How to make the best use of Live Sessions

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

edureka!

COURSE OUTLINE

Module 04

Introduction to Kubernetes

Kubernetes Architecture

Deploy app to Kubernetes Cluster

Expose App, Scale App And Update App

Managing State with Deployments

Federations, Auditing and Debugging Kubernetes, Security best practices

edureka!

Expose App, Scale App and Update App

Objectives

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

- Service
- Labels and Selectors
- Replication Controller & Replica Set
- Deployment Controller
- Scaling out a deployment using replicas
- Rolling Update
- Horizontal pod autoscaler
- Load balancing
- Ingress and its types

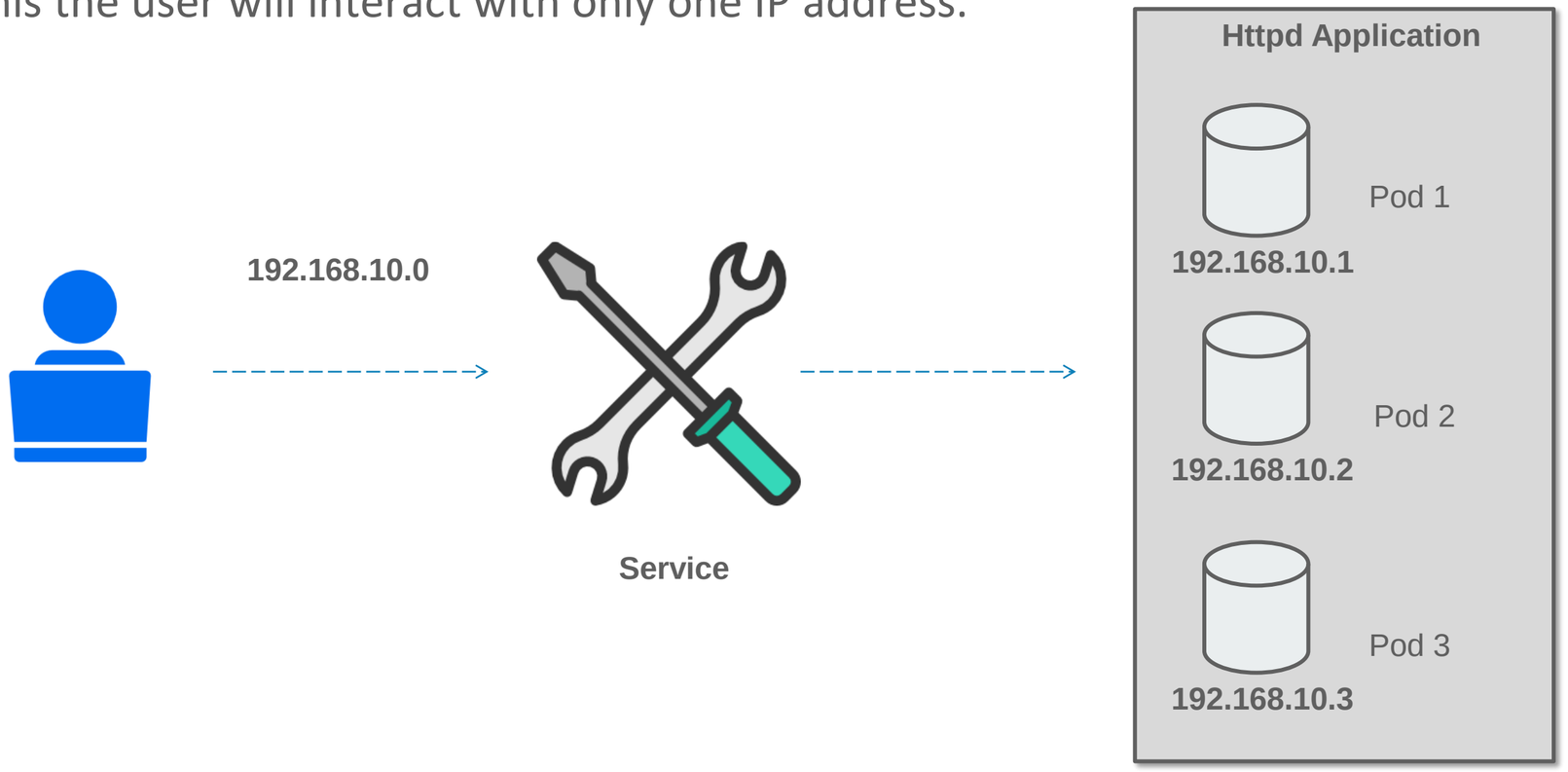
edureka!

Kubernetes Service

edureka!

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.

