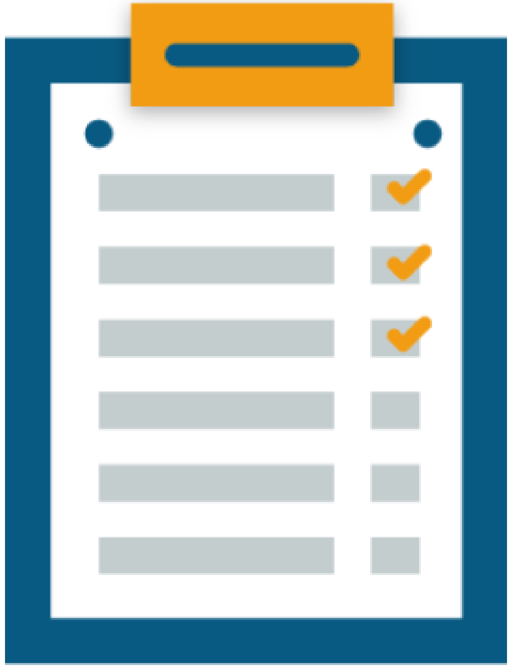
Containerization Using Kubernetes

Objectives

At the end of this module, you will be able to:

- Understand the basics of Kubernetes
- Understand Kubernetes Architecture
- Set up a Kubernetes Cluster on Ubuntu VMs
- Deploy your first app on Kubernetes using YAML file
- Deploying an On-Prem application to Kubernetes using Dashboard
- Update your Application Pod using Blue Green Deployment on Kubernetes



Revising Containers

Containers

“A container is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime environment, system tools, system libraries and settings”.

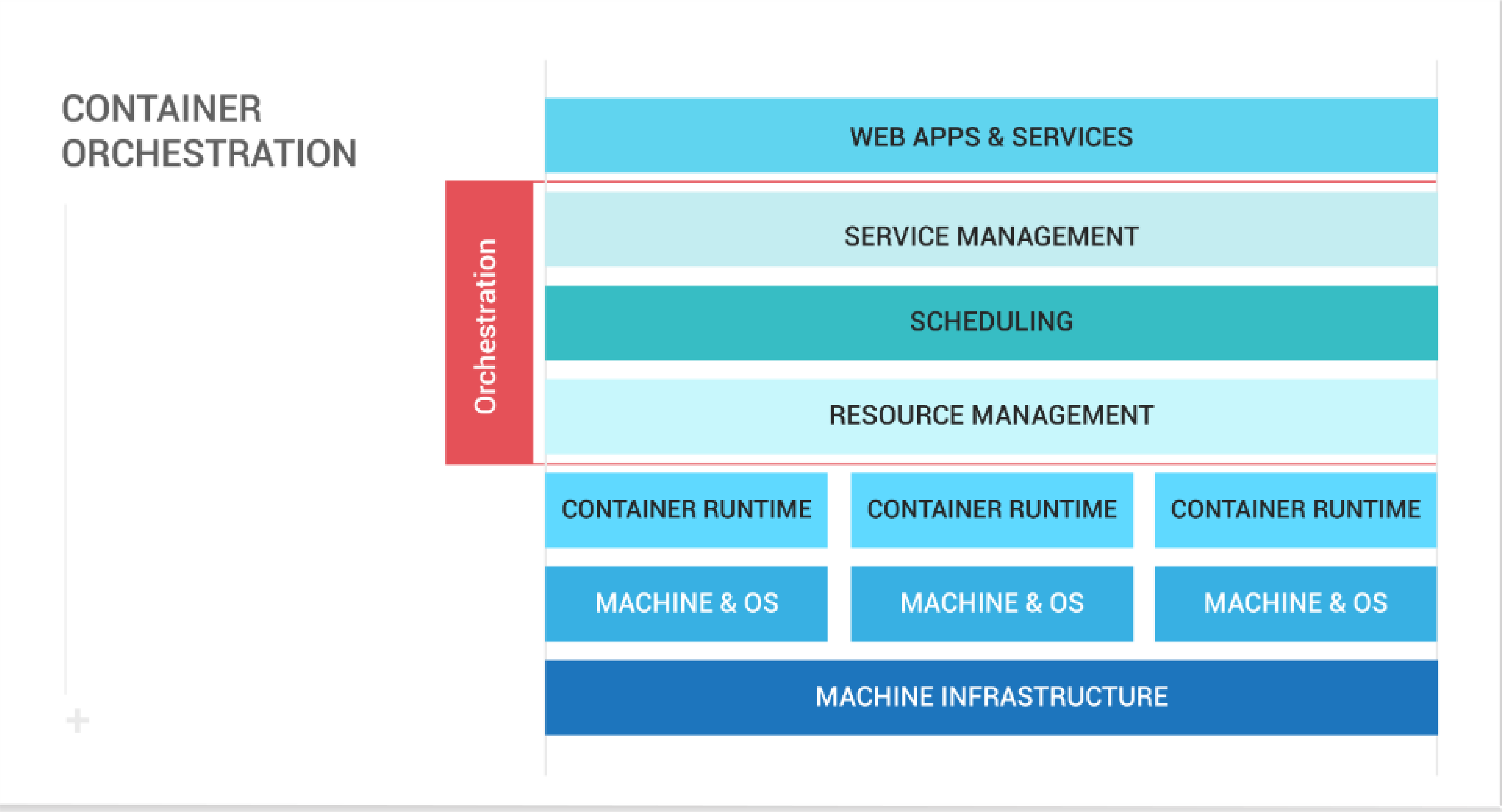
- Containers isolate software from its surroundings
- It reduces conflicts between teams running different software on the same infrastructure

Container Orchestration

Why Container Orchestration?

- Container orchestration manages the availability, scaling and networking of the containers
- It helps in monitoring the cluster i.e., group of hosts
- It helps in managing the timing of container creations
- It helps in container configuration in order to allow containers to communicate with one another

Container Orchestration



CONTAINER ORCHESTRATION

WEB APPS & SERVICES

Orchestration

SERVICE MANAGEMENT

SCHEDULING

RESOURCE MANAGEMENT

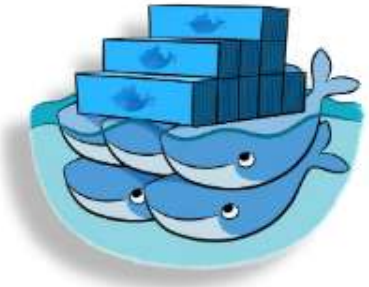
CONTAINER RUNTIME | CONTAINER RUNTIME | CONTAINER RUNTIME

MACHINE & OS | MACHINE & OS | MACHINE & OS

MACHINE INFRASTRUCTURE

The Top 3 Conatiner Orchestrators

The Top 3 Container Orchestrators are :

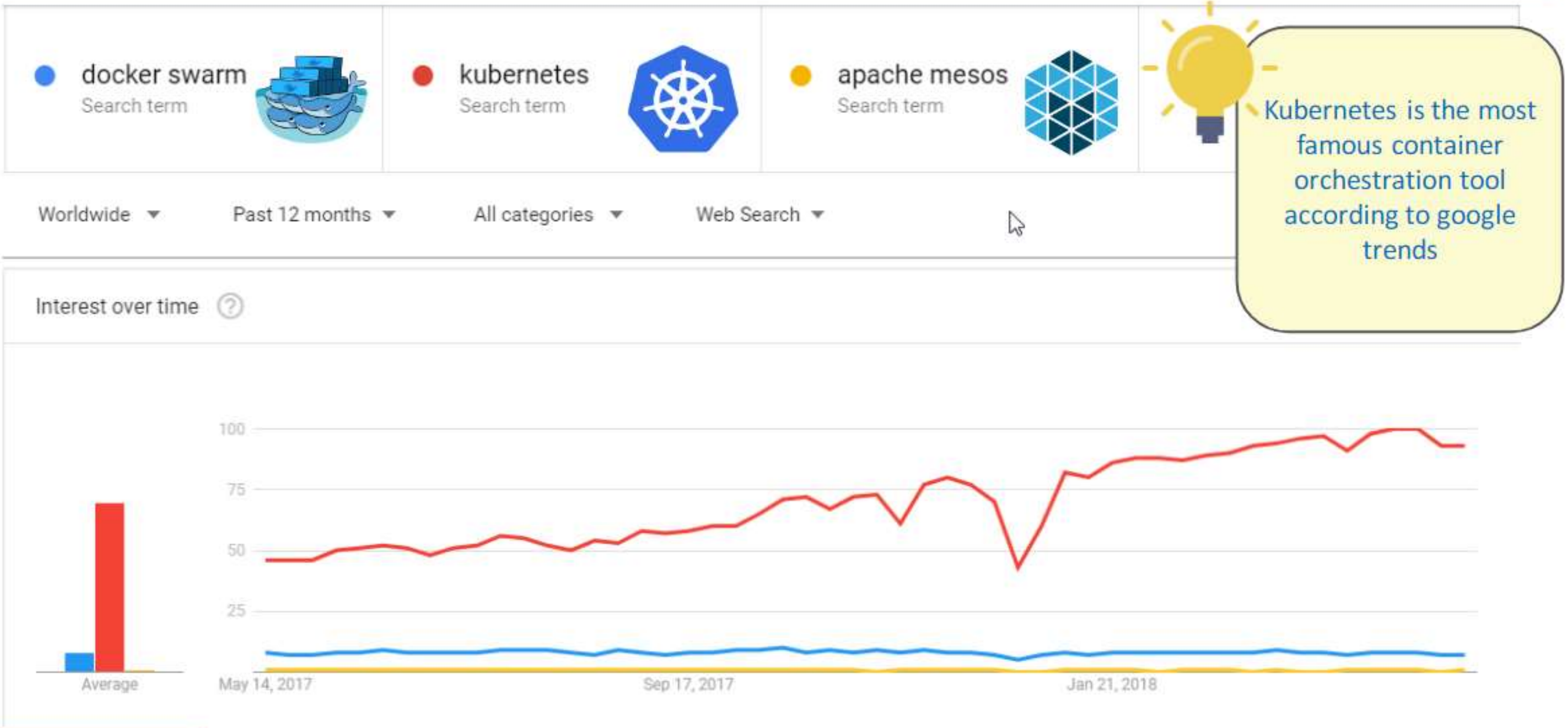


Docker Swarm

Mesos

Kubernetes

The Top 3 Conatiner Orchestrators



● docker swarm
Search term

● kubernetes
Search term

● apache mesos
Search term

Worldwide ▼ Past 12 months ▼ All categories ▼ Web Search ▼

Interest over time

Kubernetes is the most famous container orchestration tool according to google trends

Docker Swarm Vs Kubernetes

Docker Swarm

- Services are discoverable easily through the whole network in Docker Swarm
- It can easily run with other docker tools
- Local volume can be shared easily
- It provides quick container deployment as well as scaling even in very large clusters

Kubernetes

- Containers can be defined as services which makes them easily discoverable in Kubernetes
- It can easily run on any Operating System
- Volume is shared within the pods
- It provides strong guarantees at the expense of speed to cluster states

What is Kubernetes?

