Microsoft

# Module 05: Implement IaaS solutions

Microsoft

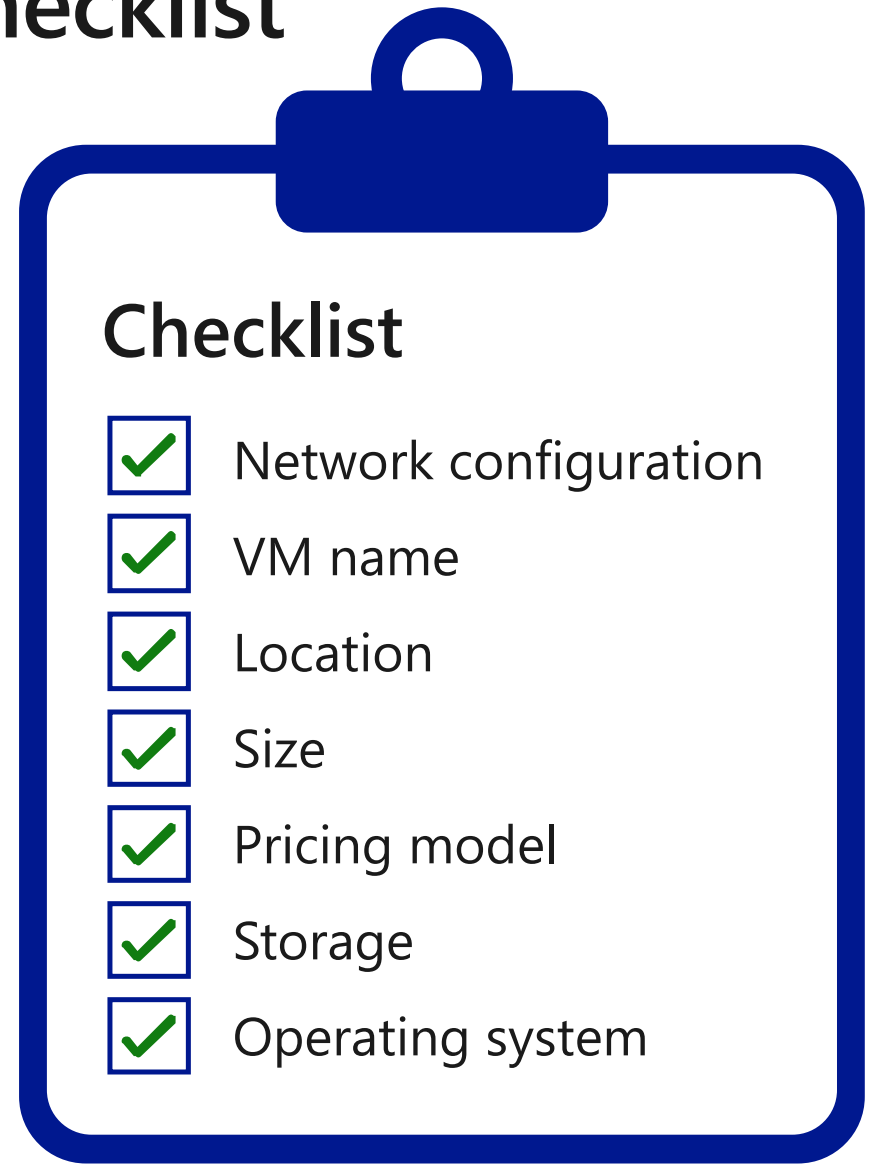
# Topics

- Provisioning VMs in Azure
- Create and deploy Azure Resource Manager templates
- Create container images for solutions
- Publish a container image to Azure Container Registry
- Create and run container images in Azure Container Instances

# Lesson 01: Provisioning VMs in Azure

# Azure virtual machine creation checklist

- Before you create a VM, you should consider the following:



# Naming a VM

- The VM name is used as the computer name, which is configured as part of the operating system
- Rules:
  - Up to 15 characters for a Windows VM
  - Up to 64 characters for a Linux VM

# Naming a VM (continued)

Current best practices for VM name choices:

| Element | Example | Notes |
| --- | --- | --- |
| **Environment** | dev, prod, QA | Identifies the environment for the resource |
| **Location** | uw (US West), ue (US East) | Identifies the region into which the resource is deployed |
| **Instance** | 01, 02 | For resources that have more than one named instance (such as web servers) |
| **Product or Service** | service | Identifies the product, application, or service that the resource supports |
| **Role** | sql, web, messaging | Identifies the role of the associated resource |

# VM pricing models

- Two primary costs for every VM:
  - **Storage –** The cost of storing data in every virtual hard disk. This cost is independent of whether the VM is running
  - **Compute –** The usage-based price for compute capacity when the VM is currently allocated

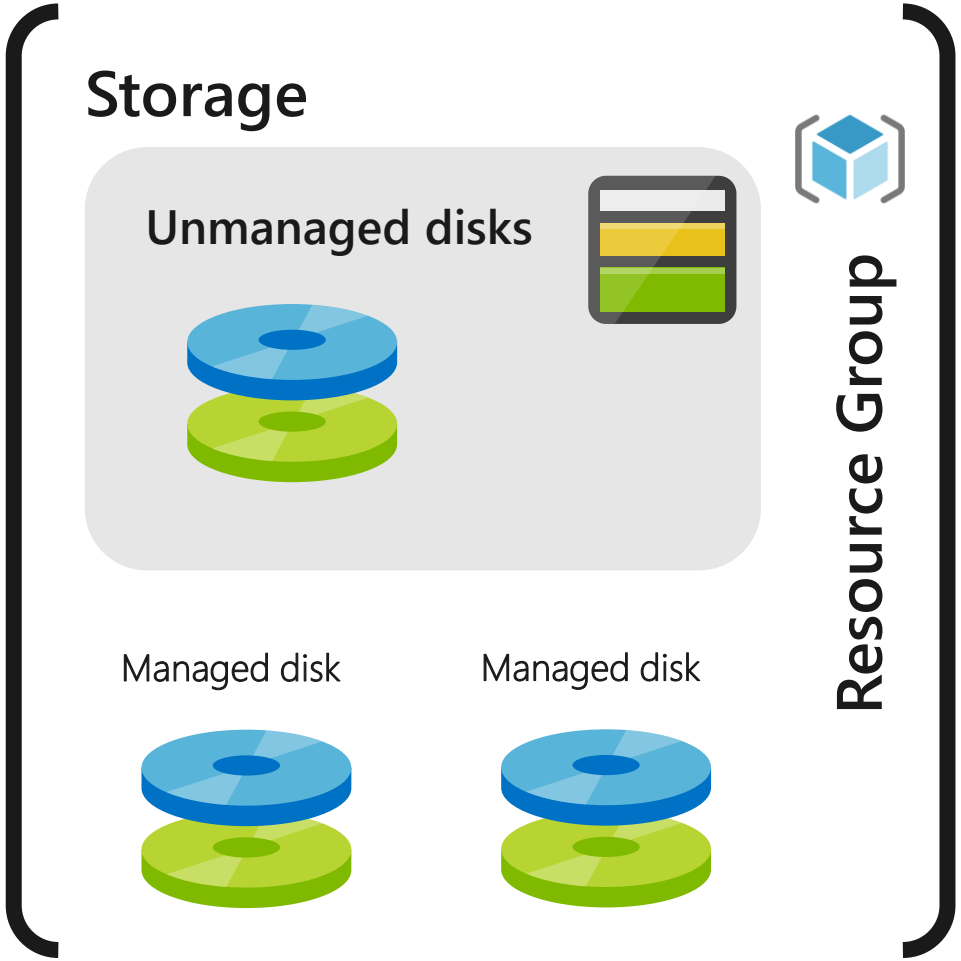
- There are two payment options for compute costs:
  - **Pay as you go –** Compute capacity is billed and paid as it is used without a long-term commitment
  - **Reserved instances –** Compute capacity can be pre-purchased at a reduced rate for anticipated usage

# VM storage options

- Virtual disks can be backed by either Standard or Premium Storage accounts
  - Azure Premium Storage leverages solid-state drives (SSDs) to enable high performance and low latency for VMs running I/O-intensive workloads
- You can choose either unmanaged disks or managed disks

# Managed and unmanaged disks

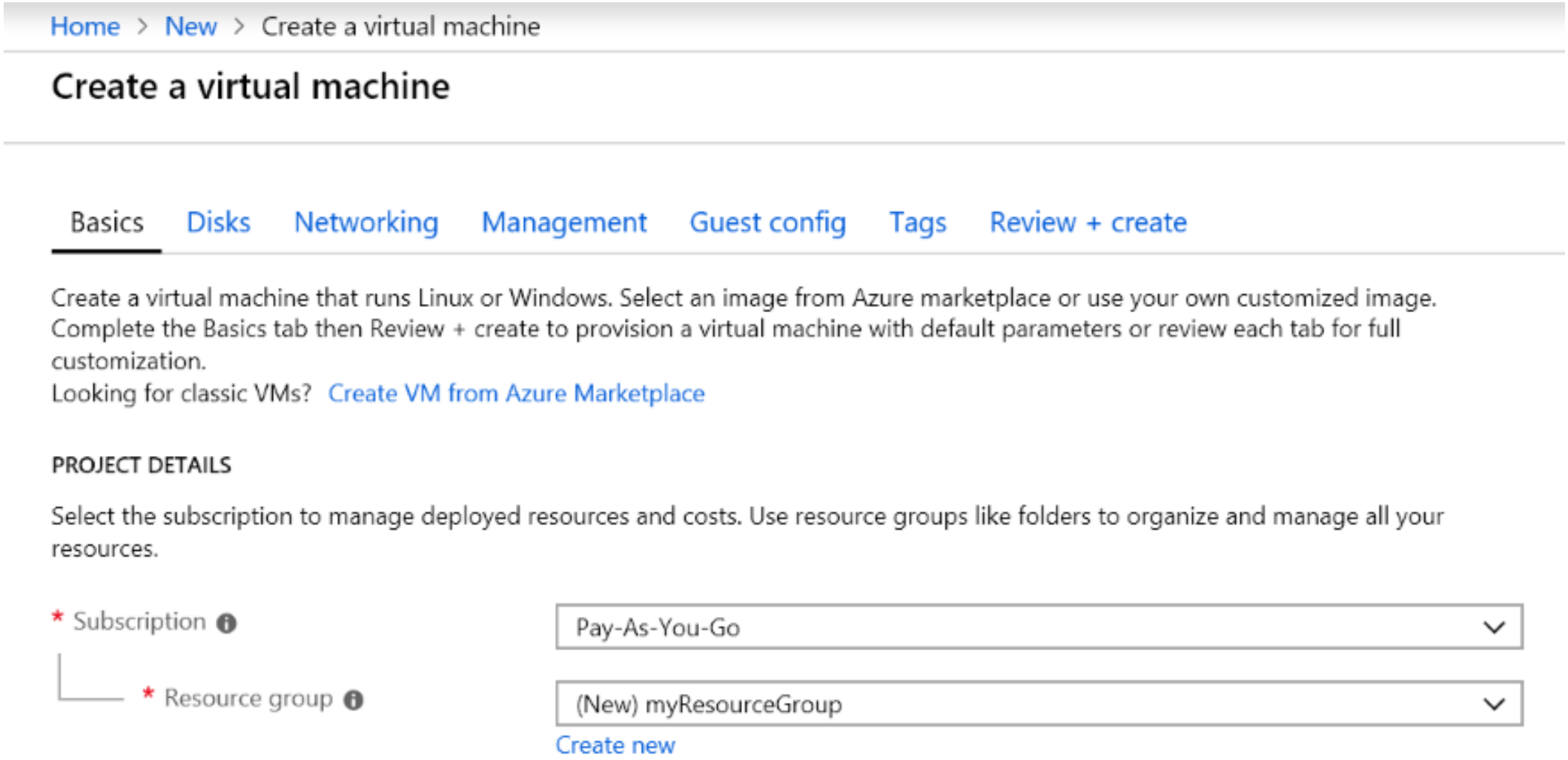
- Managed disks
  - The Azure platform manages the disk and the backing storage
  - You don't have to worry about storage account limits and thresholds
- Unmanaged disks
  - You manually create and manage virtual hard disks (VHDs) in your Storage account
  - You will need to consider account throughput and capacity limits when using this model



# Azure virtual machine creation and management

- Azure portal
  - Browser-based user interface that allows you to create and manage all your Azure resources
- Azure Resource Manager
  - Allows you to create templates, which can be used to create and deploy specific configurations of multiple Azure resources
- Azure PowerShell
  - Optional package that adds Azure-specific commands to PowerShell
- Azure CLI
  - Cross-platform command-line tool for managing Azure resources
- Programmatic (APIs)

