Get the best out of Live Sessions

HOW?

Check your Internet Connection

Log in 10 mins before, and check your internet connection to avoid any network issues during the LIVE session.

Speak with the Instructor

By default, you will be on mute to avoid any background noise. However, if required you will be **unmuted by instructor**.

Clear Your Doubts

Feel free to clear your doubts. Use the "**Questions**" tab on your webinar tool to interact with the instructor at any point during the class.

Let us know if you liked our content

Please share feedback after each class. It will help us to enhance your learning experience.

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

edureka!

DevOps Certification Training

COURSE OUTLINE
MODULE 09

1. Introduction to DevOps
2. Version Control with Git
3. Git and Jenkins
4. Continuous Integration with Jenkins
5. Configuration Management using Ansible
6. Containerization using Docker Part - I
7. Containerization using Docker Part - II
8. Container Orchestration Using Kubernetes Part-I
9. Container Orchestration Using Kubernetes Part-II
10. Monitoring Using Prometheus and Grafana
11. Provisioning Infrastructure using Terraform Part - I
12. Provisioning Infrastructure using Terraform Part - II

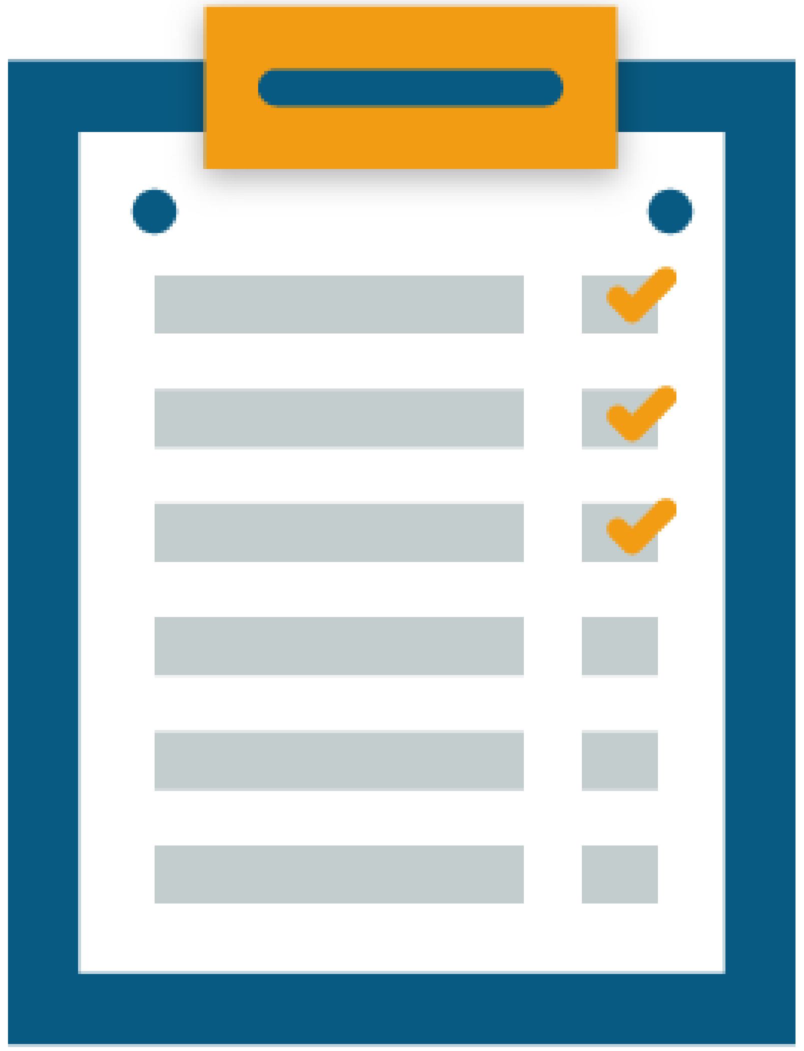
Module 9 – Container Orchestration Using Kubernetes Part-II

Copyright © edureka and/or its affiliates. All rights reserved.

Topics

Following are the topics covered in this module:

- Services
- Persistent Storage in Kubernetes
- Primitives for PersistentVolumeClaims
- Secrets and ConfigMaps
- Headless Services
- StatefulSets
- Helm Charts



edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Objectives

After completing this module, you should be able to:

- Deploy different Kubernetes Services
- Utilize Volumes to store Persistent Data
- Create Persistent Volume Claims to attach volumes to Pods
- Understand Persistent Volume Claims Primitives
- Use Headless Services in Stateful Sets
- Deploy Helm Charts

Copyright © edureka and/or its affiliates. All rights reserved.

