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

COURSE OUTLINE

Module 03

Introduction To Kubernetes

Kubernetes Architecture

Deploy App To Kubernetes Cluster

Expose, Scale And Update App

Managing State With Deployments

Federations, Auditing And Debugging Kubernetes, Security Best Practices

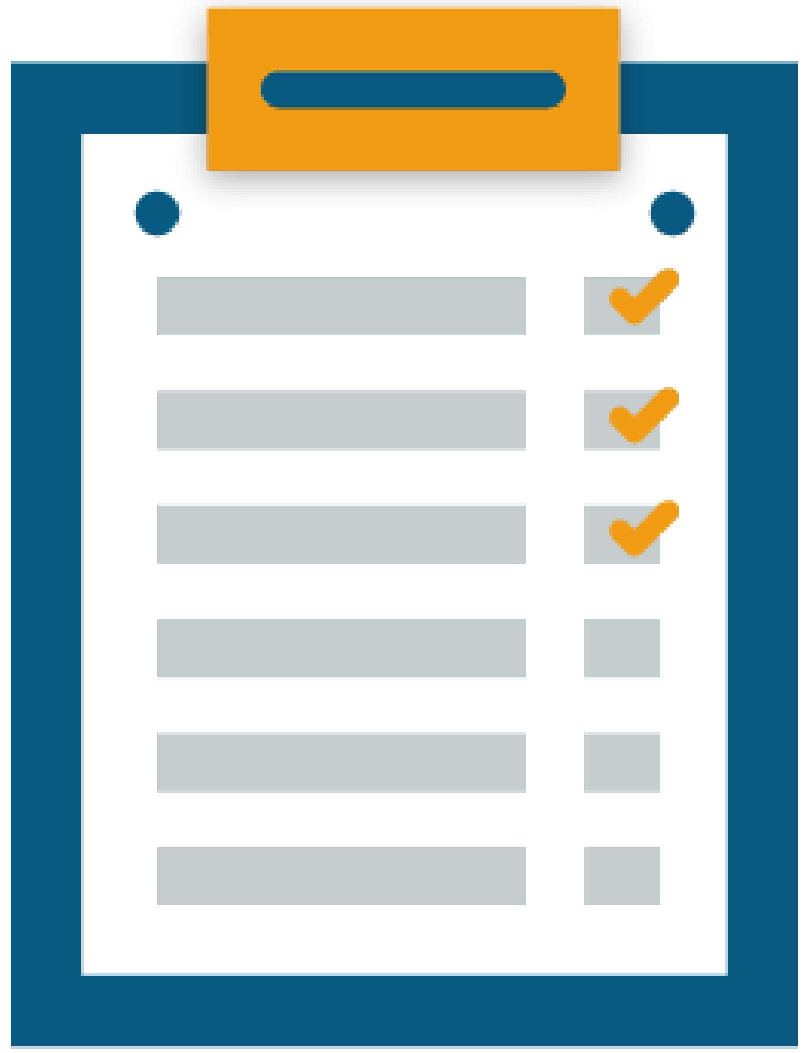
edureka!

Deploy App to Kubernetes Cluster

Objectives

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

- Pod lifecycle and its need
- How to work with Pods to manage multiple containers
- Kubernetes Nodes and their management
- kubectl basic commands
- Containerized App Deployment on Local Kubernetes Cluster



edureka!

What is a Pod?

edureka!

What is a Pod?

A Pod is the basic building block of Kubernetes

A Pod represents a unit of deployment in Kubernetes cluster

It is very easy to horizontally scale a Pod

A Pod encapsulates single or multiple containers along with storage and unique network IP

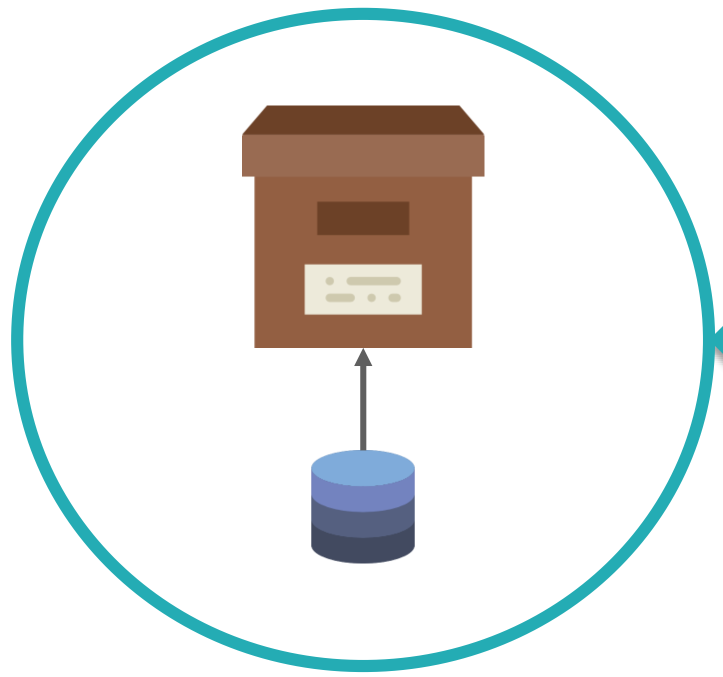
edureka!

Pod

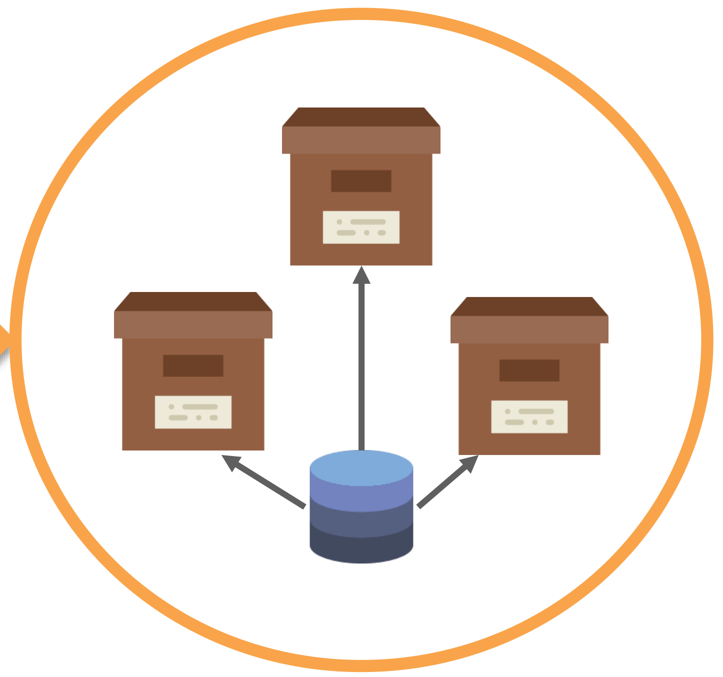
Pod can be consumed in two ways:

POD with a single container

POD with sidecars (multiple containers)



Single Container Pod

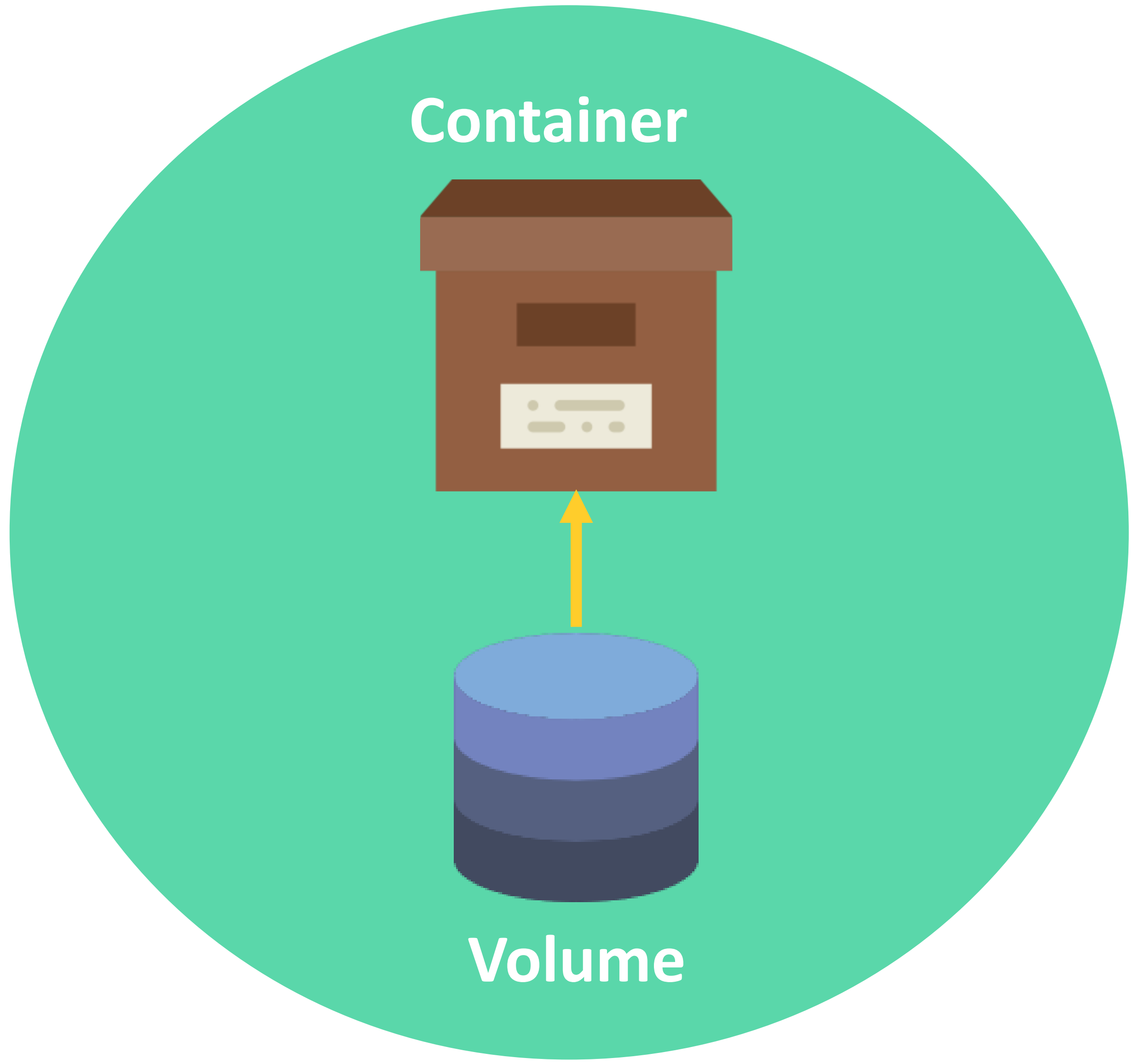


Multi Container Pod

Single Container Pod

- Commonly known as “one container per pod”
- It is widely used when a container runs on a physical machine on top of an operating system. However, the final design is dependent on application needs
- Rest all the features are the same, like
    - **Unique network IP**
    - **Storage volume for persistent data**

Note
One should not run individual pods for multiple instances of the same application

