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!

Microsoft Azure Developer Associate (AZ-204)

COURSE OUTLINE
MODULE 03

- Introduction to Azure IaaS Compute Solutions
- Implementing Azure Batch Service and Disk Encryption
- **Designing and Developing Applications That Use Containers**
- Implementing Azure App Service Web Apps and Mobile Apps
- Implementing Azure App Service API Apps and Azure Functions
- Developing Solutions That Use Azure Table Storage and Cosmos DB
- Developing Solutions That Use Relational Database And Azure Blob Storage
- Implementing Authentication and Access Control In Azure
- Implementing Secure Data Solutions and Integrate Caching & CDN
- Instrument Monitoring, Logging and Scalability Of Apps & Services
- Connecting to and Consuming Azure and Third-party Services
- Developing Event-based and Message-based Solutions in Azure

Module 3 – Designing and Developing Applications That Use Containers

edureka!

Topics

- Application Environment Components
- Containers v/s VM
- Docker
- Docker container
- Azure Container Registry (ACR)
- Azure Container Instances (ACI)
- Cluster Master
- Nodes
- Pods
- Concept Of Orchestration
- Azure Kubernetes Service (AKS)
- Kubernetes Cluster

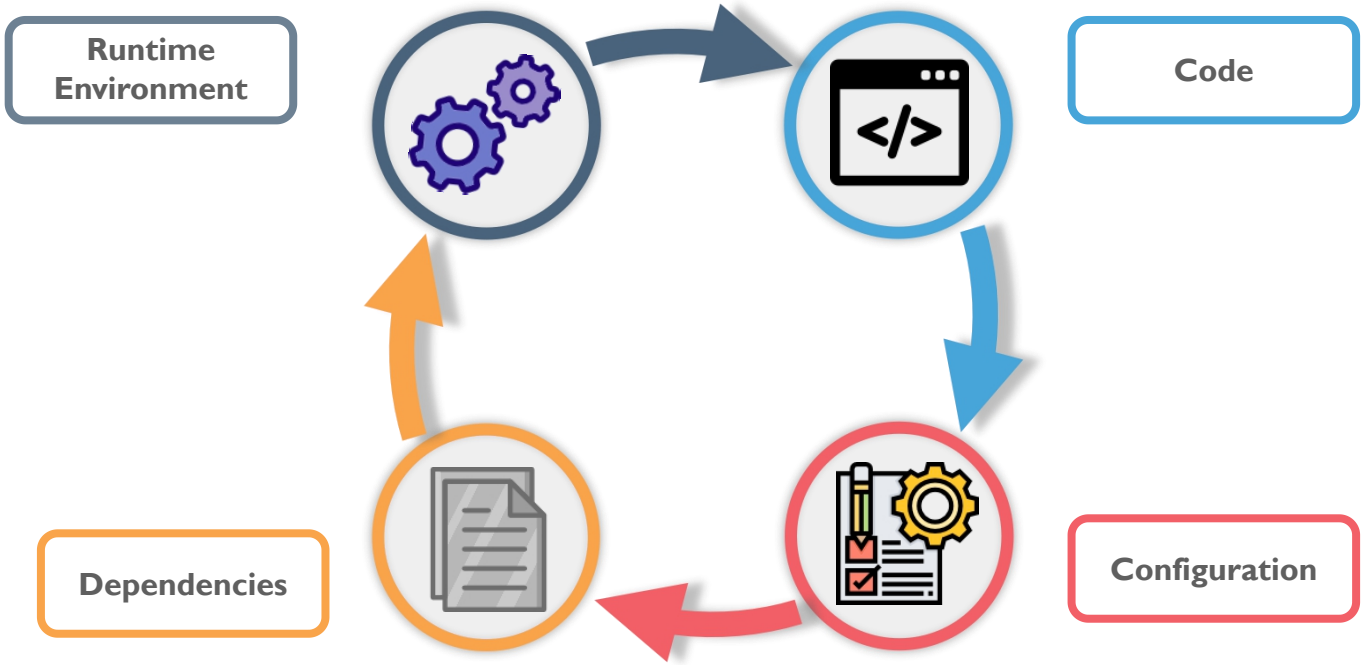
edureka!

Objectives

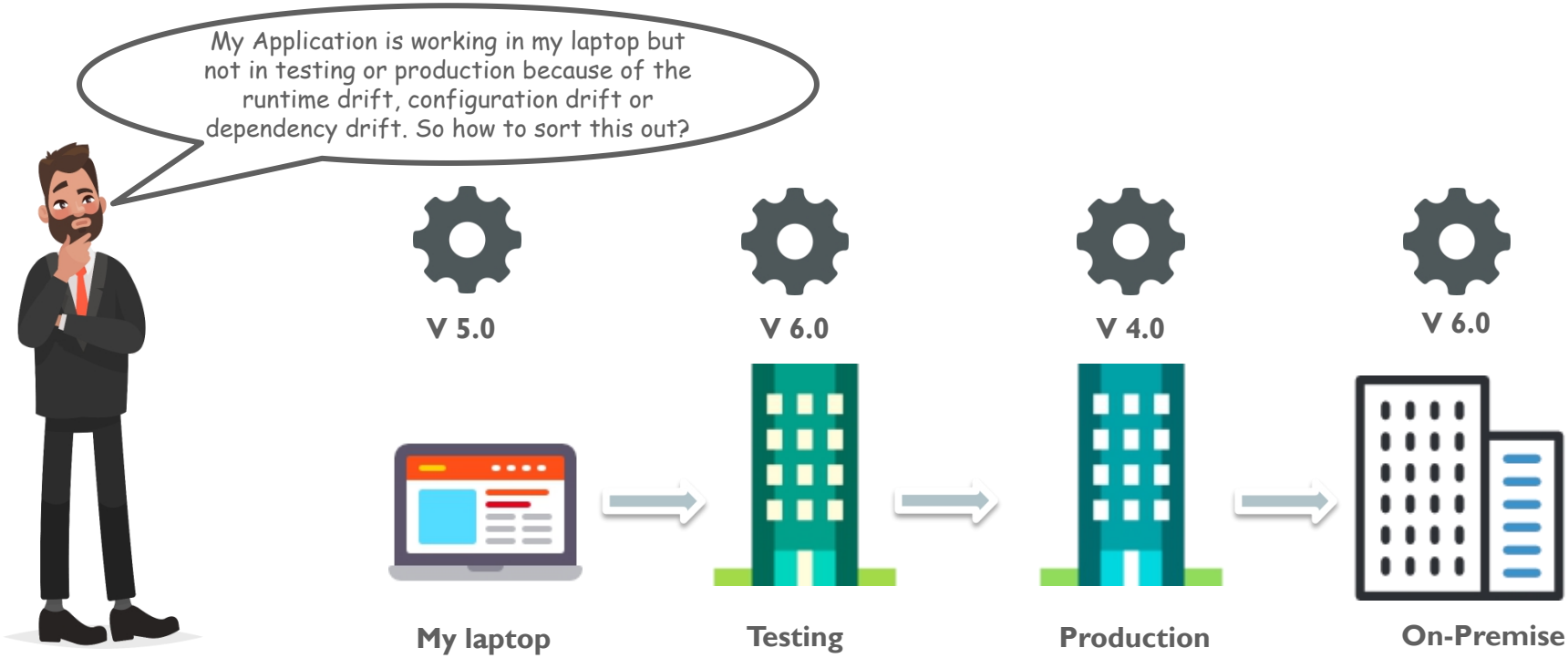
After completing this module, you should be able to:

- Deploy AKS (Azure Kubernetes Service) clusters
- Publish an image to the Azure Container Registry
- Run containers by using Azure Container Instance or AKS

