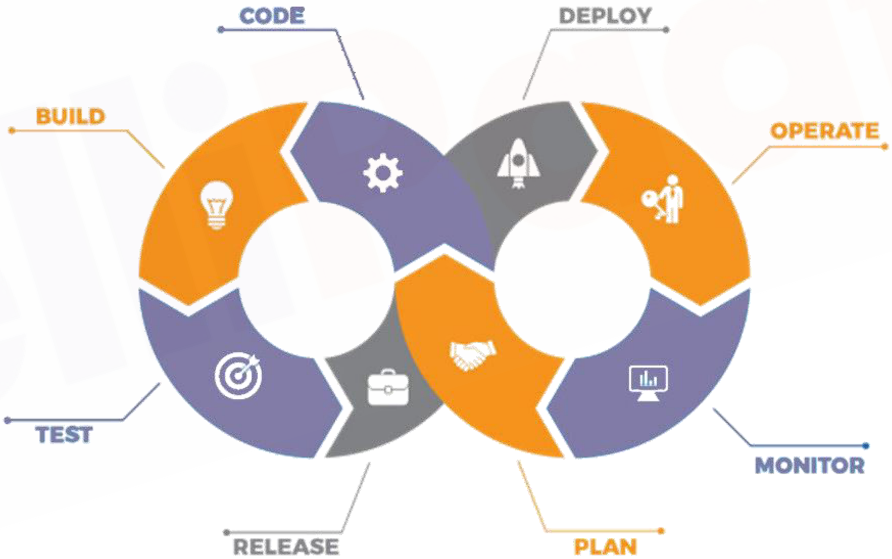
Introduction to Kubernetes



Agenda

01 Introduction to Kubernetes

02 Docker Swarm Vs. Kubernetes

03 Kubernetes Architecture

04 Kubernetes Installation

05 Working of Kubernetes

06 Deployments in Kubernetes

07 Services in Kubernetes

08 Ingress in Kubernetes

09 Kubernetes Dashboard

Introduction to Kubernetes

Introduction to Kubernetes

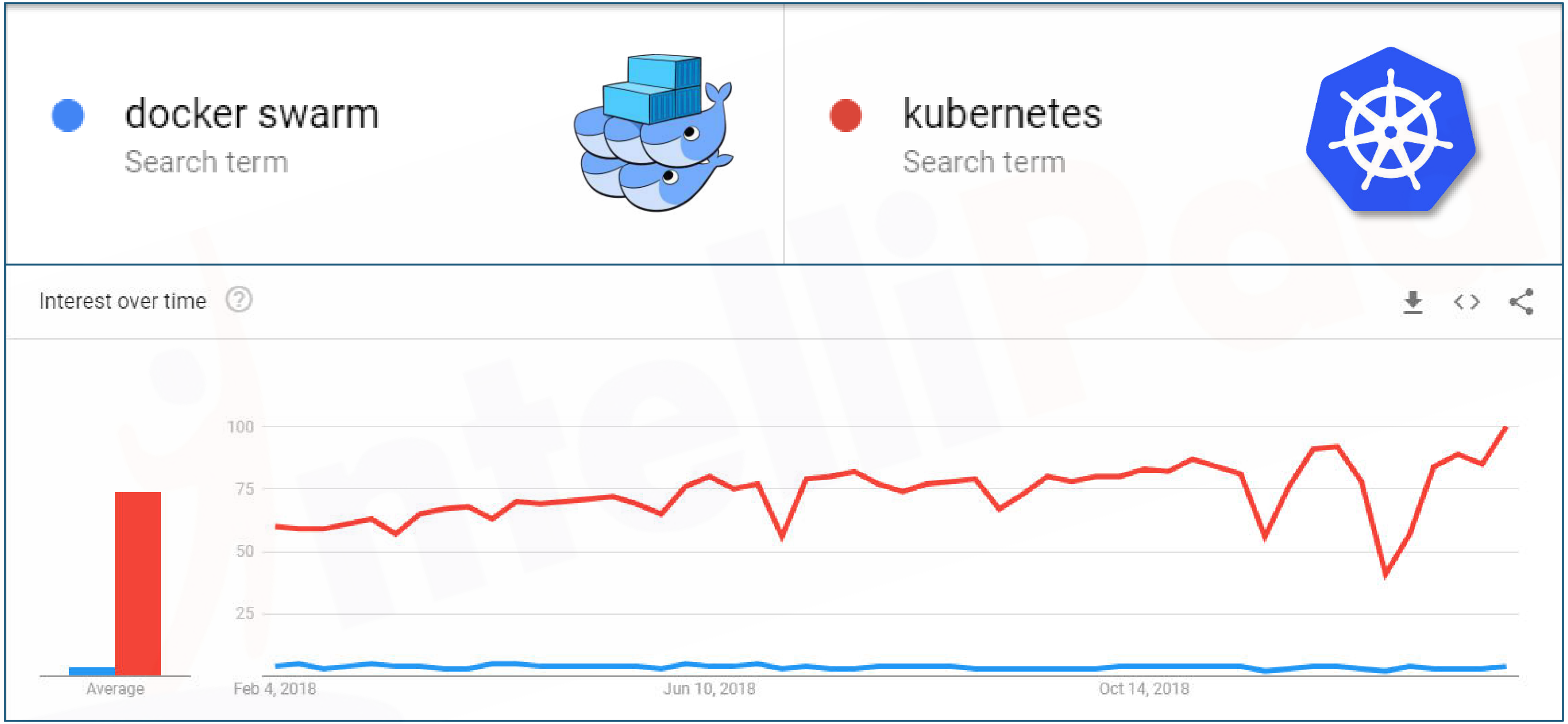
- Kubernetes is an open-source container orchestration software.
- It was originally developed by Google.
- It was first released on July 21, 2015.
- It is the ninth most active repository on GitHub in terms of number of commits.

Features of Kubernetes

- Pods
- Replication Controller
- Storage Management
- Resource Monitoring
- Health Checks
- Service Discovery
- Networking
- Secret Management
- Rolling Updates

