UNLOCKING CLOUD EXCELLENCE

THE CLOUD ENGINEER'S HANDBOOK :

AZURE CLOUD AND AZURE DEVOPS MASTERY

skillORA

AUTHOR

BY PRAVEEN GUDLA

Technical Manager, Solution Architecture

MTC Certified Trainer

Multi Cloud **Generative** AI

DevSecOps

Job-Ready Curriculum Framework

Well-designed according to industry standards.

Curriculum

A seasoned **DevSecOps** professional with extensive hands-on expertise in real-world **cloud solutions.**

- **17+ Years of IT experience**
  MNC companies
- **9+ Years of in experience**
  Cloud and DevOps Platform
- **6+ Years of Experience**
  Training & Mentoring

MCT CERTIFIED

Microsoft

Tech Mahindra

Capgemini

CSC

IBM

virtusa
Accelerating Business Outcomes

PRAVEEN G
Solution Architect, Technical Manager,
MCT Certified Trainer

Unlocking Cloud Excellence

The Cloud Engineer's Handbook: Azure Cloud & Azure DevOps Mastery

About Author:

Praveen G is a seasoned Cloud Architect with over 15 years of experience in IT infrastructure and Cloud Solutions. Specializing in Microsoft Azure and has helped numerous organizations migrate and manage their cloud environments.

Praveen is a MicroSoft Certified professional trainer and he has successfully trained many students and provided Seminars for Larger Audience.

Praveen G's passion for helping others navigate the world of cloud computing led him to write this book. He aims to provide readers with practical guidance, distilled from years of real-world experience, to master Azure administration and successfully manage cloud environments."

Iam very Thankful to

Mr. Naveen Gudla(GetEazy MD) and Mr. Kalyan Gudla for their unwavering support throughout my journey.

Mr. KrishnaReddy Kuchalakanti, Mr. ErwanthApp, Mr. Anand Barla Thank you for being my source of strength, my motivation, and my inspiration. I am forever grateful.

Shri Archana Gudla(Professor, JNTU **)** You are my greatest partner in life, and I am forever thankful for your love and support.

And all my Family, fellow colleagues and students, who has encouraged me to write this book

Disclaimer: The information provided in this book is based on the author's personal knowledge and experience with Microsoft Azure at the time of writing. While every effort has been made to ensure the accuracy of the content, the fast-paced nature of cloud technology means that tools, features, and interfaces may change over time. Readers are encouraged to refer to the official Microsoft Azure documentation and other authoritative sources for the most up-to-date information.

This book is intended for educational purposes only.

అర్జునుడు & శ్రీకృష్ణుడు – విజయం వైపు మార్గదర్శనం

అర్జునుడు: "కృష్ణ, ఈ శ్రరరంచం వేగంగా మారిపోతూ ఉంది. నాలో సందేహం ఉంది, మందుకు ఎలా సాగాలి?"

శ్రీకృష్ణుడు: "అర్జునా, భయం నీ శత్రువు! నిర్ణ్య లు తీసుకో, సాధన చెయ్, శ్రరయోగాలు చెయ్. విజయం నీ వంటే ఉంటంది!"

అర్జునుడు: "కృష్ణ, కానీ ఎటవంటి ఫలితం వసుతందో తెలియదు! ఏదైనా తపు అయితే?"

శ్రీకృష్ణుడు: "అర్జునా, విను – 'కర్మణ్యేవాధికార్స్తే మా ఫలేష్ కదాచన।'

నీ కర్తవ్యంనీ నిర్వరించు, కానీ ఫలంపై ఆశ పెటుకోకు. విజయం, రరాజయం అనేవి రండూ ఒకే నాణ్యనికి రండు శ్రరకకలు. నిజమైన శకి తశ్రశమ, రటుదల, నిర్ంతర్ అభ్యేసంలో ఉంది. అనుభవించు, నూతనమై ఆలోచంచు, భయానీ విడిచపెటు!"

Introduction to Cloud Computing

Cloud computing refers to the delivery of computing services—such as storage, servers, databases, networking, software, and analytics—over the internet, or "the cloud." Instead of relying on local servers or personal computers, organizations can access and utilize these resources on-demand from cloud service providers like Microsoft Azure, Amazon Web Services (AWS), and Google Cloud.

క్లౌడ్ కంప్యూటింగ్ అంటే, ఇంటర్నెట్ (క్లౌడ్) ద్వారా కంప్యూటింగ్ సేవలను అందించడం, మీరు మీ స్వంత Computers or Servers నిర్వహించకుండా, ఇంటర్నెట్ ద్వారా data storage, applications, servers, and other resources అనుకూలంగా access చేసుకోవడం.

ఇంటర్నెట్ ద్వారా Servers, Storage, Other Softwares లంటవి అదేదెక తీసుకోవడం

Evolution of Cloud Computing

Cloud computing has evolved from traditional IT infrastructure to virtualized environments and now to fully managed, scalable, and serverless computing solutions. Key milestones include:

- 1960s: Concept of time-sharing and utility computing emerges.
- 1990s: Virtualization and early web-based applications.
- 2000s: Emergence of large-scale cloud service providers like AWS, Microsoft Azure, and Google Cloud.
- 2010s & Beyond: Rise of AI-driven cloud services, hybrid and multi-cloud strategies, and edge computing.

Advantages of Cloud Computing:

Cloud computing is a popular option for people and businesses for a number of reasons including cost savings, increased productivity, speed and efficiency, performance, and security.

- **Cost Efficiency**
  - No Capital Investment for Infrastructure
  - Pay-As-you go model

- **Scalability and Flexibility** ○ On-Demand Scaling ○ Most of the cloud Providers having Data Centers worldwide

- **High Availability and Reliability**
  - ○ Redundancy - Providing High availability even incase of Hardware failures
  - ○ Automatic Backup and Recovery
- **Improved Collaboration**
  - ○ Centralized data access
  - ○ Integrate with many third party Applications or tools
- **Security**
  - ○ Identity access Management, encryption and access control etc.,
- **Innovation and Speed**
  - ○ Accelerate Infra and Application deployment
  - ○ Access and integrate Advance tools/Technologies like AI, Machine Learning, Bigdata, DevOps etc.,
- **Automatic Update and Maintenance**
  - ○ Cloud provider will take care of hardware and software maintenance (like security patches and updates)
- **Mobility**
  - ○ Remote access - access the services/resource anywhere and any device which is connected to the internet.

Challenges of Cloud Computing

- **Data Privacy & Security Risks:** Ensuring compliance with regulations.
- **Downtime & Reliability:** Risks associated with cloud provider outages.
- **Vendor Lock-in:** Difficulty in migrating between providers.
- **Cost Management:** Uncontrolled resource usage can lead to high costs.

Real-world Example: How Netflix Uses Cloud Computing

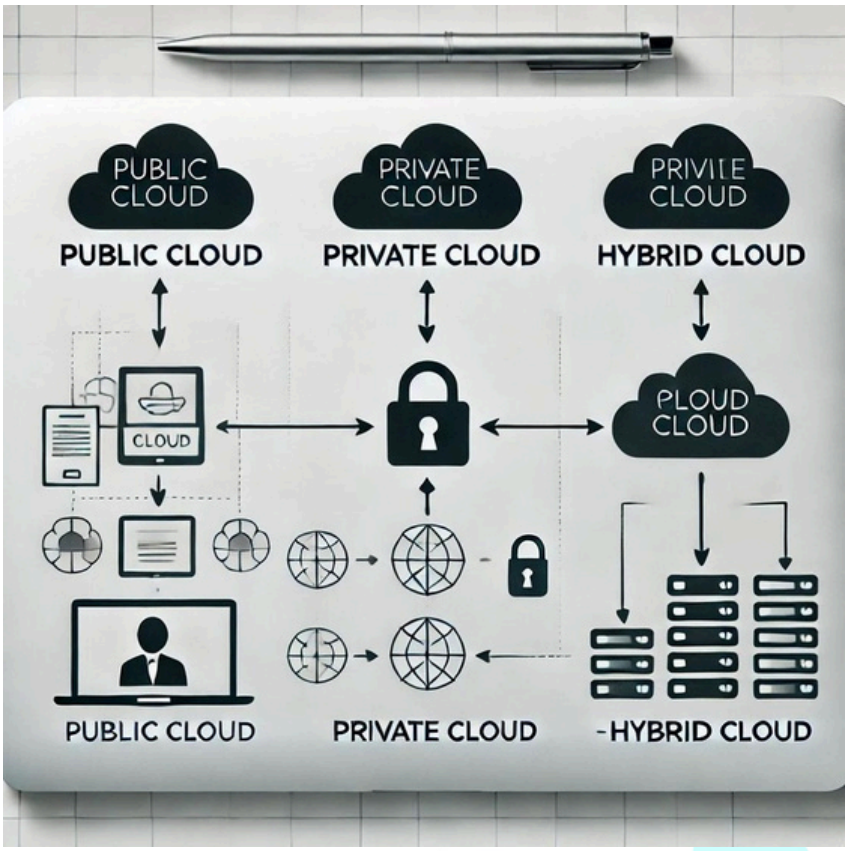
Netflix is one of the best examples of leveraging cloud computing to stream content worldwide. The company migrated its entire infrastructure to AWS, using services like:

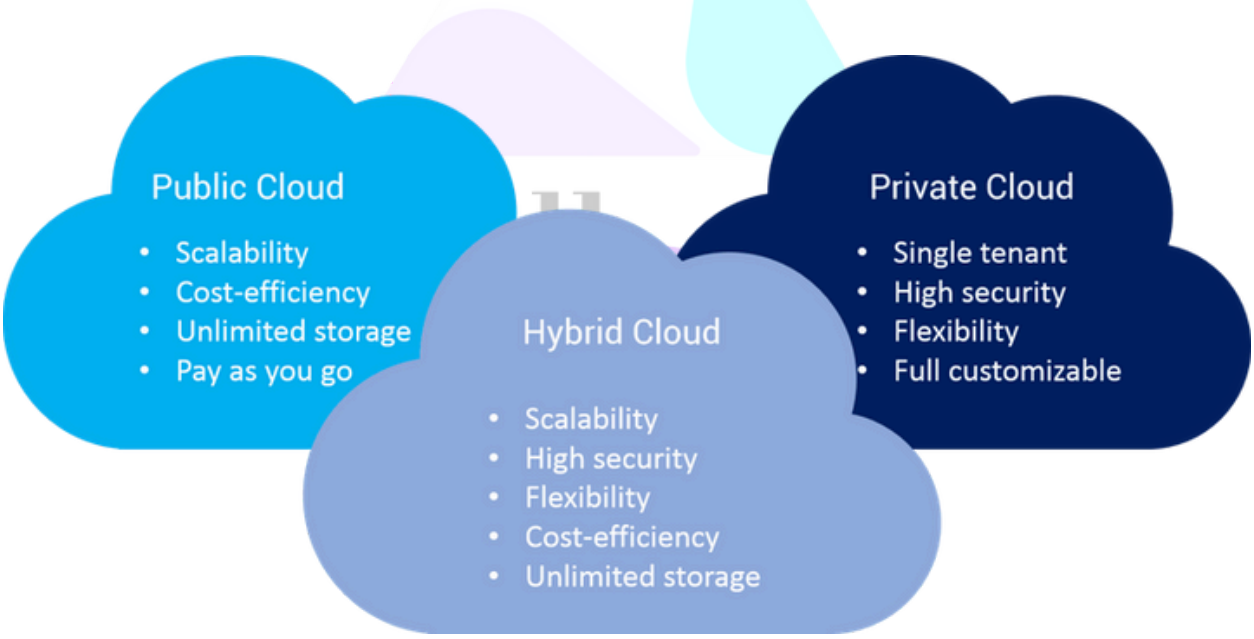
- **Amazon EC2:** To dynamically scale computing resources.

- **Amazon S3:** To store petabytes of video content.
- **CDN (CloudFront):** For fast and efficient global content delivery.
- **Big Data Analytics:** To recommend content using AI and machine learning.

Cloud Deployment Models

Deployment Model	Description	Real-World Use Case
Public Cloud	Resources are owned and managed by a cloud provider like Microsoft Azure.	A startup hosting its website on Azure.
Private Cloud	Infrastructure is dedicated to a single organization, managed internally or by a third party.	A bank running sensitive financial applications securely.
Hybrid Cloud	A mix of public and private clouds that work together.	A company storing customer data on a private cloud but using Azure for data analytics.


PUBLIC CLOUD
PRIVATE CLOUD
PRIVITE CLOUD
PUBLIC CLOUD
PRIVATE CLOUD
HYBRID CLOUD
CLOUD
PLOUD CLOUD
PUBLIC CLOUD
PRIVATE CLOUD
-HYBRID CLOUD


Public Cloud
Scalability
Cost-efficiency
Unlimited storage
Pay as you go
Private Cloud
Single tenant
High security
Flexibility
Full customizable
Hybrid Cloud
Scalability
High security
Flexibility
Cost-efficiency
Unlimited storage

IAAS, PAAS and SAAS

Adopting CloudComputing is a bigrevolution in the digital and IT industry. Instead of relying on physical servers and Infrastructure companies now have the option to leverage cloud services.

These services come in different models, each serving a specific purpose. The three main models are IaaS vs. PaaS vs. SaaS in cloud computing.

IAAS: Infrastructure as a Solution

IaaS is an excellent solution for businesses that require more control over their infrastructure.

Popular IaaS Providers:

- **AWS:** EC2, S3, VPC.
- **Azure:** Virtual Machines, Blob Storage, Virtual Networks.
- **GCP:** Compute Engine, Cloud Storage, VPC.

PAAS: Platform as a Solution

PaaS provides a platform that allows developers to build, deploy, and manage applications without worrying about the underlying infrastructure (servers, storage, network).

Popular PaaS Providers:

- **AWS:** Elastic Beanstalk, Lambda.
- **Azure:** App Service, Azure Functions.
- **GCP:** App Engine, Cloud Functions.

SAAS: Software as a Solution

SaaS is an excellent solution for businesses looking for easy-to-use, ready-to-go software solutions.

Popular SaaS Applications:

- **Google Workspace (Docs, Sheets, Drive).**
- **Microsoft Office 365.**
- **Salesforce CRM.**

IAAS: మీరు ర్నస్టారెంట్కు ఏరాటు చేయాలనుకంటునెన్నరు. మీక భూమి లేద్వ భవనం ఉండడం అవస్రం లేదు. మీరు కేవలం ఒక ఖాళీ స్క్లలం లేద్వ నిరాణం అదిదెక తీసుకంటారు అన్నె మీరు ఏరాటు చేయాలి (Interiordesign,utilitiesand Employees)

మీరు అన్నె నియంత్రంచుకోవాలనుకంటేసరైనఎంపిక .

PAAS: మీరు cooking Utensils and Interior Design అనె ఏరాటుడ అయ్యాన ర్నస్టారెంట్ అదిదెక తీసుకంటారు . కేవలం Cooking మరియు Service పై మాత్తమే మీరు దృష్టా ా స్థరించవచు.

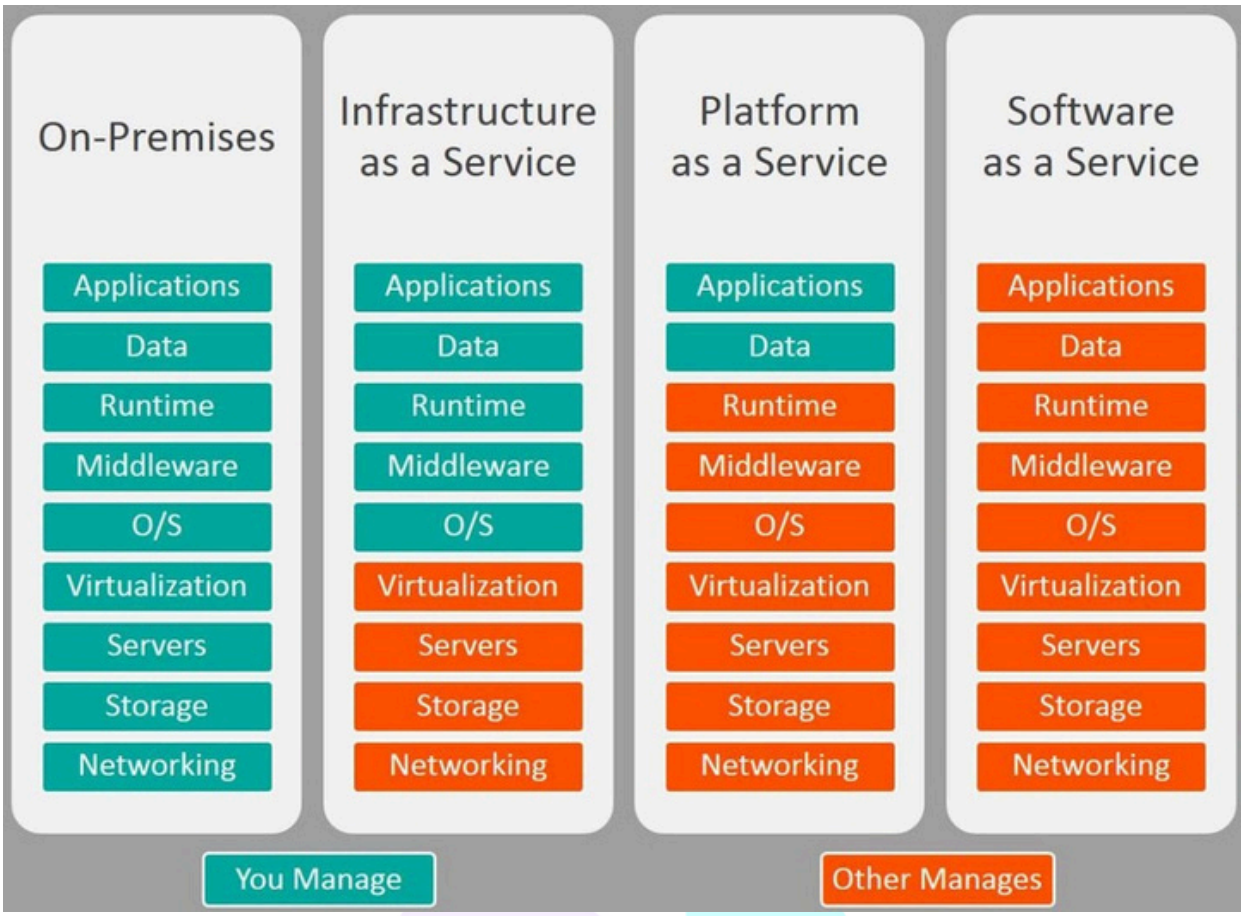
మీరు అభివృది ిపై దృష్ట ిసారించాలని ఉంటే మంచి ఎంపిక

SAAS: మీరు ర్నస్టారెంట్ నిర్హణపై ఎటువంట పాత్త పోష్టంచరు simple గ Restaurant లో భోజనం చేయడం.

సిదింిగా ఉనె స్టఫ్వేక్లర్ **వాడుకోవడంలో ఆసకి త**ఉన్నవాడు ైనద.

Comparison:

Feature	IaaS	PaaS Provides	SaaS
Definition	Provides virtualized computing resources over the internet.	developers to build applications without managing the underlying infrastructure.	Provides ready-to-use software over the internet.
User Type	IT administrators, system architects, DevOps engineers.	Developers, software engineers.	End users (businesses, individuals).
Managed By Provider	Storage, networking, virtualization, compute power.	Everything in IaaS + OS, Middleware, Runtime, and Development tools.	Everything in PaaS + Applications and Data.
Managed By User	OS, Middleware, Runtime, Applications, Data.	Applications and Data.	Nothing (only use the software).
Examples	Microsoft Azure VMs, AWS EC2, Google Compute Engine.	Azure App Service, AWS Elastic Beanstalk, Google App Engine.	Microsoft 365, Gmail, Dropbox, Salesforce.
Control & Flexibility	High – Users control OS, applications, and configurations.	Medium – Users control applications and data but not infrastructure.	Low – Users only use the software; no control over infrastructure.
Use Case	Hosting websites, virtual machines, storage, backup.	Developing, testing, and deploying applications.	Email services, collaboration tools, CRM, productivity apps.
Scalability	High – Users scale resources as needed.	Medium – Platform scales based on development needs.	Low – Limited to the features of the software.
Cost Structure	Pay for what you use (compute, storage, networking).	Pay for usage of development tools and runtime.	Subscription-based pricing.
Best For	Companies needing full control over infrastructure.	Developers needing a managed environment to build applications.	Businesses and individuals who need ready-made software.


On-Premises
Infrastructure as a Service
Platform as a Service
Software as a Service
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
You Manage
Other Manages

Real-World Use Cases and Exercises

Exercise 1: E-Commerce Startup

Scenario: You are the CTO of an e-commerce startup planning to launch a website with the following requirements:

- Host the website with scalable infrastructure.
- Store and process user data securely.
- Implement payment processing with minimal downtime.

Task: Identify the cloud service model (IaaS, PaaS, SaaS) for each component:

- Web Hosting (IaaS or PaaS?)
- Database Management (IaaS or PaaS?)
- Payment Gateway (SaaS?)

Exercise 2: DevOps in Cloud Computing

Scenario: Your company is adopting DevOps practices to improve software deployment efficiency. You need:

- A scalable infrastructure to host applications.
- Automated CI/CD pipelines for continuous deployment.
- Monitoring and logging solutions for real-time insights.

Task: Identify cloud services that fit into IaaS, PaaS, and SaaS categories for:

- Compute resources (Virtual Machines, Kubernetes, or Serverless?)
- CI/CD tools (Azure DevOps, GitHub Actions, AWS CodePipeline?)
- Monitoring solutions (Azure Monitor, AWS CloudWatch, GCP Operations?)

Exercise 3: SaaS-Based CRM Solution

Scenario: Your company wants to implement a Customer Relationship Management (CRM) tool to track customer interactions.

- Should you develop a custom CRM using IaaS or PaaS?
- Should you use an existing SaaS solution like Salesforce, HubSpot, or Microsoft Dynamics?

Task:

- Compare the pros and cons of building vs. buying a SaaS solution.
- Create an architecture diagram for a custom CRM built on cloud infrastructure.

Conclusion

By understanding cloud service models, businesses can choose the right mix of IaaS, PaaS, and SaaS to optimize costs, scalability, and efficiency.

AWS vs. Azure vs. GCP-A Comparative Study

Feature	**AWS**	**Azure**	**GCP**
Compute	EC2, Lambda	Virtual Machines, Functions	Compute Engine, Cloud Functions
Storage	S3, EBS, Glacier	Blob Storage, Files	Cloud Storage, Filestore
Networking	VPC, Direct Connect	VNet, ExpressRoute	VPC, Interconnect
Security	IAM, KMS, WAF	Active Directory, Sentinel	IAM, KMS, Security Command Center
AI/ML	SageMaker, Rekognition	Azure ML, Cognitive Services	Vertex AI, AutoML
Best For	Scalability & maturity	Hybrid & enterprise integrations	AI & data analytics

Networking Fundamentals

Simpledefinitionfornetworkingis,howdata is exchanged between devices, servers, and systems.

