Introduction Kubernetes

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

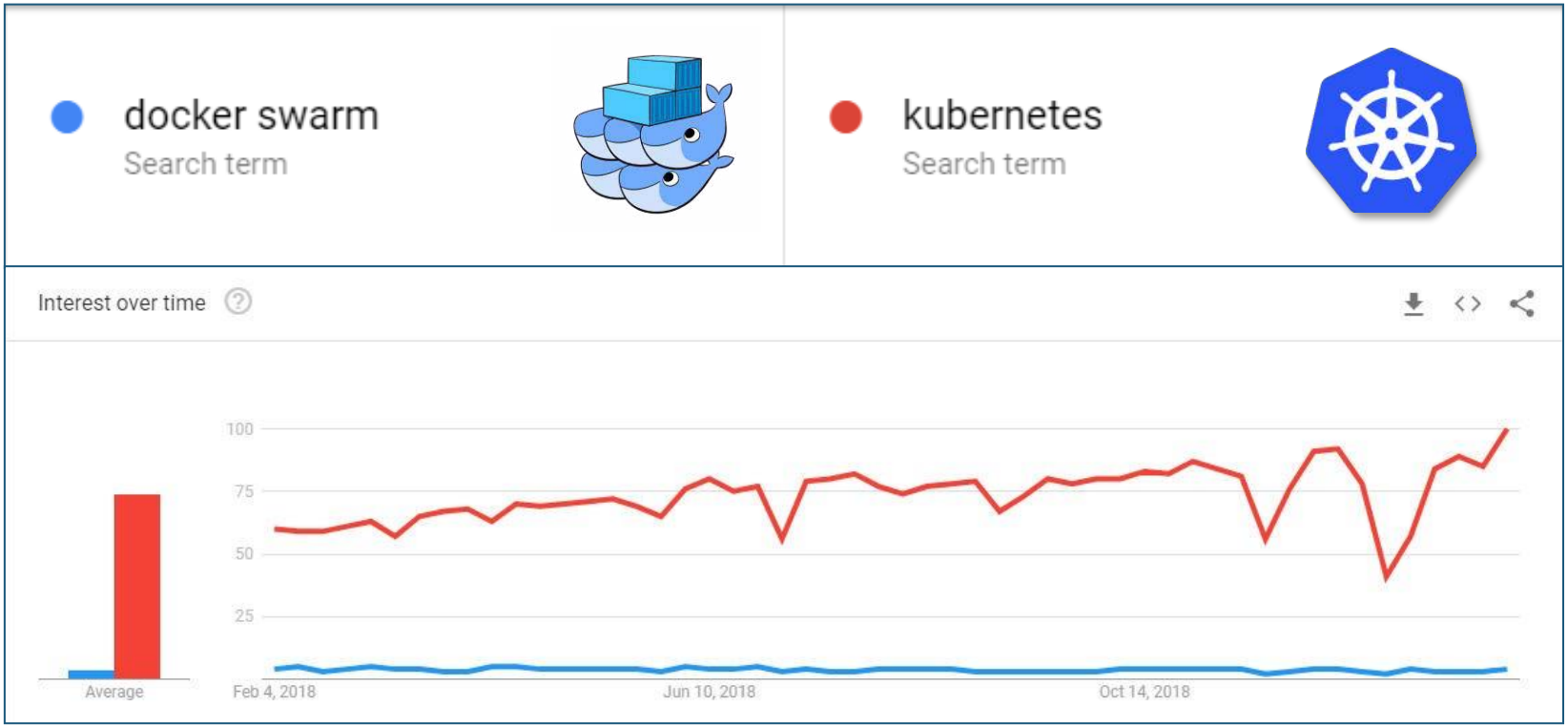
- Kubernetes is an open-source container orchestration software
- It was originally developed by Google
- Itwas first released on July 21st 2015
- It is the ninth most active repository on GitHub in terms of number of commits

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

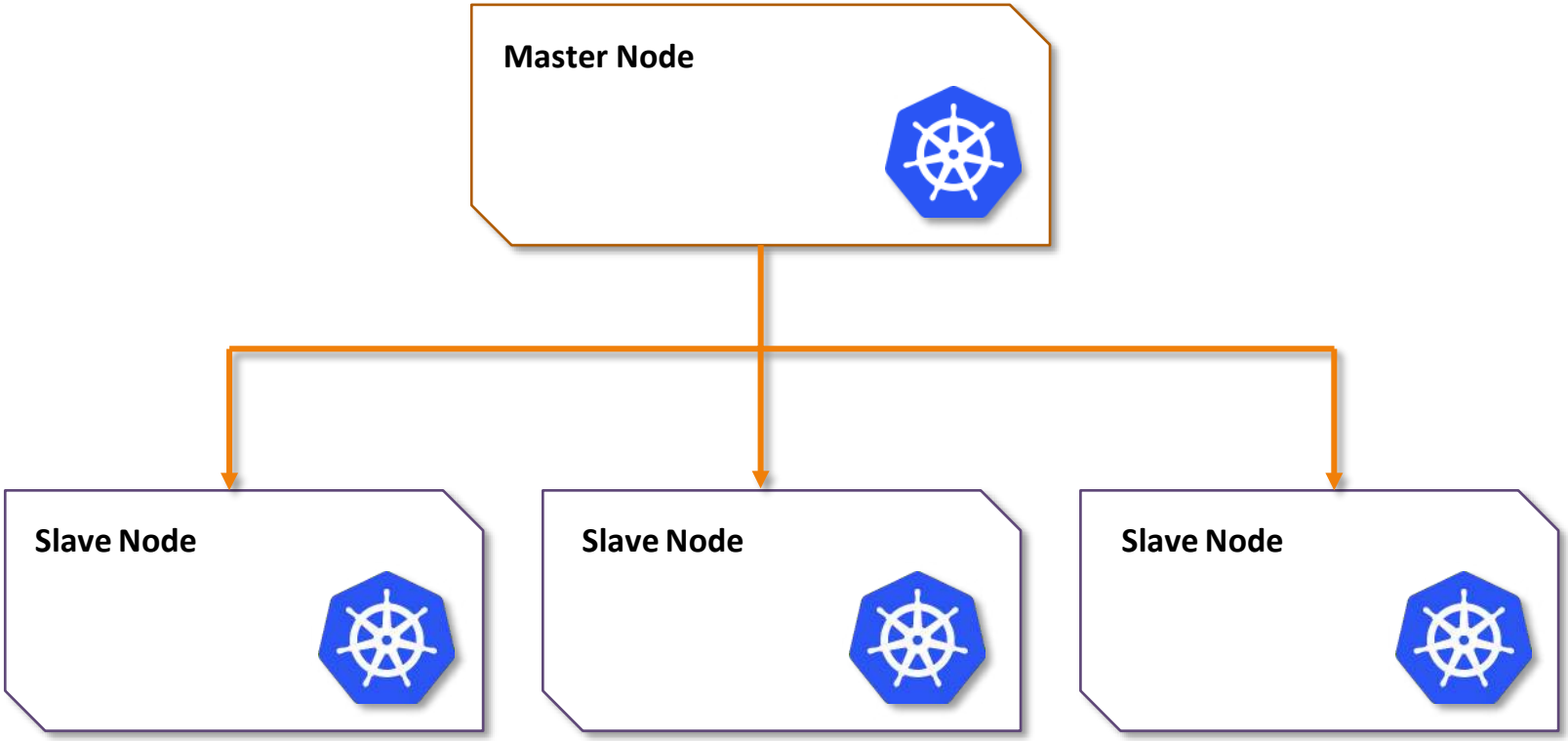
- Easy to Install and Initialize
- Faster when compared to Kubernetes
- Not Reliable and has less features