# Create an Azure VM by using the Azure portal



# Demonstration: Creating an Azure VM by using the Azure portal

# Create an Azure VM by using PowerShell

```
Connect-AzAccount

New-AzResourceGroup -Name "myResourceGroup" -Location EastUS

New-AzVM `
    -ResourceGroupName "myResourceGroup" `
    -Name "myVM" `
    -Location "East US" `
    -VirtualNetworkName "myVnet" `
    -SubnetName "mySubnet" `
    -SecurityGroupName "myNetworkSecurityGroup" `
    -PublicIpAddressName "myPublicIpAddress" `
    -OpenPorts 80,3389
```

# Demonstration: Creating an Azure VM by using PowerShell

# Accessing an Azure VM by using PowerShell

```
Get-AzPublicIpAddress -ResourceGroupName "myResourceGroup" `
 | Select "IpAddress"

mstsc /v:publicIpAddress

Install-WindowsFeature -name Web-Server –IncludeManagementTools
```

# Sizing a VM

- Each VM size offers a variation of the following characteristics:
  - Processing power
  - Memory
  - Storage capacity
- Based on the workload, you're able to choose from a subset of available VM sizes

# VM configuration options

| | |
|---|---|
| Computational performance | 1 virtual CPU (vCPU) - 128 vCPUs |
| Memory | 1 gibibyte (GiB) - 4 tebibyte (TiB) |
| Disk storage | 4GiB - 64TiB<br>Up 160,000 IOPs |
| Networking | 30 GB Ethernet<br>100 GB InfiniBand |
| Availability | Single VM service-lvel agreement (SLA) 99.9% Multi AZ SLA 99.99% |

# VM categories

## Virtual machines

- Entry level
- Burstable
- General purpose
- Compute intensive
- Memory optimized
- GPU accelerated
- High performance computing
- Storage optimized

## Purpose built


# VM categories (cont.)

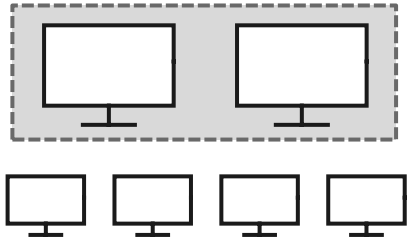
| Option | Description |
| --- | --- |
| **General purpose** | General-purpose VMs are designed to have a balanced CPU-to-memory ratio. Ideal for testing and development, small to medium databases, and low to medium traffic web servers. |
| **Compute optimized** | Compute optimized VMs are designed to have a high CPU-to-memory ratio. Suitable for medium traffic web servers, network appliances, batch processes, and application servers. |
| **Memory optimized** | Memory optimized VMs are designed to have a high memory-to-CPU ratio. Great for relational database servers, medium to large caches, and in-memory analytics. |
| **Storage optimized** | Storage-optimized VMs are designed to have high disk throughput and IO. Ideal for VMs running databases. |
| **GPU** | GPU VMs are specialized virtual machines targeted for heavy graphics rendering and video editing. These VMs are ideal options for model training and inferencing with deep learning. |
| **High performance compute** | High-performance compute is the fastest and most powerful CPU virtual machine with optional high-throughput network interfaces. |

# Manage the availability of your Azure VMs

- **Availability** is the percentage of time a service is available for use
- In the event of a physical failure within the Azure datacenter:
  - Azure will move the VM to a healthy host server automatically
  - “Self-healing” migration could take several minutes
  - If your VM is isolated to a single instance, the application(s) hosted on that VM will not be available
- VMs could also be affected by periodic updates initiated by Azure itself

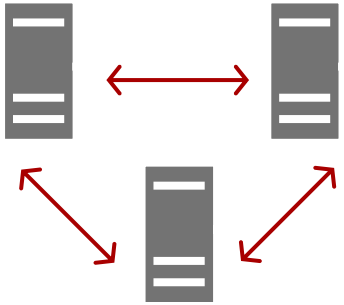
# High availability and disaster recovery

Higher Latency



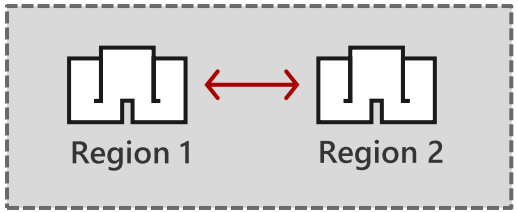
## Availability sets

Protection against failures within datacenters



## Availability zones

Protection from entire datacenter failures




## Region pairs

Protection from disaster with Data Residency compliance

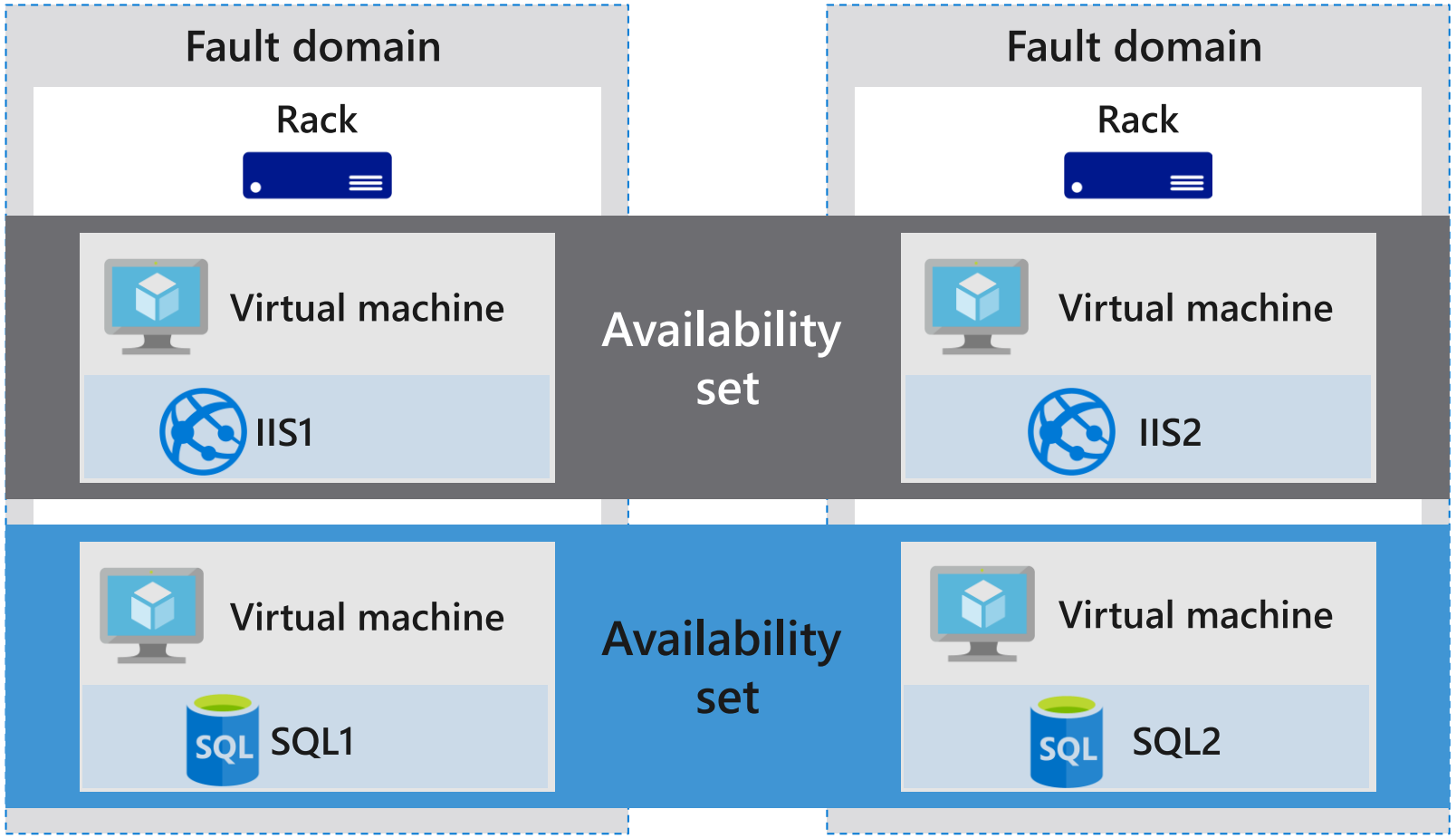
Increased resiliency

# Availability sets

- **Availability set** – logical feature used to ensure that a group of related VMs are deployed so that:
  - They are not all subject to a single physical point of failure
  - They are not all upgraded at the same time
- **Update domain –** logical group of hardware that can undergo a maintenance update at the same time

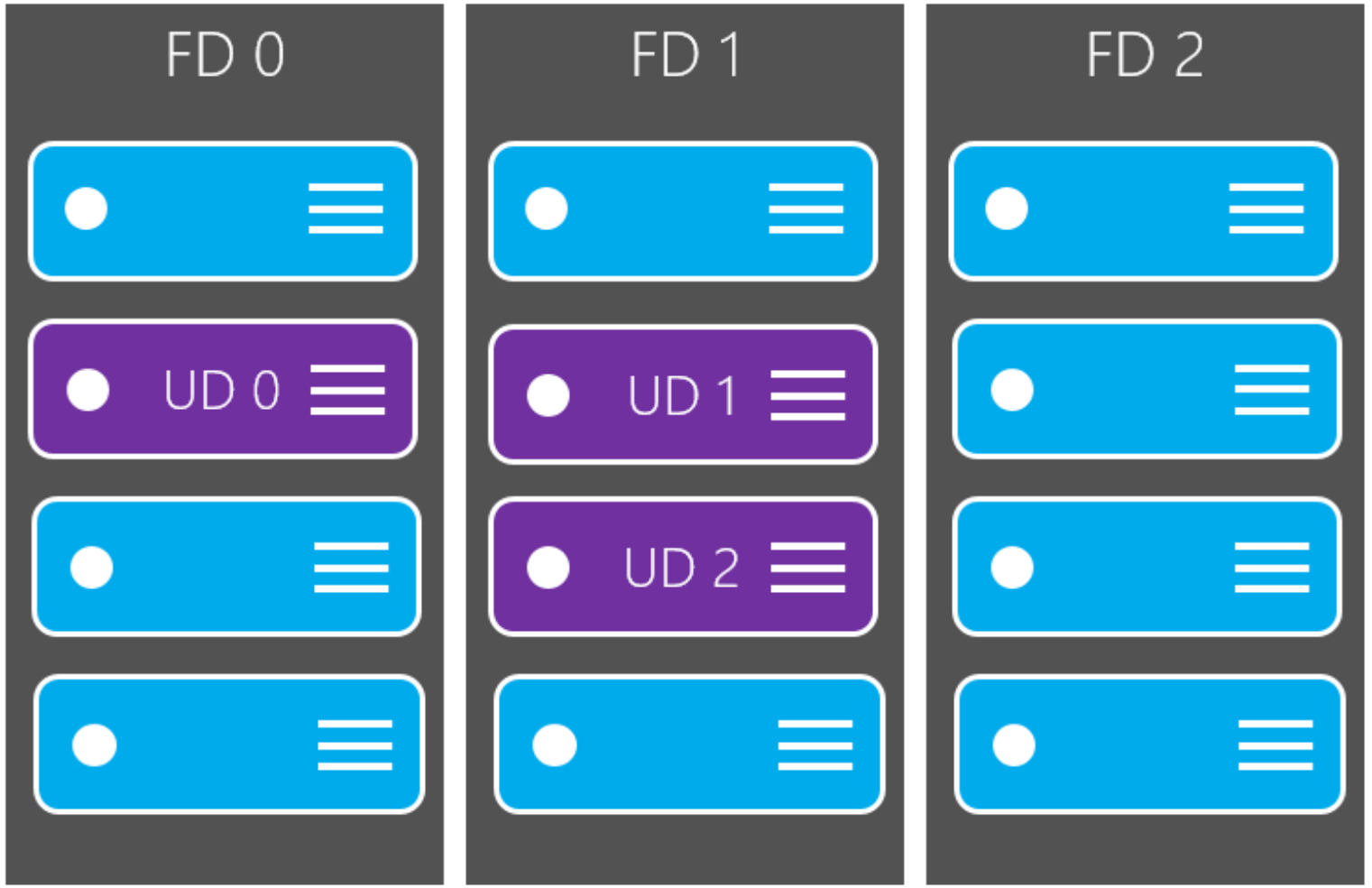
# Fault domains

A logical group of hardware in Azure that shares a common power source and network switch