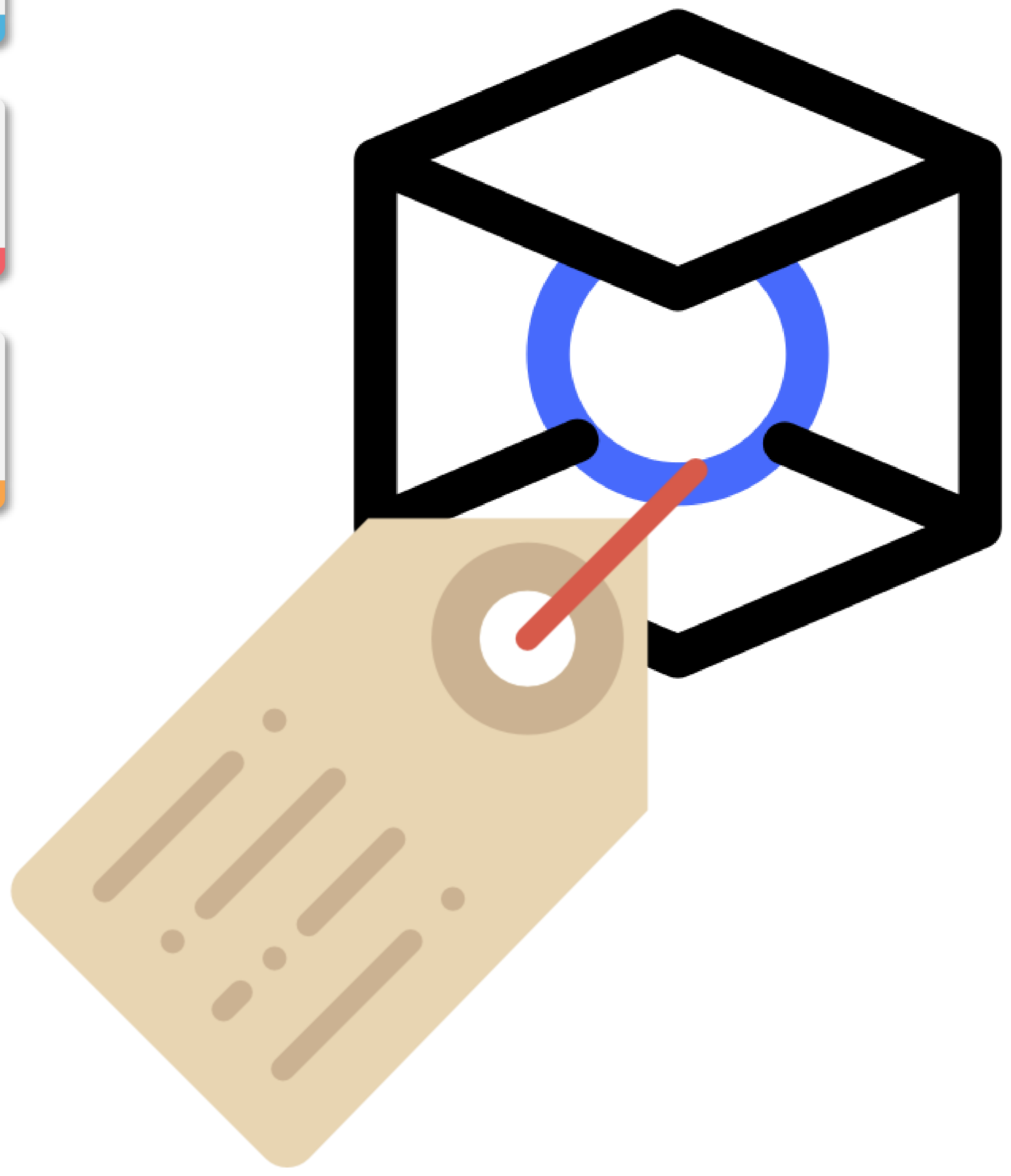
Labels, Selectors and Annotations

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Labels

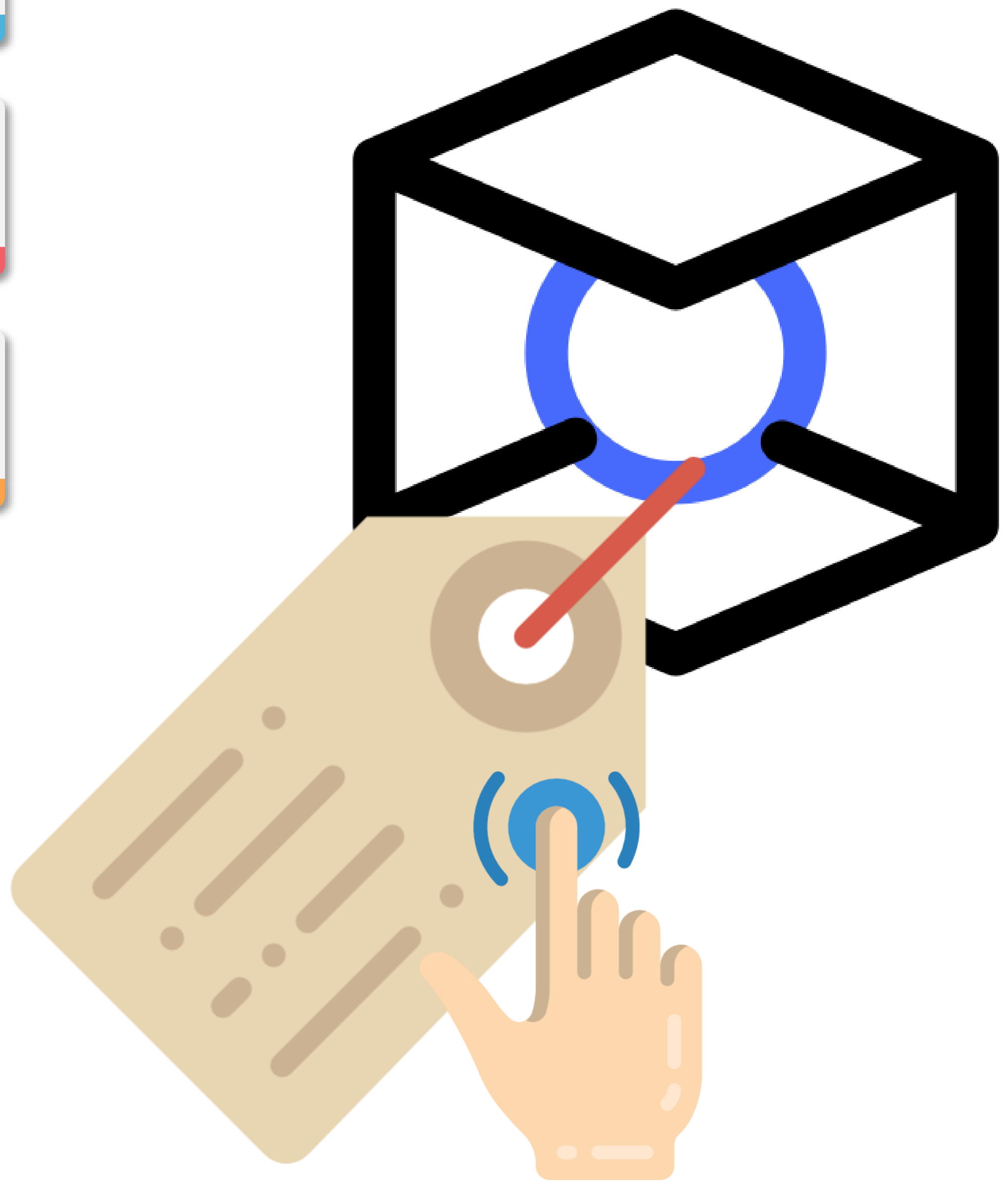
- Key-value pairs attached to Kubernetes objects such as nodes, pods, and controllers
- Used to organise objects by identifying similar attributes
- Added at the time of object creation
- Can be added/modified later in the object's lifecycle



Copyright © edureka and/or its affiliates. All rights reserved.

Selectors

- Core grouping mechanism in Kubernetes
- Users/applications can pick a group of objects by their labels using selectors
- Multiple labels can be used with selectors using a comma separator, which acts as a logical AND (&&) operator
- Different APIs have context-based implementation for empty selectors



Copyright © edureka and/or its affiliates. All rights reserved.

Types of Selectors

Currently, there are two types of selectors supported by Kubernetes API:

1 Equality-Based Selector

2 Set-Based Selector

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Equality-based Selector

- Allows filtering by key-value matching
- All the specified labels must be satisfied by the matching object
- Two kinds of operators are used:
  - == (=) : represents equality
  - != : represents inequality

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Equality-based Selector (Contd.)

- Eg : env = production
  - This selects objects having "production" as the value for the key "env"
  - env != production inversely selects all the objects not having "production" as their value for "env" key

From our example:

```
$ kubectl get pods --selector owner=edureka
NAME        READY     STATUS       RESTARTS    AGE
pi-l44jr     0/1     Completed        0        1h
```

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Set-based Selectors

- Allows filtering the keys based on a set of values
- Supports three kinds of operators
    - in
    - notin
    - exists
- Eg: team in (HR, training)
    - Finds objects having “HR” or “training” values for the key “team”

Copyright © edureka and/or its affiliates. All rights reserved.

Annotations

- Attaches non-identifying metadata to objects
- The metadata can be utilised by various tools and libraries
- Uses key-value pair with valid segments i.e. an optional prefix and name separated by / (slash)
- Metadata can have characters not permitted by labels

Copyright © edureka and/or its affiliates. All rights reserved.

Annotations (Contd.)

Annotations can store information such as:

- Build/ release/ image information
- Logging and monitoring pointers
- Analytics or audit repositories
- Library/ tool information
- Information for external ecosystems
- Custom Declarative Configuration layer fields

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

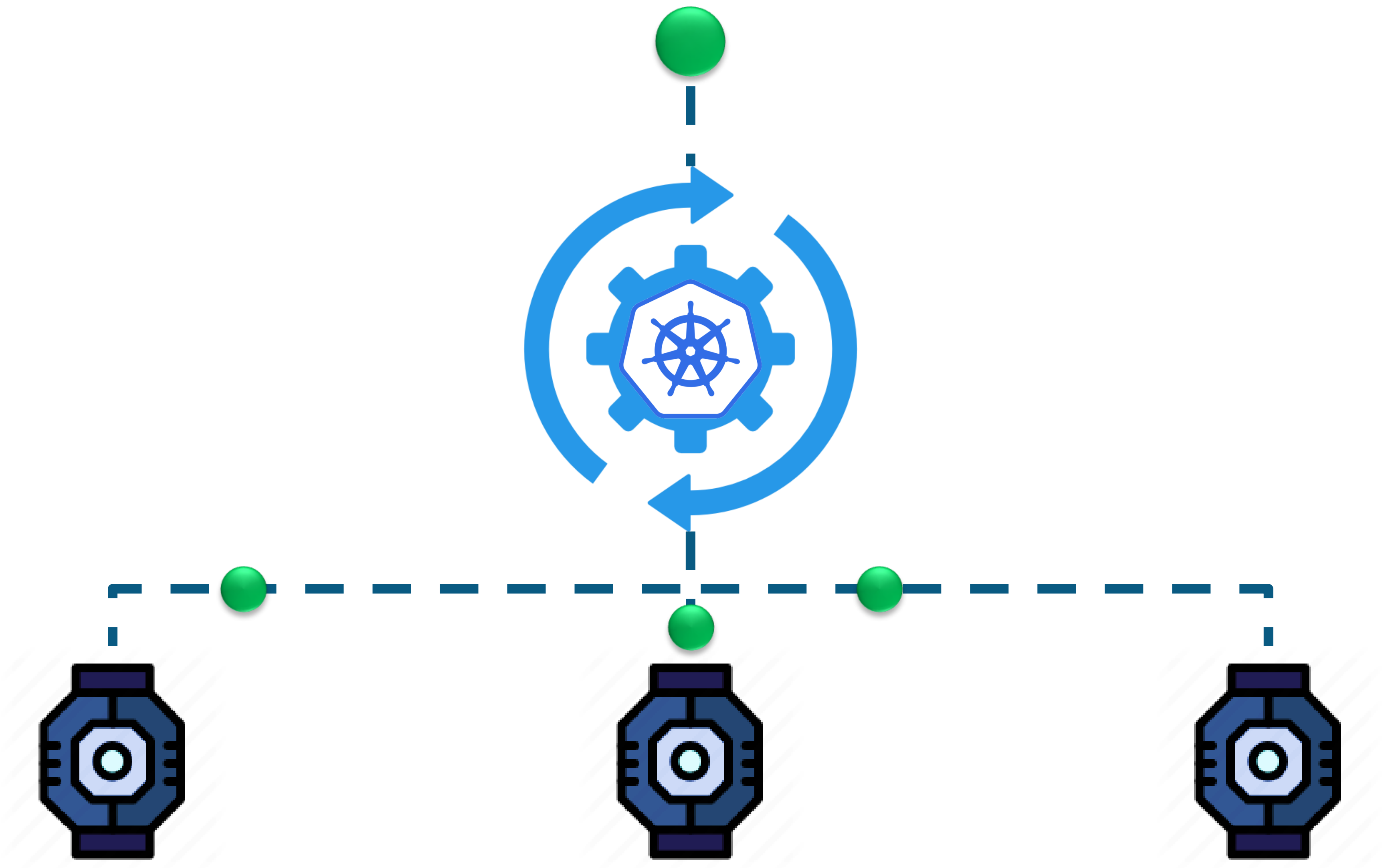
Services

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Services

- Services load-balance the requests in and out of the application
- It acts as a layer of abstraction between the containers and the network



edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Services

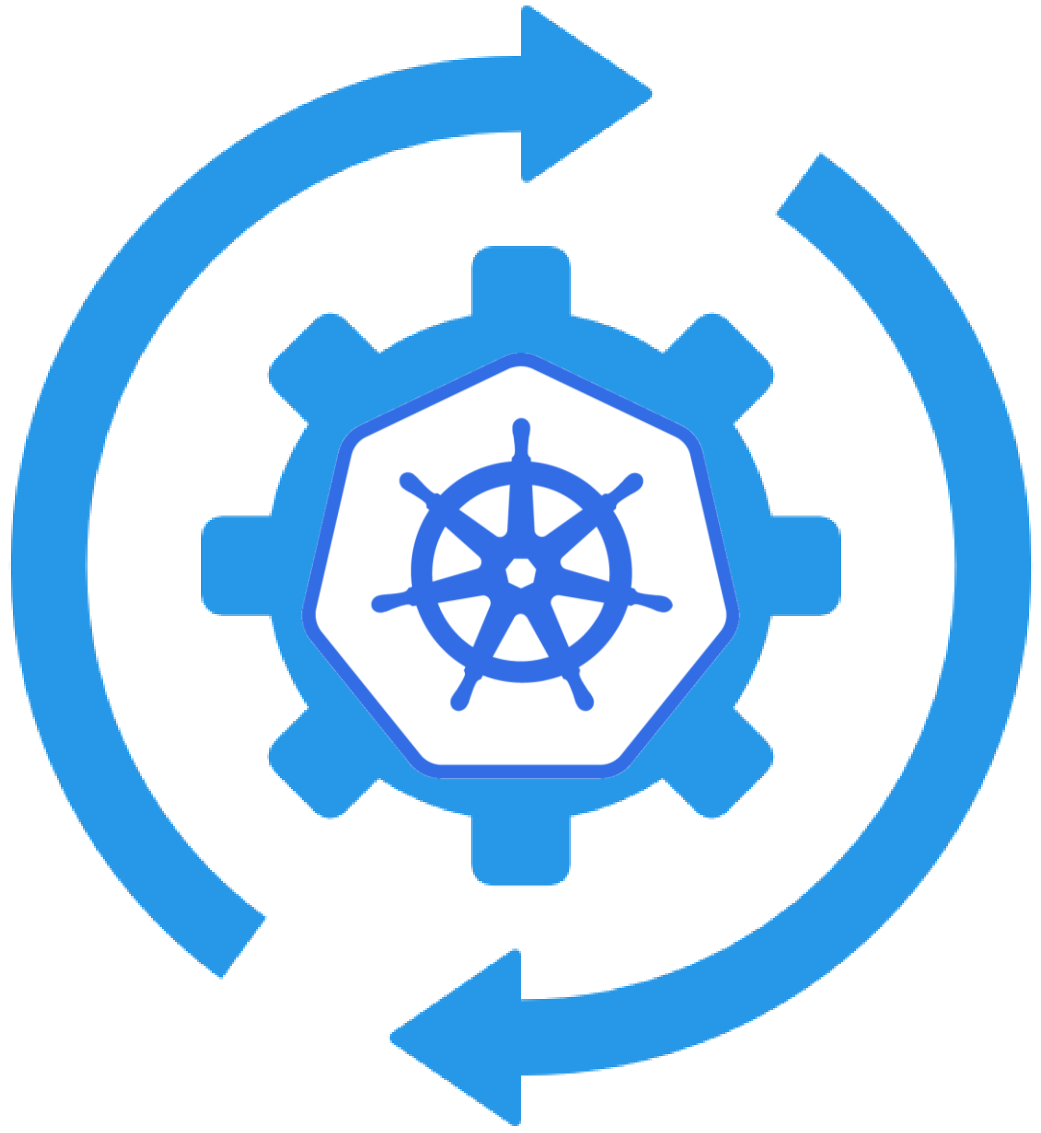
01 Services act as a single constant point of entry to a group of pods lending similar service

02 It solves pod related issues such as pods being ephemeral and horizontal scaling

03 Each service has an immutable IP address and port that will never change throughout the service's lifecycle

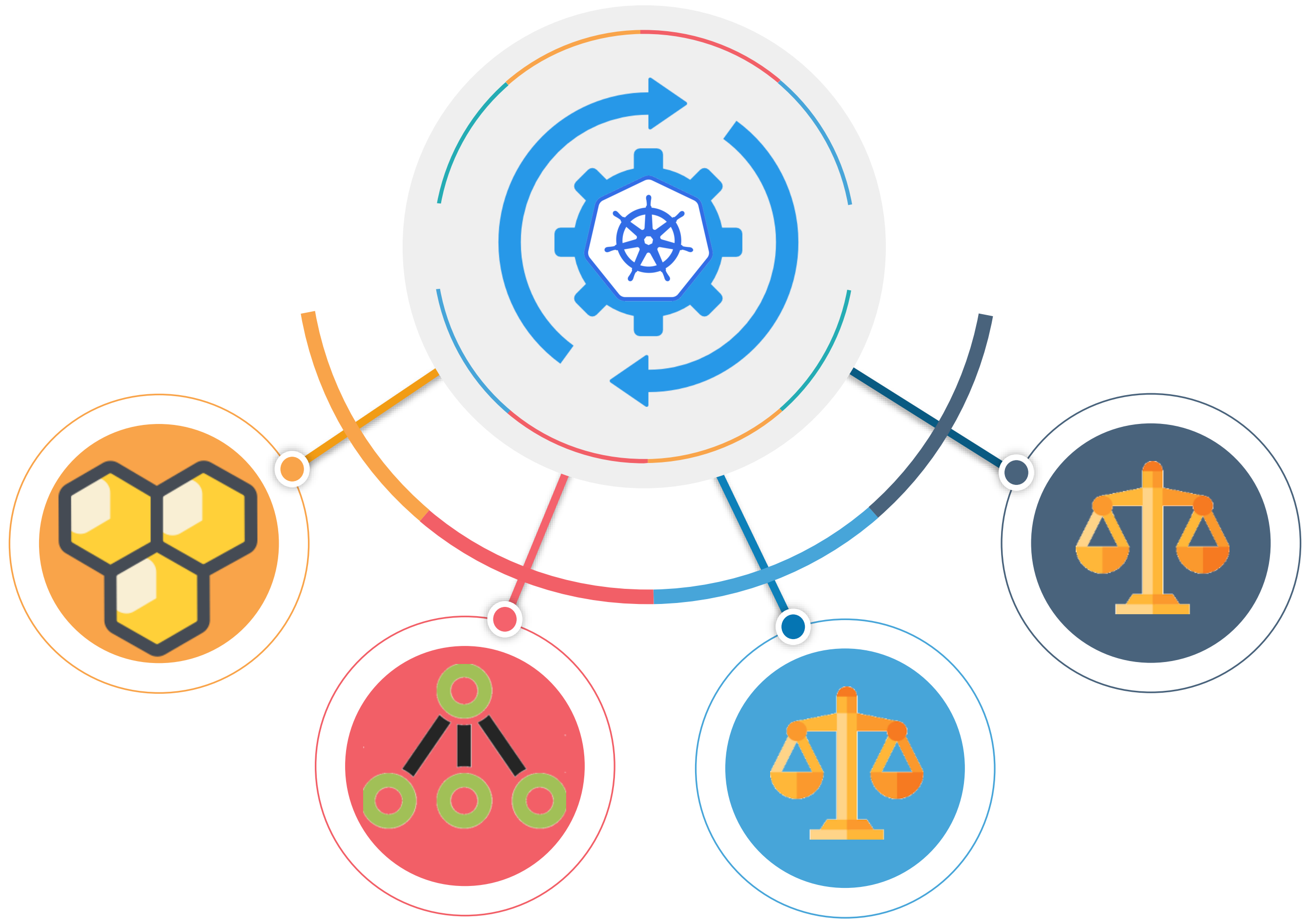
04 Clients directly communicate with the service. This allows pods to be scheduled anywhere

05 It gives resurrecting pods more consistency than a deployment



Copyright © edureka and/or its affiliates. All rights reserved.

Types of Services



ClusterIP

- Exposes the application inside the cluster itself
- Application cannot be accessed outside of the cluster

ExternalName

- A Unique service where selectors, ports and endpoints are not defined
- Returns an alias to an external service residing outside of the cluster

NodePort

- Exposes the application to a static port on the node
- Can be accessed outside of the cluster using the NodeIP and NodePort

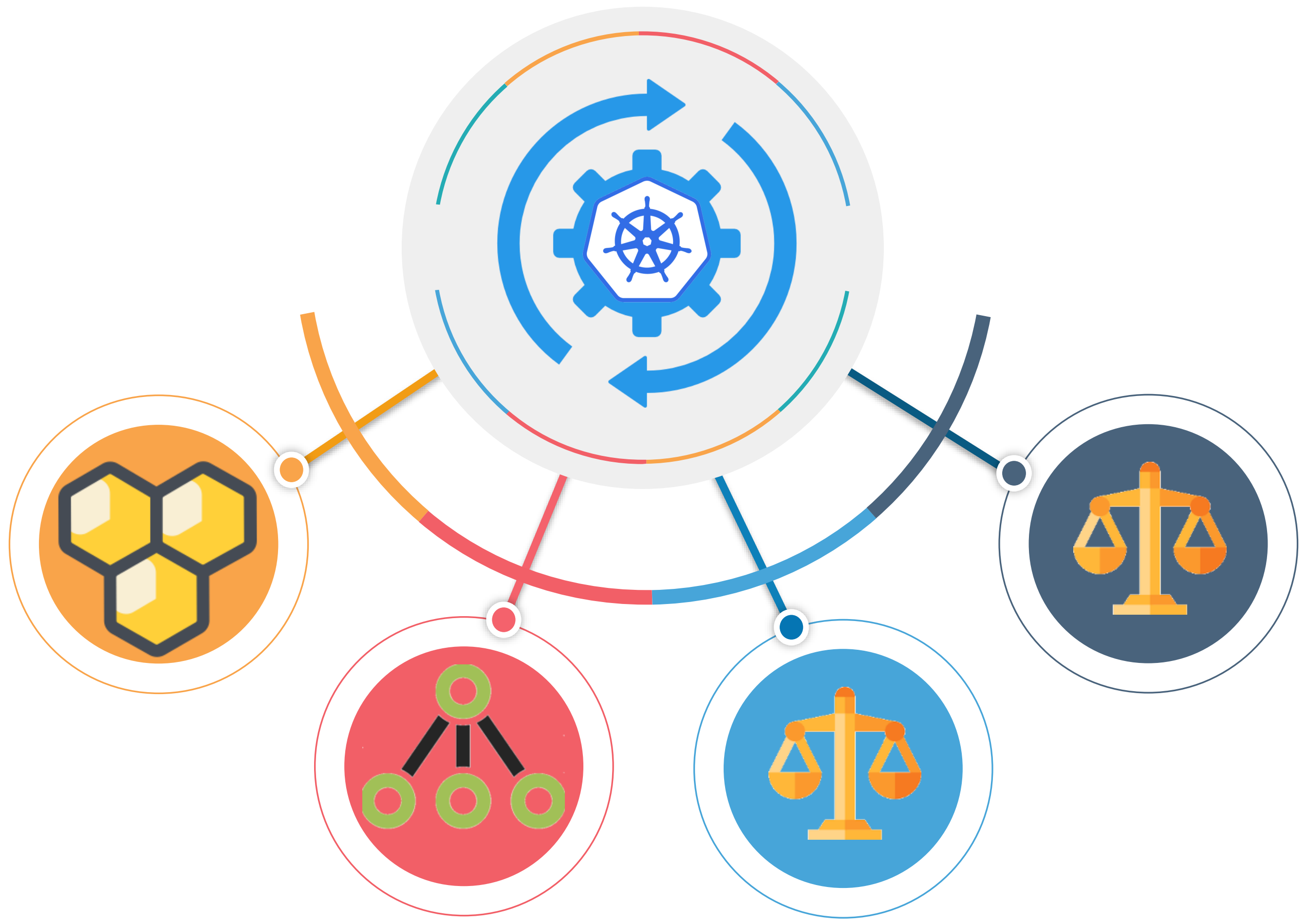
LoadBalancer

- Uses external cloud provided load balancers
- It automatically creates NodePort and ClusterIP

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Types of Services (Contd.)