Kubernetes

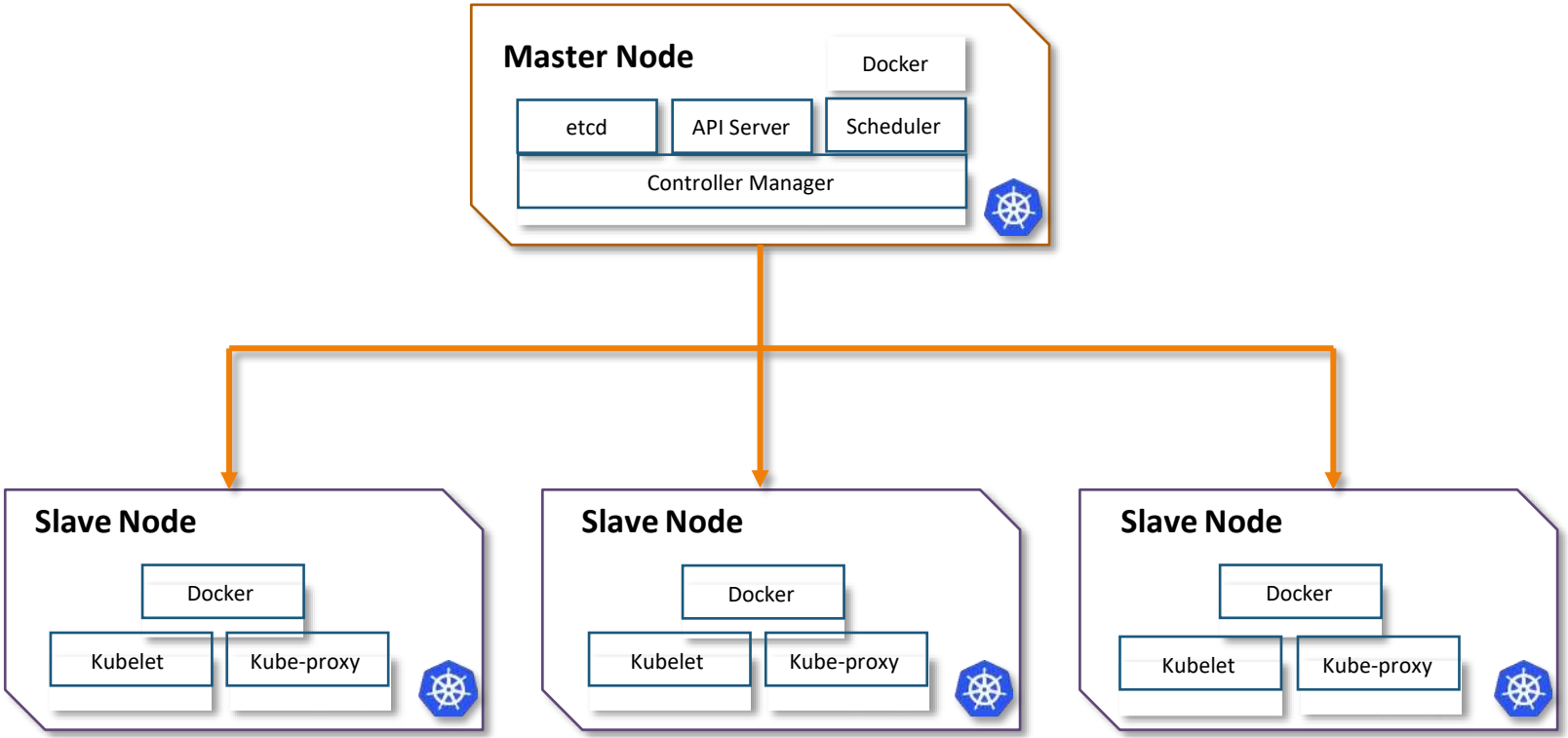
- Complex Procedure to install Kubernetes
- Slower when compared with Docker Swarm
- More Reliable and comparatively has more features