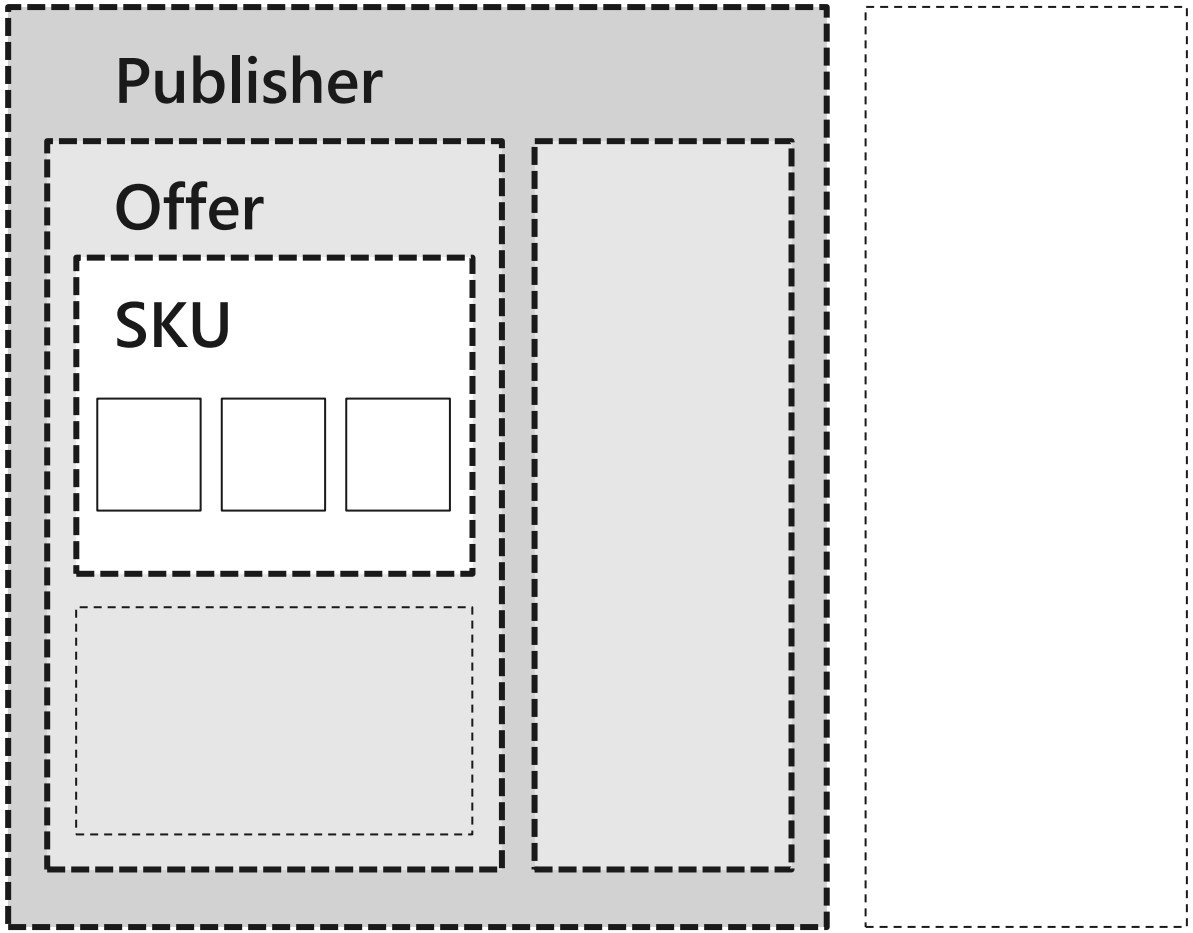
# Update domains

- Update domains enable targeting specific sets of hardware for maintenance or rebooting.



# Image in Azure Marketplace

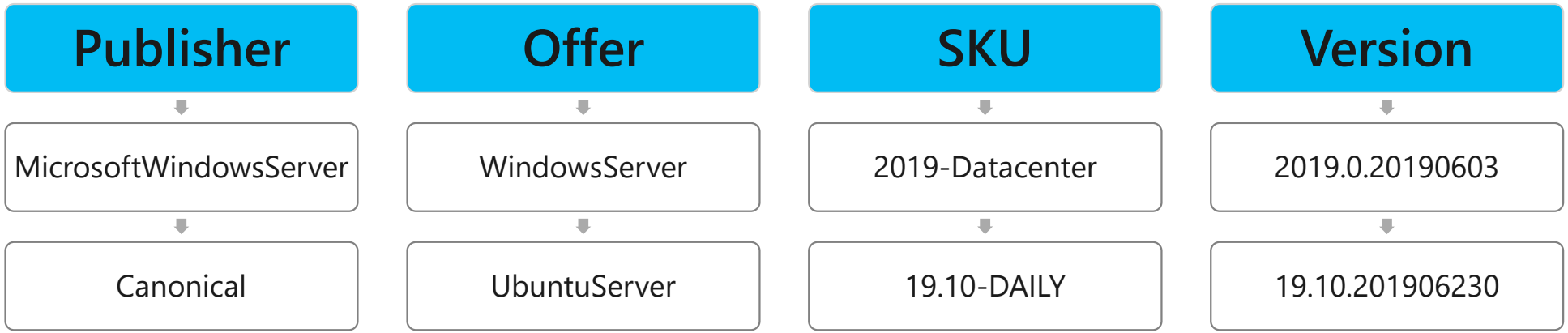
- Images in Azure Marketplace are grouped into the following categories:
  - Publisher
    - Organization that creates an image
  - Offer
    - Group of related images
  - SKU
    - Instance of an offer, typically a release
  - Version
    - A specific release version number



# Image Uniform Resource Name (URN)

Short-hand string to quickly access a known VM image Format

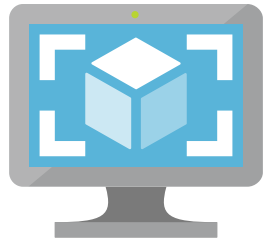
`PUBLISHER:OFFER:SKU:VERSION`



# Image Sources - Windows Server example

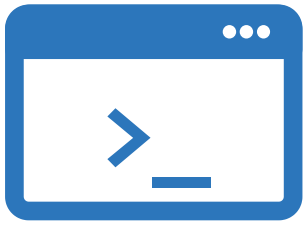
- Microsoft Windows Server PUBLISHER
  - Windows Server Semi-Annual OFFER
  - Windows Server OFFER
    - 2012 Datacenter SKU
    - 2012-R2-Datacenter SKU
    - 2016-Datacenter SKU
    - 2016-Datacenter-Core SKU
    - 2016-Datacenter-with-Containers SKU
    - 2019-Datacenter SKU
    - 2019-Datacenter-Core SKU
    - 2019-Datacenter-with-Containers SKU

2019.0.20181107 IMAGE
2019.0.20181122 IMAGE
2019.0.20181218 IMAGE
2019.0.20190115 IMAGE
2019.0.20190214 IMAGE
2019.0.20190314 IMAGE
2019.0.20190410 IMAGE
2019.0.20190603 IMAGE
2019.0.20190620 IMAGE



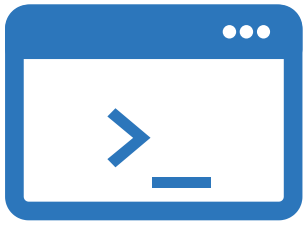
# Finding image sources by using the Azure CLI

```
# Get a list of all publishers available in the East US region
az vm image list-publishers --location eastus

# Get a list of all offers for the MicrosoftWindowsServer publisher
az vm image list-offers --location eastus --publisher MicrosoftWindowsServer

# Get a list of SKUs for the WindowsServer offer
az vm image list-skus --location eastus --publisher MicrosoftWindowsServer --
offer WindowsServer

# Get a list of all images available for the 2019-Datacenter SKU
az vm image list --all --location eastus --publisher MicrosoftWindowsServer --
offer WindowsServer --sku 2019-Datacenter
```



# Finding image sources by using the Azure CLI (cont.)

```
# Get the 2019.0.20190603 version of the VM image
az vm image show --location eastus --publisher MicrosoftWindowsServer --offer
WindowsServer --sku 2019-Datacenter --version 2019.0.20190603

# Alternatively, use an URN to get the specified version of the VM image
az vm image show --location eastus --urn
MicrosoftWindowsServer:WindowsServer:2019-Datacenter:2019.0.20190603
```



# Finding image sources by using Azure PowerShell

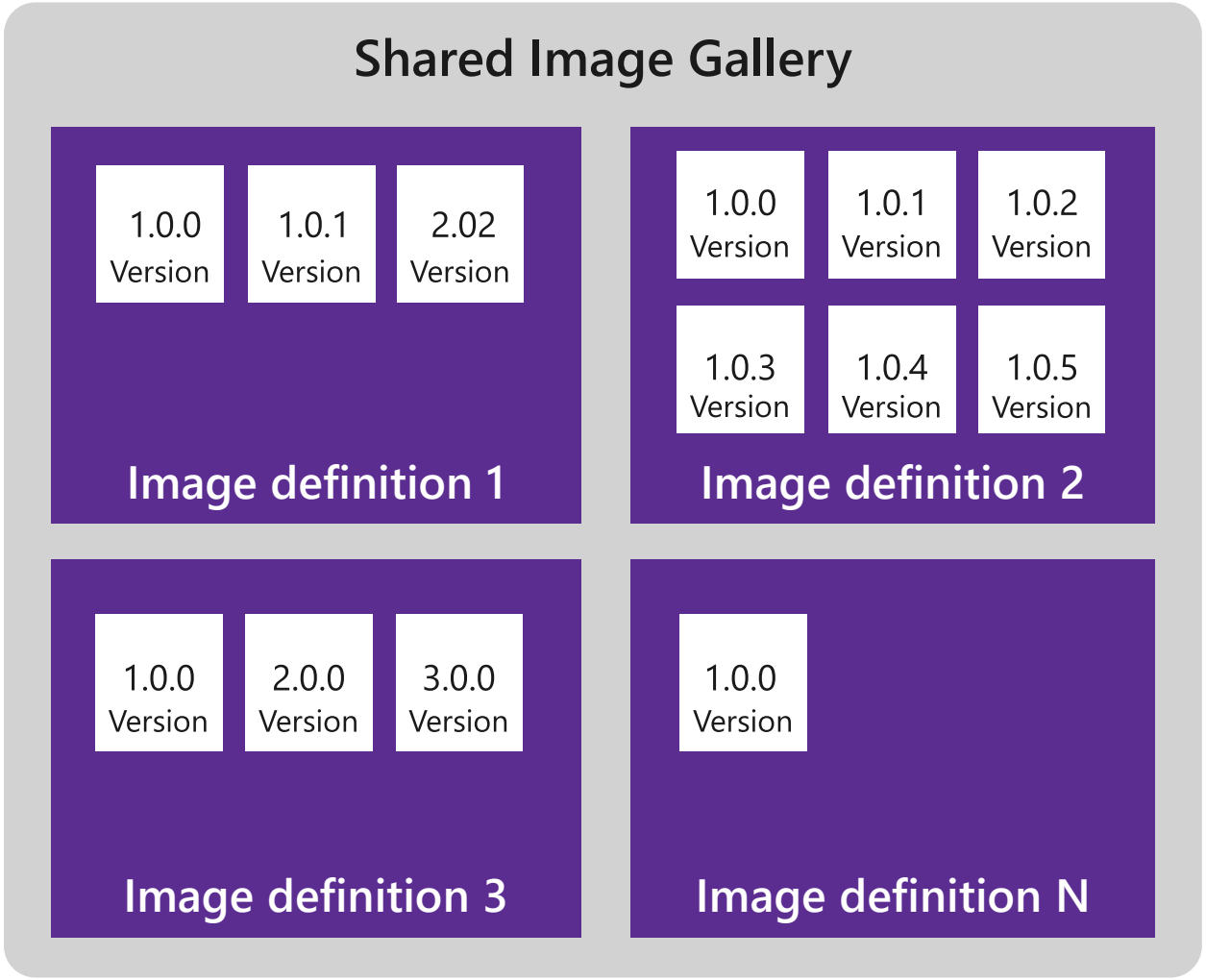
```
# Get a list of all publishers available in the East US region
Get-AzVMImagePublisher -Location eastus

# Get a list of all offers for the Canonical publisher
Get-AzVMImageOffer -Location eastus -PublisherName Canonical

# Get a list of SKUs for the UbuntuServer offer
Get-AzVMImageSku -Location eastus -PublisherName Canonical -Offer UbuntuServer

# Get a list of all images available for the 19.10-DAILY SKU
Get-AzVMImage -Location eastus -PublisherName Canonical -Offer UbuntuServer -
Sku 19.10-DAILY
```

# Finding Image Sources using Azure PowerShell (cont.)

```
# Get the 19.10.201906230 version of the VM image
Get-AzVMImage -Location eastus -PublisherName Canonical -Offer UbuntuServer -
Sku 19.10-DAILY -Version 19.10.201906230
```