In Any Cloud environment, we are responsible for virtual network configurations and physical hardware is managed by Cloud Vendor.

Networking is the backbone of cloud computing, ensuring secure and efficient communication between resources, services, and users. Cloud networking abstracts traditional networking concepts and offers highly scalable and flexible connectivity.

We have to make sure understanding the fundamentals of Networkings like., switch, Router, public DNS,private DNS, DHCP, AD, Firewall, etc.,

Real-World Example - How Enterprises Use Cloud Networking:

Senario-1: A global e-commerce company needs **low-latency connectivity** across regions.

Solution: They use **Azure Virtual Network (VNet) peering** and **AWS Direct Connect** to interconnect their offices and cloud workloads.

Senario-2: Banking Industry – Secure Hybrid Cloud:

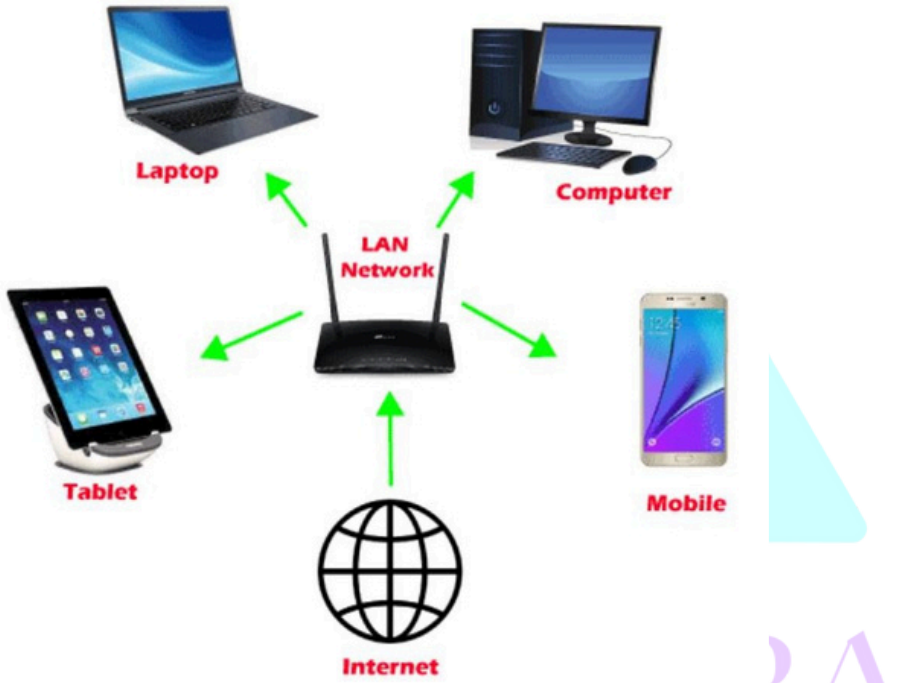
A bank needs a **secure on-prem-to-cloud connection**.

Solution: They use **Azure ExpressRoute** with **a firewall appliance** for secure, high-speed connectivity.

Types of Networking:

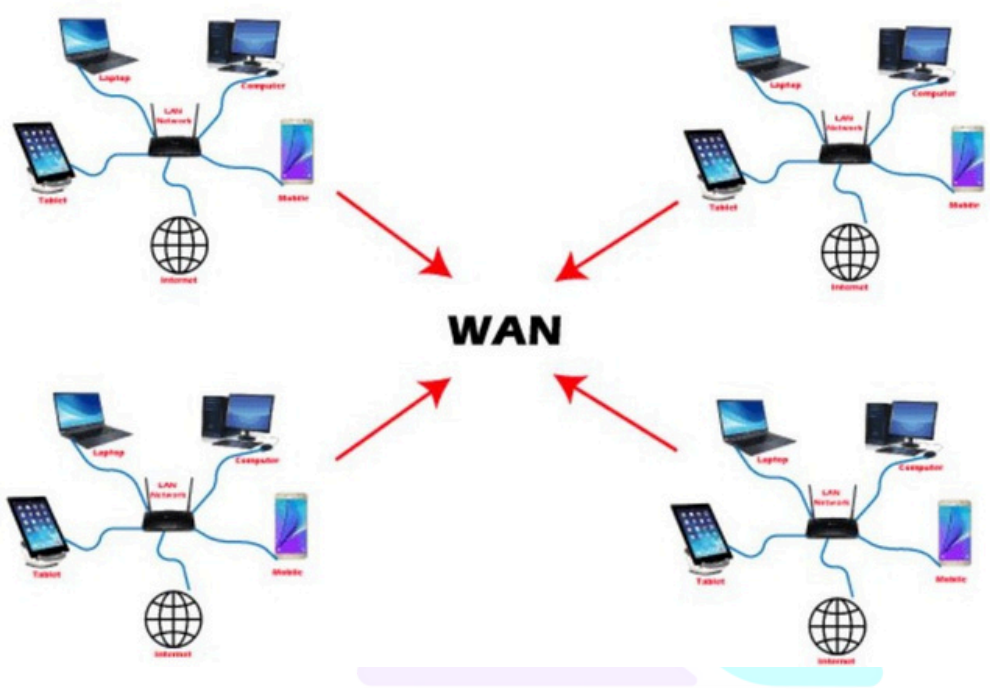
- **LAN(Local Area Network) :** Home or With in Office

LAN అనేద ఒకపరిమితత్వాంతంలోఆఫీస్ , స్కూల్ఇంట) కంప్యూటరుడ మరియు పరికరాలను కలిపే నెట్వ్లవార్.



- **WAN(WideArea Network):** Connecting computers over a wide geographical area over theInternet

WAN అనేద వివిధ ాంతాలోడ ని అనేకLAN లను ఇంటర్నెట్ లేద్వ లీజ్డ్ లైన్స్ ద్వారా అనుసంధానిసుతంద



- **DNS: Domain Name System:**

DNS translates human-readable domain names (e.g., www.google.com) into **IP addresses** (e.g., 142.250.190.14).

DNS అనేద మనం చదవగలిగే వెబసైట్లెట్ పేర్లుడ(www.google.com) కంప్యూటర్ల్లక అర్మలయు**IP అశ్రడస్** (142.250.190.14) గా మారుసుతంద.

Example: When you type www.microsoft.com, DNS resolves it to the actual IP address of Microsoft's web server.

- **DHCP : Dynamic Host Configuration Protocol**

DHCP automatically assigns **IP addresses, subnet masks, default gateways, and DNS settings** to devices in a network.

DHCP అనేద **IP అశ్రడస్, సబనేట్ మాస్క్, డిఫాల్టు గేట్‌వే, DNS సెటింగ్స్**ుంట వాటని ఆటోమేటకగాక్ల పరికరాలక కేటాయ్యసుతంద.

Example: When you connect to a Wi-Fi network, your device gets an IP address from the DHCP server.

- **Firewall:**

A **firewall** is a security system that monitors and controls incoming and outgoing network traffic based on security rules.
Firewall అనేద నెట్క్లవార్‌లోక్ల త్వవేశంచే మరియు బయటకి వెళ్లడ ఫ్ాఫికనుక్ల పరిశీలించి నియంత్రంచే **సెక్యేరిటీ సిసమ్**ు

Example: An organization uses an **Azure Firewall** to block unauthorized access to its cloud resources.
Load Balancer:
A **Load Balancer** distributes incoming network traffic across multiple servers to ensure availability and prevent overload.
Load Balancer అనేద **నెట్‌వర్క్క ఫ్రాఫిక్‌ను బహుళ సర్వర్ లమధే సములేంగా రంపిణీ** చేసి, ఫ్రివాస్ అందుబాటులో ఉండేల చేసుతంద.
Example: Azure Load Balancer distributes traffic among multiple VMs for high availability.

- **NAT -Network Address Translation:**

NAT translates private IP addresses (used inside a network) into public IP addresses (used on the internet)

NAT అనేద ప్రైవేట్ **IP అశ్రడస్లను రబ్లకల IP అశ్రడన్లుగా మారి**, అనేక పరికరాలక ఒకే పబ్లకడ IP
ను ఉపయోగంచేల చేసుంతద.

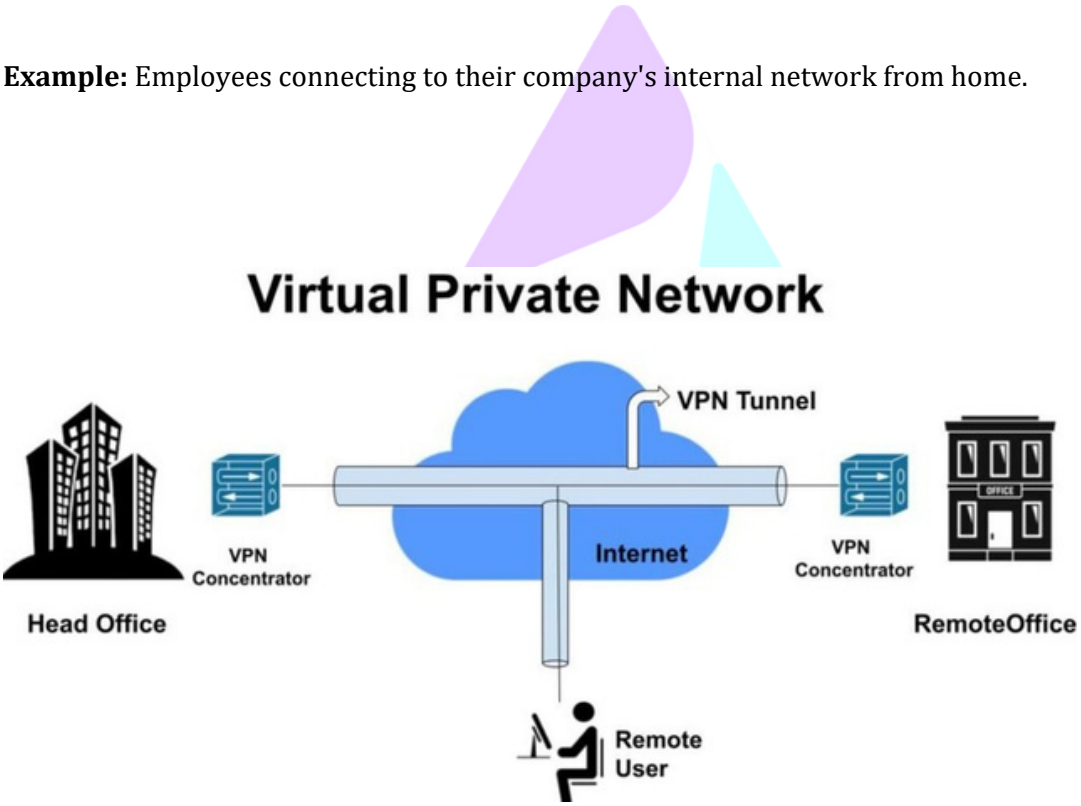
Example: Home routers use NAT to allow multiple devices to connect to the internet using one public IP.

VPN : Virtual Private Network:

A **VPN** creates a secure, encrypted connection over the internet, allowing users to access private networks remotely.

VPN అనేద ఇంటర్నెట్ ప్రైవేట్ల సుర్షితమైన, ఎన్క్రిప్టెడ్ం కనెక్షన ని ఏర్పర్చి, ప్రైవేట్ నెట్వర్క్ార్క్లలను రిమోట్గా యాక్స్స చేయడానికి అనుమరసుతంద.

Example: Employees connecting to their company's internal network from home.

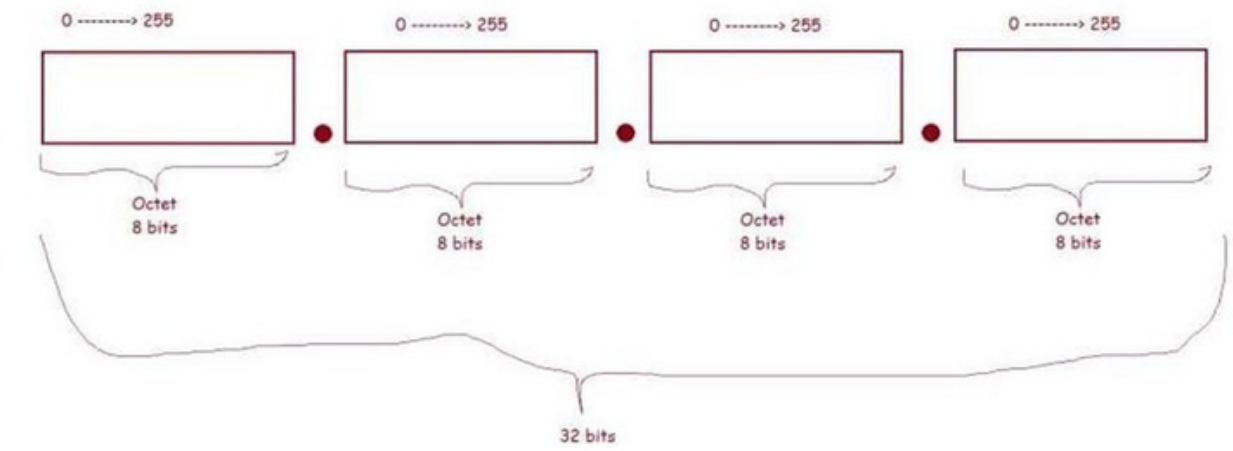


IP Addressing: A unique identifier for devices on a network.

Example: Every website and device on the internet has a unique IP address.

- **IPv4**: 32-bit address (e.g., 192.168.1.1).

- **IPv6**: 128-bit address, designed to handle more devices (e.g., 2001:0db8:85a3:0000:0000:8a2e:0370:7334).



Public IP:

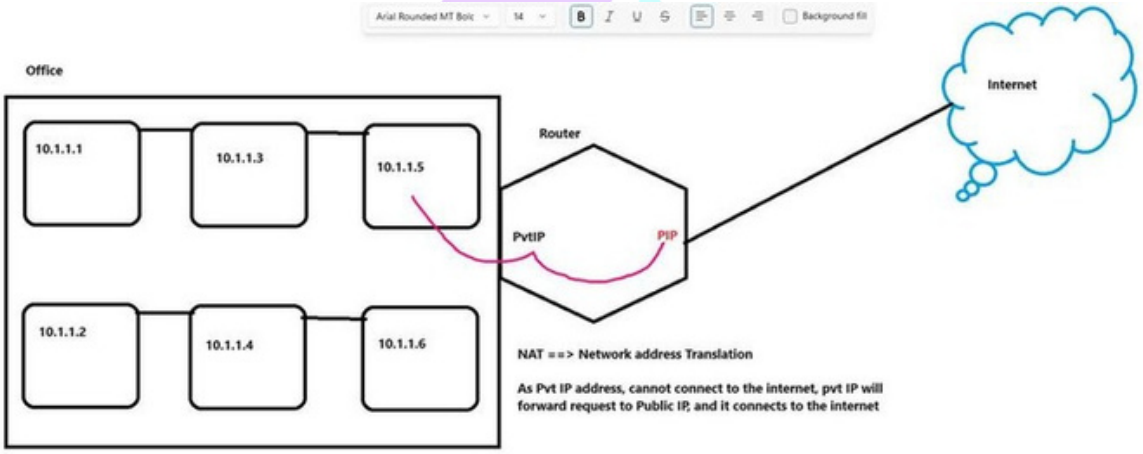
- An Address used for devices over the internet.
- These IP addresses are paid; we need to purchase from a service provider.
- Except Private IP range, rest of the all the IP's Public IPs

Private IP:

- An Address used for devices within a private Network.
- We can't use these IP addresses for Internet use.

Class	Private address range	
	start address	finish address
A	10.0.0.0	10.255.255.255
B	172.16.0.0	172.31.255.255
C	192.168.0.0	192.168.255.255

Class	Public address range	
	start address	finish address
A	0.0.0.0	126.255.255.255
B	128.0.0.0	191.255.255.255
C	192.0.0.0	223.255.255.255
D	224.0.0.0	239.255.255.255
E	240.0.0.0	254.255.255.255



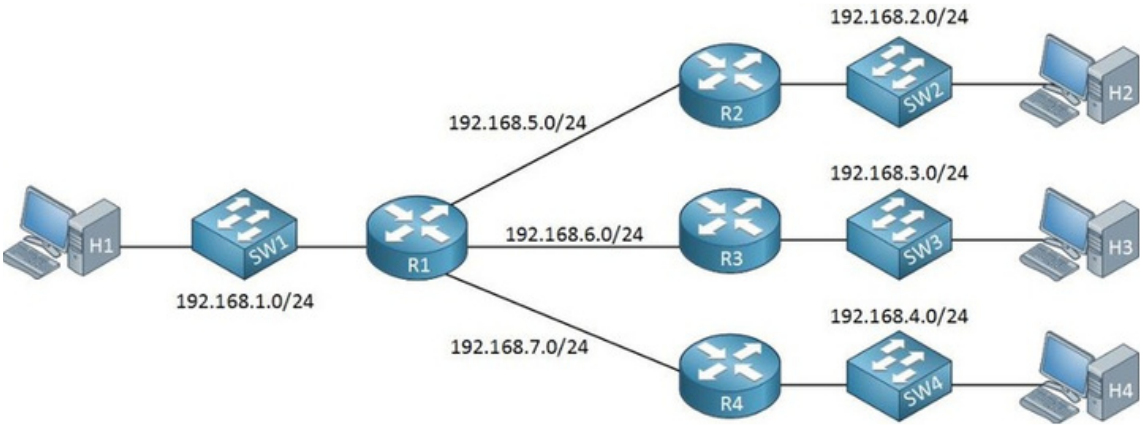
Subnetting :

Divide a large network into a smaller network.

When a bigger network is divided into smaller network, to maintain security, then that is known as Subnetting.

సబ్‌నెట్అనేదనెట్‌వర్క్‌నుచనిభ్యగాలుగావిభజంచడందీన్నివలడభత్తతమరియుపనితీరు మెరుగుపడుతంద.

Example:



Transmission rate or Bandwidth: How fast information can be transmitted over the internet/channel, it is measured in **bits per second(BPS).**

Simplex: Information transmitted only in one direction
Half-Duplex: Information transmitted in both directions, but only in one direction at a time
Full-Duplex: Information can be transmitted in both directions simultaneously

Backbone Network: There are High-Bandwidth channels that typically connect LANs with each other called Backbone Network.

Typically, in a Cloud all the Physical Datacenter connected each other High Bandwidth of Backbone Network.

Azure Fundamentals

History of Azure

Microsoft announced **Windows Azure** in October 2008 and available for customers/client on February 2010.

Microsoft rebranded Windows Azure to Microsoft Azure in the year of 2014

2015 to 2024 Microsoft included significant features like Azure AD, VM with different OS flavors, Machine Learning Tools, AI, Big Data, AKS, Cognitive Service, Azure Arc, Azure Open AI etc.,

Azure Data Centers and Regions:

Understanding Azure Data Centers and regions is very crucial when we deploying resources in azure,

Azure Geographies:

- United States
- Europe
- Asia Pacific
- Middle East
- China (operated by 21Vianet, an independent company under Chinese laws)

Azure Regions

Azure regions are geographical areas that contain one or more physical data centers

Azure ప్రాంతాలు అంటే భౌగోళికంగా వేర్వారు ప్రదేశాలోడ ఉనె డేటా సంటరుడ. ఇవి Microsoft నిర్హంచే డేటా సంటరుడ, అకూడ నుండి క్లడ్ సేవలు అందంచబడతాయ్య.

త్వర Azure ాంతంలో ఒకట లేద్వ అంతకంటే ఎకూవ డేటా సంటరుడ ఉంటాయ్య. వినియోగద్వరులు తమ అవస్రానికి తగటుగడ ఏ ాంతాని ఉపయోగంచాలో ఎంచుకోవచు. ఇద వేగంగా భత్తంగా మరియు నమాకంగా సేవలను అందంచడానికి స్థాయపడుతంద .

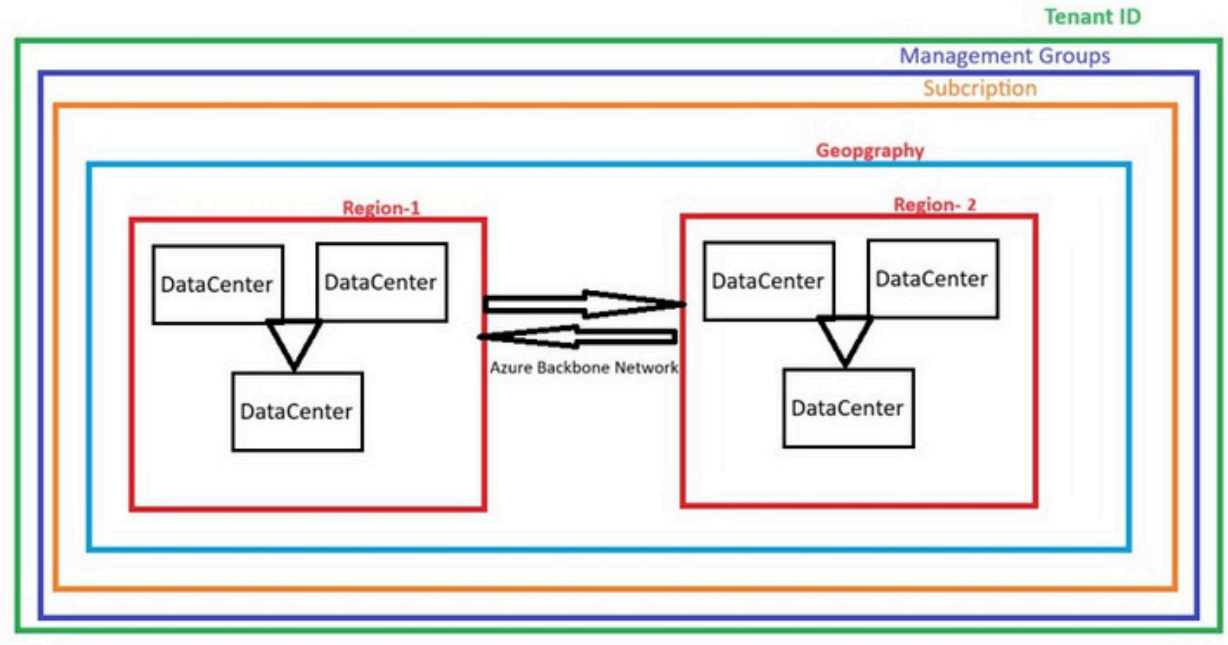
Examples of Azure Regions:

- East US, West US
- North Europe, West Europe
- Southeast Asia, Japan East
- Australia East, Brazil South

Azure Region Pairs

Region pairs are two Azure regions within the same geography that are connected and designed to support data replication and failover scenarios.

