Microsoft Azure Administrator Associate Training (AZ-104)

Module 5

Agenda

01 Azure App Services Web app for Containers

02 Azure App Service Plan

03 Deployment Slots

04 Need for Containerization

05 Docker and Docker Installation

06 Azure Container Registry

07 Kubernetes

08 Azure Kubernetes Service

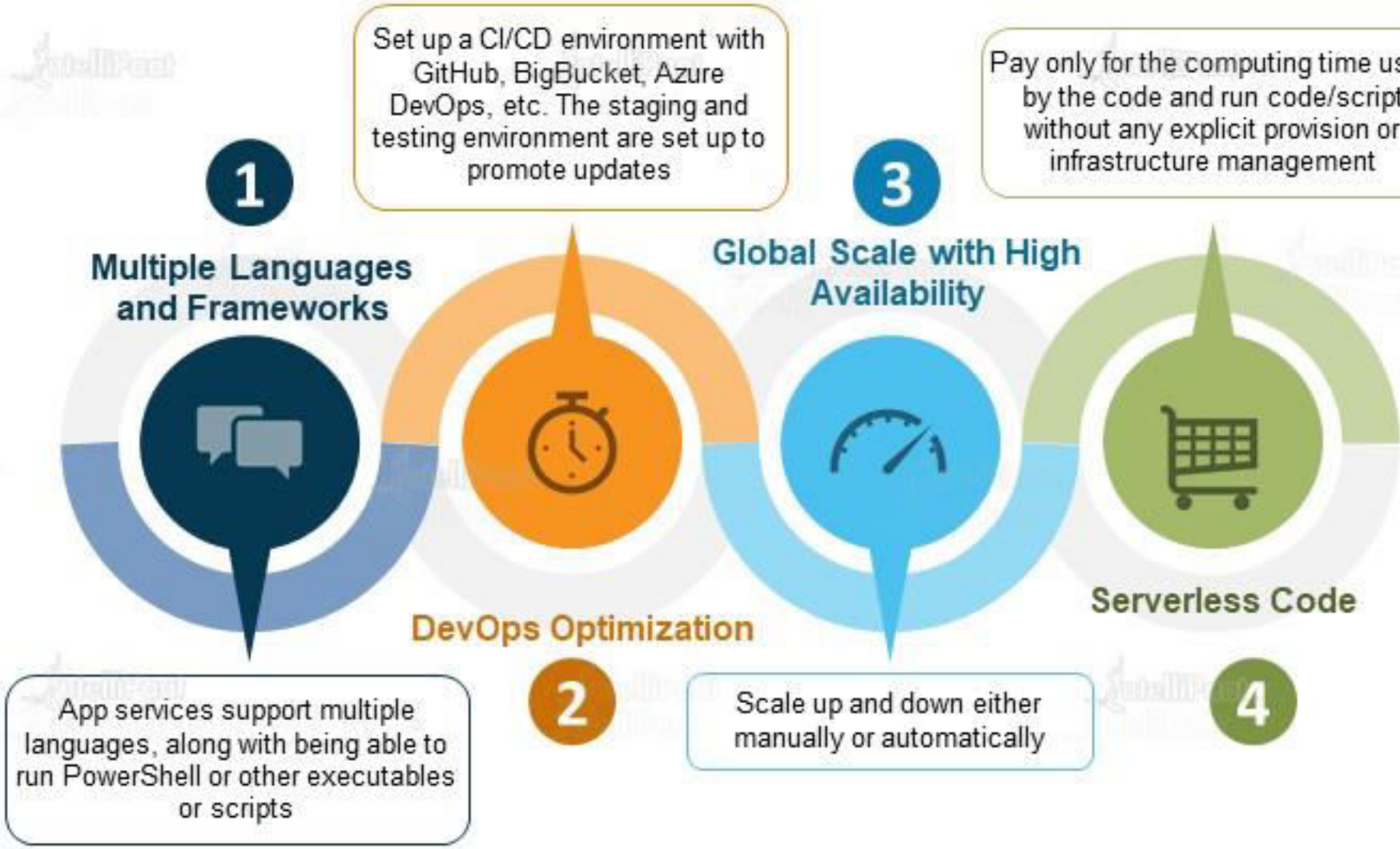
Azure App Services

What are Azure App Services?

