Get the best out of Live Sessions

HOW?

e!

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!

edureka!

DevOps Certification Training

edureka!

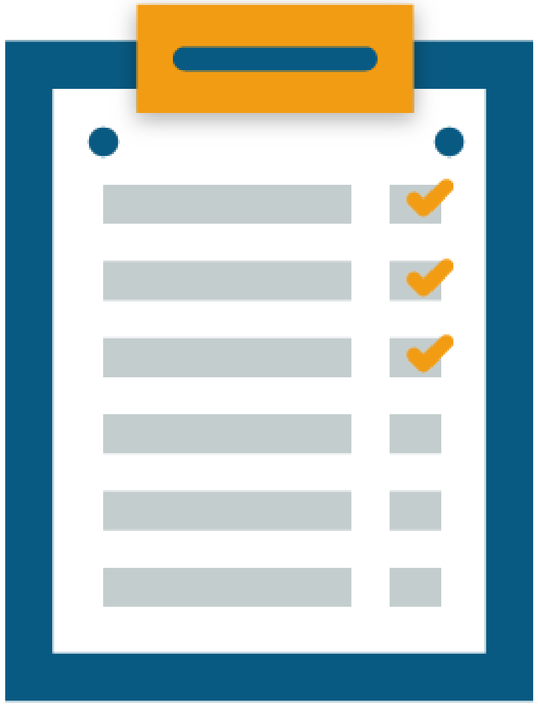
Module 8 – Container Orchestration Using Kubernetes Part-I

edureka!

Topics

Following are the topics covered in this module:

- Introduction to Container Orchestration
- Kubernetes Core Concepts
- Understanding Pods
- ReplicaSet and Replication Controller
- Deployments
- DaemonSets
- Rolling Updates and Rollbacks
- Scaling Application



edureka!

Objectives

After completing this module, you should be able to:

- Understand Container Orchestration
- Learn about Kubernetes Core Concept
- Deploy Pods
- Create Deployments to manage Pods
- Launch DaemonSets for Background applications
- Update and Rollback your Deployments
- Scale your containerized Applications

Case Study: Booz Allen Hamilton

edureka!

Booz | Allen | Hamilton

- Booz Allen Hamilton is a American Management and Information Technology Consulting Firm
- It’s home to about 27,600 employees
- The core business is driven to provide consulting, analysis, and engineering services to public and private sector organizations

Booz | Allen | Hamilton

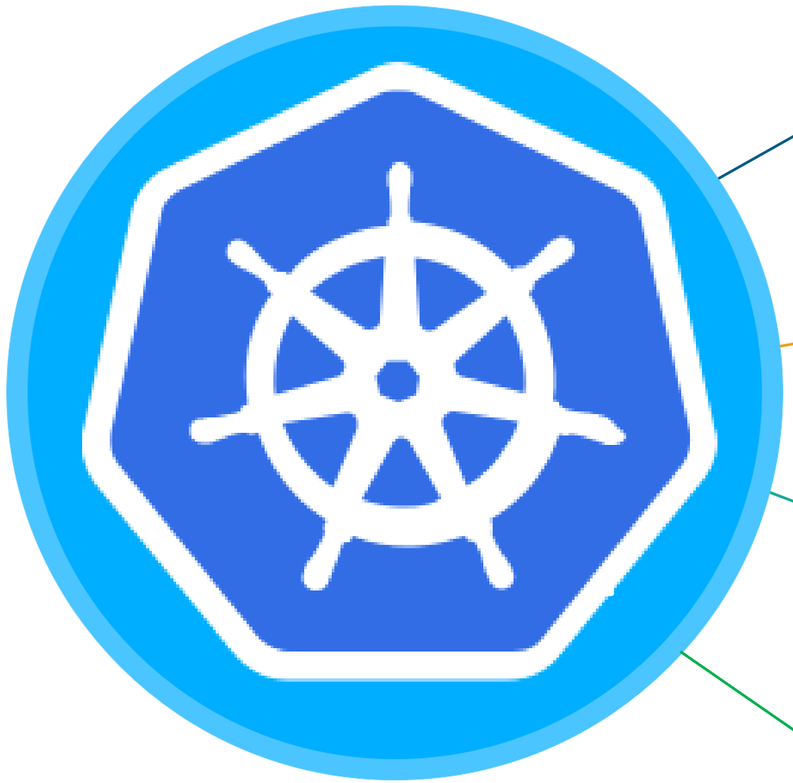
The Challenge

In 2017, BoozAllen was contracted by the federal Government to relaunch their old *recreation.gov* website

- The infrastructure for the new website needed to be Agile, Reliable, and Scalable
- The also needed the option to replicate the process in case any other government aided agency required a revamp

Booz | Allen | Hamilton

The Solution: Kubernetes



1. The first step to the solution was to create everything in the Microservice Architecture using Containers
2. Kubernetes provided the perfect platform for their dynamic and agile containerized platform
3. With the help of Kubernetes, New changes could be pushed out within 30 minutes compared to hours/days previously
4. Kubernetes has enabled a robust system enabling the 10+ deployments everyday on production