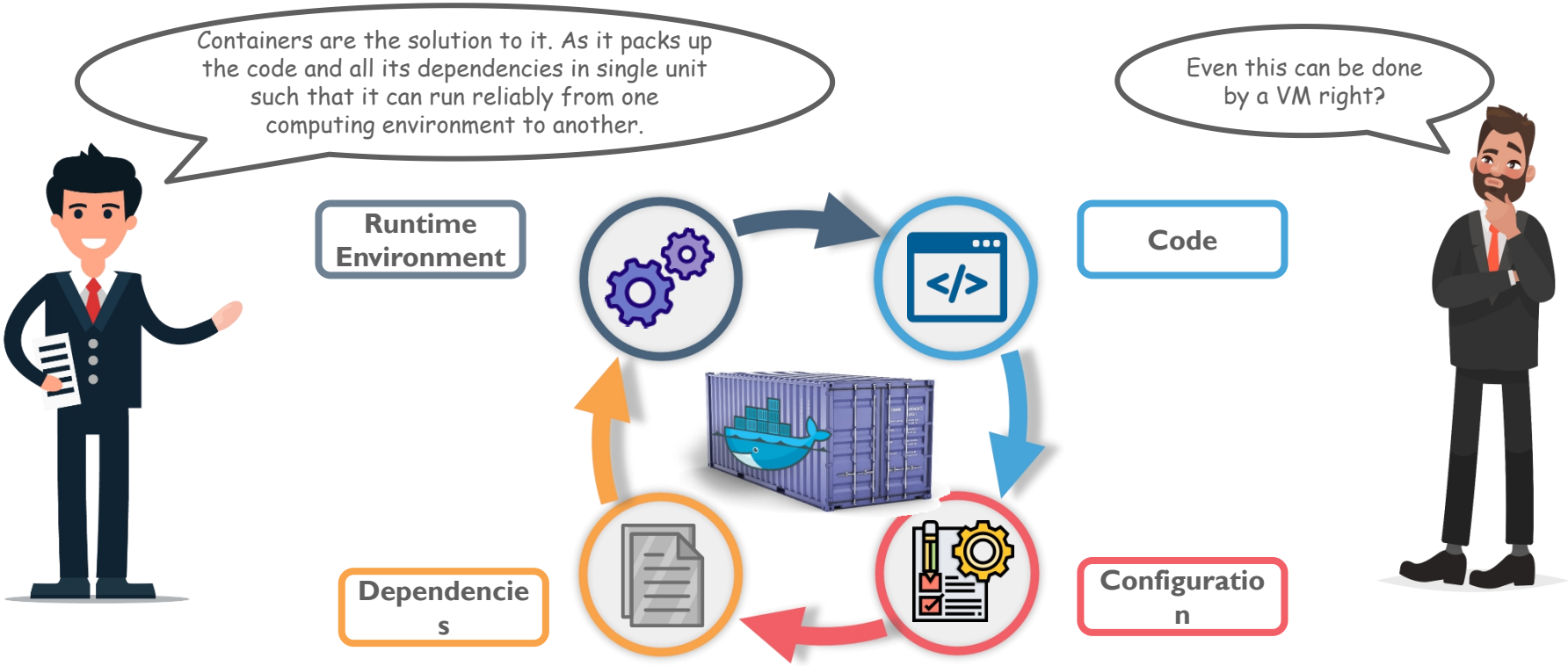
Application Environment Components



Problems Before Containers

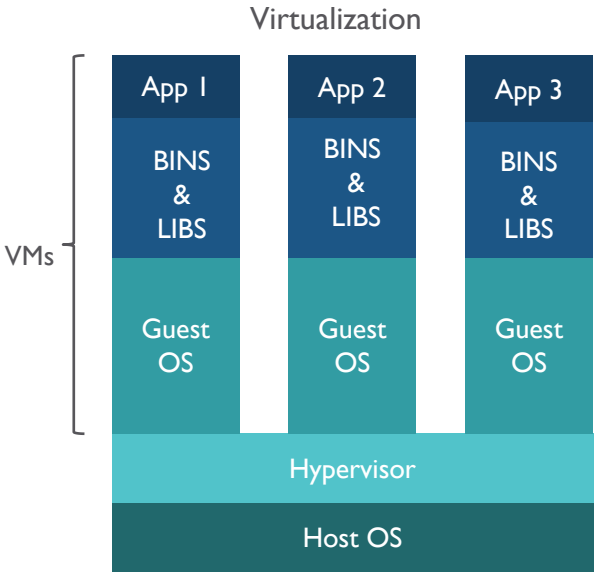


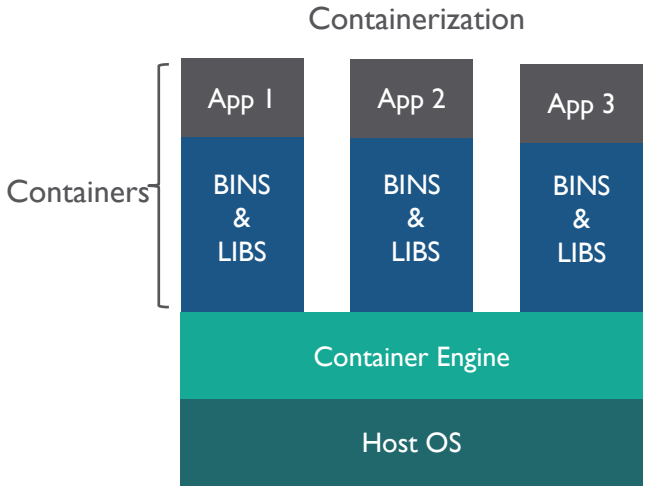
How Containers Solve That Issue?



Why Not in Virtual Machine?

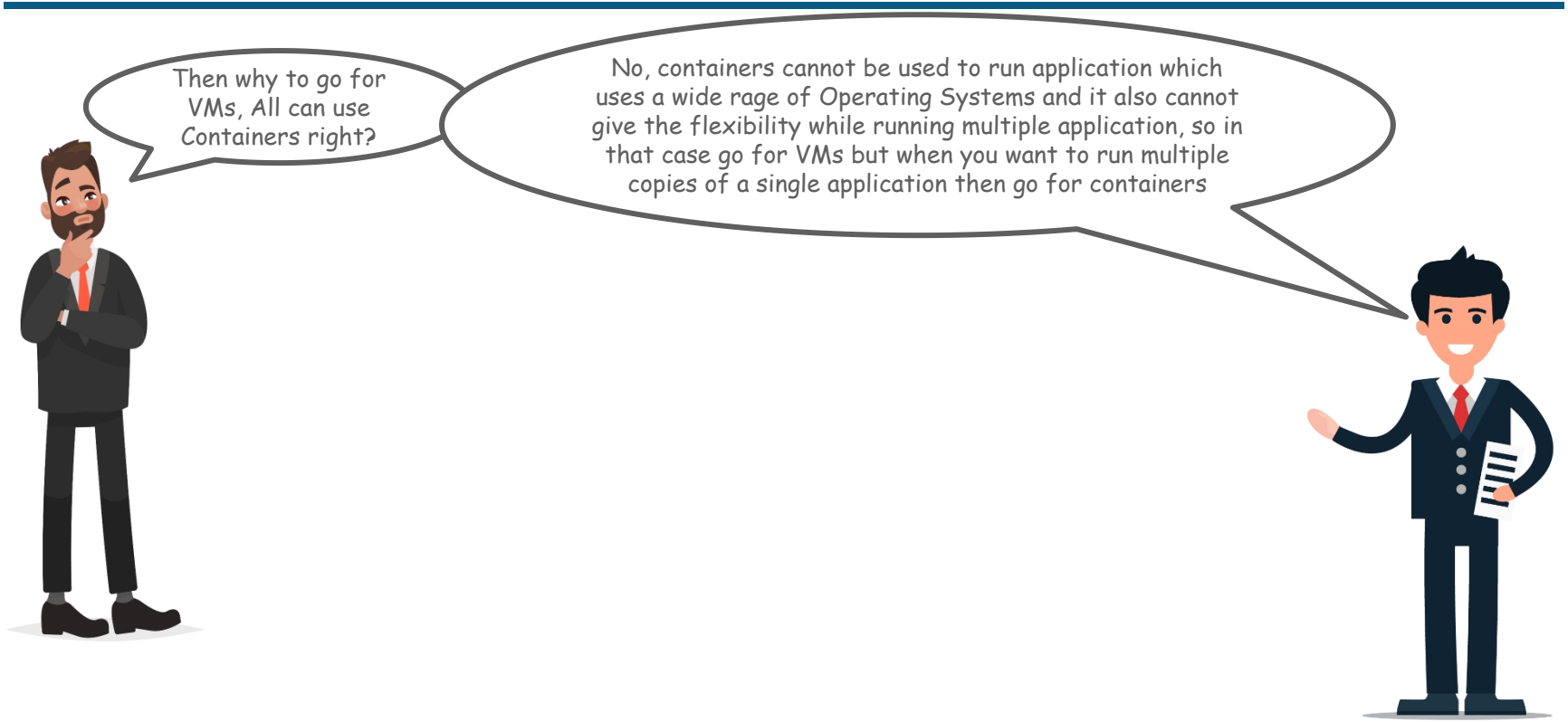
VMs take up a lot of resources as it has a full copy of a OS + virtual Copy of hardware, which adds up to lot of RAM and CPU, whereas in containers you just need supporting programs, libraries and system resources to run a application





edureka!