HTTP-based services provided by Microsoft Azure to host web applications, mobile backends, and REST APIs

They support a variety of languages such as: .NET, .NET Core, Java, Ruby, Node.js, PHP, and Python

Why Use Azure App Services?



App Service Plan

Each app service runs on an app service plan in the background. It defines a set of compute resources that are required to run the app service

We pay for our application deployment depending on this app service plan

There can be multiple app services that are using the same app service plan

This service is analogous to a server farm in relation to conventional web hosting

App Service Plan Configurations

Region

This defines the region of the deployment of our app. This may be East US, Central US, and so on

Number of VM Instances

This defines the number of virtual machine instances of the application that are to be created.

Size of VM Instances

This is to define the size of the VM instances: Small, medium, or large

Pricing tier

This defines what app service features will be accessible to us and how much we'll have to pay for them. This may be free, shared, basic premium, premiumV2, or isolated

Staging Environment in App Services

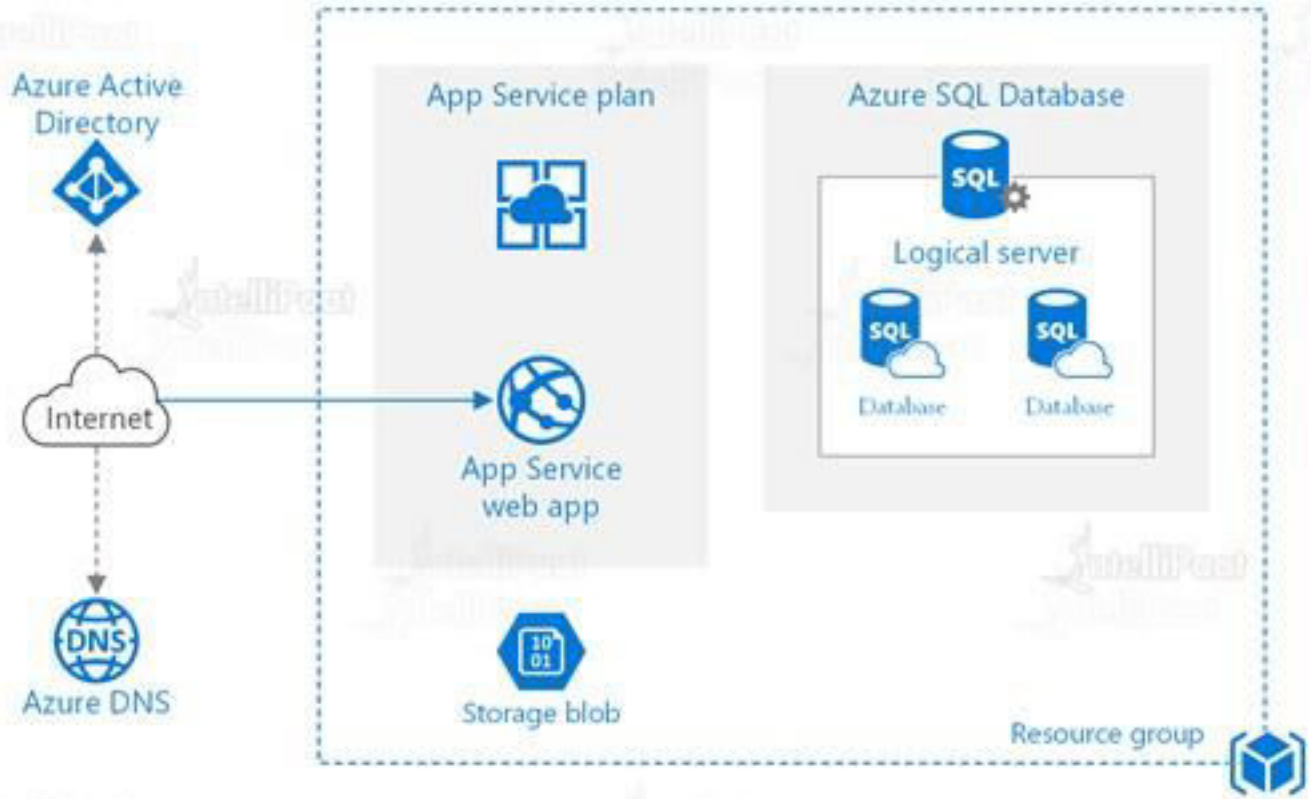
Azure offers a service to set a staging environment in app services by creating a separate **deployment slot** instead of the default production slot