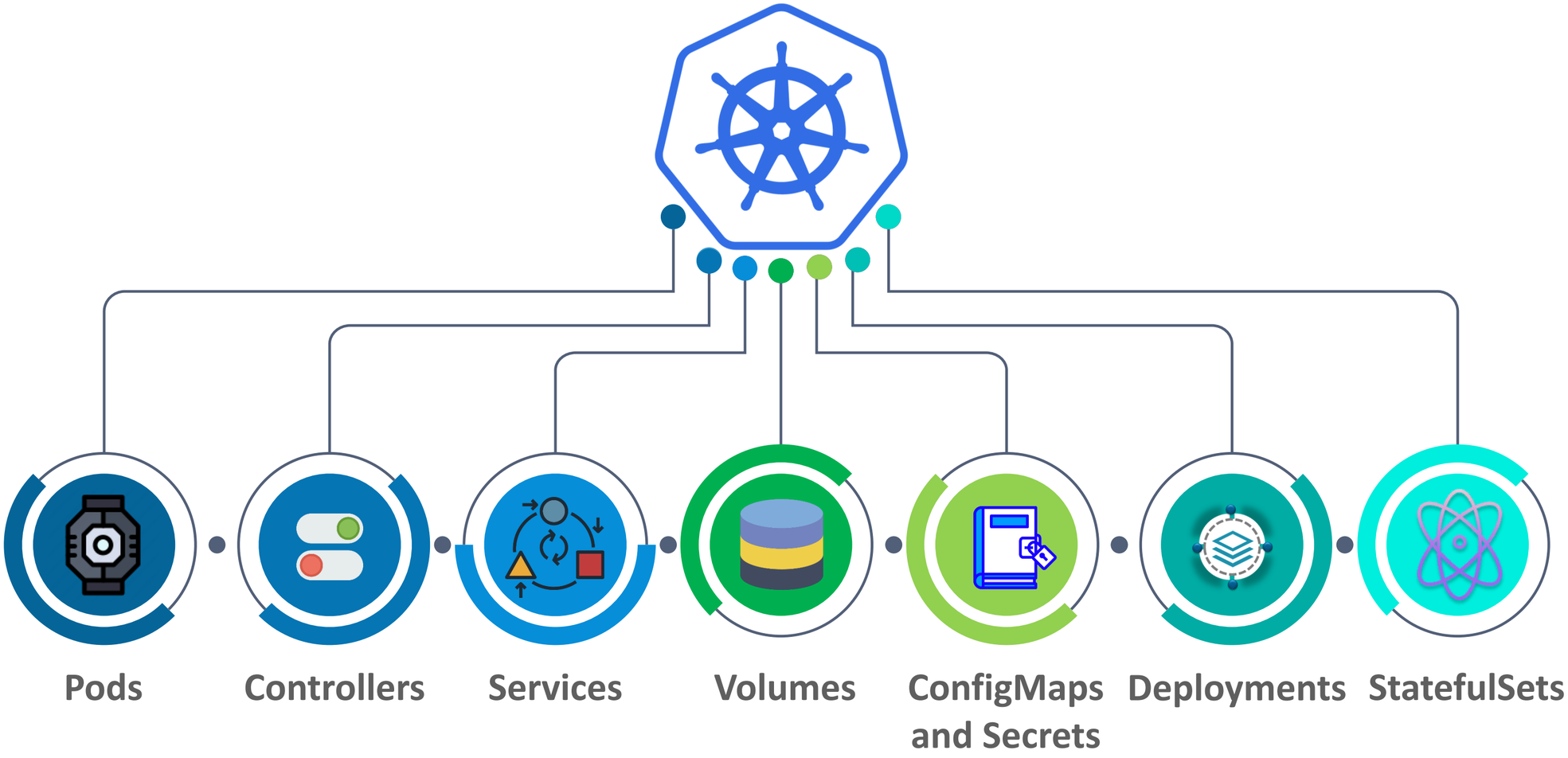
Kubernetes Core Concepts

edureka!

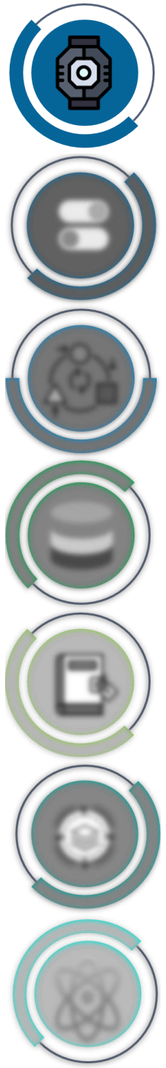
What is Kubernetes?