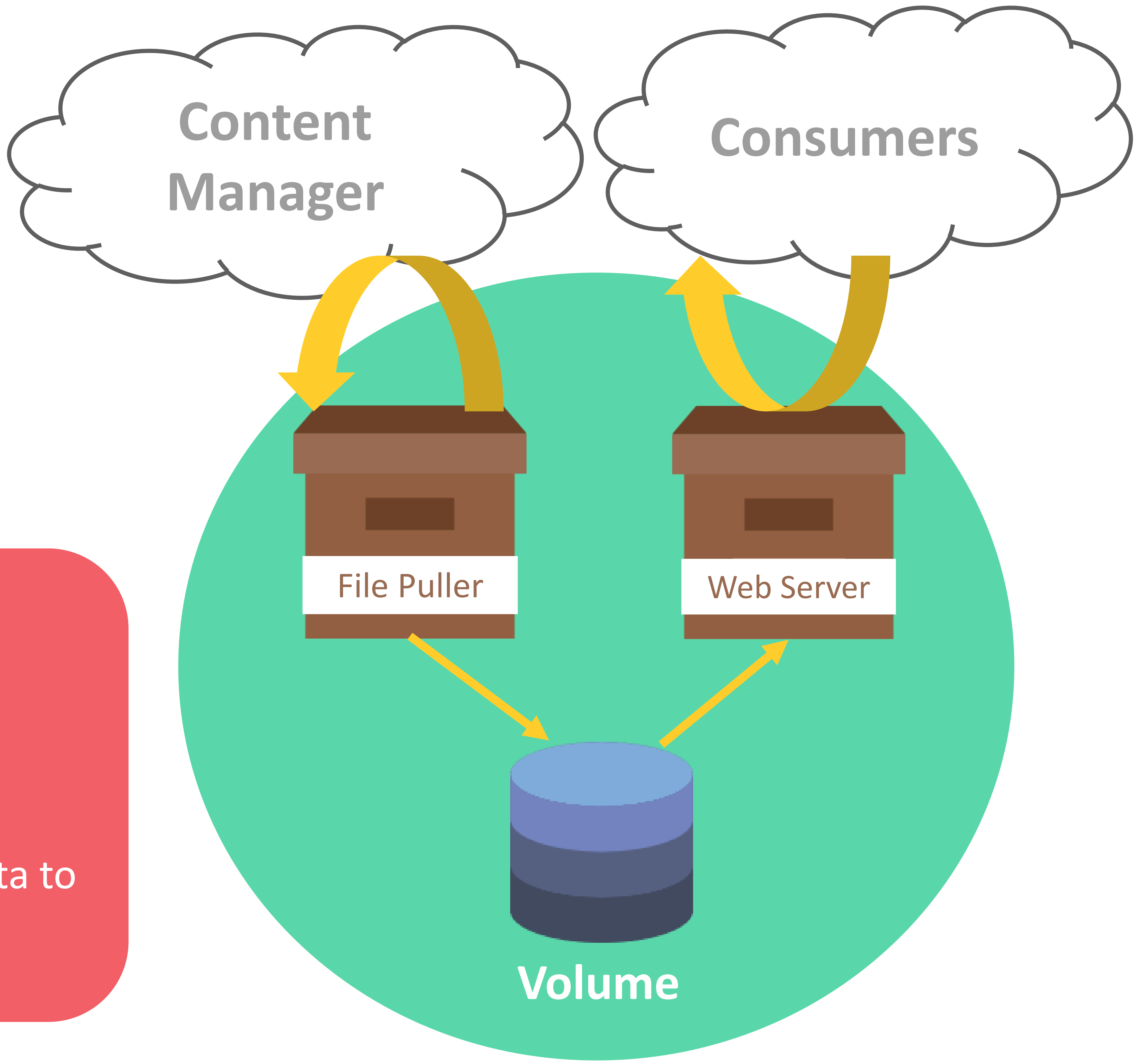


Multi Container Pod

- A common use-case would be for applications which are dependent on other applications/services for their functioning
- The second containers are generally called a **side-car**

Example : Refer to the image on right hand side

- There are two containers that share persistent volume
- One of the containers is writing data to the storage volume
- While another container processes and publishes the same data to its consumer through a web-server



edureka!

Need for a Pod

edureka!

Need for a Pod

Containers can work well alone, but when scaled up, it becomes difficult to apply patches to them and their management becomes cumbersome.

Secondly, orchestrator is required to help in scaling the containers (scale-out or scale-in).

This orchestration capability is provided by Kubernetes.

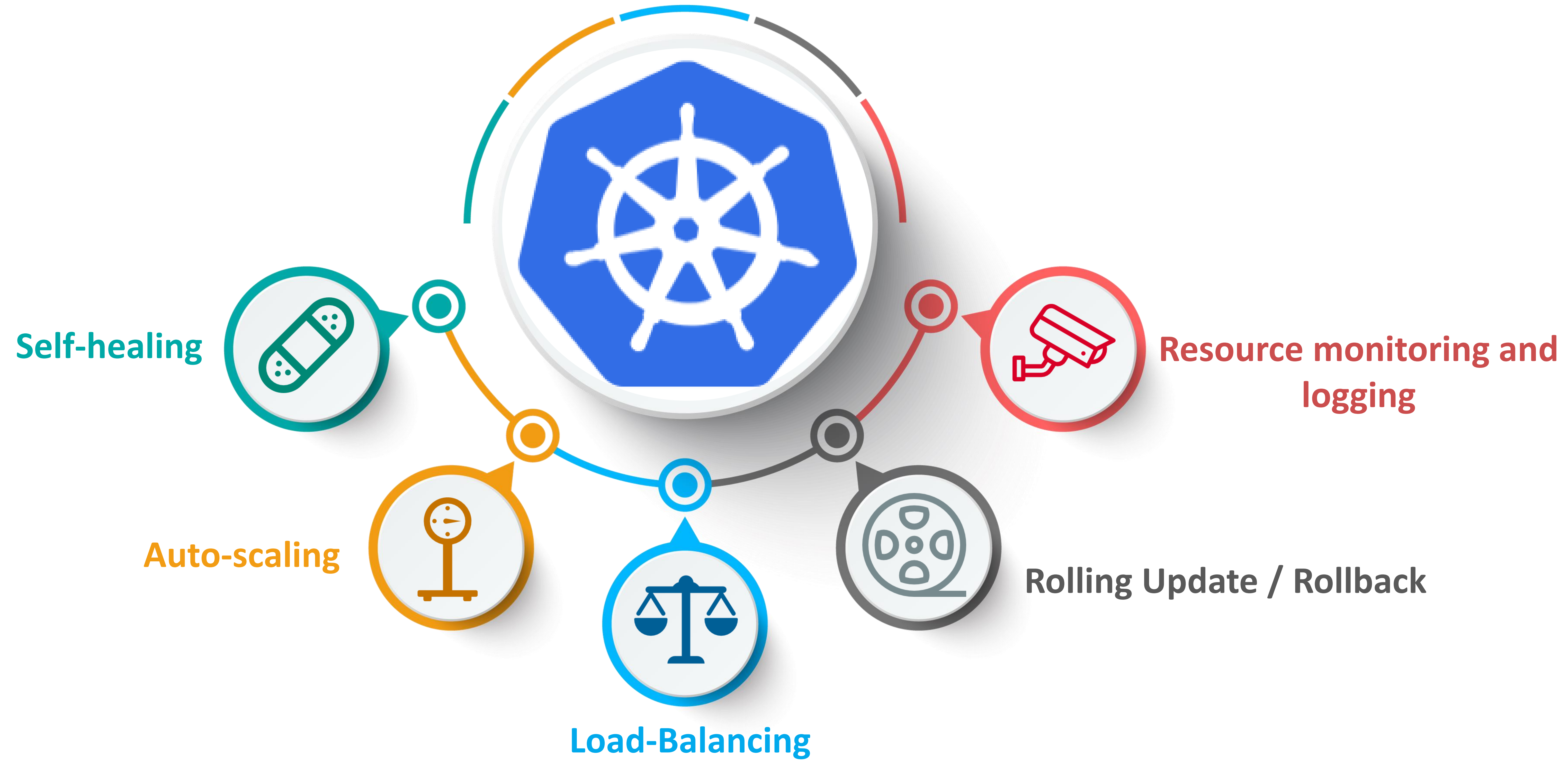
Pods are building blocks of Kubernetes.

Each Pod manages single or multiple containers depending upon the application workload.

Through Pod, one can manage multiple containers as one entity.

edureka!

Pod Features

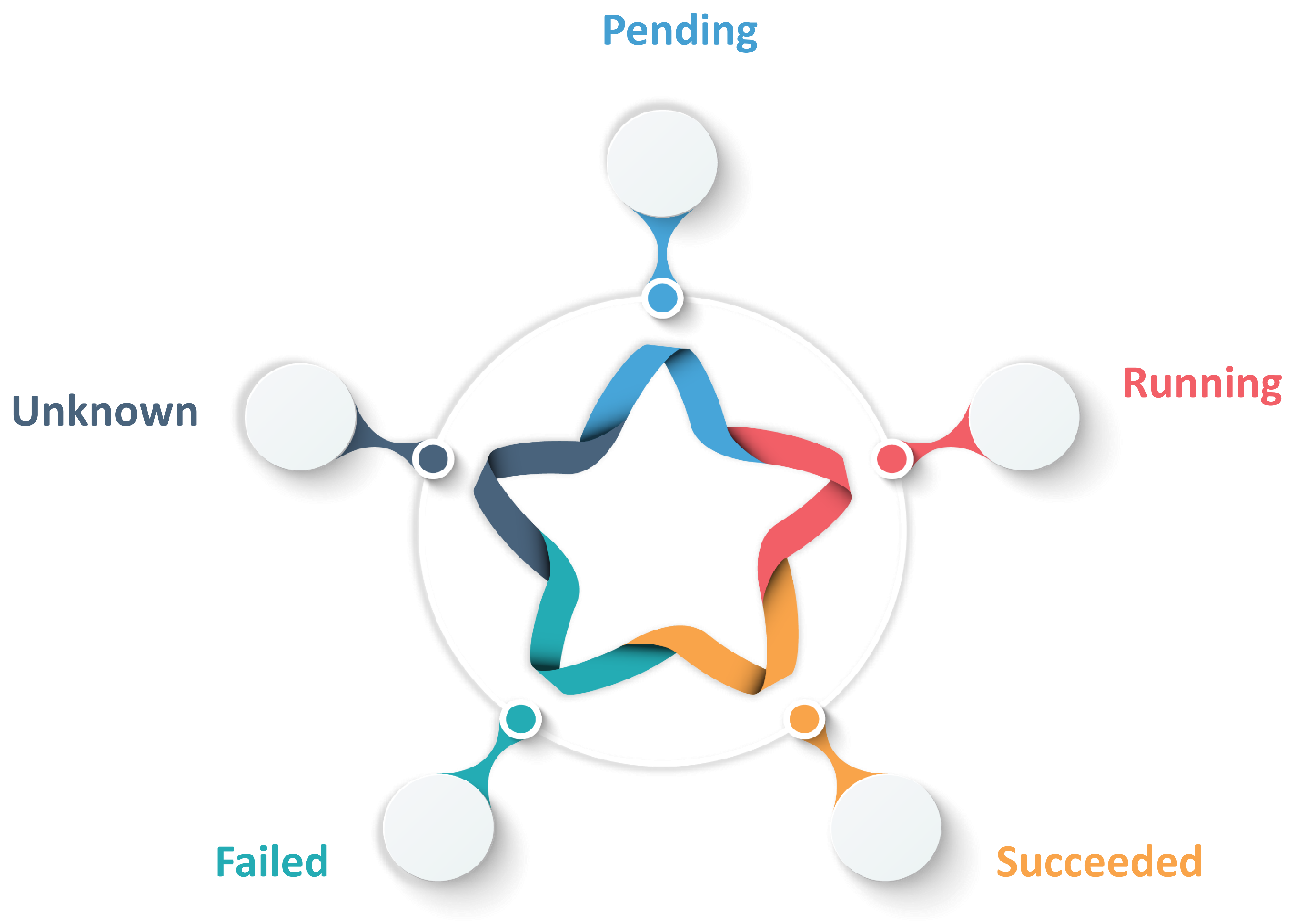


edureka!