The **swap** function (of deployment slots) is used to replace the previously staged app with the previous production app. Some of the benefits of the staging environment are:

- Changes in the staging deployment slot can be validated before swapping it with the production slot
- Using the swap and auto-swap functions, the downtime may be eliminated while deploying the app. The redirection of traffic is not hindered and no requests are dropped
- We can get our 'last known good site' back by swapping again

Azure App Services: A Use Case

The given architecture of a use case is deploying an app service web application with an app service plan associated with it. It is connected to resources such as storage blob, Azure SQL database, AAD, and Azure DNS



Hands-on: Creating and Configuring an App Service

Hands-on

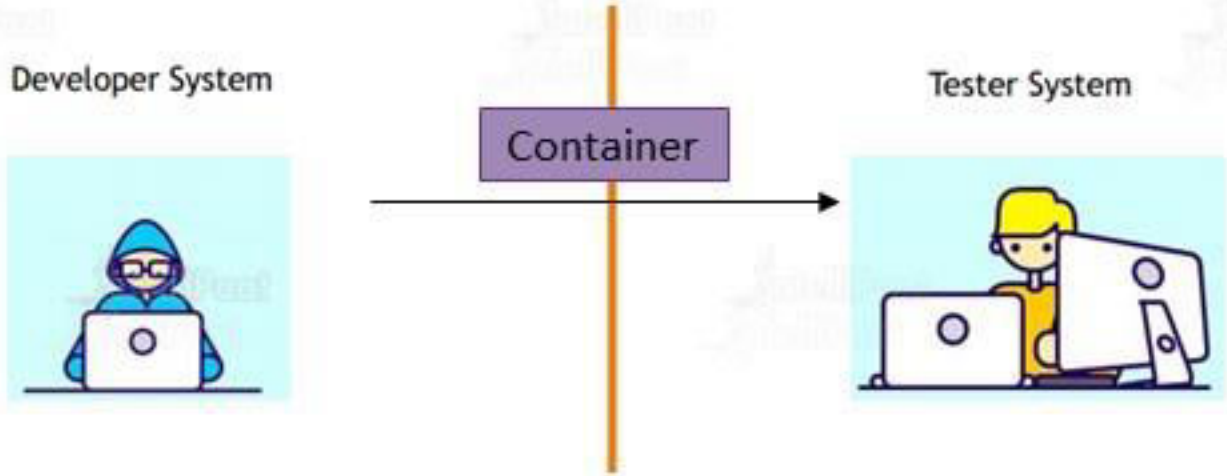
1. Create an App service using Azure Portal

    a) Deploy a simple HTML web application using the service

Azure Container Registry

What is the need for containerization?