Docker Swarm Vs. Kubernetes

Docker Swarm Vs. Kubernetes



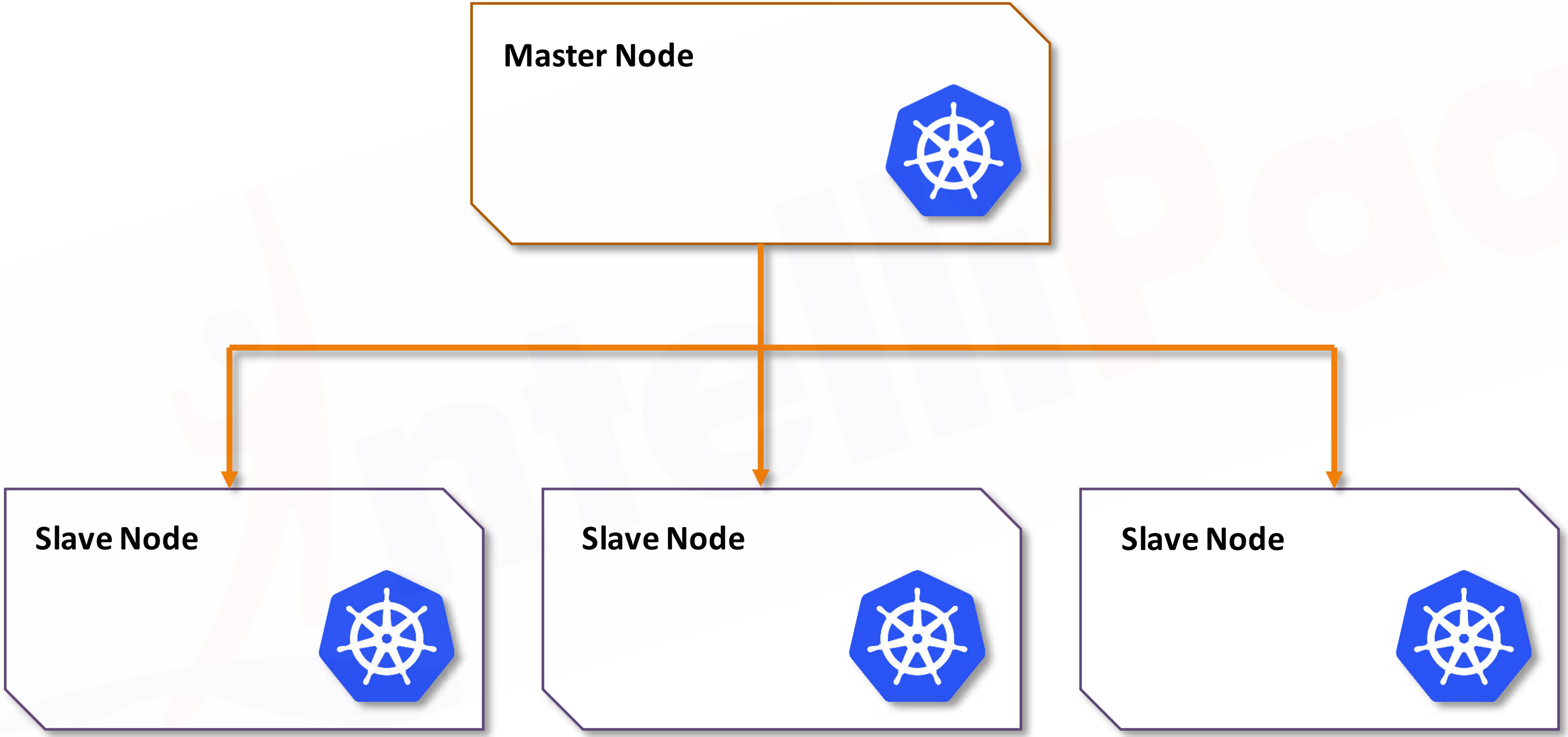
Source: trends.google.com

Docker Swarm Vs. Kubernetes

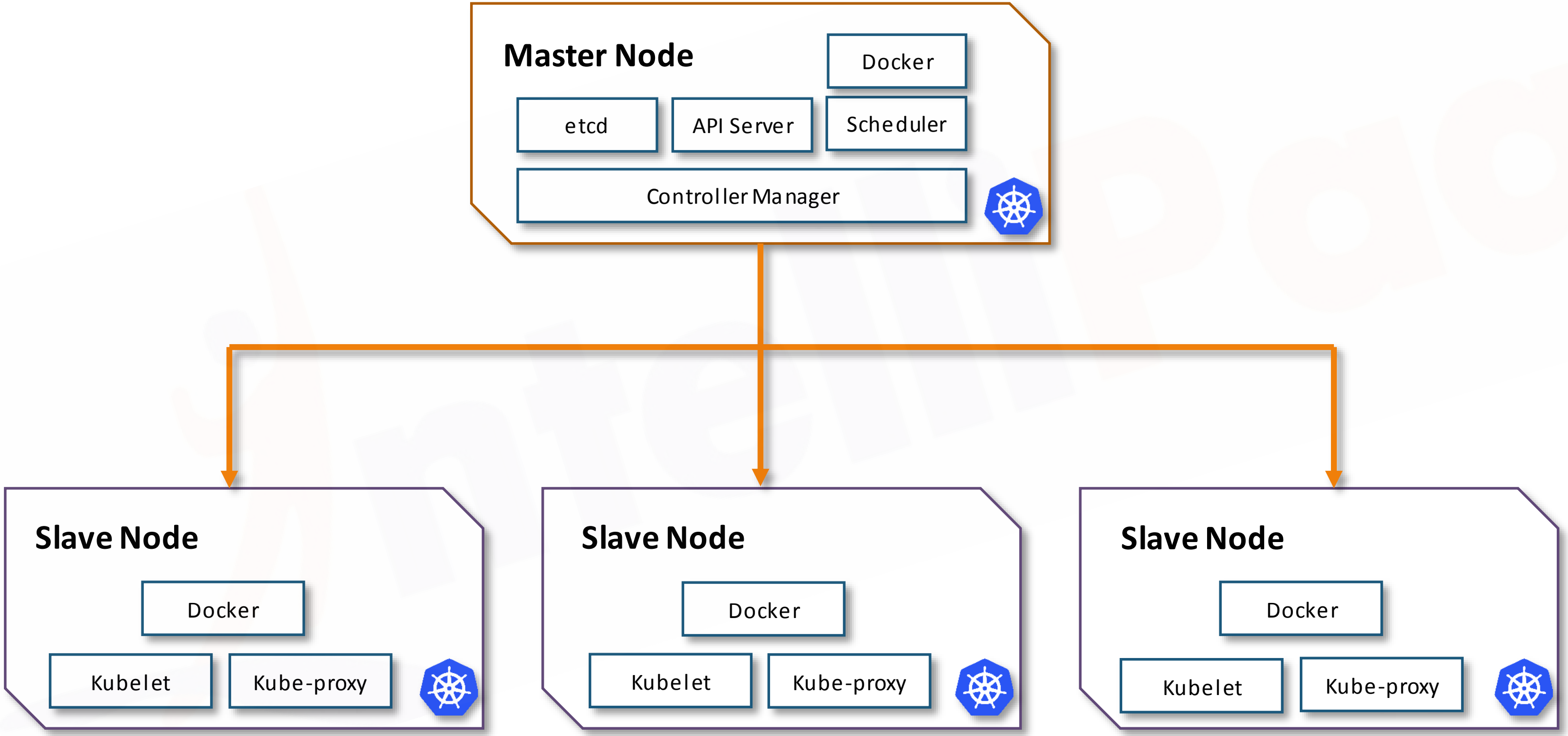
Docker Swarm	Kubernetes
Easy to install and initialize	Complex procedure to install
Faster when compared to Kubernetes	Slower when compared to Docker Swarm
Not reliable and has less features	More reliable and has more features

Kubernetes Architecture

Kubernetes Architecture



Kubernetes Architecture



Kubernetes Architecture: Master Components

Kubernetes Architecture: Master Components

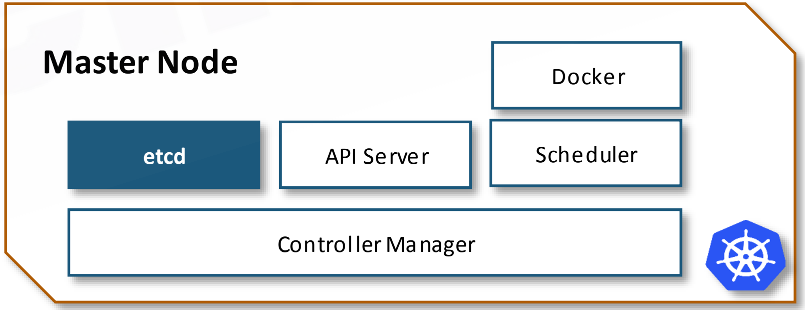
etcd

API Server

Scheduler

Controller Manager

It is a highly available distributed key–value store, which is used to store cluster wide secrets. It is only accessible by the Kubernetes API server, as it has sensitive information.