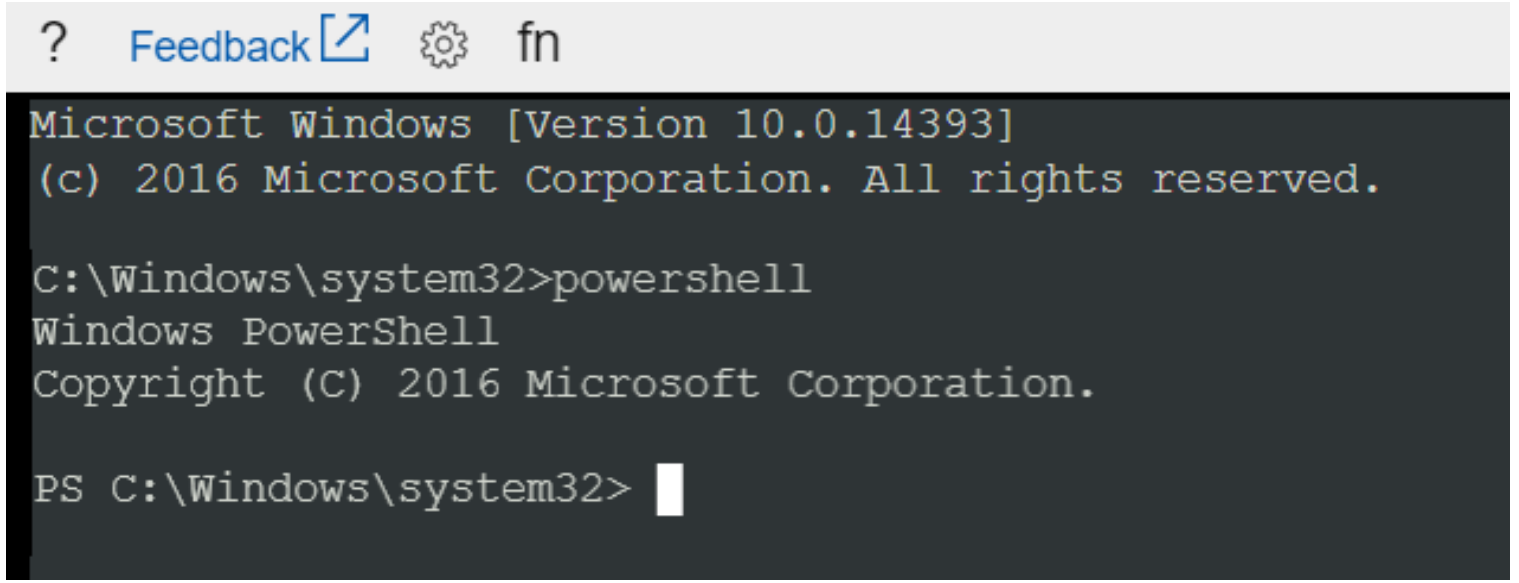
# Azure Shared Image Gallery

- Service to manage your images
- Provides
  - Global replication
  - Versioning
  - Grouping
  - High availability
  - Image sharing across subscriptions
  - Image replicas



# VM Serial Console

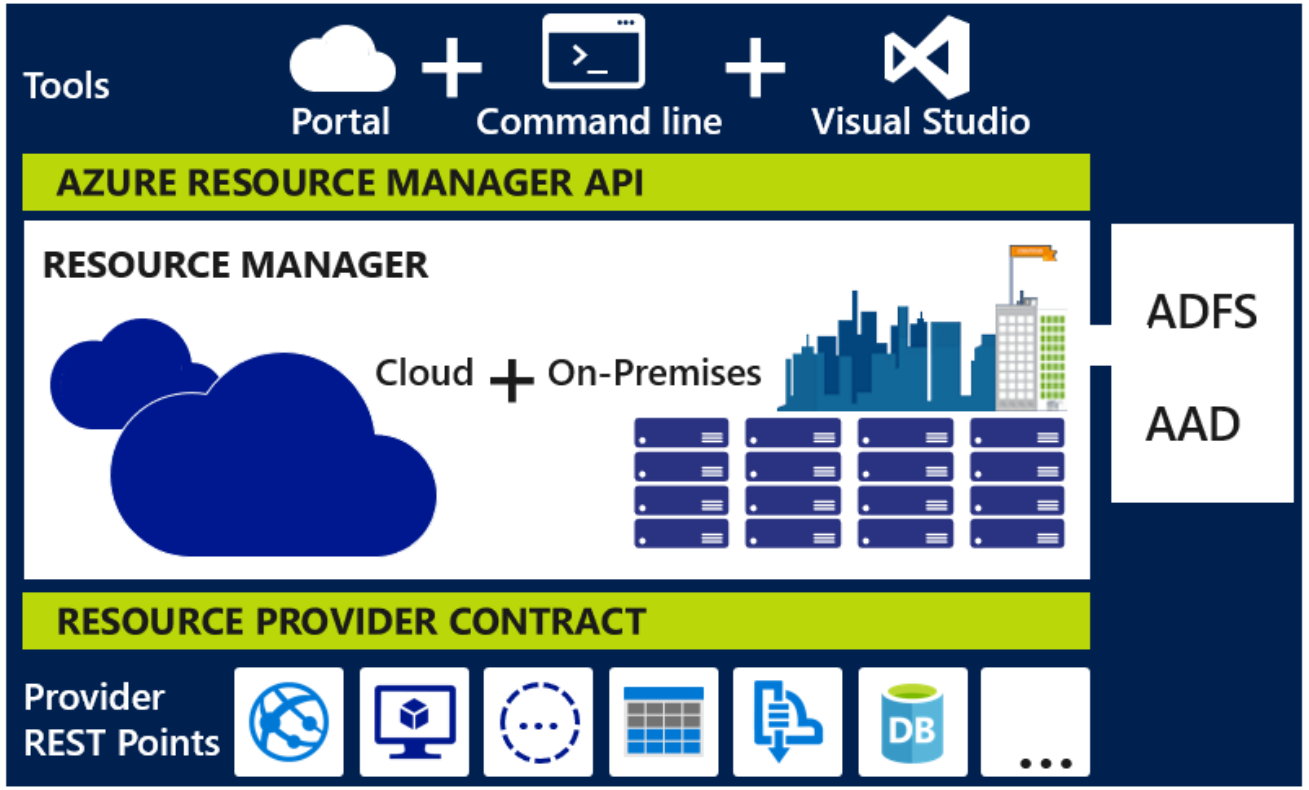
- Console access to a VM independent of network or OS state
- Available in Linux or Windows
  - Bash, CMD, PowerShell, NMI, SysRq, vi, GRUB, etc…



# Lesson 02: Create and deploy Azure Resource Manager templates

# Azure Resource Manager overview

- Resource Manager provides a consistent management layer to perform tasks
  - Azure PowerShell
  - Azure CLI
  - Azure portal
  - REST API
  - Client SDKs



# Terminology

- Resource
  - Single manageable item available through Azure
- Resource group
  - Container holding related resources
- Resource provider
  - Service that supplies resource instances in accordance with a predefined contract
- Resource Manager template
  - JSON file that defines one or more resources, specifying their resource providers, to be deployed to a resource group
- Declarative syntax
  - The act of describing your resources by using a template instead of manually creating the resources

# Azure Resource Manager templates

Azure Resource Manager Templates

- Precisely define all the Resource Manager resources in a deployment
- Are written in JavaScript Object Notation (JSON) format
- Are Declarative in nature
- Allow you to consistently deploy a solution with confidence that resources are deployed in a consistent state
- Allow you to reuse an existing template across different environments such as dev, test and production
- Allow you to re-use existing templates and customize them to your specific needs

# Why Use ARM templates?

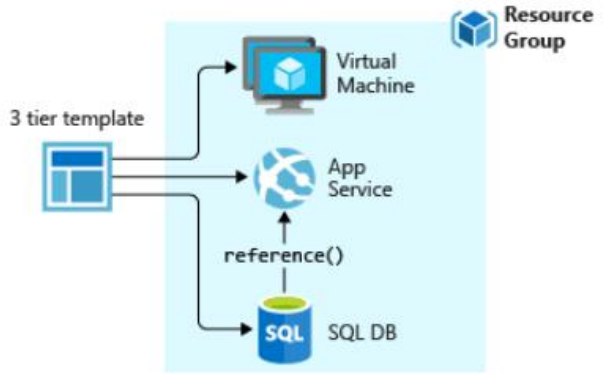
- Make deployments faster and more repeatable
- Improve consistency by providing a common language
- Enable you to deploy multiple resources in the correct order by mapping out resource dependencies
- Reduce manual, error-prone tasks
- Templates can be linked together to provide a modular solution

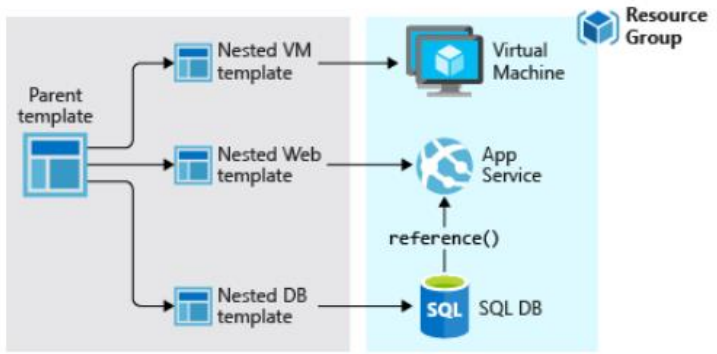
# ARM Template Elements

ARM Templates written in JSON files can contain the following sections:

- Parameters
- Variables
- Functions
- Resources
- Outputs

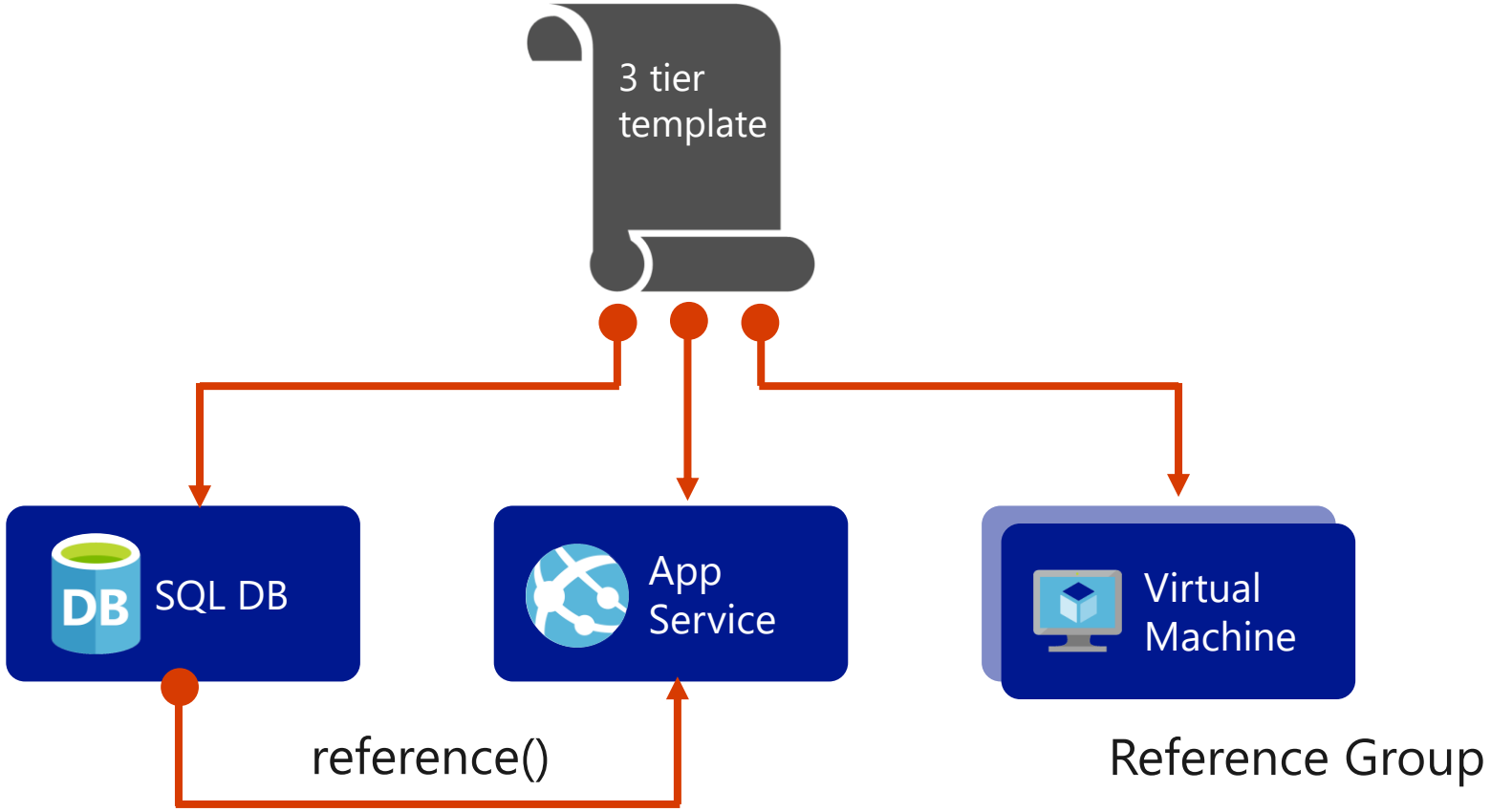
- A Deployment can be done using:
  - A  multi tier Template
  - A parent Template with nested Templates