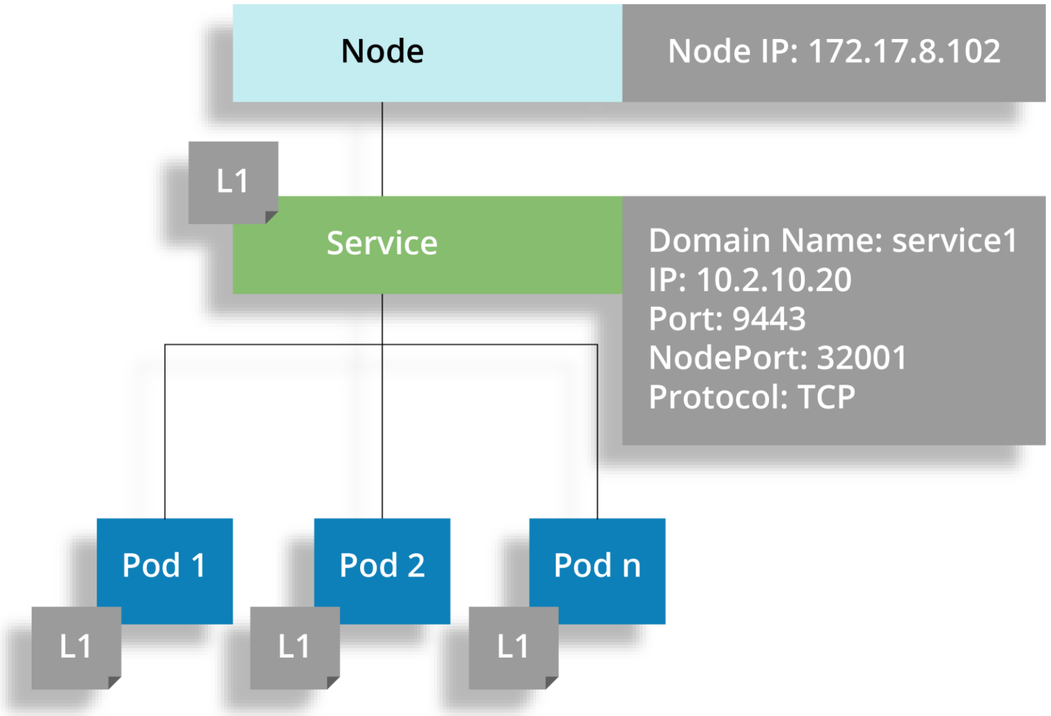


Services in Kubernetes

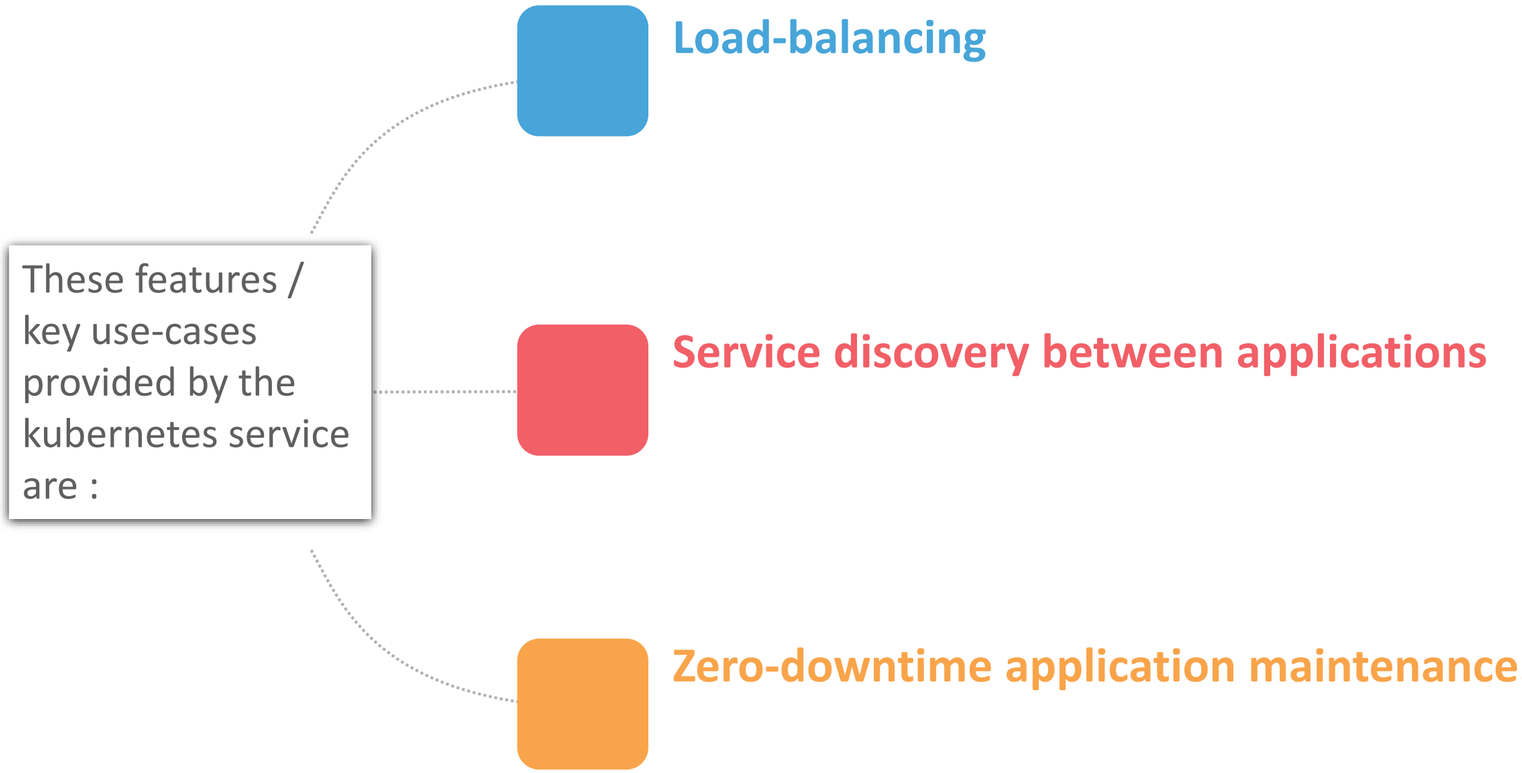
Services defines logical set of pods and the policy through which they will be accessed

They are the abstraction and sometimes called as micro-services

Label Selectors determine the set of pods to be targeted by Services



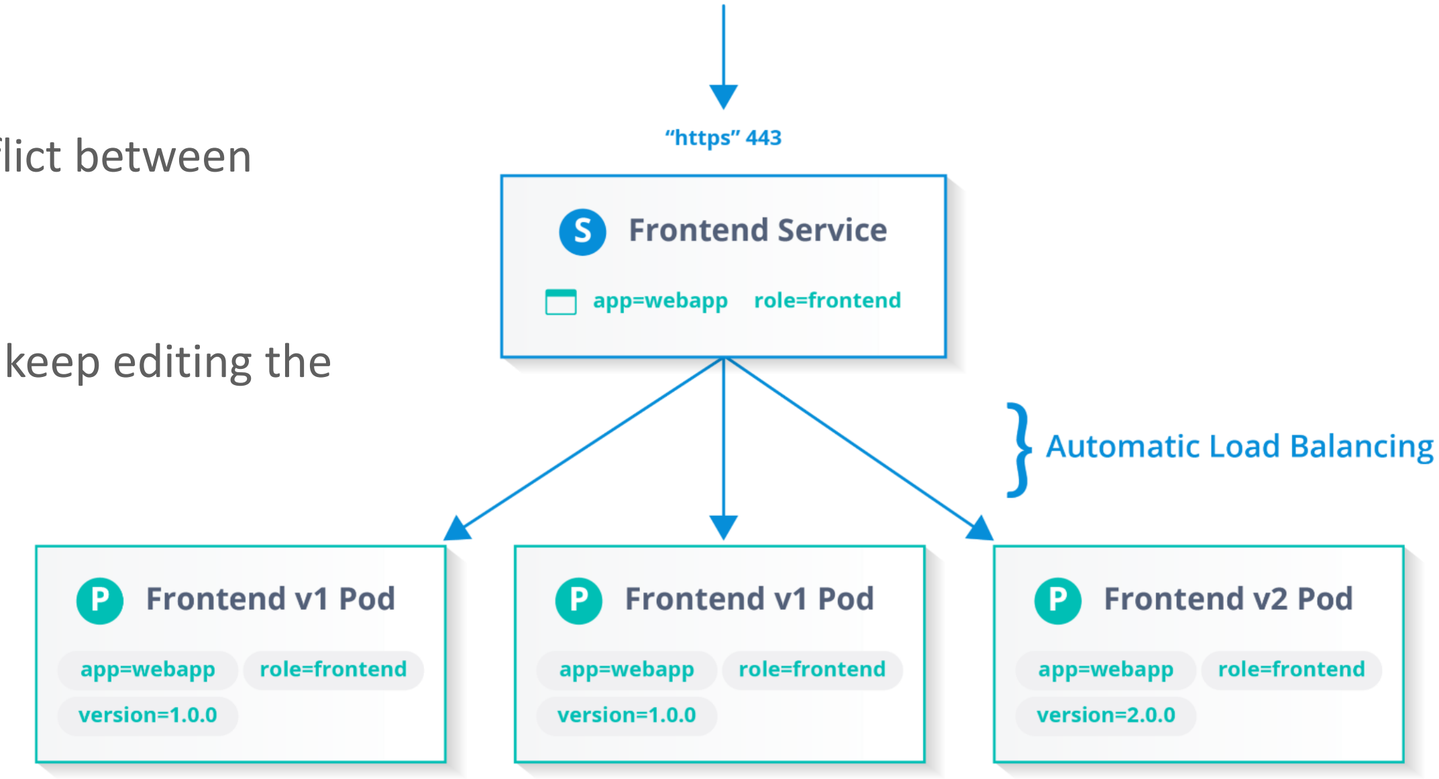
Kubernetes Service



Kubernetes service provides features that are same across the cluster.

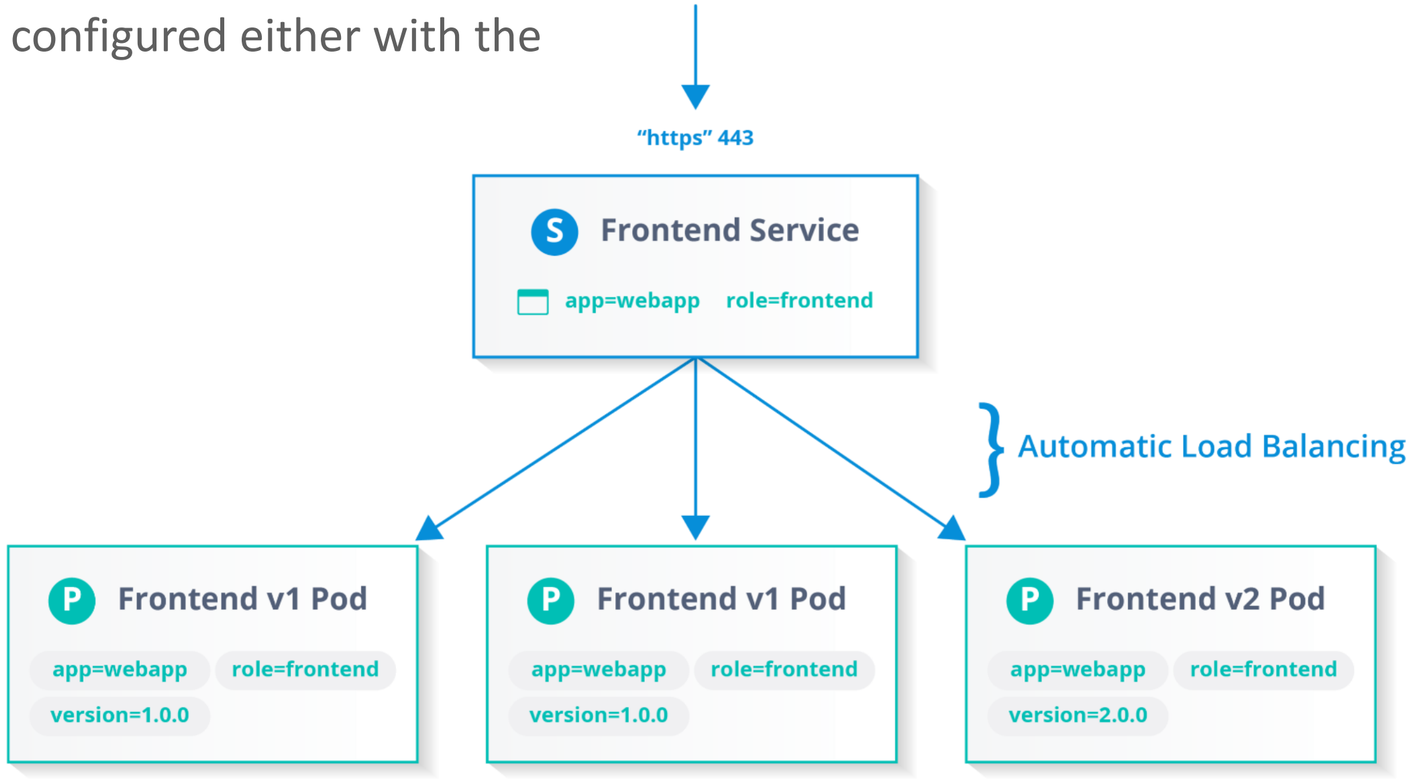
IP Routing and Load Balancing

- One of the key feature of kubernetes is to provide IP routing and load balancing
- This approach helps in avoiding port conflict between applications across the cluster.
- This way, the application doesn't have to keep editing the DB or configuration.