- Services are superset of each other
- NodePort is superset of ClusterIP and Loadbalancer is superset of NodePort

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Demo: Services

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Persistent Volumes

Copyright © edureka and/or its affiliates. All rights reserved.

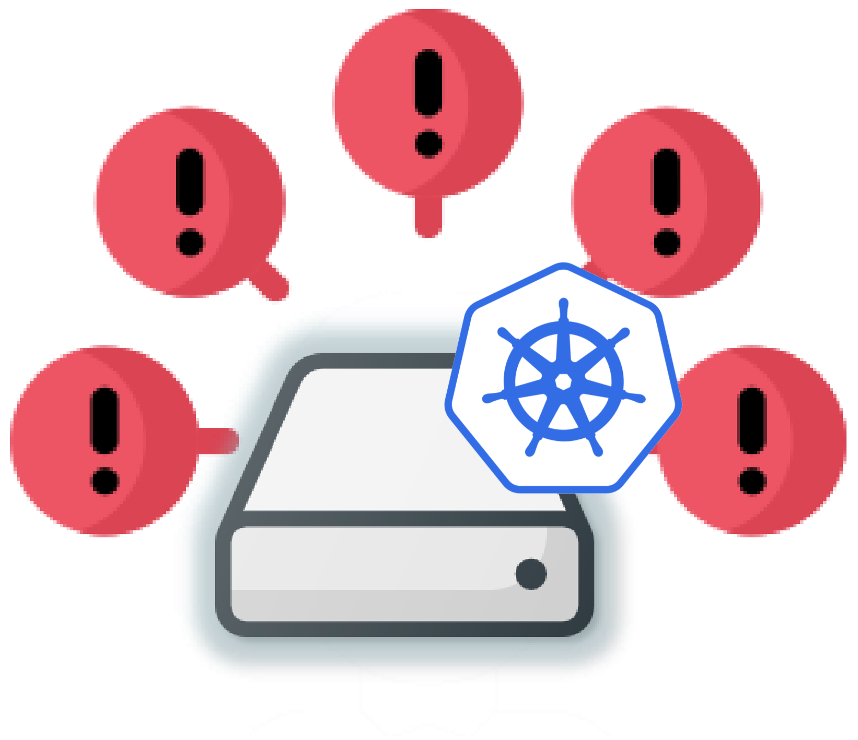
Volumes: The Problem

01 Normal volumes provide persistent storage but the layer of abstraction is lost

02 Difficult to manage tightly coupled applications which share volume storage

03 Volumes are bound to pods lifecycle and are destroyed once the pod is terminated

04 Unable to provide persistent storage to stateful/ distributed application across the cluster



Copyright © edureka and/or its affiliates. All rights reserved.

Persistent Storage in Kubernetes

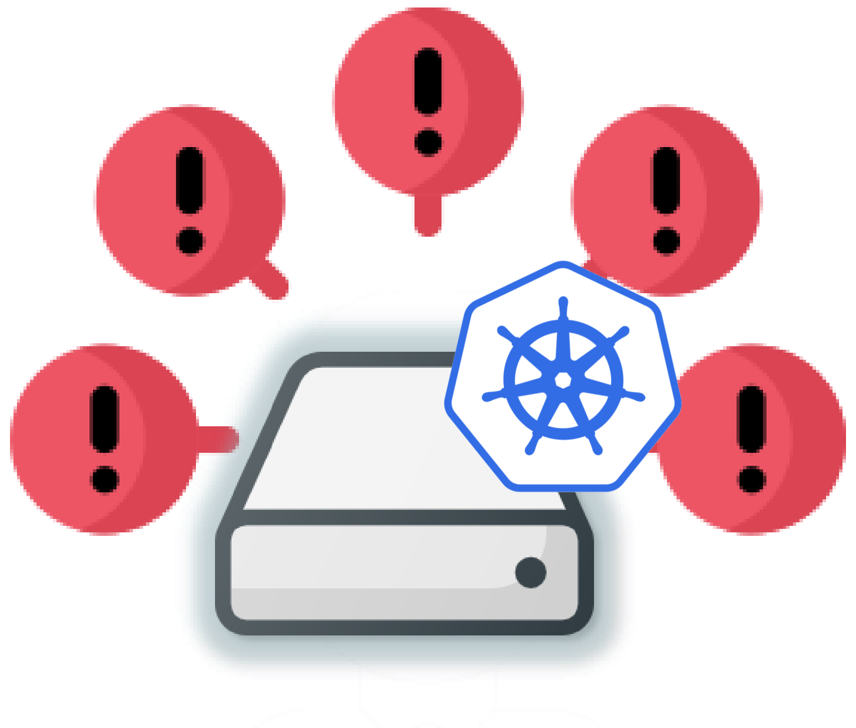
01 The layer of abstraction can be bought back using Persistent Volumes and Persistent Volume Claims

02 Persistent Volumes can be utilised as a cluster resource

03 Kubernetes binds the Persistent Volume Claim to the best possible Persistent Volume

04 The Volume Claim connects the applications with the persistent storage

05 Persistent Volumes provide the stability required for stateful and distributed applications

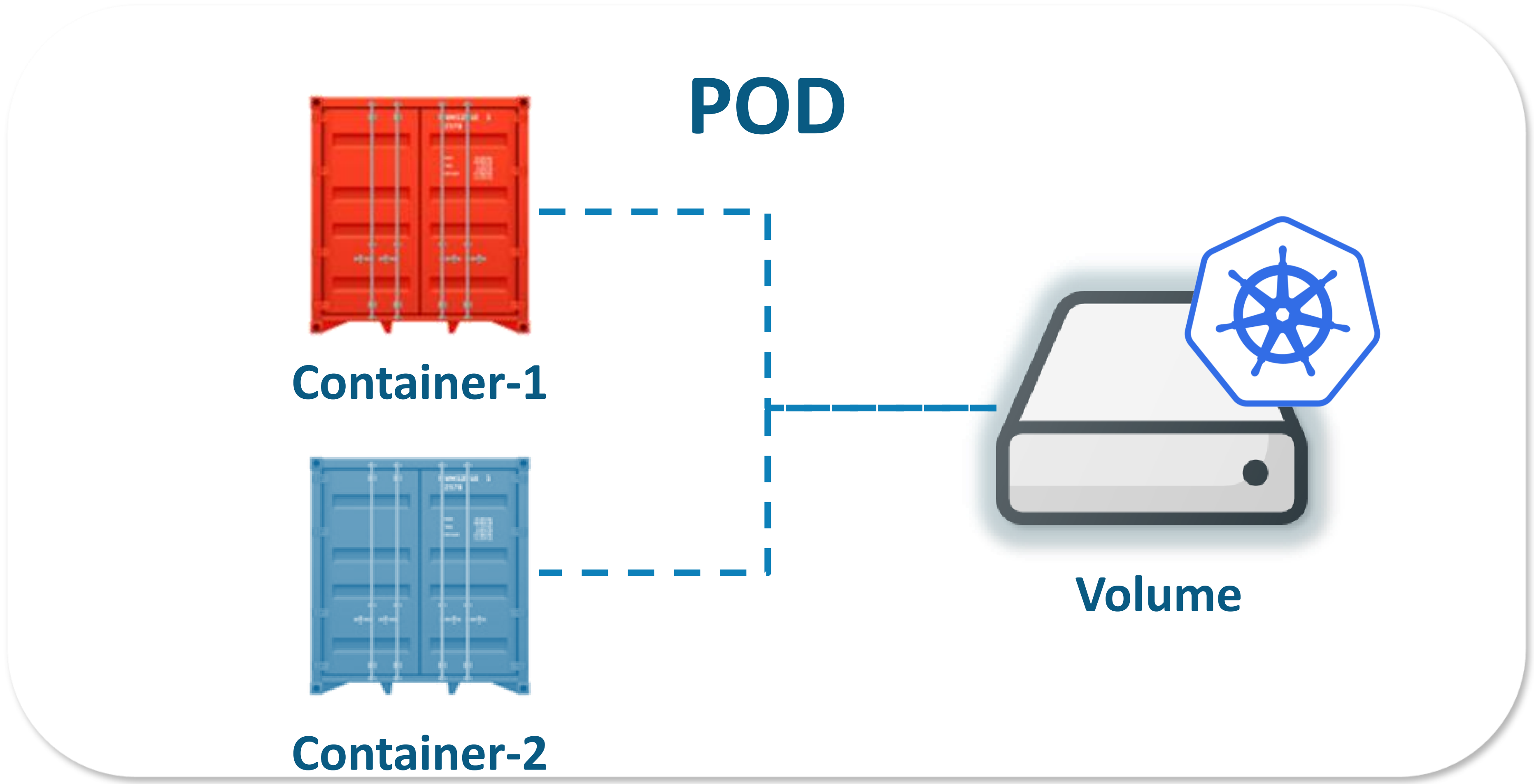


edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Persistent Storage in Kubernetes

- Kubernetes has the provision of storing Persistent Data in Volumes
- This maintains consistency for stateful applications



edureka!
Copyright © edureka and/or its affiliates. All rights reserved.