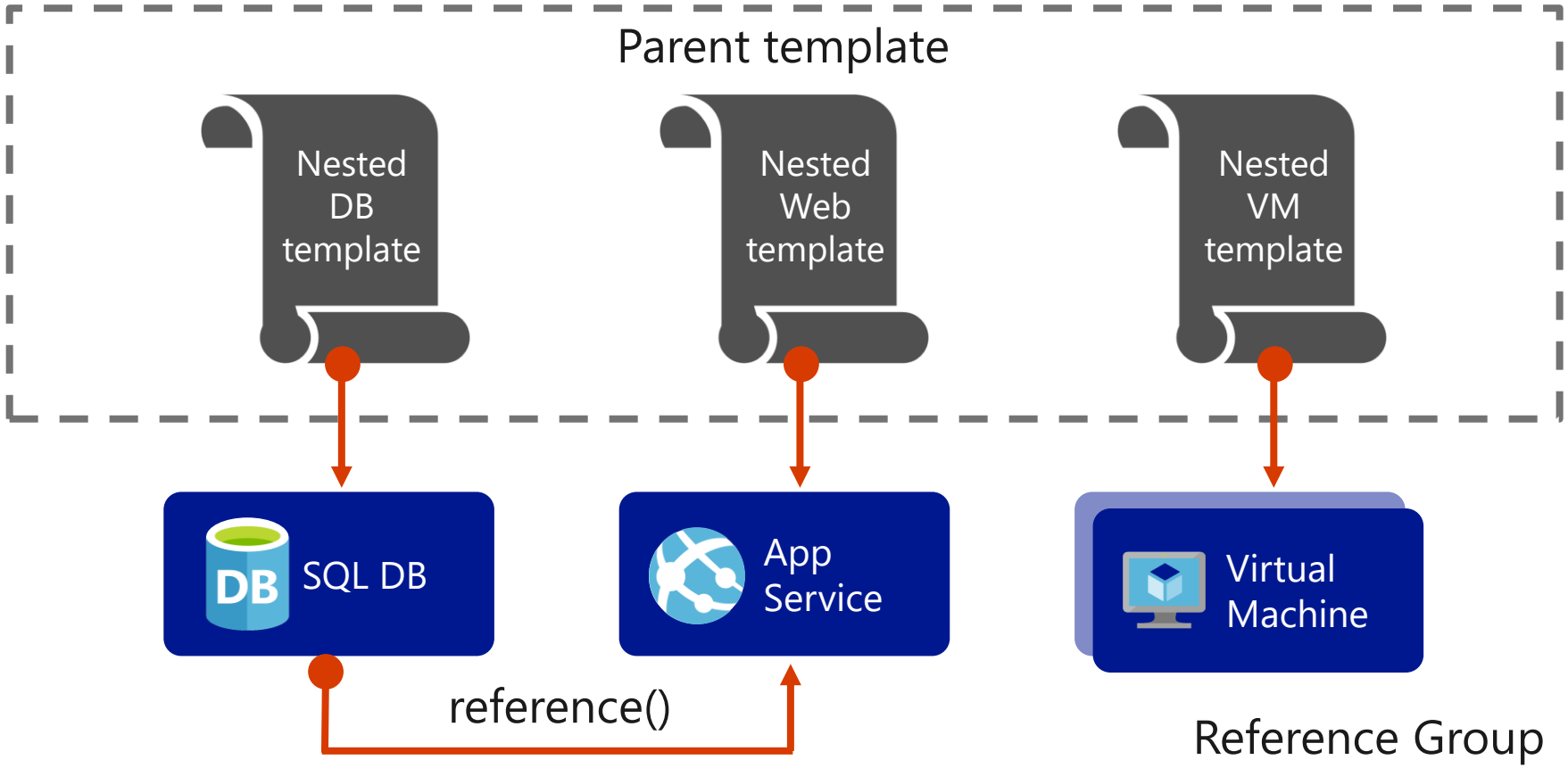
# Three-tier Azure Resource Manager template

Three-tier application through a single Resource Manager template



# Nested Resource Manager template

Nested templates deploying a similar three-tier application

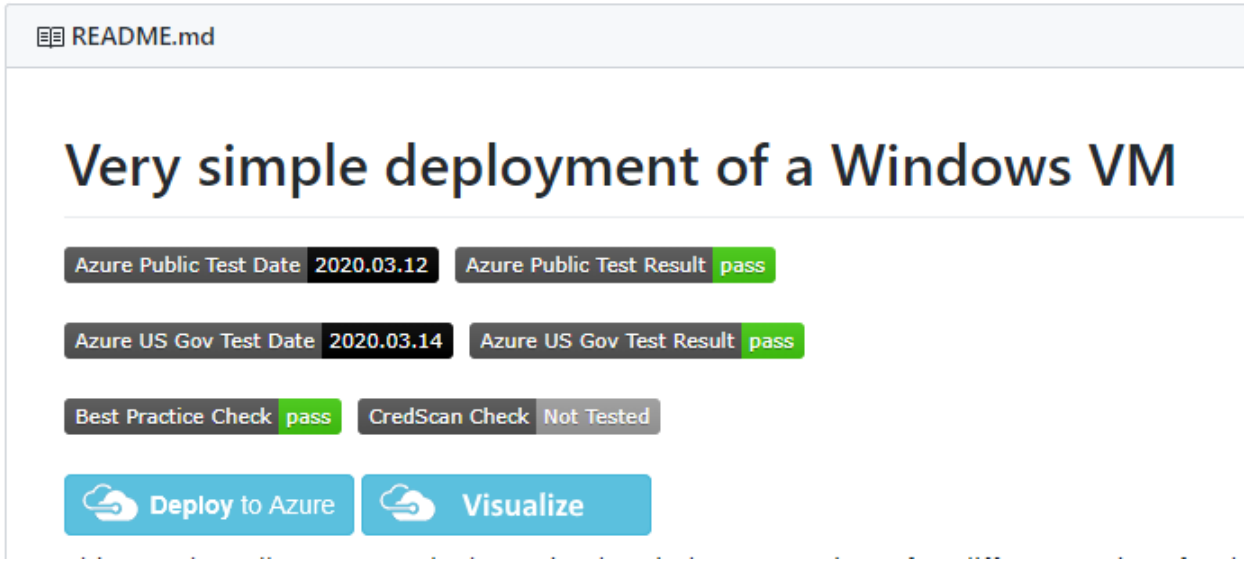


# Quickstart templates

Resource Manager templates provided by the Azure community

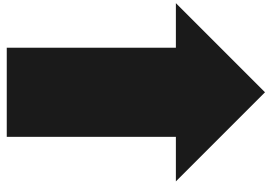
Behind the scenes it consists of:

- ARM Template
- Deploy Button
  - http://azuredeploy.net/
  - armdeploy.json
- Visualization
  - http://armviz.io/



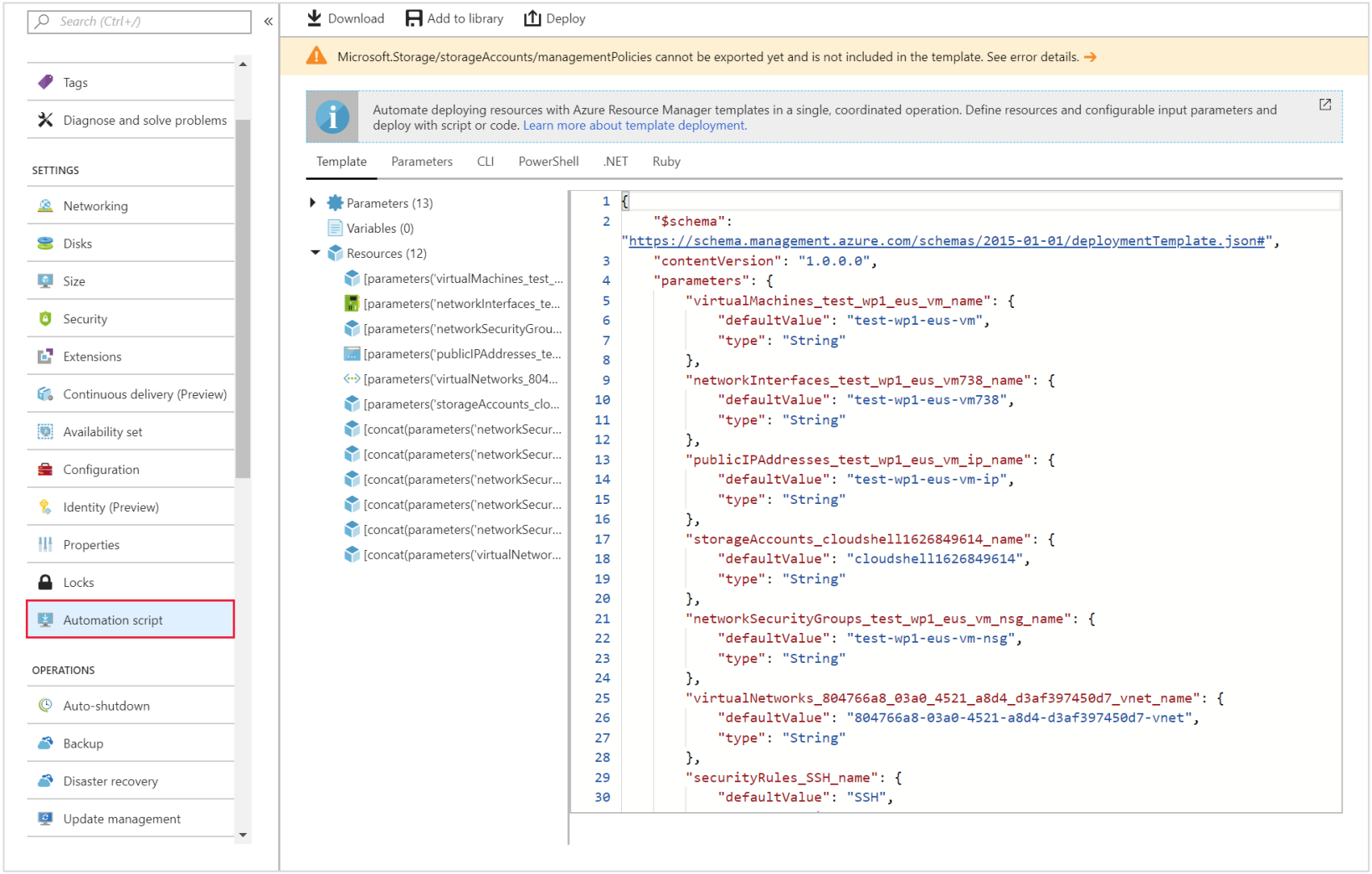
# Resource Manager template deployment

```
"resources": [
    {
        "apiVersion": "2016-01-01",
        "type":
"Microsoft.Storage/storageAccounts",
        "name": "mystorageaccount",
        "location": "westus",
        "sku": {
            "name": "Standard_LRS"
        },
        "kind": "Storage",
        "properties": {}
    }
]
```



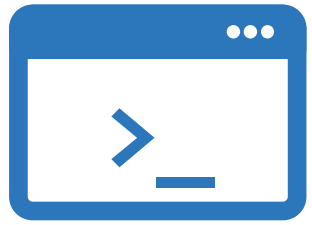
```
PUT
https://management.azure.com/subscri
ptions/{subscriptionId}/resourceGrou
ps/{resourceGroupName}/providers/Mic
rosoft.Storage/storageAccounts/mysto
rageaccount?api-version=2016-01-01
REQUEST BODY
{
    "location": "westus",
    "properties": {},
    "sku": {
        "name": "Standard_LRS"
    },
    "kind": "Storage"
}
```

# Create Resource Manager templates by using the Azure portal



# Deploying Azure Resource Manager templates by using Azure CLI

```
az group create --name $resourceGroupName --location $location

az group deployment create --name $deploymentName --resource-group
$resourceGroupName --template-file "azuredeploy.json"

az storage account show --resource-group $resourceGroupName --name
$storageAccountName
```