It's an open source orchestration system which is used for:

- Deployment of containerized application
- Scaling of containerized application
- Management of containerized application

Kubernetes enables to:

- Run multiple containers on a single machine
- Schedule containers on cluster of machines
- Run log running services such as web applications

Features Of Kubernetes

Kubernetes Features

01 Automatic bin-packing

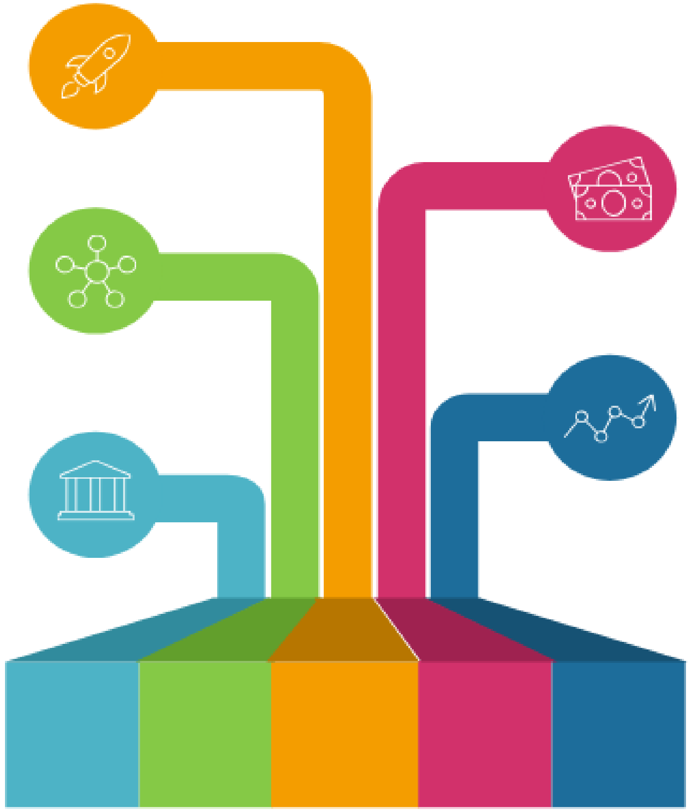
Automatically places containers based on their resource requirements and other constraints, while not sacrificing the availability.

03 Self-Healing

Automatically restarts containers that fail, replaces and reschedules containers when nodes die, kills containers that don't respond.

05 Storage-Orchestration

Automatically mount the storage system of your choice, whether from local storage or a public cloud provider such as GCP, AWS.



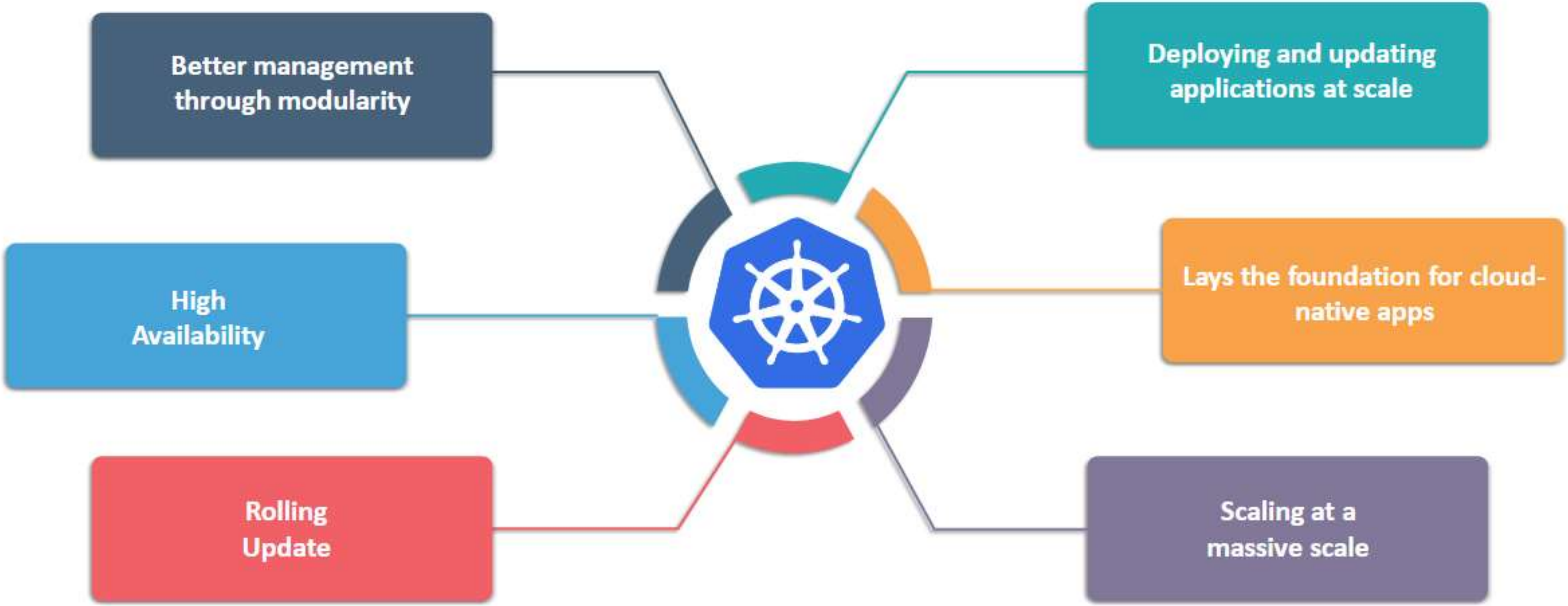
02 Horizontal Scaling

Scale your application up and down with a simple command, with a UI, or automatically based on CPU usage.

04 Batch Execution

In addition to services, Kubernetes can manage your batch and CI workloads as well as replacing containers that fail, if needed.

Advantages Of Kubernetes



Better management through modularity

High Availability

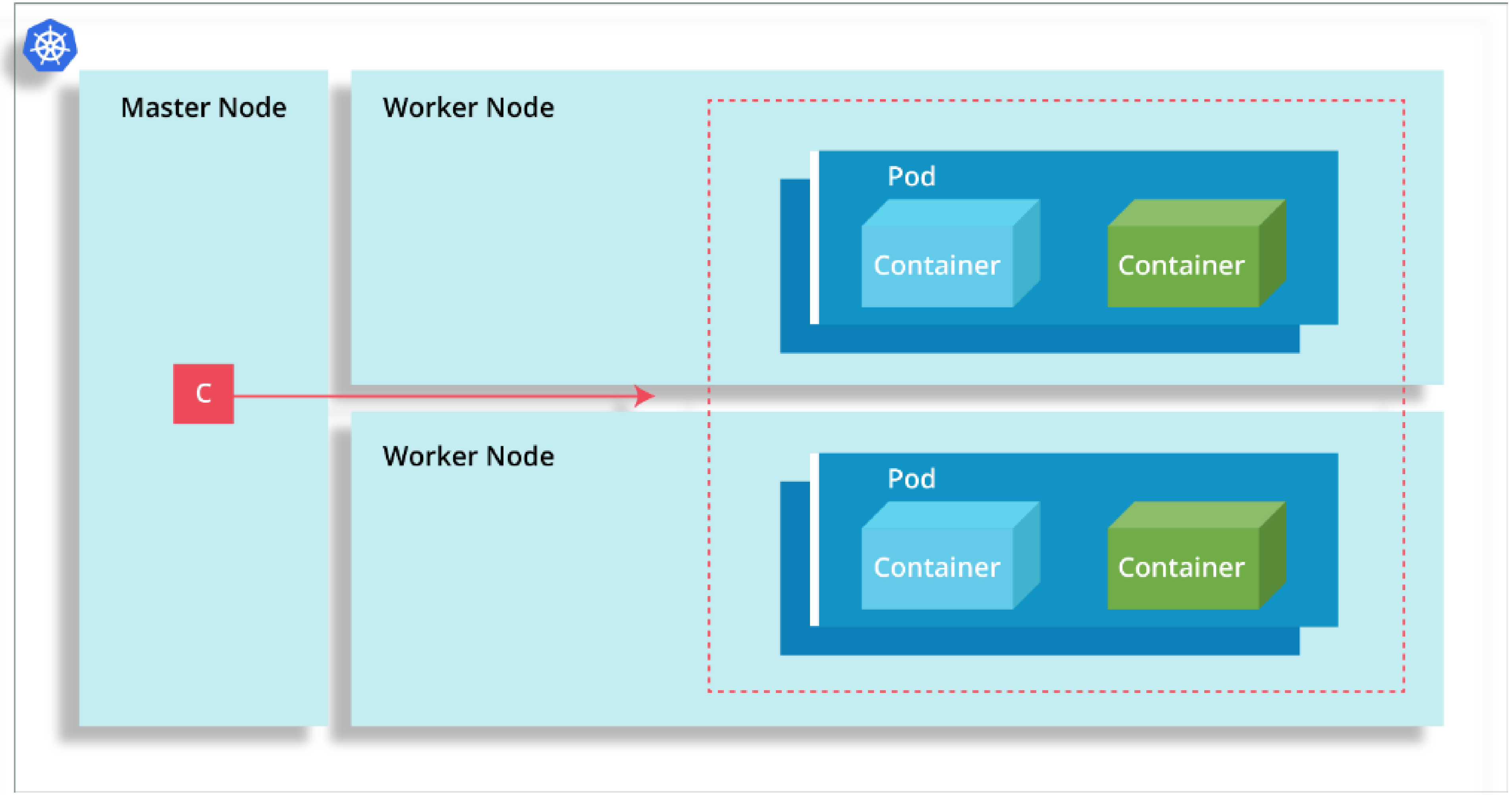
Rolling Update

Deploying and updating applications at scale

Lays the foundation for cloud-native apps

Scaling at a massive scale

Kubernetes Cluster Architecture



Kubernetes Cluster Components

Master Node : Cluster master manages the Kubernetes API server, resource controller and scheduler. It’s lifecycle is managed by Kubernetes engine when starting

Worker Node : Worker node previously known as minions may be physical machine or a virtual machine based on the cluster

Pods : Pods are group of containers which are tightly coupled together. This is done, when the containers are dependent on each other.