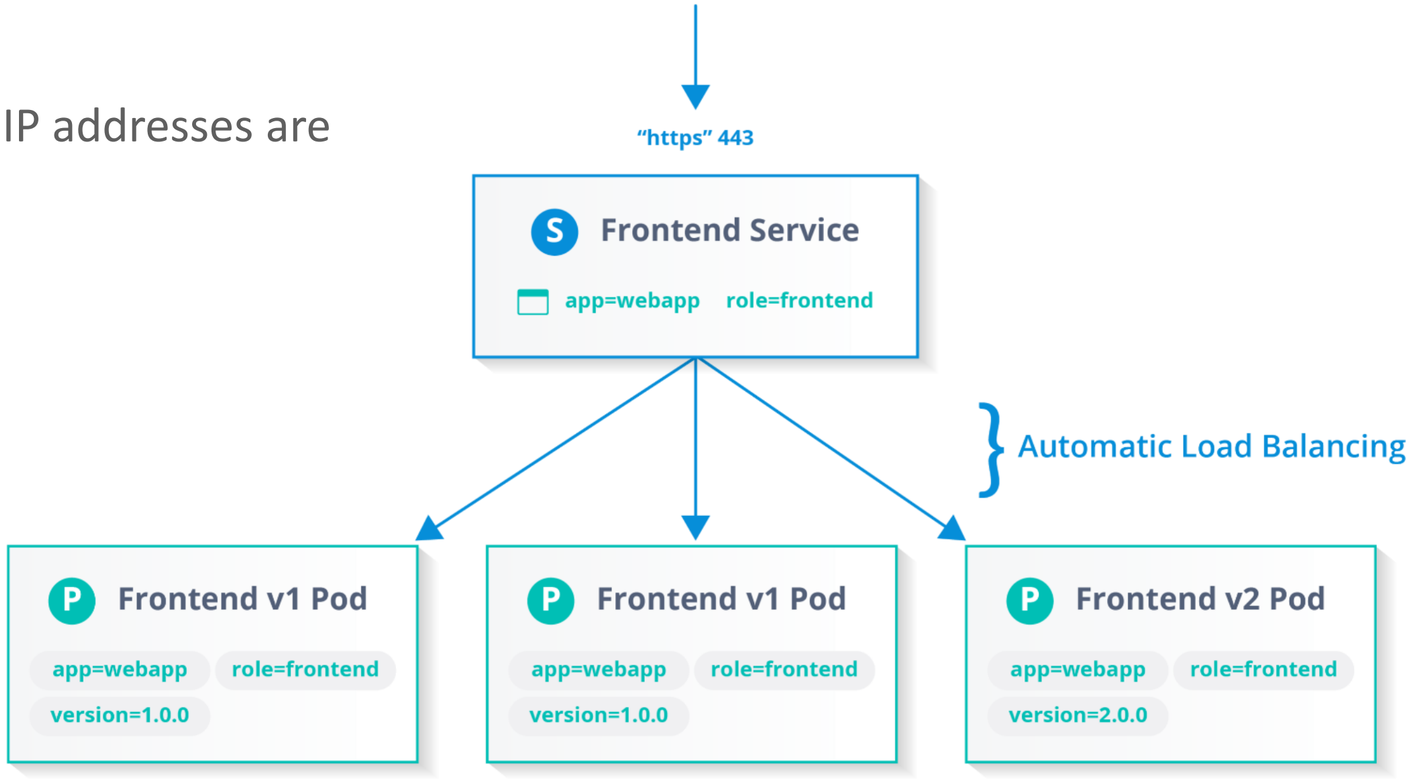
Service Discovery Between Applications

- Each kubernetes service has it’s own unique IP address and DNS name
- Applications that use these service can be configured either with the service IP or DNS name



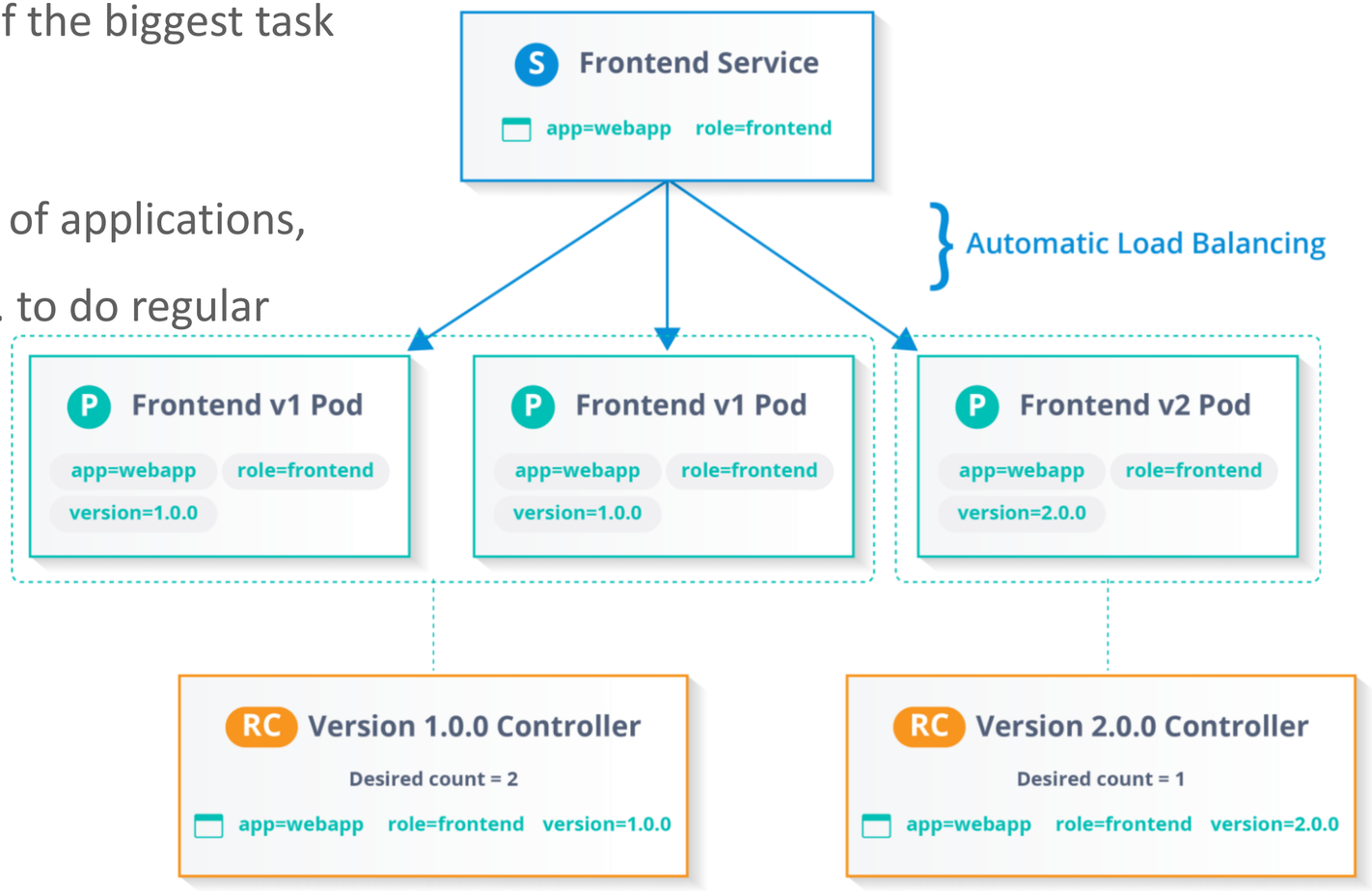
Service Discovery Between Applications

- Accordingly, traffic will be load balanced to the pods of the respective service
- For auto service discovery, all details with IP addresses are passed to all containers within the cluster