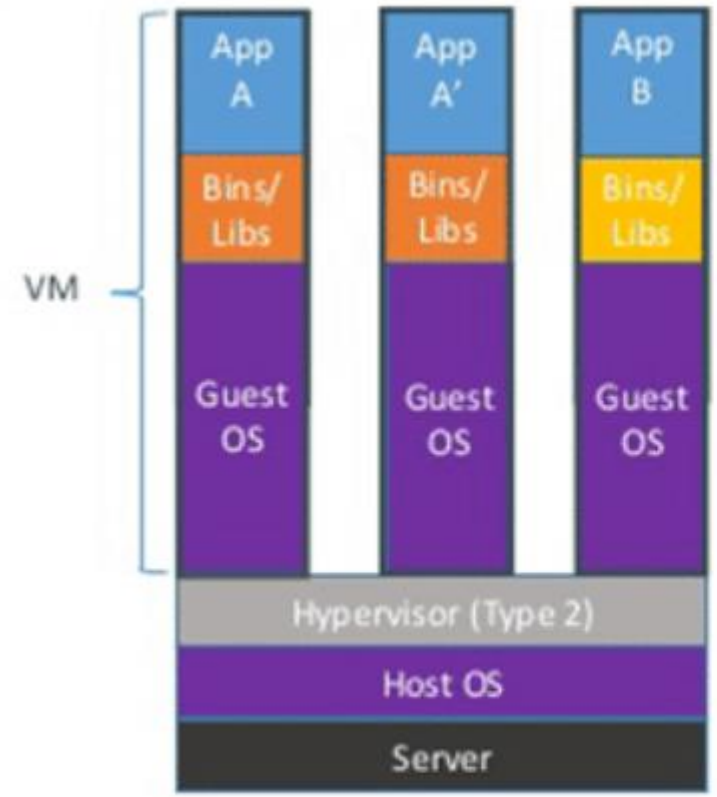


# Demonstration: Creating Azure Resource Manager templates by using the Azure portal

# Demonstration: Creating Azure Resource Manager templates by using Visual Studio Code

# Lesson 03: Create container images for solutions

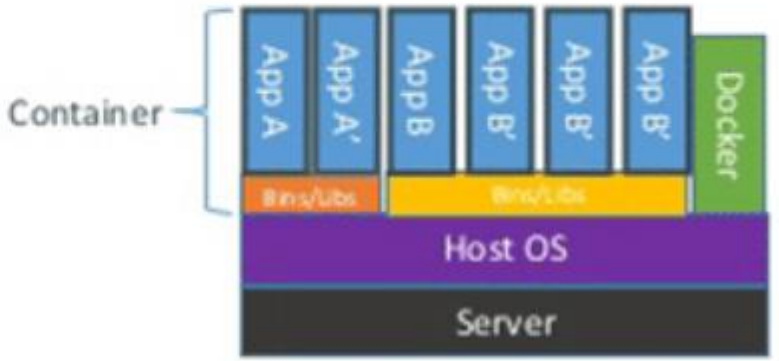
# Virtualization vs Containers




Virtual Machine

Containers are isolated, but share OS and, where appropriate, bins/libraries

...result is significantly faster deployment, much less overhead, easier migration, faster restart




Docker

docker

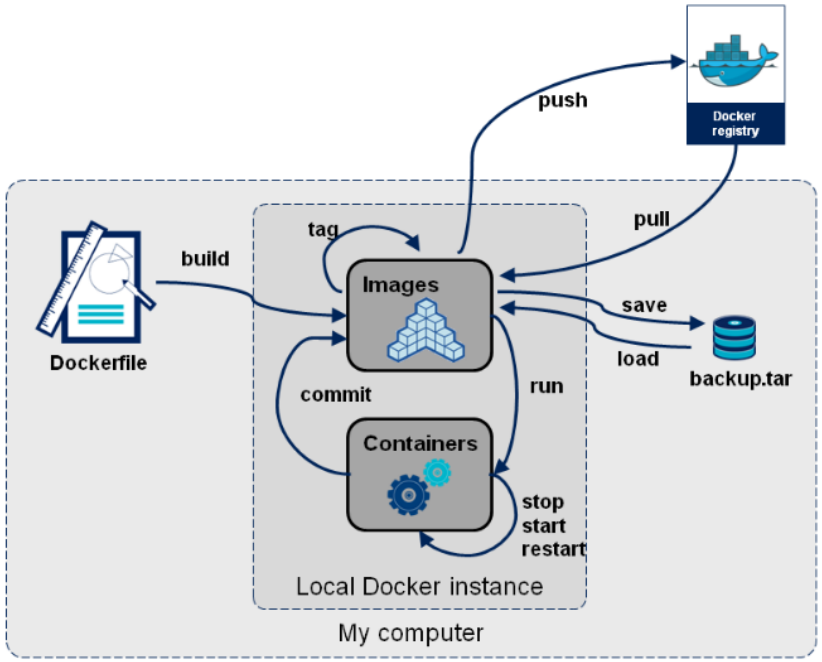
# Docker

An open platform for developing, shipping, and running applications

Ability to package & run an application in a loosely isolated environment called a container.

Main advantage for Devs:

- Seperates enviroment from hosting OS
- Stable runtime (in the future)
- Optimal for hosting (Micro-) Services

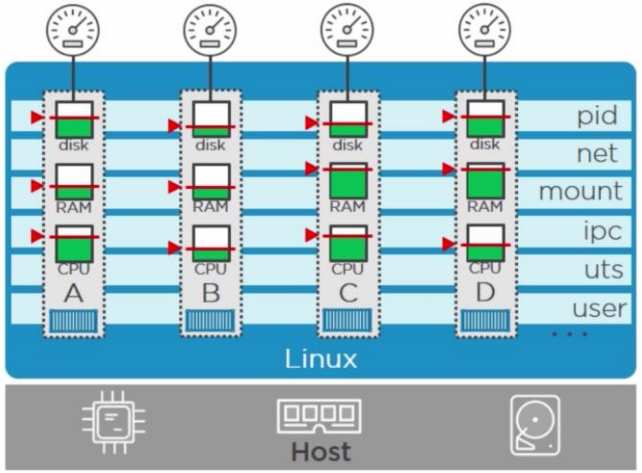


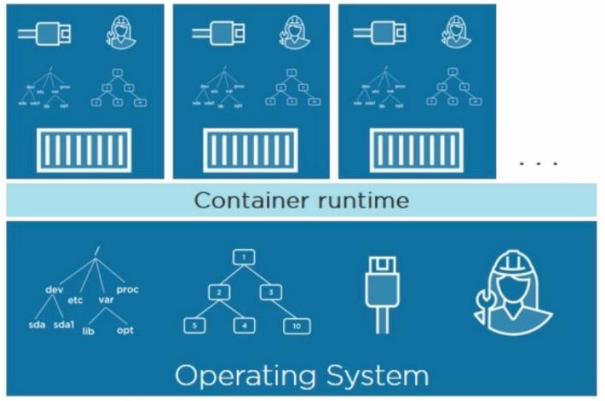
# Docker terminology

- **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
- **Build**
  - The process of creating a container image using a set of instructions
- **Pull**
  - The process of downloading a container image from a container registry
- **Push**
  - The process of uploading a container image to a container registry
- **Dockerfile**
  - A text file that contains instructions required to build a Docker image.

# Docker Containers

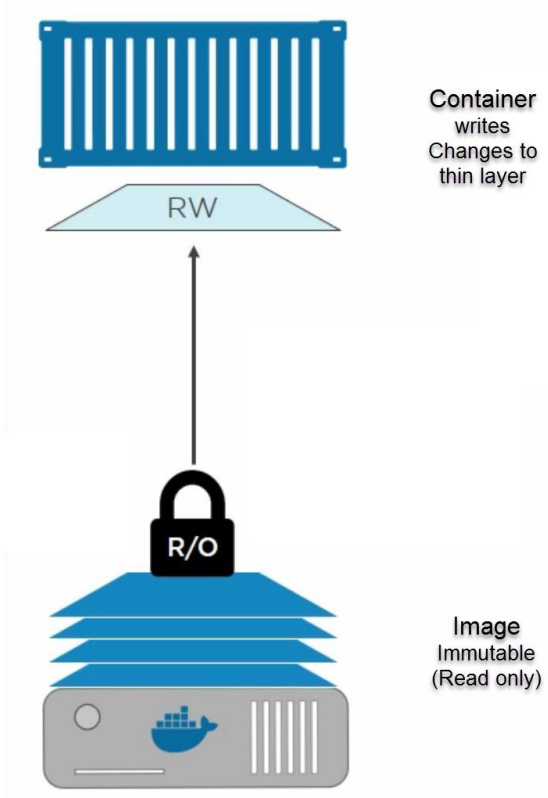
- Containers are running Instances of Docker Images
- Consume limited Ressources on the Host
- Can interact with Network, mounted Volumes
- Are executed by the Docker Daemon
- Contain all bits to run an application
- Available as
  - Linux Containers
  - Windows Containers



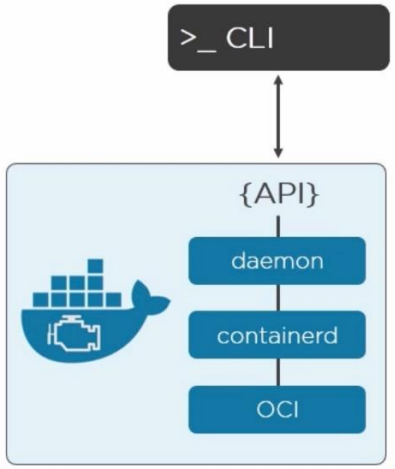
# Docker Images

- An image is an inert, immutable, file that's essentially a snapshot of a container
- Images are created with the build command
- They'll produce a container when started with run
- Images consist of Layers



# Docker CLI

- Allows you to manage & interact with daemon
  - Pull Images
    - docker pull IMAGE-NAME
  - List Containers:
    - docker container ls
  - Create Containers
    - docker build -t voucherapp .
    - "." means local folder
  - Run Containers
    - docker run --name voucherapp




docker build

docker checkpoint *

docker commit

docker config *

docker container *

docker cp

docker create

docker deploy

docker diff

docker events

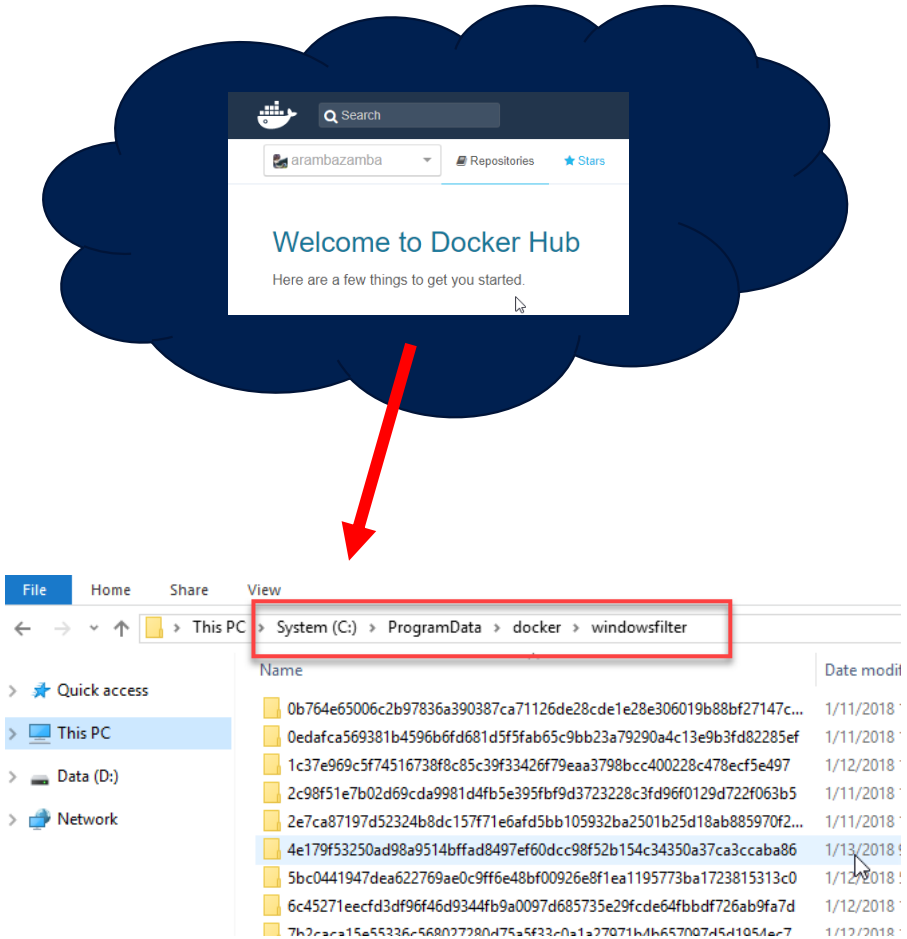
docker exec

docker export

docker history

# Loading Docker Images

- Images are loaded from Container Registry
  - ie. Dockerhub, Azure, Google Cloud ...
- Load a docker image from repository:
  - docker pull microsoft/dotnet:2.0.0-sdk
  - docker pull microsoft/mssql-server-linux
- Publish a docker image to repository:
  - docker tag skillsui arambazamba/skillsui
  - docker push arambazamba/skillsui