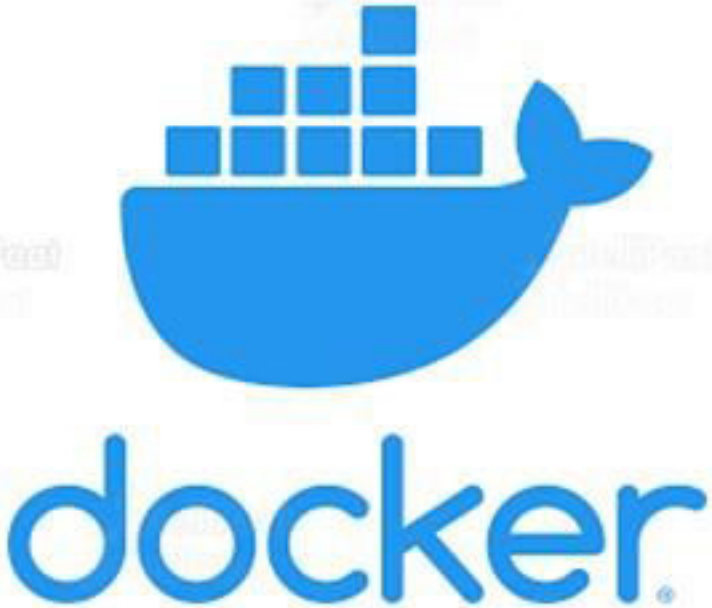
The main purpose of containers is to **solve the environment issues**. The code developed by a developer needs to be run smoothly on the tester's system as well. If this fails, it is usually because of not being able to replicate the environment the code was being run in



The simplest solution is to give the same environment to the tester to run the code. This is where **containers** come in. The developer can easily wrap his/her code in a light-weight container and pass it on to the tester's system

What is Docker?

Docker is one if the most popular tool among all other container tools in the industry. While working with Azure container registry, we deal with the Docker CLI



Docker CLI: Installation

Follow these steps to install Docker on an Ubuntu system. It is a simple two-step process

Step 1

```
sudo apt-get update
```

This command updates apt to have all the recent changes installed in our system

Step 2

```
sudo apt-get install docker.io -y
```

This command installs the Docker files in our system using apt

Docker CLI: Installation

Follow these steps to install Docker on an Ubuntu system. It is a simple two-step process

Step 1

sudo apt-get update

This command updates apt to have all the recent changes installed in our system

Step 2

sudo apt-get install docker.io -y

This command installs the Docker files in our system using apt

Common Docker Operations

Docker images

This commands helps us list all the Docker images downloaded on our system

Docker pull <image-name>

This commands helps us know the installed version of the Docker software on our system

Docker login <login server>

This commands helps us log into the Azure container registry using the login server

Docker tag

This commands helps us create an alias of the image with the fully qualified path to our registry

Common Docker Operations

Docker images

This commands helps us list all the Docker images downloaded on our system

Docker pull <image-name>

This commands helps us know the installed version of the Docker software on our system

Docker login <login server>

This commands helps us log into the Azure container registry using the login server

Docker tag

This commands helps us create an alias of the image with the fully qualified path to our registry

Common Docker Operations

Docker images

This commands helps us list all the Docker images downloaded on our system

Docker login <login server>

This commands helps us log into the Azure container registry using the login server

Docker pull <image-name>

This commands helps us know the installed version of the Docker software on our system

Docker tag

This commands helps us create an alias of the image with the fully qualified path to our registry

Common Docker Operations

Docker images

This commands helps us list all the Docker images downloaded on our system

Docker login <login server>

This commands helps us log into the Azure container registry using the login server

Docker pull <image-name>

This commands helps us know the installed version of the Docker software on our system

Docker tag

This commands helps us create an alias of the image with the fully qualified path to our registry

Common Docker Operations

Docker push <image name>

This commands helps us push the image from our system to the registry

Docker rmi <image name>

This commands is used to remove an image from our system

Docker run <login server>

This commands helps us run containers by their image names

Common Docker Operations

Docker push <image name>

This commands helps us push the image from our system to the registry

Docker rmi <image name>

This commands is used to remove an image from our system

Docker run <login server>

This commands helps us run containers by their image names

Common Docker Operations

Docker push <image name>

This commands helps us push the image from our system to the registry

Docker rmi <image name>

This commands is used to remove an image from our system

Docker run <login server>

This commands helps us run containers by their image names

Azure Container Registry

Azure container registry is a service provided by Microsoft Azure that is a managed, private Docker registry service based on the open-source Docker registry. Using the Azure container registry, we can store and manage private Docker container images

Azure Container Registry: A Use case

The main purpose of the Azure container registry is to pull images to various deployment targets. These deployment targets are set up on the receiving end. The deployment targets can be as follows:

01

Scalable Orchestration Systems are systems that manage the containerization of applications over a cluster of hosts, which are services such as Kubernetes, DC/OS, Docker Swarm, etc.