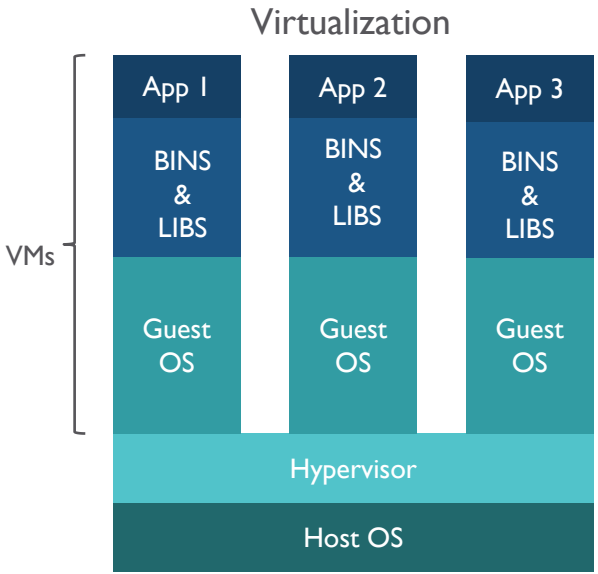
Best Fit



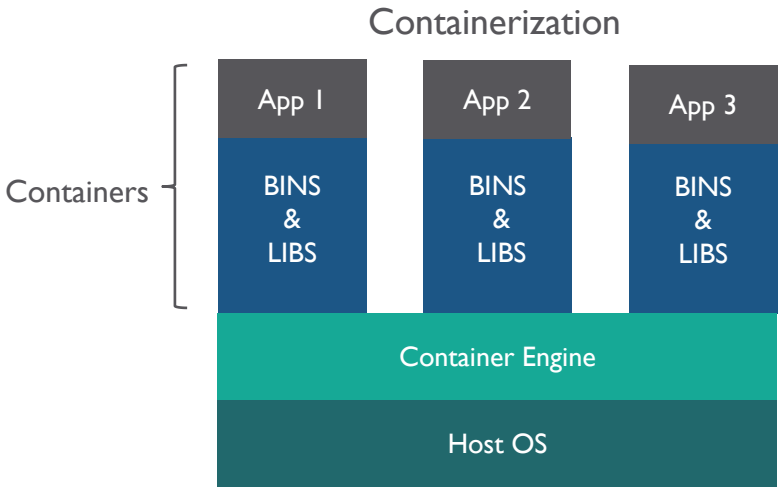
Container Vs. VM

Parameters	VM	Container
Work Scope	If you want to run multiple applications, **use VM**	If you want to run multiple copies of a single app, use Containers
Portability	Working with VMs are platform independent	Working with containers is platform dependent
Resources	VMs take up a lot of system resources	When **resources are a constraint, use Containers**
Security	You need not have **sudo privileges to import subsystems**	You need to have **sudo privileges to import subsystems**

Why Not Virtual Machines?



- VMs take up a lot of system resources
- Multiple VMs lead to unstable performance
- Long boot-up process (approx. 1 minute)
- Hypervisors are not efficient as Host OS



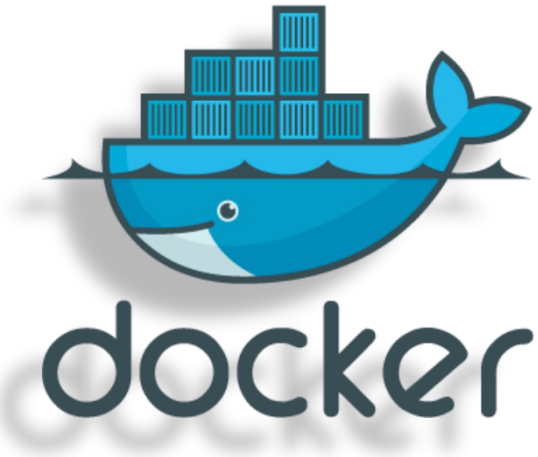
- Containers on the same OS kernel are lighter and smaller
- Better resource utilization compared to VMs
- Can run different application or application versions with different dependencies simultaneously
- Short boot-up process (1/20th of a second)

edureka!

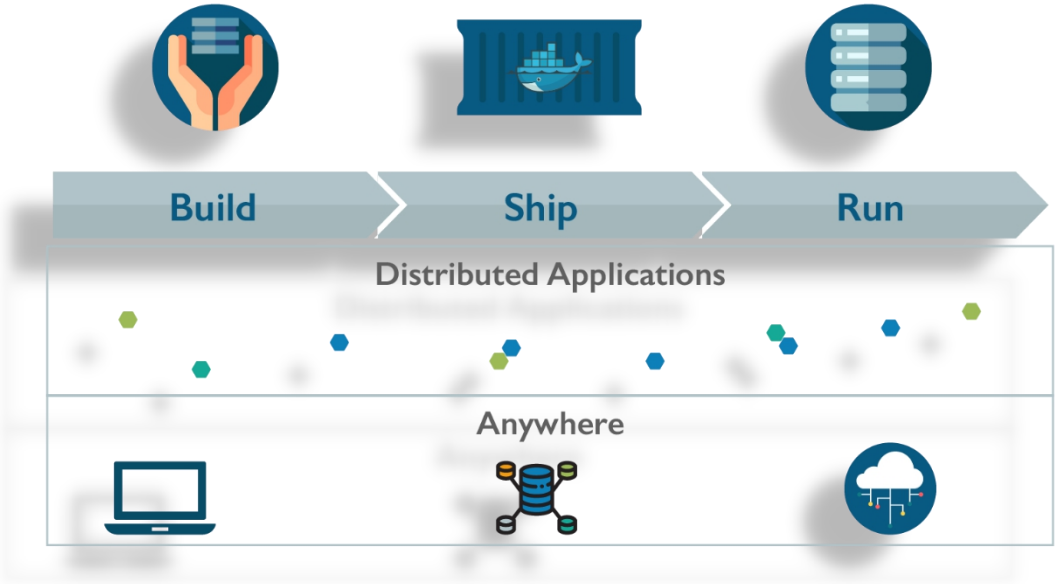
What is Docker?

"BUILD, SHIP & RUN ANY SOFTWARE ANY WHERE"

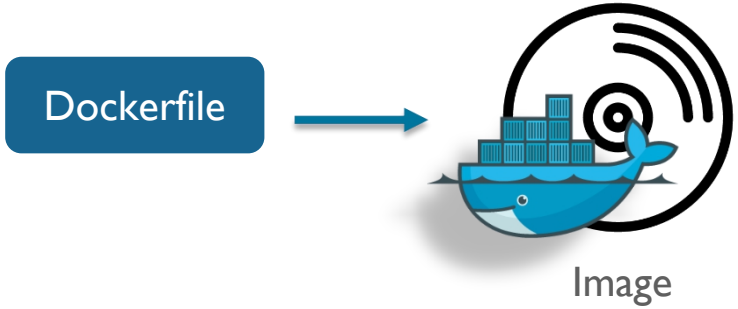
- Docker is a tool designed to **create, deploy, and run** applications with ease by using containers
- It allows a developer **packaging of an application with** all of the requirements such as **libraries and other dependencies**, ship it all as one package
- It ensure that your application **works seamlessly** in any environment be it Development, Test or Production



Why Docker?