Azure Availability Zones : Physical Data Centers in Region

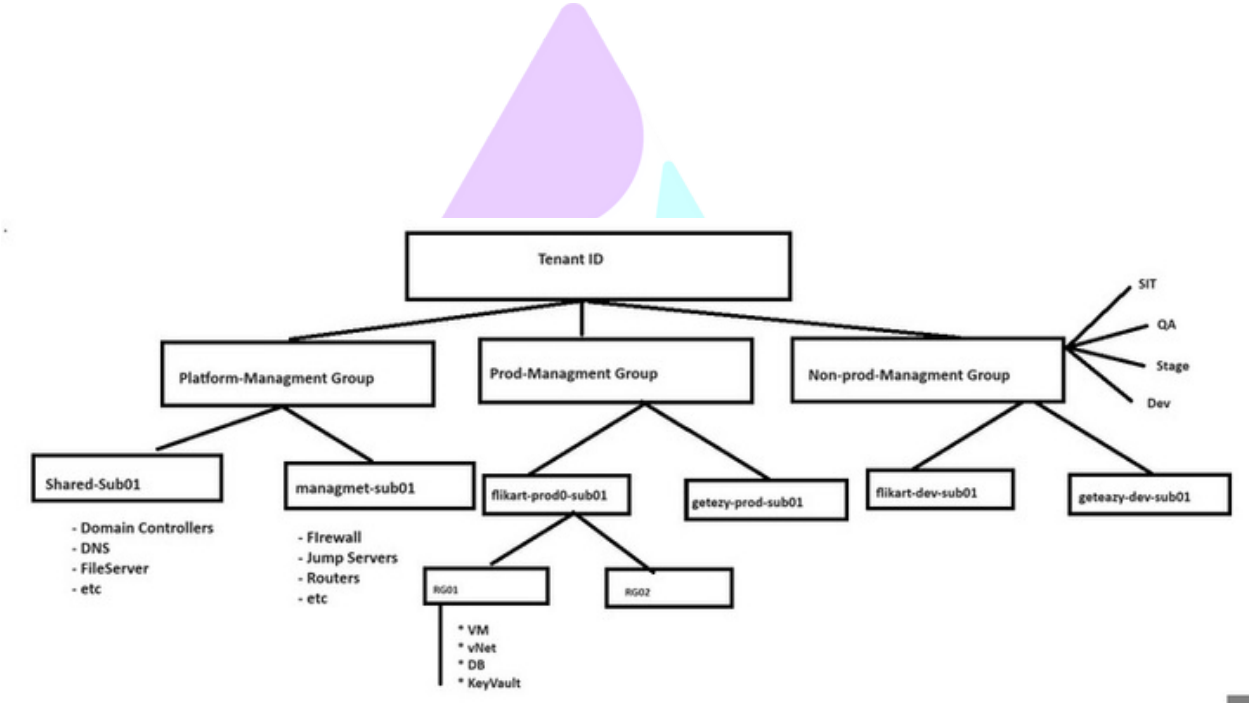


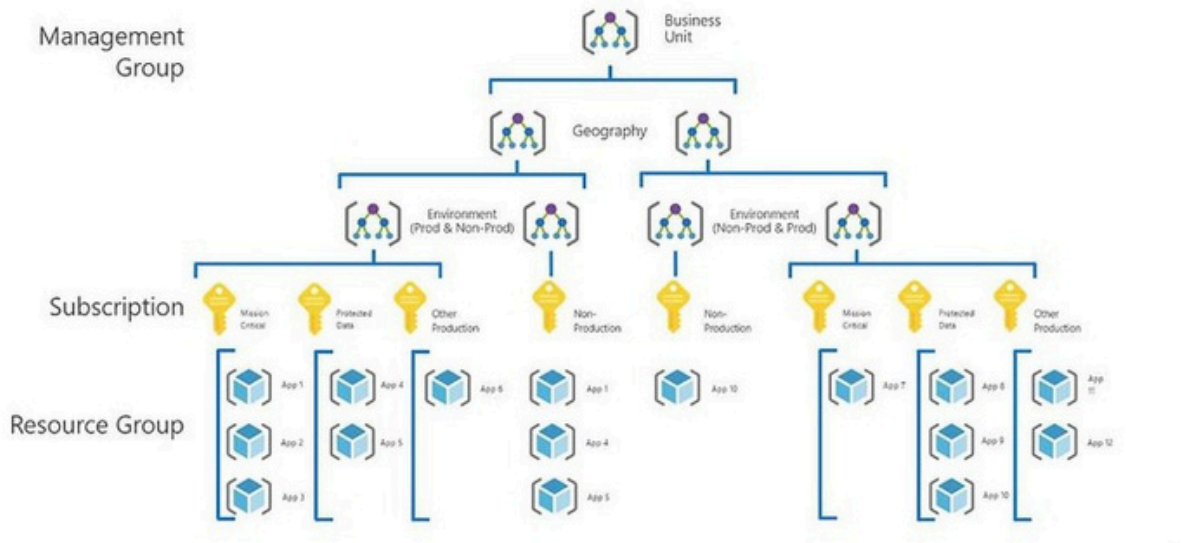
Azure Hierarchy:

Large and complex environments/Organizations follow the Azure Hierarchy for structured way of organizing and managing structured way of organizing and managing resources resources (like VM's,Storage,DBs,AppService.,etc).

Below are Advantages by implementation of Azure Hierarchy:

- Centralized Management and Governance
- Security and access control
- Cost Management and Optimization
- Scalability and flexibility
- Easy delegation and administration





Key Elements of Azure Hierarchy:

Tenant ID: when we registered with Microsoft Azure will get Tenant ID and Azure Active Directory for Identity and access management.

Azure Tenant ID అంటే **Azure Active Directory (Azure AD)** లో మీ స్సంస్కల అనుస్సంధానమైన **యూనిక (శ్రతేకమైన) గురింతపు సంఖ్**

Management Groups: Will help grouping of Subscription for easy and flexible for organizing the policies, access controls and compliance.

Subscriptions: it is a logical container to hold the azure resources. Each Subscription has its own billing, quota and access management.

Resource Group : it is a logical container to hold the azure resources, each and every resource which we creating in azure should be part of Resource Group.

Resource: Individual Services which we deploy in Azure(Like vNet,VM,SQLDB, AppService.,etc)

Real-World Scenario:

VIITOR Technologies is a **global IT services company** with operations in **India, USA, and UK**

Business Challenge:

- Different teams (**Development, Testing, and Production**) manage cloud projects.
- They need **cost-efficient, secure, and structured cloud resource management**.
- They require **centralized billing and security policies**.

Solution: Implementing Azure Hierarchy

Hierarchy Level	**VIITOR Technologies Example**
Management Group	VIITOR-Global-Mgmt – Controls all Azure subscriptions
Subscriptions	VIITOR-Prod-Subscription, VIITOR-Dev-Subscription
Resource Groups	VIITOR-Dev-App1-RG, VIITOR-Prod-ERP-RG
Resources	Virtual Machines, Storage, Databases, Networking

LAB1: Create an Azure Account

Objective: Set up a free Azure account to access cloud services

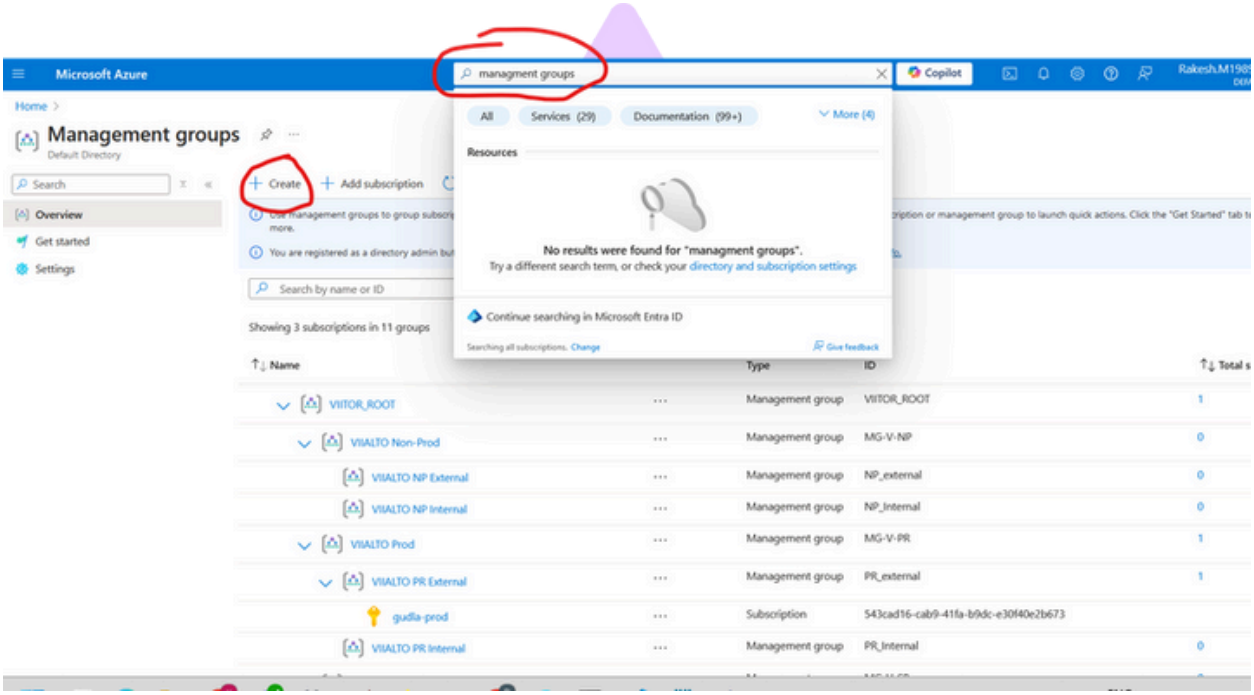
Steps:

- Go to Azure Portal(http://portal.azure.com)
- Click Start Free and sign in with a Microsoft account.
- Enter personal details (Name, Email, Phone).
- Verify identity using Credit/Debit card (No charges for free services).
- Click Sign Up → Your Azure free-tier account is ready!

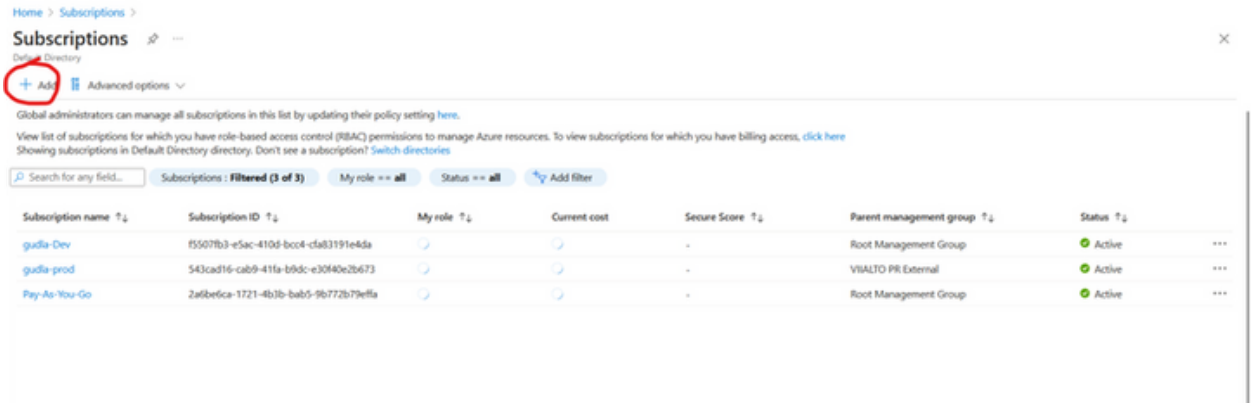
LAB2: Create Management Groups

Objective: Organize Azure subscriptions under a **Management Group**.

- Sign in to Azure Portal → Search for Management Groups.
- Click + Create Management Group.
- Management Group ID → VIITOR-Global-Mgmt
- Display Name → VIITOR Technologies Global
- Click Create → Now, this group can manage multiple subscriptions.



LAB3: Try create multiple Subscriptions



LAB 3: Create Multiple Subscriptions

Objective: Set up different subscriptions for projects (e.g., Dev & Prod).

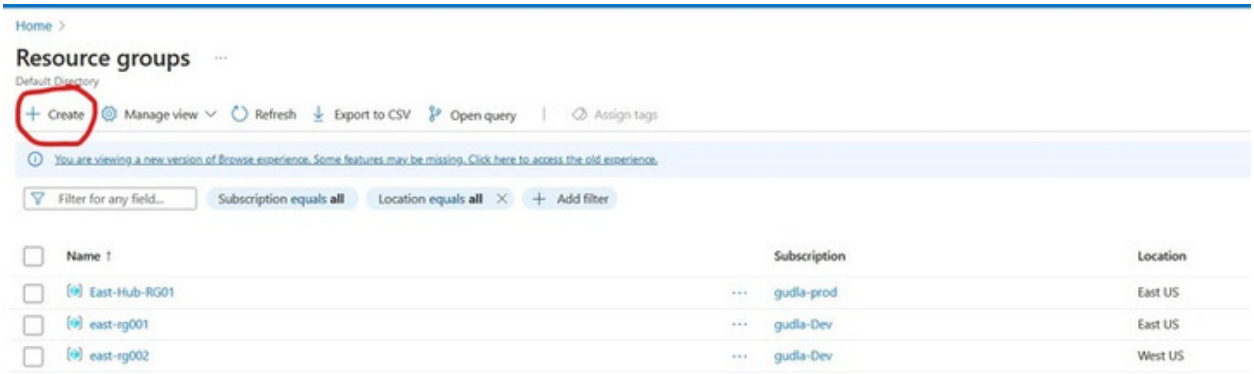
- Go to Azure Portal → Search for Subscriptions.
- Click + Add Subscription.
- Select Offer Type → Choose Pay-As-You-Go or Free Trial.
- Enter Subscription Name:
- VIITOR-Prod-Subscription (For Production)
- VIITOR-Dev-Subscription (For Development)
- Assign them to the VIITOR-Global-Mgmt group.
- Click Create → Your subscriptions are ready!

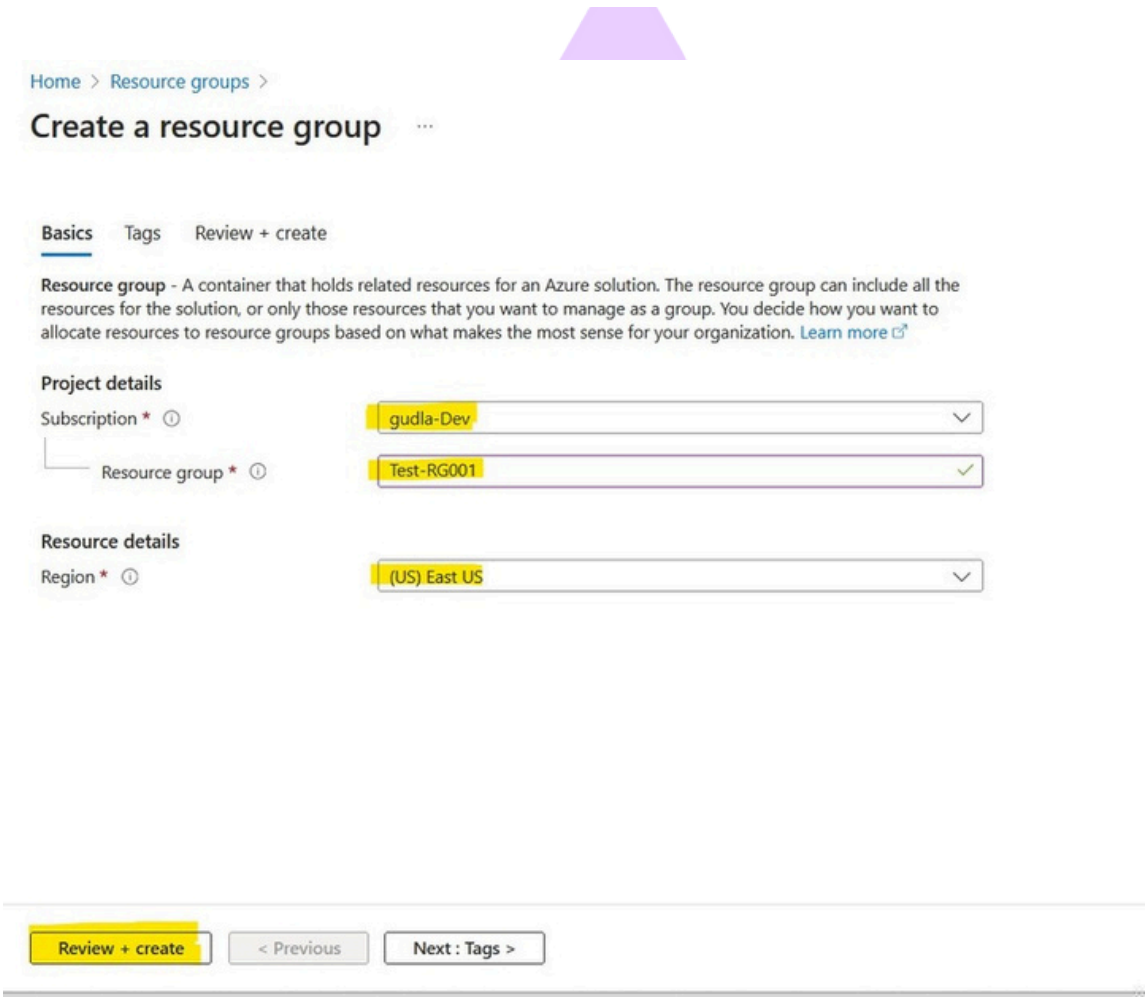
LAB4: Create a Resource Group

Objective: Organize related cloud resources into a **Resource Group**.

- **Sign in to Azure Portal → Search for Resource Groups.**
- **Click + Create Resource Group.**
- **Resource Group Name → VIITOR-Dev-App1-RG**
- **Subscription → Select VIITOR-Dev-Subscription**
- **Region → Choose a data center (e.g., East US).**

- **Click Review + Create → Your Resource Group is set up!**





Note: Make sure we providing all the Mandatory values

Azure Networking Essentials

Azure Networking is very critical component and help establish secure communication.

Networking is a crucial part of cloud computing, enabling communication between resources, users, and services. Microsoft Azure provides a highly secure and scalable networking architecture to support cloud deployments.

Azure Networking అనేద **Microsoft Azure** క్లడ్‌లోక్ల **నెట్‌వర్క్క కనెక్టివిటీ మరియు కమ్యూనికేషన** ను నిర్వహించేందుక ఉపయోగంచే సేవల స్యూహం.

Real-World Scenario: VIITOR Technologies Cloud Network

- They need secure communication between services
- They want high-speed connectivity for global users
- They require network isolation for different teams

VIITOR Technologies is migrating its **global IT infrastructure** to **Azure Cloud**.

Azure Networking Key Components

Service	Purpose	Example Use Case
Virtual Network (VNet)	Private network in Azure	VIITOR's internal servers communicate securely
Subnet	Divides a VNet for different resources	Web and Database servers in separate subnets
Network Security Group (NSG)	Controls traffic rules	Allows only HTTPS traffic to web servers
Azure Load Balancer	Distributes traffic across VMs	Ensures high availability for web apps
Azure Application Gateway	Layer 7 load balancing	Routes requests based on URLs

VPN Gateway	Connects on-premises to Azure	Secure hybrid cloud connectivity
Azure Bastion	Secure VM access without RDP	Protects against brute-force attacks

Virtual Network(vNet):

An **Azure Virtual Network (VNet)** is a logical representation of our network in the cloud

Or

An **Azure Virtual Network (VNet)** is a **private, isolated network** in Azure, similar to an on-premises data center network. It allows resources like Virtual Machines, Databases, and Applications to communicate securely

Azure Virtual Network (VNET) అనేది ప్రైవేట్ నెట్‌వర్క, ఇది అజ్యేర్క్‌లో వనర్జలను ఒకదానితో ఒకటి సుర్షితంగా కమ్యేనికేట్ చేస్తందుకు సహాయరడుంది.

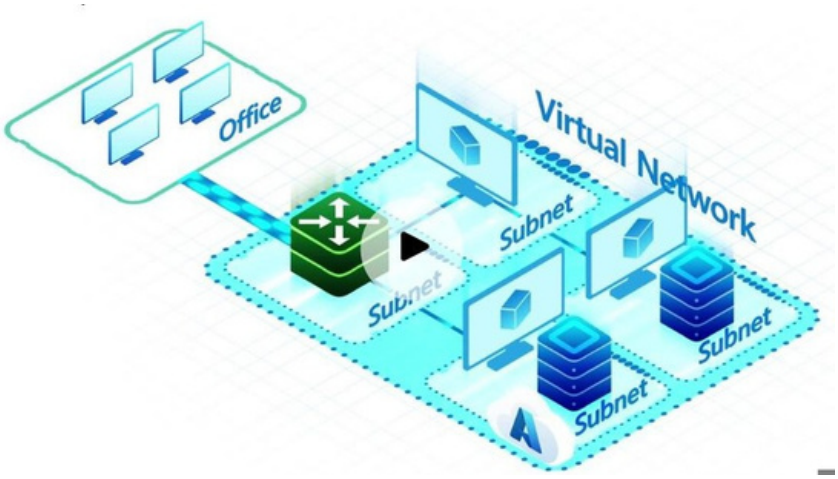
ఇద Azure లో **ప్రైవేట్ నెట్‌వర్క** లంట విధంగా పనిచేసుతంద, ఇందులో **VMs (వర్జివల్ట మిష్టనుల),** డేబేస్, మరియు ఇతర్ స్తవలు ఒకద్వనితో ఒకట కమూానికేట్ అవావచు.

Key Features of Azure VNET:

- **Isolation:** Each VNET is isolated from other networks for security.
- **Subnets:** Divide a VNET into smaller segments to organize and secure resources.
- **Network Security Groups (NSG):** Control inbound and outbound traffic.
- **VPN & ExpressRoute:** Enable secure connections to on-premises networks.
- **Load Balancers & Firewalls:** Enhance security and optimize traffic distribution.

vNet will help on below scenarios:

- Communication of Azure resources with the internet.
- Communication between Azure resources.
- Communication with on-premises resources.
- Filtering of network traffic.
- Routing of network traffic.
- Integration with Azure services.



vNet Best Practices:

- Ensure no overlapping address spaces
- subnets shouldn't cover the entire address space of the virtual network.
- Recommended to take Larger CIDR range instead of taking small Network
- Ensure attaching the NSGs and Route tables for High security

Real-World Scenario: Securing a Web Application with Azure Networking

Imagine an e-commerce company migrating its website to Azure. The company needs a secure, scalable, and highly available network setup. Here's how they design their network:

- **Web Tier (Public Subnet):** Hosts web servers behind a Load Balancer.
- **Database Tier (Private Subnet):** Stores customer data with restricted access.
- **Firewall & NSG:** Protect against unauthorized traffic.
- **VPN Gateway:** Securely connects on-premises users to Azure.