# Retrieving a new container image from Docker Hub

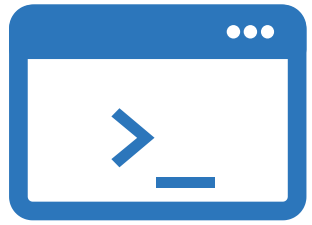
```
# Get Ubuntu container image
docker pull ubuntu
```

Container image name

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

Container image tag

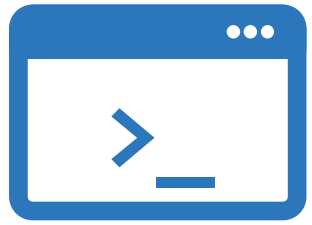
```
# Get the latest version of the nginx container image
docker pull nginx:latest
```



# Running the retrieved container image

```
# 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
```

Image name and tag



# Creating a container image specification with a Dockerfile

```
FROM node:8.9.3-alpine
RUN mkdir -p /usr/src/app
COPY ./app/ /usr/src/app/
WORKDIR /usr/src/app
RUN npm install
CMD node /usr/src/app/index.js
```

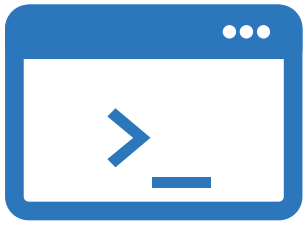
Start with this container image

Run this command

Copy these files from the host

Change the working directory

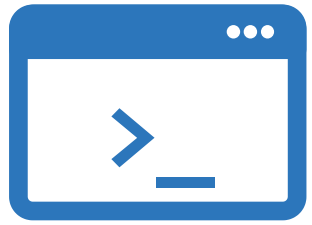
Start the container with this command



# Building the container image

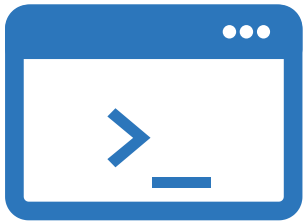
```
# Build your container
docker build ./application -t tutorial-app
```


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



# Running the custom container image as a container

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

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



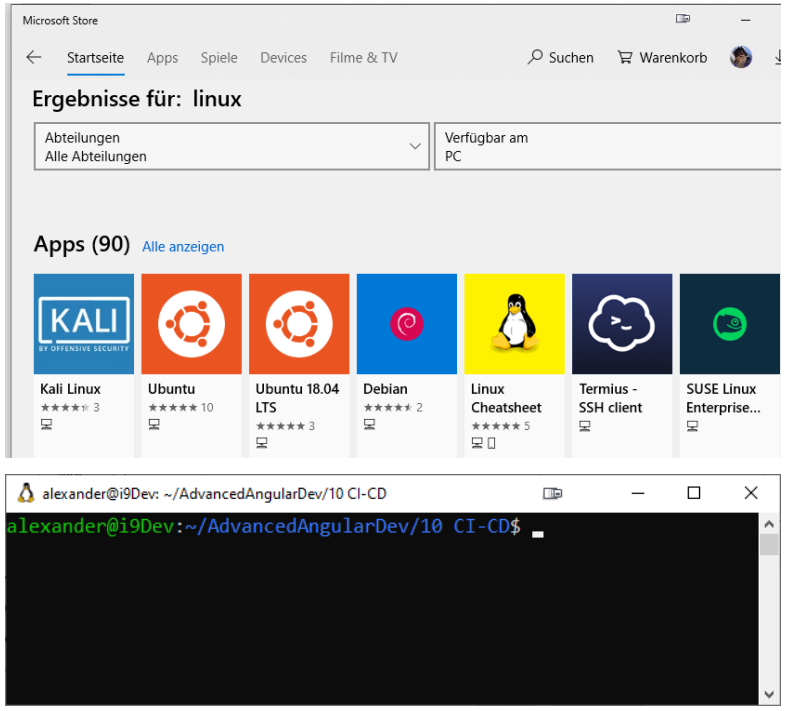
# Docker Compose

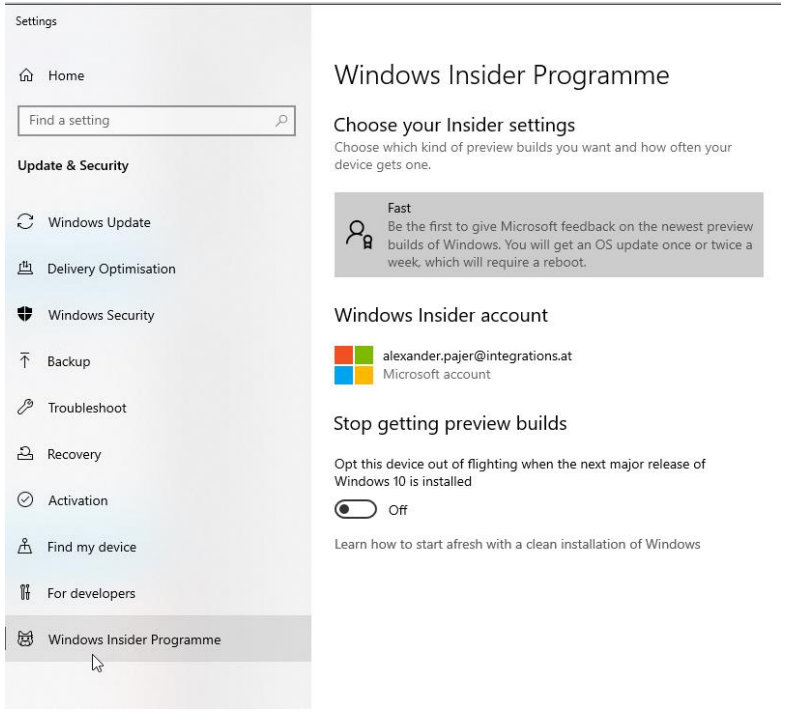
- Use Docker Compose  for defining and running multi-container Docker applications.
- Done in docker-compose.yml file
- Defines Services:
  - UI, API, SQL
  - Networks
  - Volumes
  - …

```yaml
services:
  skillsui:
    image: skillsui
    ports:
      - 8085:80
    networks:
      - skills-network
    depends_on:
      - skillsapi
  skillsapi:
    image: skillsapi
    ports:
      - 8080:5000
    networks:
      - skills-network
    depends_on:
      - sqllinux
  sqllinux:
    image: microsoft/mssql-server-linux
    ports:
      - 1433:1433
    environment:
      ACCEPT_EULA: "Y"
      SA_PASSWORD: "TiTp4SQL@dmin"
    networks:
      - skills-network
networks:
  skills-network:
    driver: bridge
```

# Windows Subsystem for Linux - WSL 2

- Part of Mai 2020 Windows 10 update - Replaces MobyLinux VM
- Can be used with any Linux Distribution that is in Microsoft Store
- What is it used for:
  - Linux based Development
  - Docker & Kubernetes
- Why:
  - No Path issues & faster!



# Demonstration: Retrieving and deploying an existing Docker image locally

# Demonstration: Creating a container image by using Docker

# Lesson 04: Publish a container image to Azure Container Registry

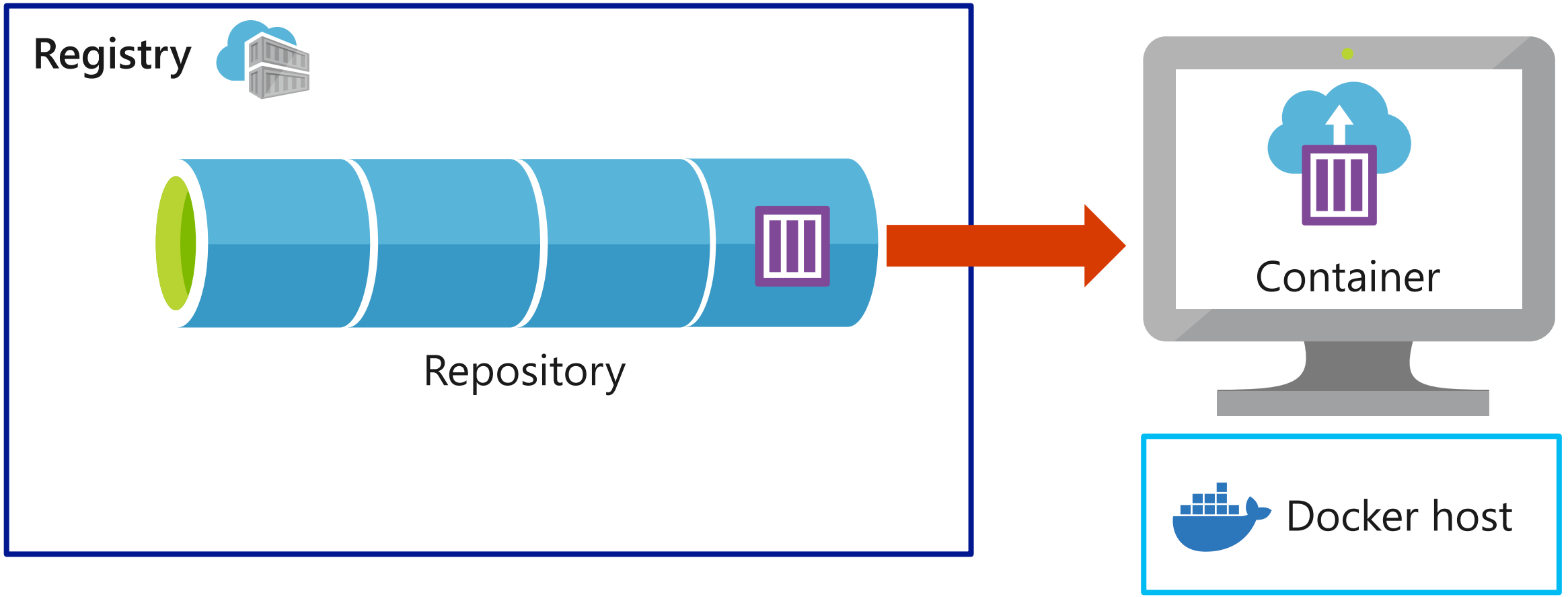
# Azure Container Registry (ACR)

- Managed Docker registry service
  - Based on the open-source Docker Registry 2.0
- Stores and manages private Docker container images
- Tight integration with multiple Azure services that support these Docker containers:
  - Azure App Service
  - Azure Batch
  - Azure Service Fabric
  - Azure Kubernetes Service

# Key terminology

- **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 and its dependencies running in an isolated environment

# Docker containers and registries

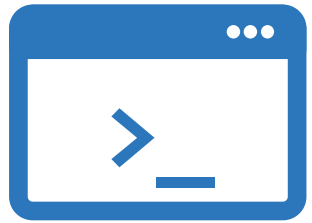


# Container Registry SKUs

| SKU | Description |
| --- | --- |
| **Basic** | • Ideal for developers learning about Container Registry<br>• Same programmatic capabilities as Standard and Premium, however, there are size and usage constraints |
| **Standard** | • Same capabilities as Basic, but with increased storage limits and image throughput.<br>• Should 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.<br>• Adds features like geo-replication for managing a single registry across multiple regions |

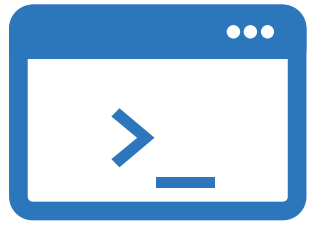
# Create a container registry by using Azure CLI

```
# 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>
```



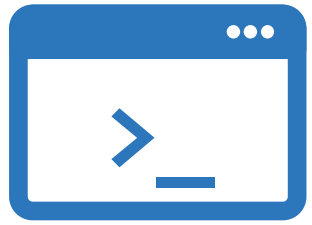
# Build a Docker image for Container Registry

```
# 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
```



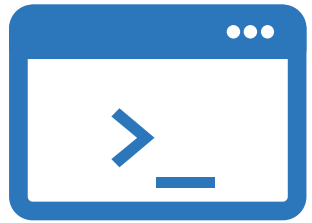
# View a deployed image in Container Registry by using Azure CLI

```
# 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
```



# Deploy an image to Container Registry by using Azure CLI

```
# 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
```