Kubernetes Architecture

Kubernetes Architecture



Kubernetes Architecture

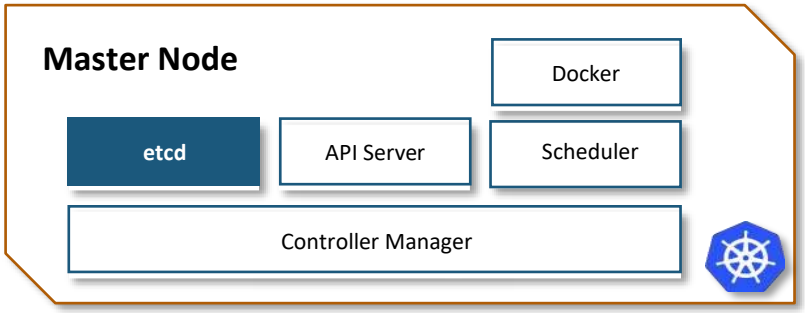


Kubernetes Architecture – Master Components

Kubernetes Architecture – Master Components

- **etcd**
- API Server
- Scheduler
- Controller Manager

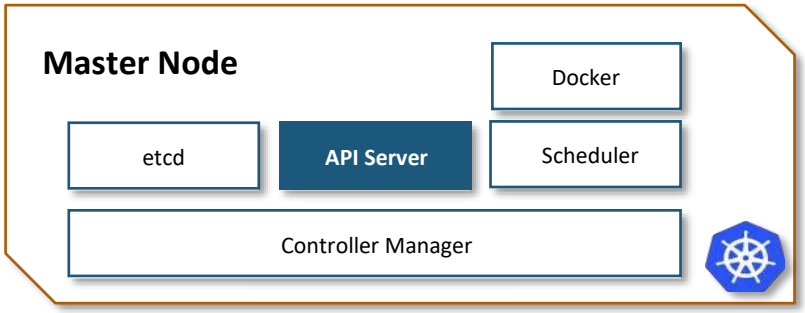
It is a highly available distributed key value store, which is used to store cluster wide secrets. It is only accessible by Kubernetes API server, as it has sensitive information.



Kubernetes Architecture – Master Components

- etcd
- **API Server**
- Scheduler
- Controller Manager

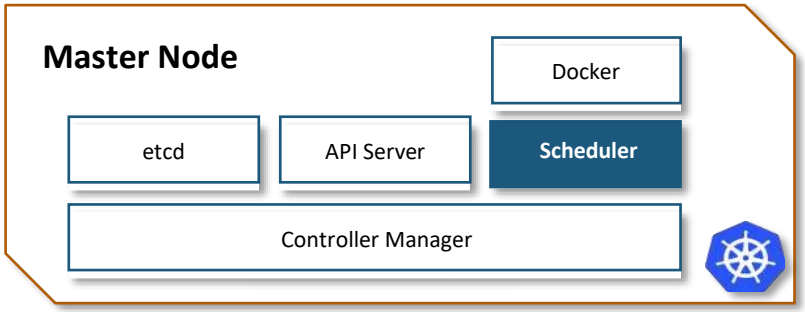
It exposes the Kubernetes API. The Kubernetes API is the front-end for Kubernetes Control Plane, and is used to deploy and execute all operations in Kubernetes



Kubernetes Architecture – Master Components

- etcd
- API Server
- **Scheduler**
- Controller Manager

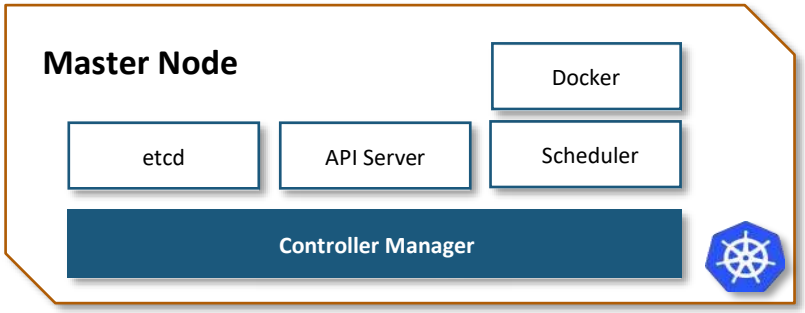
The scheduler takes care of scheduling of all the processes, Dynamic Resource Management and manages present and future events on the cluster



Kubernetes Architecture – Master Components

- etcd
- API Server
- Scheduler
- **Controller Manager**

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



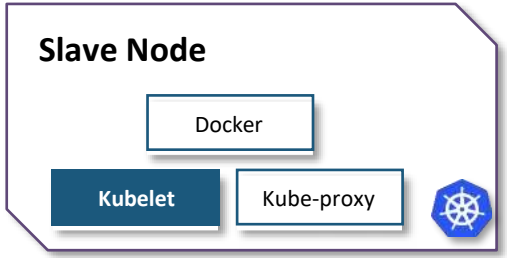
Kubernetes Architecture – Slave Components

Kubernetes Architecture – Master Components

Kubelet

Kube-Proxy

Kubelet takes the specification from the API server, and ensures the application is running according to the specifications which were mentioned. Each node has it's kubelet service