Pod Life-Cycle

edureka!

Pod Life-Cycle Phases



Pod Life-Cycle Phases

- **Pending**
- Running
- Succeeded
- Failed
- Unknown

- It reflects the time spent in downloading the container images and creating them
- It also means that system has accepted the Pod

Pod Life-Cycle Phases

- Pending
- **Running**
- Succeeded
- Failed
- Unknown

Pod is tied-up with the node, and at least one container is running

edureka!

Pod Life-Cycle Phases

- Pending
- Running
- **Succeeded**
- Failed
- Unknown

- A container's termination in Kubernetes was successful
- It will not be restarted

edureka!

Pod Life-Cycle Phases

- Pending
- Running
- Succeeded
- **Failed**
- Unknown

When one or more container’s termination is unsuccessful

Termination due to failure is either because of *non-zero exist* status of container

Pod Life-Cycle Phases

- Pending
- Running
- Succeeded
- Failed
- **Unknown**

- When there is a communication problem of a container with the host machine, status of the container cannot be obtained i.e Unknown
- Since, there is no status update, Kubernetes system marks it as "Unknown"
- For such errors, communication channels should be checked first

edureka!

What Are init Containers?

edureka!

init Containers

These containers are like regular containers but meet some special requirements

These containers always run to completion

They can have custom code which is not a part of the application image

'init' containers must run and succeed before any pod can start

Kubernetes keeps on restarting the pod till the 'init' container succeeds

Multiple init Containers

- A pod may can have one or more 'init' containers
- If there are multiple 'init' containers, all of them run before application containers are started
- Since, they have separate images from app containers, they have an added advantage for start-up related codes
- They enable access to some hidden data/secrets that app containers cannot access by default
- They provide additional security by running the utilities/services that are not part of the app containers

Pod Presets

edureka!

Pod Presets

Pod Preset is an API resource which is used to provide additional runtime requirement to a Pod at the time of creation

Pod Template authors need not satisfy each and every detail for creating the pods

Because of Pod Presets, pod template authors need not know every thing about the service they are trying to run

While pods are created, it inherits all the extra information required from the Pod Presets

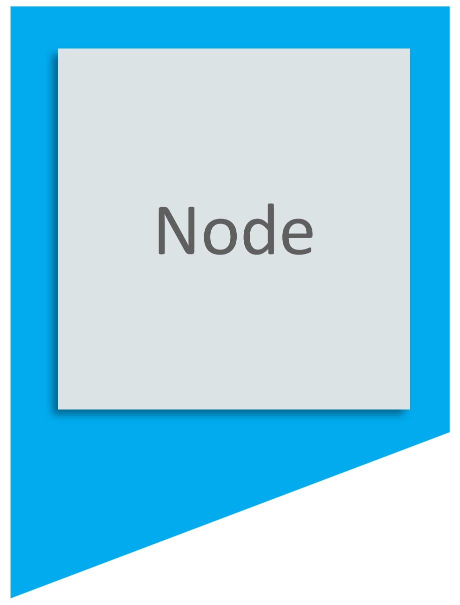
Pod Presets

How to use Pod Presets:

- Use label selectors to select a Pod, to which a Preset is applied, so a set of objects can be identified
- Precaution must be taken that label selectors of two controllers must not overlap within a namespace
- Pod Preset can be applied to none or multiple Pods

Kubernetes Nodes

Kubernetes Nodes

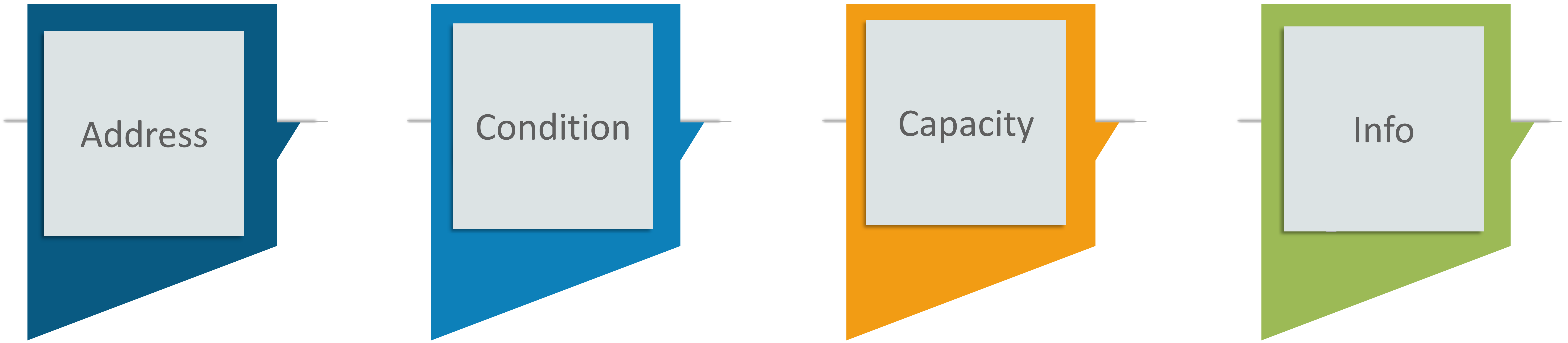


- A node in Kubernetes cluster is the main worker machine
- They are also known as minions
- It could run on a physical machine or a VM
- Node provides all the necesarry services to run Pods
- Node in the Kubernetes system is managed by the master
- Key services running on the node are :

Container (docker, rkt) | Kubelet | Kube-proxy

Kubernetes Node Status

Kubernetes node status is very helpful to plan and manage operations as it provides valuable information. Let's look into each one of them:



Kubernetes Node Status - Address

- This field provides the basic information about the node
- The information may vary depending upon the hardware, cluster, cloud provider, virtual machine (if node is VM)
- Following are the information available in the address field:

Hostname	Container engine provides the hostname, however you can override it and put a more meaningful hostname using `--hostname-override` parameter
Internal IP address	Internally routable IP address within the cluster (for internal communication only)
External IP address	Externally routable IP address to connect to/from outside cluster

edureka!

Kubernetes Node Status - Condition

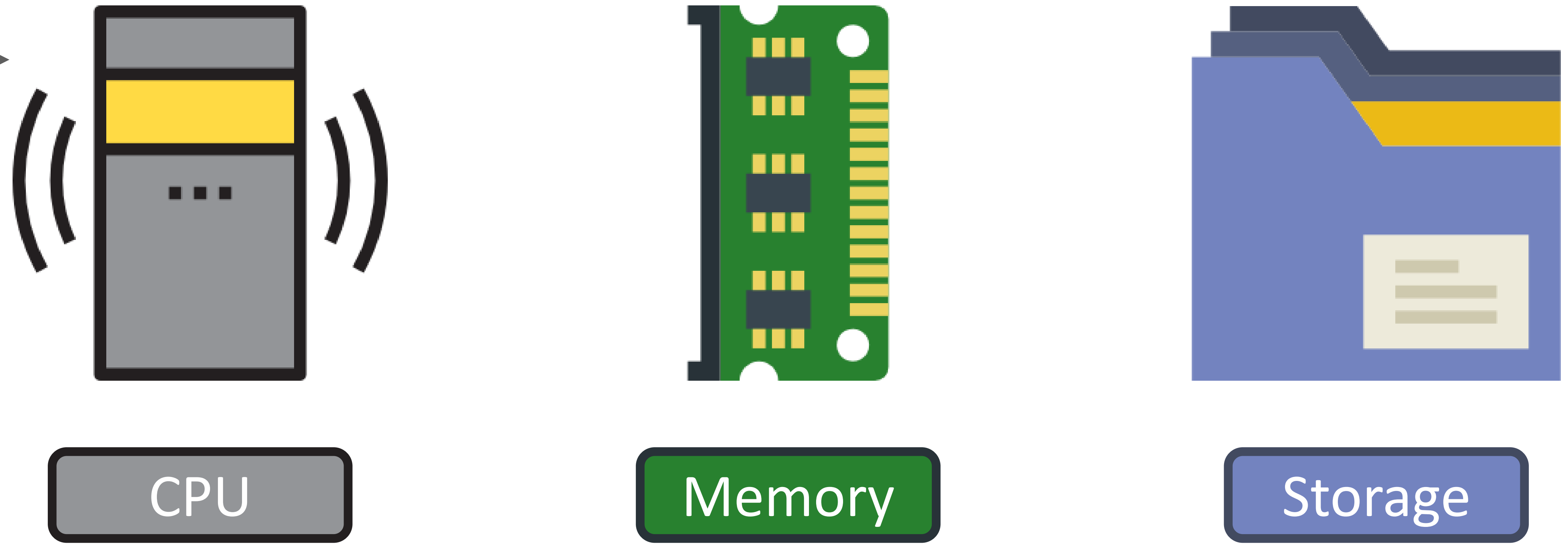
This field describes the status of all running and functional nodes.

OutOfDisk	True, if there is insufficient disk space, otherwise false
DiskPressure	True, if Disk capacity is low, otherwise false
MemoryPressure	True, if the node memory is slow, otherwise false
Networkunavailable	True, if network node is misconfigured, otherwise false
ConfigOK	True, if kubelet configuration is correct, otherwise false
Ready	True, if node is healthy; False, if something is wrong; Unknown, if nothing is heard from the node

Kubernetes Node Status - Capacity

Capacity describes the resources available on the node. Using this information, you can decide on the number of pods that can be scheduled.

These resources could be



Kubernetes Node Status - Info

Info provides the general information about the node, such as:

Kernel details

Operating system details

Kubernetes details, like version and so on

edureka!

Kubernetes Node Management

edureka!

Kubernetes Node Management

- In Kubernetes cluster, node is not inherently created by Kubernetes. They are generally created outside of it, like on a physical node, virtual machine or public cloud
- So, when Kubernetes creates a node:

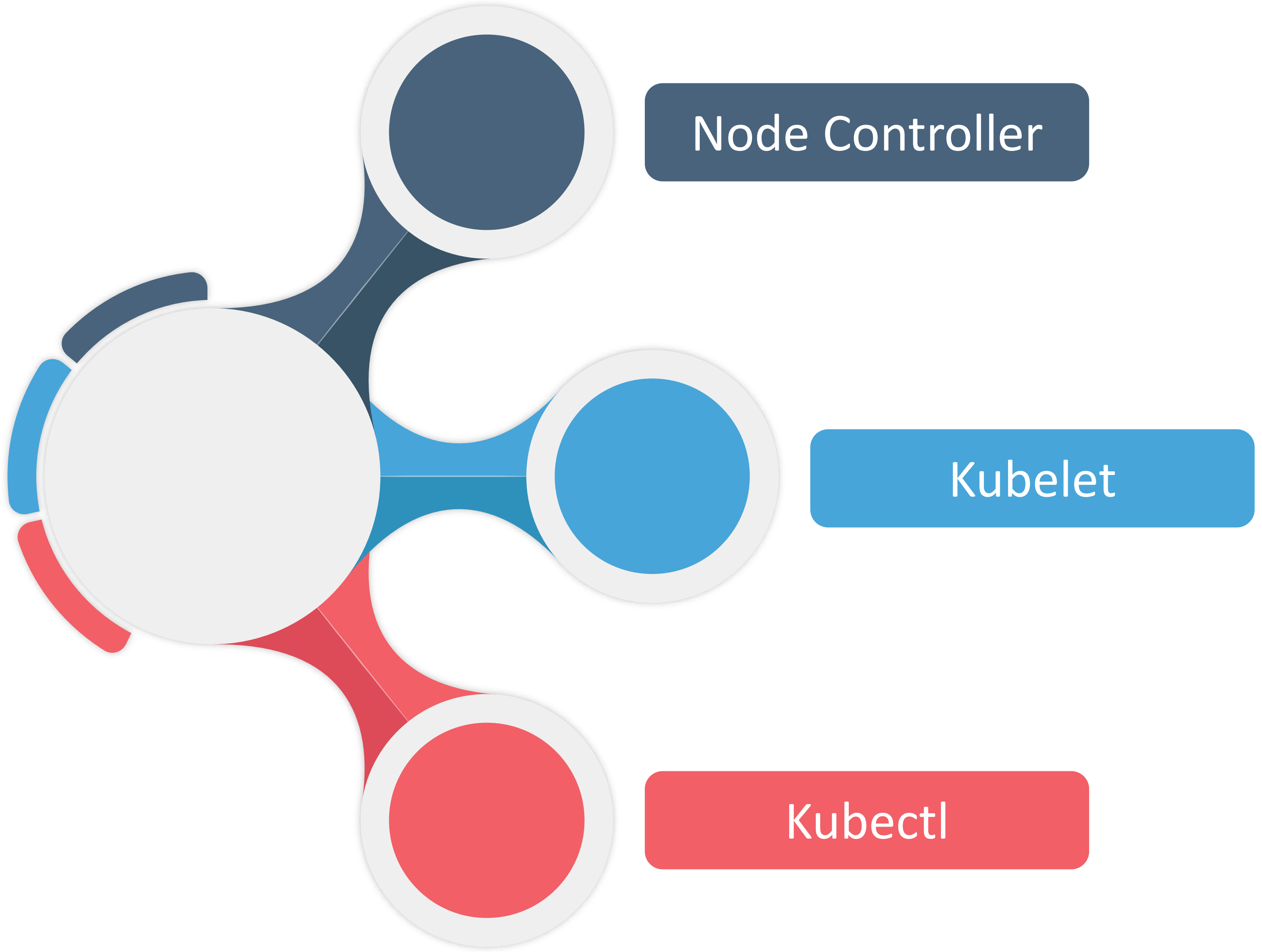
It creates an object to the present node

It validates the object to qualify it for running a Pod

If it doesn't qualify, then it will keep it in invalid state
unless it is deleted or fixed by admin to qualify

Kubernetes Node Management

There are three ways to interact with Kubernetes cluster:



Node Controller

01 It is a component on Kubernetes master

02 It is responsible for multiple tasks / roles

03 First responsibility is to assign a CIDR block to the nodes

04 Second responsibility is to maintain the list of nodes

05 Third responsibility is to monitor the health of the nodes

edureka!

Kubelet

- This is an agent service which runs on each node
- It helps in exposing gathered information from **cAdvisor** about the Pod resource usage
- All the information are exposed using REST API

Kubelet communicates with Master. It acts as a bridge between master and the compute nodes.

PodSpec explains the specification like config, environment and other details.

Kubelet works on PodSpec that are provided to it. PodSpec contains description of containers.

Kubelet makes sure that containers described in the PodSpec are running and are in the desired state

Kubelet only manages the containers that are created by Kubernetes master of the cluster to which it belongs.

edureka!

Kubectl

- This is also an agent service which runs on each node

You need to have some interface or platform using which you can pass-on commands to the cluster.

This interface is provided by kubectl.

For running commands against the Kubernetes clusters, kubectl provides the command line interface.

Kubectl command-line tool supports several different ways to create and manage Kubernetes components.

Let’s see some of the commonly used kubectl commands. However, we will be doing more extensive work on it during our hands-on lab.

Kubectl Basic Commands

edureka!

Kubectl Basic Commands

Kubectl is the command line tool to interact with Kubernetes cluster.

Creating Object

```
//Create resource(s) from file
# kubectl create -f  <filename>.yml

//Create from multiple files
# kubectl create -f  <file1>.yml  -f <file2>.yml

//start a single instance of nginx
# kubectl run nginx --image=nginx
```

Getting Information

```
#kubectl explain pods,svc
```

Kubectl Basic Commands

Viewing, Finding Resources

```
//List all services
# kubectl get services

//List all pods in all namespaces
# kubectl get pods --all-namespaces

//List all pods include uninitialized
# kubectl get pods --include-uninitialized

//List all pods in the namespace, with more details
# kubectl get pods -o wide

//List a particular deployment
# kubectl get deployment <deployment name>

# kubectl describe nodes <node-name>
# kubectl describe pods <pod-name>
```

edureka!

Kubectl Basic Commands

Scaling Resources

```
//Scale a replicaset named 'foo' to 4
# kubectl scale --replicas=4 rs/foo

//Scale a resource specified in "foo.yaml" to 2
# kubectl scale --replicas=2 -f foo.yaml
```

Kubectl Basic Commands

Deleting Resources

```
//Delete a pod using filename.
# kubectl delete -f <filename>.yml

//Delete pods and services with same names "foo" and "foo1"
# kubectl delete pod,service foo foo1

//Delete pods and services with label
# kubectl delete pods,services -l name=<labelname>

//Delete all pods and services, including uninitialized ones,
in namespace
# kubectl -n <namespace> delete po,svc --all
```

edureka!

Kubectl Basic Commands

Interacting with Pods

```
//Show Pods stdout (logs)
# kubectl logs <Pod_name>

//Run command directly on a Pod
# kubectl exec <Pod_name> -- ls /

//Forward traffic of specific port of local machine to mentioned port of Pod
# kubectl port-forward <Pod_name> aaaa:bbbb
            Where, aaaa is port on local machine and bbbb is port on Pod
```

edureka!

Kubectl Basic Commands

Interacting with Nodes & Cluster

```
//Show master and services addresses.
# kubectl cluster-info

//Mark node not available for scheduling Pods
# kubectl cordon <Pod_name>

//Move all Pods from node before maintenance.
# kubectl drain <Pod_name>
```

Containerized App Deployment on Local Kubernetes Cluster

Containerized App Deployment

Before putting the application on Kubernetes cluster, you need to perform the following steps:

- Download an nginx container and check if it’s working
- Create a static web-page
- Create a simple Docker file which will add the static web-page to the nginx container
- Build this Docker file and push it on Docker Hub
- Use this image to deploy it on Kubernetes Cluster

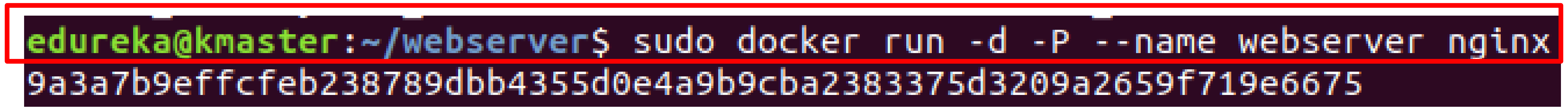
Containerized App Deployment

Deploy a simple docker container for webserver

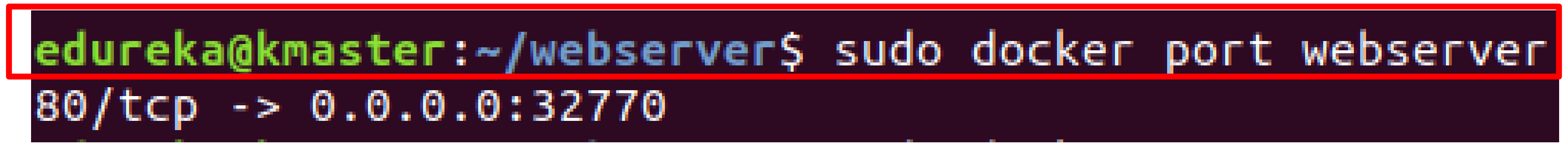
```
$sudo docker run -d -P --name webserver nginx

$sudo docker port webserver
```

Download a nginx container

Output:



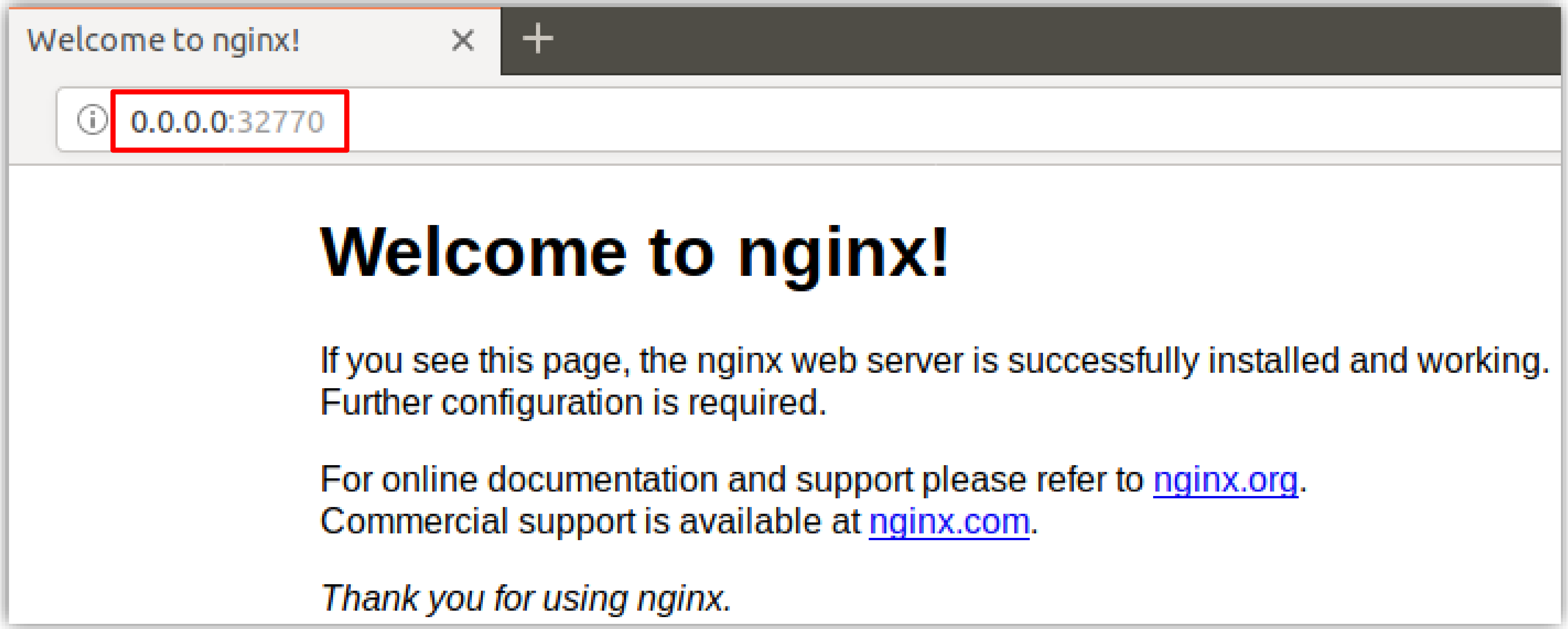
```
edureka@kmaster:~/webserver$ sudo docker run -d -P --name webserver nginx
9a3a7b9effcfeb238789dbb4355d0e4a9b9cba2383375d3209a2659f719e6675
```



```
edureka@kmaster:~/webserver$ sudo docker port webserver
80/tcp -> 0.0.0.0:32770
```

Containerized App Deployment

Open the browser and use the routable IP to your host on which container is running. In our case, it is on host ‘kmaster’: http://kmaster:32770


Welcome to nginx!

0.0.0.0:32770

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

edureka!