Kubernetes Architecture: Master Components

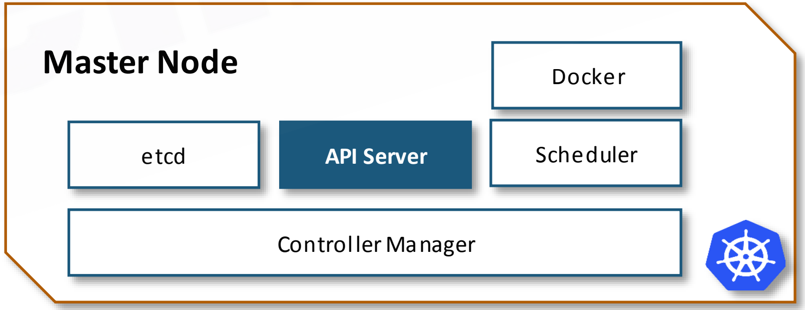
etcd

API Server

Scheduler

Controller Manager

It exposes Kubernetes API. Kubernetes API is the front-end for the Kubernetes Control Plane and is used to deploy and execute all operations in Kubernetes.



Kubernetes Architecture: Master Components

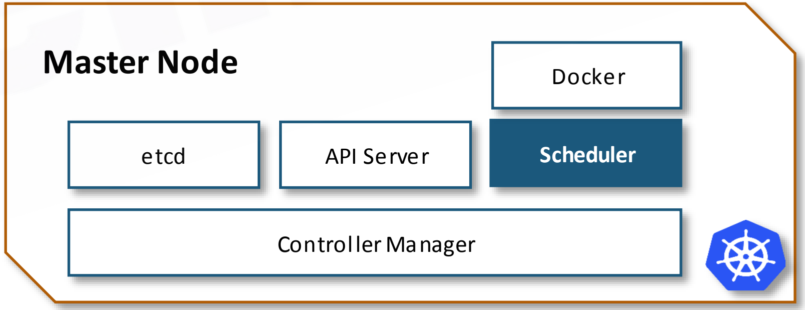
etcd

API Server

Scheduler

Controller Manager

The scheduler takes care of scheduling of all processes and the dynamic resource management and manages present and future events on the cluster.



Master Node

Docker

etcd

API Server

Scheduler

Controller Manager

Kubernetes Architecture: Master Components

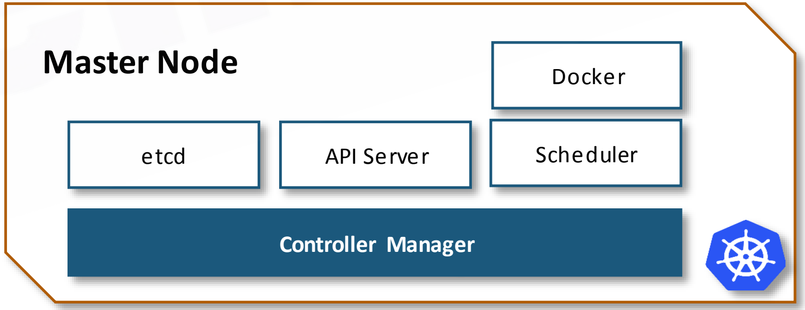
etcd

API Server

Scheduler

Controller Manager

The controller manager runs all controllers on the Kubernetes cluster. Although each controller is a separate process, to reduce complexity, all controllers are compiled into a single process. They are as follows: **Node Controller, Replication Controller, Endpoints Controller, Service Accounts and Token Controllers**.



Master Node

Docker

etcd

API Server

Scheduler

Controller Manager

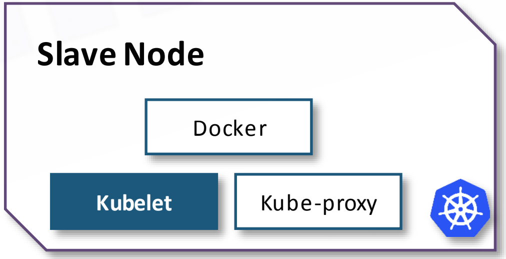
Kubernetes Architecture: Slave Components

Kubernetes Architecture: Slave Components

Kubelet

Kube-proxy

Kubelet takes the specification from the API server and ensures that the application is running according to the specifications which were mentioned. Each node has its own kubelet service.

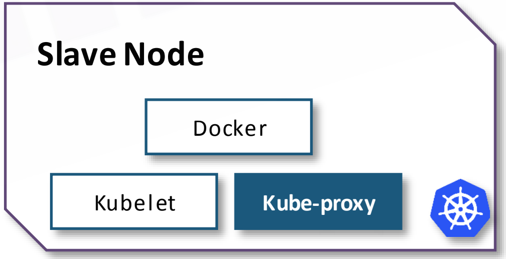


Kubernetes Architecture: Slave Components

Kubelet

Kube-proxy

This proxy service runs on each node and helps in making services available to the external host. It helps in connection forwarding to the correct resources. It is also capable of doing primitive load balancing.

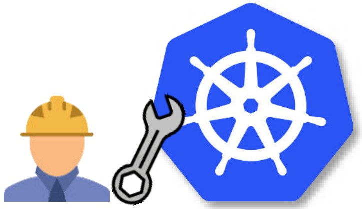


Kubernetes Installation

Kubernetes Installation

There are numerous ways to install Kubernetes. Following are some of the popular ways:

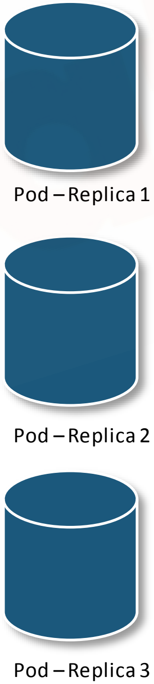
- **Kubeadm**: Bare Metal Installation
- **Minikube**: Virtualized Environment for Kubernetes
- **Kops**: Kubernetes on AWS
- **Kubernetes on GCP**: Kubernetes running on Google Cloud Platform