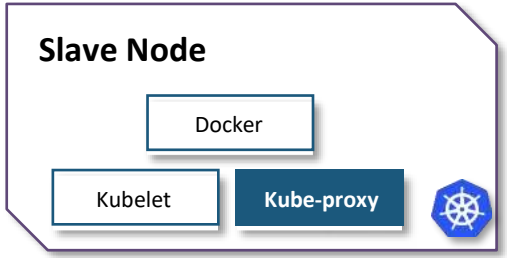


Kubernetes Architecture – Master Components

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

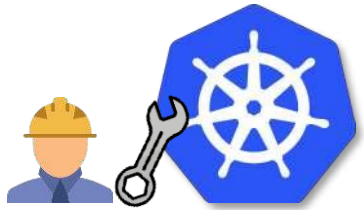


Kubernetes Installation

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

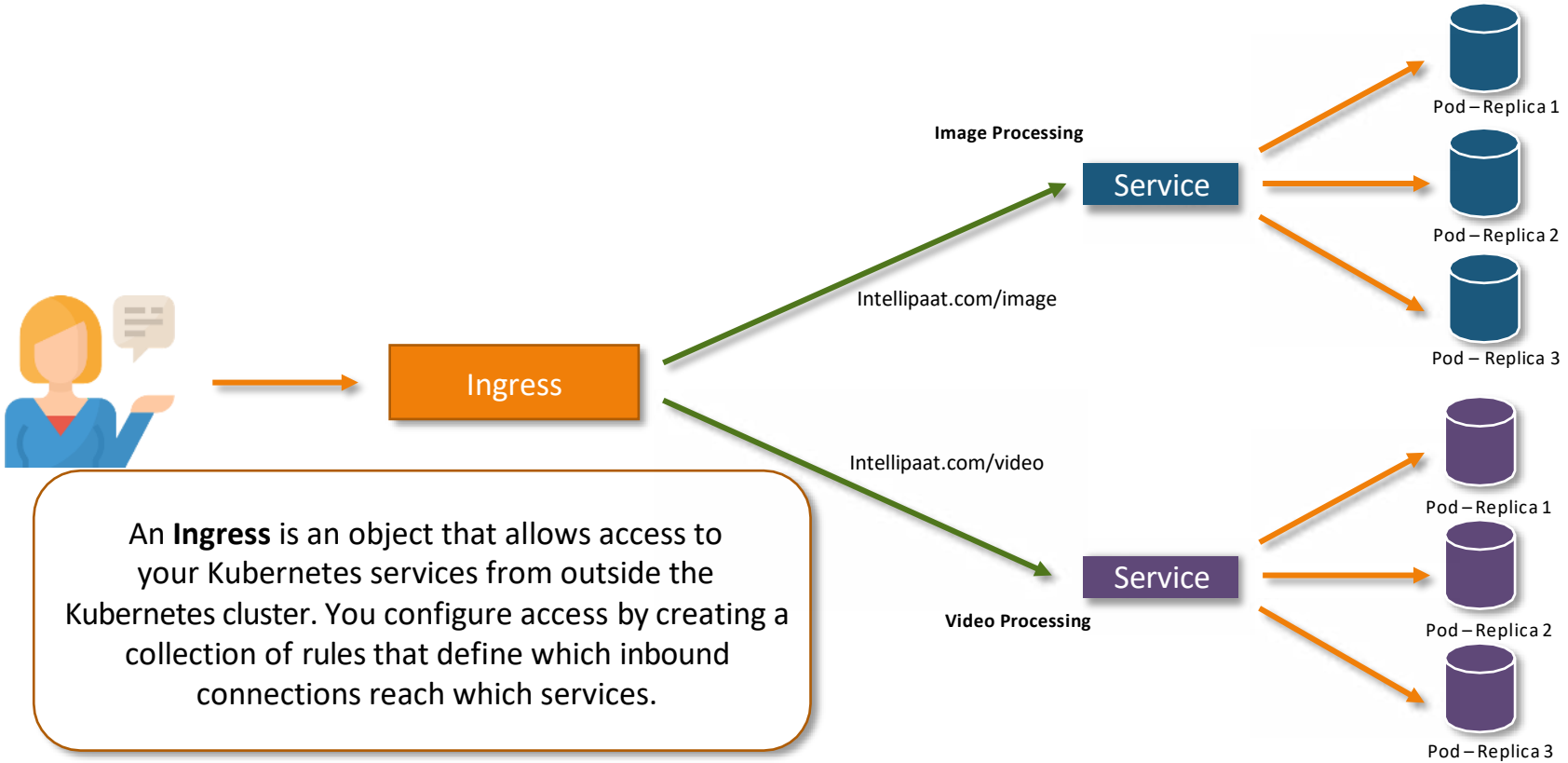
Service

Pod – Replica 1

Pod – Replica 2

Pod – Replica 3

Working of Kubernetes

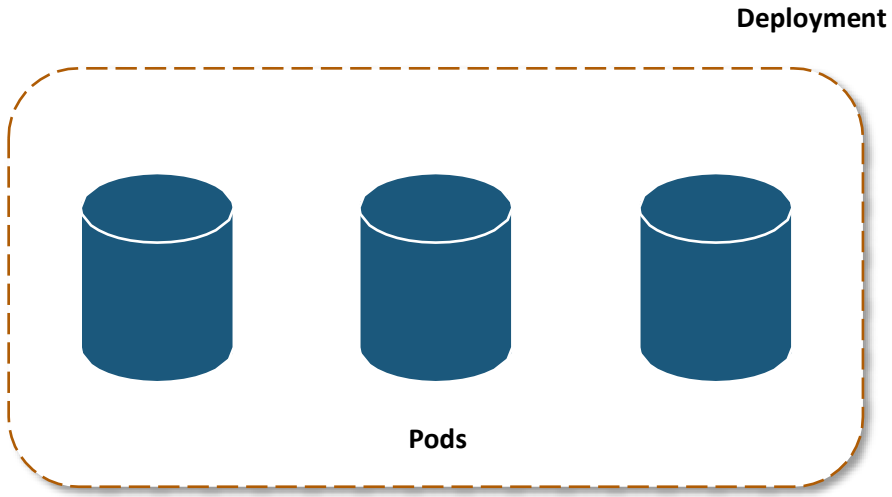


An **Ingress** is an object that allows access to your Kubernetes services from outside the Kubernetes cluster. You configure access by creating a collection of rules that define which inbound connections reach which services.

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 maintain the desired state which is 3 pods, until this deployment is deleted