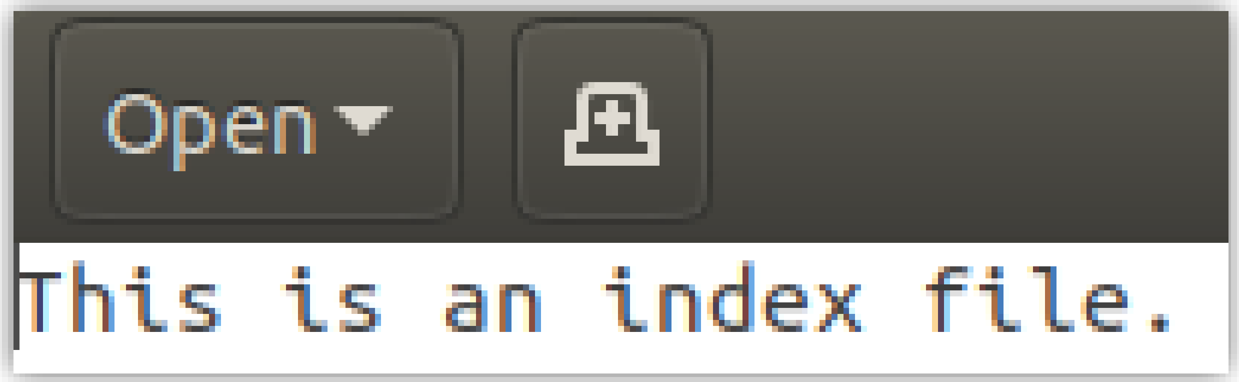
Containerized App Deployment

Create a custom file for a static web-page

```
$mkdir webserver
$cd webserver

$gedit index.html

$cat index.html
```

Create a file 'index.html'

View the contents of the file



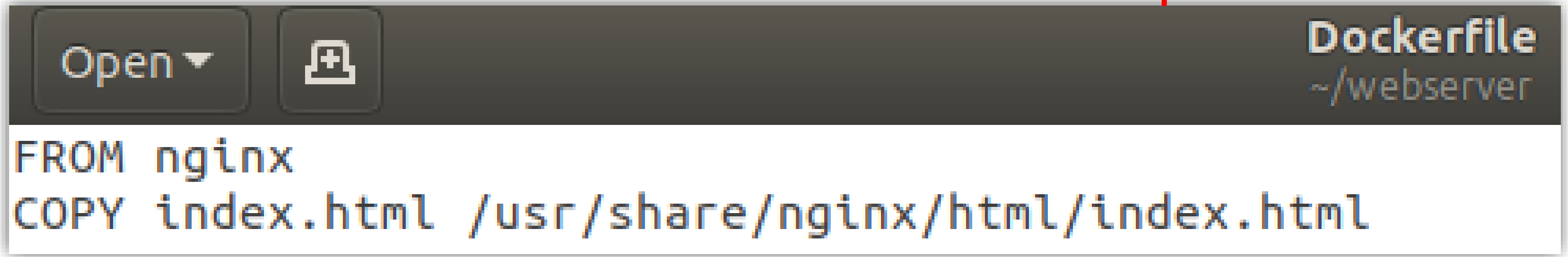
```
This is an index file.
```

Containerized App Deployment

Now, create a Dockerfile

```
$gedit Dockerfile
```

Enter the following inside it



```
FROM nginx
COPY index.html /usr/share/nginx/html/index.html
```

This will add the static web-page to the nginx container

edureka!

Containerized App Deployment

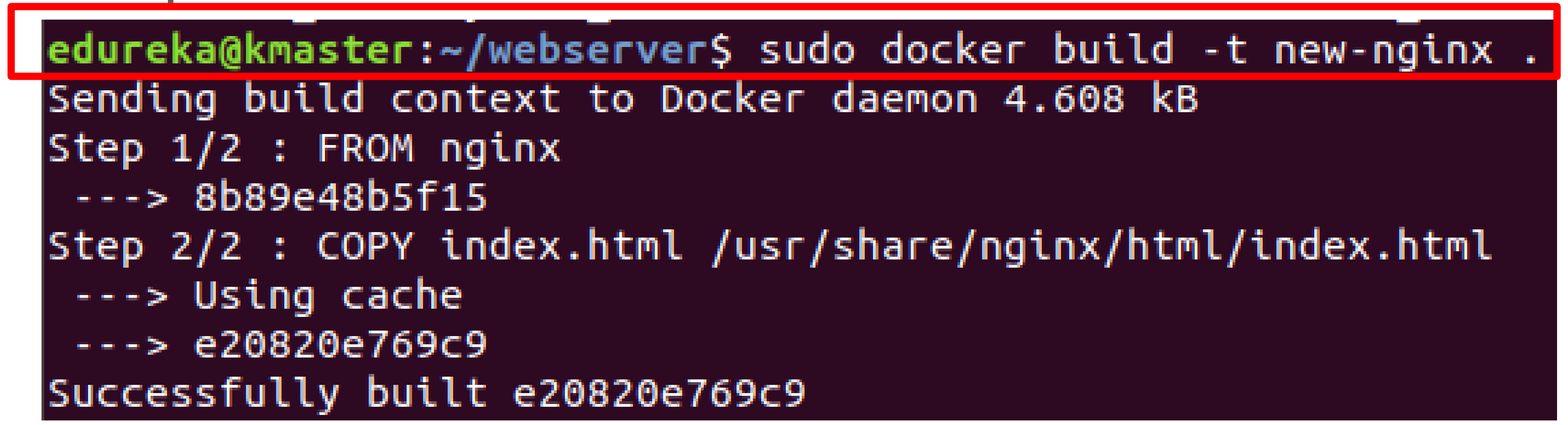
Create a custom container image

```
$sudo docker build -t new-nginx .
$sudo docker images
```

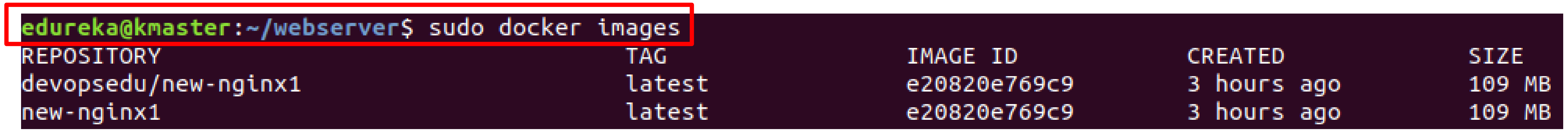
Build the Docker file

Verify the image

Output:



```
edureka@kmaster:~/webserver$ sudo docker build -t new-nginx .
Sending build context to Docker daemon 4.608 kB
Step 1/2 : FROM nginx
 ---> 8b89e48b5f15
Step 2/2 : COPY index.html /usr/share/nginx/html/index.html
 ---> Using cache
 ---> e20820e769c9
Successfully built e20820e769c9
```



```
edureka@kmaster:~/webserver$ sudo docker images
REPOSITORY                  TAG           IMAGE ID        CREATED        SIZE
devopsedu/new-nginx1        latest        e20820e769c9    3 hours ago    109 MB
new-nginx1                  latest        e20820e769c9    3 hours ago    109 MB
```

Containerized App Deployment

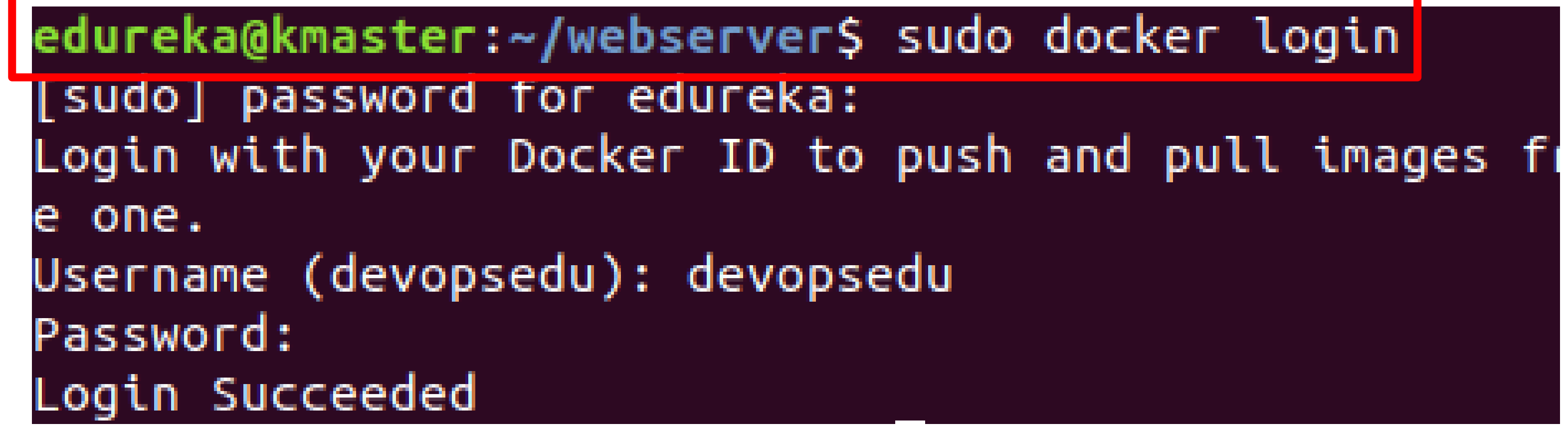
Push the image to the docker hub

```
$sudo docker login

$sudo docker push devopsedu/newnginx1
```

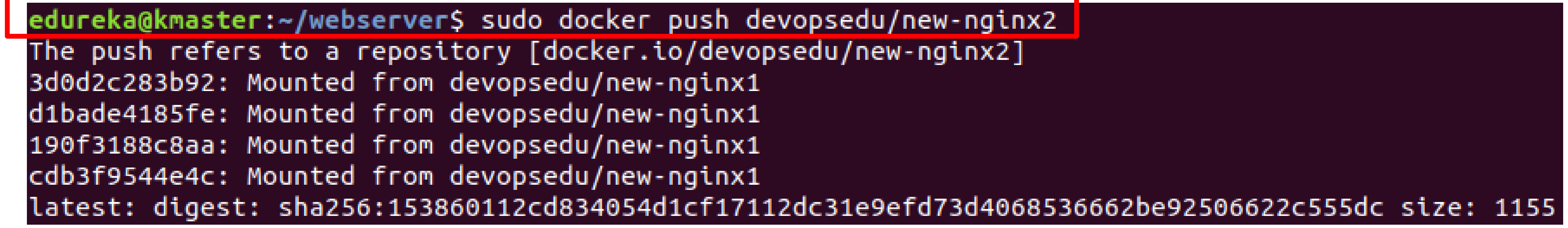
First login into it

Then push the image

Output:



```
edureka@kmaster:~/webserver$ sudo docker login
[sudo] password for edureka:
Login with your Docker ID to push and pull images fr
e one.
Username (devopsedu): devopsedu
Password:
Login Succeeded
```



```
edureka@kmaster:~/webserver$ sudo docker push devopsedu/new-nginx2
The push refers to a repository [docker.io/devopsedu/new-nginx2]
3d0d2c283b92: Mounted from devopsedu/new-nginx1
d1bade4185fe: Mounted from devopsedu/new-nginx1
190f3188c8aa: Mounted from devopsedu/new-nginx1
cdb3f9544e4c: Mounted from devopsedu/new-nginx1
latest: digest: sha256:153860112cd834054d1cf17112dc31e9efd73d4068536662be92506622c555dc size: 1155
```

edureka!