Hands-on: Installing Kubernetes Using Kubeadm

Working of Kubernetes

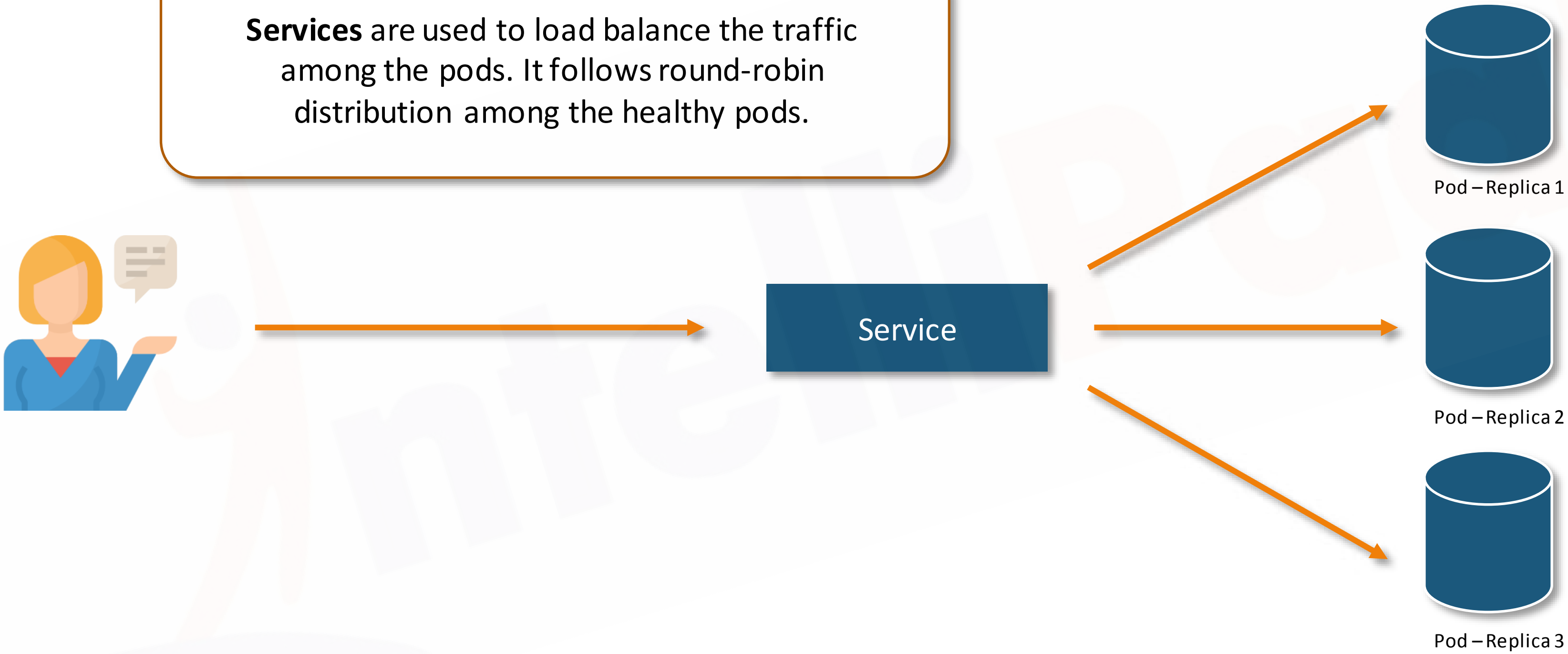
Working of Kubernetes



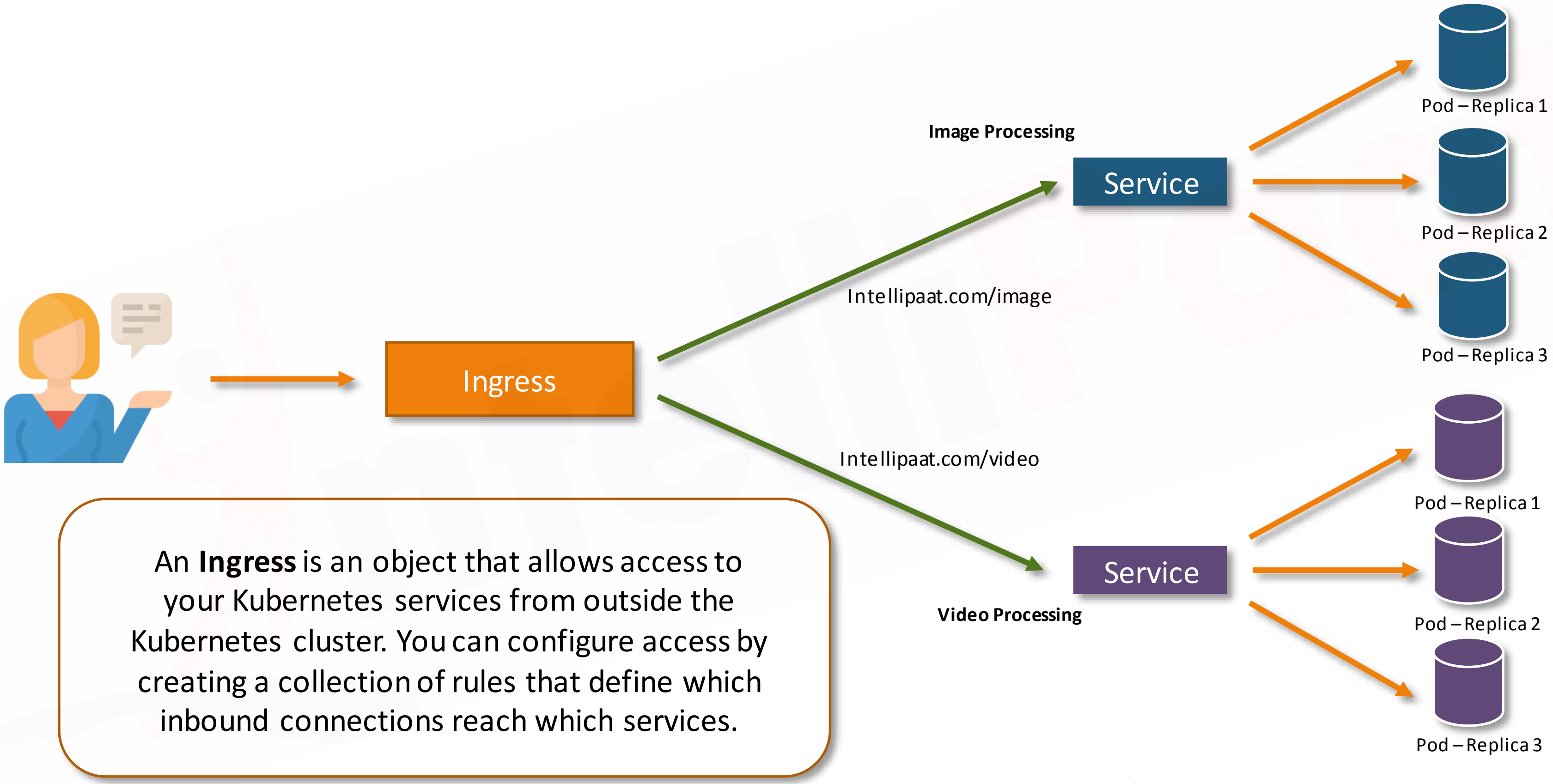
Pods can have one or more containers coupled together. They are the basic unit of Kubernetes. To increase high availability, we always prefer pods to be in replicas.

Working of Kubernetes

Services are used to load balance the traffic among the pods. It follows round-robin distribution among the healthy pods.



Working of Kubernetes

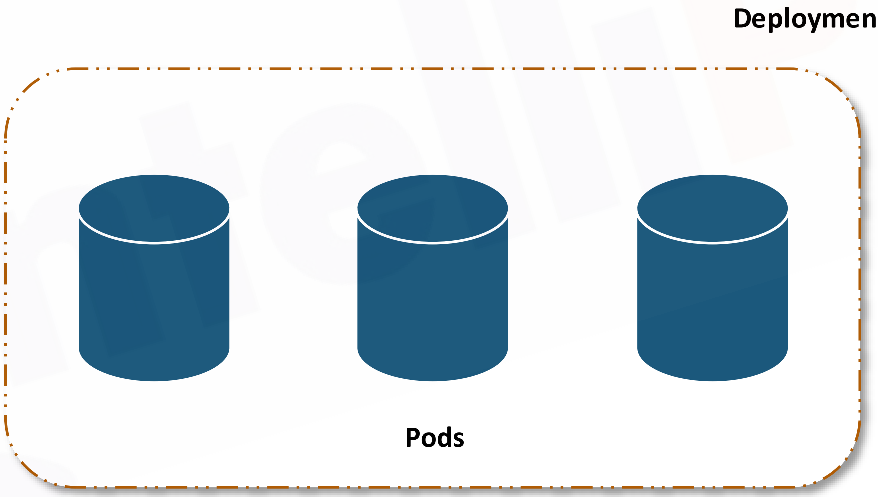


An **Ingress** is an object that allows access to your Kubernetes services from outside the Kubernetes cluster. You can configure access by creating a collection of rules that define which inbound connections reach which services.

Deployments in Kubernetes

Deployments in Kubernetes

Deployment in Kubernetes is a controller which helps your applications reach the desired state; the desired state is defined inside the deployment file.



