Caltech | Center for Technology & Management Education

Post Graduate Program in Cloud Computing

Powered by simplilearn

Caltech | Center for Technology & Management Education

PG CC - Microsoft Azure Architect Technologies: AZ:303

Powerd by simplilearn

Implement Container-Based Applications

Learning Objectives

By the end of this lesson, you will be able to:

- Implement a container image
- Configure an image with Docker hub
- Configure the Azure Kubernetes Service
- Configure an Azure container registry and its instances

A Day in the Life of an Azure Architect

You are working for an organization as a Cloud Architect that works with containers as they provide significant benefits over a VM.

You are asked to advise an azure solution that can help assist a developer in your company in deploying containers on the Microsoft Azure public cloud without having to provision or manage any underlying infrastructure.

Additionally, the organization is looking for a solution that must include serverless Kubernetes, a seamless CI/CD experience, and enterprise-grade security and governance.

To achieve all the above along with some additional features, we will be learning a few concepts in this lesson that will help you find a solution for the given scenario.

Virtualization and Containers

Virtualization and Containers

The following figure shows the difference between virtualization and containers:

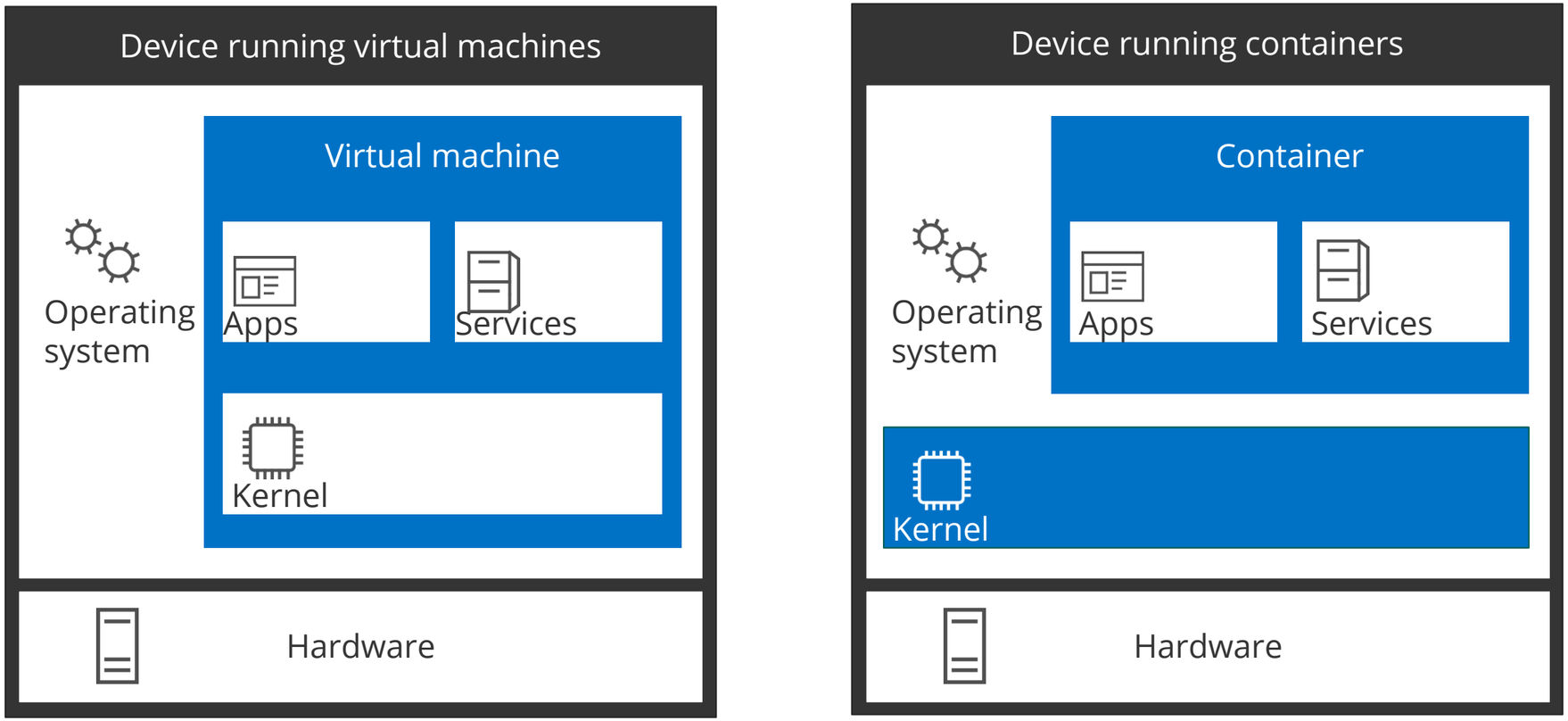


image source: https://docs.microsoft.com/en-in/

Containers

A container is a loosely isolated environment that allows the user to build and run software packages.