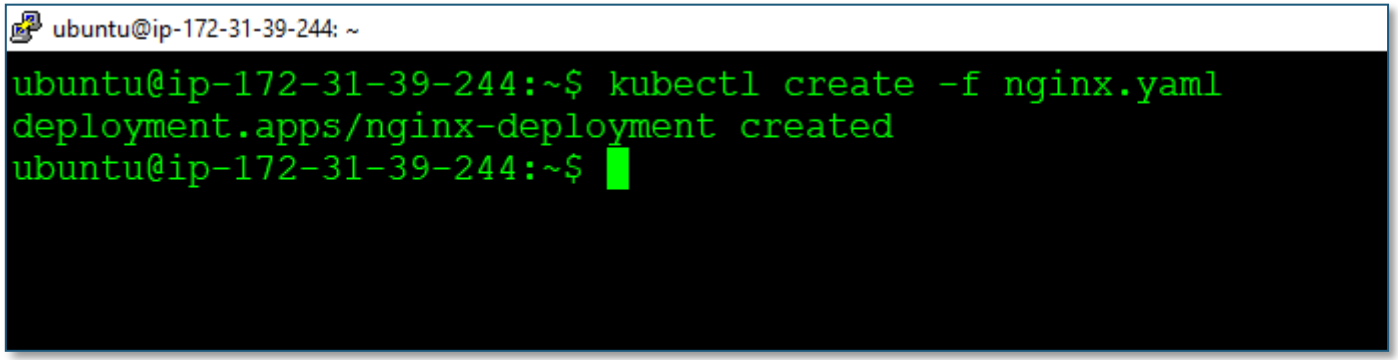
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
  matchLabels:
     app: nginx
  template:
  metadata:
     labels:
       app: nginx
   spec:
   containers:
     - name: nginx
       image: nginx:1.7.9
       ports: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

Creating a Deployment

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

Syntax

```
kubectl create –f nginx.yaml
```

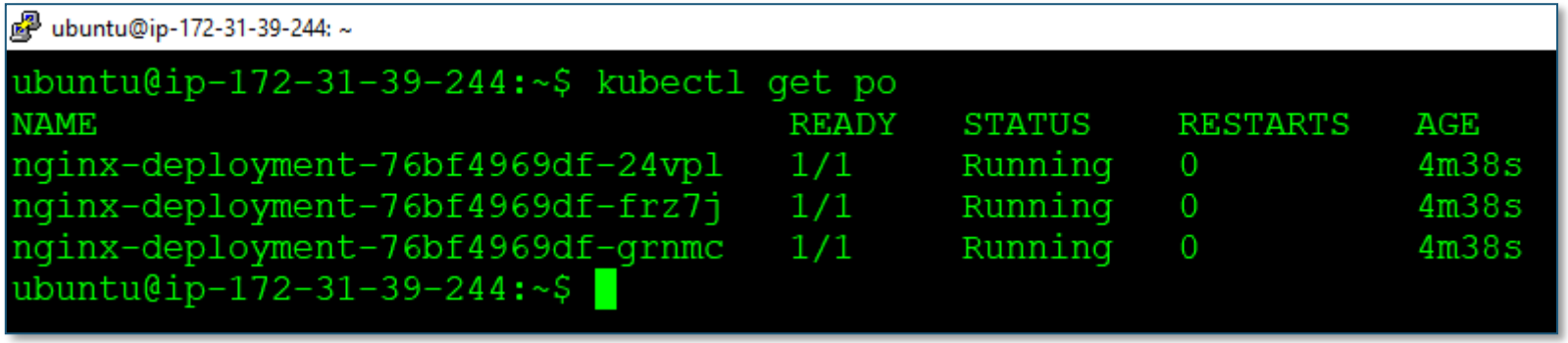


List the Pods

To view the pods, type the following command:

Syntax

```
kubectl get po
```

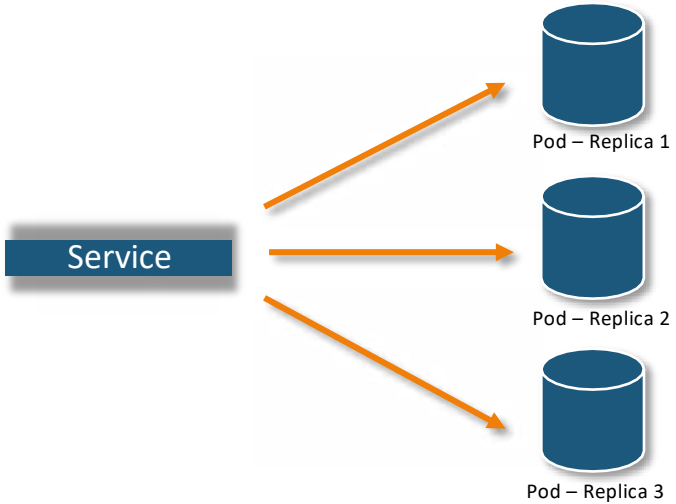


As you can see, the number of pods are matching with the number of replicas specified in the deployment file

Creating a Service

Creating a Service

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



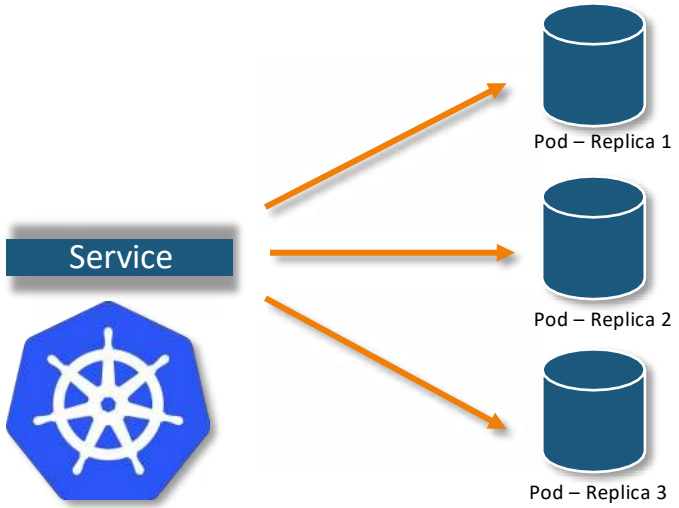
Service Types

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 contents of the ExternalName