YAML Syntax for Deployments

This YAML file will deploy 3 pods for nginx and will maintain the desired state, which is 3 pods, until this deployment is deleted.

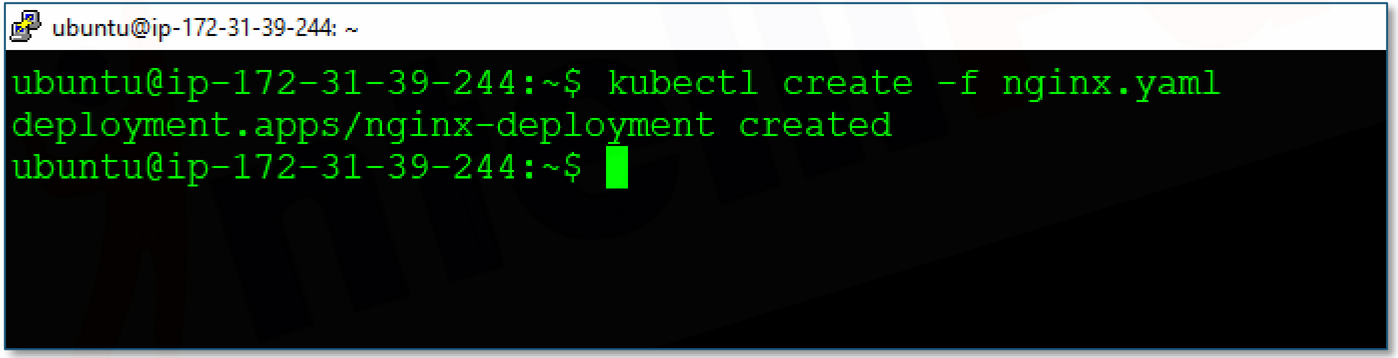
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

Creating a Deployment

Once the file is created, to deploy this deployment use the following syntax:

Syntax

kubectl create –f nginx.yaml



ubuntu@ip-172-31-39-244: ~

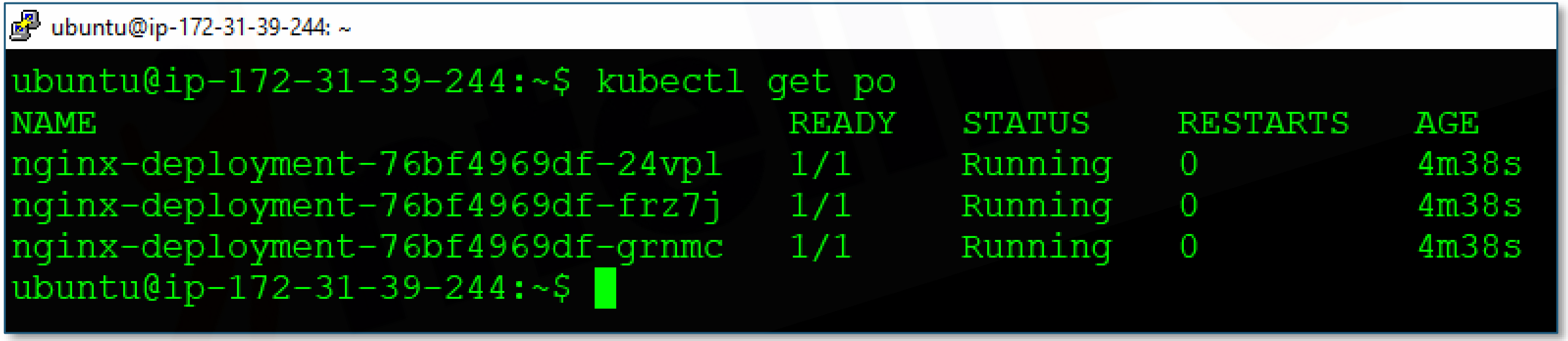
```
ubuntu@ip-172-31-39-244:~$ kubectl create -f nginx.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-39-244:~$
```

Listing the Pods

To view the pods, type the following command:

Syntax

```
kubectl get po
```



ubuntu@ip-172-31-39-244: ~

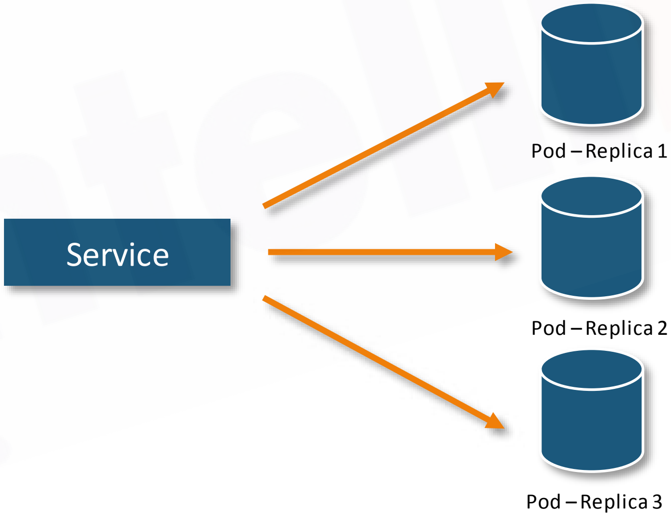
```
ubuntu@ip-172-31-39-244:~$ kubectl get po
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-76bf4969df-24vpl   1/1     Running   0          4m38s
nginx-deployment-76bf4969df-frz7j   1/1     Running   0          4m38s
nginx-deployment-76bf4969df-grnmc   1/1     Running   0          4m38s
ubuntu@ip-172-31-39-244:~$
```

As you can see, the number of pods are matching with the number of replicas specified in the deployment file.

Creating a Service

Creating a Service

A Service is basically a round-robin load balancer for all pods, which matches with its name or selector. It constantly monitors the pods; in case a pod gets unhealthy, the service will start deploying the traffic to other healthy pods.



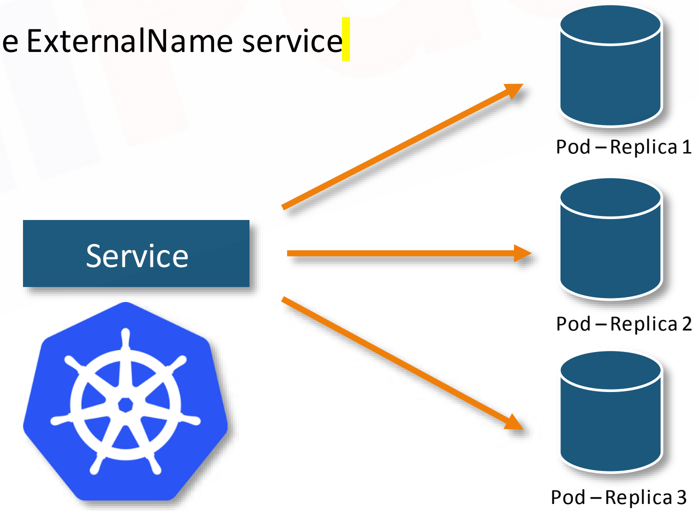
Service Types

ClusterIP: Exposes the service on cluster-internal IP

NodePort: Exposes the service on each Node's IP at a static port

LoadBalancer: Exposes the service externally using a cloud provider's load balancer

ExternalName: Maps the service to the DNS Name mentioned with the ExternalName service