Let’s talk about each of these in detail:

Kubernetes Master

Master

Worker Node

Pods

Kubernetes master is a collection of three processes:

- **Kube-apiserver** : It validates and configures all the data for the API objects which include pods, services, replica controllers, and others
- **Kube-controller-manager** : It's a daemon that includes the non terminating loops (that regulates the state of the system) shipped with Kubernetes
- **Etcd :** It is a distributed key-value store designed to reliably and quickly preserve and provide access to critical data
- **Kube-scheduler** : The Kubernetes scheduler is a workload-specific function that significantly impacts availability, performance, and capacity. Workload-specific requirements will be exposed through the API as required

Kubernetes Worker Node

Master

Worker Node

Pods

Worker node in the cluster runs two processes:

- **Kubelet** : It's a foremost node agent running on each node works under the terms of PodSec. A PodSpec is a YAML or JSON object that describes a pod
- **Kube-Proxy** : Kubenetes network proxy runs on each node
- **Container Runtime** : The container runtime is the software that is responsible for running containers. Kubernetes supports several runtimes: Docker, rkt, runc and any OCI runtime-spec implementation.

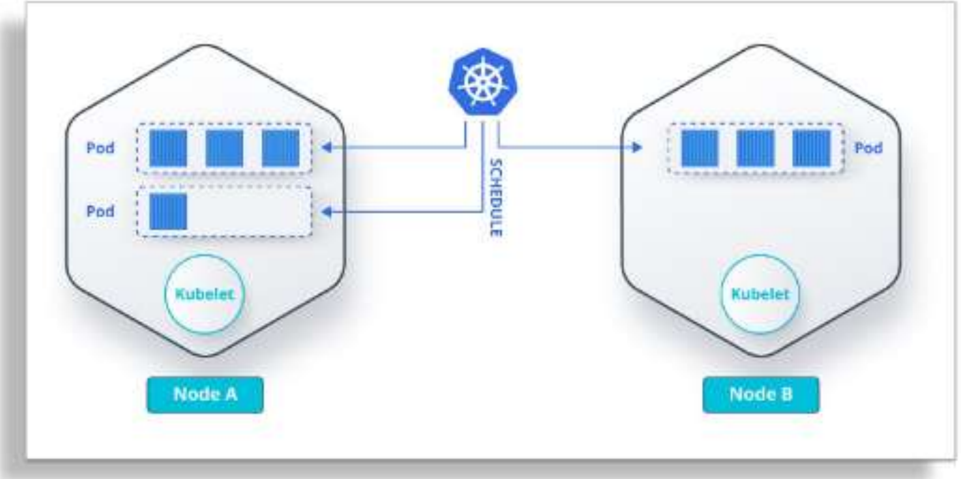
Kubernetes Pods

Master

Worker Node

Pods

- Containers are deployed and scheduled through Kubernetes in a group called pods
- These are tightly coupled containers i.e the applications running on them are dependent on each other
- 1 to 5 tightly coupled containers can be stored in a pod that collaborate to provide a service



Setting up the Kubernetes Cluster

Enough of theory! Let us start with setting up a Kubernetes Cluster!

Demo 1 - Setting up the Kubernetes Cluster

Setting up the Kubernetes Cluster

- Ensure, you have docker installed on your system. You require to do some additional configuration steps before proceeding, please refer the installation document for kubernetes provided to you in the LMS for the same.
- Install Kubeadm
- Once Kubeadm is installed initialize the master
- Install a Pod Network Configuration, for this demo we will install calico
- Install Kubernetes Dashboard once each pod is up and running
- Once each pod is running join the node to the master.
- Your Kubernetes Cluster is ready!

Exploring your Cluster

Exploring your Cluster

- You installed your cluster using the kubeadm (Kube Administration) command. Now, to explore your cluster you will be using the kubectl command(Kube Control).
- But, before using kubectl command you need to run a set of commands as a normal user to be able to access the cluster.

```
-->mkdir -p $HOME/.kube
-->sudo cp -i /etc/kubernetes/admin.conf
$HOME/.kube/config
-->sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Kubectl Introduction

- It is a Kubernetes command-line tool which is used to deploy and manage applications on Kubernetes

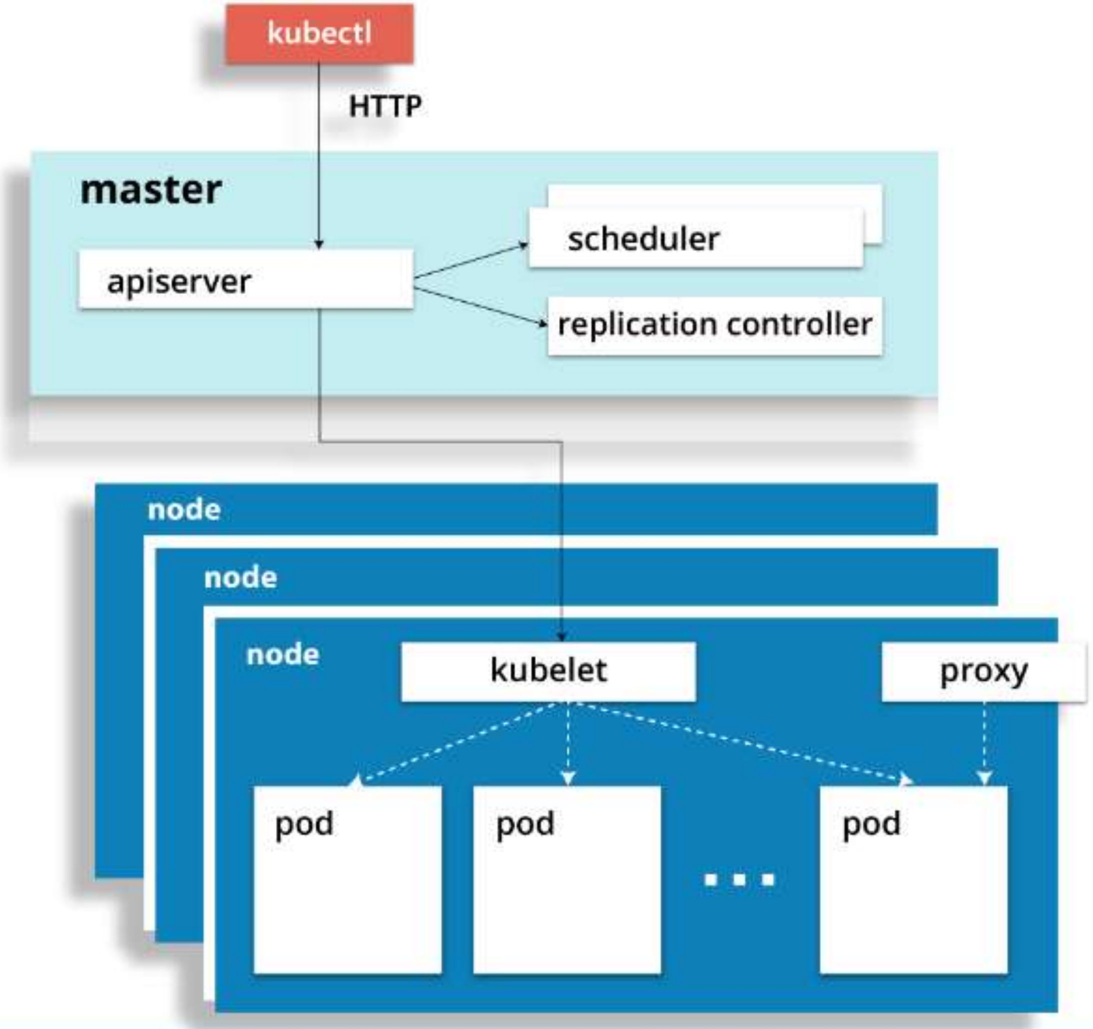
- It helps in inspecting the Kubernetes cluster resources

- By using Kubectl, we can create , delete, and update components on Kubernetes cluster

Some Kubectl Commands

- kubectl version : It displays both the client and the sever version
- kubectl get nodes : It will display the nodes available in the cluster
- kubectl run : It will create a new deployment
- kubectl get deployments : It will show the list of deployment
- Kubectl proxy : Creates connection between host and the Kubernetes cluster
- curl http://localhost:8001/version : Enables to interact directly through API

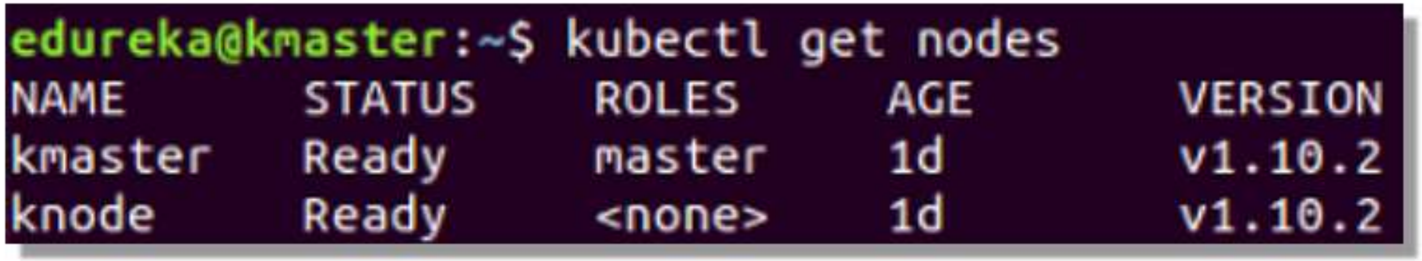
Working of kubectl



Exploring your Cluster

- To get the running master and it’s worker nodes, please run the following command.