Persistent Volumes and Persistent Volume Claims

01 Persistent Volume once created, acts as a resource in the cluster

02 Persistent Volume Claims are requests for the created resources(PVs)

03 PVCs ensure that the application will be able to use the required storage

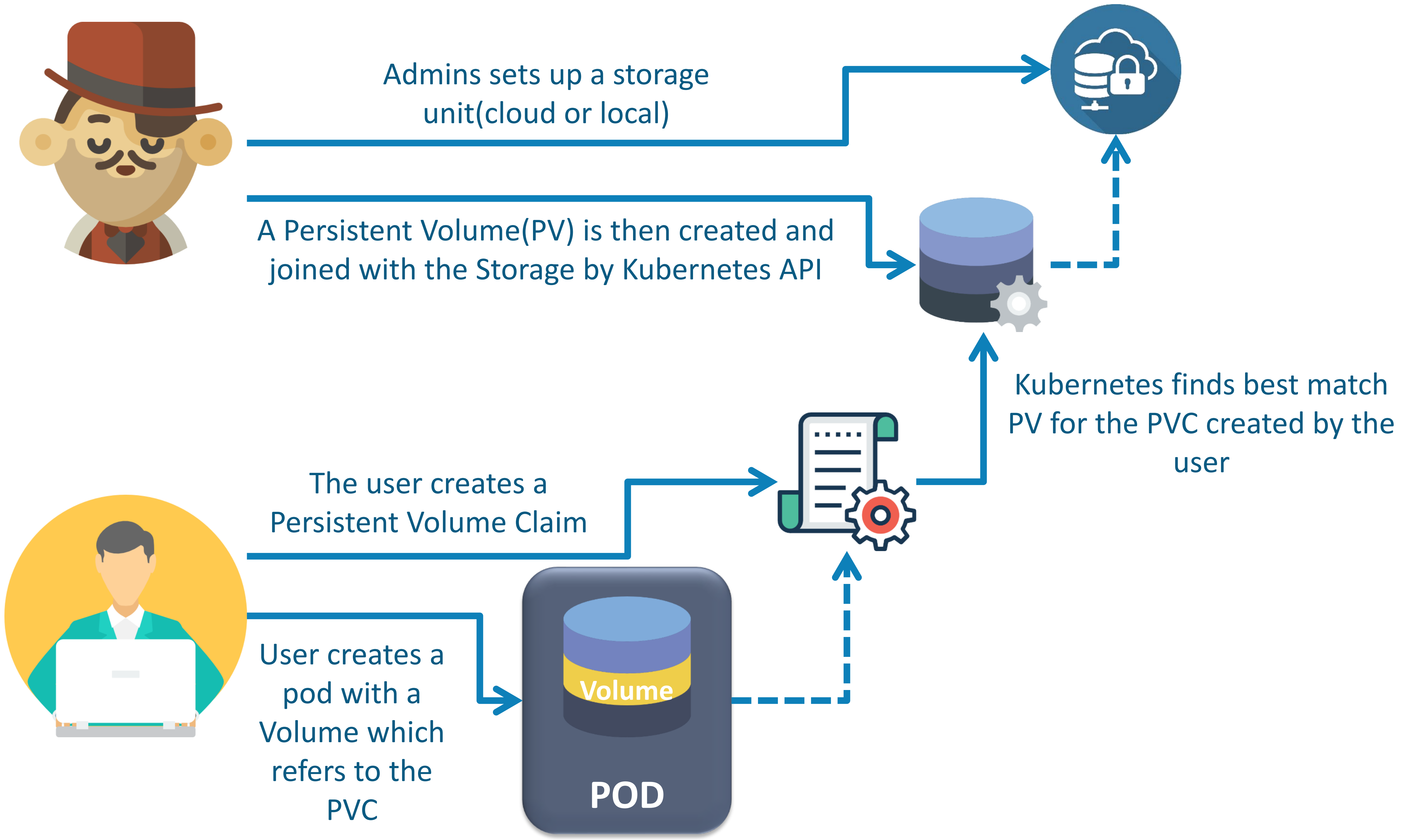
04 PVs and PVCs abstract the storage implementation from the application

05 The application needs to specify certain parameters for its storage requirement to the PVC

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Persistent Volumes and Persistent Volume Claims


Admins sets up a storage unit(cloud or local)
A Persistent Volume(PV) is then created and joined with the Storage by Kubernetes API
Kubernetes finds best match PV for the PVC created by the user
The user creates a Persistent Volume Claim
User creates a pod with a Volume which refers to the PVC
Volume
POD

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Access Modes for Persistent Volumes

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Access Modes for Persistent Volumes

There are three types of Access Modes for Persistent Volumes:

RWO

1

Read
Write
Once

ROX

2

Read
Only
Many

RWX

3

Read
Write
Many

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Access Modes for Persistent Volumes (Contd.)

RWO

1

The Volume can only be mounted by a single node in read-write mode

ROX

2

The Volume can be mounted in read-only mode by many nodes

RWX

3

The Volume can be mounted in the read-write mode by many nodes

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Persistent Volume Claims Primitives

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

PVC Primitives

Access Modes	PVCs have provision to ask for volumes with specific access modes
Volume Modes	PVC can define if it wants to use the volume as a FileSystem or a block device
Resources	PVCs can request for certain quantity of resources(storage)
Selector	Selectors can be defined based on which a PV is attached to the PVC
Class	If a PVC wants to bind directly through a Storage Class, the class attribute is used

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Demo: Persistent Volumes and Persistent Volume Claims

Copyright © edureka and/or its affiliates. All rights reserved.