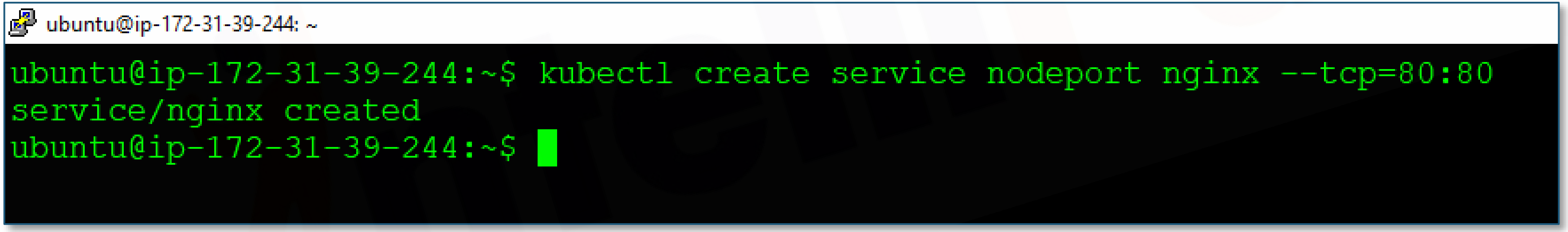


Creating a NodePort Service

We can create a NodePort service using the following syntax:

Syntax

kubectl create service nodeport <name-of-service> --tcp=<port-of-service>:<port-of-container>



ubuntu@ip-172-31-39-244: ~

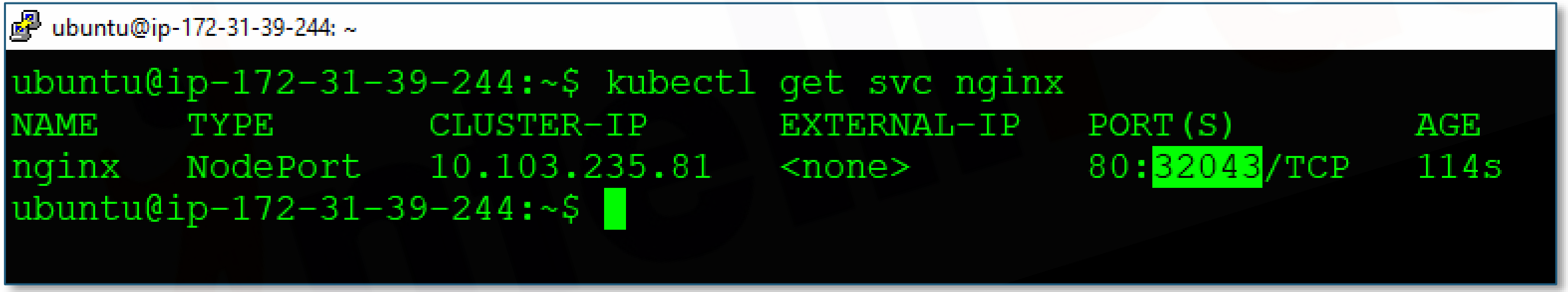
```
ubuntu@ip-172-31-39-244:~$ kubectl create service nodeport nginx --tcp=80:80
service/nginx created
ubuntu@ip-172-31-39-244:~$
```

Creating a NodePort Service

To know the port, on which the service is being exposed, type the following command:

Syntax

```
kubectl get svc nginx
```



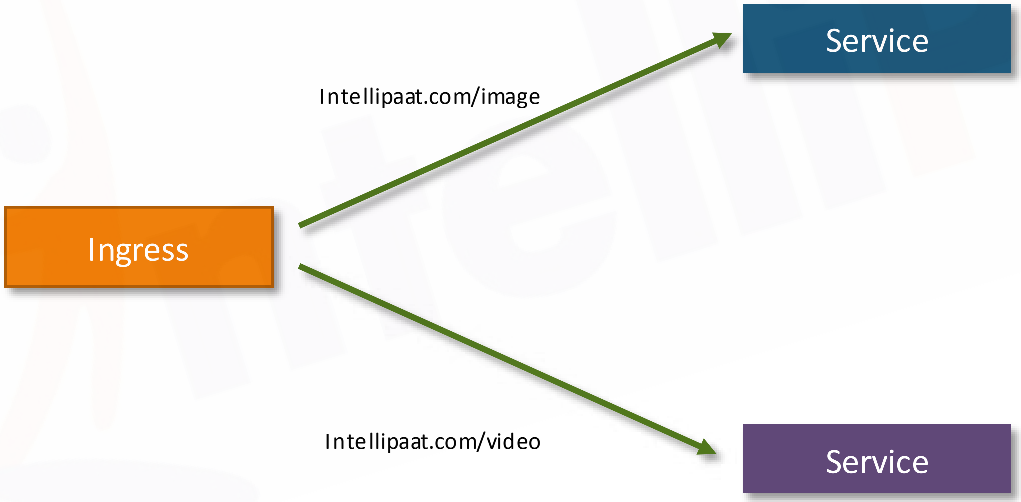
ubuntu@ip-172-31-39-244: ~

```
ubuntu@ip-172-31-39-244:~$ kubectl get svc nginx
NAME     TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
nginx    NodePort    10.103.235.81    <none>        80:32043/TCP   114s
ubuntu@ip-172-31-39-244:~$
```