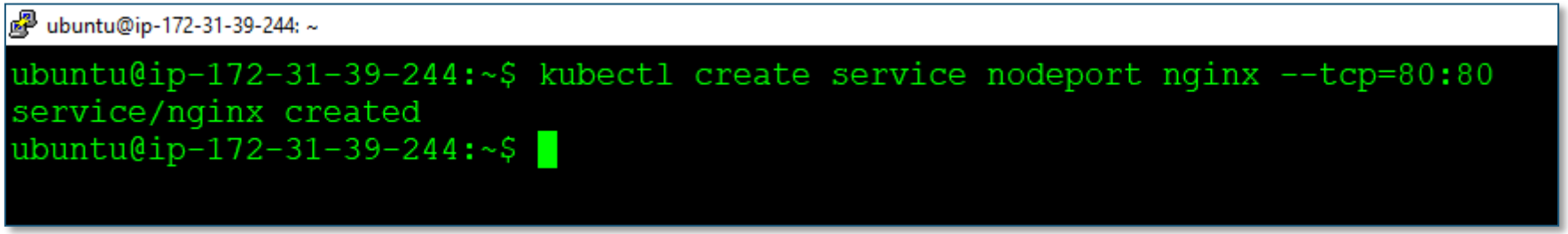


Creating a NodePort Service

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>
```

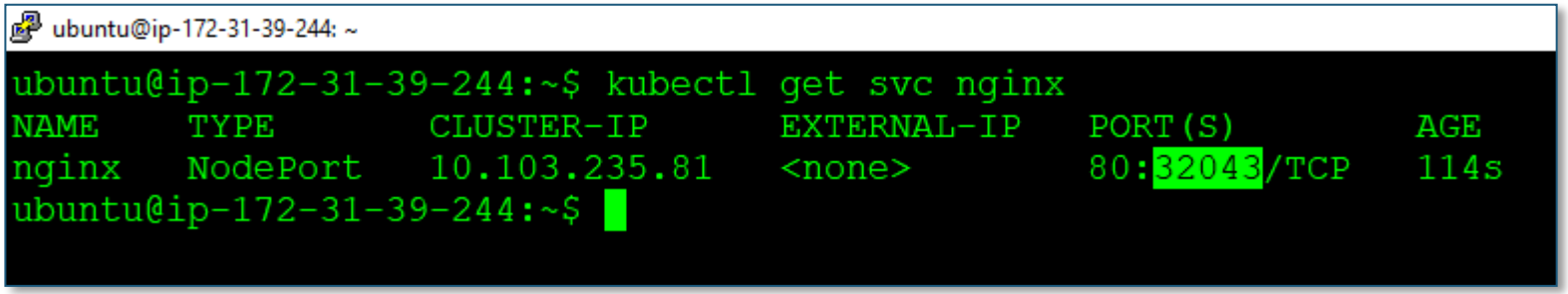


Creating a NodePort Service

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

Syntax