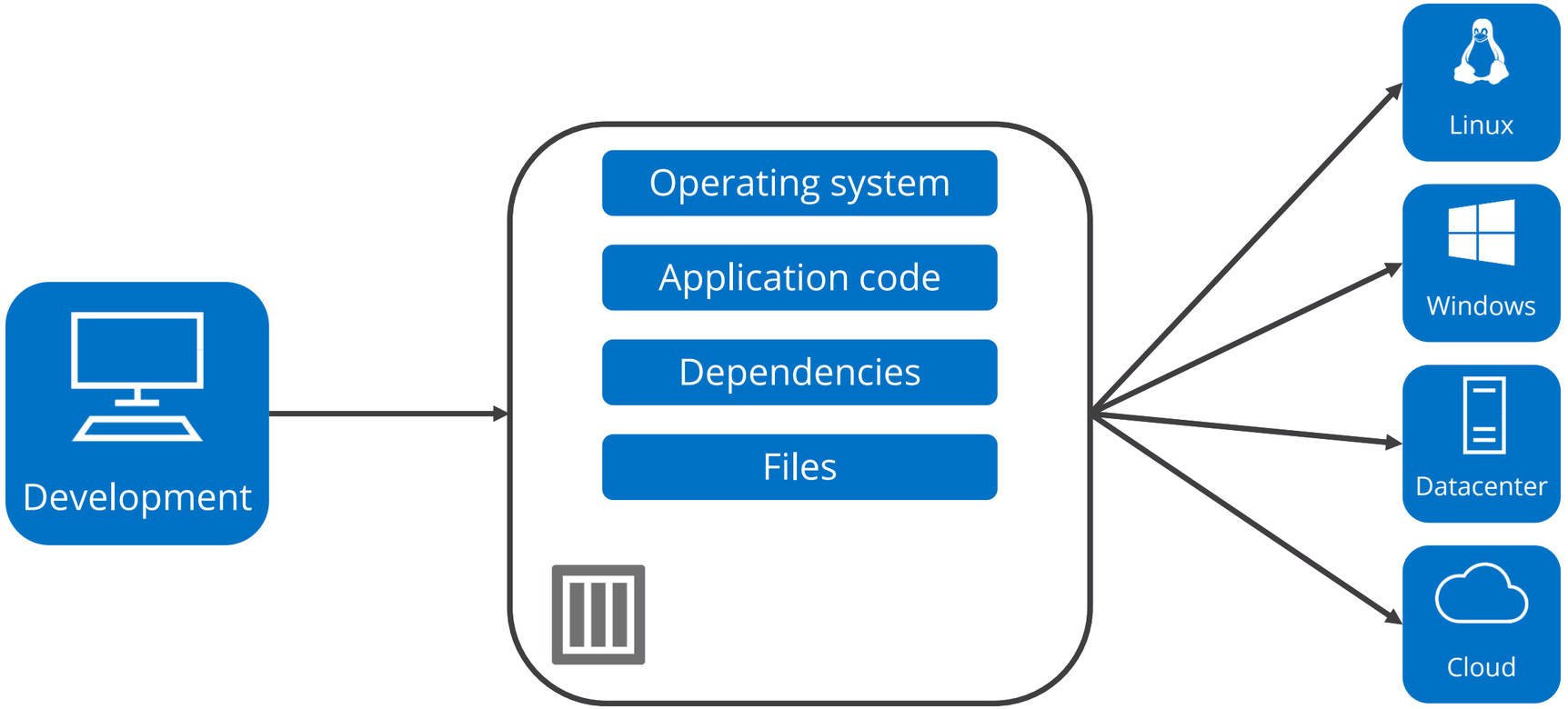
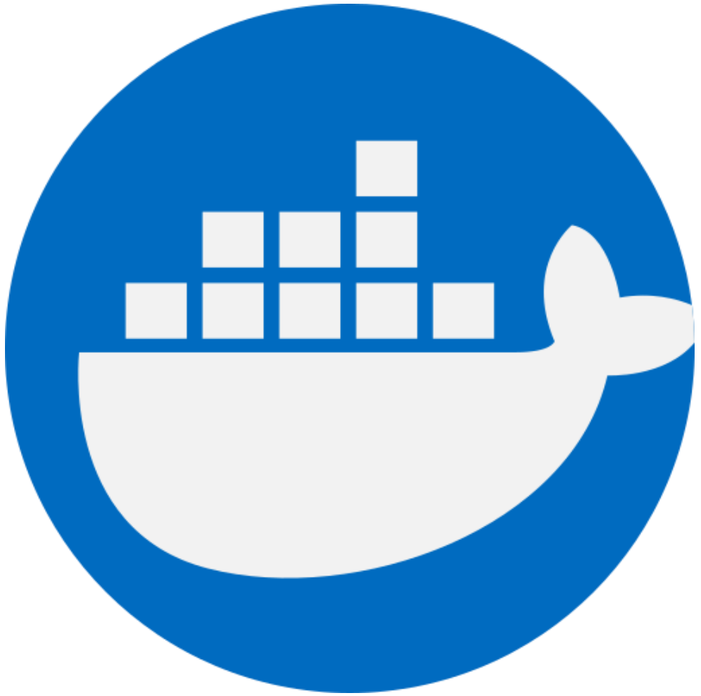


image source: https://docs.microsoft.com/en-in/

Docker

Docker allows a user to run an application in a container, which is a loosely isolated environment. It doesn't require a hypervisor because it operates on top of an operating system.