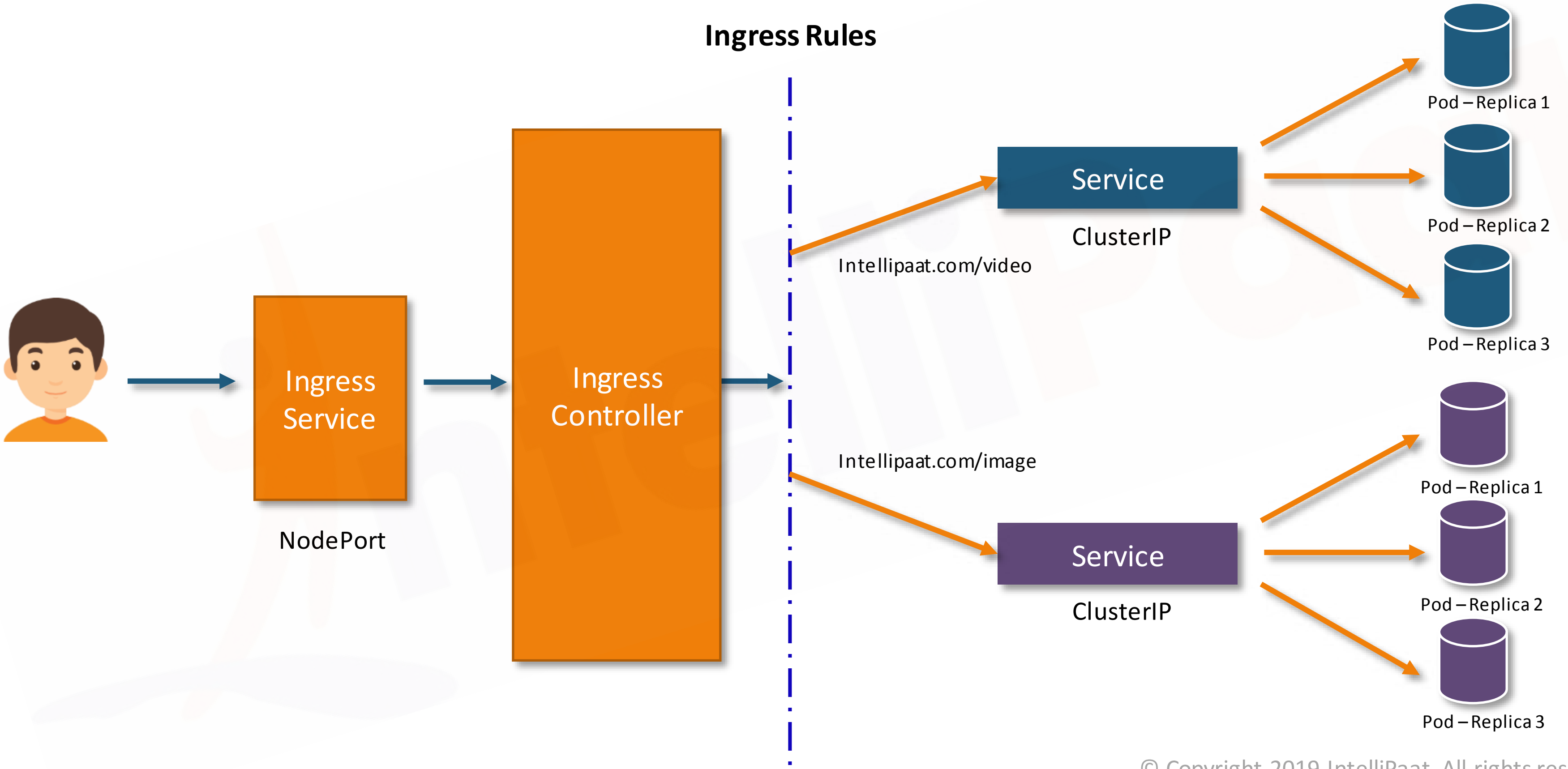
Creating an Ingress

What is an Ingress?

Kubernetes ingress is a collection of routing rules that govern how external users access services running in a Kubernetes cluster.



What is an Ingress?



Installing Ingress Controller

We will be using the nginx ingress controller for our demo. We can download it from the following link:

Link

https://github.com/kubernetes/ingress-nginx/blob/master/docs/deploy/index.md

Defining Ingress Rules

The following rule, will redirect traffic which asks for /foo to nginx service. All other requests will be redirected to ingress controller’s default page.

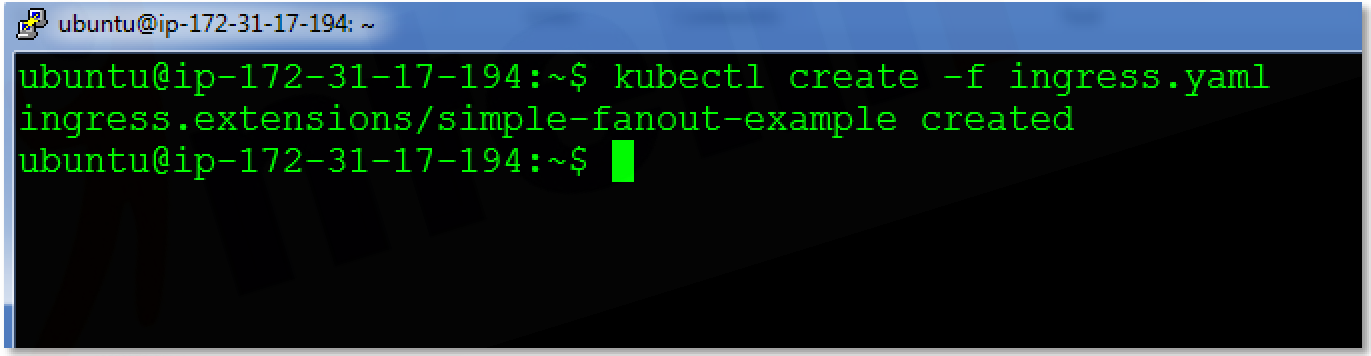
```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: simple-fanout-example
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /foo
        backend:
          serviceName: nginx
          servicePort: 80
```

Deploying Ingress Rules

To deploy ingress rules, we use the following syntax:

Syntax

kubectl create –f ingress.yaml



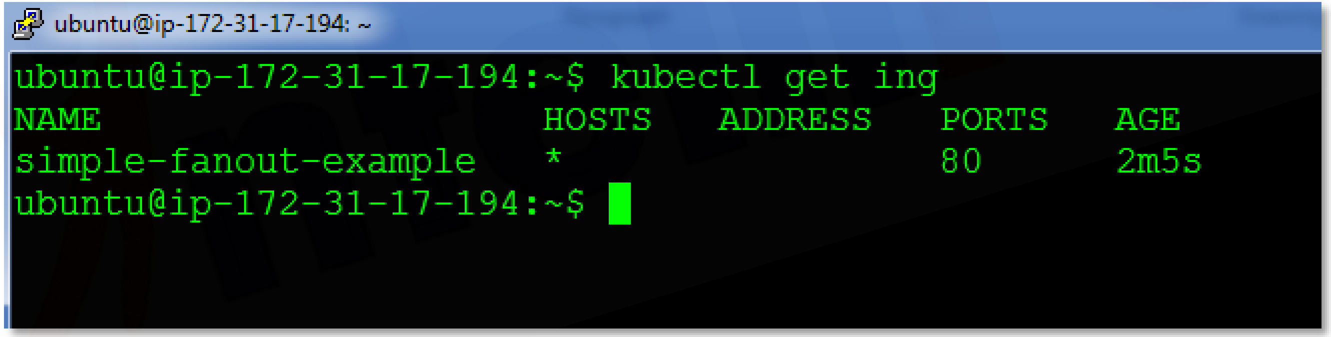
```
ubuntu@ip-172-31-17-194:~$ kubectl create -f ingress.yaml
ingress.extensions/simple-fanout-example created
ubuntu@ip-172-31-17-194:~$
```

Viewing Ingress Rules

To list the ingress rules we use the following syntax:

Syntax

kubectl get ing

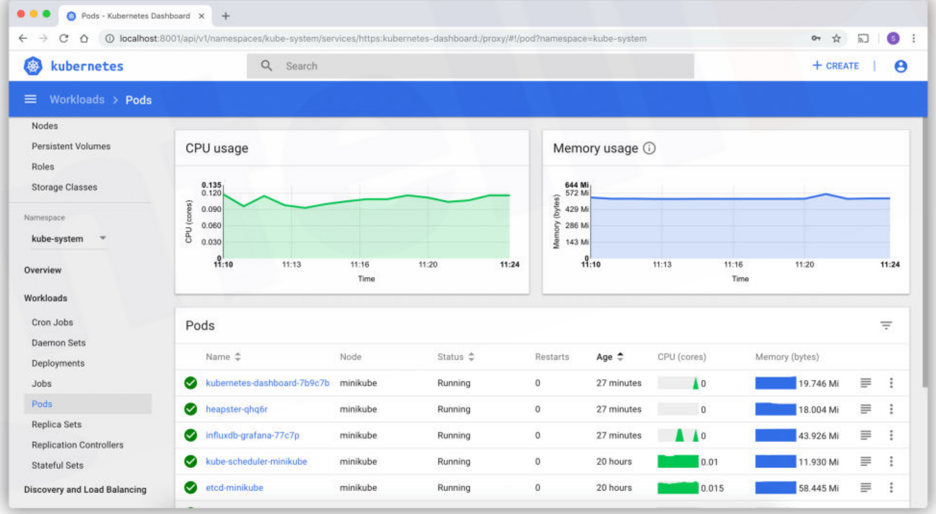


```
ubuntu@ip-172-31-17-194:~$ kubectl get ing
NAME                     HOSTS    ADDRESS    PORTS    AGE
simple-fanout-example    *                   80       2m5s
ubuntu@ip-172-31-17-194:~$
```

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 cluster resources.