02

Azure services are the services deployed by Microsoft Azure such as Azure Kubernetes services, app services, etc., which support building and scaling up running applications

Hands-on: Pushing and Automating Image Deployment to Azure Container Registry

Hands-on

1. Deploy the Azure Container Registry on Azure Portal

2. Login to Azure Portal from an Ubuntu Machine

3. Connect to Azure Container Registry. Pull an image and Push it to the newly created registry.

    a) The image will reflect on the Azure Portal

Azure Kubernetes Services

Introduction to Kubernetes

Kubernetes is an open-source orchestration software for deploying, managing, and scaling containers

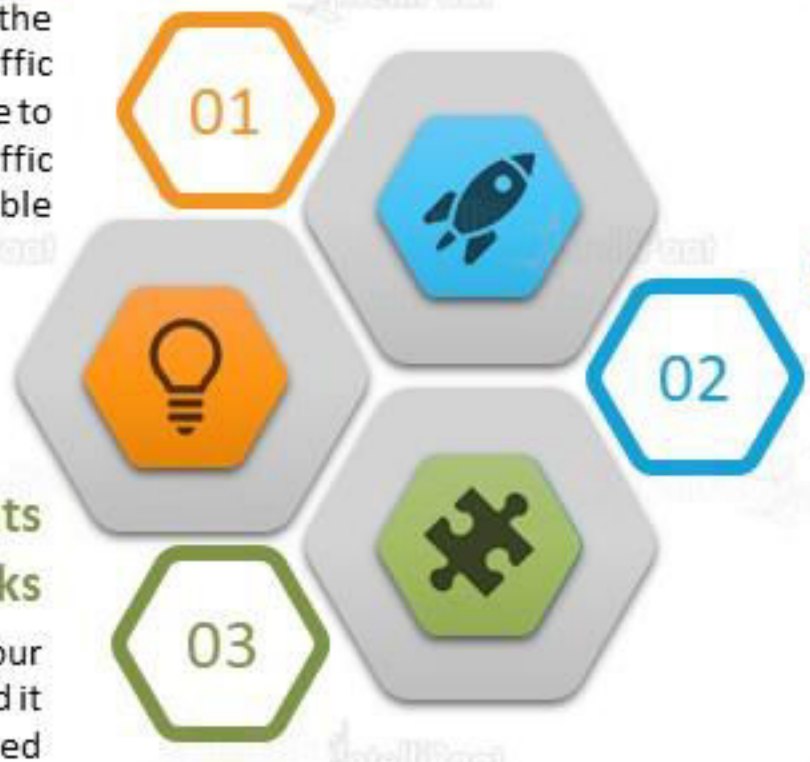
In a production environment, we need to manage the containers that run our applications and ensure that there is no downtime. This is where Kubernetes comes into picture

Kubernetes provides us with a framework to run distributed systems resiliently

Features of Kubernetes

Service Discovery and Load Balancing

Kubernetes can expose a container using the DNS name or using its own IP address. If traffic to a container is high, Kubernetes is able to load balance and distribute the network traffic so that the deployment is stable



Storage Orchestration

Kubernetes allows us to automatically mount a storage system of our choice, such as local storages, public cloud providers, and more

Automated Rollouts and Rollbacks

We can describe the desired state for our deployed containers using Kubernetes, and it can change the actual state to the desired state at a controlled rate