- Kubernetes is an open-source, portable platform for automating deployment, scaling, and management of containerized workloads and applications
- It groups containers that make up an application into logical units for easy management and discovery

Kubernetes Core Concepts

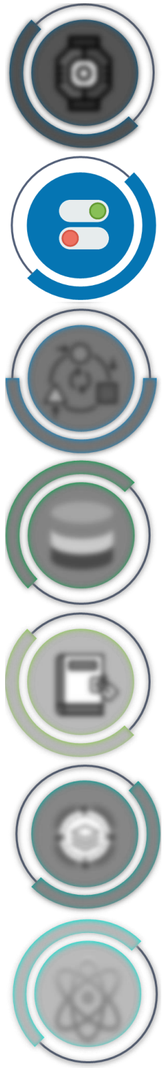


Kubernetes: Pods



- Basic building blocks of Kubernetes
- To reduce coupling as much as possible, single-container pods are used
- Multi-container pods can be deployed for cohesive processes
- Multi-container pods are more efficient since they share the same user-space

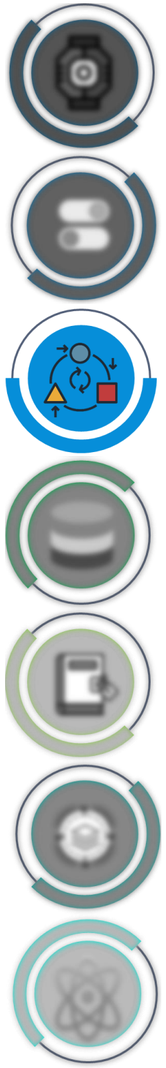
Kubernetes: Controllers



- Controllers are control loops that monitor the state of the cluster through the API Server
- Maintains and brings the cluster to the desired state
- All controllers are separate but are compiled into a single binary running as a single process
- Example: Deployment, ReplicaSet, NodeController

edureka!

Kubernetes: Services



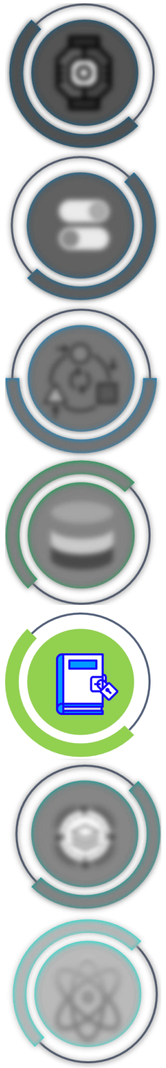
- Services add a layer of abstraction over the pods
- The abstraction defines a policy through which the pods are accessed
- Services load balance the requests across a set of pods

Kubernetes: Volumes

- Containers are ephemeral, and once the container crashes, all its data is lost
- Volumes provide a definite storage structure which can be attached to a pod
- Lifecycle of volume can be bound to the pod
- Volumes help restore stopped/unavailable containers to their previous state

edureka!

Kubernetes: ConfigMaps and Secrets



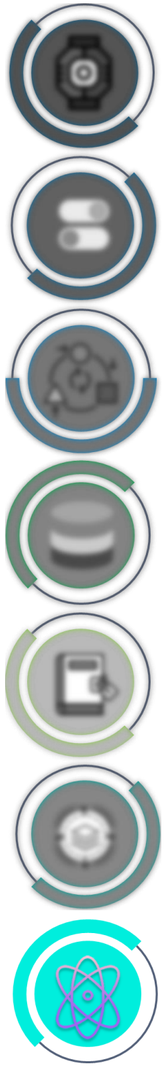
- ConfigMaps and Secrets allow users to configure applications externally
- Allows passing sensitive information without hardcoding it inside the container spec
- ConfigMaps are used for non-confidential data
- Secrets are used for confidential data

edureka!

Kubernetes: Deployments

- Deployment transitions current environment to the desired state in a controlled manner
- Deploys a replication controller or a replica set which further deploy the pods
- Used to rollout updates to already deployed applications
- Rollbacks are possible in case of faulty rollout