Containerized App Deployment

Now, let’s create a .yaml file to create Kubernetes deployment with 2 replicaset:

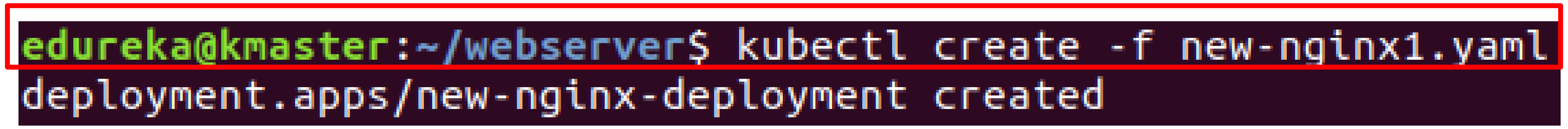
```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
 name: new-nginx-deployment
spec:
 selector:
   matchLabels:
     app: new-nginx1
 replicas: 2
 template:
   metadata:
     labels:
       app: new-nginx1
   spec:
     containers:
     - name: new-nginx
       image: devopsedu/new-nginx1
       ports:
       - containerPort: 80
```

Containerized App Deployment

Create the Kubernetes deployment

```
$kubectl create -f new-nginx1.yaml
```

Output:



```
edureka@kmaster:~/webserver$ kubectl create -f new-nginx1.yaml
deployment.apps/new-nginx-deployment created
```

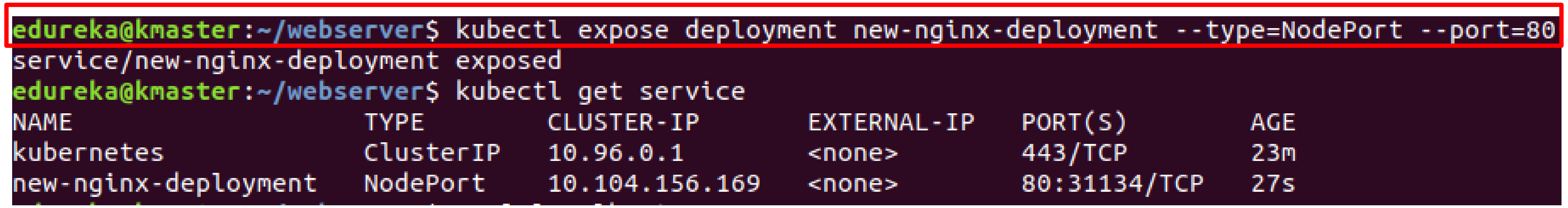
edureka!

Containerized App Deployment

Expose the service to external network and note the port to which it is exposed. Here it is 31134:

```
$kubectl expose deployment new-nginx-deployment --type=NodePort --port=80

$kubectl get service
```

Output:



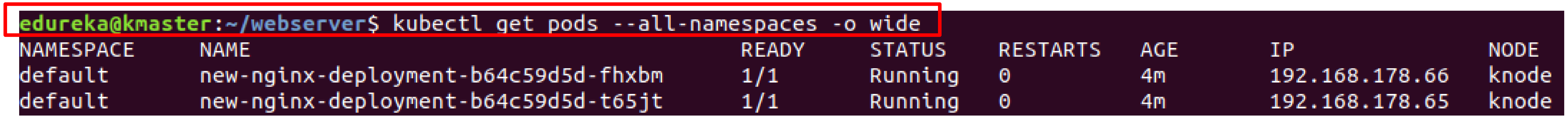
```
edureka@kmaster:~/webserver$ kubectl expose deployment new-nginx-deployment --type=NodePort --port=80
service/new-nginx-deployment exposed
edureka@kmaster:~/webserver$ kubectl get service
NAME                   TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes             ClusterIP   10.96.0.1        <none>        443/TCP        23m
new-nginx-deployment   NodePort    10.104.156.169   <none>        80:31134/TCP   27s
```

Containerized App Deployment

Note the node on which it is running

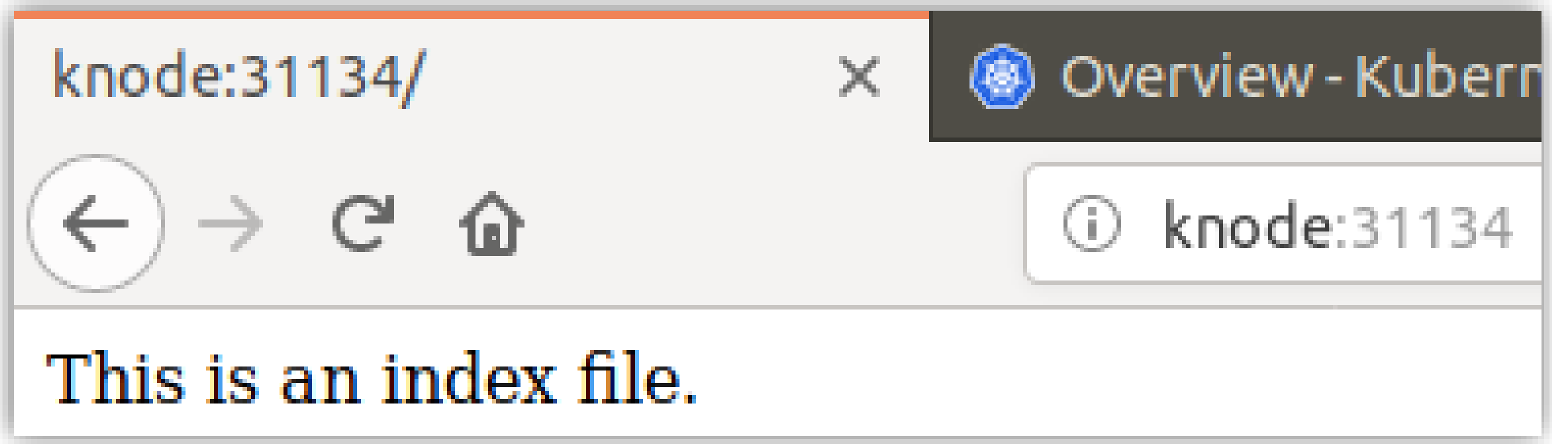
```
$kubectl get pods -o wide
```

Output:



```
edureka@kmaster:~/webserver$ kubectl get pods --all-namespaces -o wide
NAMESPACE     NAME                                   READY     STATUS    RESTARTS   AGE       IP               NODE
default       new-nginx-deployment-b64c59d5d-fhxbm   1/1       Running   0          4m        192.168.178.66   knode
default       new-nginx-deployment-b64c59d5d-t65jt   1/1       Running   0          4m        192.168.178.65   knode
```

Now, use knode:31134 to browse through



This is an index file.

List All Local Deployments

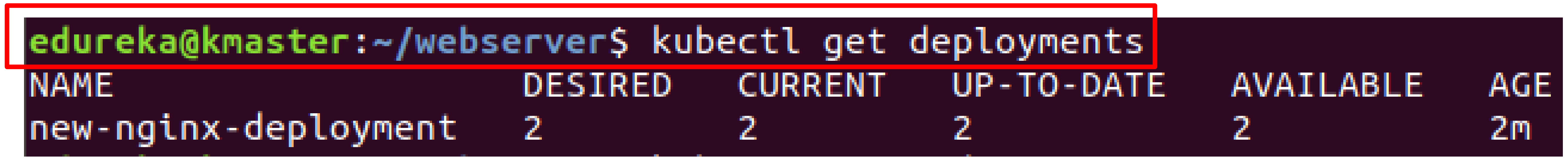
edureka!

List All Local Deployment

To list all local deployments, use the following command

```
$kubectl get deployments
```

Output:



```
edureka@kmaster:~/webserver$ kubectl get deployments
NAME                   DESIRED    CURRENT    UP-TO-DATE    AVAILABLE    AGE
new-nginx-deployment   2          2          2             2            2m
```

edureka!

Create a kubectl Proxy

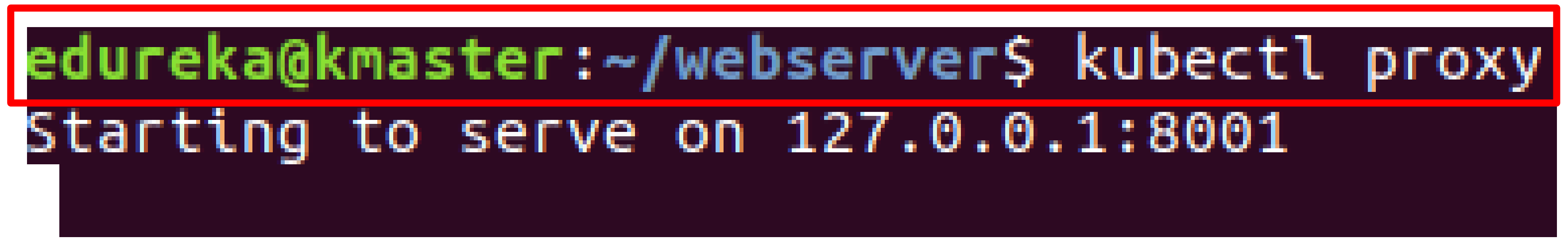
Create a kubectl Proxy

Create a kubectl proxy for forwarding communication to cluster-wide private network.

Start the proxy

```
$ kubectl proxy
Starting to serve on 127.0.0.1:8001
```

Output:



```
edureka@kmaster:~/webserver$ kubectl proxy
Starting to serve on 127.0.0.1:8001
```

edureka!

Curl to Verify if App is Running

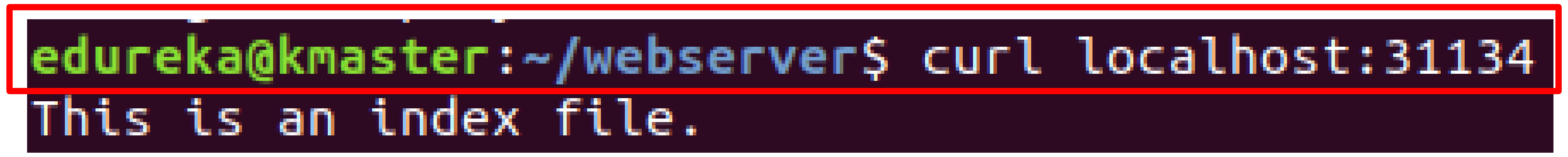
edureka!

Curl to Verify if App is Running

Curl to verify that app is running using the following command

```
$ curl localhost:31134
This is an index file.
```

Output:



```
edureka@kmaster:~/webserver$ curl localhost:31134
This is an index file.
```

edureka!