Dockerfile



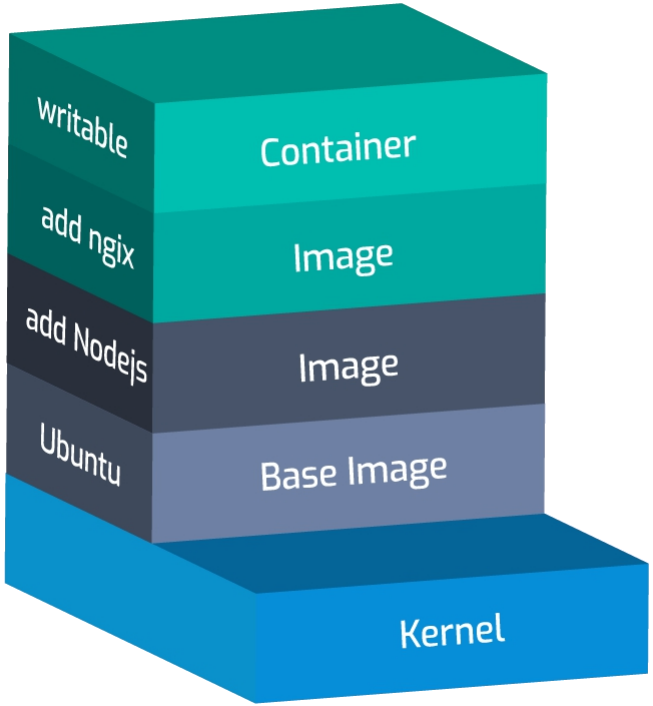
01 Dockerfile is the **basic building block** of Docker containers

02 Dockerfile is a file with **a set of instructions** and forms the basis for any Docker Ima

03 Every time, **base image** is going to **be based upon another image**. You are going to pick up a base image and build up on that image

edureka!

What is an Image?



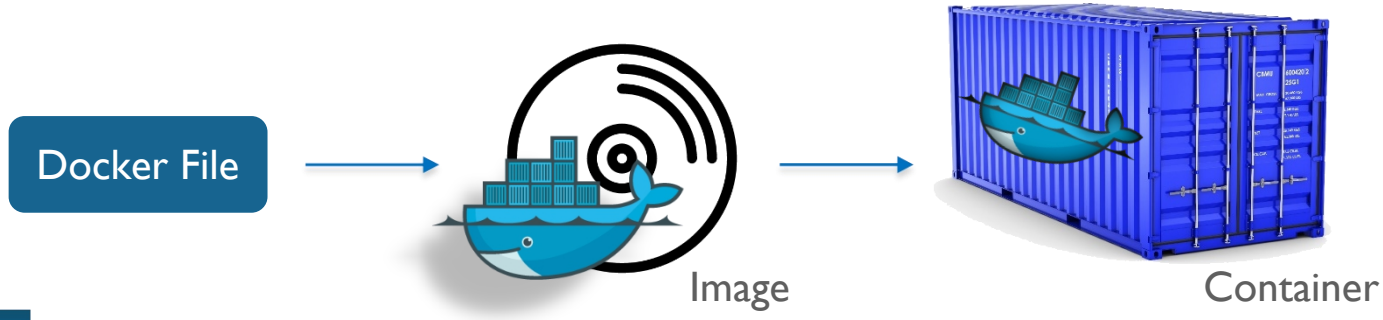
Images An image is a read-only template with instructions for creating a Docker container. Often, an image is based on another image, with some additional customization.

For example, you may build an image which is based on the ubuntu image, but installs the Apache web server and your application, as well as the configuration details needed to make your application run.

You might create your own images or you might only use those created by others and published in a registry.

What is Docker Container?

- To build your own image, you create a Dockerfile with a simple syntax for defining the steps needed to create the image and run it
- Each instruction in a Dockerfile creates a layer in the image. When you change the Dockerfile and rebuild the image, only those layers which have changed are rebuilt
- This is part of what makes images so lightweight, small, and fast, when compared to other virtualization technologies
- A container is a runnable instance of an image.It provides an isolated environment in which an app along with its environment is being run.



Demo 1 – Getting Started With Docker

(Refer Demo doc 1)

edureka!

Azure Container Registry (ACR)

edureka!

Azure Container Registry (ACR)

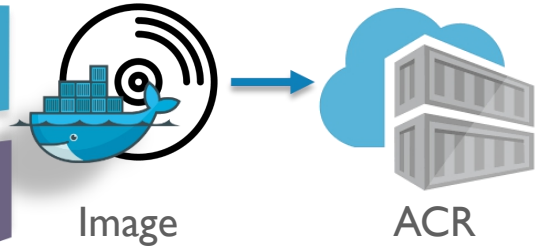
Azure Container Registry is a **managed docker registry** to store and mange your private Docker container images

ACR is available in three SKUs, namely ***Basic, Standard and Premium***

All the images in your registry are ***encrypted at rest***

Geo-Redundant Storage feature in ACR automatically replicates your images to multiple data centre to prevent the ***loss of storage failure*** in a region

Geo-Replication is a feature in premium registry to prevent ***the loss of registry*** in the event of total region failure and just the storage



Azure Container Registry – Use Cases

Pull images from an Azure container registry to various deployment targets:

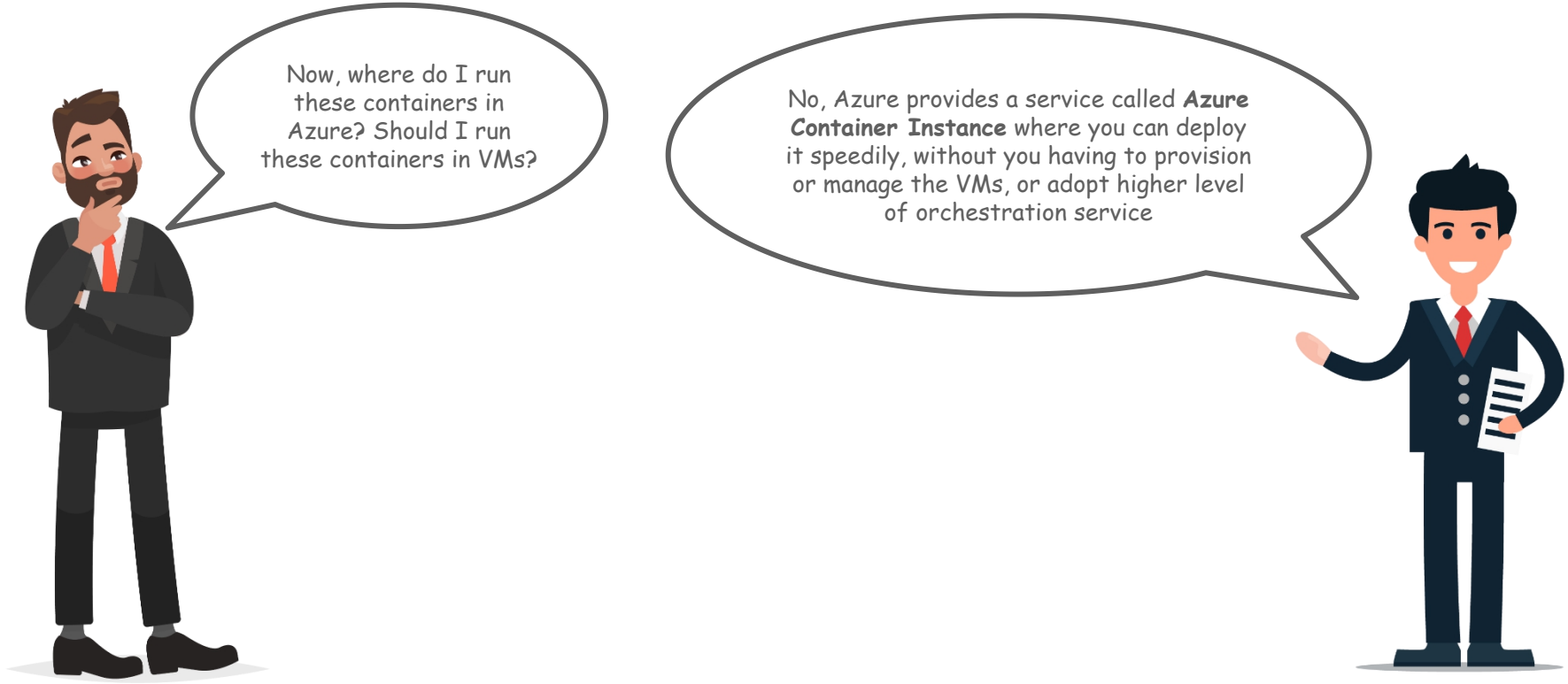
- **Scalable orchestration systems** that manage containerized applications across clusters of hosts, including
  - Kubernetes
  - DC/OS
  - Docker Swarm
- **Azure services** that support building and running applications at scale, including
  - Azure Kubernetes Service (AKS)
  - App Service
  - Batch
  - Service Fabric, and others

Demo 2 – Publish an Image into ACR

(Refer Demo doc 2)

edureka!