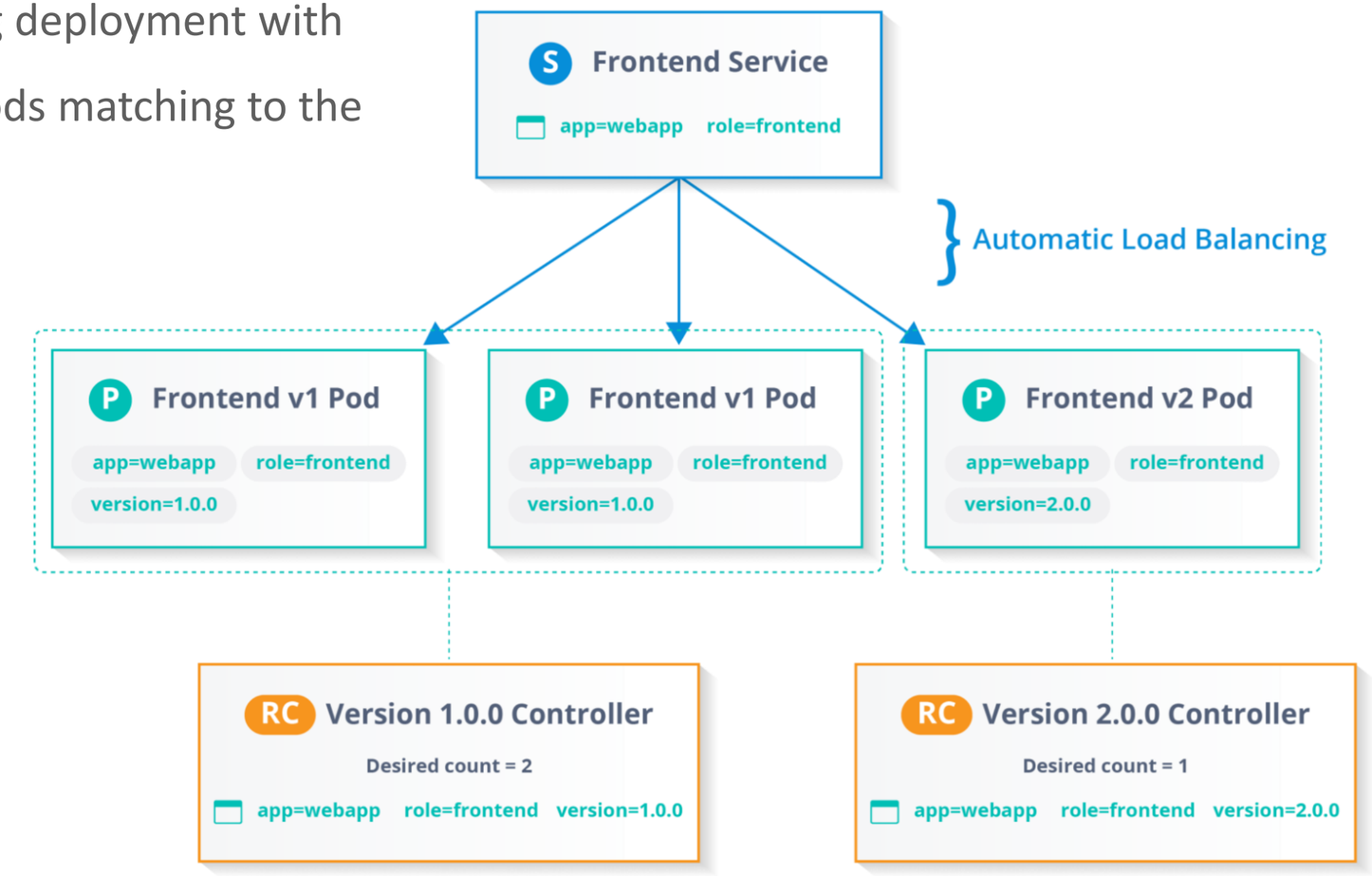
Zero-Downtime Application Maintenance

- Rolling deployment or updates is one of the biggest task for any application team
- They need to prepare the complete list of applications, sub-components, interdependency etc. to do regular maintenance



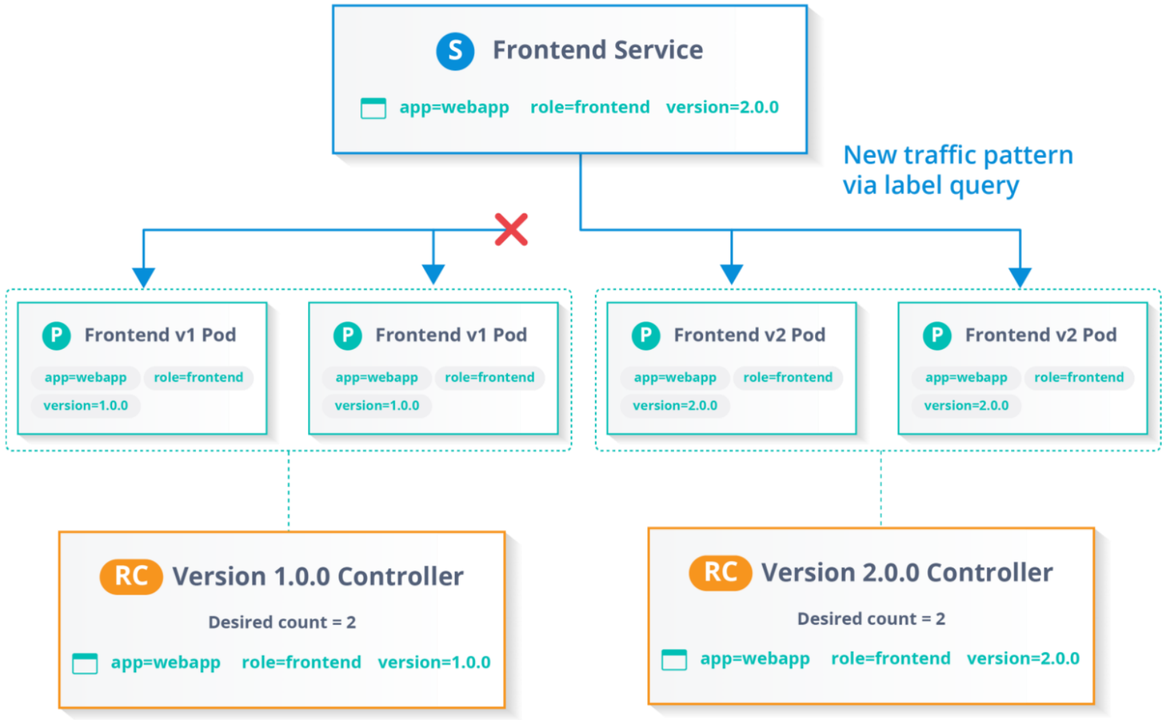
Zero-Downtime Application Maintenance

- With service, you can have the rolling deployment with inbound traffic being routed to the pods matching to the respective service



Zero-Downtime Application Maintenance

- So with service, you can create new pods with updated application,
- Do the traffic shift and applications start pointing to the list of new pods
- This, it avoided the downtime, intense preparation of any application maintenance activity.



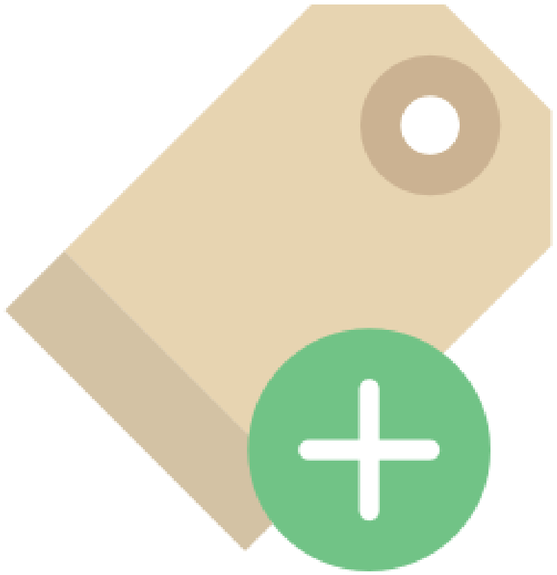
Demo 1: Creating our First Service

edureka!

Labels and Selectors

edureka!

What are Labels?



- Labels are key-value pair attached to objects like pods, replication controllers etc.
- They are used to identify attributes of objects and can be used to organize objects
- They can be created at the time of creation of resource
- Can also be added or modified after the resource is created.

Demo 2: Creating Labels

edureka!

Labels

- To add a label to an existing pod

```
$ kubectl labels pods <podName> label=value
```

- To verify if the label's been added

```
$ kubectl get pods –show-labels
```

What are Selectors?

Unlike PIDs or UID's, label selectors do not provide uniqueness to the resources they are attached to.

In fact, the correct usage for a label would be when the same label is applied to multiple resources.

Using a Label Selector, users can identify a set of resources/objects.

The resources can be selected based on two requirements:

Equality Based Requirements

Set Based Requirements

Types of Selectors

Equality Based Requirements

Set Based Requirements

- Filtering by Equality or Inequality means filtering based on key and values
- The Matching objects must specify all the specified label constraints.