Kubernetes: StatefulSet



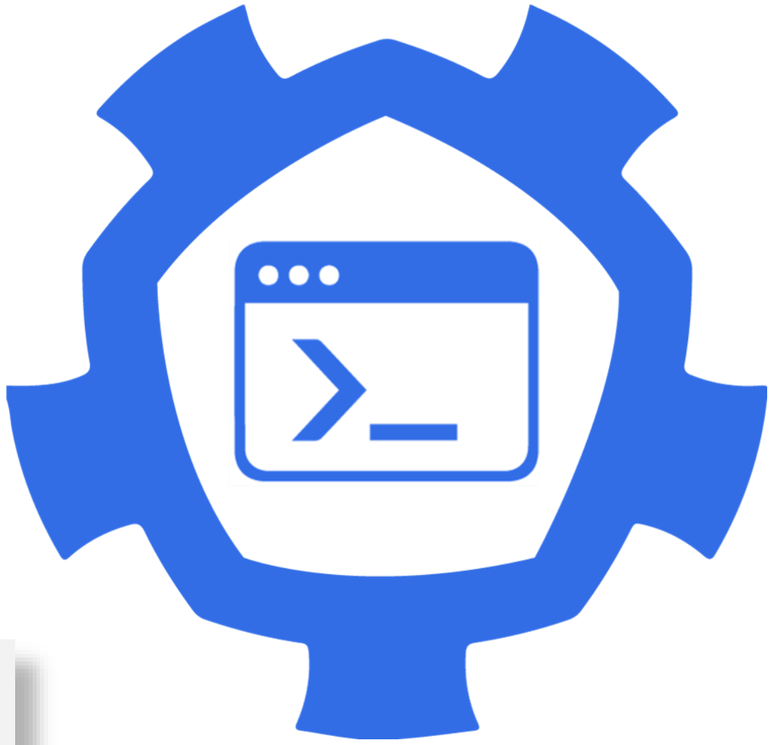
- StatefulSet is ReplicaSet equivalent for stateful applications
- Maintains uniqueness of each pod across different deployments and scaling operations
- Requires a headless service to provide network identity to the pods

Kubectl Common Commands

edureka!

Introduction to Kubectl

- Kubectl is the command line tool which manages the Kubernetes cluster
- It takes configurations from $HOME/.kube directory
- Interacts with the api-server to manage the cluster



Note: kubectl cannot be used to bootstrap a Kubernetes cluster

Kubectl Commands: Create

- **Create**
- Get
- Run
- Expose
- Delete
- Apply
- Edit

- Creates Kubernetes resources using a file or standard input
- The file format could be YAML or JSON
- Resources such as deployments, services, clusterRoles, and jobs can be created using the create command

```
$kubectl create –f fileName
```

edureka!

Kubectl Commands: Get

- Create
- **Get**
- Run
- Expose
- Delete
- Apply
- Edit

- Displays the resources and their current state
- Results are displayed in a tabular format and can be filtered according to the user's requirement

```
$kubectl get resourceType
```

edureka!

Kubectl Commands: Run

- Create
- Get
- **Run**
- Expose
- Delete
- Apply
- Edit

- Run directly deploys a particular image in the format user wants
- Generally, a deployment is created with a replication controller unless specified otherwise

```
$kubectl run name --image=imageName
```

edureka!

Kubectl Commands: Expose

- Create
- Get
- Run
- **Expose**
- Delete
- Apply
- Edit

- Used to expose Kubernetes resources as a service
- The selector for the service is the same as the selector for the resource
- To expose deployments or replicaSets, the equality-based selector must be used

```
$kubectl expose resourceType resourceName
```

edureka!

Kubectl Commands: Delete

- Create
- Get
- Run
- Expose
- **Delete**
- Apply
- Edit

- Deletes selected resources on the cluster
- Resources can be deleted using filename, stdin, resourceName, and labels & selectors
- Graceful deletion of pods is possible. This allows the current process to end before deletion

```
$kubectl delete –f fileName
```

edureka!

Kubectl Commands: Apply

- Create
- Get
- Run
- Expose
- Delete
- **Apply**
- Edit

- Changes the configuration of an already deployed resource
- If the resource does not exist, a new resource is created

```
$kubectl apply –f fileName
```

edureka!

Kubectl Commands: Edit

- Create
- Get
- Run
- Expose
- Delete
- Apply
- **Edit**

- Edits the specification YAML of a deployed resource
- Allows direct modification to any API resource which can be retrieved using the command line tools

```
$kubectl edit [–f filename]/[ResourceName]
```

edureka!

Other Useful Kubectl Commands

Command	Description
label	Creates/updates the labels on a resource
scale	Scales the current size of the pod-controlling resources to the desired size
describe	Gives a detailed description about the resource
exec	Used to enter and execute commands inside of the container shell
logs	Displays the logs for a pod or a resource
cluster-info	Displays the current state of the cluster
cp	Copies files/directories to and from containers
explain	Describes the API resources
set	Used to make changes to deployed application resources