Azure Container Instances



edureka!

Azure Container Instances (ACI)

edureka!

Azure Container Instances (ACI)

Azure Container Instance provides a **platform** where we can **run isolated** Docker containers, including simple applications, build jobs and task automation

Benefits of ACI

Domain name and **IP addresses** are attached to the containers and exposed to the internet using it

Can optimise the use of Containers by *specifying* the **exact no. of cores** and **memory** you will be using for application

Persistent storage can be achieved by **mounting the Azure File Share**

Hypervisor level security ensures that your application *isolated* even when it is used by **multi-tenants**

Supports **Scheduling** of multiple containers which share the **common** host, storage, network and lifecycle

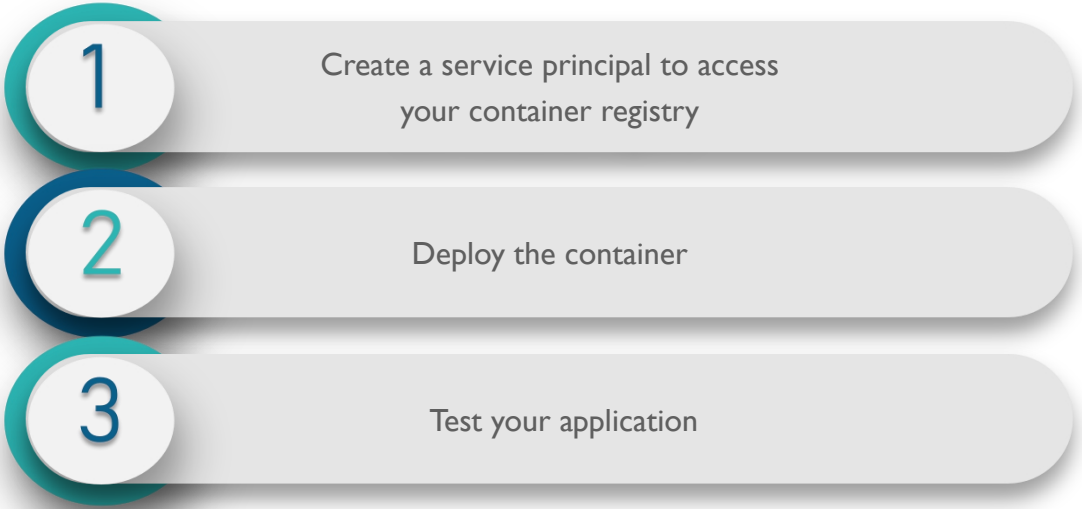
It starts the containers **within seconds**, *without* the need to **provision** or **manage** VMs

edureka!

Demo 3 – Deploy an Application in ACI

edureka!

Demo 3 – Deploy an Application in ACI

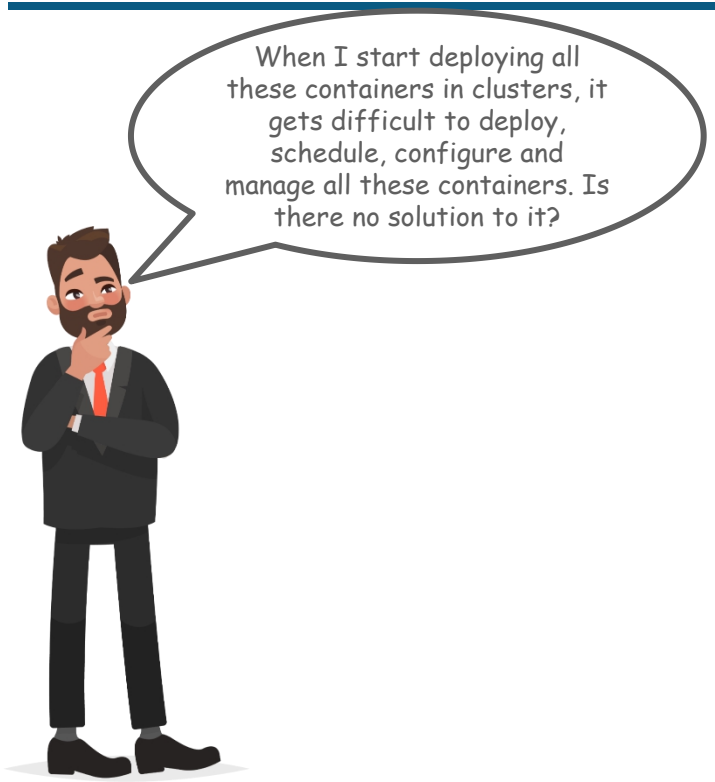


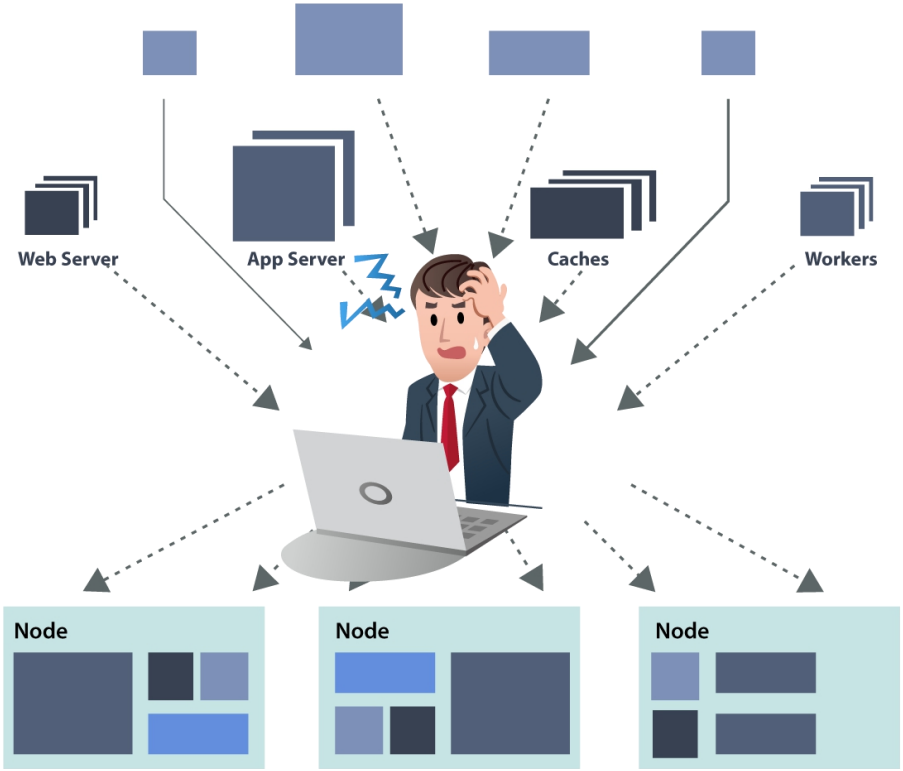
Note: Refer to the Demo doc-3 in LMS to see the detailed steps

Concept of Orchestration

edureka!