Equal	Not Equal
=	!=

For Example:

```
Environment=developer
Tier!=frontend
```

edureka!

Types of Selectors

Equality Based Requirements

Set Based Requirements

- Filtering by Set means filtering based on key and a set of values
- There are three kinds of operator in set based requirements:

in **notin** **exists**

For Example:

environment in (production, qa)
tier notin (frontend, backend)
Partition
!partition

edureka!

DEMO: Labels and Selectors

edureka!

Labels and Selectors: Commands

- To use an Equity Selector

```
$ kubectl get pods -l labelName=value
```

- To use a Set-Based Selector

```
$ kubectl get pods -l 'labeName <operator> (value(s))'
```

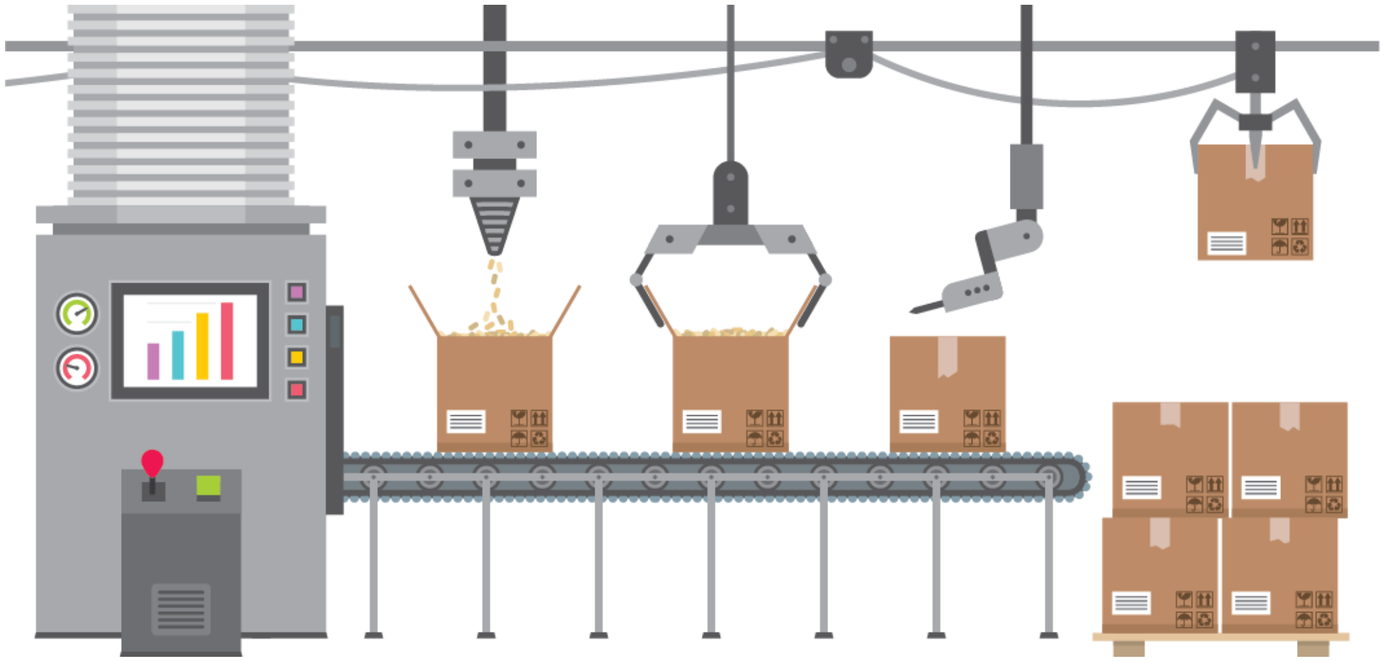
ReplicationController

edureka!

ReplicationController

- ReplicationController, Replica sets and Deployment provide moreover (not exactly same) same functionality

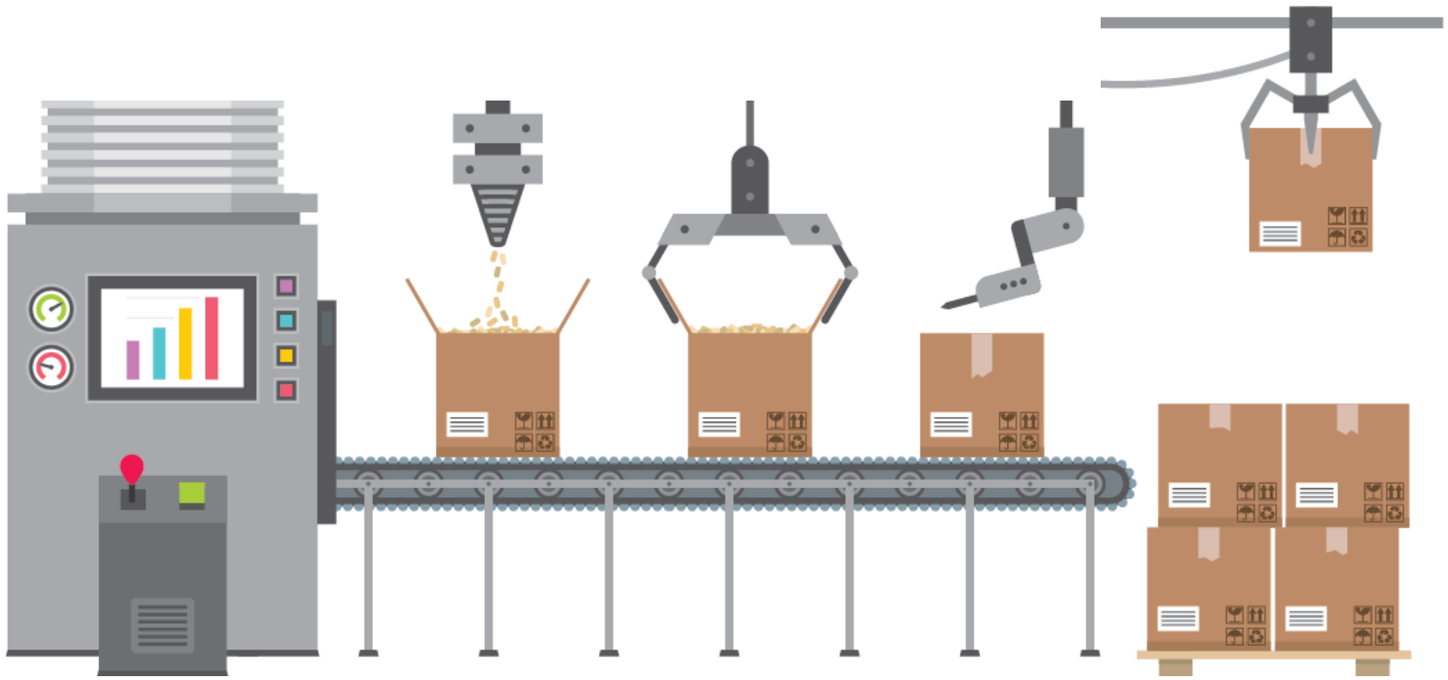
- Only supports equality-based selector requirements



edureka!

ReplicationController: Job

- It helps achieve the desired state by making sure that desired number of pods always exists
- If a pod crashes, the replication controller creates a new one
- Update or delete multiple pods with single command
- It helps to create, scale and maintain multiple pods as part of the desired state



edureka!

Replication Controller: YAML File

Create this YAML file to create a Replication Controller

```
spec:
  replicas: 3
```

Desired number of replicas required

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    app: nginx
  template:
    metadata:
      name: nginx
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

edureka!

Create Replication Controller

On your cluster, give following command

```
$ kubectl create -f replicationset.yaml
```

On your master run the following command to check active pods

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

ReplicaSet

edureka!

ReplicaSet

ReplicaSet are more powerful than ReplicationController

- Almost everything remains the same as ReplicationController
- It also ensures that a specified number of pod replicas are running at any given time
- Main difference comes for the support of selectors
- ReplicaSet supports the set-based selector whereas ReplicationController supports only equality based selector
- ReplicaSet helps to provide custom update orchestration or don’t require updates at all, unlike Deployment which supports declarative format.

edureka!

ReplicaSet: YAML File

Create this yaml file to create a ReplicaSet

```
spec:
  replicas: 3
```

Desired number of replicas required

```
apiVersion: apps/v1beta2
kind: ReplicaSet
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports :
        - containerPort: 80
```

edureka!