List All Existing Pods

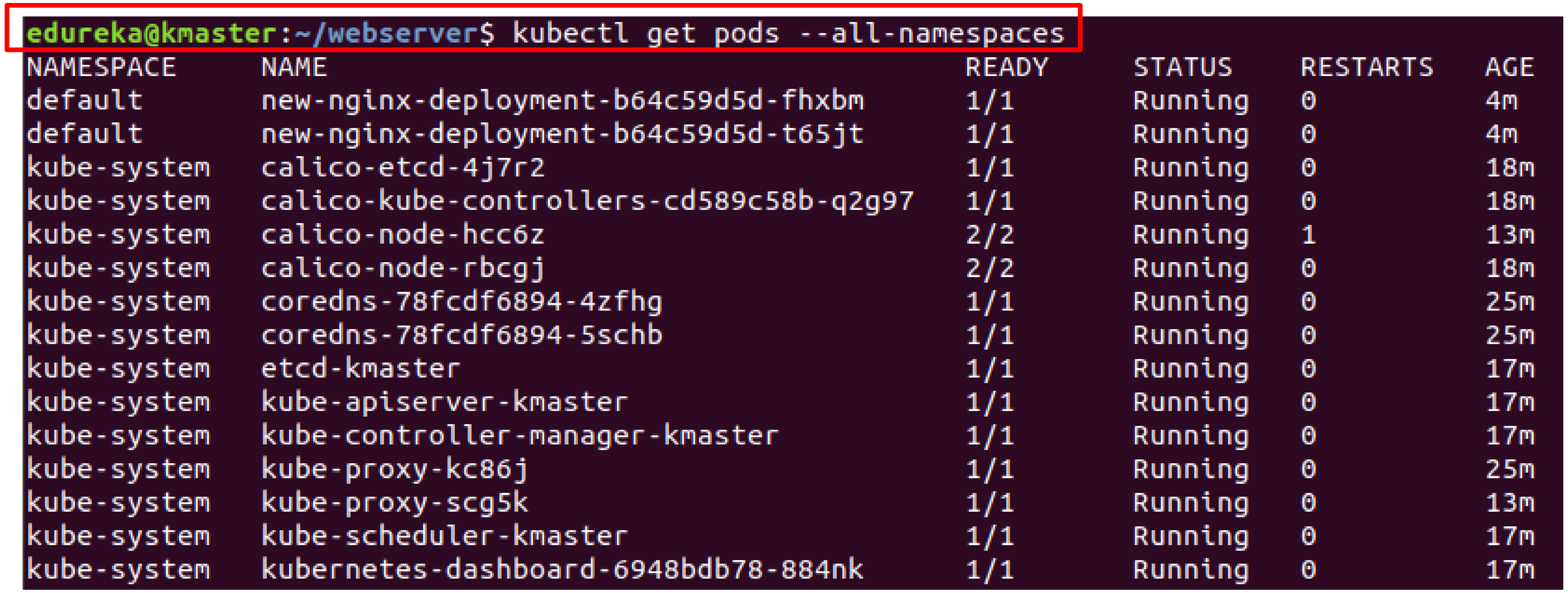
edureka!

List All Existing Pods

List all existing Pods using the following command

```
$kubectl get pods --all-namespaces
```

Output:



```
edureka@kmaster:~/webserver$ kubectl get pods --all-namespaces
NAMESPACE     NAME                                        READY    STATUS    RESTARTS   AGE
default       new-nginx-deployment-b64c59d5d-fhxbm        1/1      Running   0          4m
default       new-nginx-deployment-b64c59d5d-t65jt        1/1      Running   0          4m
kube-system   calico-etcd-4j7r2                           1/1      Running   0          18m
kube-system   calico-kube-controllers-cd589c58b-q2g97     1/1      Running   0          18m
kube-system   calico-node-hcc6z                           2/2      Running   1          13m
kube-system   calico-node-rbcgj                           2/2      Running   0          18m
kube-system   coredns-78fcdf6894-4zfhg                    1/1      Running   0          25m
kube-system   coredns-78fcdf6894-5schb                    1/1      Running   0          25m
kube-system   etcd-kmaster                                1/1      Running   0          17m
kube-system   kube-apiserver-kmaster                      1/1      Running   0          17m
kube-system   kube-controller-manager-kmaster             1/1      Running   0          17m
kube-system   kube-proxy-kc86j                            1/1      Running   0          25m
kube-system   kube-proxy-scg5k                            1/1      Running   0          13m
kube-system   kube-scheduler-kmaster                      1/1      Running   0          17m
kube-system   kubernetes-dashboard-6948bdb78-884nk        1/1      Running   0          17m
```

edureka!

Get Description of a Specific Pod

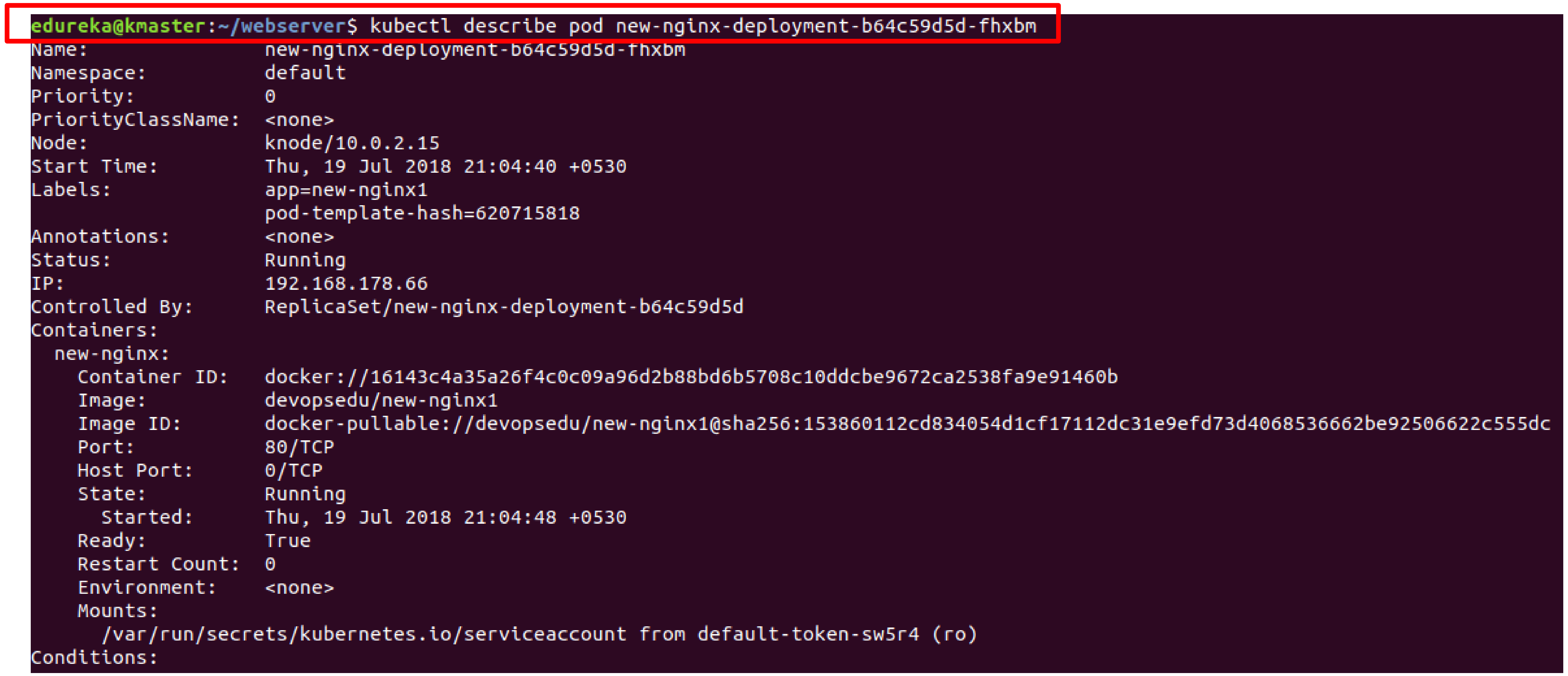
edureka!

Get Description of a Specific Pod

Get description of a specific Pod using the following command

```
$kubectl describe pod <pod-name>
```

Output:



```
edureka@kmaster:~/webserver$ kubectl describe pod new-nginx-deployment-b64c59d5d-fhxbm
Name:               new-nginx-deployment-b64c59d5d-fhxbm
Namespace:          default
Priority:           0
PriorityClassName:  <none>
Node:               knode/10.0.2.15
Start Time:         Thu, 19 Jul 2018 21:04:40 +0530
Labels:             app=new-nginx1
                    pod-template-hash=620715818
Annotations:        <none>
Status:             Running
IP:                 192.168.178.66
Controlled By:      ReplicaSet/new-nginx-deployment-b64c59d5d
Containers:
  new-nginx:
    Container ID:   docker://16143c4a35a26f4c0c09a96d2b88bd6b5708c10ddcbe9672ca2538fa9e91460b
    Image:          devopsedu/new-nginx1
    Image ID:       docker-pullable://devopsedu/new-nginx1@sha256:153860112cd834054d1cf17112dc31e9efd73d4068536662be92506622c555dc
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Thu, 19 Jul 2018 21:04:48 +0530
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-sw5r4 (ro)
Conditions:
```

edureka!

View Logs of the Container

View Logs of the Container

View logs of the container using the following commands

```
$docker ps -a

$docker logs <container-id>

$kubectl get pods

$kubectl logs <podname>
```

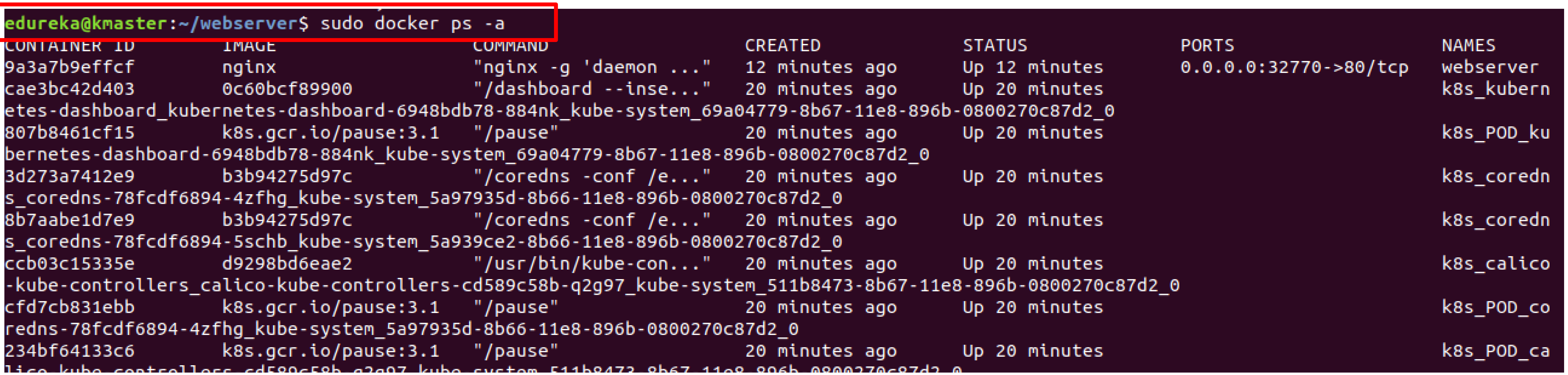
Show all containers

Get the container Id

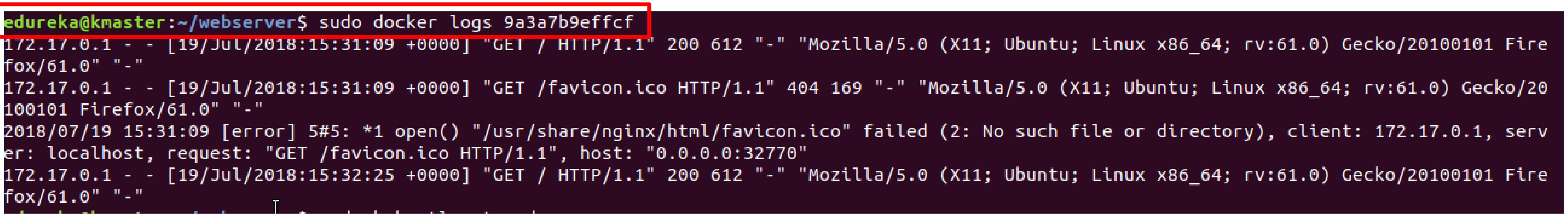
Get the Pod name

edureka!