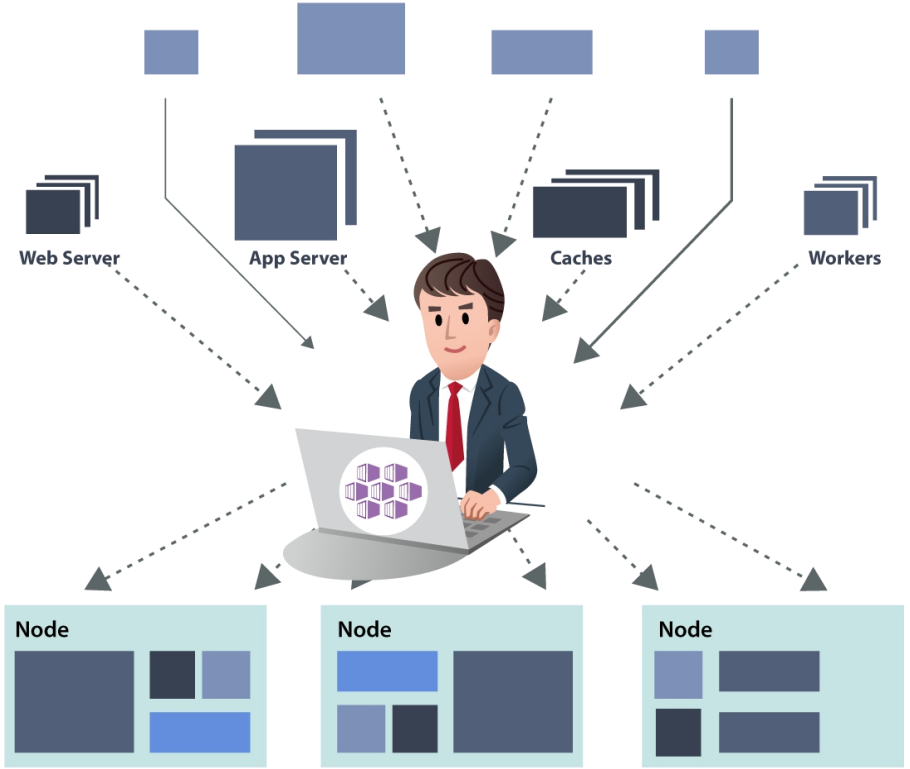
Concept of Orchestration





edureka!

Concept of Orchestration (Cont.)

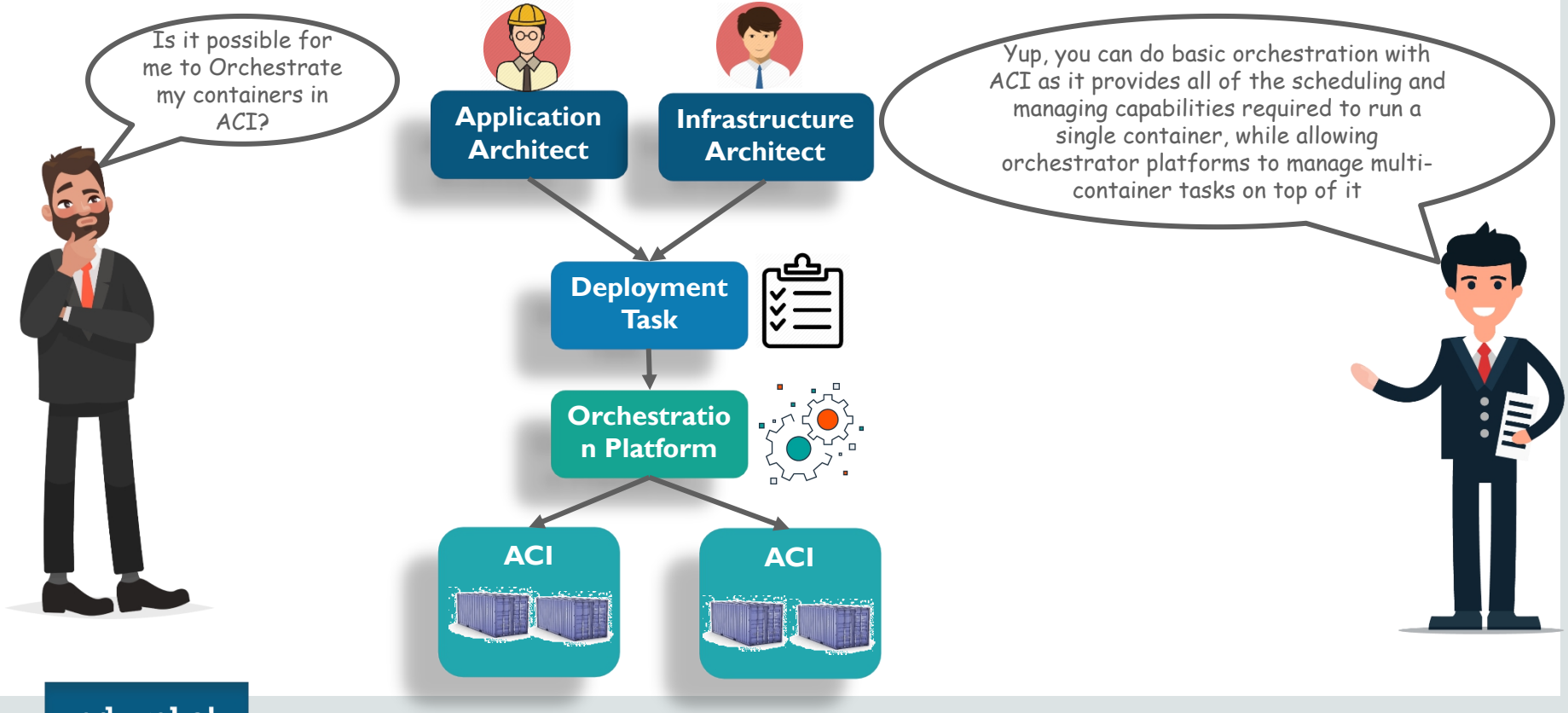


edureka!

Tasks in Traditional Orchestration

1. **Scheduling**: To run a specific container on a machine, specify the image and a resources request for it
2. **Affinity**: Specify the set of resources that you want to run nearby
3. **Health monitoring**: Automatically reschedule the deployment of the containers, if it faces any failures
4. **Scaling**: Add or remove container instance to match the demand
5. **Service discovery**: Helps the containers to locate each other, even when they change the IP address or move between the host
6. **Coordinated application upgrades**: Reduces the downtime by enabling rollback if something goes wrong or if something has to be updated

Orchestration in Azure Container Instances



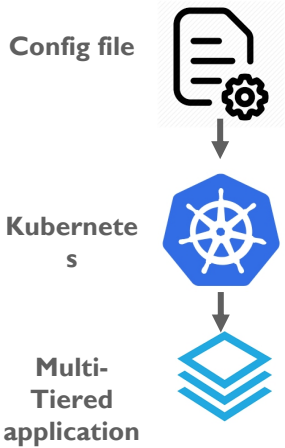
edureka!

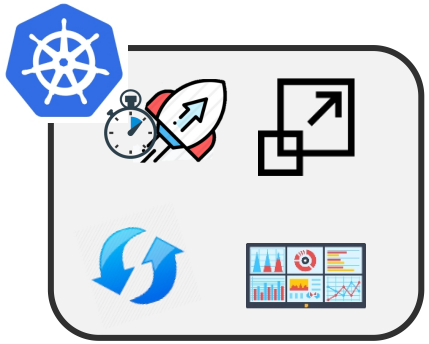
Azure Kubernetes Service (AKS)

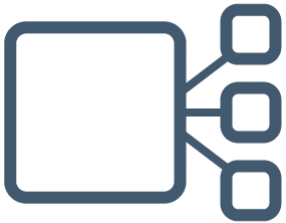
edureka!

What is Kubernetes?

Kubernetes is a platform which is designed for container-based applications to **build a loosely coupled collection** of components which are deployed, maintained and scaled **centrally**







Can be used to deploy a cluster of multi-tiered containers in a single configuration file

Can be quickly deployed, scaled, rolled out and optimise the use of resources

Use Case: Portable and microservices-based applications

edureka!