Create ReplicaSet

On your cluster, give following command

```
$ kubectl create -f replicaset.yaml
```

On your master run the following command to check active pods

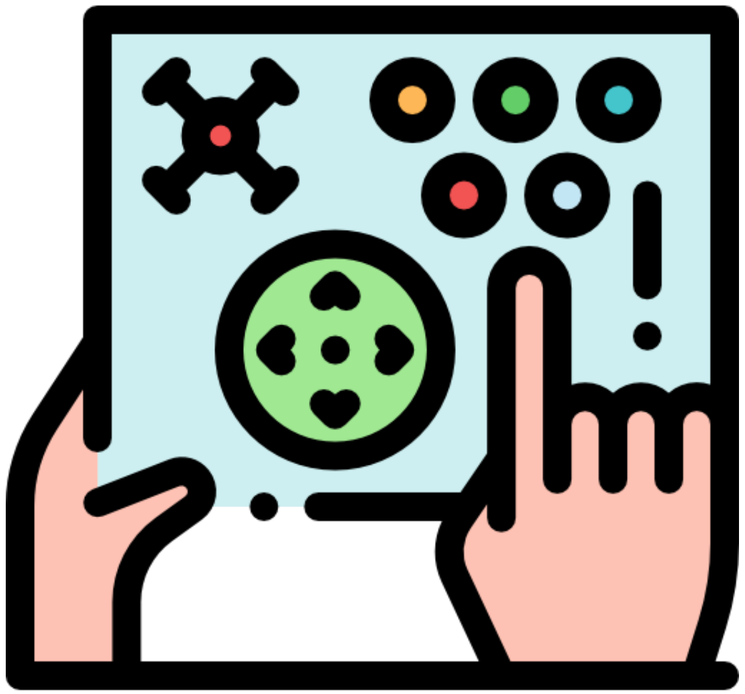
```
$ kubectl get pods -o wide
```

Deployment Controller

edureka!

Deployment Controller

- Deployment Controller are declarative in nature to provide the required information / updates to pods and ReplicaSets
- Deployment Controller changes the actual state of the defined desired state of controller object
- Introduction of deployment enable new use-cases for kubernetes cluster based solutions
- It is declarative in nature
- Deployment compliments pods and ReplicaSets



Deployment Controller – Key Use cases

Rollout of new replicaset

Rollback to earlier deployment version

Scale-up to handle more load

Define new state of the pods

Deployment Controller - Rolling Update Use case

- Create a deployment.yaml file
- Edit the file and define the older version of nginx
- image: nginx:1.7.9
- Sample file

```
$ cat deployment.yml
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
```

edureka!

Deployment Controller - Rolling Update Use case

- Create the deployment and make a note of deployed number of replicas and nginx version
- To Create the deployment execute the following

```
$kubectl create -f deployment.yml --record
```

- To check the deployment

```
$kubectl describe deployment deploymentName
```

Deployment Controller - Rolling Update Use case

Now, say that we want to roll out new version for nginx to 1.9.1 from 1.7.9
There are two approach to do it.

01

```
$ kubectl set image deployment/nginx-deployment nginx=nginx:1.9.1
```

02

```
//Edit the file and make the changes to nginx version
$kubectl edit deployment/nginx-deployment

//After saving the file, you can see the rollout happening using
$ kubectl rollout status deployment/nginx-deployment
```

edureka!

Deployment Controller - Rolling Update Use case

- To verify the updated image:

```
$kubectl describe deployment deploymentName
```

Deployment Controller - Rolling Update Use case

- See all the rollout history :

```
$ kubectl rollout history deployment/deploymentName
```

- To delete the pod :

```
$ kubectl delete deployment,services -l app=nginx
```

DEMO: Rolling Update

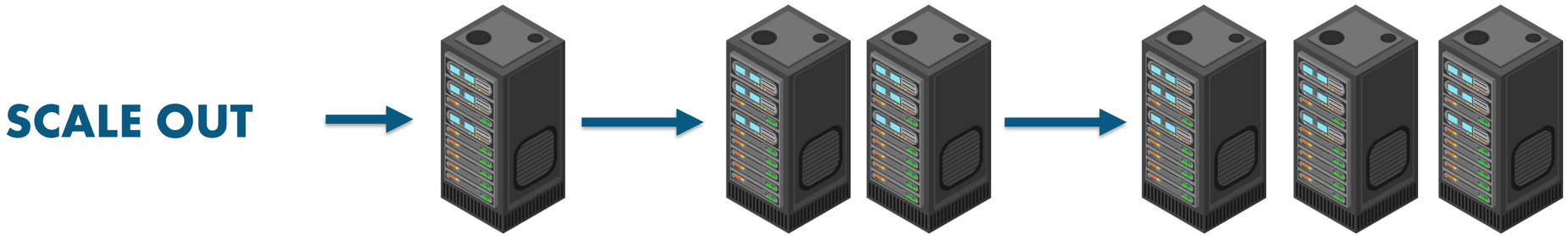
edureka!

Scaling out a deployment using replicas

edureka!

Scaling Out a Deployment using Replicas

- Kubernetes comes with lots of features which supports new age application (new age applications are applications which are more dynamic and agile in nature. Containerized applications)
- Autoscaling, scales-in and scales-out pods depending upon the workload resource utilization
- However, you can also do the same using manual process
    - Admin manually monitors the resource utilization and performance of the application.
        - Incase applications requires more pods to handle the load then he creates the new pods and once utilization drops back he can reduce the number of pods.



Scaling Out a Deployment using Replicas

- Below command shows the number of deployments

```
$ kubectl get deployments
```

- Say the current replicas of the deployment are 2

- To scale out the replicas from 2 to 10 use the scale command

```
$ kubectl scale deployment deploymentName --replicas=10
```

- In the same manner you can scale down replicas by simply giving a lower number than that of current replicas

DEMO: Scaling a Deployment

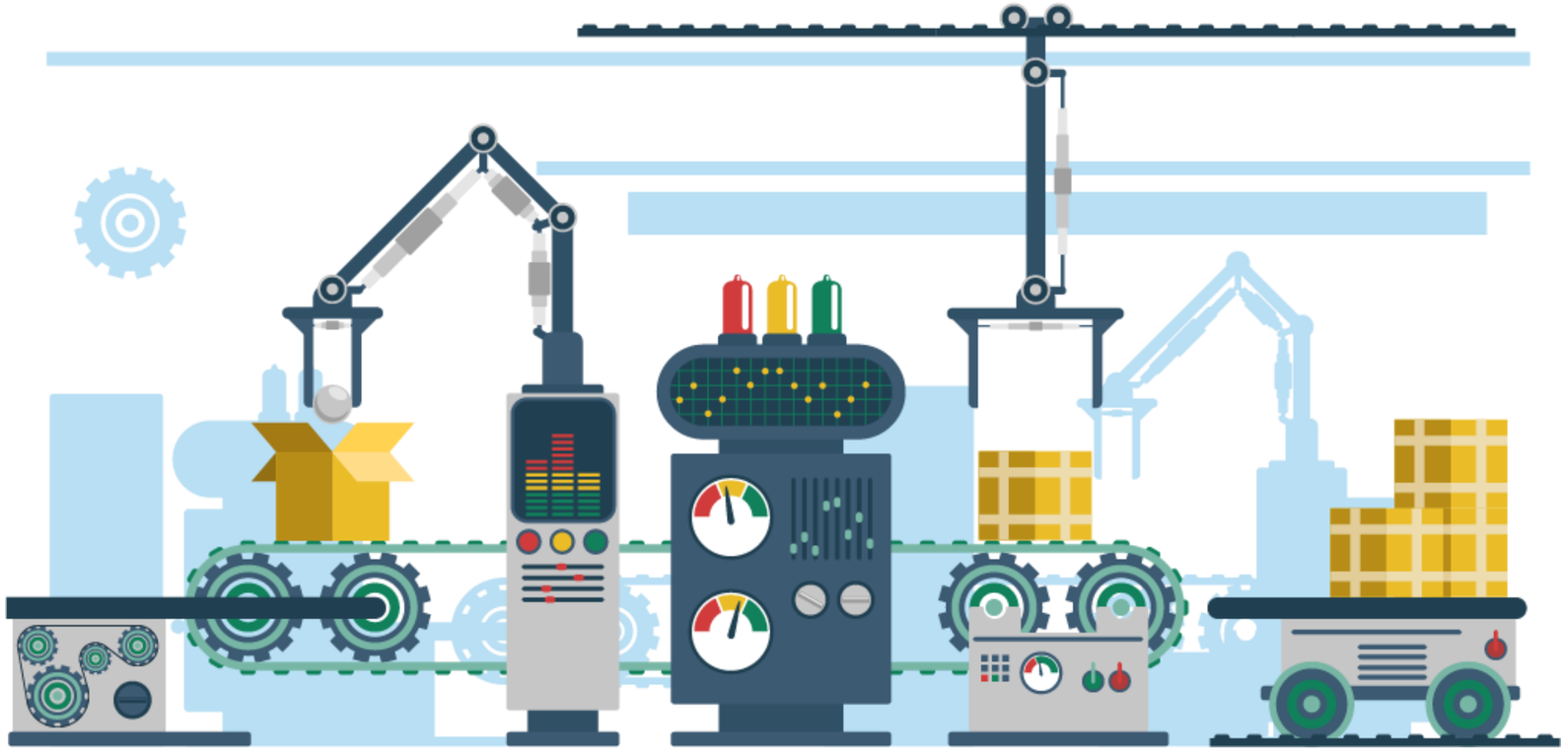
edureka!