Demo: Kubectl Common Commands

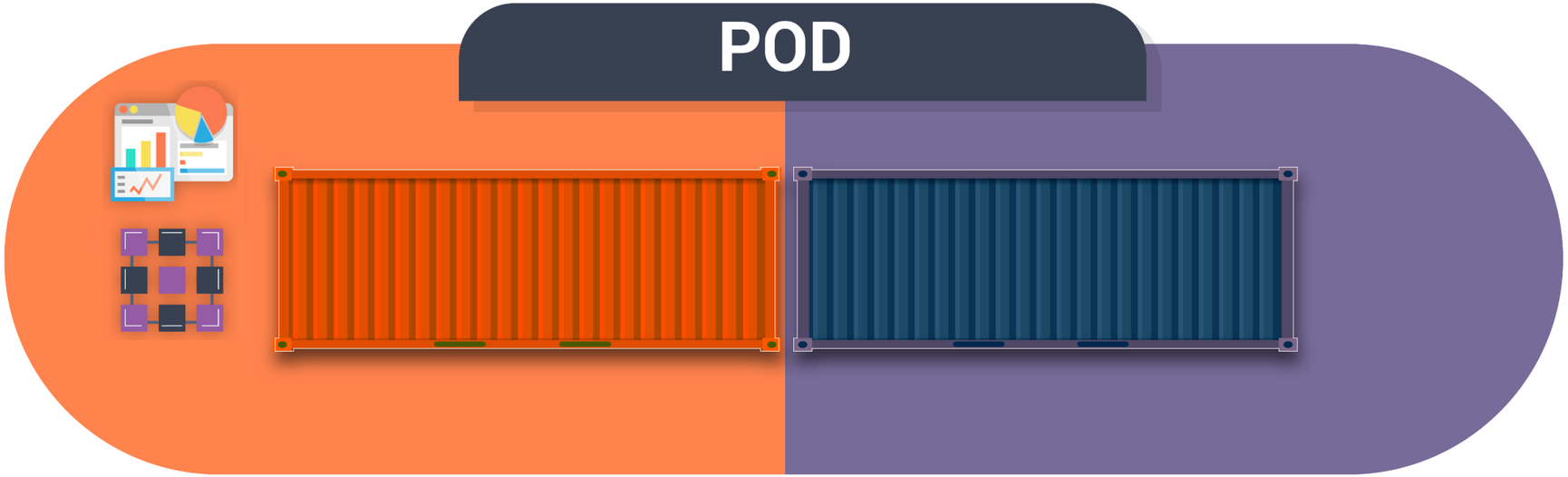
edureka!

Understanding Pods

edureka!

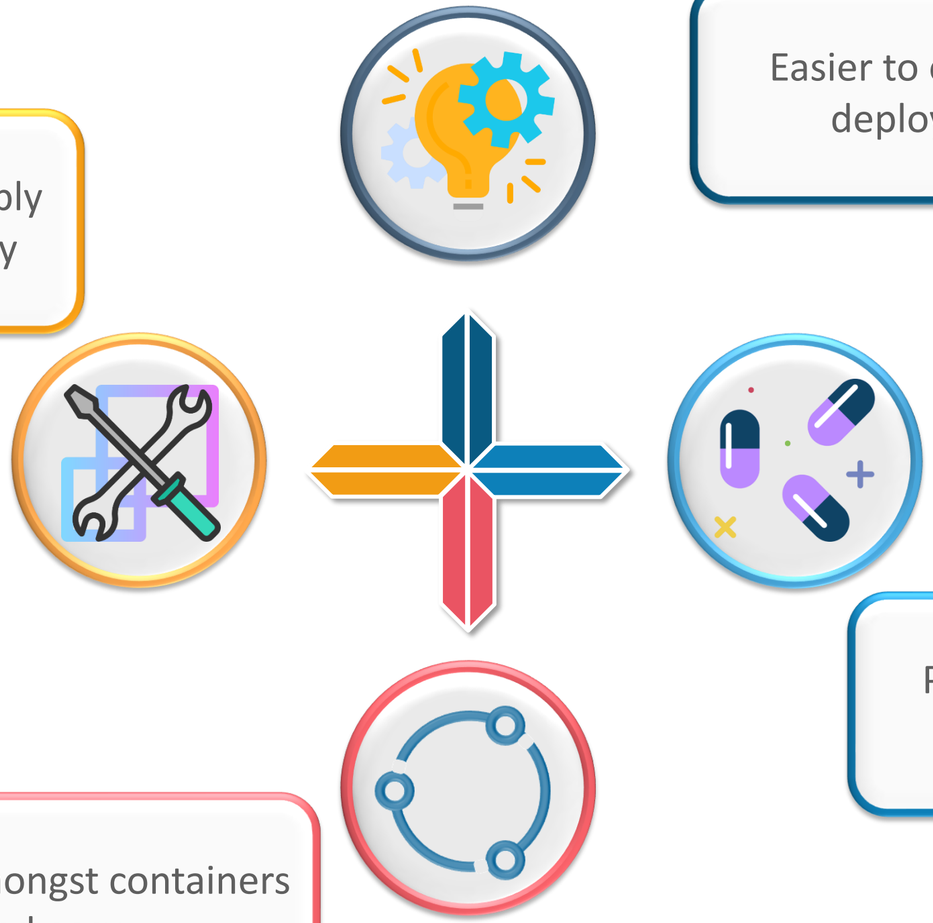
What is a Pod?