Kubernetes Cluster Architecture

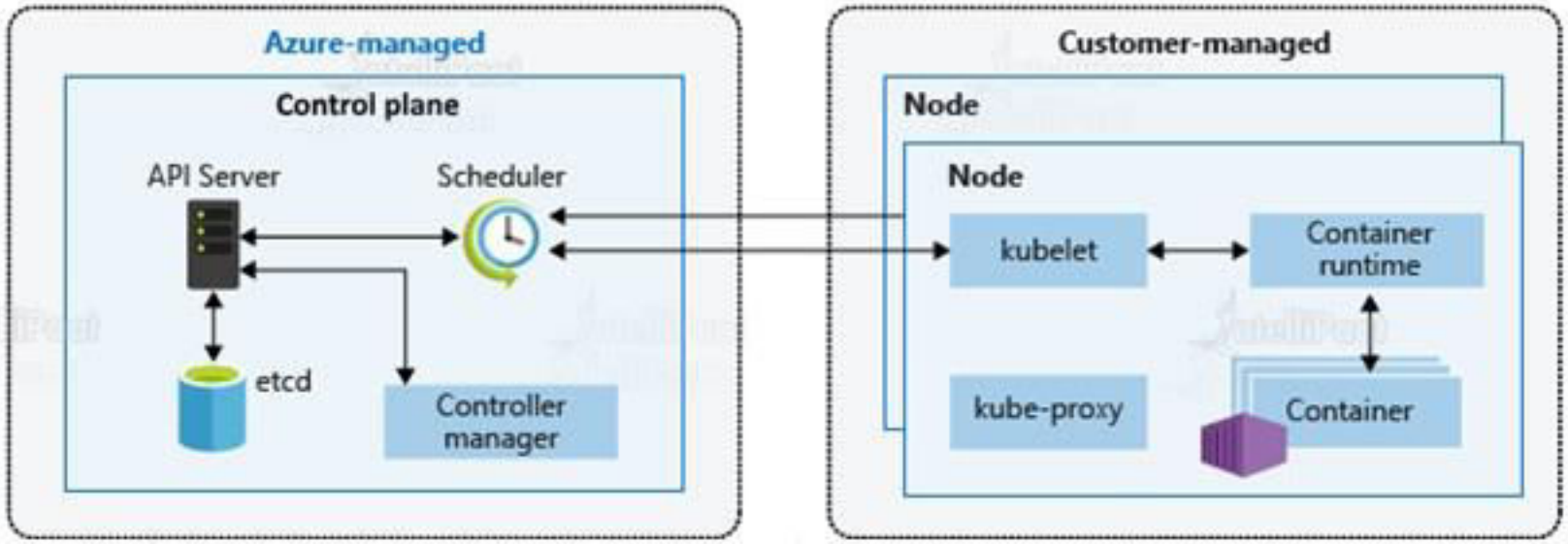
A Kubernetes cluster consists of two parts:

Control Plane

Control plane provides management of the cluster orchestration of core services

Nodes

Nodes run the application workloads



Kubernetes Cluster Architecture

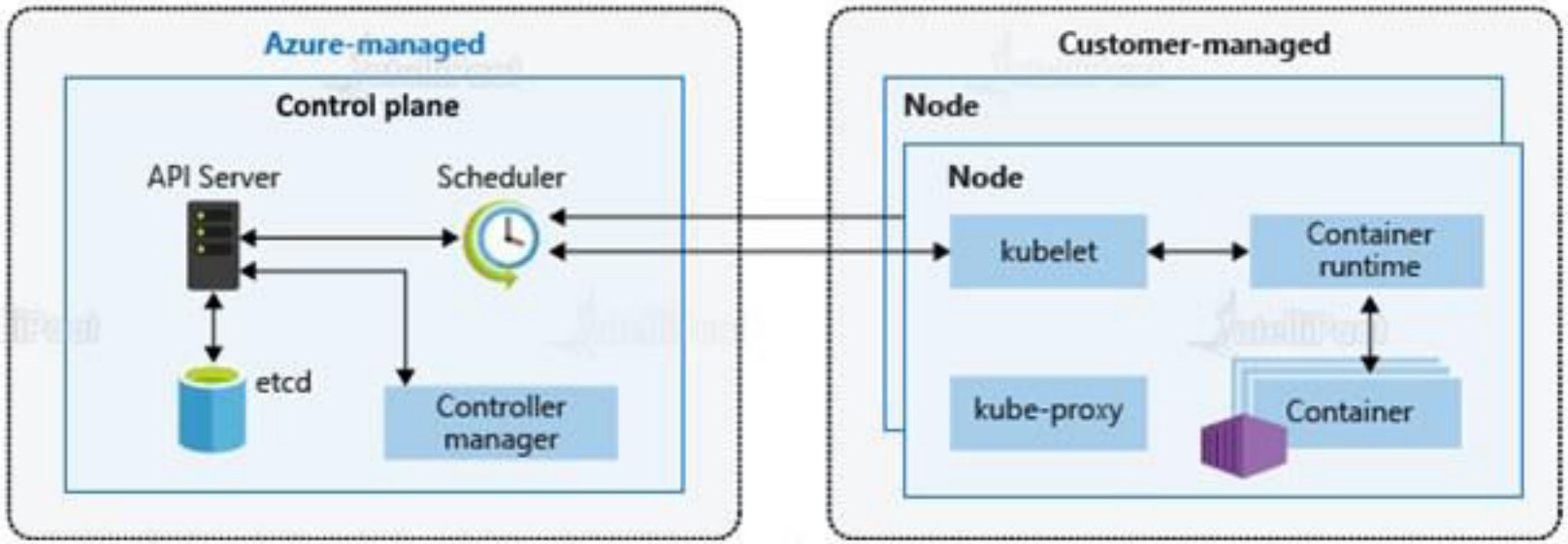
A Kubernetes cluster consists of two parts:

Control Plane

Control plane provides management of the cluster orchestration of core services

Nodes

Nodes run the application workloads

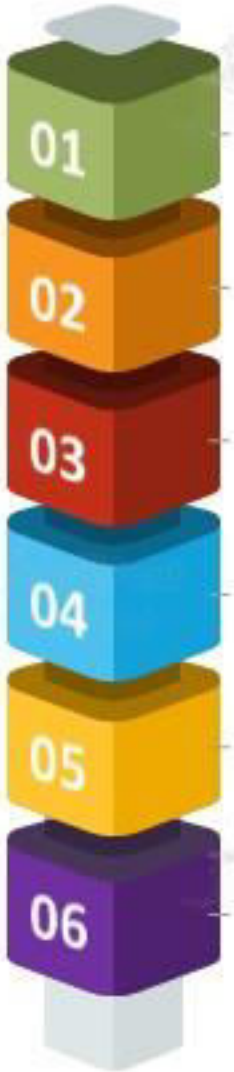


Azure Kubernetes Services

Azure Kubernetes service (AKS) is a service provided by Microsoft Azure that manages a hosted Kubernetes environment

We need not have expertise in container orchestration to be able to quickly and easily manage and deploy containerized applications

AKS Configurations



Basics

Minimal amount of information asked to create a Kubernetes service , e.g., Kubernetes cluster name, Kubernetes version, DNS name prefix, and the primary node pool

Authentication

To control user access to the cluster. It also looks after what all options a user has once authenticated

Monitoring

To enable monitoring capabilities and get insights into the performance and health of our Kubernetes cluster

Scale

To allow flexible capacity and scaling options within our cluster. This can be done by enabling virtual nodes and VM scale sets