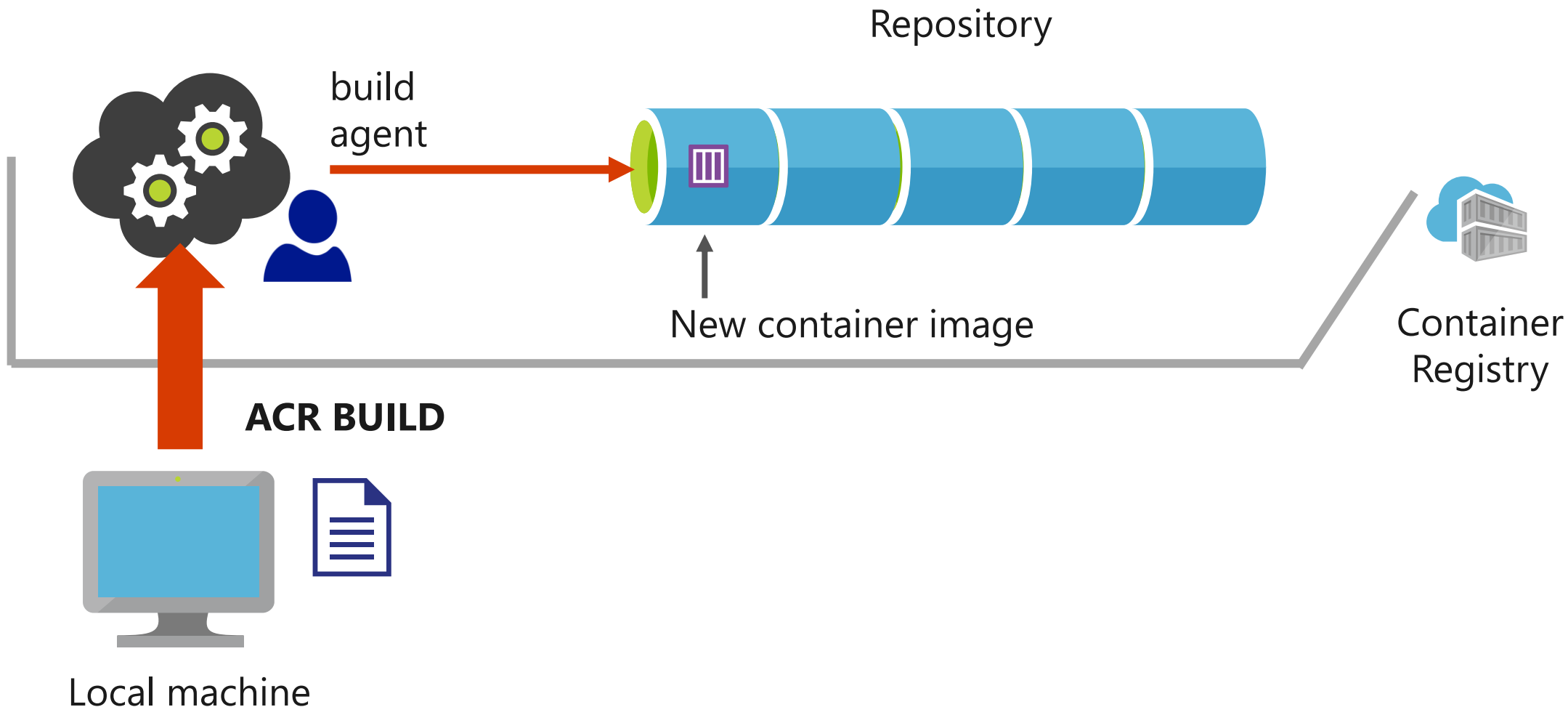


# Demonstration: Deploying an image to ACR by using Azure CLI

# Azure Container Registry Build (ACR Build)

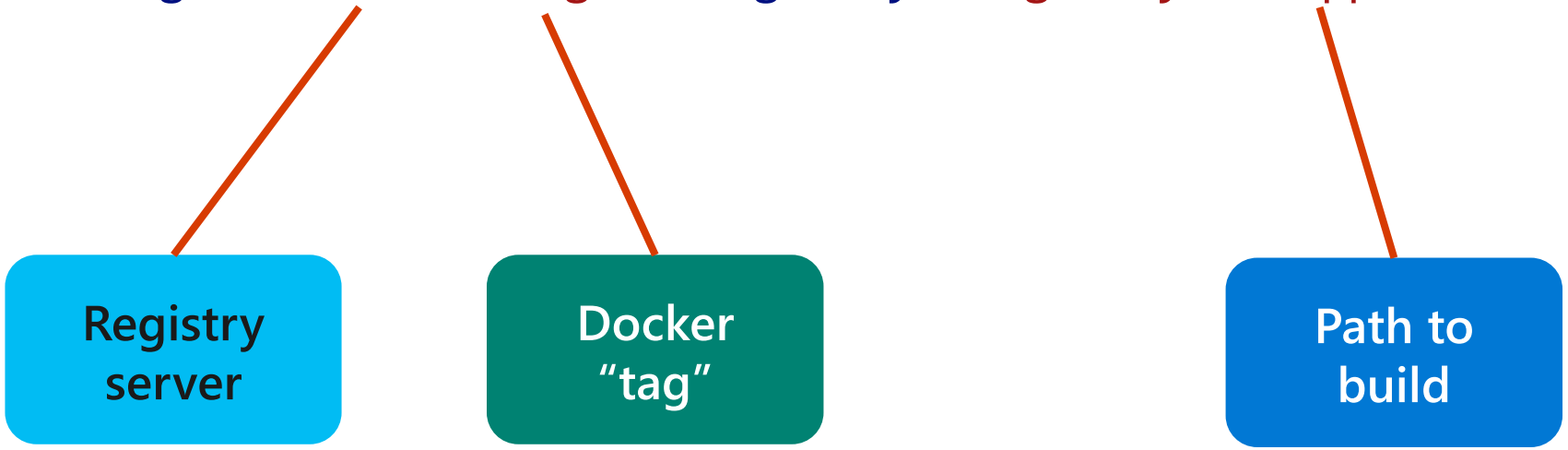
- Suite of features within Container Registry that provides streamlined and efficient Docker container image builds in Azure
  - Offloads **docker build** operations to Azure
  - Replaces manual build by using Docker tools on your local machine
  - Build on demand
- Fully automate builds with source code commit and base image update build triggers

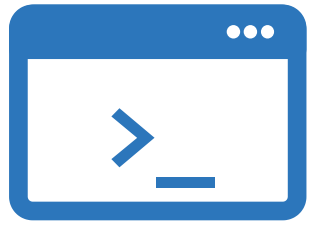
# Building images in Container Registry



# Trigger ACR Build by using Azure CLI

```
# Trigger build in Azure
az acr build --image <server>/<tag> --registry <registry> ./app
```



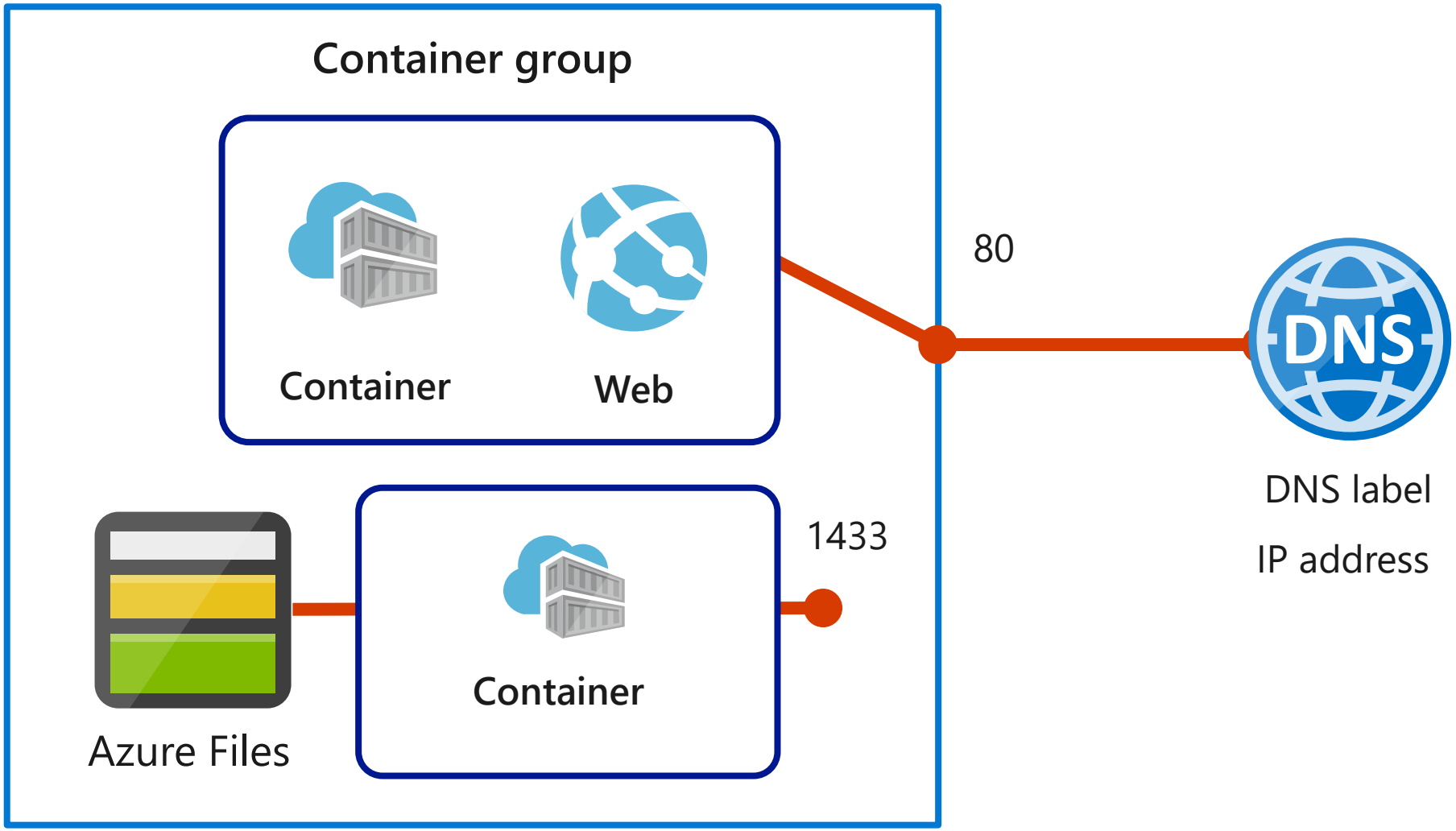



# Lesson 05: Create and run container images in Azure Container Instances

# Azure Container Instances (ACI)

- 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 instances:
  - 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

# Container groups

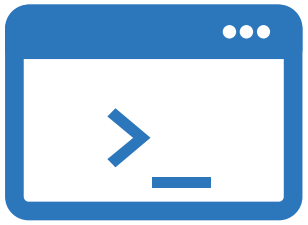


# Container Instances features

| 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 multicontainer groups that share host machine resources |
| **Virtual network deployment** | Container Instances can be deployed into an Azure virtual network |

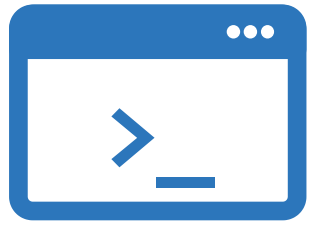
# Deploy a container to Container Instances

```
# Get name of container registry login server
az acr show --name <acrName> --query loginServer

# Get container registry password
az acr credential show --name <acrName> --query "passwords[0].value"

# Deploy container
az container create --resource-group myResourceGroup --name aci-tutorial-app --image
<acrLoginServer>/aci-tutorial-app:v1 --cpu 1 --memory 1 --registry-login-server
<acrLoginServer> --registry-username <acrName> --registry-password <acrPassword> --dns-
name-label <aciDnsLabel> --ports 80
```



# Verify a deployed container in Container Instances

```
# Verify deployment progress
az container show --resource-group myResourceGroup --name aci-tutorial-app --query
provisioningState

# View application URL
az container show --resource-group myResourceGroup --name aci-tutorial-app --query
ipAddress.fqdn

# View container logs
az container logs --resource-group myResourceGroup --name aci-tutorial-app
```



# Demonstration: Running Azure Container Instances by using Cloud Shell

# Introduction to Helm

# What is HELM


**Helm** is a tool that uses package management to enable creating, installing, and managing applications inside of Kubernetes much easier, configurable and repeatable.

# How doeas Helm work

# Why to use Helm?

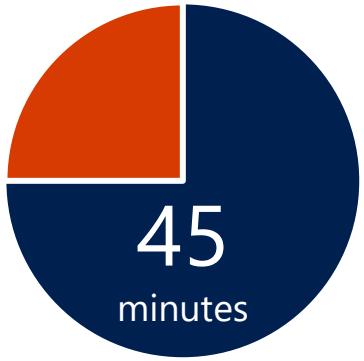
- Find prepackaged software (charts) to install and use
- Easily create and host your own packagesInstall packages into any Kubernetes cluster
- Query the cluster to see what packages are installed and running
- Update, delete, rollback, or view the history of installed packages

# What is a Helm Chart

- Basically it is an abstraction layer that defines the pods of a Kubernetes Application
- It is a package of filtes that will be used to deploy applications to Kubernetes
- Most important files:
  - Chart.yaml
  - Values.yaml

# Lab: Deploying compute workloads by using images and containers

## Duration




## Lab sign-in information