- Pod is the most basic unit of deployment in Kubernetes hierarchy
- It hosts container(s) inside it and is responsible for ensuring they're healthy and functioning as expected



Advantages of Pods

Pods can scale, configure, and apply patches to containers effortlessly



Easier to orchestrate containers deployed at a large scale

Pods are self-healing and do not require manual intervention

Resource sharing amongst containers is secure and easy

Ways to Consume Pods

Single Container Pods

Single container pod consists of one individual container

Preferred way of deploying pods in the Kubernetes environment

Removes the risk of coupling related issues

Pod acts as a wrapper to the container application

Multi-Container Pods

Multi-container pods contain multiple containers inside a single pod

Used when the application requires two or more processes to work as a cohesive unit

Usually, one of the containers provides service to the other container

Containers in the multi-container pod share IP-Addresses, Volumes, and so on

ReplicaSet and ReplicationController

edureka!

ReplicaSet

- Maintains a definite number of pods on the cluster at any given time
- Uses a set-based selector to identify and acquire pods
- PodSpec template and the number of pods are defined in the ReplicaSet YAML
- Pods have a metadata.ownerReference field through which ReplicaSet takes ownership of the pod
- ReplicaSet assumes ownership of pods without owners

ReplicationController

- Ensures a specific number of pods are always running
- Pods can be scaled up and down depending upon the current requirement
- Uses an equality-based selector to acquire pods
- Acts as a process supervisor, supervising pods across nodes on the cluster
- Maintains the number of pods in the case of a node-failure

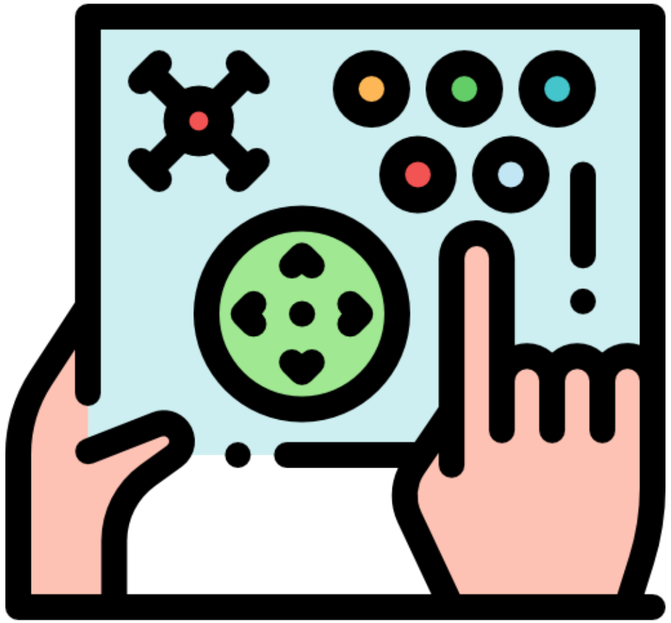
edureka!

Deployments

edureka!

Deployments

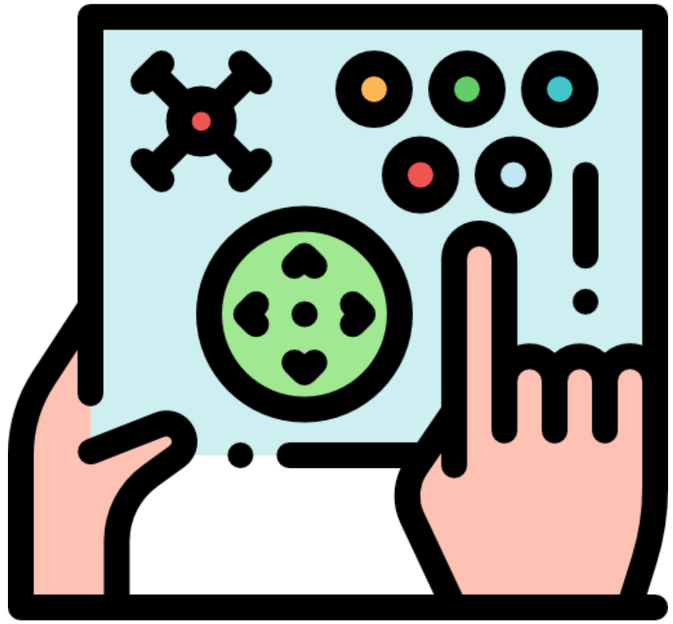
Deployments are controllers which provide declarative updates to pods and brings the current state of the system to the desired state



edureka!

Deployments (Contd.)

- Deploy a ReplicaSet which controls the pods
- Current state of the pods can be changed using deployments
- Update are rolled out in a gradual manner
- In-case of a faulty update, rollback to a previous version is possible