Horizontal Pod Autoscaler

edureka!

Horizontal Pod Autoscaler

- Some applications require extra resources on the fly
- Autoscaling enables admins to handle resource demands, by setting an environment to auto-scale pods whenever the threshold limit is reached



Horizontal Pod Autoscaler

As mentioned earlier, there are many features which are must-to-have for all the new age applications which resource demand changes on the fly.

Auto-scaling is one of them, where admin team doesn't have to do the scaling out of pods using manual commands but they can set up the environment using which number of pods auto-scale when the threshold limit is touched.

```
//TO AUTOSCALE THE DEPLOYMENT
$ kubectl autoscale deployment deploymentName --min=3 --max=5 --cpu-percent=50
```

```
//TO CHECK THE STATUS OF DEPLOYMENTS
$ kubectl get deployments
```

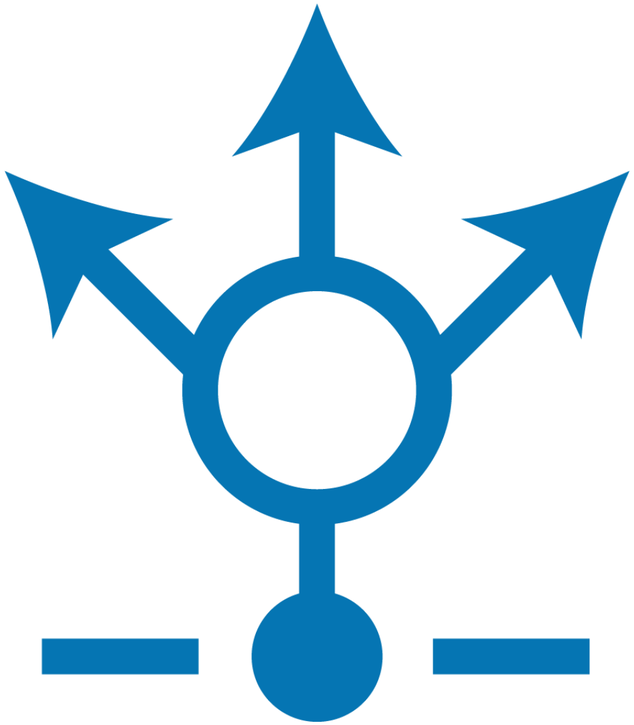
Now the moment CPU load reaches upto 50% number of pods would be increased.
And once the load is back to normal, autoscale will reduce the number of pods back 3

Load balancing

edureka!

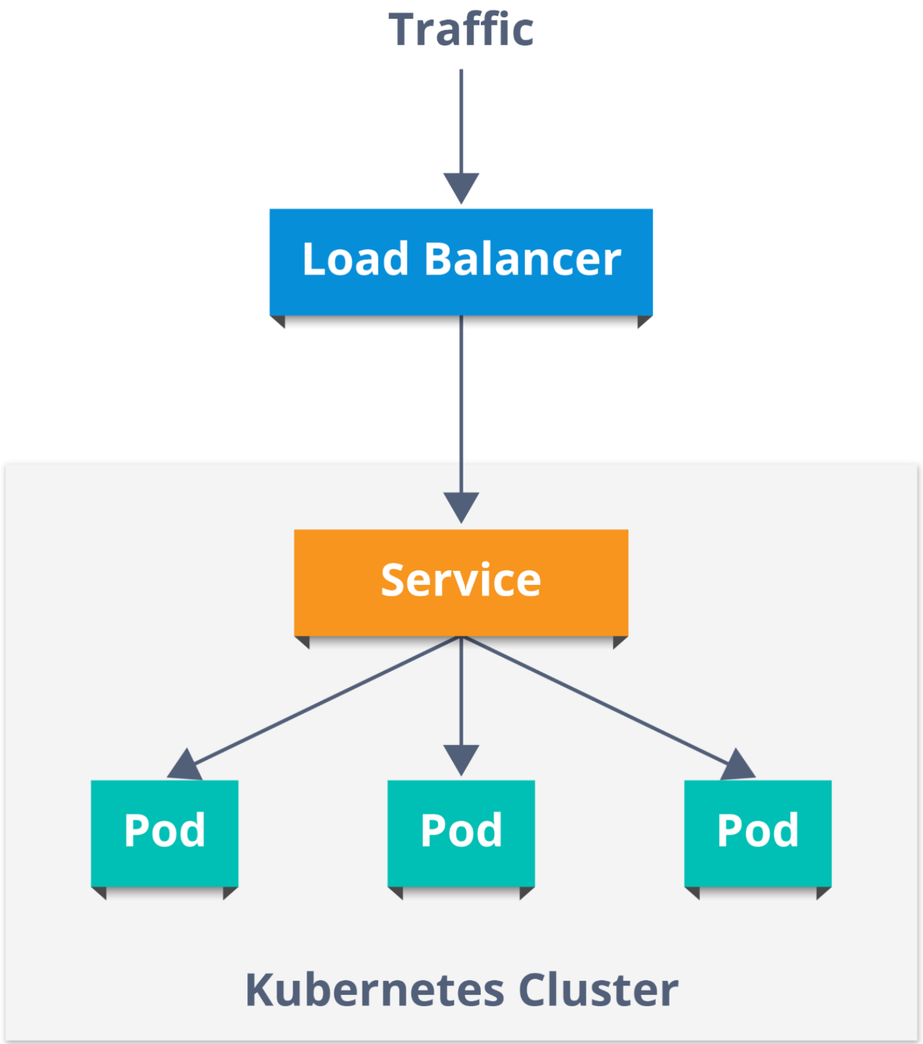
Load Balancing

- A load balancer is one of the most common and standard way to exposing service
- You can have External or Internal load balancer depending upon your working environment
    - External LB : Traffic from external load is directed to the backend pods



Load Balancing

- Load-balancer configuration also depends upon the cloud provider
    - Few cloud provider allow to configure your own IP
    - Incase IP is not specified then a temporary (ephemeral) IP is assigned.
- Admin team needs to understand the Load-balancer configuration depending upon the cloud provider
    - Common Cloud Providers are: GCP, Miscrosoft Azure, Amazon AWS



edureka!

What is an Ingress?

edureka!

Ingress

Ingress is the most powerful way of exposing service

01

02

Services and pods have IPs only routable within the cluster

03

It acts like an entrypoint to the kubernetes cluster

04

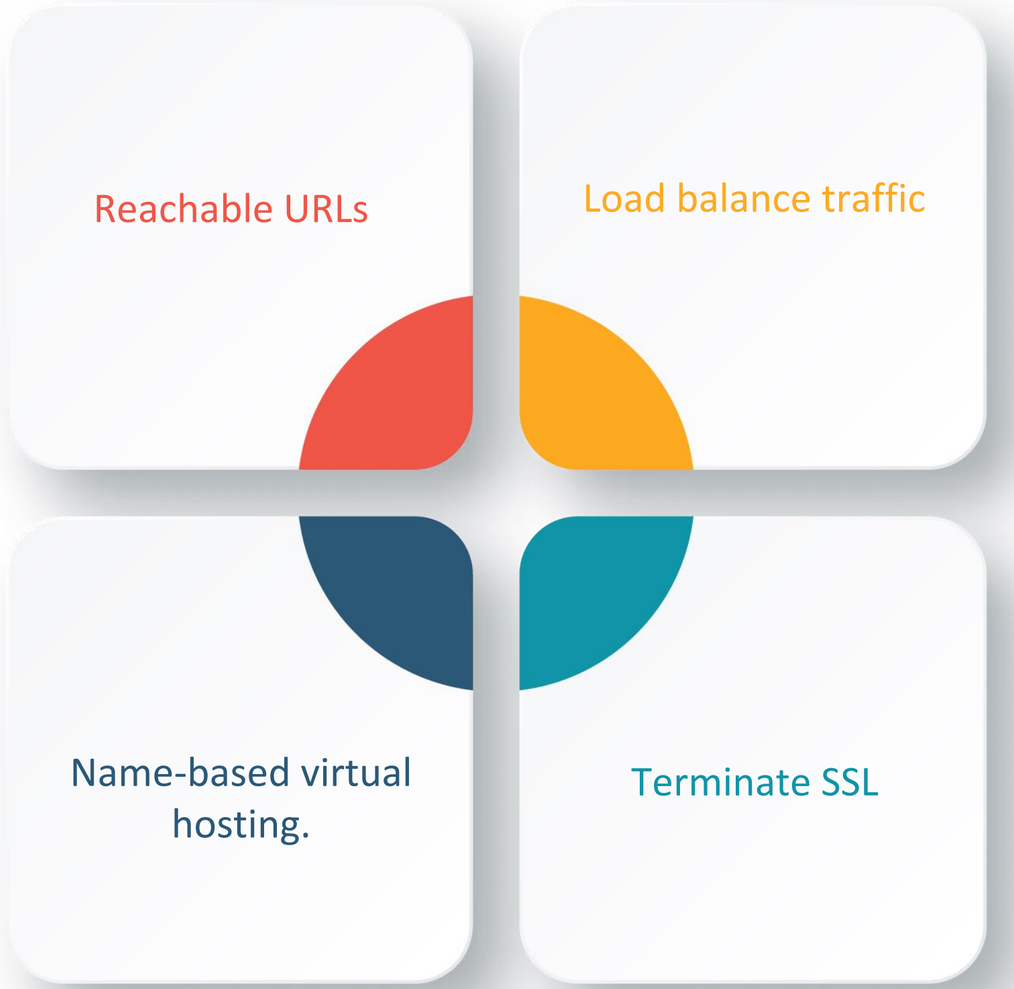
Ingress is an API object that manages external access to the services in a cluster, usually HTTP

