can create new Pods.

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Update strategies : Rolling Updates

- This is the default strategy if nothing is been specified under .spec.updateStrategy.
- And default is to implement automated rolling update of the Pods in a StatefulSet.
- For example, it will follow the proper procedure in which Stateful controller will delete and create each Pod one at a time depending upon the status of the previous Pod.



edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Update strategies (Cont...)

- Another useful approach is to have partition your strategy.
- By specifying **.spec.updateStrategy.rollingUpdate.partition**, you can plan your strategy.
- All Pods with an ordinal that is less than to the partition will not be updated.
    - Even if they are deleted, they will be recreated to the previous state.
- Partition also helps when you want to stage updates or perform a phased roll out.
    - If .spec.updateStrategy.rollingUpdate.partition is greater than its .spec.replicas, then updates to the template will not be propagated to the respective Pods.

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Quiz

1. How StatefulSet is different from Deployment Set ?

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Answers

A

1. How StatefulSet is different from Deployment Set ?

Answer : Thing it does different from Deployment set is it creates sticky identity to it .

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Quiz

2. Sticky identity for each Pods refers to the persistent identifier that is maintained across any scheduling or re-scheduling.

   a. True

   b. False

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Answers

A

2. Sticky identity for each Pods refers to the persistent identifier that is maintained across any scheduling or re-scheduling.

   a. True

   b. False

Answer A: True

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Quiz

3. When it is better to use replicaset or Deployment Set instead of StatefulSet. Though both ensure that current state of the kubernetes Pods are same as that of desired state .

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Answers

A

3. When it is better to use replicaset or Deployment Set instead of StatefulSet. Though both ensure that current state of the kubernetes Pods are same as that of desired state .

Answer :
If any application that doesn’t require ordered deployment, scaling, termination or updates then it is better to have a stateless set like Deployment or ReplicaSet for such requirement.

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Quiz

4. With the latest release of kubernetes, It is optional to have ‘headless service’ for the network identity of the Pods, Though for previous release it was mandatory to have.
    a. True
    b. False

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Answers

A

4. With the latest release of kubernetes, It is optional to have ‘headless service’ for the network identity of the Pods, Though for previous release it was mandatory to have.

   a. True

   b. False

Answer B : False

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Quiz

5. What is the default policy for StatefulSet ?

   a. OrderedReady Pod Management Policy

   b. Parallel Pod Management Policy

   c. It is mandatory to define the policy. This field cannot be felt blank

   d. None of these

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Answers

A

5. What is the default policy for StatefulSet ?
   a. OrderedReady Pod Management Policy
   b. Parallel Pod Management Policy
   c. It is mandatory to define the policy. This field cannot be felt blank
   d. None of these

Answer A: OrderedReady Pod Management Policy

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Quiz

6. What do you understand from ‘updateStrategy under StatefulSet ? And which are the two fields which are used primarily?

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Answers

A

6. What do you understand from ‘updateStrategy under StatefulSet ? And which are the two fields which are used primarily?

Answer : In StatefulSet’s under spec , updateStrategy allows to configure and disable automated rolling updates, resource requests, put limits etc. for the Pods. Primarily there are two fields which are widely used :

On Delete

Rolling Updates

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Quiz

7. For headless, service, DNS A record resolves to bunch of records and each record belong to the backend service which resolves to port number and CNAME record.
   a. True
   b. False

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Answers

A

7. For headless, service, DNS A record resolves to bunch of records and each record belong to the backend service which resolves to port number and CNAME record.

   a. True

   b. False

Answer B : False

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Quiz

8. What is the simplest way to configure ‘headless service’ ?

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Answers

A

8. What is the simplest way to configure ‘headless service’ ?

Answer :
By simply defining ‘ClusterIP: none’, headless service gets configured.

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Quiz

9. What are the access modes of PersistentVolume ?

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Answers

A

9. What are the access modes of PersistentVolume ?

Answer :
ReadWriteOnce: The Volume can be mounted as read-write by a single node.
ReadOnlyMany: The Volume can be mounted read-only by many nodes.
ReadWriteMany: The Volume can be mounted as read-write by many nodes.

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Quiz

10. What are the options available to reclaim the PersistentVolume ?

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Answers

A

10. What are the options available to reclaim the PersistentVolume?

Answer :
Retained
Recycled
Deleted

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Quiz

11. Pods are deployed in specific order and deleted in same order to what they get deployed.

    a. True

    b. False

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Answers

A

11. Pods are deployed in specific order and deleted in same order to what they get deployed.

    a. True

    b. False

Answer B: False

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

Summary

- In this module, you should have learnt:
  - Daemon Set
  - Taints and Tolerations
  - Pod Scheduling
  - Service publishing
  - Application environment configuration : ConfigMap
  - Maintenance of a node in a kubernetes cluster
  - Stateful set
  - Pod management policies
  - Cluster DNS
  - Headless services
  - Persistent Volumes

edureka!

Copyright © 2018, edureka and/or its affiliates. All rights reserved.

edureka!

Copyright © 2017, edureka and/or its affiliates. All rights reserved.

edureka!

Copyright © 2017, edureka and/or its affiliates. All rights reserved.

Thank You

For more information please visit our website
www.edureka.co