Demo: Deployments

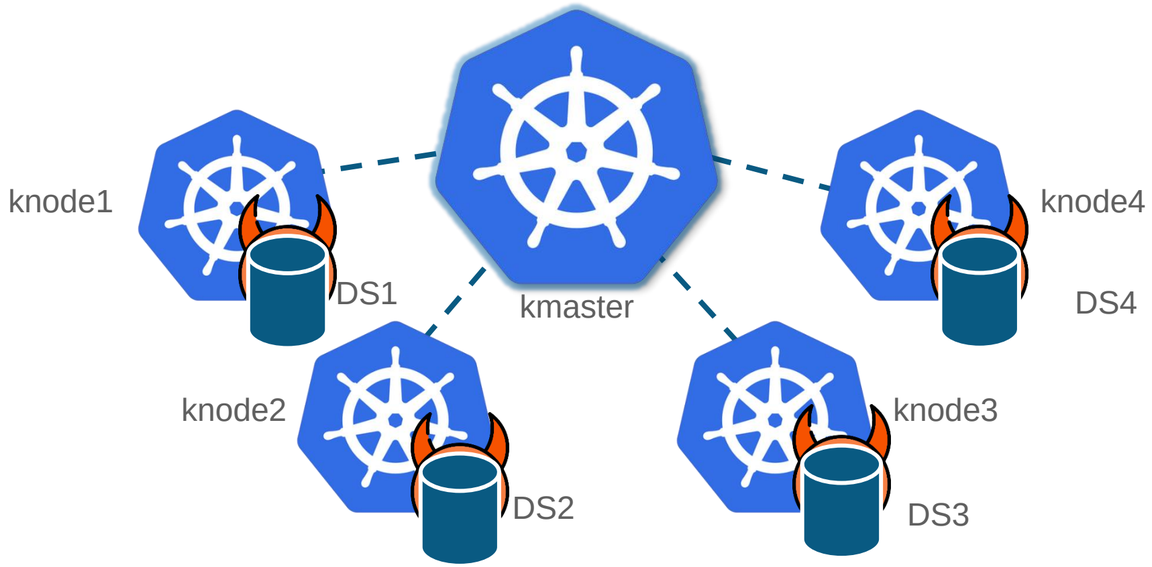
edureka!

DaemonSet

edureka!

DaemonSet

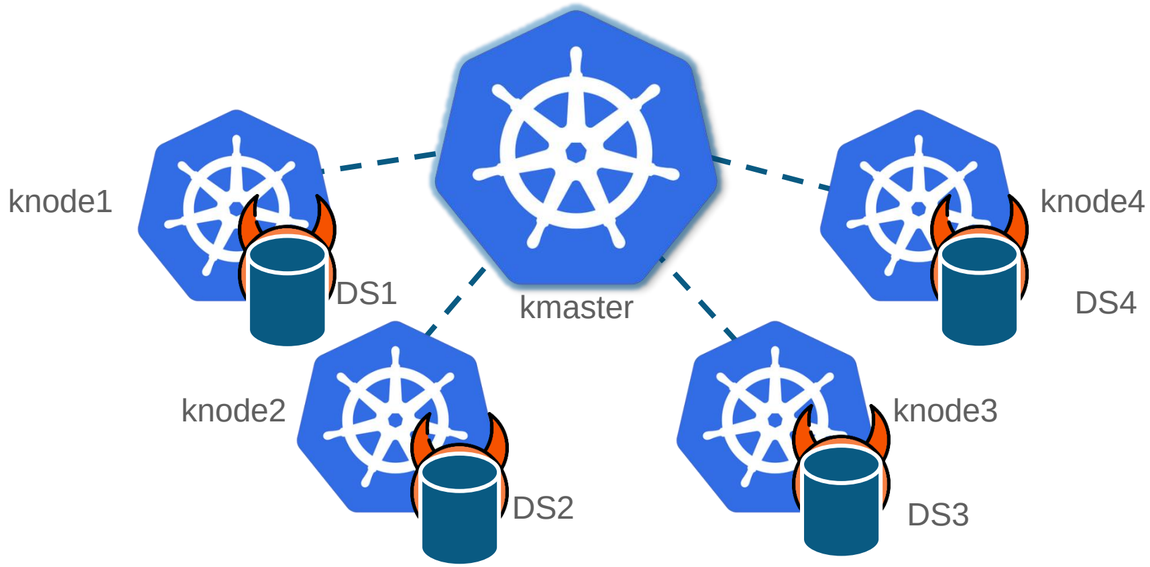
DaemonSet ensures that an instance of a pod is always running on all or selected node(s) in the cluster



edureka!

DaemonSet (Contd.)

- Used to run daemon processes for cluster applications/nodes
- Does not require scheduler intervention to schedule pods
- A common use-case is to use it as a logging/ monitoring agent
- When deleted, it terminates all the pods it has scheduled



Demo: DaemonSet

edureka!

Rolling Updates and Rollbacks

edureka!

Rolling Updates and Rollbacks

edureka!

Rolling Update

- Deployments changed how updates for applications are handled in Kubernetes
- The declarative nature of deployments makes it easy to rollout updates
- This reduces the application down-time significantly

Rolling Update (Contd.)

- Deployments manage rolling updates by creating a new ReplicaSet with the updated content
- The new ReplicaSet scales up while the old ReplicaSet scales down the pods
- The number of pods is maintained and there is no down-time
- The old ReplicaSet remains in the cluster and is not deleted

Note: The rolling-update command is applicable only to the replication controller

Rolling Update (Contd.)