```
-->kubectl get nodes
```

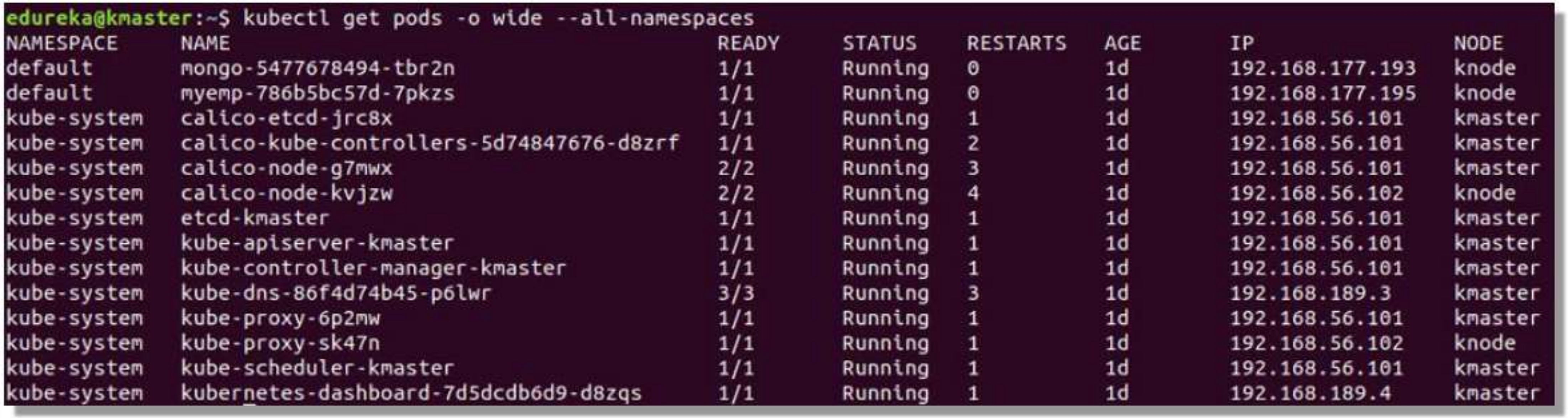


```
edureka@kmaster:~$ kubectl get nodes
NAME      STATUS    ROLES     AGE       VERSION
kmaster   Ready     master    1d        v1.10.2
knode     Ready     <none>    1d        v1.10.2
```

Exploring your Cluster

- To get the running pods on your cluster use the following command

```
-->kubectl get pods –o wide –all-namespaces
```



```
edureka@kmaster:~$ kubectl get pods -o wide --all-namespaces
NAMESPACE     NAME                                       READY     STATUS    RESTARTS   AGE       IP                NODE
default       mongo-5477678494-tbr2n                     1/1       Running   0          1d        192.168.177.193   knode
default       myemp-786b5bc57d-7pkzs                     1/1       Running   0          1d        192.168.177.195   knode
kube-system   calico-etcd-jrc8x                          1/1       Running   1          1d        192.168.56.101    kmaster
kube-system   calico-kube-controllers-5d74847676-d8zrf   1/1       Running   2          1d        192.168.56.101    kmaster
kube-system   calico-node-g7mwx                          2/2       Running   3          1d        192.168.56.101    kmaster
kube-system   calico-node-kvjzw                          2/2       Running   4          1d        192.168.56.102    knode
kube-system   etcd-kmaster                               1/1       Running   1          1d        192.168.56.101    kmaster
kube-system   kube-apiserver-kmaster                     1/1       Running   1          1d        192.168.56.101    kmaster
kube-system   kube-controller-manager-kmaster            1/1       Running   1          1d        192.168.56.101    kmaster
kube-system   kube-dns-86f4d74b45-p6lwr                  3/3       Running   3          1d        192.168.189.3     kmaster
kube-system   kube-proxy-6p2mw                           1/1       Running   1          1d        192.168.56.101    kmaster
kube-system   kube-proxy-sk47n                           1/1       Running   1          1d        192.168.56.102    knode
kube-system   kube-scheduler-kmaster                     1/1       Running   1          1d        192.168.56.101    kmaster
kube-system   kubernetes-dashboard-7d5dcdb6d9-d8zqs      1/1       Running   1          1d        192.168.189.4     kmaster
```

Deploying your first app in Kubernetes

Deploying your First App in Kubernetes

- In order to deploy your app in Kubernetes. The app should be first setup inside a container.
- To deploy the container(s) of your application on kubernetes, one needs to write a **YAML file.**
- A YAML file is a superset of a JSON file, which means that any valid JSON file is also a valid YAML file.
- Fortunately, there are only two types of structures you need to know about in YAML:
  - Maps
  - Lists

Let’s start learning YAML with Maps.

YAML has two types of structures

- **YAML Maps**
- YAML Lists

YAML Maps

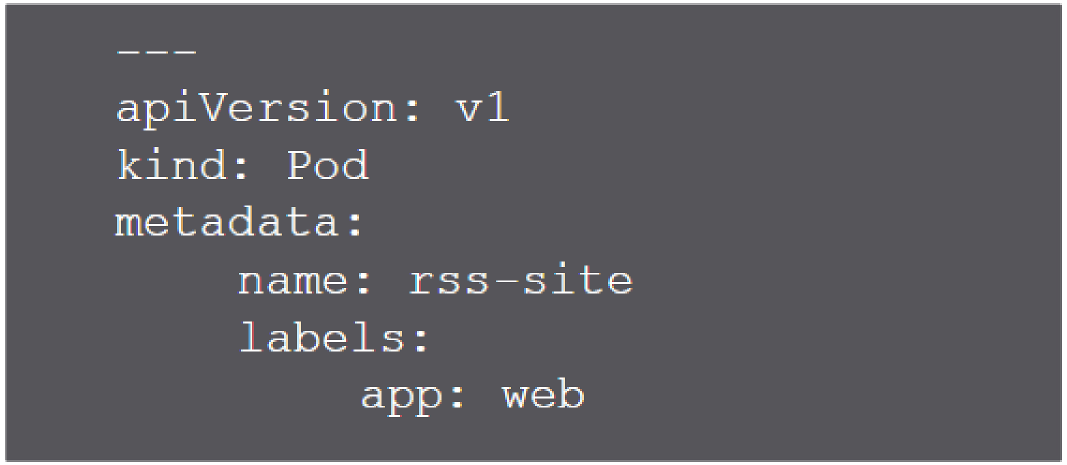
- **Maps** let you associate name-value pairs, which of course is convenient when you’re trying to set up configuration information. For example, please look at the below sample of a YAML file.

```
---
apiVersion: v1
kind: Pod
```

- The first line is a separator and is optional unless you are writing multiple structures in a single file

YAML Maps

- One can go ahead and create more **complicated structures** by creating **a key that maps to another map**