LAB: Create a vNet at east us region with 4 dedicated Subnet

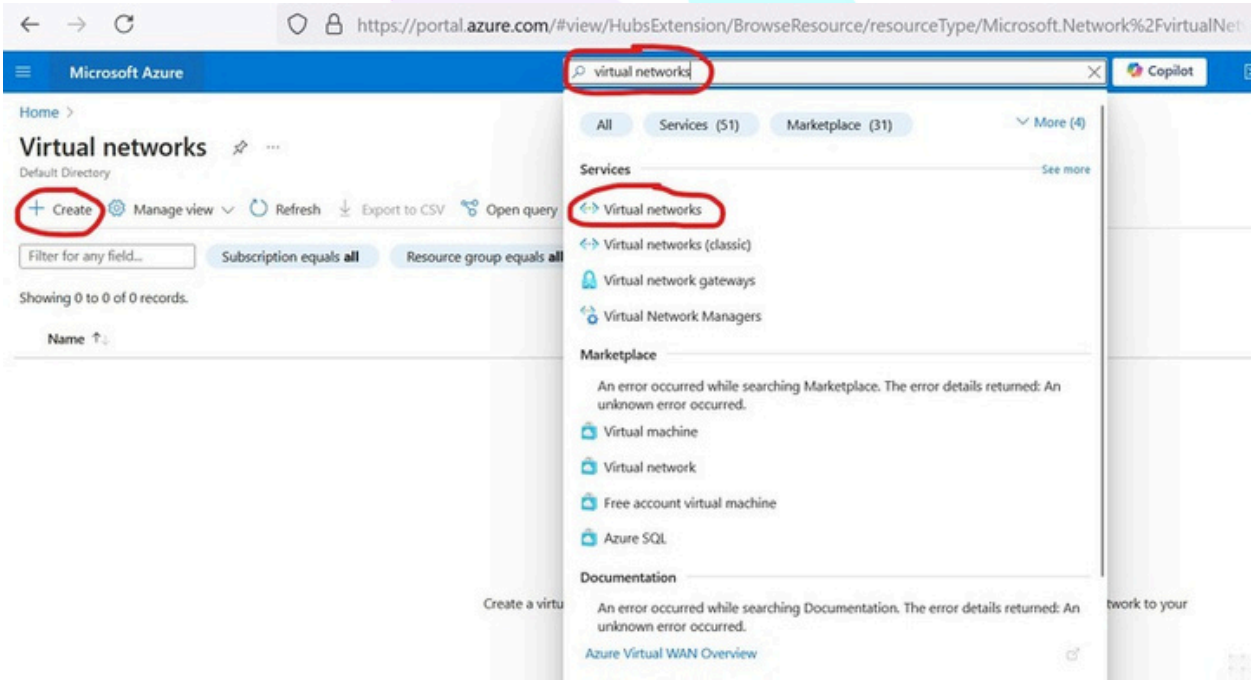
Pre-requisite:

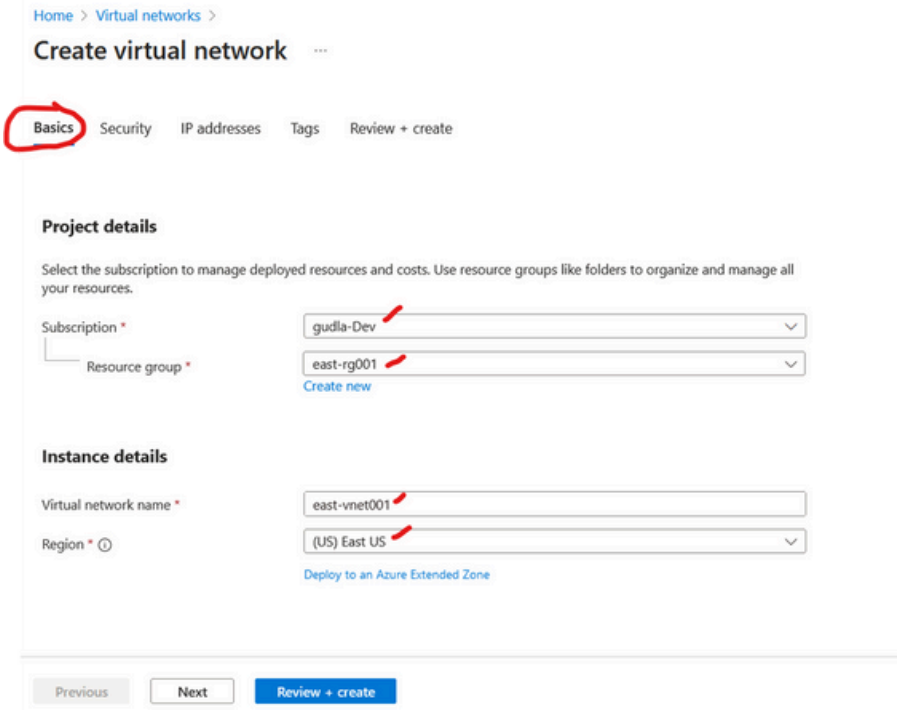
- Create a Resource Group
- Choose the CIDR range

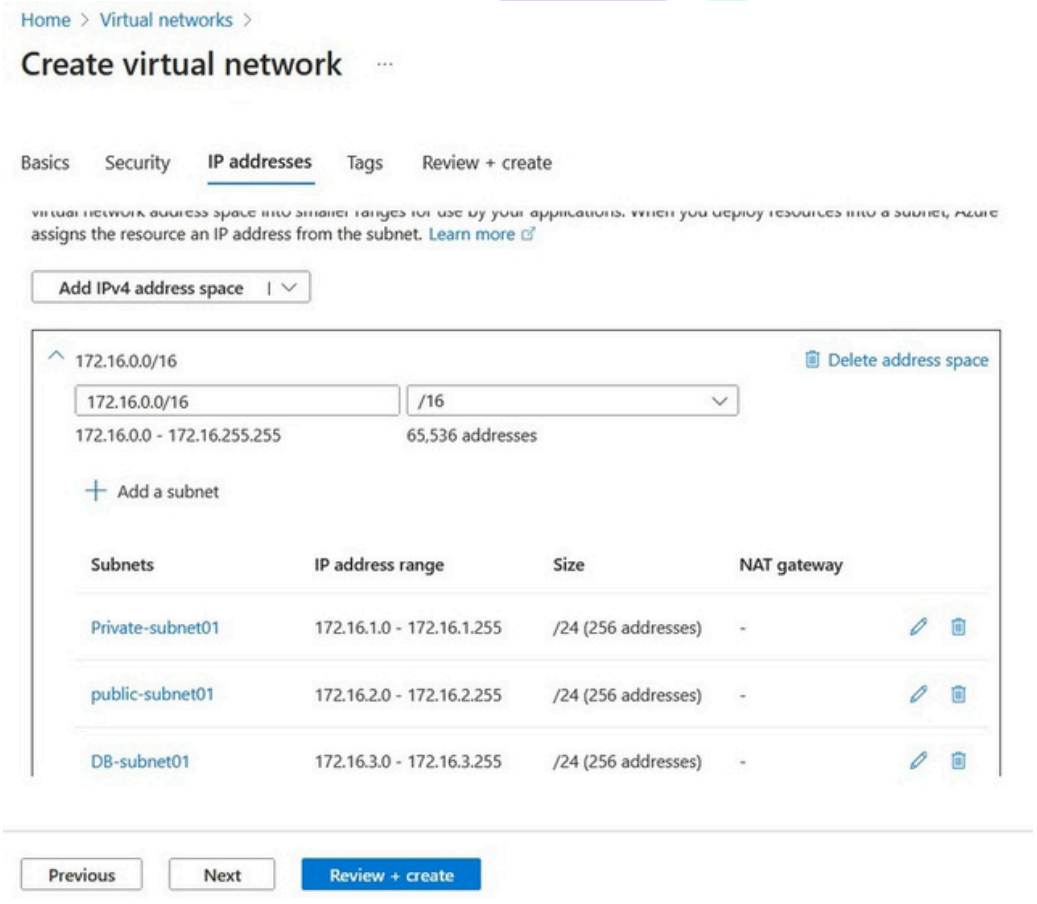
- Step 1: Go to Azure Portal → Search for Virtual Network → Click Create
- Step 2:
  - o VNet Name: VIITOR-Global-VNet
  - o Address Space: 172.16.0.0/16
  - o Region: East US

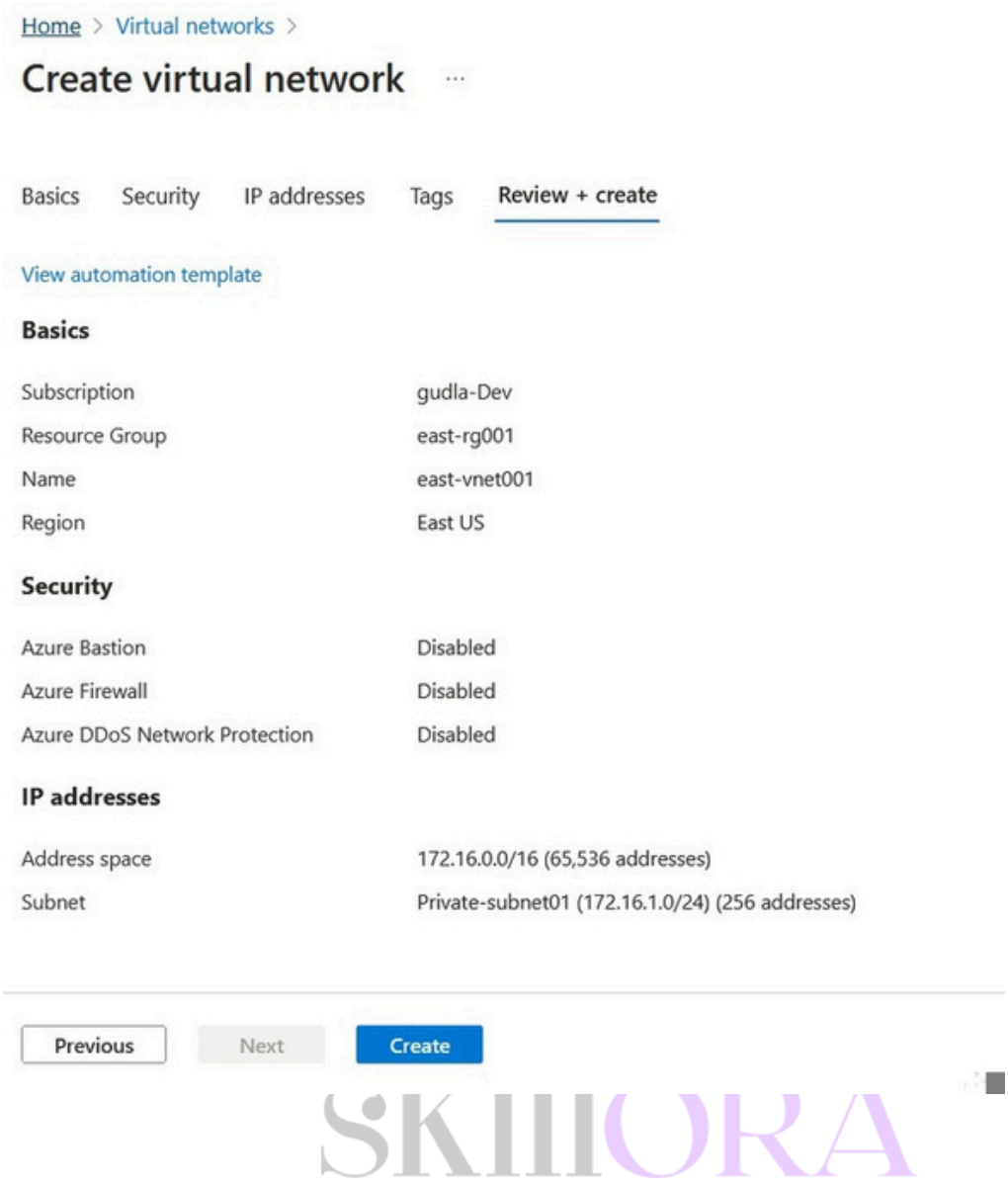
- Step 3: Add Subnets:
  - o Web Subnet: 172.16.1.0/24
  - o Database Subnet: 172.16.2.0/24
  - o ApplicationGateway Subnet: 172.16.3.0/24
  - o managment Subnet: 172.16.4.0/24

Step 4: Click Review + Create




Home > Virtual networks >
Create virtual network
Basics
Security
IP addresses
Tags
Review + create
Project details
Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.
Subscription *
gudla-Dev
Resource group *
east-rg001
Create new
Instance details
Virtual network name *
east-vnet001
Region *
(US) East US
Deploy to an Azure Extended Zone
Previous
Next
Review + create


Home > Virtual networks >
Create virtual network
Basics
Security
IP addresses
Tags
Review + create
assigns the resource an IP address from the subnet. Learn more
Add IPv4 address space
172.16.0.0/16
Delete address space
172.16.0.0/16
/16
172.16.0.0 - 172.16.255.255
65,536 addresses
Add a subnet
Subnets
IP address range
Size
NAT gateway
Private-subnet01
172.16.1.0 - 172.16.1.255
/24 (256 addresses)
-
public-subnet01
172.16.2.0 - 172.16.2.255
/24 (256 addresses)
-
DB-subnet01
172.16.3.0 - 172.16.3.255
/24 (256 addresses)
-
Previous
Next
Review + create



LAB: Create a Virtual Network with Class C and 4 Subnets on westus region

- Create a resource Group
- Create a vnet CIDR 192.168.0.0/16
- Private-Subnet CIDR 192.168.1.0/24, Public-Subnet CIDR 192.168.2.0/24 DB-subnet CIDR 192.168.3.0, AzureBastion 192.168.4.0/24

vNet peering:

Peering will establish the communication between two or more virtual network. Peering provides low-latency, high-bandwidth communication across Azure resources, without the need for a gateway or public internet access.

VNet Peering అనేద **Azure లో రండు Virtual Networks (VNet) లను ప్రైవేట్ కనెక్షన దావరా కలిపే రదతిధ**. ఇద **దావరా శ్రాఫిక ఇంటరిట్ లేదా VPN గేట్‌వే లేకుండా** నేరుగా ఒక VNet నుండి మరో VNet కి వెళ్లడతంద.

- ✅ **ఆలాగే రండు లేదా ఎకుకవ VNets కలిపి రని చేయాల్నపుడు**
- ✅ **భినిమైన Azure శ్రరంతాలోల ఉని VNet లను కనెక ంచేయడానికి VNet మధే**
- ✅ **కమ్యేనికేషన వేగంగా & సుర్థితంగా ఉండాలి అనుకునిపుడు నెట్‌వర్క**
- ✅ **లెటెనీ తగ్గంగచాలి అనుకునిపుడు**

Key features:

- Private communication
- Intra region and Global Region peering
- High Performance
- Full Mesh connectivity
- No Downtime, while enabling the peering
- vNet peering is not Transitive
    - ex: If **VNet A** is peered with **VNet B**, and **VNet B** is peered with **VNet C**, resources in **VNet A** cannot communicate directly with resources in **VNet C** unless additional peering connections are established.

Real-World Use Case: VIITOR Technologies' Global Infrastructure

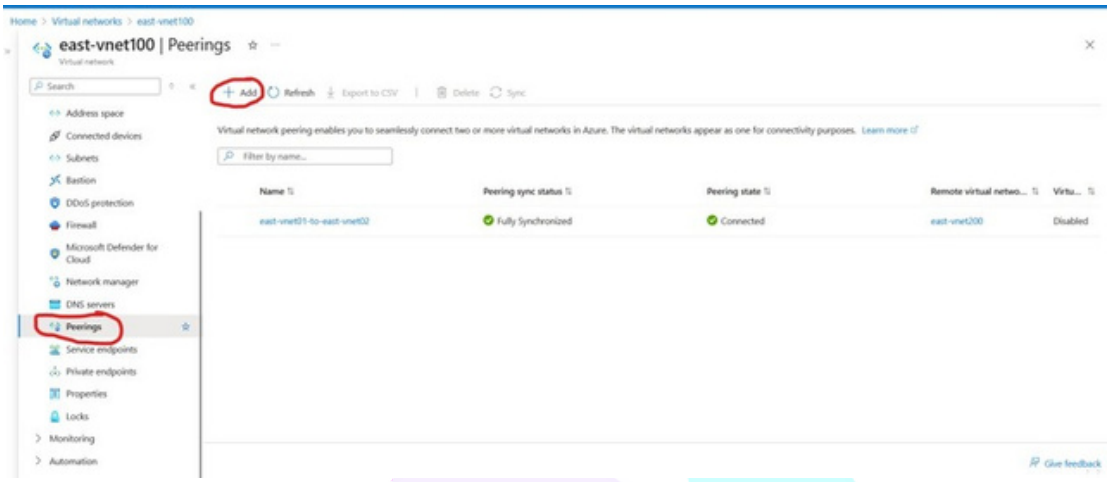
VIITOR Technologies has its development and production environments in separate VNets, both located in different **regions**.

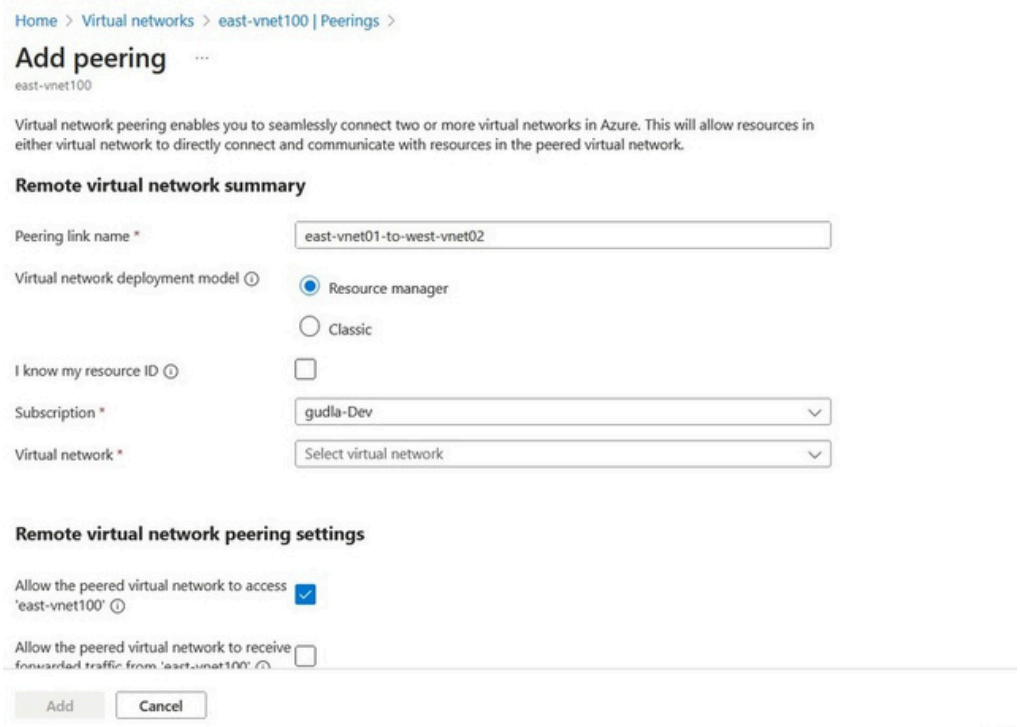
- The development team needs to access production data, so a **VNet Peering connection** is established.

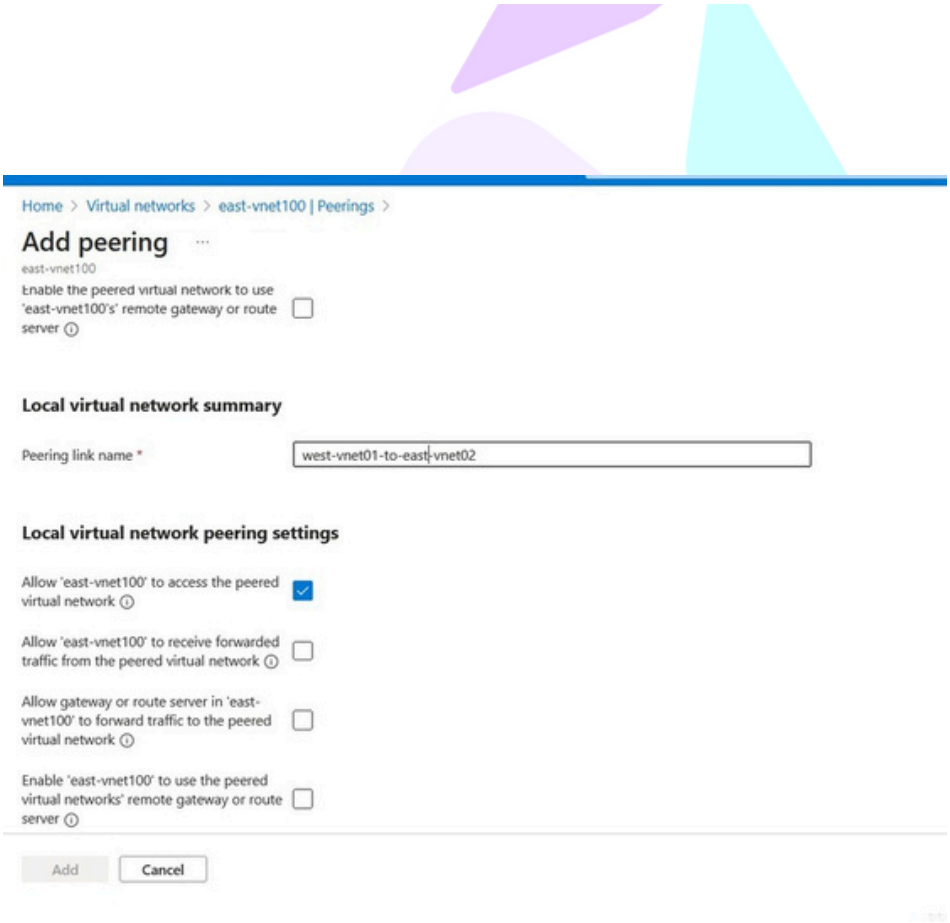
- Thedata in production should remain secure, and only necessary resources can access it.

LAB7: TestvNet peering

- Create Multiple Virtual Networks (ex: east-vnet01 and west-vnet01) with respective subnets.
- Create a Peering (If we create peering source vNet.., automatically peering will create on Second vNet)




Home > Virtual networks > east-vnet100 | Peerings >
Add peering
east-vnet100
Virtual network peering enables you to seamlessly connect two or more virtual networks in Azure. This will allow resources in either virtual network to directly connect and communicate with resources in the peered virtual network.
Remote virtual network summary
Peering link name *
east-vnet01-to-west-vnet02
Virtual network deployment model
Resource manager
Classic
I know my resource ID
Subscription *
gudla-Dev
Virtual network *
Select virtual network
Remote virtual network peering settings
Allow the peered virtual network to access 'east-vnet100'
Allow the peered virtual network to receive
Add
Cancel


Home > Virtual networks > east-vnet100 | Peerings >
Add peering
east-vnet100
Enable the peered virtual network to use 'east-vnet100's' remote gateway or route server
Local virtual network summary
Peering link name *
west-vnet01-to-east-vnet02
Local virtual network peering settings
Allow 'east-vnet100' to access the peered virtual network
Allow 'east-vnet100' to receive forwarded traffic from the peered virtual network
Allow gateway or route server in 'east-vnet100' to forward traffic to the peered virtual network
Enable 'east-vnet100' to use the peered virtual networks' remote gateway or route server
Add
Cancel

Azure Virtual Network Gateway:

An **Azure Virtual Network Gateway** is a critical component for enabling secure connectivity between on-premises networks and Azure resources. It acts as a bridge between different networks, allowing organizations to extend their on-premises data centers securely into the cloud.