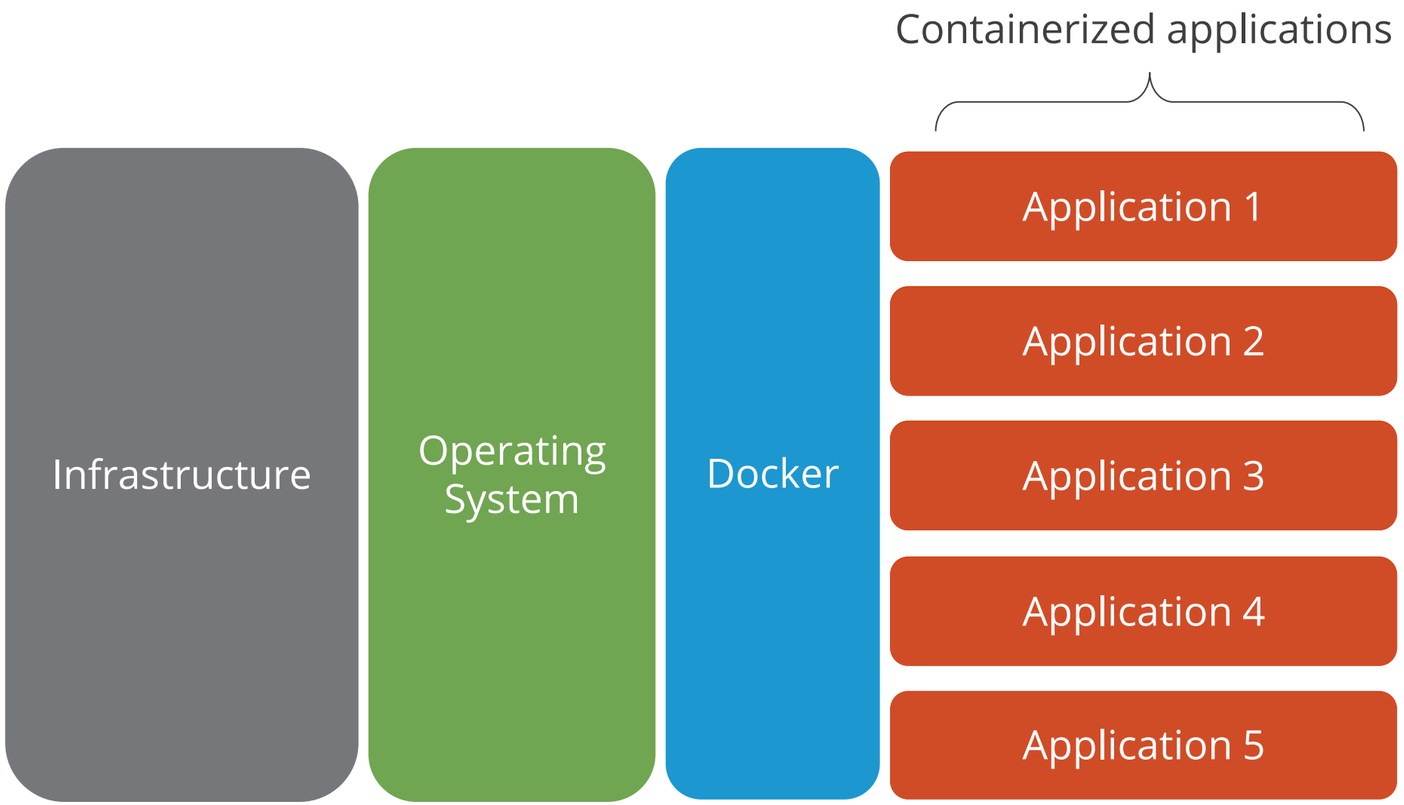


Docker can run on anywhere:

- Desktop or laptop
- Server environment
- DevOps tools
- Cloud services

Docker: Structure of a system

The following figure shows the structure of a system using docker:



Docker: Terminologies

Container

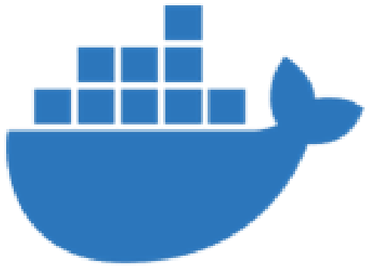
A standardized unit of software that contains everything required for an application to run

Container Image

A template that can be used to create one or more containers

Dockerfile

A text file that contains instructions required to build a Docker image



Build

The process of creating a container image using a set of instructions

Push

The process of downloading a container image from a container registry

Pull

The process of uploading a container image to a container registry

Retrieving a New Container Image

To retrieve a new container image from the Docker Hub, run the following commands:

```
# Get Ubuntu container image
docker pull ubuntu

# Get version 5.7 of MySQL container image
docker pull mysql:5.7

# Get the latest version of the nginx container image
docker pull nginx:latest
```

Container image name

Container image tag

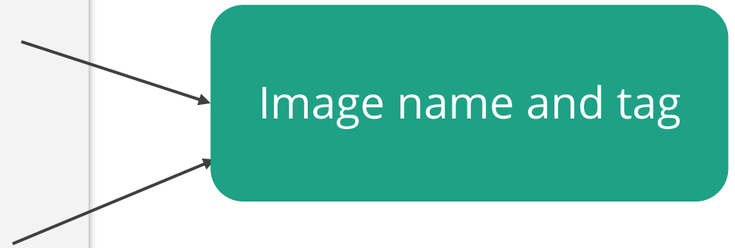
Running the Retrieved Container Image

Commands to run the container image which is retrieved using Docker Hub

```
# Get the .NET application sample container image
docker pull mcr.microsoft.com/dotnet/core/samples:dotnetapp


# Run your container locally
docker run mcr.microsoft.com/dotnet/core/samples:dotnetapp


# View running containers
docker container ls -a
```