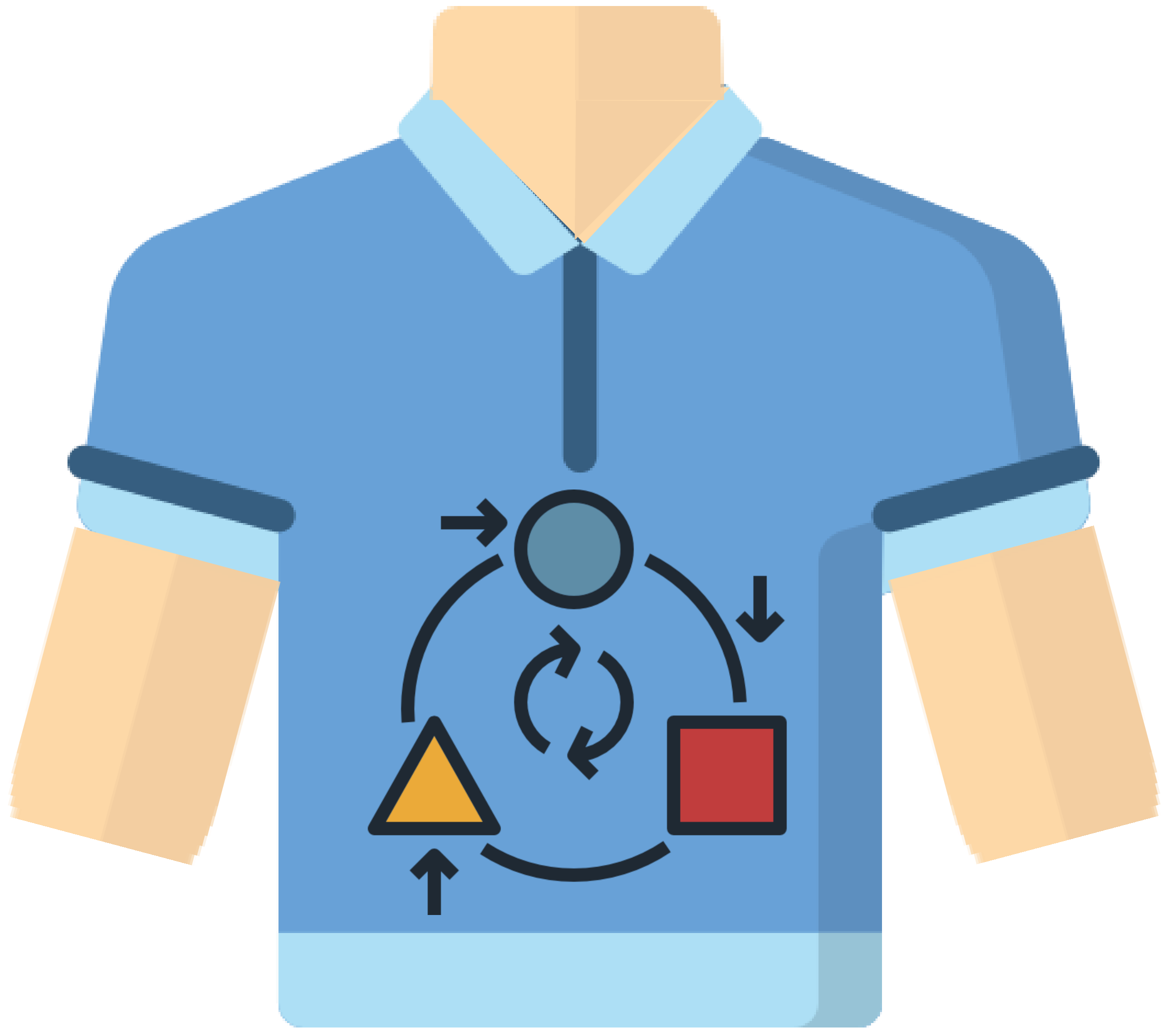
Headless Service

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Headless Service

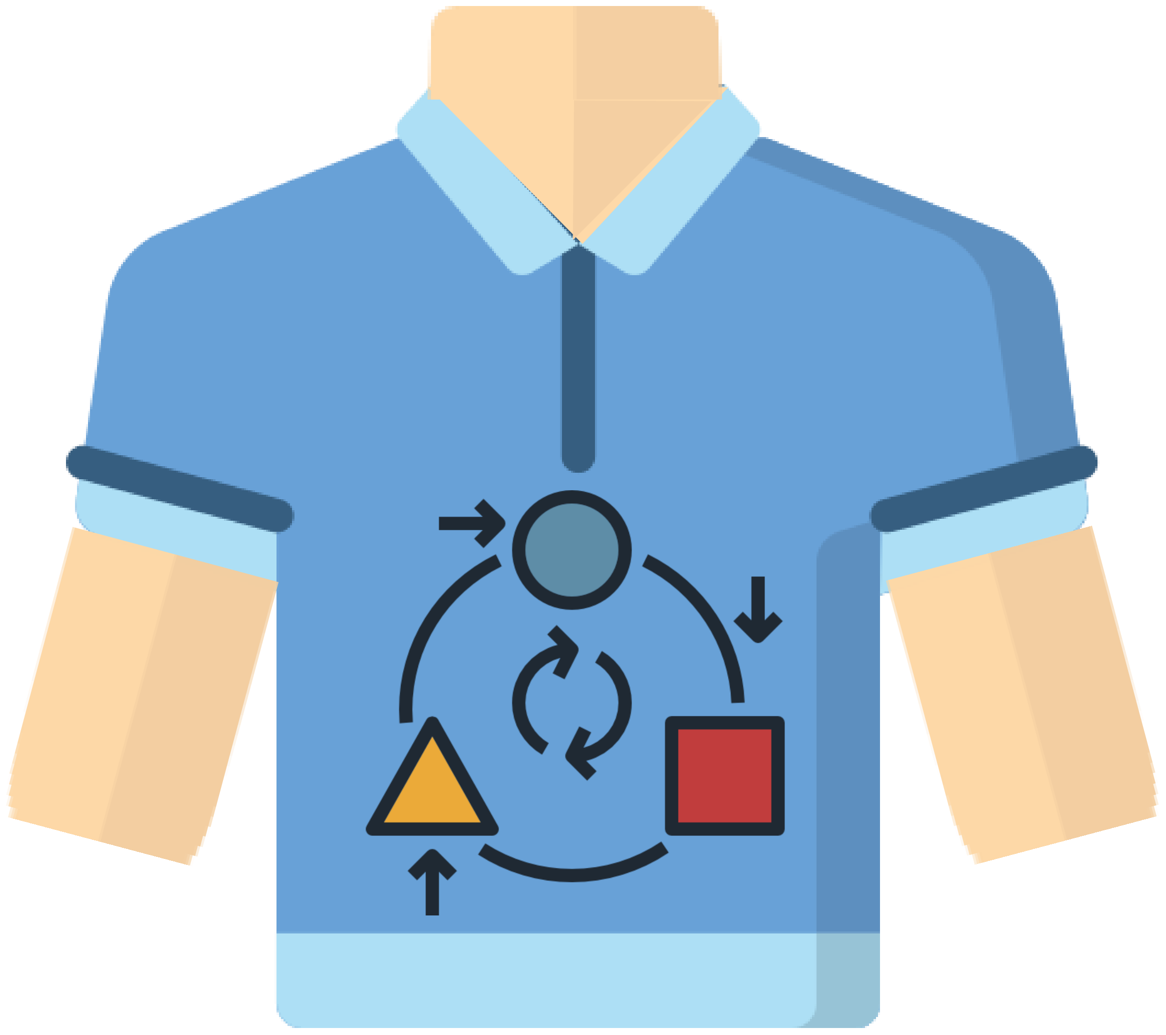
A service with its ClusterIP set to **none** is known as a Headless Service



Note: Unlike other services, headless service does not provide any load-balancing or proxy capabilities

Copyright © edureka and/or its affiliates. All rights reserved.

Headless Service (Contd.)



- Helps decouple service from Kubernetes giving an alternate form of service discovery
- DNS entries are still recorded on the cluster to discover pods through the service
- Headless Service helps the client to connect to all the pods
- StatefulSet requires a headless service for its pods to be discovered by the network
- Running a DNS query for a Headless Service returns the list of pods attached to it

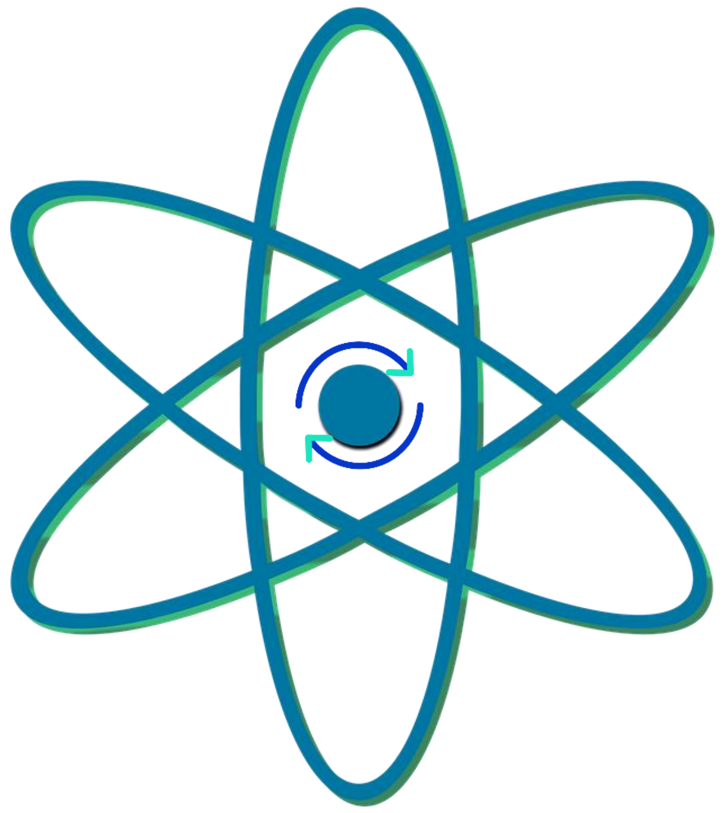
Copyright © edureka and/or its affiliates. All rights reserved.

StatefulSets

Copyright © edureka and/or its affiliates. All rights reserved.

StatefulSets

- StatefulSets create pods with sticky identities
- The pods retain their identity through various reschedulings



edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

StatefulSets (Contd.)

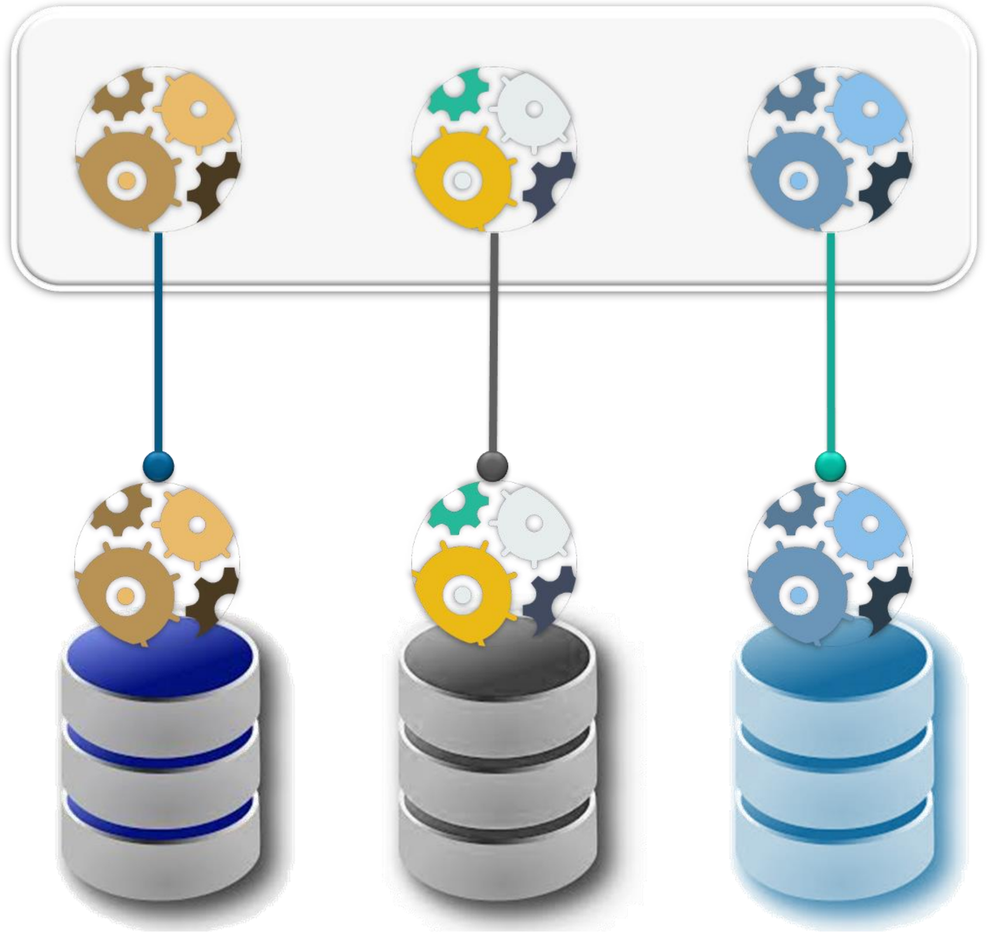
01 StatefulSets provides a stable identity and state for non-fungible applications