```
kubectl get svc nginx
```

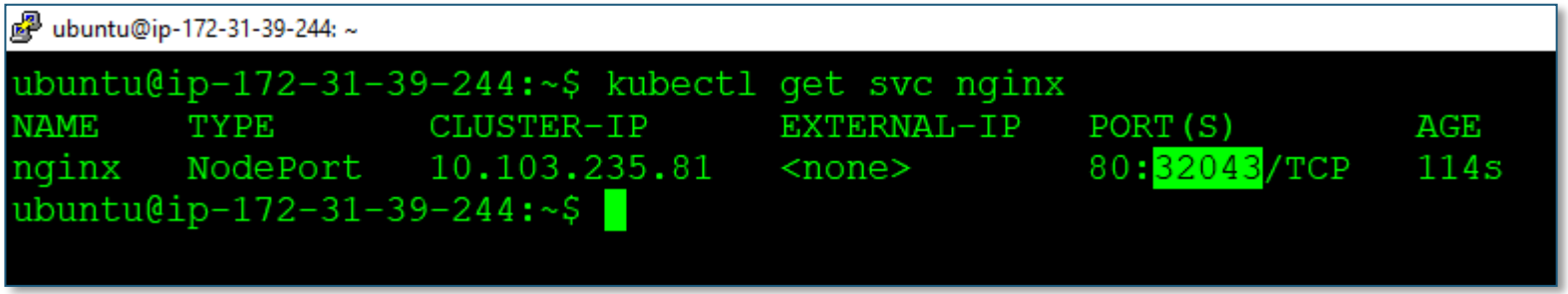


Creating a NodePort Service

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

Syntax

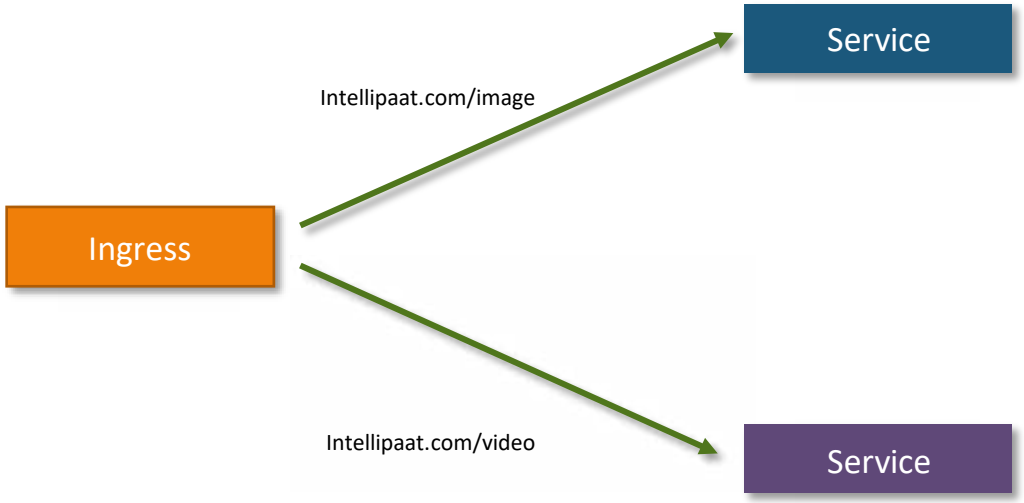
```
kubectl get svc nginx
```



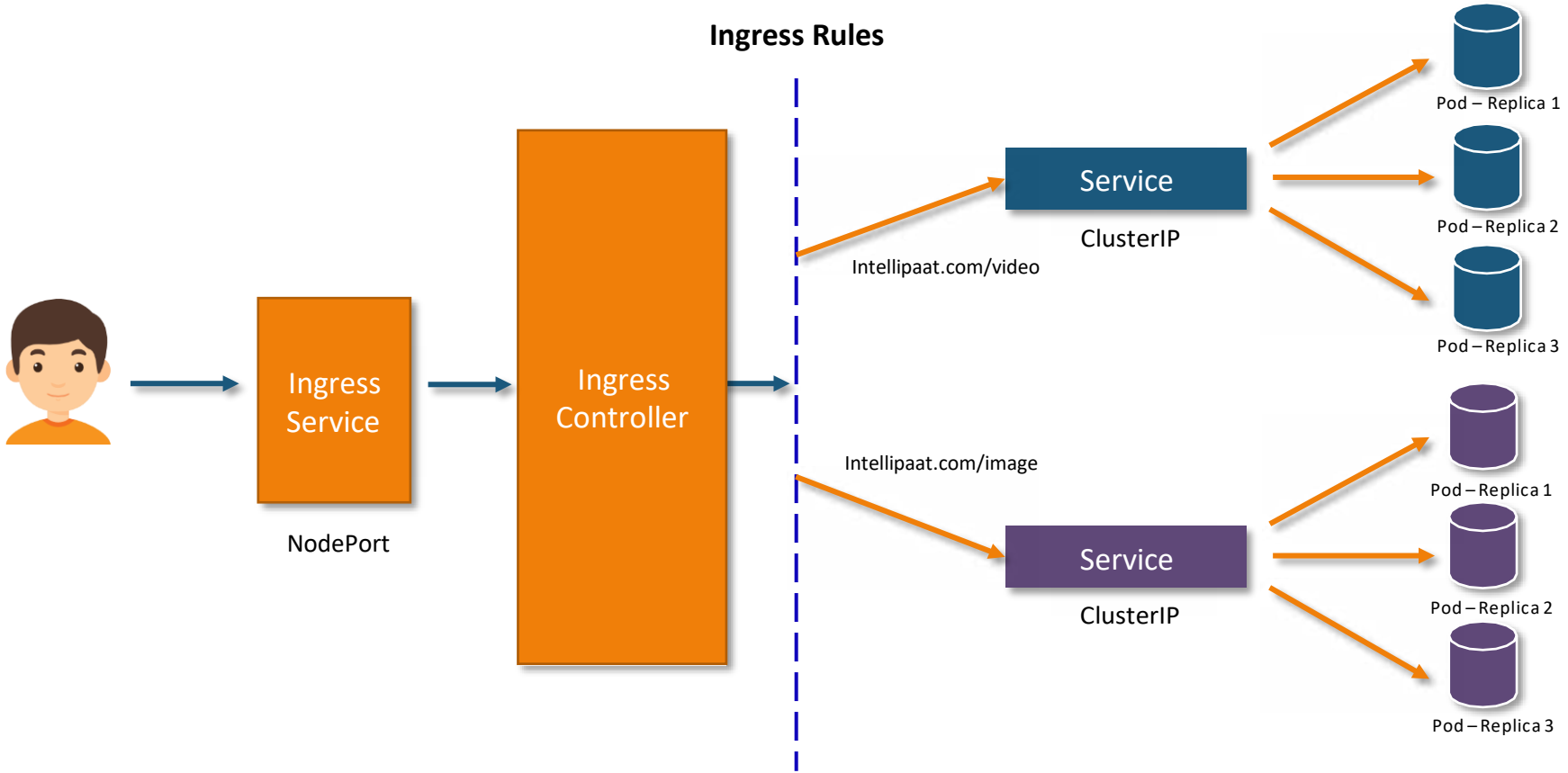
Creating an Ingress

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

Define Ingress Rules

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

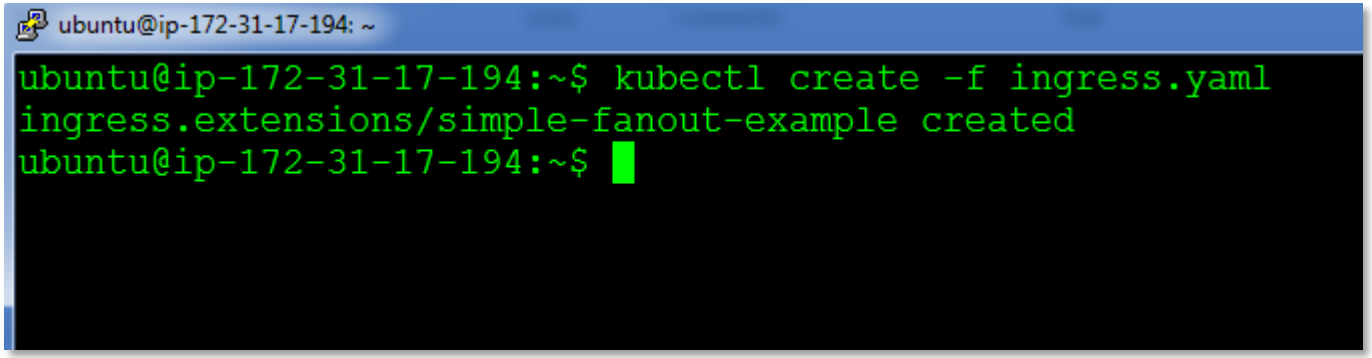
```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: simple-fanout-example
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /foo
        backend:
          serviceName: nginx
          servicePort: 80
```

Deploying Ingress Rules

To deploy the ingress rules, we use the following syntax:

Syntax