Azure Virtual Network Gateway అనేద భద్దతాపర్మైన కనెకొవాటీని అందంచడానికి ఉపయోగపడే ముఖ్ంామైన భాగం. ఇద ఆన్స-తీమైసస్ నెటు్లవంార్క్లను అజ్యార్క్లతో అనుస్ంధానించడానికి ఉపయోగస్టతరు.

- **Secure Communication:** Establishes encrypted VPN tunnels between on-premises networks and Azure.
- **Multiple Connection Modes:** Supports Site-to-Site (S2S), Point-to-Site (P2S), and ExpressRoute connections.
- **Scalability:** Can handle high volumes of traffic based on gateway SKU selection.
- **Redundancy:** Can be deployed in an active-active mode for high availability.
- **Integration with NSGs and Firewalls:** Ensures traffic is controlled and monitored.

Real-World Use Case: VIITOR Technologies' Secure Hybrid Cloud

VIITOR Technologies has an **on-premises data centre** and a **cloud presence** in Azure.

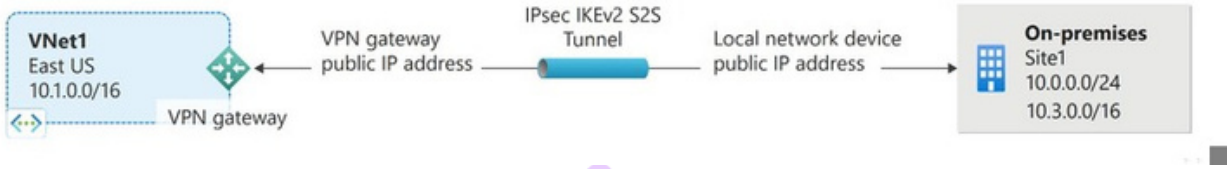
- Employees in the **on-premises office** need to access **cloud resources** securely.
- A **VPN Gateway** is configured to establish a **secure tunnel** between on-premises and Azure VNet.

Types of Azure Virtual Network Gateways:

1. **VPN Gateway**: Establishes a secure, encrypted connection over the internet between an on-premises network and an Azure VNET.
    - Supports **Site-to-Site (S2S) VPN** for corporate networks.
    - Supports **Point-to-Site (P2S) VPN** for individual remote users.

- Uses **IPsec/IKE protocols** for encryption.

2. **ExpressRoute Gateway**: Provides a **private connection** between on-premises networks and Azure using a dedicated circuit.
    - Bypasses the public internet, ensuring **higher security, reliability, and lower latency**.
    - Suitable for large enterprises requiring **consistent network performance**



Real-World Scenario: Hybrid Cloud Connectivity with Azure Virtual Network Gateway

A multinational company wants to connect its on-premises data center securely to Azure while enabling remote employees to access resources. Here's how they design their network:

- **Site-to-Site VPN:** Securely connects the corporate office network to Azure.
- **Point-to-Site VPN:** Allows remote employees to work securely from home.
- **ExpressRoute:** Ensures dedicated, high-performance connectivity for critical workloads.
- **Network Security Groups (NSG) and Firewalls:** Restrict unauthorized access.

Steps to Create VPN Gateway:

1. **Go to Azure Portal → Search for VPN Gateway**
2. **Create a VPN Gateway** in the **VNet** where it will reside
3. Configure the **VPN Gateway Settings**:
    - Choose **Public IP Address**
    - Select **VPN type** (Route-based or Policy-based)
4. After the VPN Gat
5. eway is created, configure the **VPN Device** on your on-premises side.

Network Security Group:

Azure Network Security Group will help to control Inbound and outbound traffic to Azure resources. It acts like a Firewall for Azure resources.

Network Security Group (NSG) అనేద **Azure లో శ్రాఫిక ని నియంశ్రతించే Firewall లాంటిది.** ఇద **ఇనక్మింగ్స (Inbound) & అవుట్గోయింగ్స (Outbound) శ్రాఫిక** ని **రూల్ట్ ఆధార్ంగా అనుమతించడానికి లేదా నిరోధించడానికి (Allow/Deny)** ఉపయోగస్టతరు.

సింప్పల్ గా చెపాలంటే, NSG ద్వారా Azure లో మీ నెట్క్లవార్ & అపికేడషనడభత్తను మెరుగుపర్వచు.

Key Features:

- Inbound and Outbound Rules**:**
- Security Rules**:**
- Associating NSGs with Subnets and NICs**:**
- service Tags and Application Security Groups (ASGs)**:**
- Default Security Rules

● NSGs can be integrated with **Azure Network Watcher** for diagnostic capabilities

Real-World Use Case:

VIITOR Technologies has a **public web application** and needs to **secure its traffic**.

NSGs are applied to the **web server subnet** to only allow **HTTP and HTTPS traffic**.

All other traffic, such as **SSH and RDP**, is **blocked** for enhanced security.

Access control lists	Rules that allow or deny access based on priority, direction, source and destination, and port and protocol.
Default rulesets	Default rules that cannot be changed but can be overridden.
Priority	Using higher priority rules, you can override the default rules, spacing priority numbers you assign to allow future flexibility.
Naming strategy	Assigning a name that defines the NSG's purpose is essential for longer-term management.
Service tags	A group of IP address prefixes managed by Microsoft to allow or deny access to a given Azure service.
Application security groups (ASGs)	Group network interface cards into applications or services for easier management.
Topic	Description
Network watcher	Provides tools that can be used for monitoring, alerting, and troubleshooting.

NSG rules:

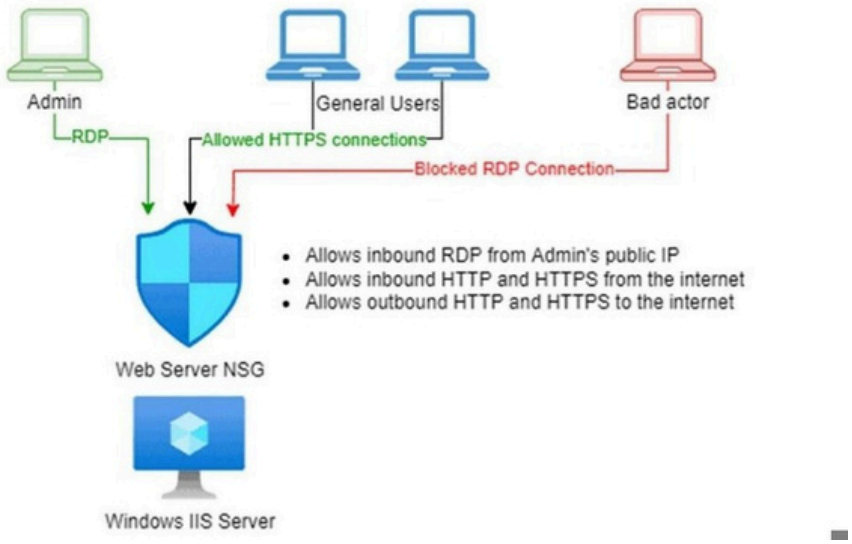
An **NSG rule** in Azure defines whether traffic to and from Azure resources is allowed or denied based on specified criteria such as source/destination IP, port, and protocol. Each rule is applied to inbound or outbound traffic and is evaluated based on a priority number. Rules with lower priority numbers are processed first.

Key Elements of an NSG Rule:

- Priority
- Name
- Source
- Source Port Range
- Destination
- Destination Port Range
- Protocol
- Direction
- Action

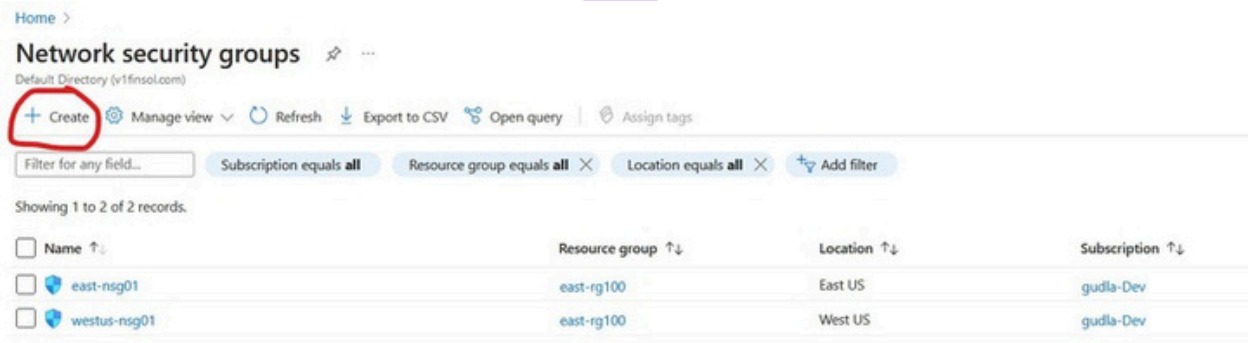
Below is the scenario for better understanding

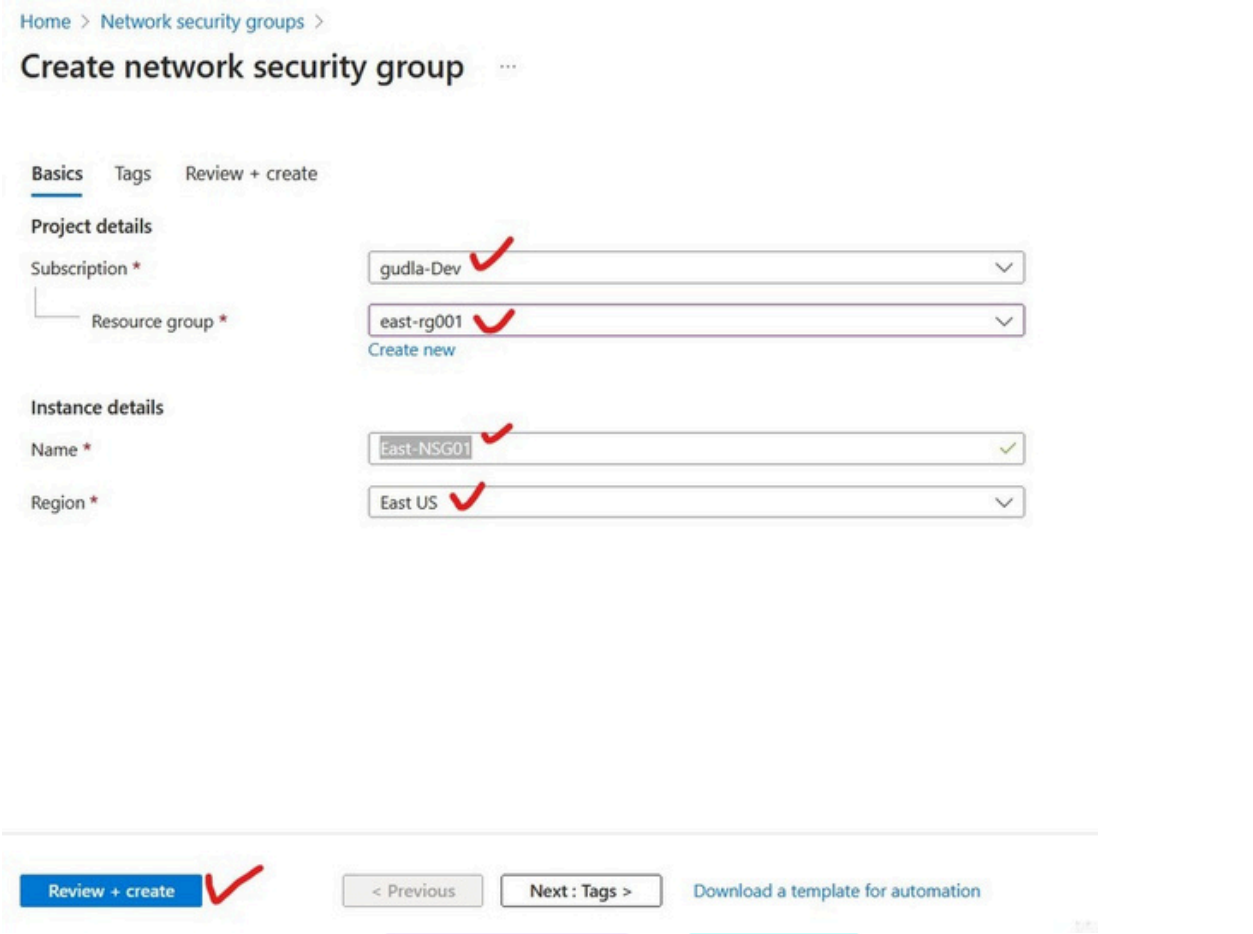
The below image shows a basic NSG filtering incoming traffic to a Windows IIS server



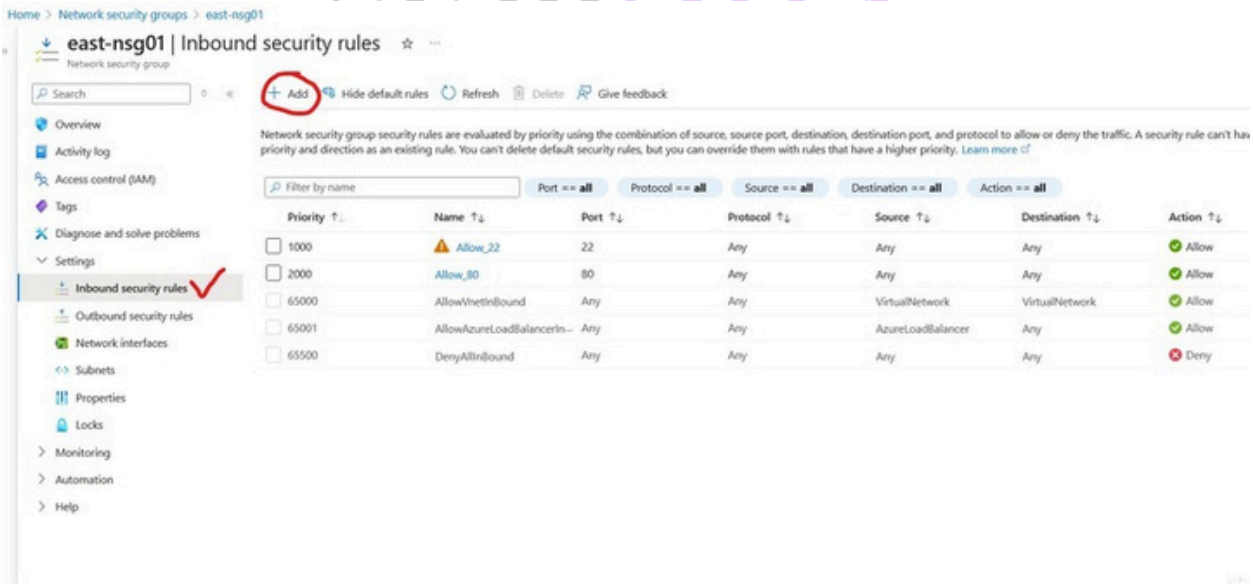
LAB: Create a NSG and Assign to Subnet or NIC card

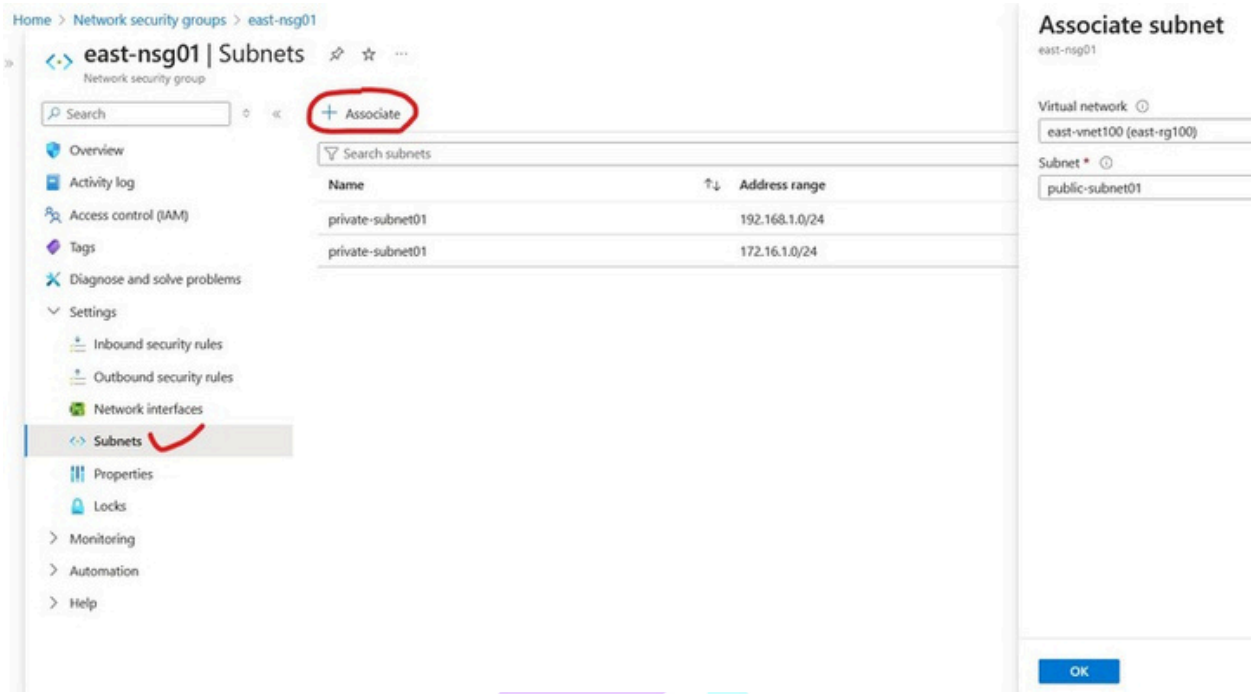
1. **Go to Azure Portal → Search for Network Security Groups** → Click **Create**
2. Define **NSG Name** and **Region**
3. Create **Inbound/Outbound Rules**:
    - **Allow HTTP (Port 80)**
    - **Allow HTTPS (Port 443)**
    - **Deny all other traffic**
4. Associate the **NSG** with a **Subnet or NIC**

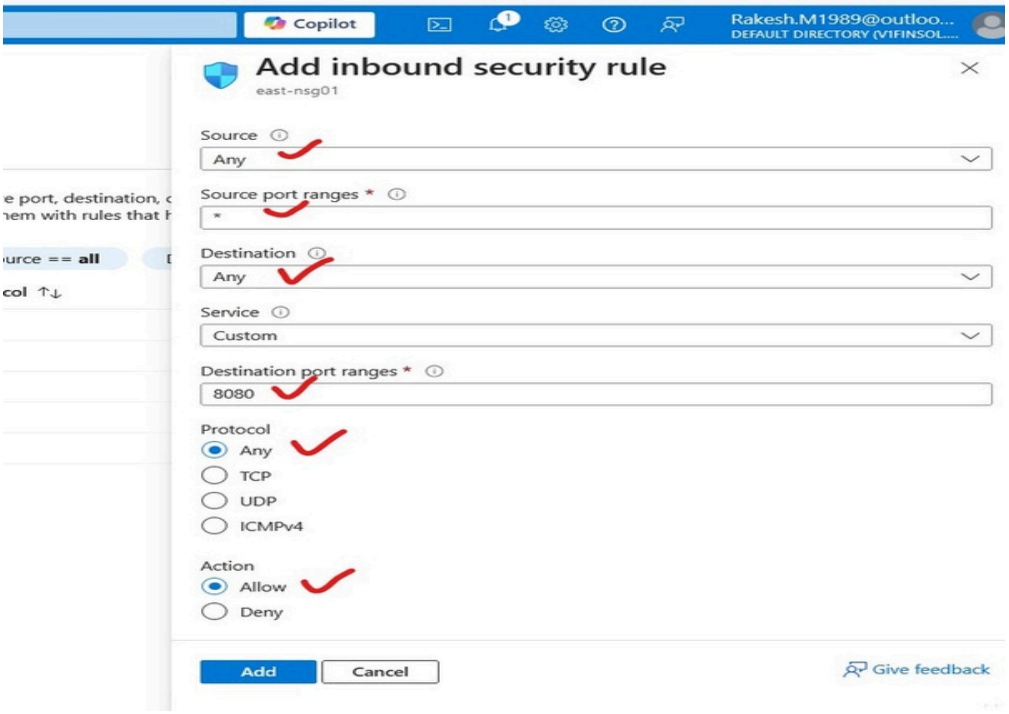


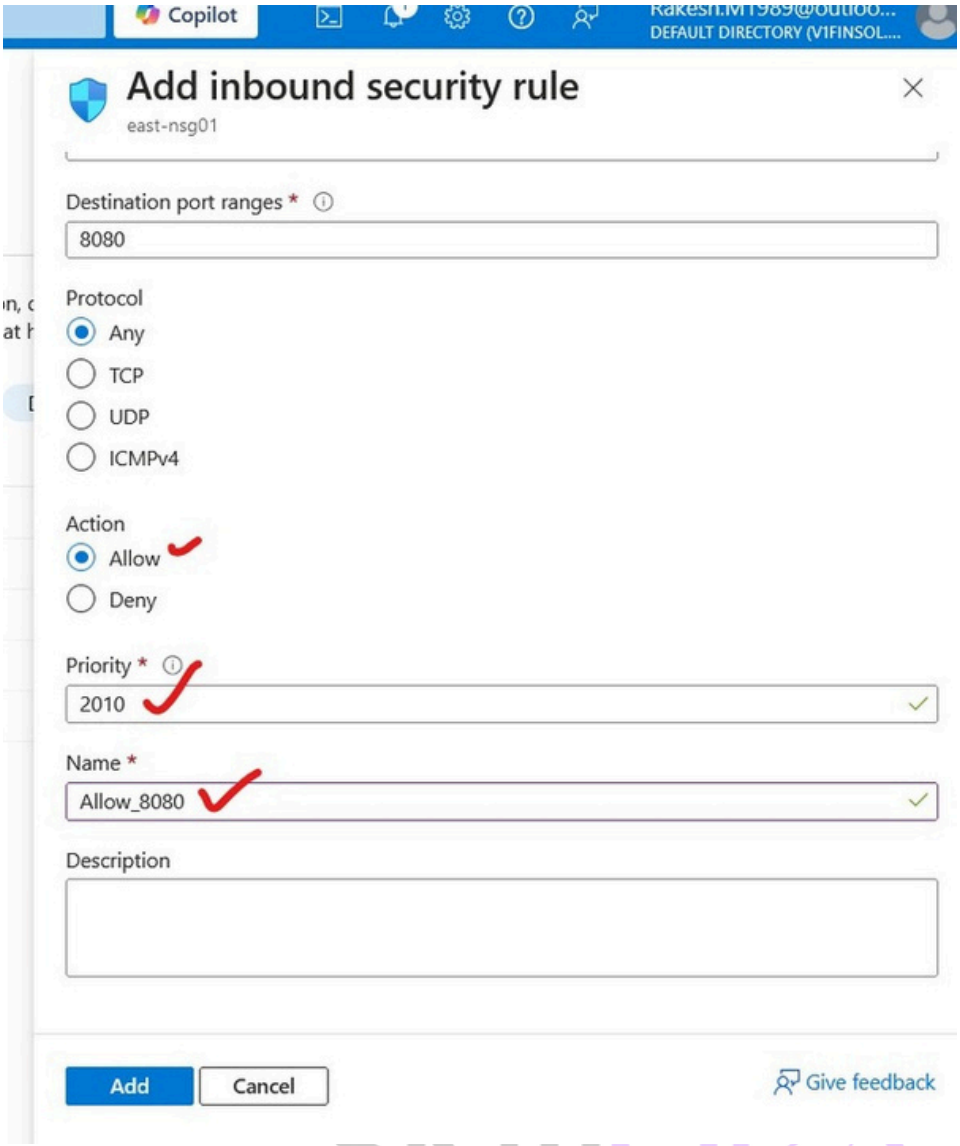


LAB9: Create NSG rule




Home > Network security groups > east-nsg01
east-nsg01 | Subnets
Network security group
Search
Associate
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Settings
Inbound security rules
Outbound security rules
Network interfaces
Subnets
Properties
Locks
Monitoring
Automation
Help
Search subnets
Name
Address range
private-subnet01
192.168.1.0/24
private-subnet01
172.16.1.0/24
Associate subnet
east-nsg01
Virtual network
east-vnet100 (east-rg100)
Subnet *
public-subnet01
OK