02 Keeps track of the number of pods deployed and ensures that the said number is maintained

03 Pod template is specified as a part of StatefulSet’s specification

04 The Pods scheduled are not exact replicas of each other. Each pod has its own storage unit to work with

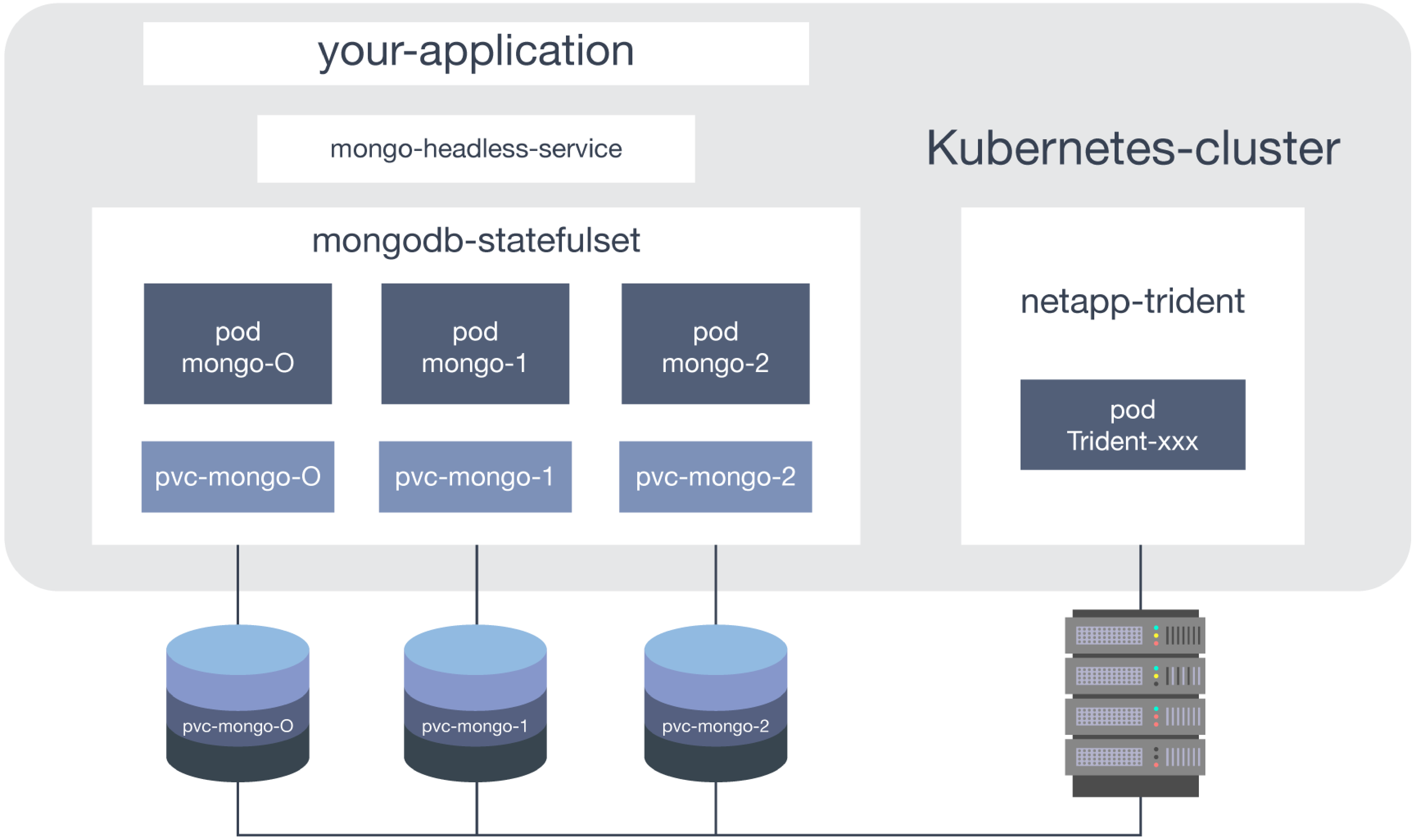
05 Rolling-updates can be applied to pods associated with StatefulSets



edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Sample Use Case for Stateful Applications



Copyright © edureka and/or its affiliates. All rights reserved.

Demo: StatefulSets

Copyright © edureka and/or its affiliates. All rights reserved.

ConfigMaps and Secrets

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

ConfigMaps

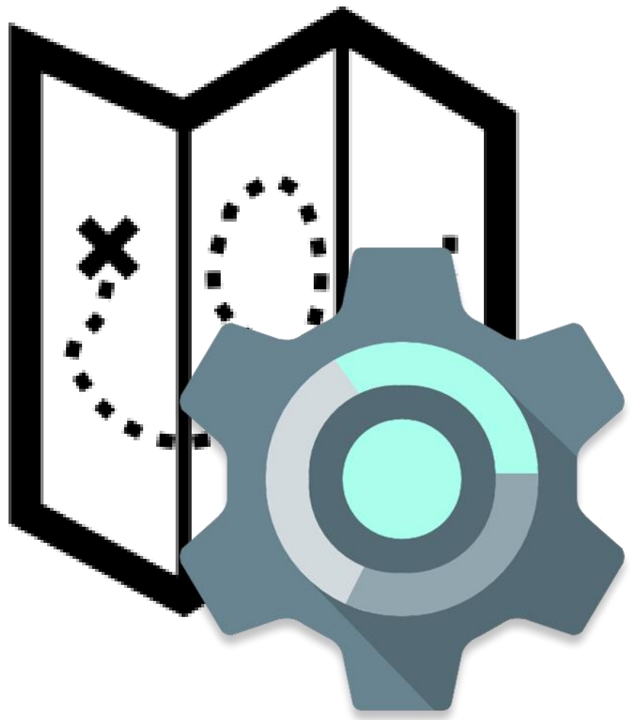
01 Exports configuration data into a container to avoid hardcoding it inside the application

02 Employs the ConfigMap API resource which contains configuration data in the form of key-value pairs

03 Used to pass non-sensitive information to the containers

04 The contents of the ConfigMap are not directly passed to the application

05 Content is passed either to the container as environment variables or to the volumes as files



edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Secrets

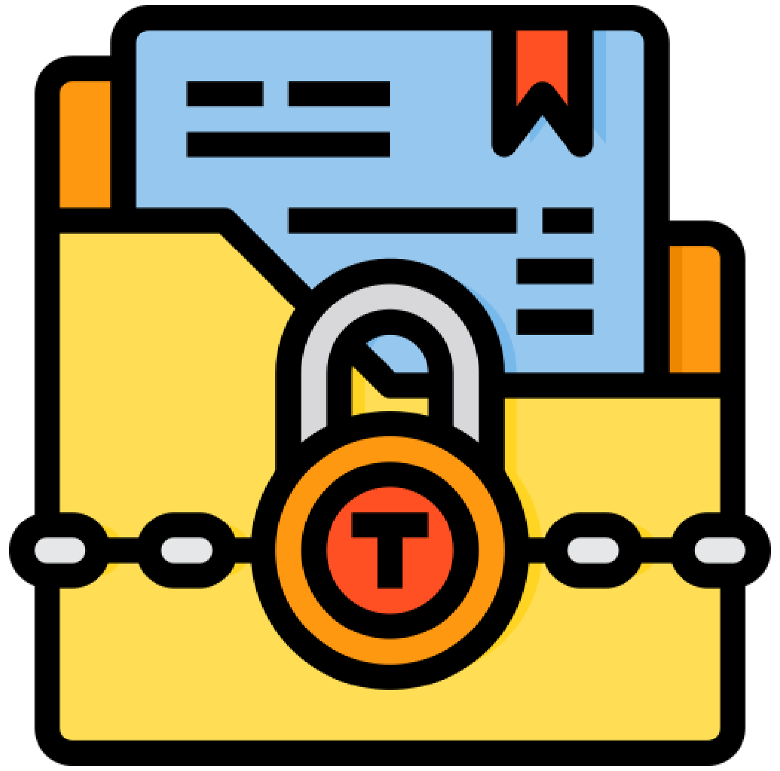
01 Secrets allow passing sensitive information to the containers

02 Holds information in key-value pairs and can be passed as environment variables or files in a volume

03 Only distributed to the nodes that are running the pod which need access to the secrets content

04 Stored on memory (RAM) of the node instead of the physical memory (HDD or SSD)

05 On the master node, secrets are stored in encrypted form on the etcd storage



Copyright © edureka and/or its affiliates. All rights reserved.

Demo: ConfigMaps and Secrets

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Kubernetes Package Manager: Helm

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Helm

- Packages collection of Kubernetes logic which can be deployed later
- The packages can be distributed and reused as HELM Charts

Copyright © edureka and/or its affiliates. All rights reserved.

Helm (Contd.)

- Official Cloud Native Foundation supported open-source Kubernetes project
- It can install software, its dependencies, update existing software, configure deployments and so on
- Uses a command line utility called **`helm`** and a companion component called **`tiller`**
- **`Tiller`** runs on the cluster to handle configurations and deployments. It constantly listens for incoming **`helm`** commands
- Stores packages in the form of helm charts and maintains a chart repository

Note: Tiller has been removed from Helm Version3

Copyright © edureka and/or its affiliates. All rights reserved.

Helm Charts

- Consists of YAML configurations and templates
- Configurations are passed as manifest files to Kubernetes

```
Directory structure of a Helm Chart:
     package-name/
     - charts/
     - templates/
     - Chart.yaml
     - LICENCE
     - README.md
     - requirements.yaml
     - Values.yaml
```

Copyright © edureka and/or its affiliates. All rights reserved.

Helm Charts (Contd.)

charts/	Dependencies which are managed manually are placed in this directory
templates/	Contains config files combined with values which are tuned into Kubernetes manifests
Chart.yaml	Holds metadata about the chart with information such as chart name, version, keywords and so on
LICENCE	Plaintext file containing the licencing information of the helm chart's content
README.md	Readme information document for the helm chart
requirements.yaml	Consists of the chart's dependencies
Values.yaml	Holds the default configuration values for the chart

edureka!

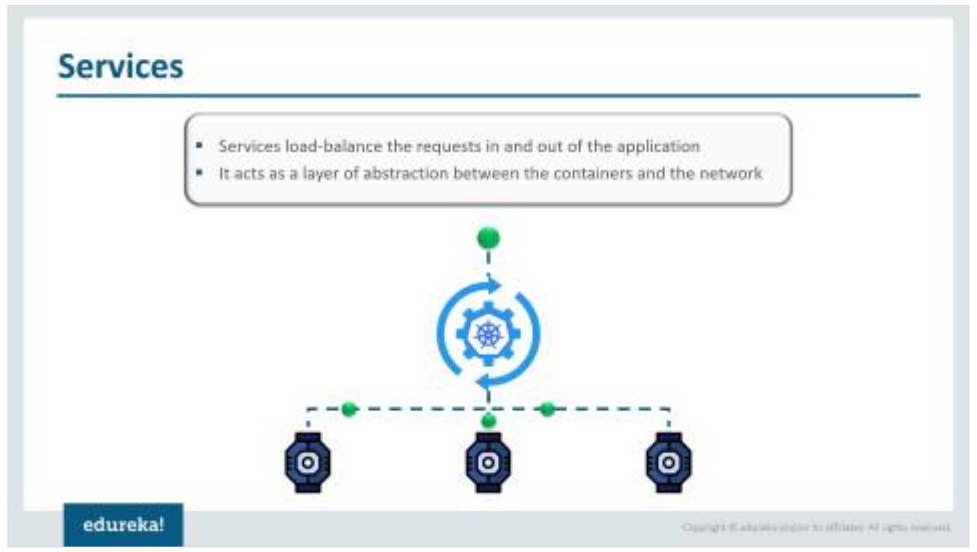
Copyright © edureka and/or its affiliates. All rights reserved.