Networking

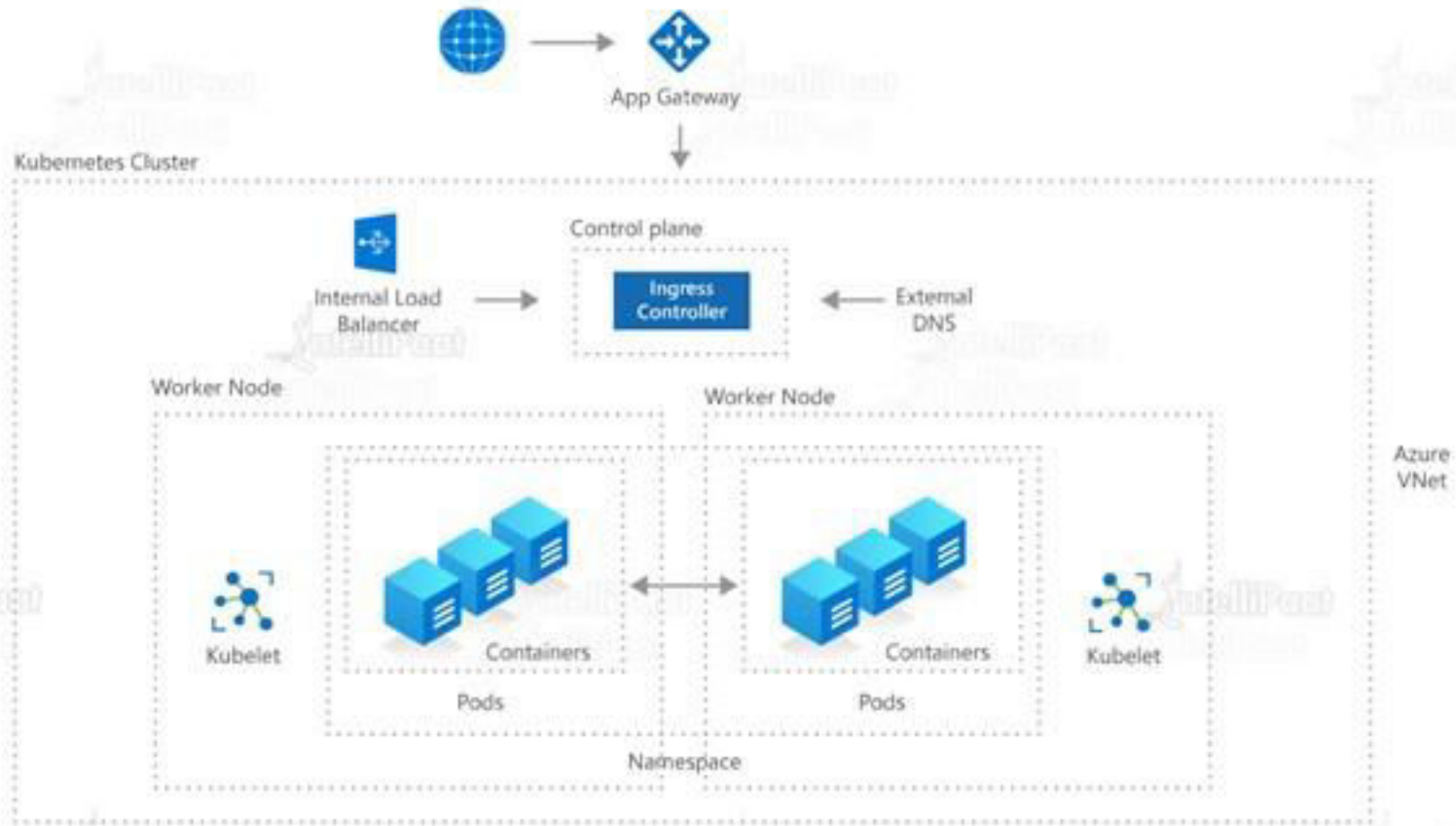
To manage and monitor our cluster's networking settings. We can enable the HTTP application routing or change the VNet configurations as basic or advanced

Tags

To set key–value pairs or labels to put on our clusters, making it easy to organise and filter our clusters later

Azure Kubernetes Services: A Use Case

Here is an example of the architecture of a Kubernetes cluster deployed in an Azure VNet. It contains a pool of two containers and each is managed using a control plane and an internal load balancer



Hands-on: Configuring Azure Kubernetes Service

Hands-on

1. Deploy Azure Kubernetes Service on Azure Portal

    a) Create a single cluster and add nodes to it

2. Deploy your application on the primary node and check if it's running

India: +91-7847955955

US: 1-800-216-8930 (TOLL FREE)

support@intellipaat.com

24/7 Chat with Our Course Advisor