Copilot
Rakesh.M1989@outloo...
DEFAULT DIRECTORY (VIFINSOL....
Add inbound security rule
east-nsg01
Source
Any
Source port ranges *
*
Destination
Any
Service
Custom
Destination port ranges *
8080
Protocol
Any
TCP
UDP
ICMPv4
Action
Allow
Deny
Add
Cancel
Give feedback


Copilot
DEFAULT DIRECTORY (V1FINSOL....
Add inbound security rule
east-nsg01
Destination port ranges *
8080
Protocol
Any
TCP
UDP
ICMPv4
Action
Allow
Deny
Priority *
2010
Name *
Allow_8080
Description
Add
Cancel
Give feedback

LAB: **Allow Web Traffic (HTTP and HTTPS)**, **Allow vNet to vNet, Deny All Inbound Traffic Except VNet**:

Priority: 100

Name: Allow_HTTP_HTTPS

Source: Internet

Source Port: *

Destination: * (or the specific IP of the web server)

Destination Port: 80,443

Protocol: TCP

Action: Allow

Direction: Inbound

Deny All Inbound Traffic Except VNet:

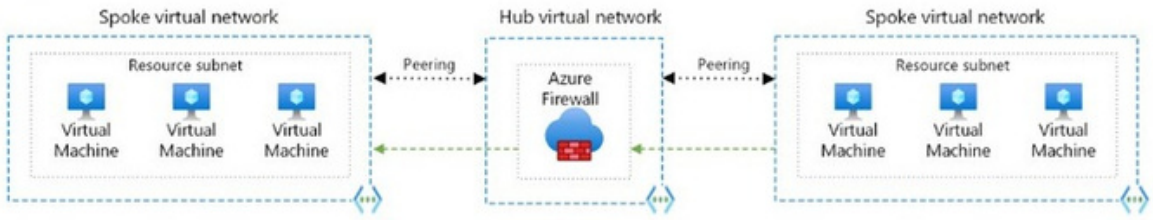
Priority: 200 Name: Allow_VNet Source: VirtualNetwork Source Port: * Destination: * Destination Port: * Protocol: Any Action: Allow Direction: Inbound

Priority: 300 Name: Deny_All Source: * Source Port: * Destination: * Destination Port: * Protocol: Any Action: Deny Direction: Inbound

HUB & SPOKE Architecture

The **Hub and Spoke architecture**is a commonnetworktopologyused in **Azure** to efficiently manage and secure resources across multiple virtual networks (VNets). In the Hub and Spoke architecture, a central **Hub VNet** acts as a common point of connectivity for multiple **Spoke Vnets.**

హబ మరియు స్పుక టాపలక్లజీ అనేద ఒక **నెట్‌వర్కక డిజైన** ఇందులో ఒక **కేంశ్రద (Hub)** నెట్‌్వర్కక అని స్పుక (Spoke) నెట్‌్వర్కక్‌తో కనెకా ాఅవుతంద. ఇద స్టధార్ణంగా **ఇంటర్కప్రైజ్-స్యయ్‌లౌడ్ నెట్‌వర్కక్** కోస్ం ఉపయోగంచబడుతంద.



Hub virtual network: The hub virtual network hosts shared Azure services

Spoke virtual networks: Spoke virtual networks isolate and manage workloads separately in each spoke (Each spoke we can consider an ISOLATED Application)

Key Benefits of Hub and Spoke:

- Centralized Security
- Cost Efficiency
- Network Isolation
- Scalability
- Simplified Management
- Transitive Routing

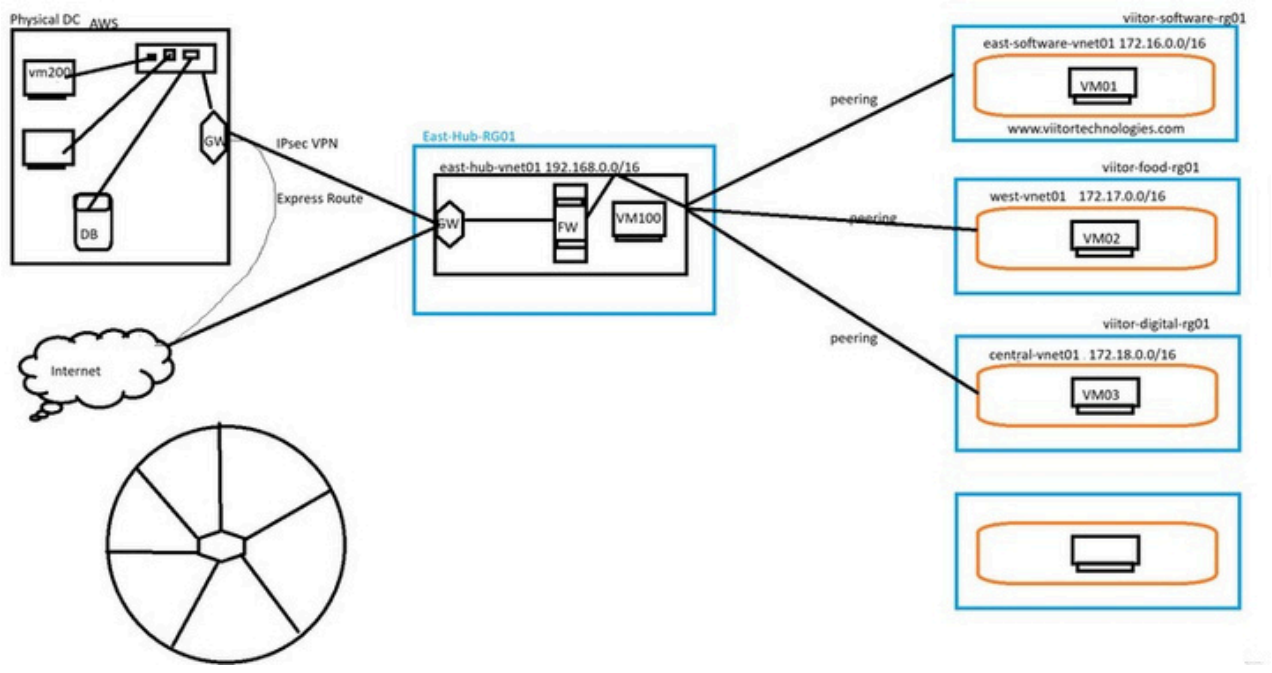
Considerations while implementing the Hub and Spoke:

- **Performance** : We routing all the traffic from Hub, if we not planned enough resources to handle the traffic it can be bottleneck.

- **Cost:** Depends on the size of our network, additional resource cost will be added like Firewall, Gateways.,etc
- **Security**: We have to be ensure

Key components:

- **Virtual Network Gateway:** Enables connectivity to onprem network through VPN or Express route.
- **Azure Firewall or 3rd Party Firewall:** The Hub location should have Firewall to control traffic TO and FROM spoke network
- **Peering:** vNet peering is established between the Hub and each Spoke to allow communication
- **UDR(User Define Routs):** UDRs are custom routes you create in Azure to control network traffic flow
    - o Routing traffic between spokes across different hubs
    - o Routing all traffic from spoke virtual networks to an Azure Firewall



Real-World Use Case: VIITOR Technologies' Global Cloud Infrastructure

Scenario:

VIITOR Technologies has a multi-region Azure deployment. They have:

- **A Hub Network** where common resources (VPN Gateway, Bastion Host, NTP server) are hosted.
- **Multiple Spokes** for different departments:
  - **Spoke 1**: Development Network with resources like **development VMs** and **dev databases**.
  - **Spoke 2**: Production Network with resources like **production VMs** and **production databases**.
  - **Spoke 3**: Analytics Network with resources for **big data processing** and **data warehouses**.

Each spoke communicates with the Hub, but **spokes do not directly communicate with each other**, ensuring isolation and security.

How Does Hub and Spoke Work in Azure?

Key Components:

1. **Virtual Networks (VNets)**:
   - **Hub VNet**: This is the central network in the topology.
   - **Spoke VNets**: These are connected to the Hub VNet but isolated from each other.
2. **VNet Peering**:
   - **Peering** is set up between the **Hub VNet** and each **Spoke VNet**.
   - **VNet Peering** allows communication between the Hub and Spokes.
3. **Network Security Groups (NSGs)**:
   - **NSGs** are used to control the traffic between the Hub and Spokes, ensuring only authorized communication is allowed.
4. **Shared Services**:
   - Common services like **VPN Gateway**, **Azure Firewall**, or **Bastion** are usually deployed in the Hub.
5. **Route Tables**:

- **Route tables** ensure that traffic from the Spokes goes through the Hub for any inter-network communication.

Azure Compute Services

Azure Compute services provide scalable, on-demand computing power for applications, workloads, and services. It includes Virtual Machines, Containers, Serverless computing, and App Services

Key Azure Compute Services:

- **Virtual Machines (VMs)** – Full control over OS and applications.
- **Azure App Services** – Managed web hosting platform.
- **Azure Functions**
- **Azure Kubernetes Service (AKS)** – Scalable container management.
- **Azure Functions** – Event-driven, serverless computing.

Azure virtual Machine:

Azure Virtual Machines (VMs) are one of the primary services in Azure's Infrastructure-as-a-Service (IaaS) offering. They provide scalable, on-demand compute resources in the cloud, allowing you to run applications, host services, and perform tasks without needing to manage physical hardware. Azure VMs offer flexibility, enabling you to select the operating system, configuration, and scale to meet your business needs.

Azure Virtual Machines (VMs) allow users to run applications and workloads in a customizable cloud environment. A VM is a virtualized server that provides the same capabilities as a physical server but with greater flexibility and scalability.

Key Features of Azure Virtual Machines

- **Scalability:** Easily scale up or down based on demand.
- **Customization:** Choose OS, CPU, RAM, and storage options.
- **Availability:** Deploy across multiple regions for redundancy.
- **Security:** Use Network Security Groups (NSGs) and firewalls for protection.
- **Integration:** Connect with Azure Storage, Networking, and Monitoring services.

Types of Azure Virtual Machines

1. **General Purpose VMs:** Balanced CPU-to-memory ratio, ideal for most workloads.
2. **Compute Optimized VMs:** High CPU power for processing-intensive applications.
3. **Memory Optimized VMs:** More RAM for large databases and in-memory caching.
4. **Storage Optimized VMs:** High disk throughput for big data workloads.
5. **GPU VMs:** Suitable for AI, ML, and graphics-intensive tasks.

This table now provides a **comprehensive overview** of commonly used Azure VM series.

VM Series	Purpose	Use Cases
B-Series (Burstable VMs)	Cost-effective VMs with low to moderate CPU usage, capable of bursting when needed.	Small web servers, dev/test environments, lightweight applications, low-traffic databases.
D-Series (General-Purpose VMs)	Balanced CPU, memory, and disk performance for general workloads.	Enterprise applications, relational databases, web applications, medium-sized workloads.
E-Series (Memory-Optimized VMs)	High-memory VMs designed for workloads requiring large memory capacity.	In-memory databases, high-performance analytics, caching services, SAP applications.
F-Series (Compute-Optimized VMs)	High CPU-to-memory ratio, optimized for compute-intensive workloads.	Gaming servers, batch processing, AI model training, data analysis.
M-Series (High-Memory VMs)	Extremely high-memory VMs for large-scale databases and big data analytics.	SAP HANA, large-scale enterprise applications, data warehousing.
L-Series (Storage-Optimized VMs)	Optimized for workloads requiring low latency and high disk throughput.	NoSQL databases, big data processing, high-speed transaction processing.
N-Series (GPU-Optimized VMs)	Designed for graphics and AI/ML compute-intensive workloads with GPU acceleration.	AI/ML workloads, video rendering, gaming, deep learning model training.

H-Series (High-Performance Computing VMs)	Designed for computationally intensive workloads requiring high processing power.	Scientific simulations, financial risk modeling, weather simulations, deep learning.

Real-World Scenario: Hosting an E-Commerce Website on Azure VMs

Imagine a growing e-commerce startup that wants to host its online store with high availability and performance. By deploying multiple Azure VMs behind a load balancer, the website ensures:

- **Scalability** during high traffic seasons.
- **High Availability** with VMs in different Azure regions.
- **Security** with firewalls and Network Security Groups.
- **Cost Optimization** by auto-scaling during low demand periods.

How to Connect Azure VM (Azure VM ను ఎలా కనెక ిఅవావలి)?

- **Windows VM – RDP (Remote Desktop - Port 3389)** ద్వారా కనెకా ిఅవాాలి.
- **Linux VM – SSH (Secure Shell - Port 22)** ద్వారా కనెకా ిఅవాాలి.

గమనిక: RDP & SSH యాక్స్స్ కోసం NSG లో అనుమతించాలి

Azure Virtual Machine helps you to deploy your applications without worrying about the underlying infrastructure.

LAB: Create a virtual Machine

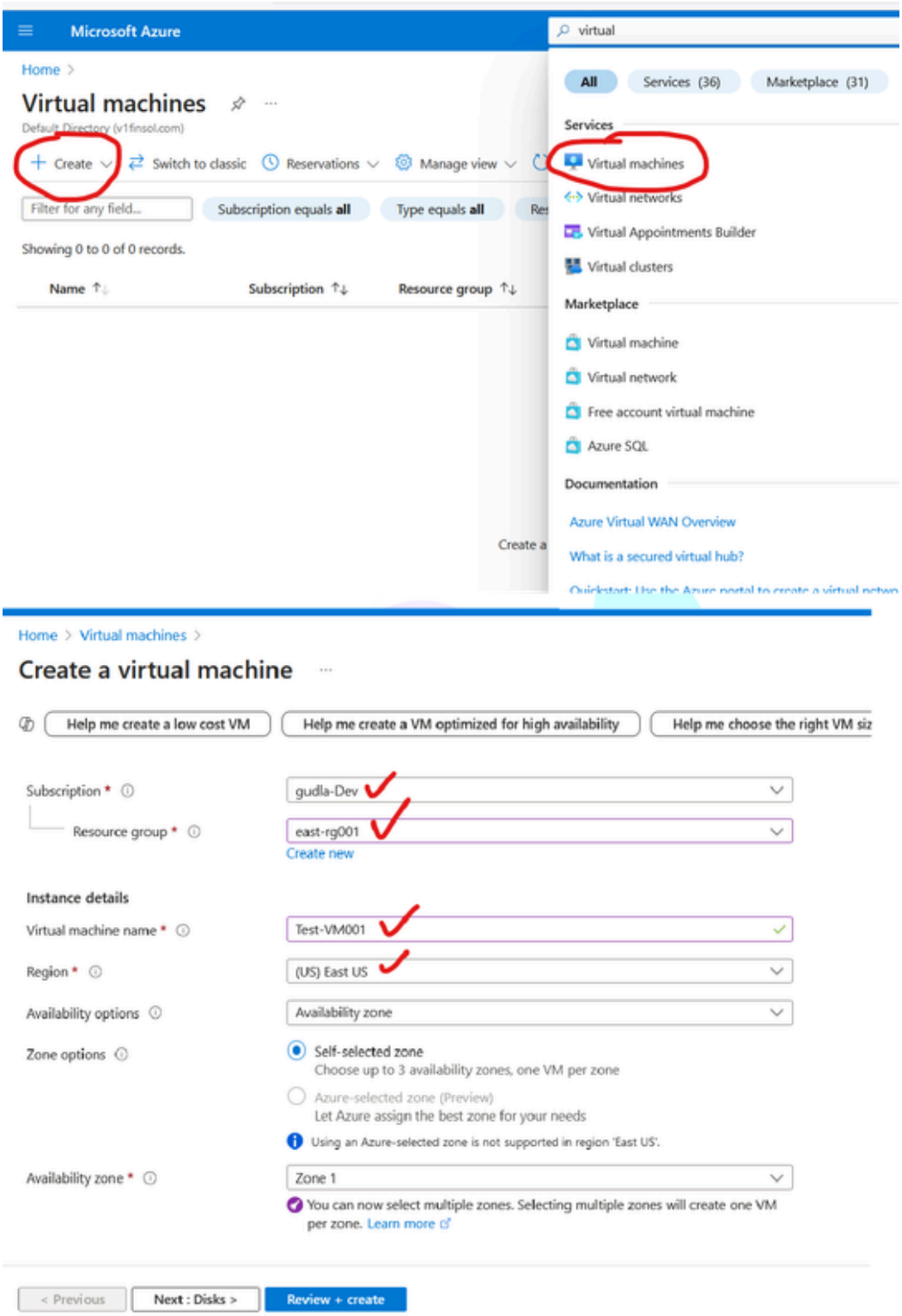
Prerequisite :

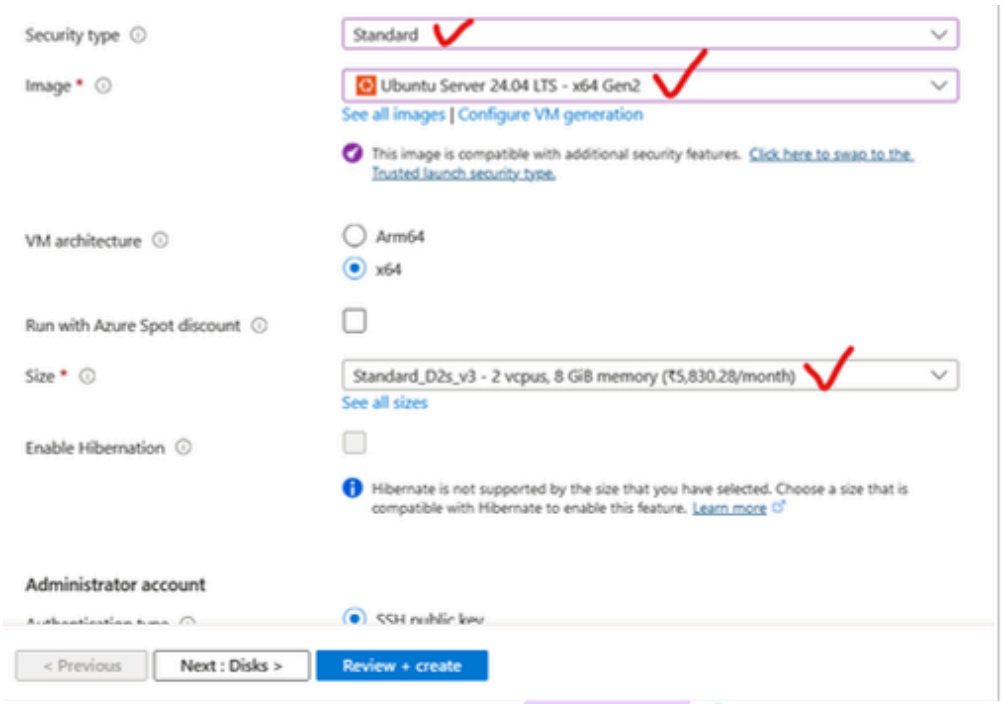
- Resource Group
- vNet and Subnet creation

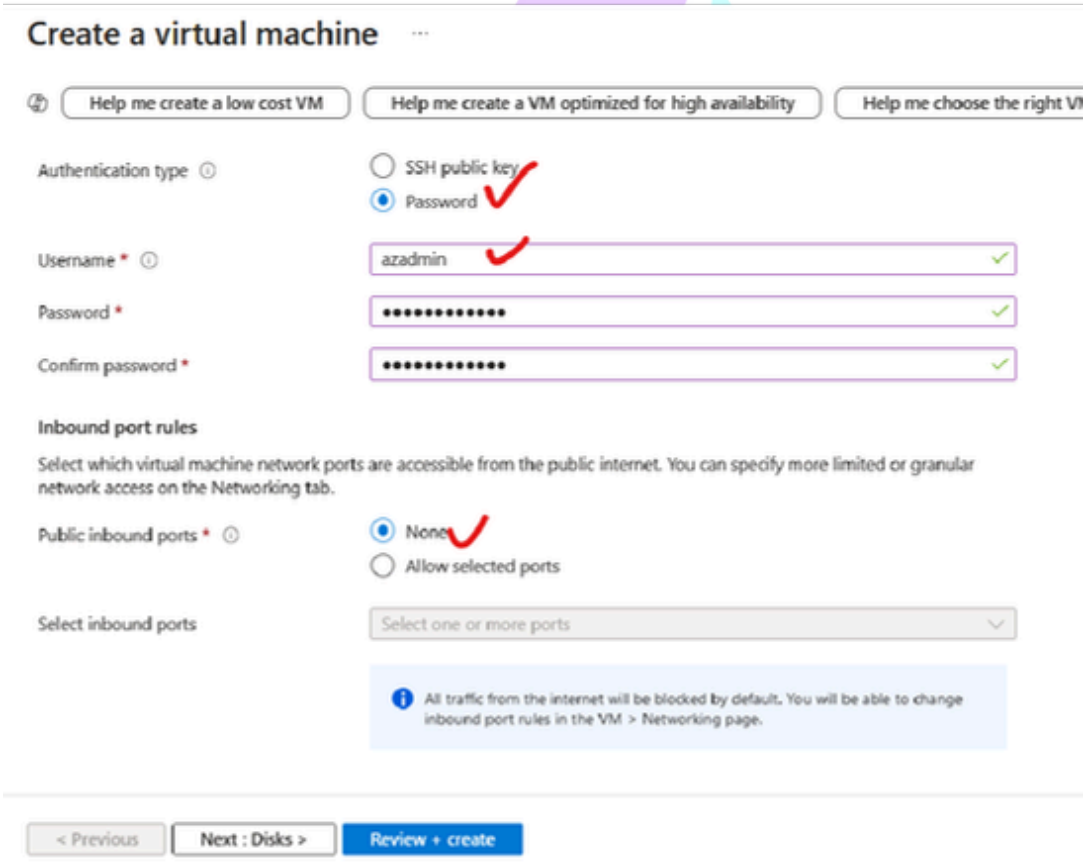
Steps:

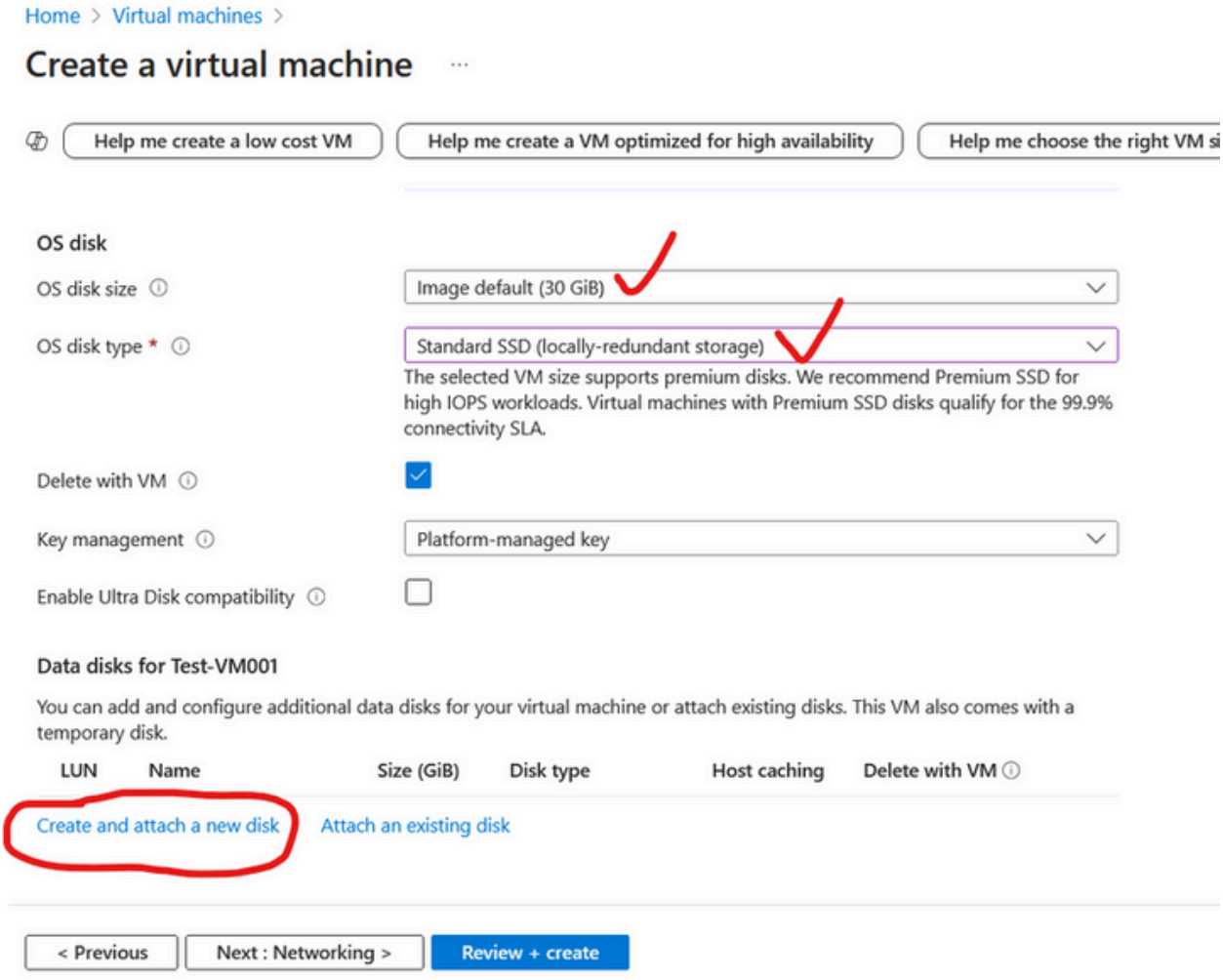
- Go to Azure Portal and click on Create a Resource.
- Select Virtual Machine from the marketplace.
- Choose the operating system (Windows or Linux).
- Provide basic details like VM name, Region, and Size.

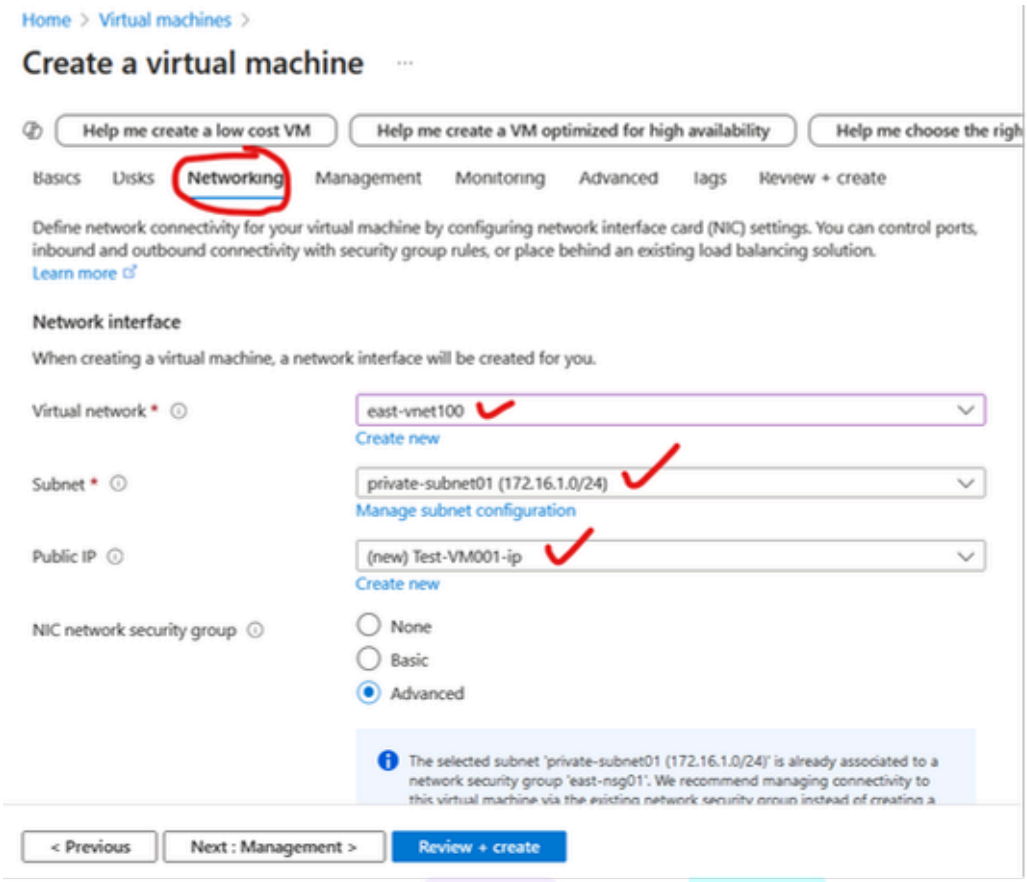
- Configure Networking settings.
- Set up admin credentials (username and password).
- Review and create the VM.

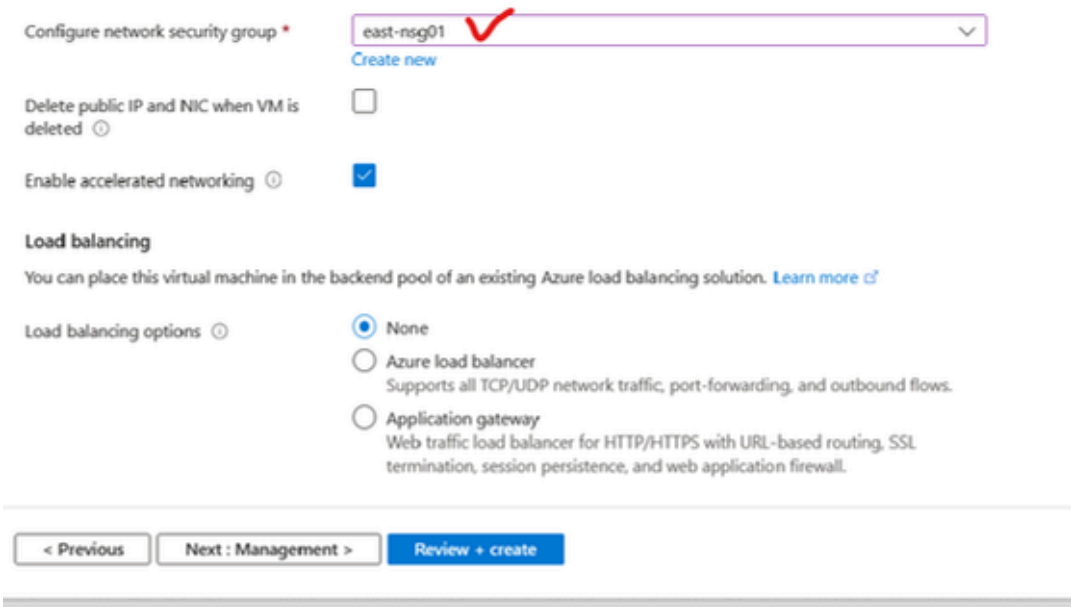


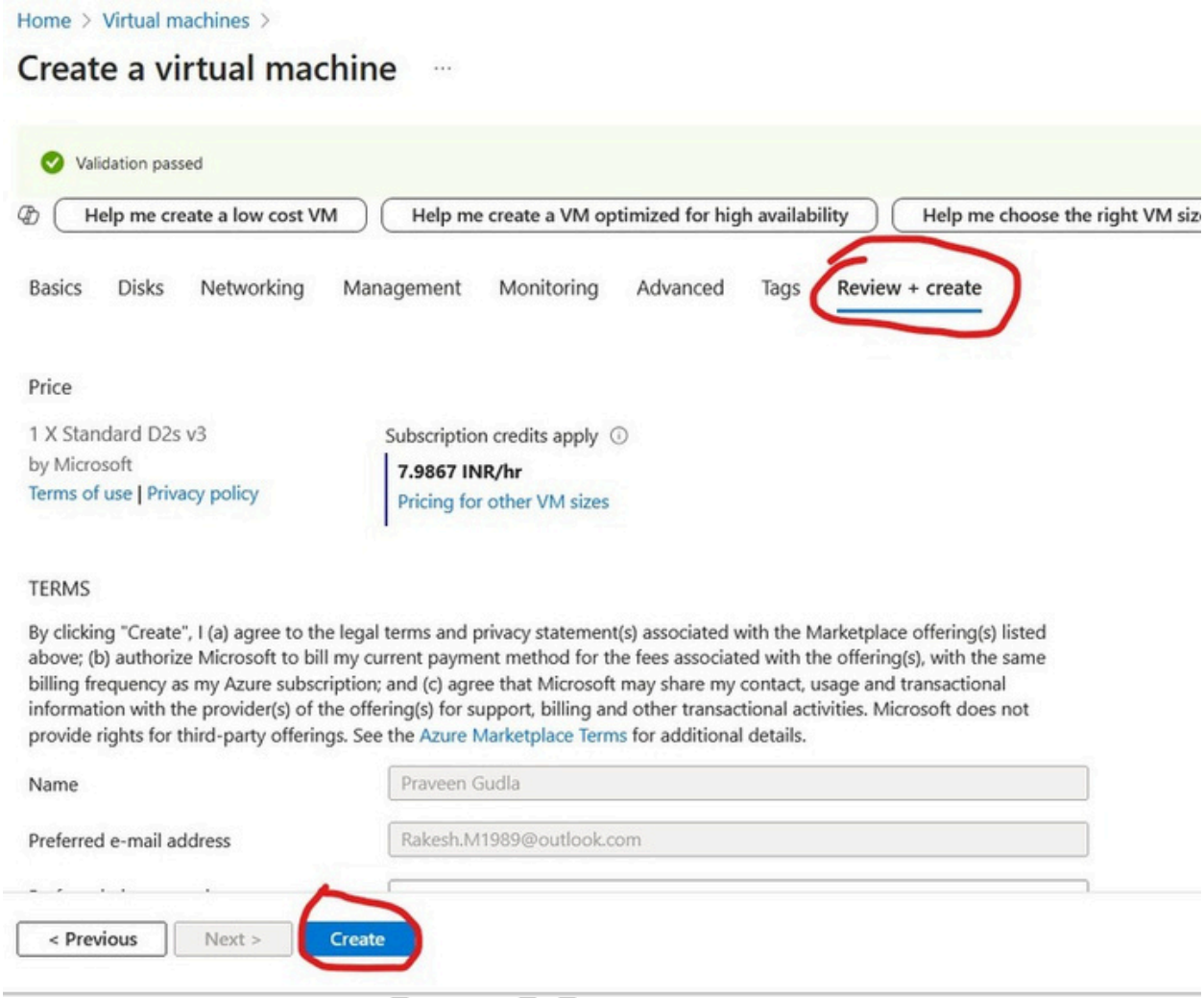

Security type
Standard
Image *
Ubuntu Server 24.04 LTS - x64 Gen2
See all images | Configure VM generation
This image is compatible with additional security features. Click here to swap to the Trusted launch security type.
VM architecture
Arm64
x64
Run with Azure Spot discount
Size *
Standard_D2s_v3 - 2 vcpus, 8 GiB memory (₹5,830.28/month)
See all sizes
Enable Hibernation
Hibernate is not supported by the size that you have selected. Choose a size that is compatible with Hibernate to enable this feature. Learn more
Administrator account
SSH public key
< Previous
Next : Disks >
Review + create


Create a virtual machine
Help me create a low cost VM
Help me create a VM optimized for high availability
Help me choose the right VM
Authentication type
SSH public key
Password
Username *
azadmin
Password *
Confirm password *
Inbound port rules
Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.
Public inbound ports *
None
Allow selected ports
Select inbound ports
Select one or more ports
All traffic from the internet will be blocked by default. You will be able to change inbound port rules in the VM > Networking page.
< Previous
Next : Disks >
Review + create


Home > Virtual machines >
Create a virtual machine
Help me create a low cost VM
Help me create a VM optimized for high availability
Help me choose the right VM si
OS disk
OS disk size
Image default (30 GiB)
OS disk type *
Standard SSD (locally-redundant storage)
The selected VM size supports premium disks. We recommend Premium SSD for high IOPS workloads. Virtual machines with Premium SSD disks qualify for the 99.9% connectivity SLA.
Delete with VM
Key management
Platform-managed key
Enable Ultra Disk compatibility
Data disks for Test-VM001
You can add and configure additional data disks for your virtual machine or attach existing disks. This VM also comes with a temporary disk.
LUN
Name
Size (GiB)
Disk type
Host caching
Delete with VM
Create and attach a new disk
Attach an existing disk
< Previous
Next : Networking >
Review + create


Home > Virtual machines >
Create a virtual machine
Help me create a low cost VM
Help me create a VM optimized for high availability
Help me choose the righ
Basics
Disks
Networking
Management
Monitoring
Advanced
Tags
Review + create
Define network connectivity for your virtual machine by configuring network interface card (NIC) settings. You can control ports, inbound and outbound connectivity with security group rules, or place behind an existing load balancing solution.
Learn more
Network interface
When creating a virtual machine, a network interface will be created for you.
Virtual network *
east-vnet100
Create new
Subnet *
private-subnet01 (172.16.1.0/24)
Manage subnet configuration
Public IP
(new) Test-VM001-ip
Create new
NIC network security group
None
Basic
Advanced
The selected subnet 'private-subnet01 (172.16.1.0/24)' is already associated to a network security group 'east-nsg01'. We recommend managing connectivity to
< Previous
Next : Management >
Review + create


Configure network security group *
east-nsg01
Create new
Delete public IP and NIC when VM is deleted
Enable accelerated networking
Load balancing
You can place this virtual machine in the backend pool of an existing Azure load balancing solution. Learn more
Load balancing options
None
Azure load balancer
Supports all TCP/UDP network traffic, port-forwarding, and outbound flows.
Application gateway
Web traffic load balancer for HTTP/HTTPS with URL-based routing, SSL termination, session persistence, and web application firewall.
< Previous
Next : Management >
Review + create

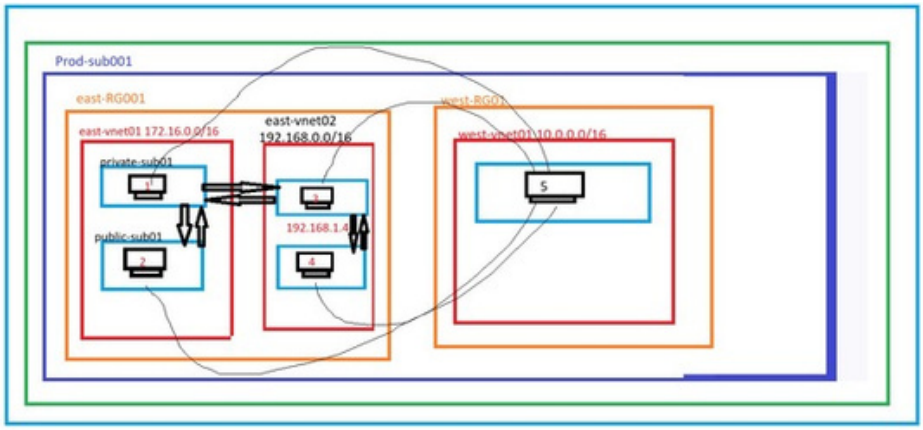


Validation: Try access the Servers from Putty using port SSH.

LAB: Prepare an Environment as per below architecture and test the communication.

Prerequisites:

- Resource Group
- Create multiple Virtual Network and enable Peering
- Create NSGs and add NSG rules (22 and 3389)
- Create VM's and Test the communication using the ping



Eastvnet01 VM01 => West vNet VM03 ?? NO

Eastvnet01 => Peering <= Westvnet01

VM01 can communicate VM04 ?? Yes

Test communication from

east-vnet01 <==> east-vnet02

east-vnet01 <==> west-vent01

east-vnet02c <==> west-vent01

Azure AppService Plan and AppService – Deploying and Managing Web Applications

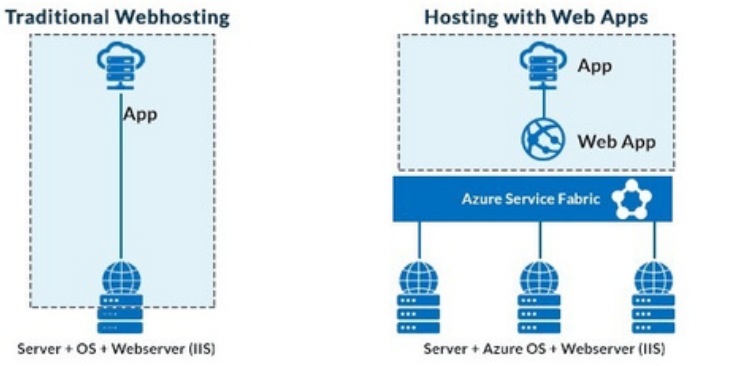
Azure App Service is a **fully managed Platform as a Service (PaaS)** that allows developers to **build, deploy, and scale** web applications, APIs, and mobile backends without managing infrastructure.

Azure Web App is a part of **Azure App Service**, designed to host **static or dynamic websites** with high availability and scalability.

Azure App Service enables developers to focus on creating outstanding applications rather than worrying about infrastructure administration.

Azure App Service is a versatile and scalable platform for developing and deploying web and mobile applications, allowing developers to rapidly and easily create applications that can be accessed by users globally.

Feature	Description
Managed Hosting	No need to manage infrastructure; Azure takes care of it.
Multiple Language Support	Supports .NET, Java, PHP, Node.js, Python, etc.
Scaling	Auto-scaling based on traffic and demand.
Security	SSL, Authentication, and Firewall protection.
Integration	Works with Azure DevOps, GitHub CI/CD, and Containers.



App Service Plan Details

Azure App Service Plans determine the pricing, performance, and scalability of your application. Below are the different types of plans:

Plan Type	Features	Use Cases
Free (F1)	60 minutes/day compute, Shared CPU	Learning, testing, personal projects
Shared (D1)	Shared CPU, Basic networking	Small websites with low traffic
Basic (B1-B3)	Dedicated VM, No auto-scaling	Low to medium traffic applications
Standard (S1-S3)	Auto-scaling, Staging slots	Business applications, moderate workloads
Premium (P1-P3, P1v2-P3v2)	High performance, more scaling options	Large enterprise apps, high traffic sites
Isolated (I1-I3, I1v2-I3v2)	Runs in Azure Virtual Network, high security	Compliance-driven applications, private networks

AzureApp Service

Azure App Service is a **fully managed** platform-as-a-service (PaaS) that allows you to build, deploy, and scale web applications, APIs, and mobile backends quickly.

Key Features of Azure App Service:

- **Supports Multiple Languages:** .NET, Java, Python, Node.js, PHP, and more.
- **Fully Managed Platform:** Handles scaling, patching, and load balancing automatically.
- **Built-in Security:** Provides SSL certificates, authentication, and authorization.
- **CI/CD Integration:** Supports deployment from GitHub, Azure DevOps, and Docker.
- **Global Availability:** Deployed across Azure's worldwide data centers.
- **Hybrid and On-premises Connectivity:** Integrates with VPN, ExpressRoute, and private networks.
- **Monitoring & Log**
- **ging:** Built-in tools like Application Insights for performance tracking.

App ServiceType	Purpose	Use Cases
Web Apps	Host and run web applications.	Websites, blogs, web portals.
API Apps	Deploy and manage RESTful APIs.	Microservices, API backends for mobile apps.
Mobile Apps	Backend services for mobile applications.	Push notifications, authentication, offline sync.
Azure Functions	Event-driven compute execution.	Background tasks, automation, event processing.