```
---
apiVersion: v1
kind: Pod
metadata:
    name: rss-site
    labels:
        app: web
```

- In this case, we have a key, metadata, that has as its value as a map with 2 more keys, name and labels. The labels key itself has a map as its value. You can nest these as far as you want to.

YAML has two types of structures

○ YAML Maps

○ **YAML Lists**

YAML Lists

- **YAML lists** are literally a sequence of objects. For example:

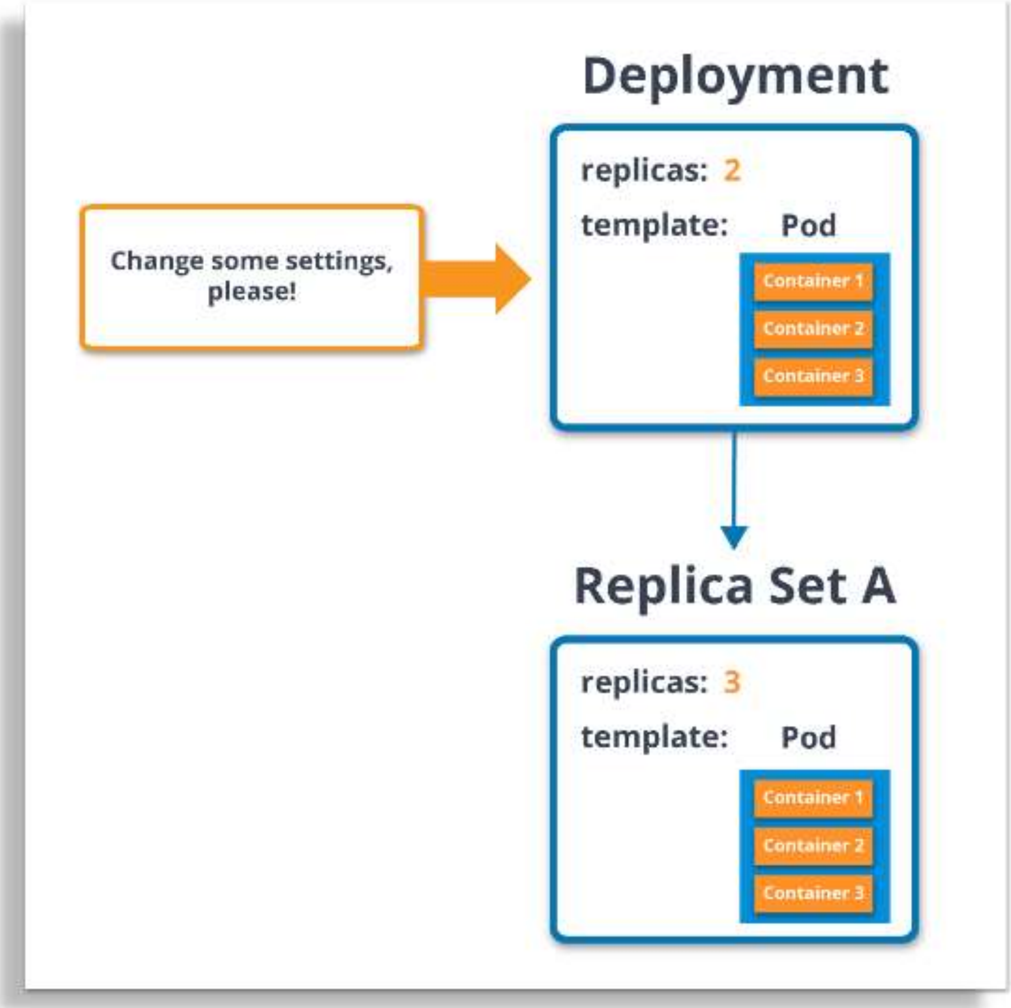
```
args
     – sleep
     – “1000”
     – message
     – “Bring back the firefly”
```

- As you can see here, you can have virtually any number of items in a list, which is defined as items that start with a dash (-) indented from the parent.

Deployments in Kubernetes

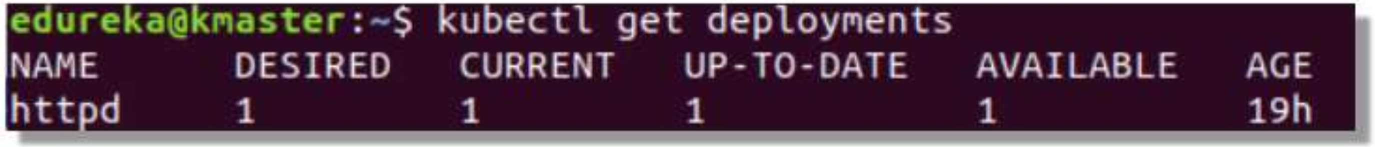
Deployment

- Deployment can be defined to create new replicasets
- It can also be defined to remove the existing deployment and use all their resources with new Deployments
- Selector field defines how the pods management sequence is determined by deployment



Various Fields In Deployment

kubectl get deployments will display the output like this :

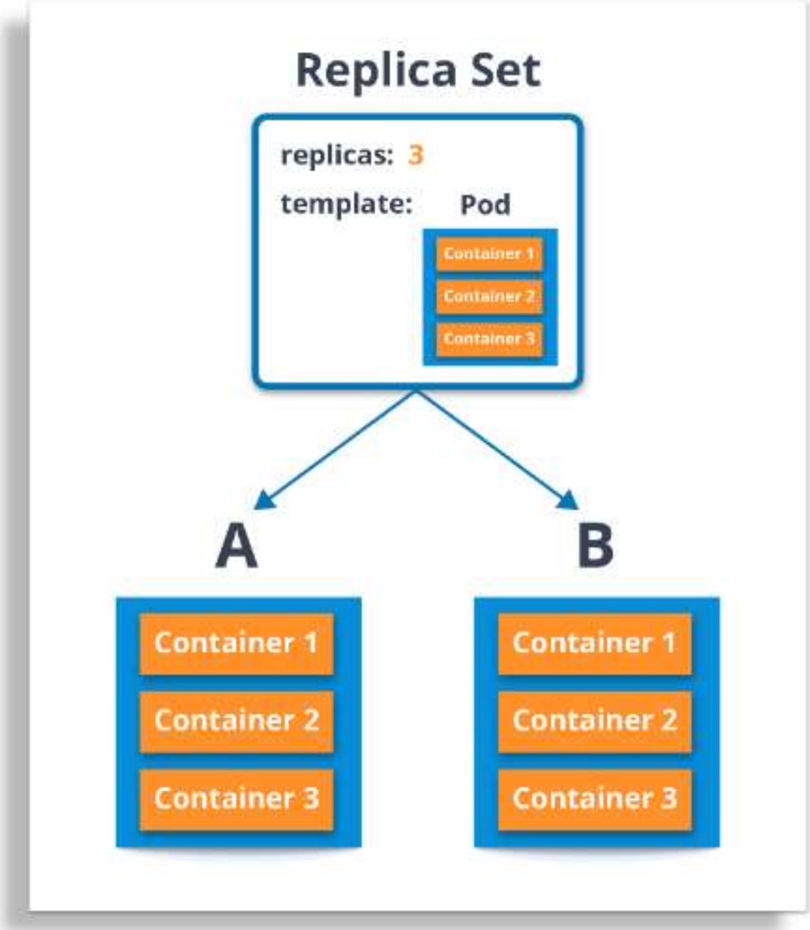


```
edureka@kmaster:~$ kubectl get deployments
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
httpd     1         1         1            1           19h
```

- **NAME:** It will lists the deployments available in the cluster
- **DESIRED:** It will display the number of replicas of the application which is defined while creating the deployment
- **CURRENT:** It will list the currently running replicas in the cluster
- **UP-TO-DATE:** It will show the required number of replicas to achieve the desired state
- **AVAILABLE:** It will display the number of replicas of application available for users
- **AGE:** It displays the amount of time since the application has been running

ReplicaSet

- A ReplicaSet makes sure that stated number of pod replicas are running at any instant of time
- It can be scaled up or down by just updating the **.spec.replicas** field
- ReplicaSet controller ensures that a desired number of pods with a matching label selector are available and operational

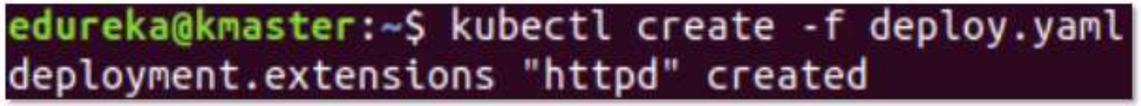


Deploying an App in Kubernetes using YAML

Following is the YAML file for deploying an apache pod with 3 replicas:

```
apiVersion: extensions/v1beta1
  kind: Deployment
  metadata:
    name: httpd
  spec:
    replicas: 3
    template:
      metadata:
        labels:
          app: httpd
       spec:
        containers:
          - name: front-end
            image: httpd
            ports:
              - containerPort: 80
```

deploy.yaml

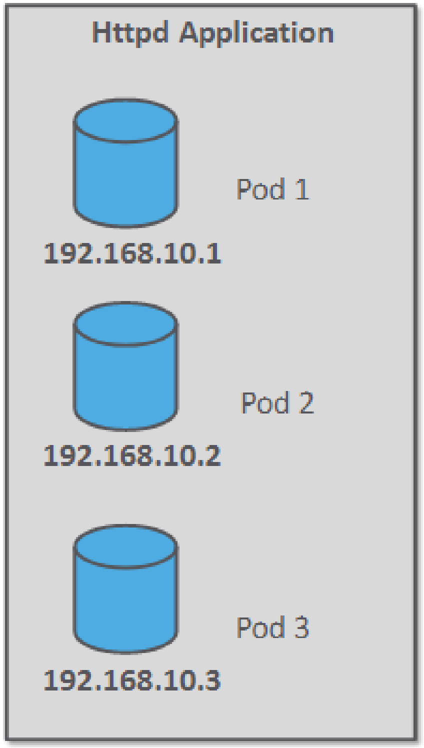