Azure Kubernetes Service (AKS)

AKS is a service which gives a **managed Kubernetes cluster** in Azure to ***reduce*** the complexity of maintaining and operating the containers by provisioning, upgrading and scaling resources **on demand** *without* any **downtime**

Azure Kubernetes Service has two components

Cluster master

- Managed by Azure
- No cost
- Orchestrates the workload of your application

Nodes

- Managed by Customer
- Pay as you use
- Runs the application workloads

edureka!

Cluster Master

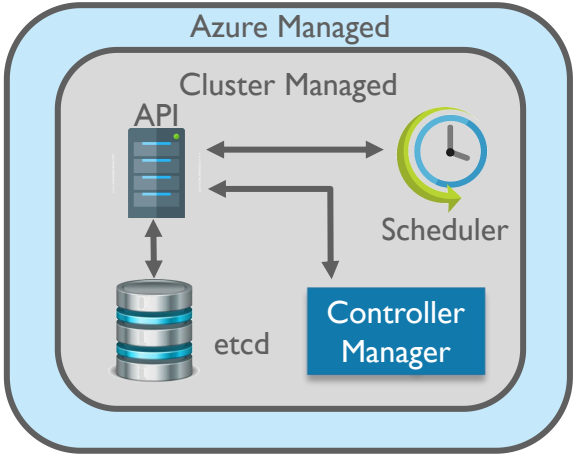
It is a **component** in AKS which gets created and configured as soon as the cluster is launched, which can be abstracted by the user.

kube-apiserver: Keeps account of the Kubernetes APIs and helps in interaction between the components through it using kubectl

etcd: Maintains the state of the cluster and configurations

kube-scheduler: *When you create or scale the application, it determines which node can run it and starts them*

kube-controller-manager: *Controls the smaller controllers that performs the actions such as replicating and handling pod operations*



edureka!

Nodes

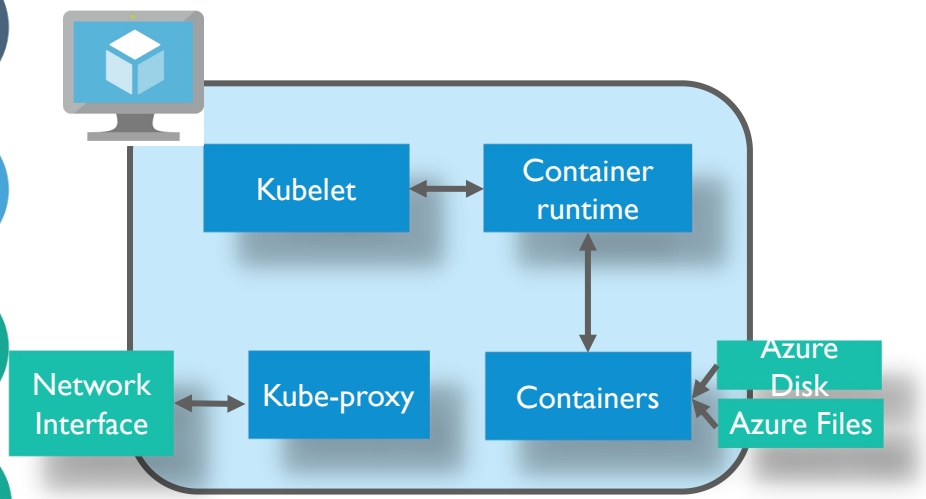
Nodes are the **Virtual machines** which are used to run your application and supporting services.

Kublet is a agent which processes the orchestration request from the master and secludes the task for the requested containers

Kube-proxy is the one which manages the IP for services and routes the network traffic to it on each node

Container runtime is the one which allows the applications to run and interact with additional resources such as storage and network

Size decides how many CPUs, how much memory and storage you require, but if you require more than that, then you can increase the size or scale the nodes accordingly



edureka!

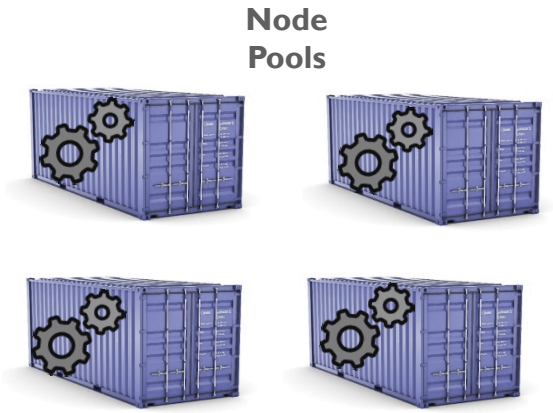
Node Pools

01 Same configuration nodes which are grouped together to form the Node Pools

02 Initial number of nodes and size which you specify while creating a cluster, creates a default node pools

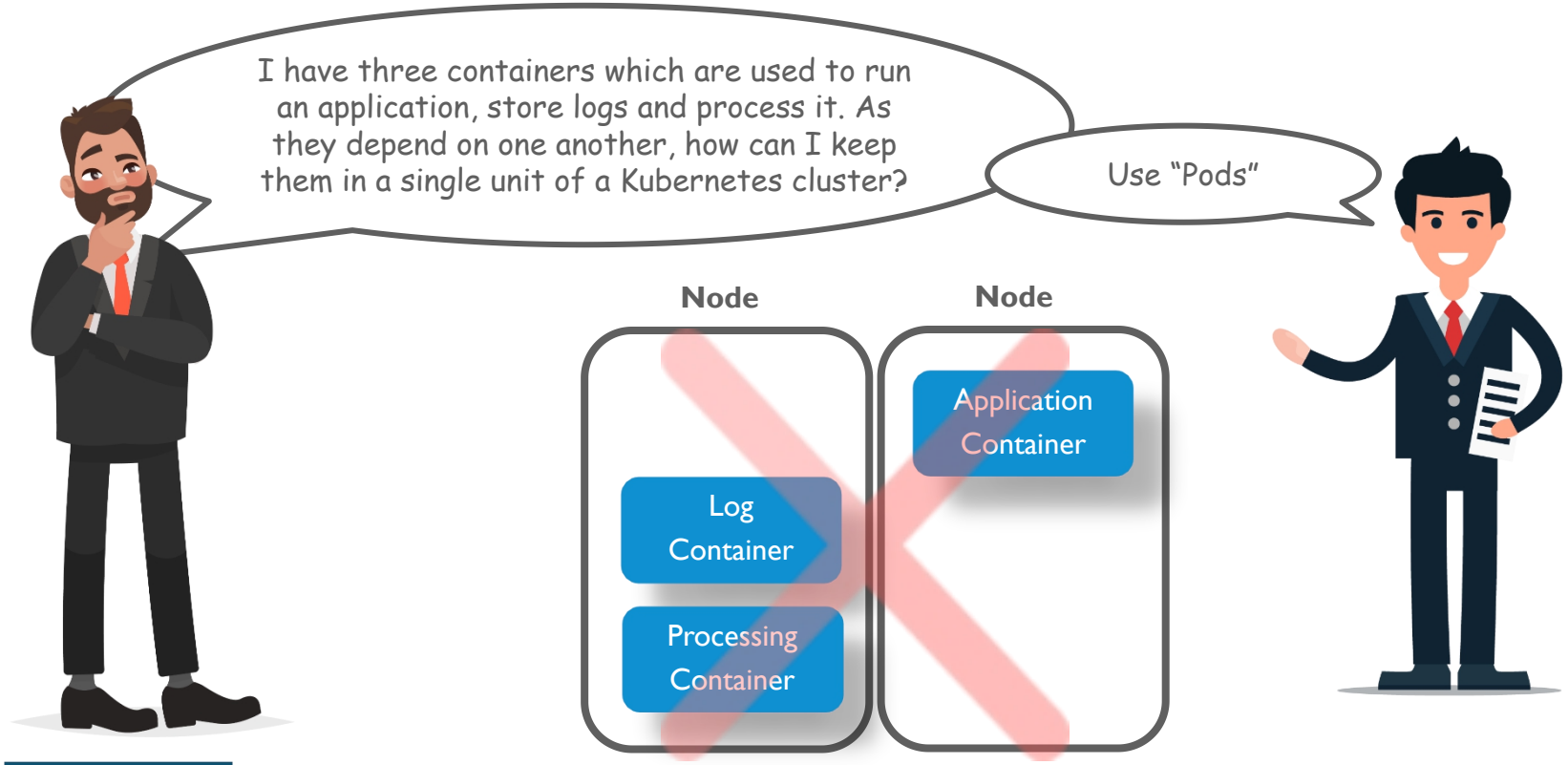
03 Scale and upgrade operation is done on a default node pool

04 Until the nodes are upgraded, the running container are scheduled on other nodes



edureka!