Creating a Container Image

To create a container image specification with a Dockerfile:

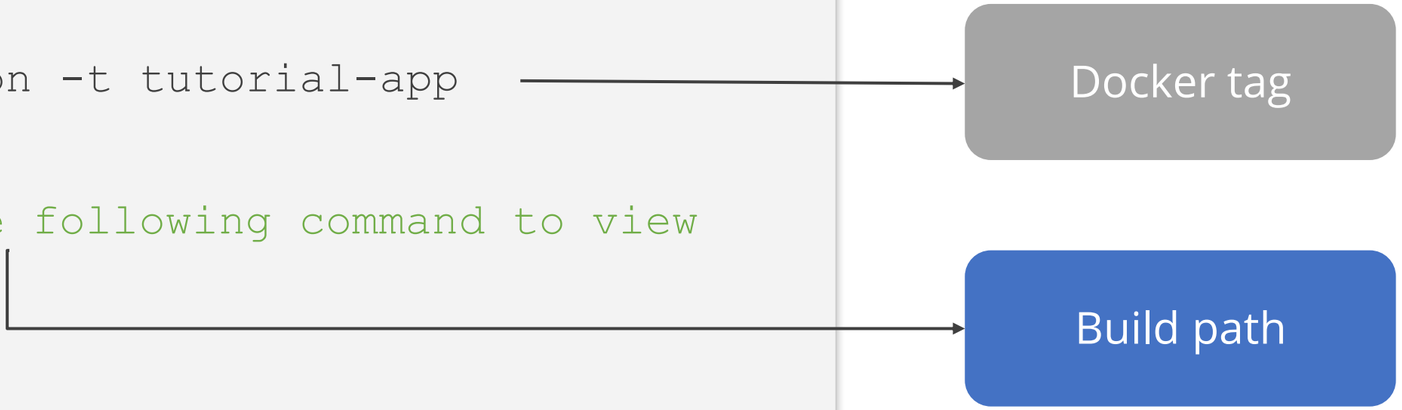
Commands	Description
FROM `node:8.9.3-alpine`	Start with the container image
RUN `mkdir -p /usr/src/app`	Run this command
COPY `./app/ /usr/src/app/`	Copy these files from the host
WORKDIR `/usr/src/app`	Change the working directory
RUN `npm install`	Install node package manager
CMD `node /usr/src/app/index.js`	Start the container with this command

Building the Container Image

To build the custom container image, run the following commands

```
# Build your container
docker build ./application -t tutorial-app


# After building, use the following command to view
your new container image
docker images
```



Running the Custom Container Image

To run the custom container image as a container, run the following commands

```
# Run your container locally
docker run -d -p 8080:80 tutorial-app

# View running containers
docker container ls -a
```

Assisted Practice

Create a Container Instance

Duration: 10 Min.

Problem Statement:

You've been assigned the task of creating a Container Instance to assist a developer in your company in deploying containers on the Microsoft Azure public cloud without having to provision or manage any underlying infrastructure.

Assisted Practice: Guidelines

Steps to create a container instance are:

1. Login to your Azure portal
2. Click on Create a resource
3. Select Containers and click on Container Instances
4. Create a Container Instance

Azure Kubernetes Service

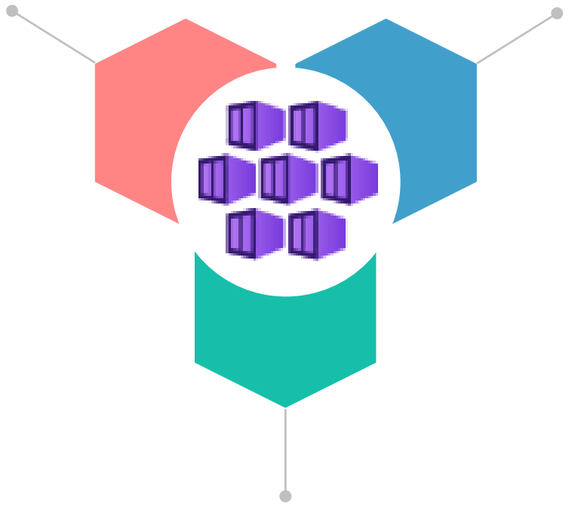
Azure Kubernetes Service

Azure Kubernetes Service (AKS) manages your hosted Kubernetes environment and makes it simple to deploy and manage containerized applications in Azure.