```
kubectl create –f ingress.yaml
```

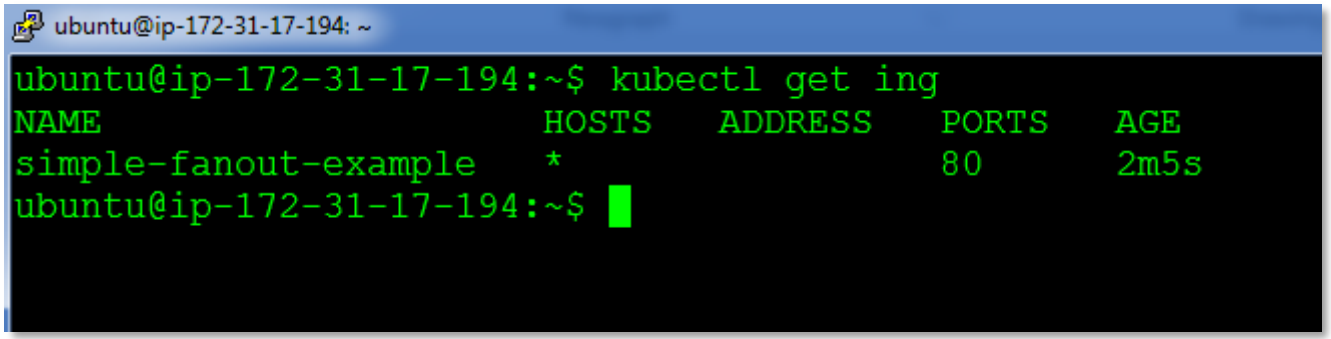


Viewing Ingress Rules

To deploy the ingress rules, we use the following syntax:

Syntax

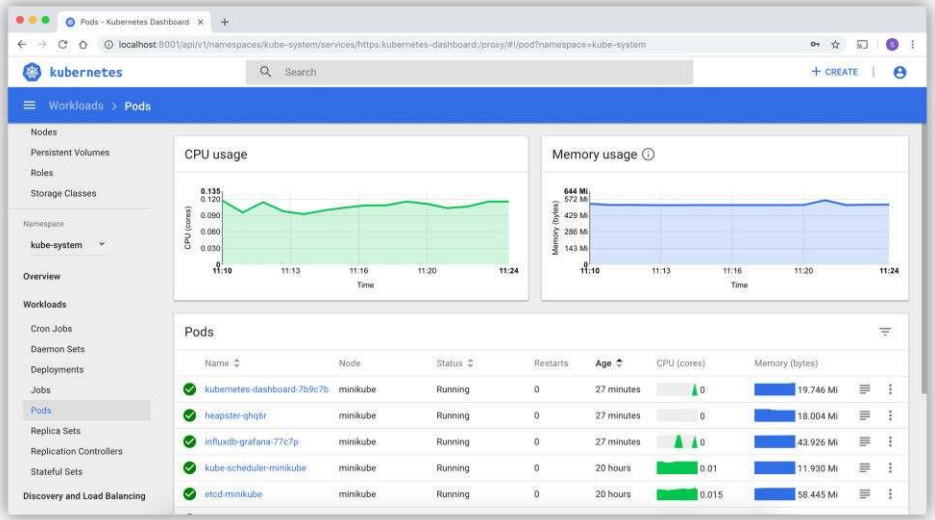
```
kubectl get ing
```



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 resources.

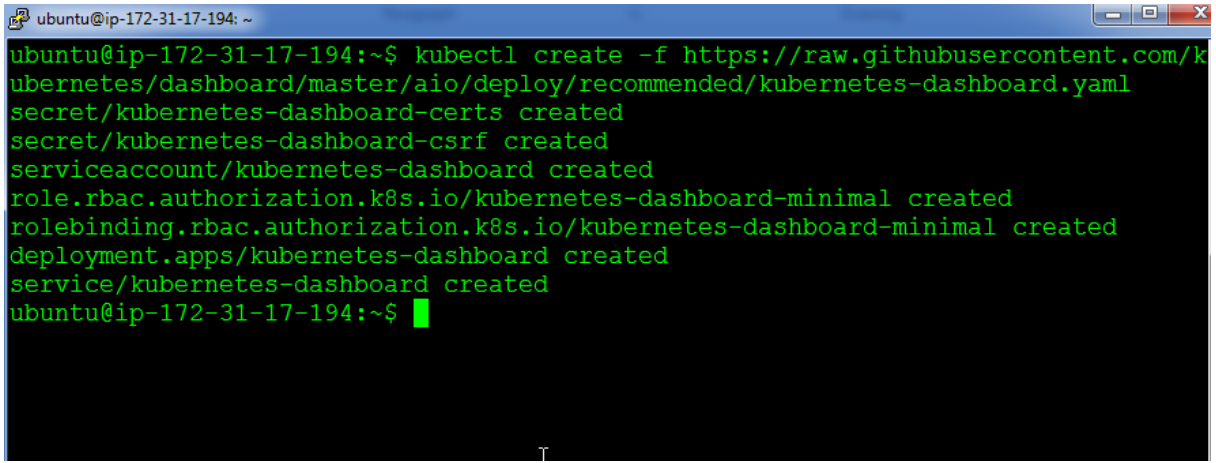


Installing Kubernetes Dashboard

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
```

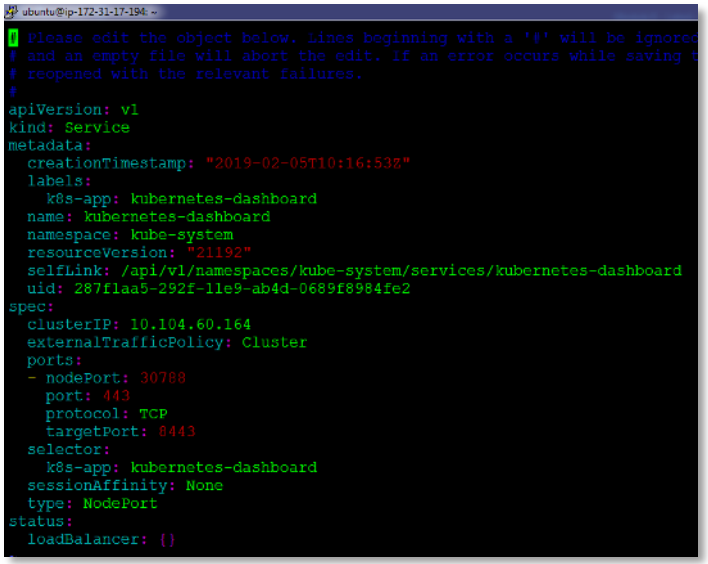


Accessing Kubernetes Dashboard

Change the service type for Kubernetes-Dashboard to Nodeport

Syntax

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



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, it 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