```
edureka@kmaster:~$ kubectl create -f deploy.yaml
deployment.extensions "httpd" created
```

Deploying an App in Kubernetes using YAML

Now you have deployed the pods to your cluster. But how to access these pods?

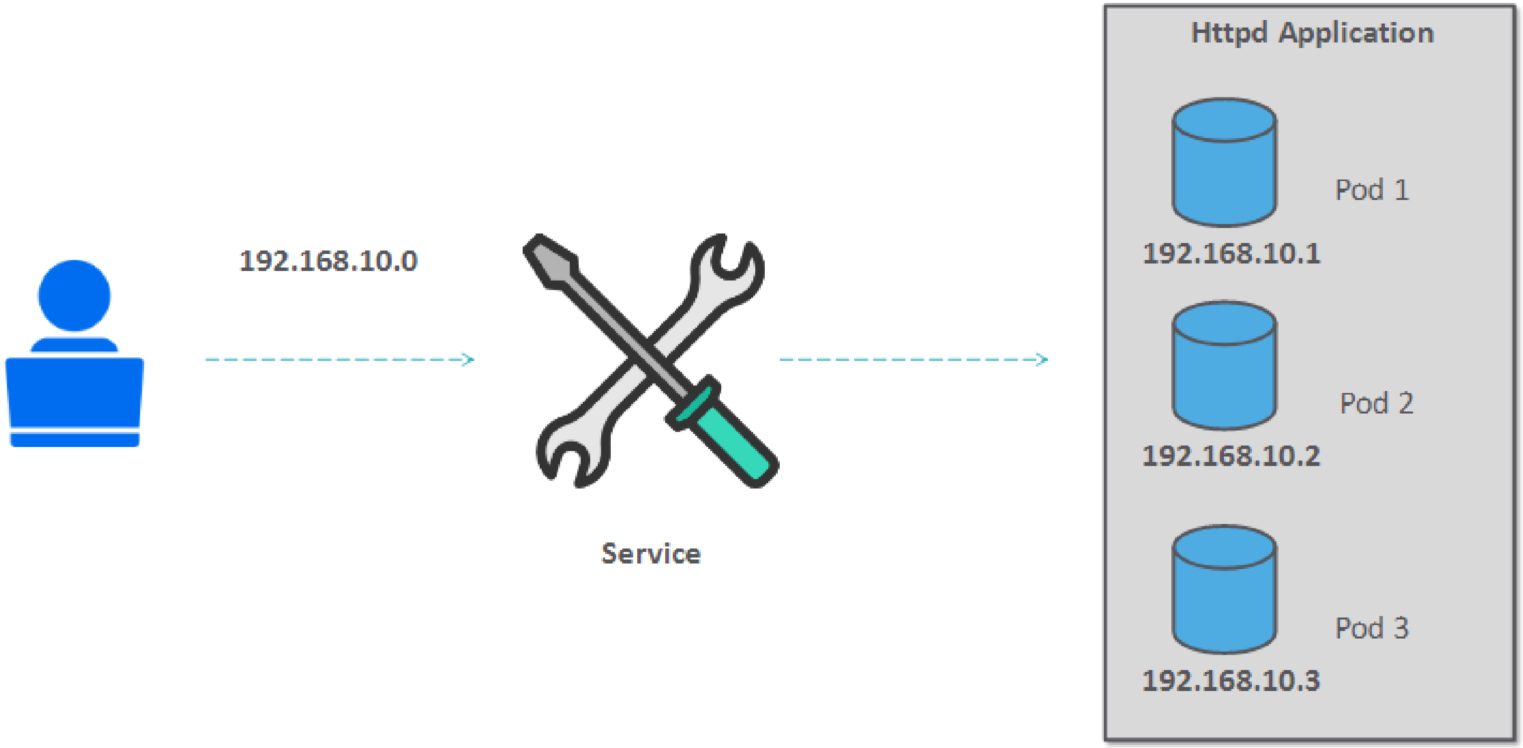
All these **pods are running the same application**, but their IP addresses are different. How would you know which IP address to connect to?



Services in Kubernetes

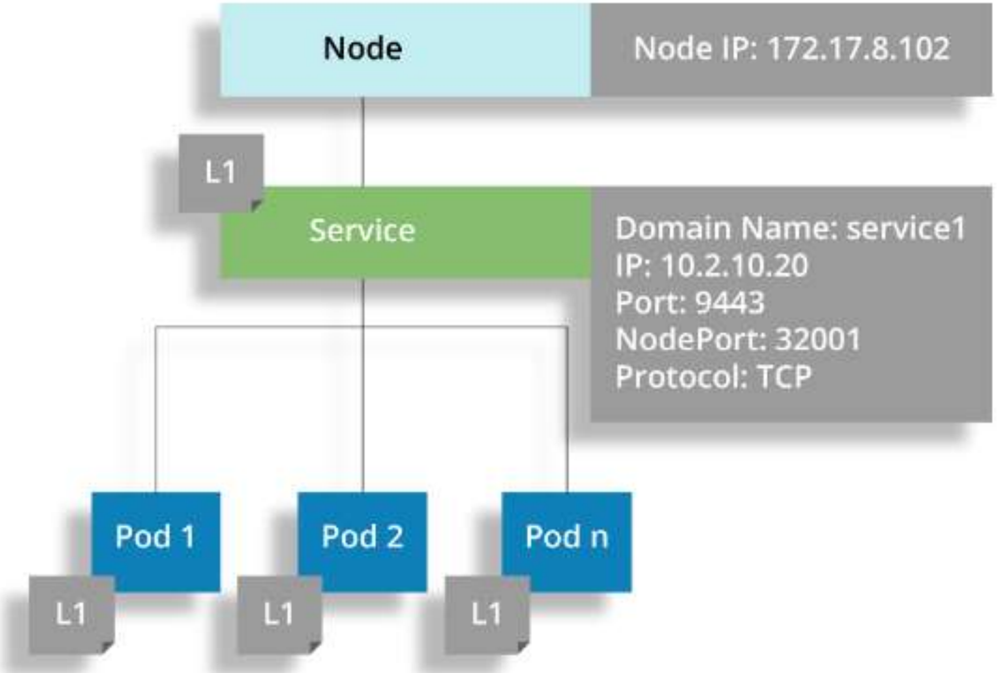
Services in Kubernetes

Services act like load balancers in Kubernetes, they also have an IP address. This IP address automatically routes to a healthy pod. In case, the pod becomes unhealthy the service automatically routes to next healthy pod. Hence, with this the user will interact with only one IP address.



Services in Kubernetes

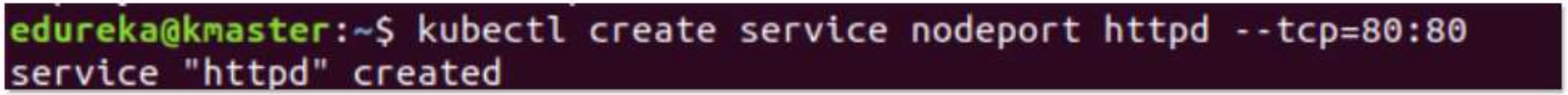
- Services defines logical set of pods and the policy through which they will be accessed
- They are the abstraction and sometimes called as micro-services
- Label Selectors determine the set of pods to be targeted by Services



Demo 2: Accessing your Application through the Service

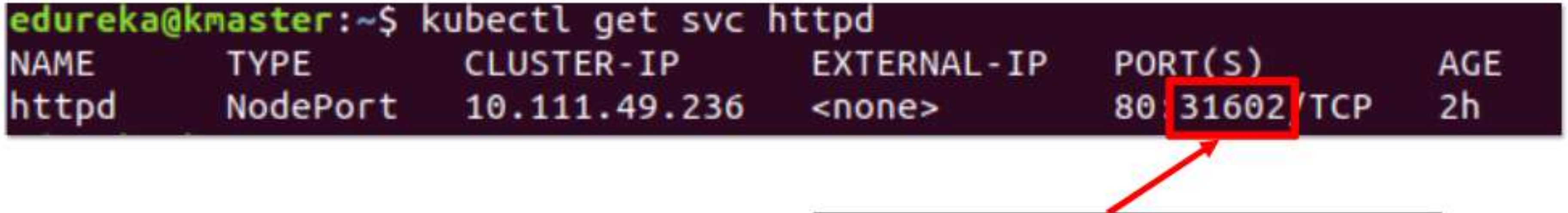
Deploy a Service in Kubernetes

To create a service use the following command:



```
edureka@kmaster:~$ kubectl create service nodeport httpd --tcp=80:80
service "httpd" created
```

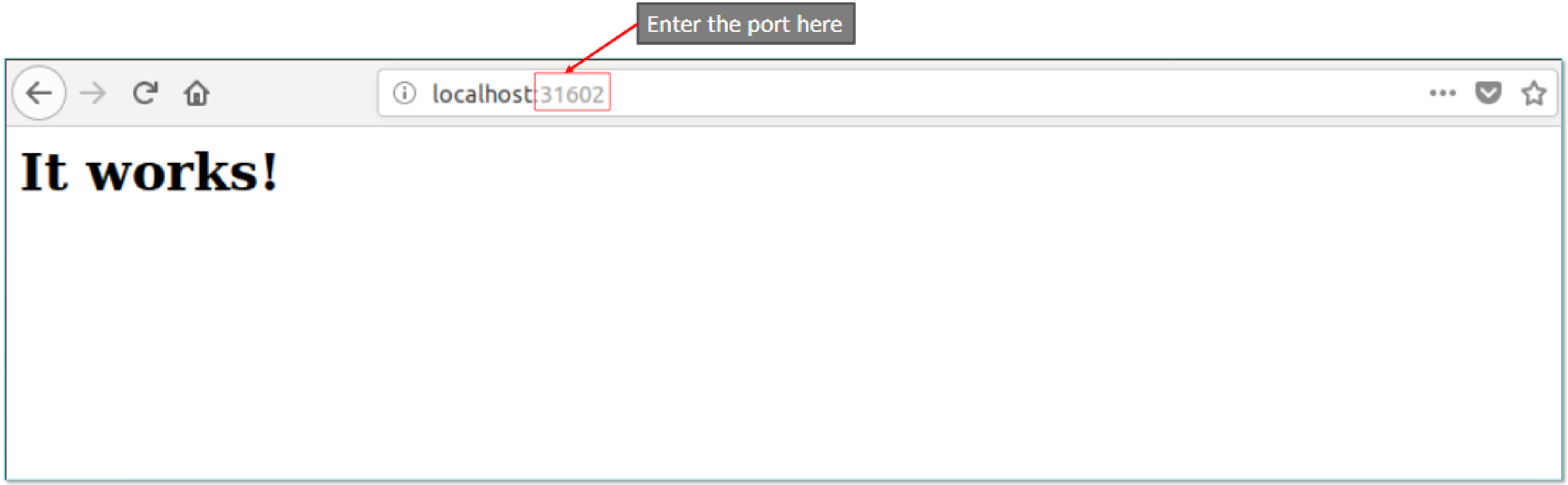
To get the port on which the service is running, run the following command:



```
edureka@kmaster:~$ kubectl get svc httpd
NAME      TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
httpd     NodePort   10.111.49.236   <none>        80:31602/TCP   2h
```

This is where your service is running

Accessing your Application Through Browser



Enter the port here

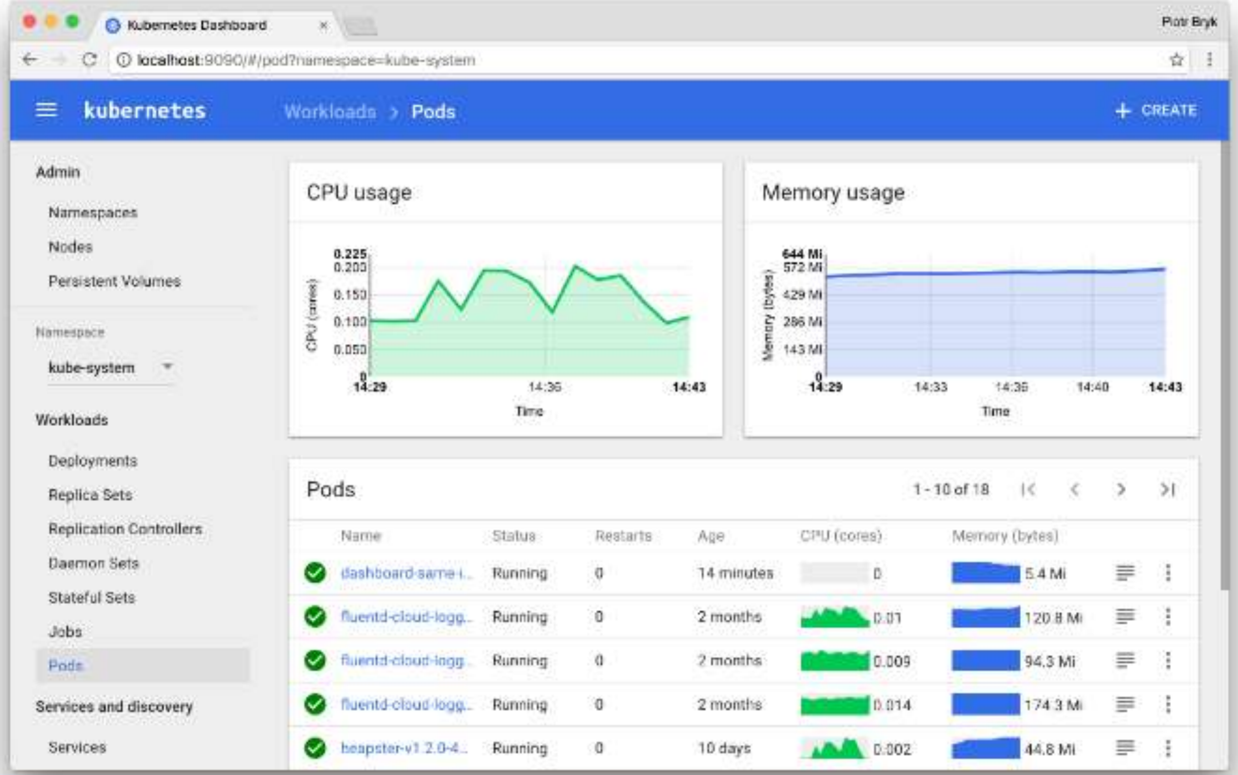
localhost:31602

It works!

Kubernetes Dashboard

Kubernetes Dashboard

Dashboard is a web-based Kubernetes user interface. You can use Dashboard to deploy containerized applications to a Kubernetes cluster, troubleshoot your containerized application, and manage the cluster itself along with its attendant resources.

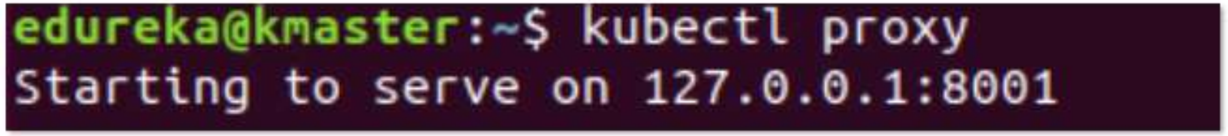


	Name	Status	Restarts	Age	CPU (cores)	Memory (bytes)
✓	dashboard-same-i...	Running	0	14 minutes	0	5.4 Mi
✓	fluentd-cloud-logg...	Running	0	2 months	0.01	120.8 Mi
✓	fluentd-cloud-logg...	Running	0	2 months	0.009	94.3 Mi
✓	fluentd-cloud-logg...	Running	0	2 months	0.014	174.3 Mi
✓	heapster-v1.2.0-4...	Running	0	10 days	0.002	44.8 Mi

Accessing the Dashboard

For accessing the dashboard, pass the following command in the terminal:

```
kubectl proxy
```



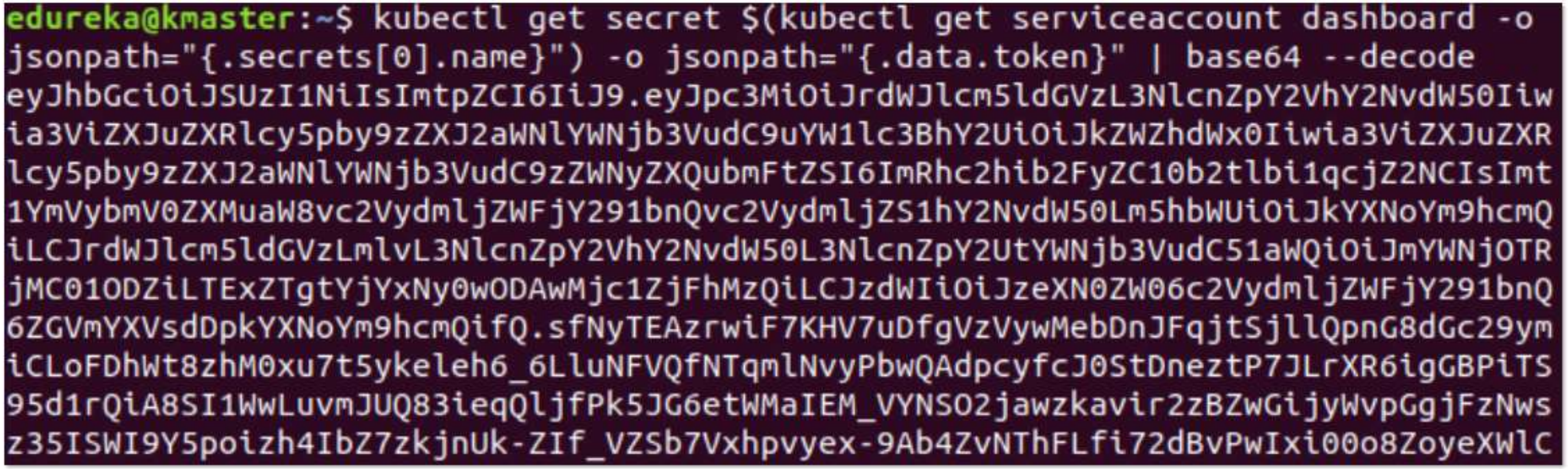
```
edureka@kmaster:~$ kubectl proxy
Starting to serve on 127.0.0.1:8001
```

On a new terminal, run the following commands:

```
// To create a service account for your dashboard
kubectl create serviceaccount dashboard -n default
// To add clustering binding rules for your roles on dashboard
kubectl create clusterrolebinding dashboard-admin -n default \
--clusterrole=cluster-admin \
--serviceaccount=default:dashboard
// To get the secret for your dashboard "token". Copy the output of this command
kubectl get secret $(kubectl get serviceaccount dashboard -o jsonpath="{.secrets[0].name}") -
o jsonpath="{.data.token}" | base64 --decode
```

Accessing the Dashboard

The output will look something like this, copy this.

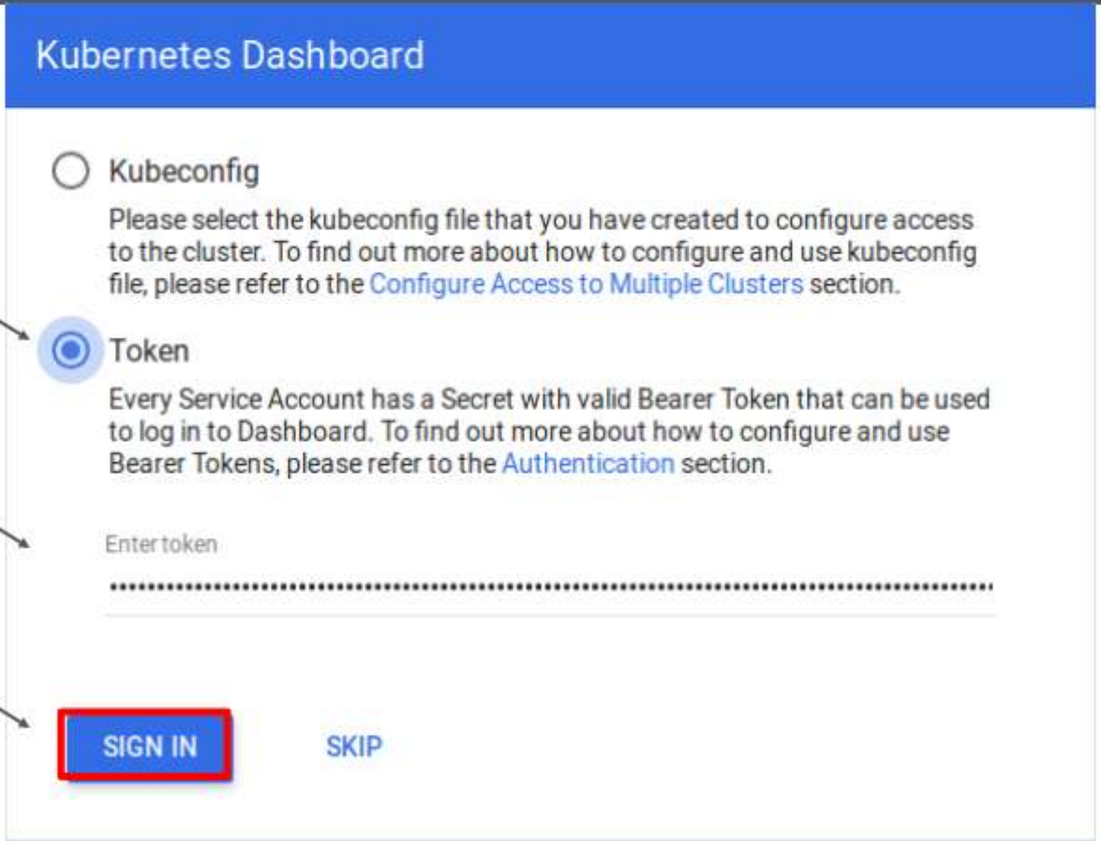


```
edureka@kmaster:~$ kubectl get secret $(kubectl get serviceaccount dashboard -o
jsonpath="{.secrets[0].name}") -o jsonpath="{.data.token}" | base64 --decode
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiw
ia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJkZWZhdWx0Iiwia3ViZXJuZXR
lcy5pby9zZXJ2aWNlYWNjb3VudC9zZWNyZXQubmFtZSI6ImRhc2hib2FyZC10b2tlbi1qcjZ2NCIsImt
1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50Lm5hbWUiOiJkYXNoYm9hcmQ
iLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiJmYWNjOTR
jMC01ODZiLTExZTgtYjYxNy0wODAwMjc1ZjFhMzQiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ
6ZGVmYXVsdDpkYXNoYm9hcmQifQ.sfNyTEAzrwiF7KHV7uDfgVzVywMebDnJFqjtSjllQpnG8dGc29ym
iCLoFDhWt8zhM0xu7t5ykeleh6_6LluNFVQfNTqmlNvyPbwQAdpcyfcJ0StDneztP7JLrXR6igGBPiTS
95d1rQiA8SI1WwLuvmJUQ83ieqQljfPk5JG6etWMaIEM_VYNSO2jawzkavir2zBZwGijyWvpGgjFzNws
z35ISWI9Y5poizh4IbZ7zkjnUk-ZIf_VZSb7Vxhpvyex-9Ab4ZvNThFLfi72dBvPwIxi00o8ZoyeXWlC
```

Accessing the Dashboard

And go to the following link:

```
http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/
```



○ Kubeconfig

Please select the kubeconfig file that you have created to configure access to the cluster. To find out more about how to configure and use kubeconfig file, please refer to the Configure Access to Multiple Clusters section.

● Token

Every Service Account has a Secret with valid Bearer Token that can be used to log in to Dashboard. To find out more about how to configure and use Bearer Tokens, please refer to the Authentication section.

Enter token

••

Select the token option

Enter the "token" here

Finally, click on Sign in

Demo-3: Deploying an App through Kubernetes Dashboard

Demo-3: Deploying an App through Dashboard

1. The ***httpd*** image deployed in the previous demo, we will deploy it using the dashboard in this demo.
2. Open the dashboard and click on create.
3. Select Create an App and enter the relevant information
4. Go to services and click on view/update YAML for the new deployed service
5. In the “type”, change from ***LoadBalancer*** to ***NodePort***

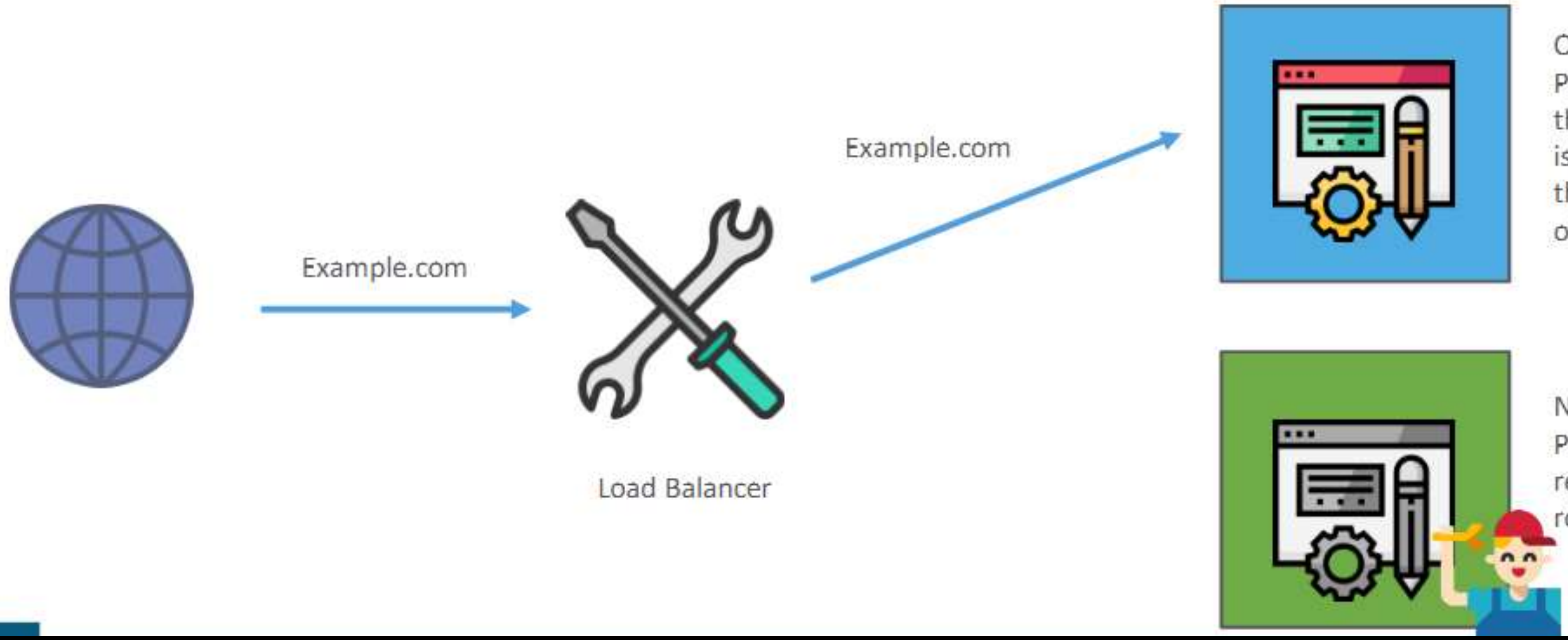
Rolling Updates In Kubernetes

Rolling Updates in Kubernetes

- A rolling update is the process of updating an application — whether it is a new version or just updated configuration — in a serial fashion. By updating one instance at a time, you are able to keep the application up and running.
- Rolling update in Kubernetes follows the Blue Green Deployment Model.

Blue Green Deployment Model

- Blue-green deployment is a technique that reduces downtime and risk by running two identical production environments called Blue and Green.
- The updated application gets setup in the new environment(Green), while the old application remains in it's own environment(Blue) untouched. The traffic stays with the blue environment until the green environment is ready.



Example.com

Load Balancer

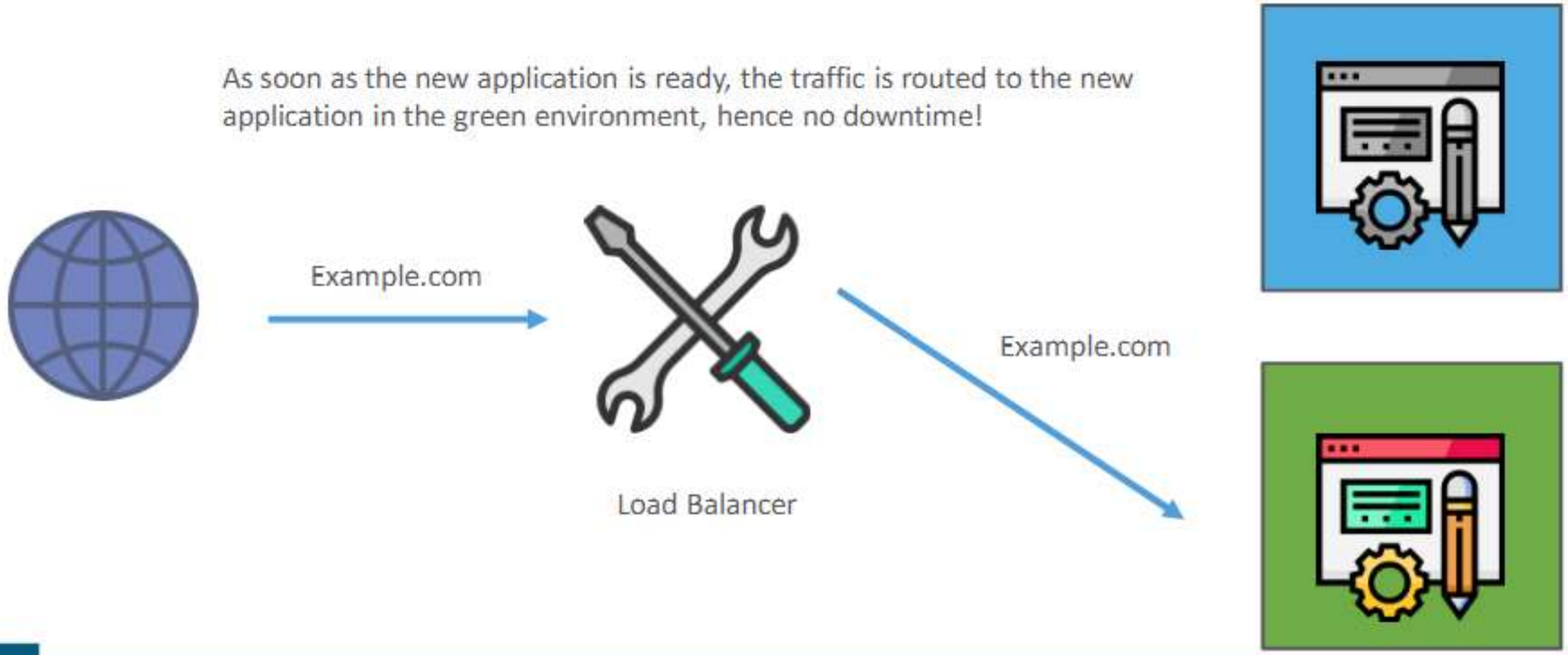
Example.com

Old Application in Production. While the new application is getting ready, all the traffic routes to old environment.

New Application in Production getting ready. No Traffic is routed here.

Blue Green Deployment Model

- Blue-green deployment is a technique that reduces downtime and risk by running two identical production environments called Blue and Green.
- The updated application gets setup in the new environment(Green), while the old application remains in it's own environment(Blue) untouched. The traffic stays with the blue environment until the green environment is ready.



As soon as the new application is ready, the traffic is routed to the new application in the green environment, hence no downtime!

Example.com

Load Balancer

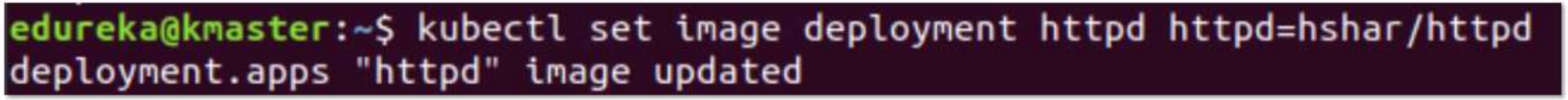
Example.com

Demo-4: Rolling Updates in Kubernetes

Following is the command for performing a rolling update on an existing deployment:

```
kubectl set image deployment <deployment> <container>=<image>
```

For example:



```
edureka@kmaster:~$ kubectl set image deployment httpd httpd=hshar/httpd
deployment.apps "httpd" image updated
```

Demo-4: Rolling Updates In Kubernetes

Demo-4: Rolling Updates in Kubernetes

1. Pull the **httpd** docker image, and edit the index.html text.
2. Commit the container and name the container with a relevant tag.
3. Push the container to DockerHub.
4. Perform a rolling update on the httpd deployment created previously with this new docker image.