Benefits of Azure Kubernetes Service

Manages container-based applications:

- Along with networking and storage requirements
- Focused on application workloads instead of infrastructure components



Describe applications declarative:

- Uses YAML files to describe an application
- Kubernetes handles management and deployment

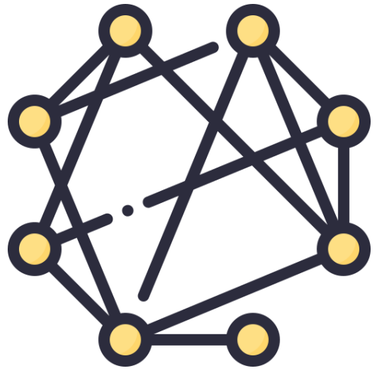
Makes it easier to orchestrate large solutions using a variety of containers like:

- Application container
- Storage container
- Middleware container

Kubernetes Cluster Architecture

Kubernetes cluster architecture consists of the following components given:

Cluster Master



Nodes

Kubernetes Cluster Architecture

The following diagram shows how nodes are shared between customer-managed and Azure-managed workloads:

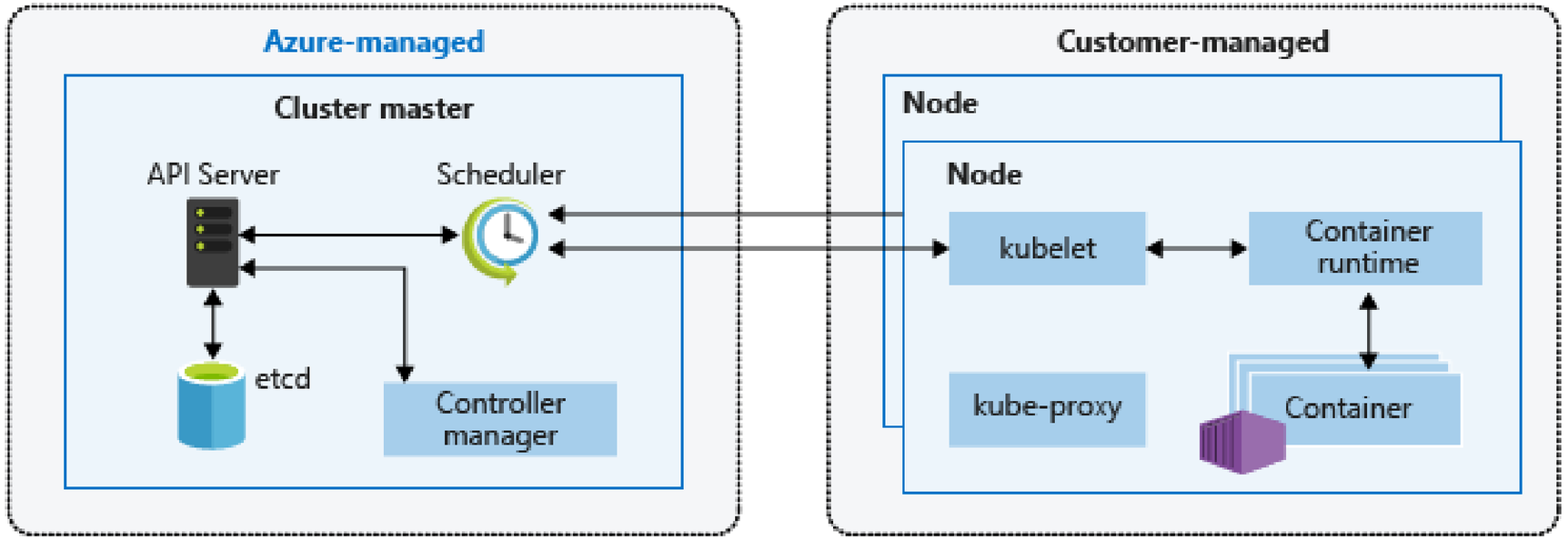
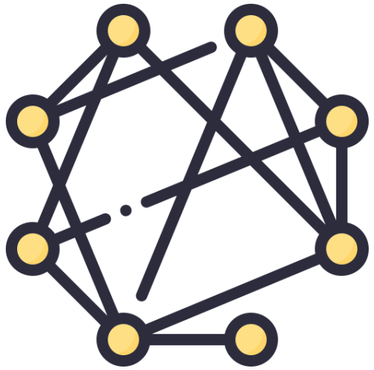


image source: https://docs.microsoft.com/en-in/

Kubernetes Nodes

Each node in Kubernetes is an Azure virtual machine (VM) that contains:



Nodes

- Kubernetes node components needed to communicate with the cluster master and the internet
- Container docker for your applications

Kubernetes Nodes

The following diagram shows is the structure of a Kubernetes node:

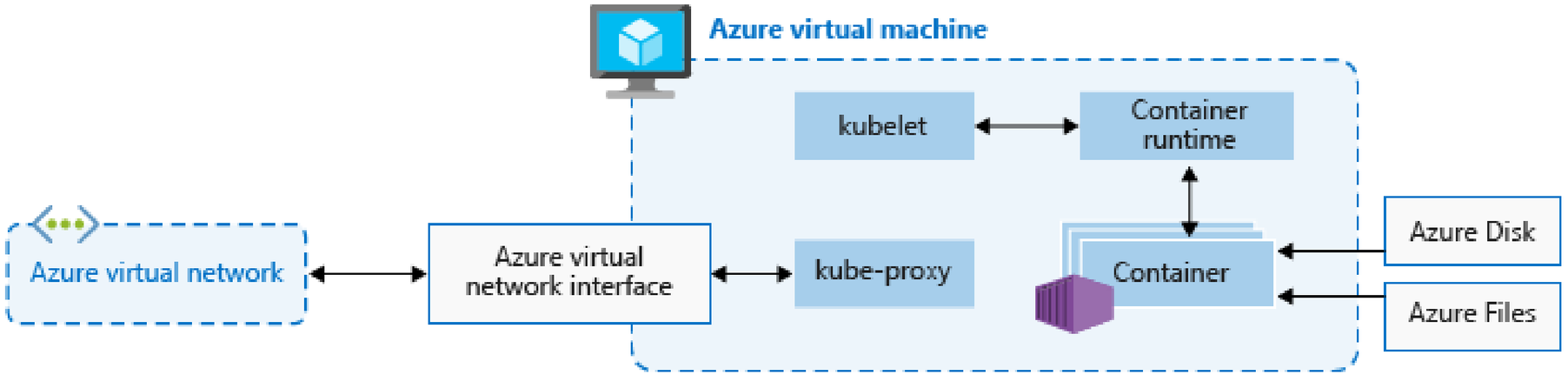


image source: https://docs.microsoft.com/en-in/

Kubernetes Terminology

Nodes

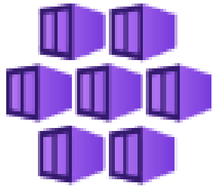
Individual VM running containerized applications