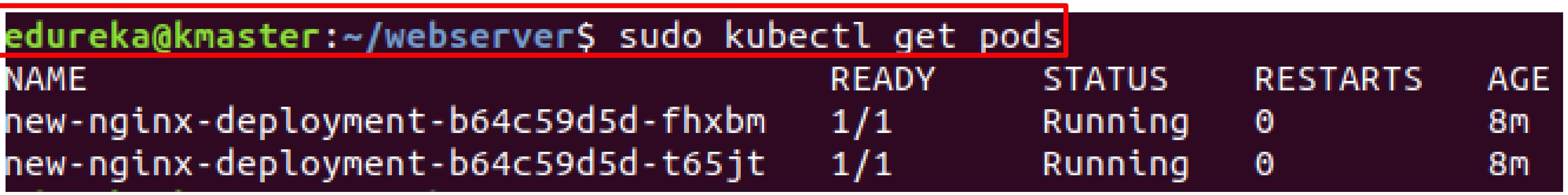
View Logs of the Container



```
edureka@kmaster:~/webserver$ sudo docker ps -a
CONTAINER ID        IMAGE                    COMMAND                  CREATED             STATUS              PORTS                  NAMES
9a3a7b9effcf        nginx                    "nginx -g 'daemon ..."   12 minutes ago      Up 12 minutes       0.0.0.0:32770->80/tcp   webserver
cae3bc42d403        0c60bcf89900             "/dashboard --inse..."   20 minutes ago      Up 20 minutes                              k8s_kubern
etes-dashboard_kubernetes-dashboard-6948bdb78-884nk_kube-system_69a04779-8b67-11e8-896b-0800270c87d2_0
807b8461cf15        k8s.gcr.io/pause:3.1   "/pause"                 20 minutes ago      Up 20 minutes                              k8s_POD_ku
bernetes-dashboard-6948bdb78-884nk_kube-system_69a04779-8b67-11e8-896b-0800270c87d2_0
3d273a7412e9        b3b94275d97c             "/coredns -conf /e..."   20 minutes ago      Up 20 minutes                              k8s_coredn
s_coredns-78fcdf6894-4zfhg_kube-system_5a97935d-8b66-11e8-896b-0800270c87d2_0
8b7aabe1d7e9        b3b94275d97c             "/coredns -conf /e..."   20 minutes ago      Up 20 minutes                              k8s_coredn
s_coredns-78fcdf6894-5schb_kube-system_5a939ce2-8b66-11e8-896b-0800270c87d2_0
ccb03c15335e        d9298bd6eae2             "/usr/bin/kube-con..."   20 minutes ago      Up 20 minutes                              k8s_calico
-kube-controllers_calico-kube-controllers-cd589c58b-q2g97_kube-system_511b8473-8b67-11e8-896b-0800270c87d2_0
cfd7cb831ebb        k8s.gcr.io/pause:3.1   "/pause"                 20 minutes ago      Up 20 minutes                              k8s_POD_co
redns-78fcdf6894-4zfhg_kube-system_5a97935d-8b66-11e8-896b-0800270c87d2_0
234bf64133c6        k8s.gcr.io/pause:3.1   "/pause"                 20 minutes ago      Up 20 minutes                              k8s_POD_ca
lico-kube-controllers-cd589c58b-q2g97_kube-system_511b8473-8b67-11e8-896b-0800270c87d2_0
```



```
edureka@kmaster:~/webserver$ sudo docker logs 9a3a7b9effcf
172.17.0.1 - - [19/Jul/2018:15:31:09 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Fire
fox/61.0" "-"
172.17.0.1 - - [19/Jul/2018:15:31:09 +0000] "GET /favicon.ico HTTP/1.1" 404 169 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20
100101 Firefox/61.0" "-"
2018/07/19 15:31:09 [error] 5#5: *1 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory), client: 172.17.0.1, serv
er: localhost, request: "GET /favicon.ico HTTP/1.1", host: "0.0.0.0:32770"
172.17.0.1 - - [19/Jul/2018:15:32:25 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Fire
fox/61.0" "-"
```



```
edureka@kmaster:~/webserver$ sudo kubectl get pods
NAME                                  READY     STATUS    RESTARTS   AGE
new-nginx-deployment-b64c59d5d-fhxbm   1/1       Running   0          8m
new-nginx-deployment-b64c59d5d-t65jt   1/1       Running   0          8m
```

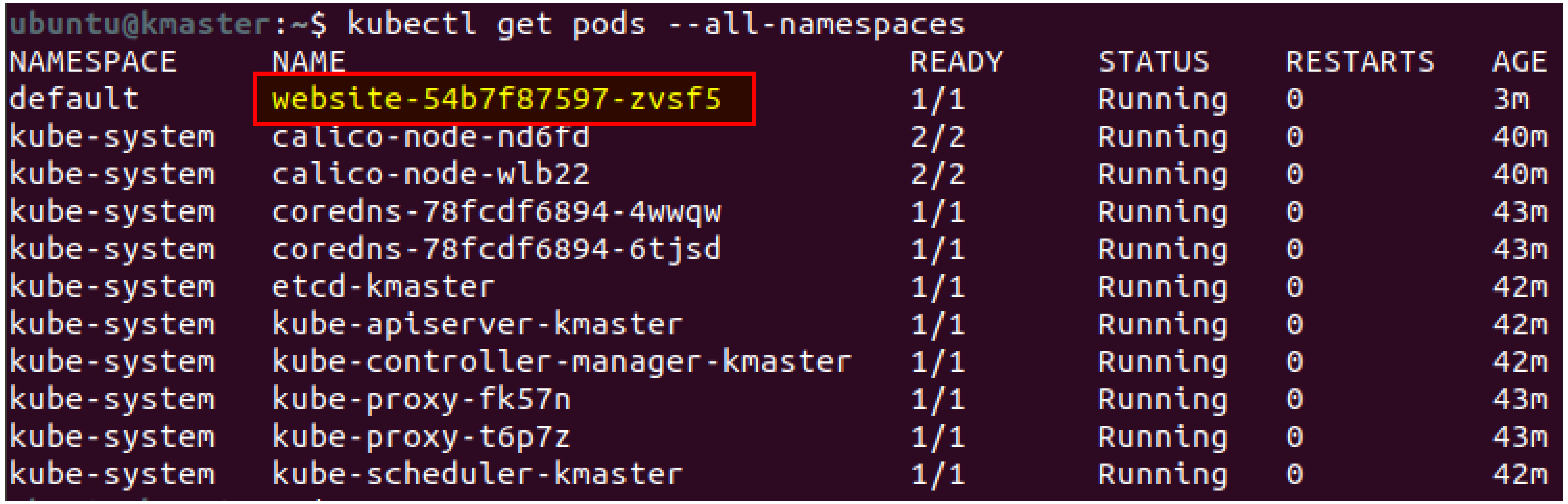
Executing Commands on Container

edureka!

Execute Commands on Container

Let's see which pods are running on our cluster, we have to get into website pod's container

```
$ kubectl get pods –all-namespaces
```



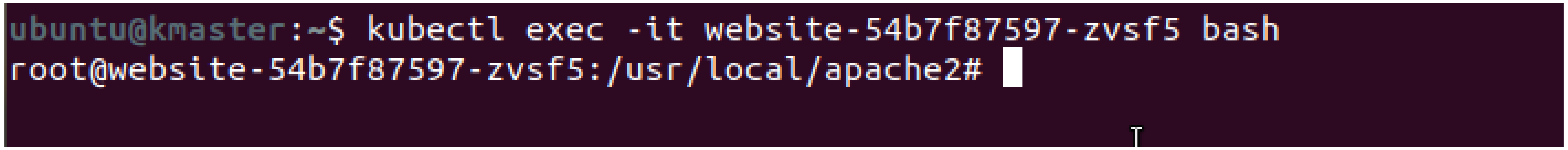
```
ubuntu@kmaster:~$ kubectl get pods --all-namespaces
NAMESPACE     NAME                               READY     STATUS    RESTARTS   AGE
default       website-54b7f87597-zvsf5           1/1       Running   0          3m
kube-system   calico-node-nd6fd                  2/2       Running   0          40m
kube-system   calico-node-wlb22                  2/2       Running   0          40m
kube-system   coredns-78fcdf6894-4wwqw           1/1       Running   0          43m
kube-system   coredns-78fcdf6894-6tjsd           1/1       Running   0          43m
kube-system   etcd-kmaster                       1/1       Running   0          42m
kube-system   kube-apiserver-kmaster             1/1       Running   0          42m
kube-system   kube-controller-manager-kmaster    1/1       Running   0          42m
kube-system   kube-proxy-fk57n                   1/1       Running   0          43m
kube-system   kube-proxy-t6p7z                   1/1       Running   0          43m
kube-system   kube-scheduler-kmaster             1/1       Running   0          42m
```

edureka!

Execute Commands on Container

The below command, will get you inside the pod's container

```
$ kubectl exec -it website-54b7f87597-zvsf5 bash
```



```
ubuntu@kmaster:~$ kubectl exec -it website-54b7f87597-zvsf5 bash
root@website-54b7f87597-zvsf5:/usr/local/apache2#
```

Quiz

Quiz

1. State any 2 characteristics of Pods.

Answers

1. State any 2 characteristics of Pods.

Answer :

A POD is the basic building block of kubernetes.

A POD encapsulates single or multiple containers along with storage and unique network IP.

A POD represents a unit of deployment in kubernetes cluster.

It is very easy to horizontally scale POD.

Quiz

2. What are sidecars ?

Answers

2. What are sidecars ?

Answer :

Two or more container inside the pod are called sidecars

Quiz

3. Name any three lifecycle phases of pod.

Answers

3. Name any three lifecycle phases of pod.

Answer :

- Running
- Succeeded
- Failed

edureka!

Quiz

4. From the below three ways, which is the most common way to access the Kubernetes cluster ?

   a. Node controller

   b. kubectl

   c. kubelet

Answers

4. From the below three ways, which is the most common way to access the Kubernetes cluster ?

   a. Node controller

   b. [✓ kubectl]

   c. kubelet

Answer B: kubectl

Quiz

5. Provide the commands for :

   a. List all the services running on the Kubernetes cluster

   b. List all the pods with their namespaces

Answers

5. **Provide the commands for :**

    a. **List all the services running on the Kubernetes cluster**

    b. **List all the pods with their namespaces**

Answer :

```
# kubectl get services                    # List all services
# kubectl get pods --all-namespaces       # List all pods in all namespaces
```

Quiz

6. Define kubelet and give any two characteristics of kubelet.

Answers

6. Define kubelet and give any two characteristics of kubelet.

Answer :

Kubelet communicates with Master. It acts as a bridge between master and the compute nodes.

Get the podSpec from Master. podSpec is the specification (config, environment and other details).

Kubelet works on PodSpec that are provided to it and ensure that containers (docker , rkt) which are described in the PodSpec are running and health.

It only manages the container that are created by Kubernetes master to which it belongs.

edureka!

Quiz

7. Provide commands for following use-case

   a. Create resource from file

   b. Create resource from multiple files

   c. Start single instance of nginx

   d. Command should be using kubectl

Answers

7. Provide command for following use-case

   a. Create resource from file

   b. Create resource from multiple files

   c. Start single instance of nginx

   d. Command should be using kubectl

Answer :

```
#kubectl create -f  <filename>.yml                    # create resource(s) from file
# kubectl create -f  <file1>.yml  -f <file2>.yml          # create from multiple files
# kubectl run nginx --image=nginx                    # start a single instance of nginx
```

edureka!

Summary

In this module, you should have learnt:

- How to work with pods to manage multiple containers
- Node Management
- kubectl basic commands
- How to containerize app deployment on local Kubernetes cluster

Questions

edureka!

Comments

Ratings

Suggestions

FEEDBACK

Survey

Likes

Ideas

Thank You

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