- The `rollout --history` command is used to track changes/updates
- The cause of the change can be recorded
- A simple way to rollout an update is to edit the deployment using the `edit` command

Rollbacks

- After a rollout, the old ReplicaSet remains in the cluster
- If a rollback is required, the old ReplicaSet is scaled up and the current ReplicaSet is scaled down
- The required version can be retrieved from revision history

Demo: Rolling-Update and Rollback

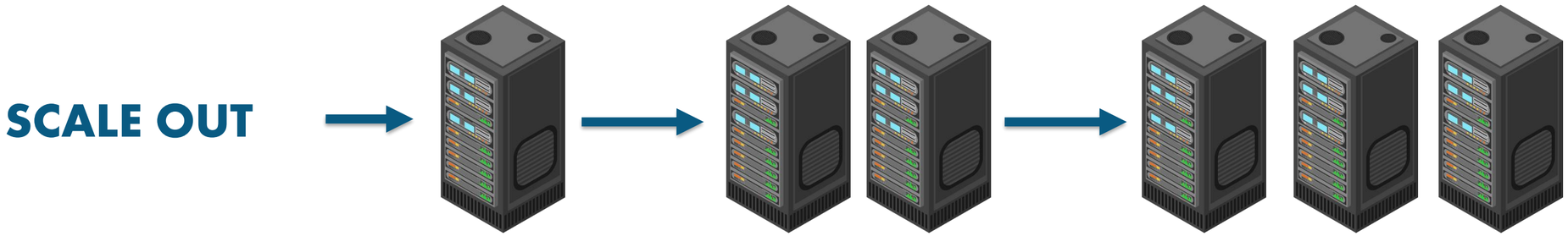
edureka!

Scaling in Kubernetes

edureka!

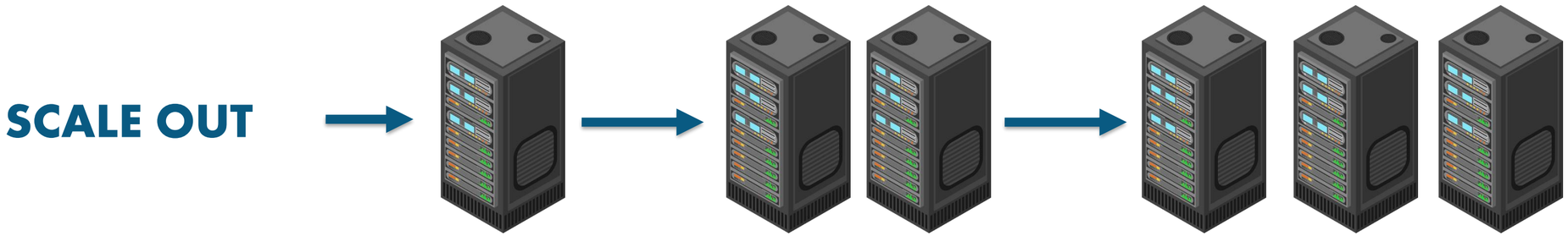
Kubernetes: Scaling

Pods can be scaled using direct CLI commands or by editing controller specification



Kubernetes: Manual Scaling

- For lighter workload demands, admins tend to scale their application manually
- The admin increases/decreases the number of pods according to the requirement session

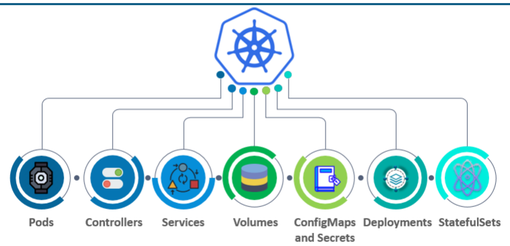


Demo: Scaling in Kubernetes

edureka!

Summary

Kubernetes Core Concepts



edureka!

Introduction to Kubectl

- Kubectl is the command line tool which manages the Kubernetes cluster
- It takes configurations from $HOME/.kube directory
- Interacts with the api-server to manage the cluster

Note: kubectl cannot be used to bootstrap a Kubernetes cluster

edureka!

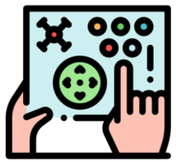
What is a Pod?

- Pod is the most basic unit of deployment in Kubernetes hierarchy
- It hosts container(s) inside it and is responsible for ensuring they're healthy and functioning as expected

edureka!

Deployments

Deployments are controllers which provide declarative updates to pods and brings the current state of the system to the desired state



edureka!

Rolling Update

- Deployments changed how updates for applications are handled in Kubernetes
- The declarative nature of deployments makes it easy to rollout updates
- This reduces the application down-time significantly

edureka!

Kubernetes: Scaling

Pods can be scaled using direct CLI commands or by editing controller specification

edureka!

edureka!

edureka!

edureka!

Thank You

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