1

Pools

2

Group of nodes with identical configurations



Pods

3

Single instance of an application, A single node can contain multiple containers

Manifests

4

YAML file describing a deployment

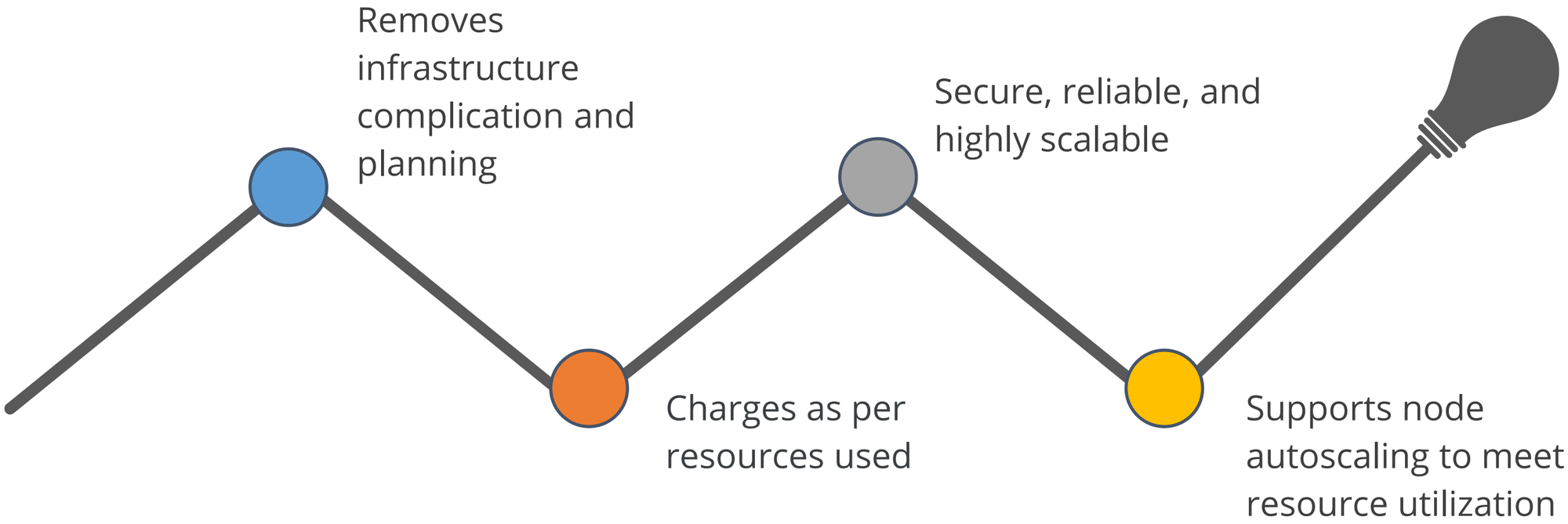
Deployments

5

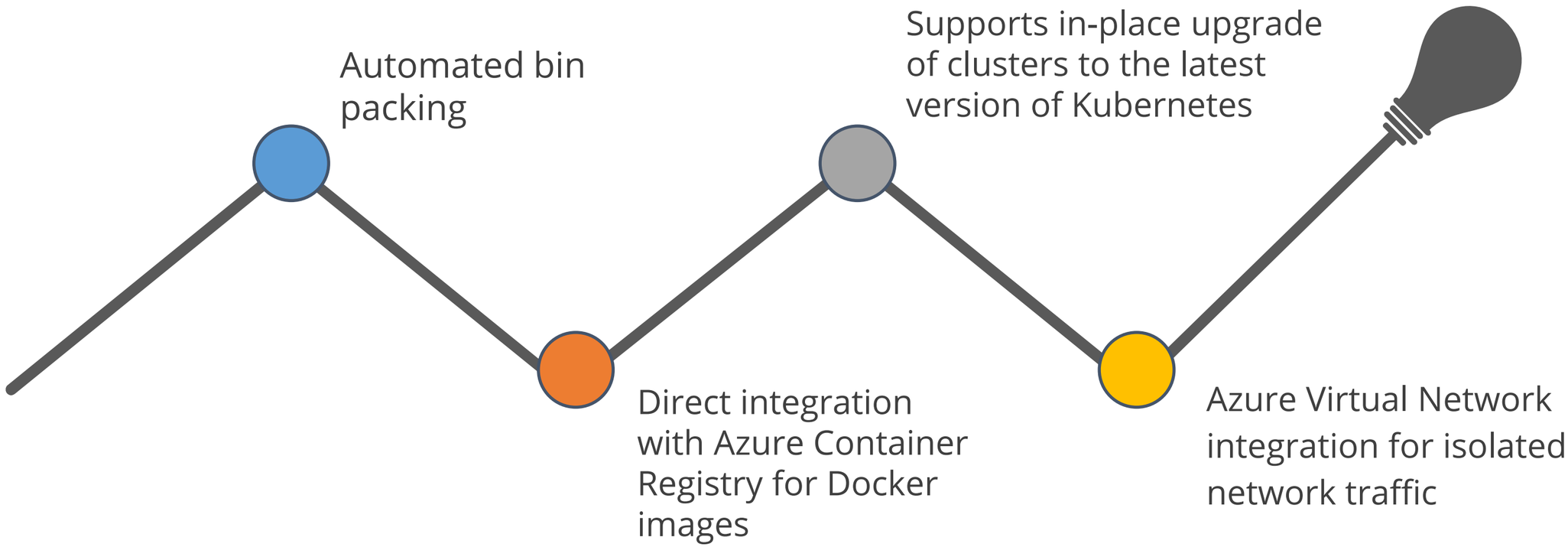
One or more identical pods managed by Kubernetes

Kubernetes Service Features

Various features of Azure Kubernetes Service:



Kubernetes Service Features



Assisted Practice

Azure Kubernetes Service **Duration: 10 Min.**

Problem Statement:

You've been assigned a project to implement Azure Kubernetes Service, which includes serverless Kubernetes, a seamless CI/CD experience, and enterprise-grade security and governance. On Kubernetes clusters, you'll be able to deploy and execute web apps, function apps, logic apps, API management, and Event Grid.

Assisted Practice: Guidelines

Steps to create Kubernetes service are:

1. Login to your Azure portal
2. Create a resource
3. Enable Kubernetes Service
4. Configure the Basic page and Authentication Page
5. Deploy Kubernetes service

Azure Container Registry

Azure Container Registry

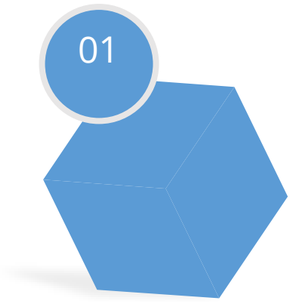
Azure Container Registry is a managed Docker registry service based on the open-source Docker Registry 2.0.

Azure Container Registry

These are the features of Azure Container Registry:

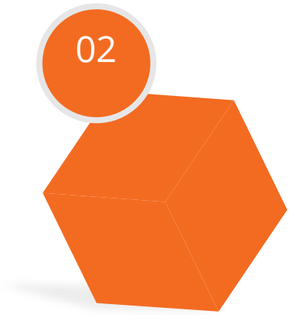
- Stores and manages private Docker container images
- Provides tight integration with multiple Azure services that support the following Docker containers:
  - Azure App Service
  - Azure Batch
  - Azure Service Fabric
  - Azure Kubernetes Service

Azure Container Registry Terminologies



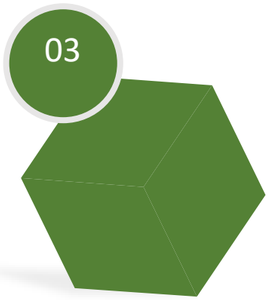
Registry

A service that stores container images



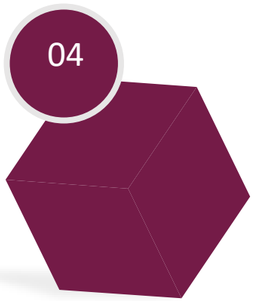
Repository

A group of related container images



Image

A point-in-time snapshot of a Docker-compatible container



Container

A software application running in an isolated environment

Container Registry SKUs

SKU	Description
Basic	• Ideal for developers learning about Container Registry • Same programmatic capabilities as Standard and Premium • However, there are size and usage constraints
Standard	• The same capabilities as Basic, but with increased storage limits and image throughput • May satisfy the needs of most production scenarios
Premium	• Higher limits on constraints, such as storage and concurrent operations, including enhanced storage capabilities to support high-volume scenarios • Adds features like geo-replication to manage a single registry across multiple regions

Container Registry

To create a container registry, use the Azure CLI, and then run the following commands

```
# Create a Container Registry instance
az acr create --resource-group <group> --name <acr-name> --sku Basic

# Login to Container Registry
az acr login --name <acrName>
```