05

It sits in front of multiple services and act as a ‘Smart router’

Ingress

It is Collection of rules that allow inbound connections which can be configured to give services externally-



Ingress

So, if we converge all the points then we define Ingress as a collection of rules that allow inbound connections which can be configured to give services externally through reachable URLs, load balance traffic, or by offering name-based virtual hosting.

edureka!

Ingress

Types of controller

Single Service Ingress

- There are existing approaches in kubernetes using which you can expose a single service. However, with ingress you do not have define any rules for it

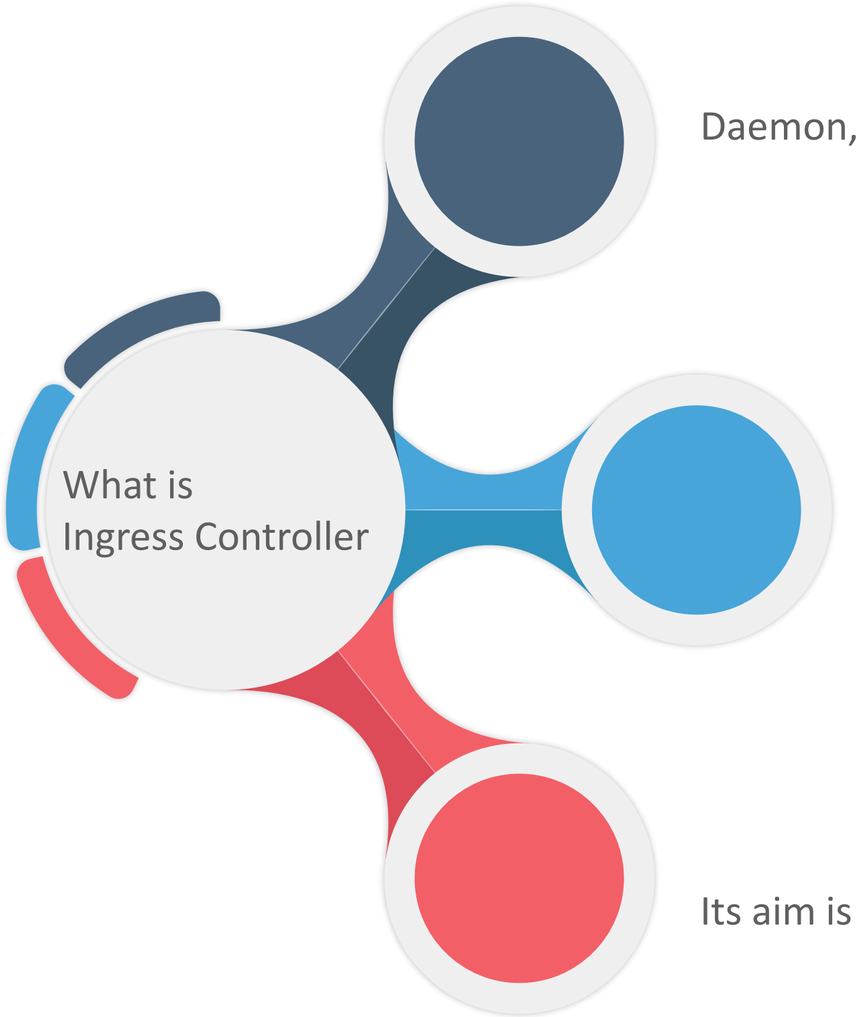
Name based virtual hosting

- Name-based virtual hosts use multiple host names for the same IP address.

Simple fanout

- As discussed earlier, that clusterIP are within kubernetes network, therefore, we need something in the front (edge) that accepts ingress traffic and proxy it to the right endpoints.
- These edge points are highly available load balancer.
- An Ingress allows you to keep the number of load balancers down to a minimum.

Ingress Controller



Daemon, deployed as a Kubernetes Pod

Monitors apiserver's /ingresses endpoint for updates

Its aim is to satisfy requests for Ingresses.

Ingress Controller

- There are different type of Ingress controller which does multiple things.
- At present, kubernetes supports GCE and nginx controller.
- There are other 3rd party providers which provides controller service for kubernetes.

Demo: Ingress

edureka!

Quiz

1. Kubernetes service provides abstraction layer which creates a logical set of Pods.
    a. True
    b. False

edureka!

Answers

A

1. Kubernetes service provides abstraction layer which creates a logical set of Pods.
   a. True
   b. False

Answer A: True

edureka!

Quiz

2. Labels are key-value pair attached to an object like pods, replication controllers which is responsible for proper functionality of kubernetes. Without label, pods cannot run.
    a. True
    b. False

Answers

A

2. Labels are key-value pair attached to an object like pods, replication controllers which is responsible for proper functionality of kubernetes. Without label, pods cannot run.

   a. True

   b. False

Answer B: False

edureka!

Quiz

3. Define selector and name types of Selectors.

Answers

A

3. Define selector and name types of Selectors.

Answer :
Labels do not provide uniqueness. Same object may carry the same label
With Selectors, user / application can pick the group of objects by it’s label
This is the core grouping mechanism in kubernetes cluster.

Currently there are two types of selector supported by kubernetes API
Equity based selector
Set-based selector

Quiz

4. Tell the type of selector in below example :

```
$ kubectl get pods --selector owner=edureka

NAME      READY    STATUS      RESTARTS   AGE

pi-l44jr      0/1         Completed      0           1h
```

Answers

A

4. Tell the type of selector in below example :

$ kubectl get pods --selector owner=edureka

NAME READY STATUS RESTARTS AGE

pi-l44jr 0/1 Completed 0 1h

Answer : Equity based selector

edureka!

Quiz

5. ReplicationController was the oldest one. And It supports all type of selector requirements.

   a. True

   b. False

Answers

A

5. ReplicationController was the oldest one. And It supports all type of selector requirements.

   a. True

   b. False

Answer B: False

Quiz

6. Tell any one difference between Replication controller and Replicaset.

Answers

A

6. Tell any one difference between Replication controller and Replicaset.

Answer : ReplicaSet supports the set-based selector whereas Replicationcontroller supports only equality based selector.

edureka!

Quiz

7. Provide any two use-case of Deployment controller

Answers

A

7. Provide any two use-case of Deployment controller

Answer :
Rollout of new replicaset
Rollback to earlier deployment version
Scale-up to handle more load

edureka!

Quiz

8. Tell any one job of replication controller.

Answers

A

8. Tell any one job of replication controller.

Answer :
It helps to achieve the desired state by making sure that number of pods as desired always exists.
If a pod crash, then replication controller creates a new one.
It helps to create and maintain multiple pods as part of the desired state.

Quiz

9. Autoscaling can be done only automatically, you cannot scaleout or scale-in manually.

   a. True

   b. False

Answers

A

9. Autoscaling can be done only automatically, you cannot scaleout or scale-in manually.

   a. True

   b. False

Answer B: False

Quiz

10. Name the types of Ingress Controller ?

Answers

A

10. Name the types of Ingress Controller ?

Answer :

Single Service Ingress: There are existing approach in kubernetes using which you can expose a single service. However, with Ingress also you can do by not defining any rules for it.

Simple fanout: We discussed multiple times, that clusterIP are within kubernetes network, therefore, we need something in the front (edge) that accepts ingress traffic and proxy it to the right endpoints.

These edge points are highly available load balancer.

An Ingress allows you to keep the number of load balancers down to a minimum.

Name based virtual hosting: Name-based virtual hosts use multiple host names for the same IP address.

Summary

- In this module, you should have learnt:
  - Service
  - Labels and Selectors
  - Replication Controller & Replica Set
  - Deployment Controller
  - Scaling out a deployment using replicas
  - Rolling Update
  - Horizontal pod autoscaler
  - Load balancing
  - Ingress and its types

edureka!

edureka!

edureka!

Thank You

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