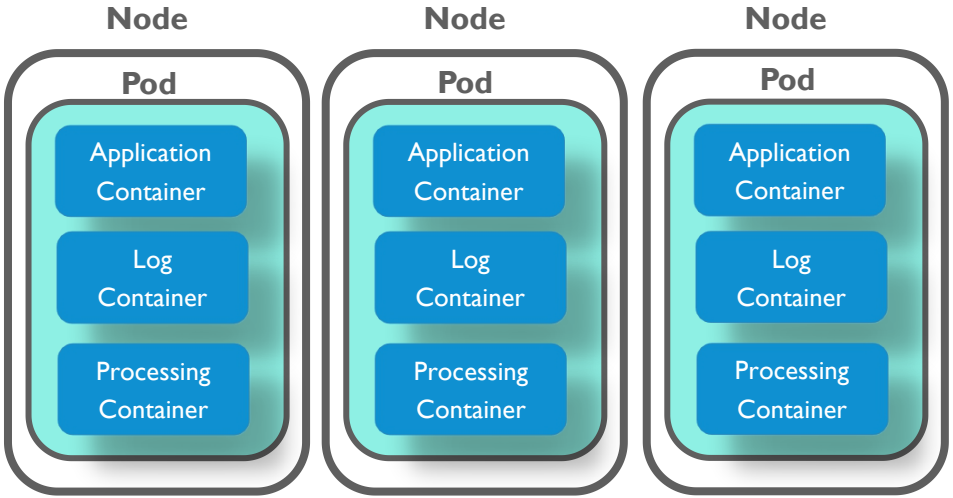
Pods



edureka!

Pods

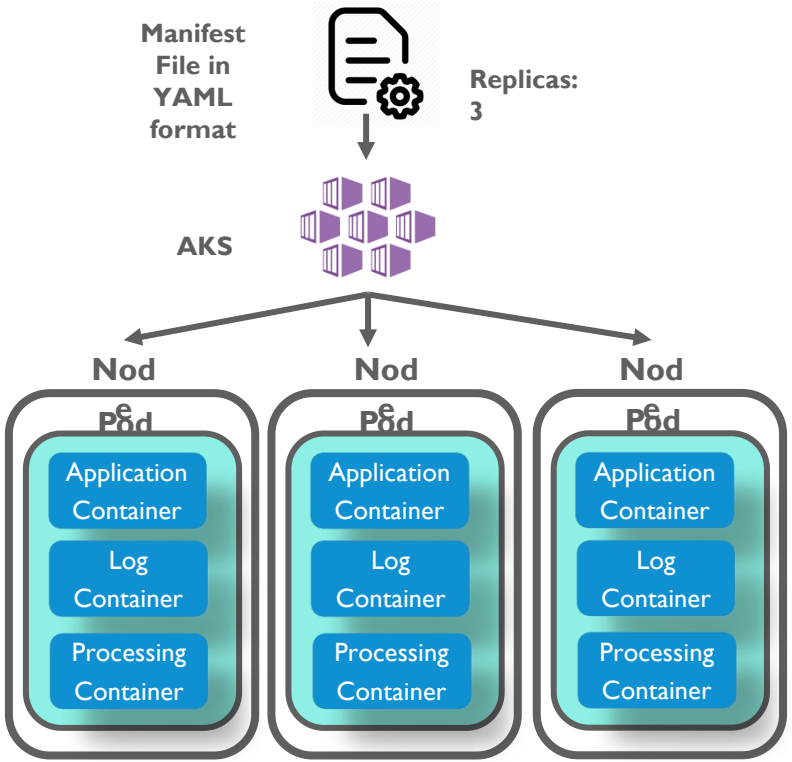
Pods are the **group of containers** which are deployed **together** on the *same* host to run an application on your instance.



Pods are deployed and managed by **Deployment controller**.

Set-up the resource limit to specify the amount of CPU and memory you require and maximum resource

Deployment



- **Deployment** defines the **no. of replicas to create** and **scheduler ensures** the pods are on the **healthy nodes**
- Update on deployment like changing configuration, container image or storage, is done by **deployment controller**
- It **drains and terminates** the **current replicas** and **create new replicas** from new deployment definition
- **Pod Disruption Budgets** defines the minimum no. of replicas that can be down during an update or deployment
- Through **manifest file** you can create a deployment

edureka!

Demo 4 – Create and Run an Application on Kubernetes Cluster

edureka!

Demo4 – Kubernetes Cluster

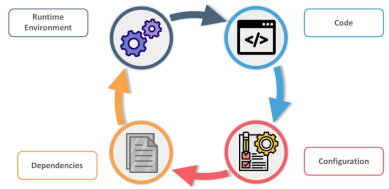
1. Get an Application
2. Create Container Image
3. Create a Container Registry and push the image into it
4. Create a Kubernetes service cluster
5. Run an application in AKS
6. Test your application

Note: Refer to the Demo doc 4 in LMS to see the detailed steps

edureka!

Summary

Application Environment Components



edureka!

Container V/S VM

Parameters	VM	Container
Work Scope	If you want to run multiple applications, **use VM**	If you want to run multiple copies of a single app, use Containers
Portability	Working with VMs are platform independent	Working with containers is platform dependent
Resources	VMs take up a lot of system resources	When **resources are a constraint, use Containers**
Security	You need not have **sudo privileges to import subsystems**	You need to have **sudo privileges to import subsystems**

edureka!

What Is Docker?

"BUILD, SHIP & RUN ANY SOFTWARE ANY WHERE"

- Docker is a tool designed to **create, deploy, and run** applications with ease by using containers
- It allows a developer **packaging of an application with** all of the requirements such as **libraries and other dependencies**, ship it all as one package
- It ensure that your application **works seamlessly** in any environment be it Development, Test or Production

edureka!

Azure Container Registry (ACR)

Azure Container Registry is a **managed docker registry** to store and mange your private Docker container images

ACR is available in three SKUs, namely ***Basic, Standard and Premium***

All the images in your registry are ***encrypted at rest***

Geo-Redundant Storage feature in ACR automatically replicates your images to multiple data centre to prevent the ***loss of storage failure*** in a region

Geo-Replication is a feature in premium registry to prevent ***the loss of registry*** in the event of total region failure and just the storage

edureka!

Azure Container Instances (ACI)

Azure Container Instance provides a **platform** where we can **run isolated** Docker containers, including simple applications, build jobs and task automation

Benefits of ACI

Domain name and **IP addresses** are attached to the containers and exposed to the internet using it

Can optimise the use of Containers by *specifying* the **exact no. of cores** and **memory** you will be using for application

Persistent storage can be achieved by **mounting the Azure File Share**

Hypervisor level security ensures that your application *isolated* even when it is used by **multi-tenants**

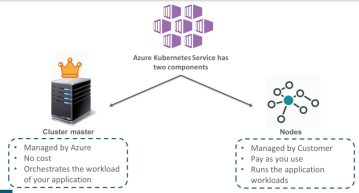
Supports **Scheduling** of multiple containers which share the **common** host, storage, network and lifecycle

It starts the containers **within seconds**, *without* the need to **provision** or **manage** VMs

edureka!

Azure Kubernetes Service (AKS)

AKS is a service which gives a **managed Kubernetes cluster** in Azure to ***reduce*** the complexity of maintaining and operating the containers by provisioning, upgrading and scaling resources **on demand** *without* any **downtime**



edureka!

edureka!

edureka!

Thank You

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