Docker Image for Container Registry

To build a Docker image for the Container Registry, run the following commands

```
# Pull existing Docker image
docker pull microsoft/aci-helloworld

# Obtain the full login server name of the Container Registry instance
az acr list --resource-group <group> --query "[].{acrLoginServer:loginServer}" --output
table

# Tag image with full login server name prefix
docker tag microsoft/aci-helloworld <acrLoginServer>/aci-helloworld:v1

# Push image to Container Registry
docker push <acrLoginServer>/aci-helloworld:v1
```

Image in Container Registry

To deploy an image to Container Registry, run the following commands

```
# Enable admin user
az acr update --name <acrName> --admin-enabled true

# Query for the password
az acr credential show --name <acrName> --query "passwords[0].value"

# Deploy container image
az container create --resource-group <group> --name acr-quickstart --image
<acrLoginServer>/aci-helloworld:v1 --cpu 1 --memory 1 --registry-username <acrName> --
registry-password <acrPassword> --dns-name-label <fqdn> --ports 80

# View container FQDN
az container show --resource-group myResourceGroup --name acr-quickstart `
    --query instanceView.state
```

Image in Container Registry

To view a deployed image in Container Registry, run the following commands

```
# List container images
az acr repository list --name <acrName> --output table

# List the tags on the aci-helloworld repository
az acr repository show-tags --name <acrName> --repository aci-helloworld --
output table
```

Assisted Practice

Create a Container Registry

Duration: 10 Min.

Problem Statement:

You've been assigned the task of creating a Container Registry, which will allow you to build, store, and manage container images and artifacts for all types of container deployments in a private registry. The Azure container registries can be used with your current container development and deployment processes.

Assisted Practice: Guidelines

Steps to create a container registry are:

1. Login to your Azure portal
2. Click on Create a resource
3. Click on Containers, and select Container Registry
4. Add the details and create

Azure Container Instance

Azure Container Instance

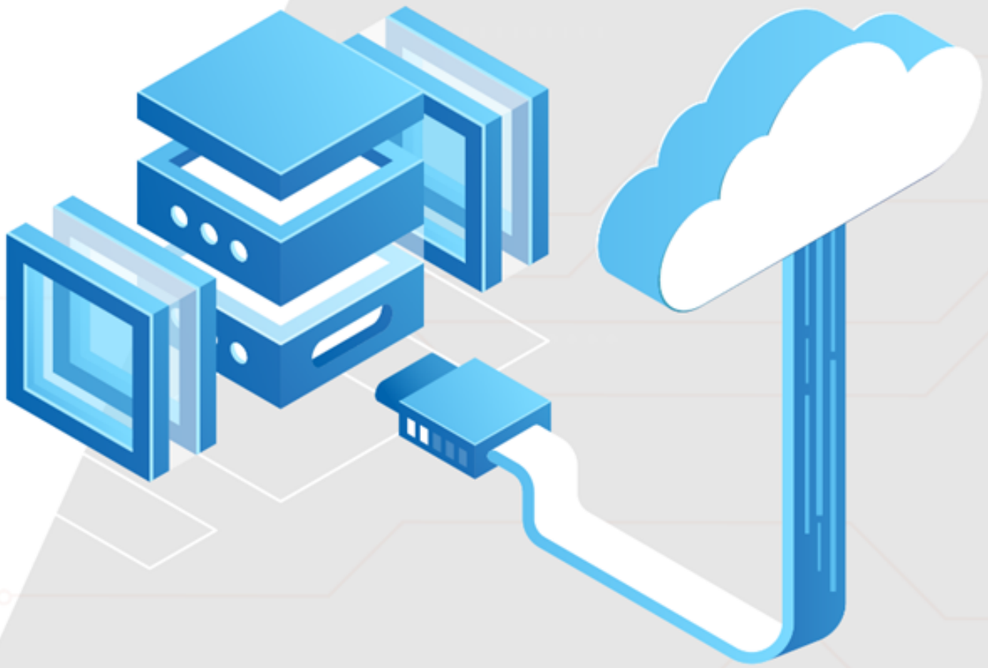
- Simplest way to run a container in Azure:
  - Doesn’t require IaaS provisioning
  - Doesn’t require the adoption of a higher-level service
- Ideal for one-off, isolated container such as:
  - Simple applications
  - Task automation
  - Build jobs
- Supports Linux and Windows containers
- Supports direct mounting of Azure Files shares
- Container can be provisioned with public IP address and DNS name

Features of Container Instance

Feature	Description
Fast startup times	Containers can start in seconds without the need to provision and manage VMs
Public IP connectivity and DNS name	Containers can be directly exposed to the internet with an IP address and a fully qualified domain name (FQDN)
Hypervisor-level security	Container applications are as isolated in a container as they would be in a VM
Custom sizes	Container nodes can be scaled dynamically to match actual resource demands for an application
Persistent storage	Containers support direct mounting of Azure Files shares
Linux and Windows containers	The same API is used to schedule both Linux and Windows containers
Co-scheduled groups	Container Instances supports scheduling of multi-container groups that share resources of host machine
Virtual network deployment	Container Instances can be deployed into an Azure virtual network

Key Takeaways

- Container is a loosely isolated environment that allows the user to build and run software packages.
- Azure Kubernetes Service manages the hosted Kubernetes environment and simplifies deployment of containerized applications in Azure.
- Azure Container Registry stores and manages private Docker container images.
- Azure Container Registry allows publishing and automation of Image Management.



Thank you