Pods

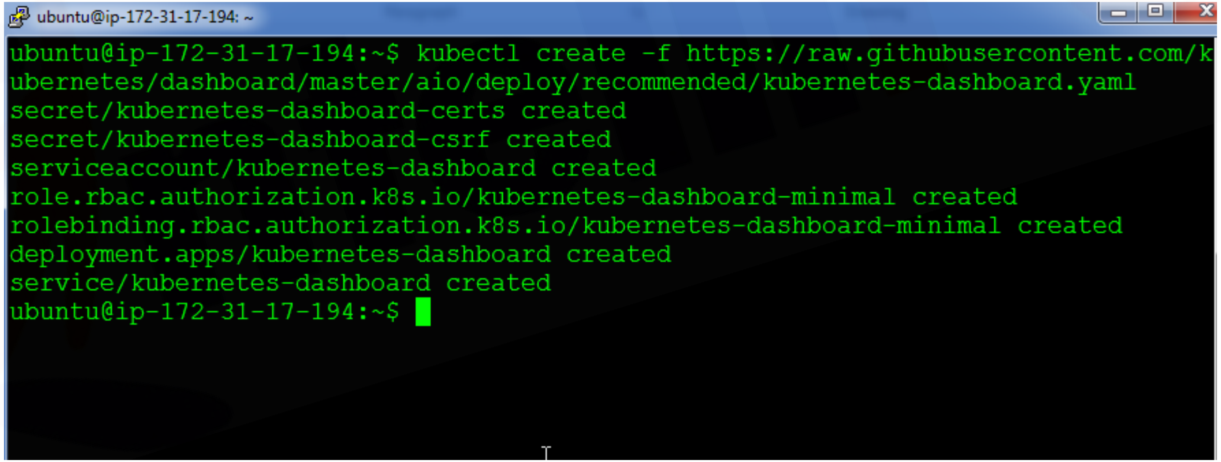
Name	Node	Status	Restarts	Age	CPU (cores)	Memory (bytes)
kubernetes-dashboard-7b9c7b	minikube	Running	0	27 minutes	0	19.746 Mi
heapster-qhq6r	minikube	Running	0	27 minutes	0	18.004 Mi
influxdb-grafana-77c7p	minikube	Running	0	27 minutes	0	43.926 Mi
kube-scheduler-minikube	minikube	Running	0	20 hours	0.01	11.930 Mi
etcd-minikube	minikube	Running	0	20 hours	0.015	58.445 Mi

Installing Kubernetes Dashboard

To install Kubernetes Dashboard, execute the following command:

Syntax

```
kubectl create -f
https://raw.githubusercontent.com/kubernetes/dashboard/master/aio/deploy/recommended/kubernetes-
dashboard.yaml
```



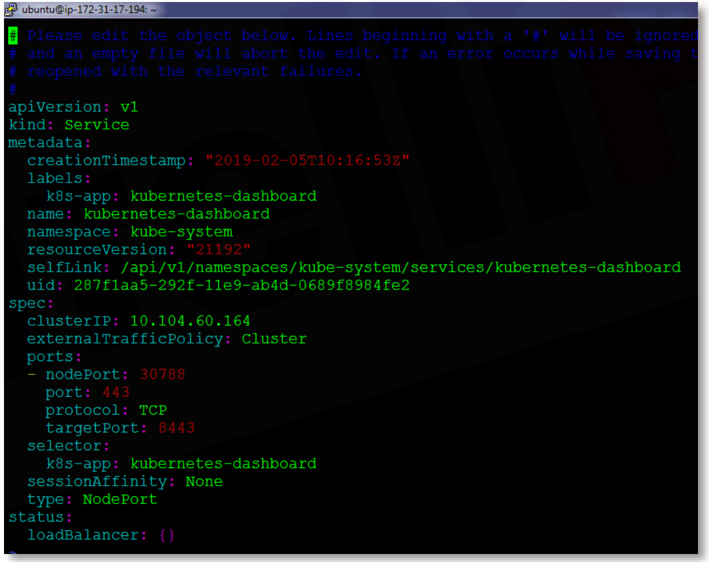
```
ubuntu@ip-172-31-17-194:~$ kubectl create -f https://raw.githubusercontent.com/k
ubernetes/dashboard/master/aio/deploy/recommended/kubernetes-dashboard.yaml
secret/kubernetes-dashboard-certs created
secret/kubernetes-dashboard-csrf created
serviceaccount/kubernetes-dashboard created
role.rbac.authorization.k8s.io/kubernetes-dashboard-minimal created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard-minimal created
deployment.apps/kubernetes-dashboard created
service/kubernetes-dashboard created
ubuntu@ip-172-31-17-194:~$
```

Accessing Kubernetes Dashboard

Change the service type for kubernetes-dashboard to NodePort

Syntax

```
kubectl -n kube-system edit service kubernetes-dashboard
```



ubuntu@ip-172-31-17-194: ~

```
# Please edit the object below. Lines beginning with a '#' will be ignored
# and an empty file will abort the edit. If an error occurs while saving t
# reopened with the relevant failures.
#
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: "2019-02-05T10:16:53Z"
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kube-system
  resourceVersion: "21192"
  selfLink: /api/v1/namespaces/kube-system/services/kubernetes-dashboard
  uid: 287f1aa5-292f-11e9-ab4d-0689f8984fe2
spec:
  clusterIP: 10.104.60.164
  externalTrafficPolicy: Cluster
  ports:
  - nodePort: 30788
    port: 443
    protocol: TCP
    targetPort: 8443
  selector:
    k8s-app: kubernetes-dashboard
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
```

Logging into Kubernetes Dashboard

1. Check the NodePort from the kubernetes-dashboard service
2. Browse to your cluster on the Internet browser, and enter the IP address
3. Click on Token, which will ask you for the token entry
4. Generate a token using the following command:

```
$ kubectl create serviceaccount cluster-admin-dashboard-sa
$ kubectl create clusterrolebinding cluster-admin-dashboard-sa \
  --clusterrole=cluster-admin \
  --serviceaccount=default:cluster-admin-dashboard-sa

$ TOKEN=$(kubectl describe secret $(kubectl -n kube-system get secret | awk '/^cluster-admin-dashboard-sa-
token-/{print $1}') | awk '$1=="token:"{print $2}')

$ echo $TOKEN
```

5. Finally, enter the token and login to your dashboard

Hands-on: Deploying an App Using Dashboard

Quiz

Quiz

1. **Which of these is an installation method of Kubernetes cluster?**

A. Kubeadm

B. Kops

C. Both A and B

D. None of these

Quiz

1. Which of these is an installation method of Kubernetes cluster?

A. Kubeadm

B. Kops

[✓ C. Both A and B]

D. None of these

Quiz

2. Which of these components is a distributed key–value store?

A. Kubelet

B. Scheduler

C. etcd

D. None of these

Quiz

2. Which of these components is a distributed key–value store?

A. Kubelet

B. Scheduler

[✓ C. etcd]

D. None of these

Quiz

3. Which type of service is used to expose application without using Cloud Native Support?

A. Load Balancer

B. NodePort

C. ExternalName

D. None of these

Quiz

3. Which type of service is used to expose application without using Cloud Native Support?

A. Load Balancer

[✓ **B. NodePort**]

C. ExternalName

D. None of these

Quiz

4. Which of these is not a component of Kubernetes Slave?

A. Kubelet

B. Docker

C. Scheduler

D. None of these

Quiz

4. Which of these is not a component of Kubernetes Slave?

A. Kubelet

B. Docker

[✓ **C. Scheduler**]

D. None of these

Quiz

5. Which of these components helps us to route traffic based on the user request?

A. Deployment

B. Service

C. Ingress

D. None of these

Quiz

5. Which of these components helps us to route traffic based on the user request?

A. Deployment

B. Service

[✓ **C. Ingress**]

D. None of these

IntelliPaat

☎ India: +91-7847955955

US: 1-800-216-8930 (TOLL FREE)

✉ support@intellipaat.com

24/7 Chat with Our Course Advisor