Demo: Helm Charts

edureka!

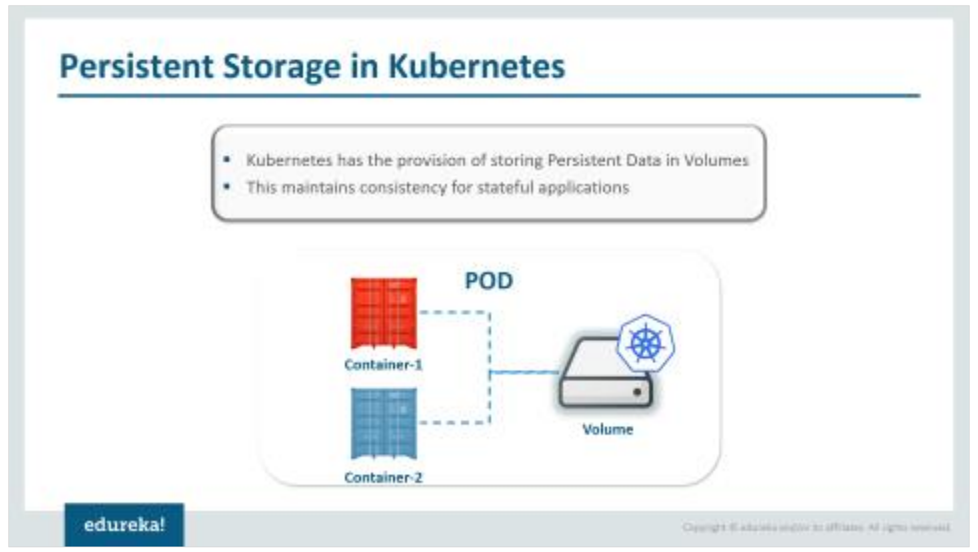
Copyright © edureka and/or its affiliates. All rights reserved.

Summary


Services

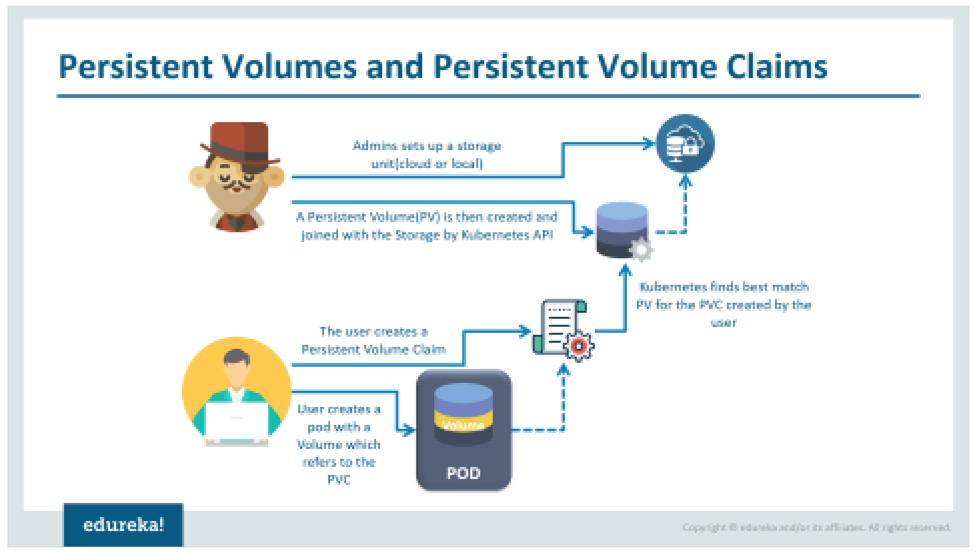
- Services load-balance the requests in and out of the application
- It acts as a layer of abstraction between the containers and the network

edureka!


Persistent Storage in Kubernetes

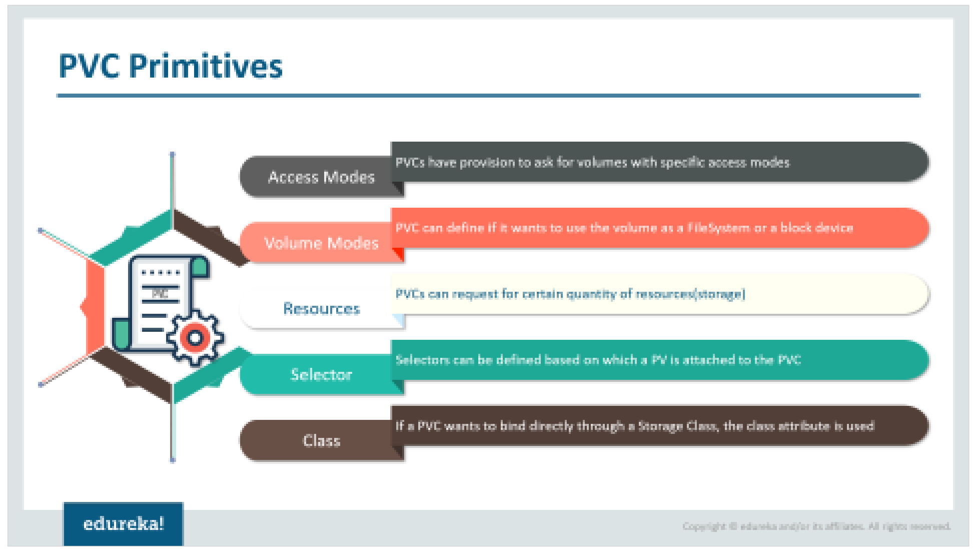
- Kubernetes has the provision of storing Persistent Data in Volumes
- This maintains consistency for stateful applications

edureka!


Persistent Volumes and Persistent Volume Claims

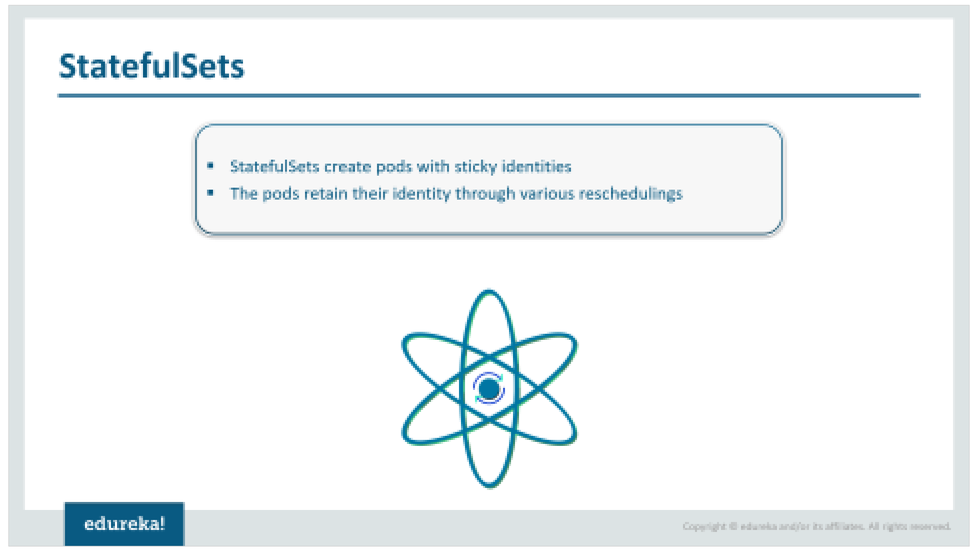
Admins sets up a storage unit(cloud or local)
A Persistent Volume(PV) is then created and joined with the Storage by Kubernetes API
Kubernetes finds best match PV for the PVC created by the user
The user creates a Persistent Volume Claim
User creates a pod with a Volume which refers to the PVC
Volume
POD

edureka!


PVC Primitives

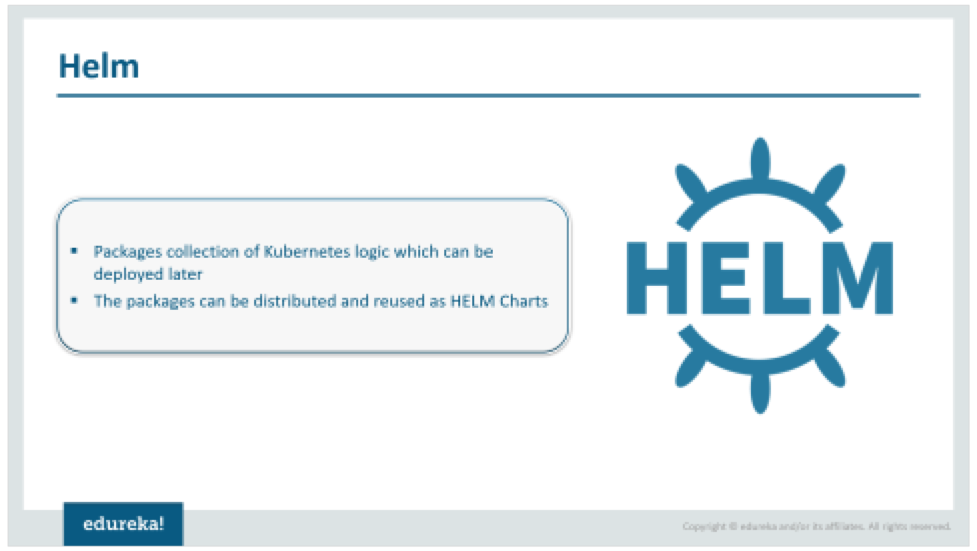
Access Modes	PVCs have provision to ask for volumes with specific access modes
Volume Modes	PVC can define if it wants to use the volume as a FileSystem or a block device
Resources	PVCs can request for certain quantity of resources(storage)
Selector	Selectors can be defined based on which a PV is attached to the PVC
Class	If a PVC wants to bind directly through a Storage Class, the class attribute is used

edureka!


StatefulSets

- StatefulSets create pods with sticky identities
- The pods retain their identity through various reschedulings

edureka!


Helm

- Packages collection of Kubernetes logic which can be deployed later
- The packages can be distributed and reused as HELM Charts

edureka!

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Questions

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Copyright © edureka and/or its affiliates. All rights reserved.

Thank You

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