Real-World Scenario: E-commerce Website using Azure App Service

1. Corporate Website Hosting (Web App + Standard Plan)

A corporate IT company **wants to host a** business website **on Azure.**
Solution: **Web App +** Standard App Service Plan (S1/S2)
Benefit: **Hassle-free hosting with auto-scaling.**

2. E-commerce Platform (Web App + Premium Plan)

A startup **is looking for a** scalable and high-performance **e-commerce application.**
Solution: **Web App +** Premium Plan (P2v2)
Benefit: **Handles high traffic with fast loading times.**

3. Enterprise CRM Software (Web App + Isolated Plan)

A banking institution **needs to host a** secure CRM application **within a private network.**
Solution: **Web App +** Isolated App Service Plan (I3)
Benefit: **Enhanced security with private networking.**

A retail company wants to host its e-commerce website on Azure without worrying about infrastructure management. Here's how they utilize Azure App Service:

By using Azure App Service, the company ensures high availability, security, and scalability for its online store.

- **Web Apps for Frontend:** The website is deployed as a Web App.
- **API Apps for Backend Services:** The company's product catalog, payments, and order management are exposed via REST APIs.
- **Auto-Scaling and Load Balancing:** Automatically handles traffic spikes during festive sales.

Lab: Deploying a Web App on Azure

Step 1: Create an App Service Plan

1. Log in to the **Azure Portal**.
2. Search for **App Services** and click **+ Create**.
3. Choose **Subscription** & **Resource Group**.
4. Select **App Service Plan** and **Pricing Tier (Free/Standard/Premium)**.
5. Click **Review + Create** to deploy.

Step 2: Create a Web App

1. Navigate to **App Services > + Add** and select **Web App**.
2. Provide an **App Name** and choose **Runtime Stack (.NET, Java, Python, etc.)**.
3. Select an **App Service Plan** and click **Review + Create**.
4. Once deployed, access your Web App via the **provided URL**.

Validation: A web application is deployed successfully, and you can access it via a public URL provided by Azure.

Azure Functions

Azure Functions is a serverless computeservicethatallows you to run small pieces of code (functions) without managing the infrastructure. Functions are triggered by events, such as HTTP requests, timers, or messages in a queue. It is ideal for workloads that need to be event-driven or require rapid scaling.

Real-World Use Case:

- **Use Case**: VIITOR Technologies uses **Azure Functions** for processing real-time data related to student activity on their **SKILLORA platform**. When a student completes a lesson, an Azure Function is triggered to update their progress in the database and notify the instructor.

Azure API Management (APIM):

Azure API Management (APIM) is a **fully managed service** that allows organizations to publish, secure, analyze, and manage APIs in a centralized manner. It enables businesses to expose APIs securely to external and internal consumers while enforcing policies like rate-limiting, authentication, and logging

Key Features of Azure API Management:

- **API Gateway:** Routes API requests, applies security policies, and manages traffic.
- **Developer Portal:** Self-service portal for API documentation and testing.
- **Security & Authentication:** Supports OAuth2, JWT, and Azure AD integration.
- **Rate Limiting & Throttling:** Controls access and prevents abuse.
- **Analytics & Monitoring:** Tracks API performance, usage, and error rates.

Industry	Scenario	How APIM Helps
E-commerce	A retailer wants to expose product catalog APIs to third-party vendors.	APIM secures API access, applies rate limits, and provides analytics on API usage.
Banking & Finance	A bank needs to expose account transaction APIs securely to fintech partners.	APIM ensures secure authentication, compliance with banking regulations, and API analytics.
Healthcare	A hospital system integrates APIs for patient data sharing with insurance providers.	APIM enforces HIPAA compliance, monitors API traffic, and manages access policies.
Logistics & Supply Chain	A logistics company provides shipment tracking APIs to customers and vendors.	APIM helps control API access, prevents excessive requests, and logs API performance.
IoT & Smart Devices	A smart home company allows third-party developers to access device control APIs.	APIM secures API endpoints, implements user-based authentication, and limits API calls per user.

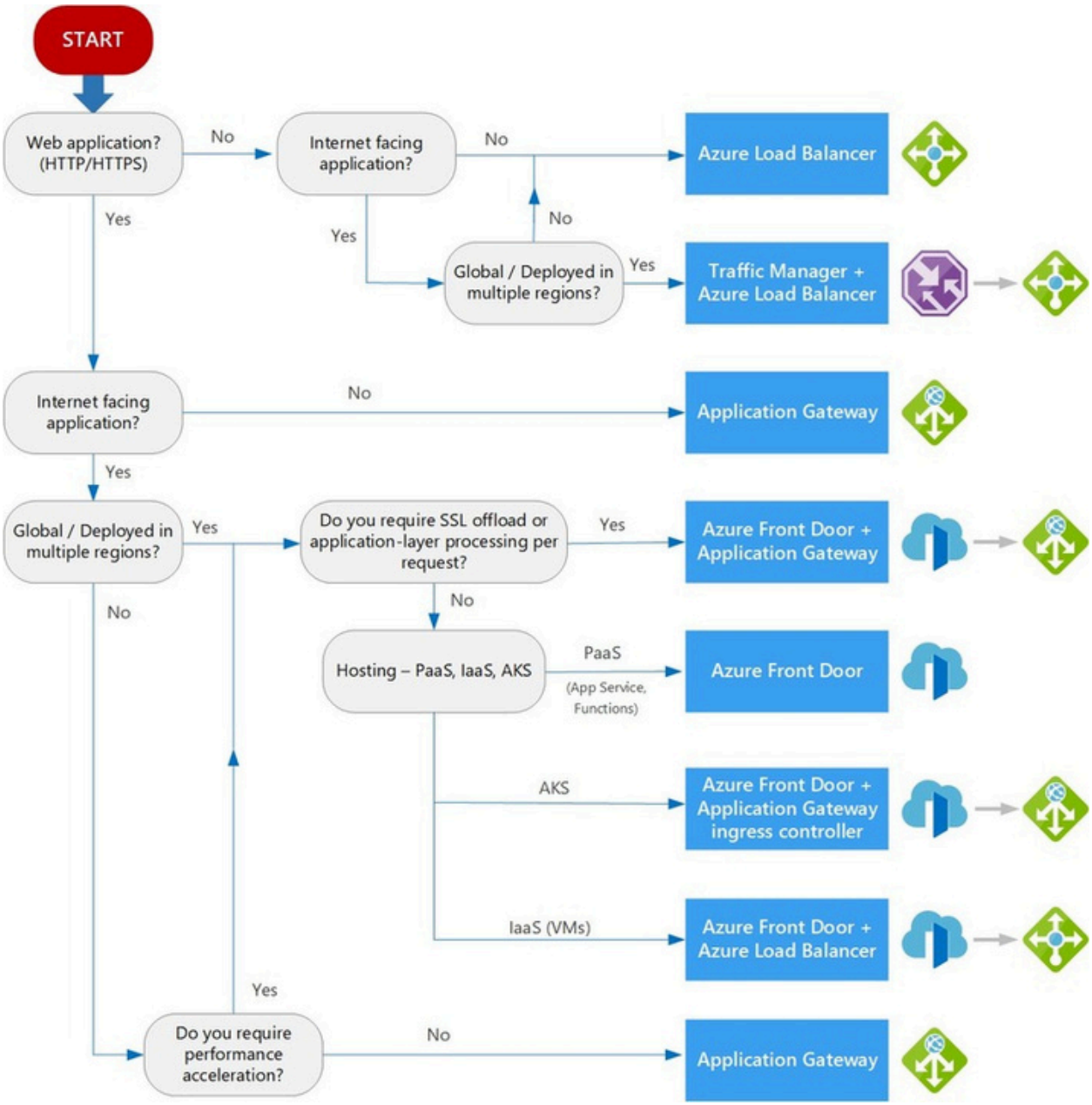
Azure Load Balancers

Azure Load Balancer is a **highly available and scalable service** that distributes incoming traffic across multiple virtual machines (VMs) to ensure high availability and reliability of applications.

Key Features of Azure Load Balancer:

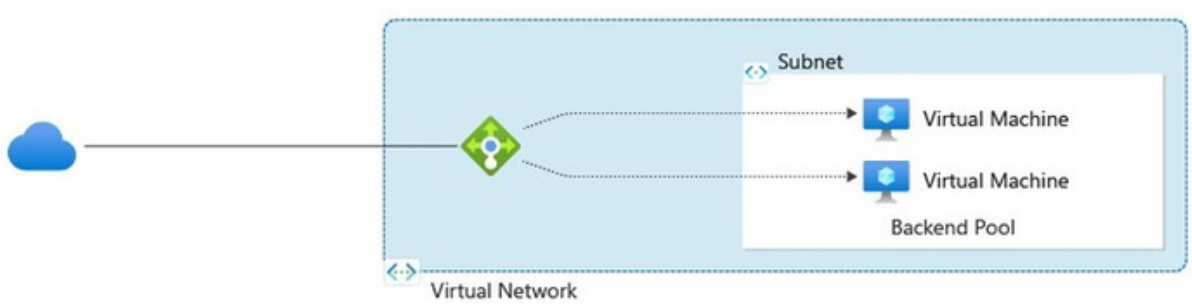
- **Automatic Load Distribution:** Balances traffic across backend servers for optimal performance.
- **High Availability:** Ensures that applications remain accessible even if individual VMs fail.
- **Public & Internal Load Balancing:** Supports external applications and internal workloads.
- **Health Probes:** Monitors the status of backend VMs and removes unhealthy instances.
- **Security Integration:** Works with **Azure Firewall** and **DDoS Protection** for enhanced security.
- **Scalability:** Handles millions of requests per second with minimal latency.
- **Outbound Connectivity:** Provides NAT functionality for outbound connections from virtual networks.

Service	Layer	Purpose	Best Use Case
Azure Load Balancer	Layer 4 (TCP/UDP)	Distributes VM traffic	High availability for backend services
Application Gateway	Layer 7 (HTTP/HTTPS)	Web traffic management	Secure, intelligent web request routing
Traffic Manager	DNS-Based	Global traffic routing	Multi-region failover and geo-routing
Azure Front Door	Global Application Acceleration	Fast, secure web application delivery	Global web applications with low latency needs


START
Web application? (HTTP/HTTPS)
No
Internet facing application?
No
Azure Load Balancer
Yes
Yes
No
Global / Deployed in multiple regions?
Yes
Traffic Manager + Azure Load Balancer
Internet facing application?
No
Application Gateway
Yes
Global / Deployed in multiple regions?
Yes
Do you require SSL offload or application-layer processing per request?
Yes
Azure Front Door + Application Gateway
No
No
Hosting – PaaS, IaaS, AKS
PaaS
(App Service, Functions)
Azure Front Door
AKS
Azure Front Door + Application Gateway ingress controller
IaaS (VMs)
Azure Front Door + Azure Load Balancer
Yes
Do you require performance acceleration?
No
Application Gateway

Azure Load Balancer (Layer 4 - TCP/UDP):

Azure Load Balancer is a **Layer 4 (TCP/UDP) load balancer** that distributes incoming traffic across multiple virtual machines (VMs) or virtual machine scale sets (VMSS) to ensure high availability and scalability.



Key Features:

- **Automatic Load Distribution:** Balances traffic across backend servers for optimal performance.
- **High Availability:** Ensures that applications remain accessible even if individual VMs fail.
- **Public & Internal Load Balancing:** Supports external applications and internal workloads.
- **Health Probes:** Monitors the status of backend VMs and removes unhealthy instances.
- **Security Integration:** Works with **Azure Firewall** and **DDoS Protection** for enhanced security.
- **Scalability:** Handles millions of requests per second with minimal latency.
- **Outbound Connectivity:** Provides NAT functionality for outbound connections from virtual networks.

Use Case: High Availability for Backend Database Servers

Real World-Scenario:

A financial institution runs critical transaction-processing applications that require a highly available and fault-tolerant backend database system. They have multiple database servers running on Azure Virtual Machines (VMs) within a Virtual Network (VNet).

Solution with Azure Load Balancer (Layer 4 - TCP/UDP):

1. **Deploy an Azure Load Balancer:**
    - Configure it in **Internal Mode** (for private traffic within Azure).
    - Use **TCP-based load balancing** to distribute incoming database requests.
2. **Set Up Health Probes:**
    - Monitor the availability of backend database servers to route traffic only to healthy nodes.
3. **Automatic Failover:**
    - If a database server fails, the load balancer removes it from the pool and directs traffic to the remaining healthy servers.

Key Configurations of Azure Layer 4 (L4) Load Balancer

1. Load Balancer Type

- **Public Load Balancer** – Distributes internet traffic to VMs in a backend pool.
- **Internal Load Balancer** – Distributes private network traffic within Azure Virtual Network (VNet).

2. Frontend IP Configuration

- Assigns a **public or private IP address** to receive incoming traffic.
- Supports **Static and Dynamic IP allocation**.

3. Backend Pool

- Group of Virtual Machines (VMs) or Virtual Machine Scale Sets that receive traffic.
- Supports **multiple backend pools** for different applications.

4. Load Balancing Rules

- Defines how traffic is distributed between backend servers.
- Supports **TCP and UDP** protocols at **Layer 4**.
- Configurable **port mapping** (e.g., mapping incoming port 80 to backend port 8080).

5. Health Probes

- Continuously checks the health of backend servers.
- If a VM is **unhealthy**, the load balancer stops sending traffic to it.
- Configurable **protocols:** HTTP, HTTPS, TCP, and UDP.

6. NAT Rules (Inbound & Outbound)

- **Inbound NAT Rules** – Forward traffic from a specific port on the Load Balancer to a single VM.
- **Outbound Rules** – Enable VMs to make outbound internet connections using the Load Balancer's IP.

7. Session Persistence (Sticky Sessions)

- **None (Default):** Randomly distributes traffic across backend VMs.
- **Client IP-based Affinity:** Ensures requests from the same client go to the same VM.

8. Availability Zones & Redundancy

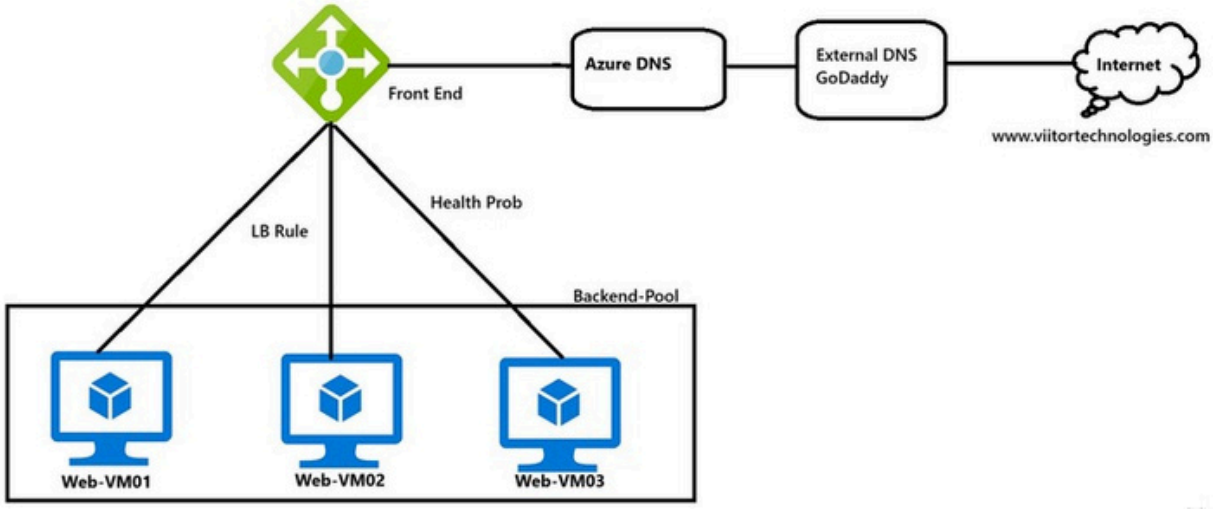
- Supports **Zone Redundant** and **Regional** configurations for high availability.
- Can distribute traffic across multiple **Availability Zones**.

9. Integration with Azure Monitor

- Provides insights into traffic flow and backend health.

- Supports **metrics, alerts, and diagnostic logs**.

LAB: This lab will guide you through the process of creating an **Azure Load Balancer (Layer 4)** and configuring it to distribute traffic across multiple backend virtual machines.



Step 1: Create a Resource Group

1. Go to **Azure Portal** → Search for **Resource Groups** → Click **Create**.
2. Provide a **Resource Group Name** (e.g., MyLoadBalancerRG).
3. Choose a **Region** (e.g., East US) and click **Review + Create** → **Create**.

Step 2: Create Virtual Machines (Backend Servers)

1. Go to **Azure Portal** → Search for **Virtual Machines** → Click **Create**.
2. Select the **Resource Group** created earlier.
3. Provide a **VM Name** (e.g., VM1 and VM2).
4. Choose an **Image** (e.g., Ubuntu Server or Windows Server).
5. Configure **Networking**:
    - Select **Create a new Virtual Network (VNet)**.
    - Create a **Subnet** (e.g., BackendSubnet).
6. Allow **RDP (for Windows) or SSH (for Linux)** for remote access.

7. Click **Review + Create → Create**.
8. Repeat the steps to create **another VM (VM2)** in the same **VNet and Subnet**.

Step 3: Install Web Server (For Testing Load Balancing)

1. Connect to both **VM1** and **VM2** using RDP/SSH.
2. Run the following commands:

For Linux (Ubuntu):

```
sudo apt update -y
sudo apt install apache2 -y
echo "Welcome to VM1" | sudo tee /var/www/html/index.html
sudo systemctl restart apache2
```

(On VM2, change "Welcome to VM1" to "Welcome to VM2".)

For Windows Server (IIS Web Server):

1. Open **PowerShell as Administrator**.
2. Run the command:

```
powershell
CopyEdit
Install-WindowsFeature -name Web-Server -IncludeManagementTools
```

3. Open C:\inetpub\wwwroot\index.html and add:

```
css
CopyEdit
Welcome to VM1
```

(Modify the message on VM2 to "Welcome to VM2".)

Step 4: Create an Azure Load Balancer (L4)

1. Go to **Azure Portal** → Search for **Load Balancers** → Click **Create**.
2. Select **Public Load Balancer**.
3. Choose the **Resource Group** created earlier.
4. Provide a **Name** (e.g., MyLoadBalancer).
5. Under **SKU**, select **Standard** for high availability.
6. In **Frontend IP Configuration**, select **Public IP** → **Create a new public IP** → Name it (e.g., LBPublicIP).
7. Click **Review + Create** → **Create**.

Step 5: Configure Backend Pool

1. Open **MyLoadBalancer** → Click **Backend Pools** → **Add Backend Pool**.
2. Provide a **Name** (e.g., BackendPool).
3. Select **Associated Virtual Network** and add both **VM1** and **VM2**.
4. Click **Add**.

Step 6: Create a Health Probe

1. Open **MyLoadBalancer** → Click **Health Probes** → **Add Health Probe**.
2. Provide a **Name** (e.g., HTTP-Probe).
3. Select **Protocol: HTTP**, **Port: 80**, and **Path: /**.
4. Click **OK**.

Step 7: Configure Load Balancing Rule

1. Open **MyLoadBalancer** → Click **Load Balancing Rules** → **Add Rule**.
2. Provide a **Name** (e.g., HTTP-Rule).
3. Select **Frontend IP:** LBPublicIP.

4. Choose **Backend Pool:** BackendPool.
5. Select **Protocol:** TCP, **Port:** 80, and **Backend Port:** 80.
6. Choose **Health Probe:** HTTP-Probe.
7. Enable **Session Persistence (Optional, Client IP-based Affinity)**.
8. Click **OK**.

Step 8: Testing the Load Balancer

1. Copy the **Public IP** of the Load Balancer (LBPublicIP).
2. Open a browser and go to:

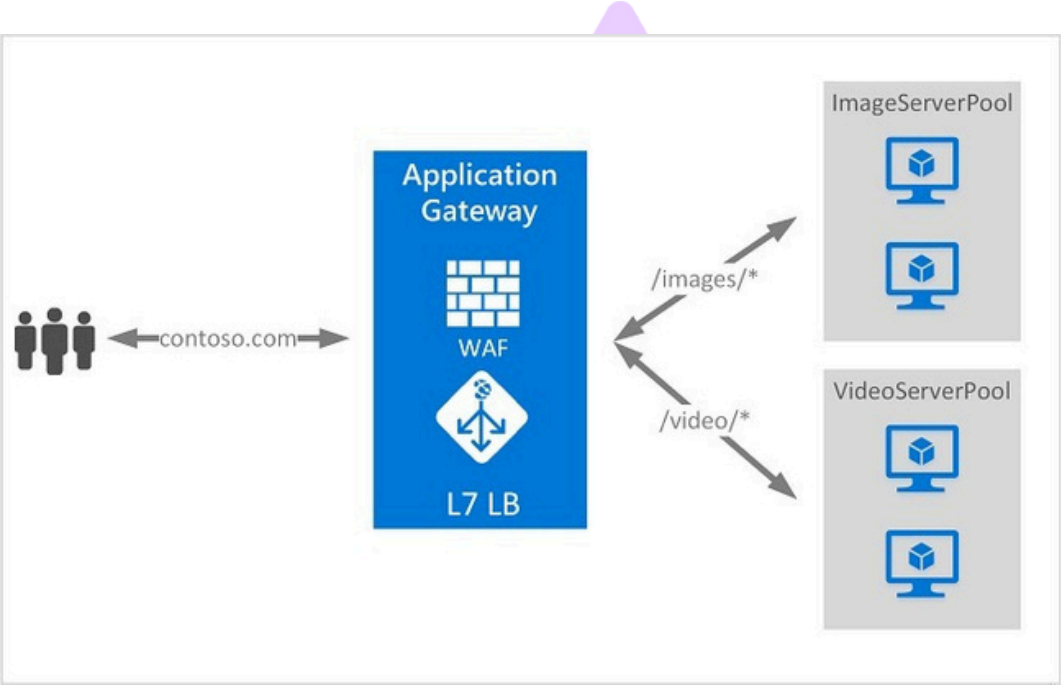
   cpp
   CopyEdit
   http://<LoadBalancer-Public-IP>

3. Refresh the page multiple times.
4. You should see **requests being distributed** between VM1 and VM2.

Azure Application Gateway (L7 Load Balancer)

Introduction

Azure Application Gateway is a **Layer 7 (Application Layer) load balancer** designed to manage web traffic efficiently. Unlike Azure Load Balancer, which works at **Layer 4 (Transport Layer)**, Application Gateway makes **routing decisions based on HTTP requests**. It supports advanced features like **URL-based routing, SSL termination, Web Application Firewall (WAF), and session affinity**.



Key Features of Azure Application Gateway

Feature	Description
Layer 7 Load Balancing	Routes traffic based on HTTP/S requests, URLs, and hostnames.
Path-Based Routing	Directs requests to different backend pools based on the request URL.
Multi-Site Hosting	Host multiple applications on the same gateway and route them to different backends.
SSL Termination	Decrypts incoming SSL requests and forwards unencrypted traffic to backend servers.
Web Application Firewall (WAF)	Protects applications from security threats like SQL injection and cross-site scripting (XSS).
Auto Scaling	Scales up or down automatically based on traffic.
Custom Health Probes	Monitors the health of backend instances and ensures traffic is routed only to healthy servers.

Types of Azure Application Gateway

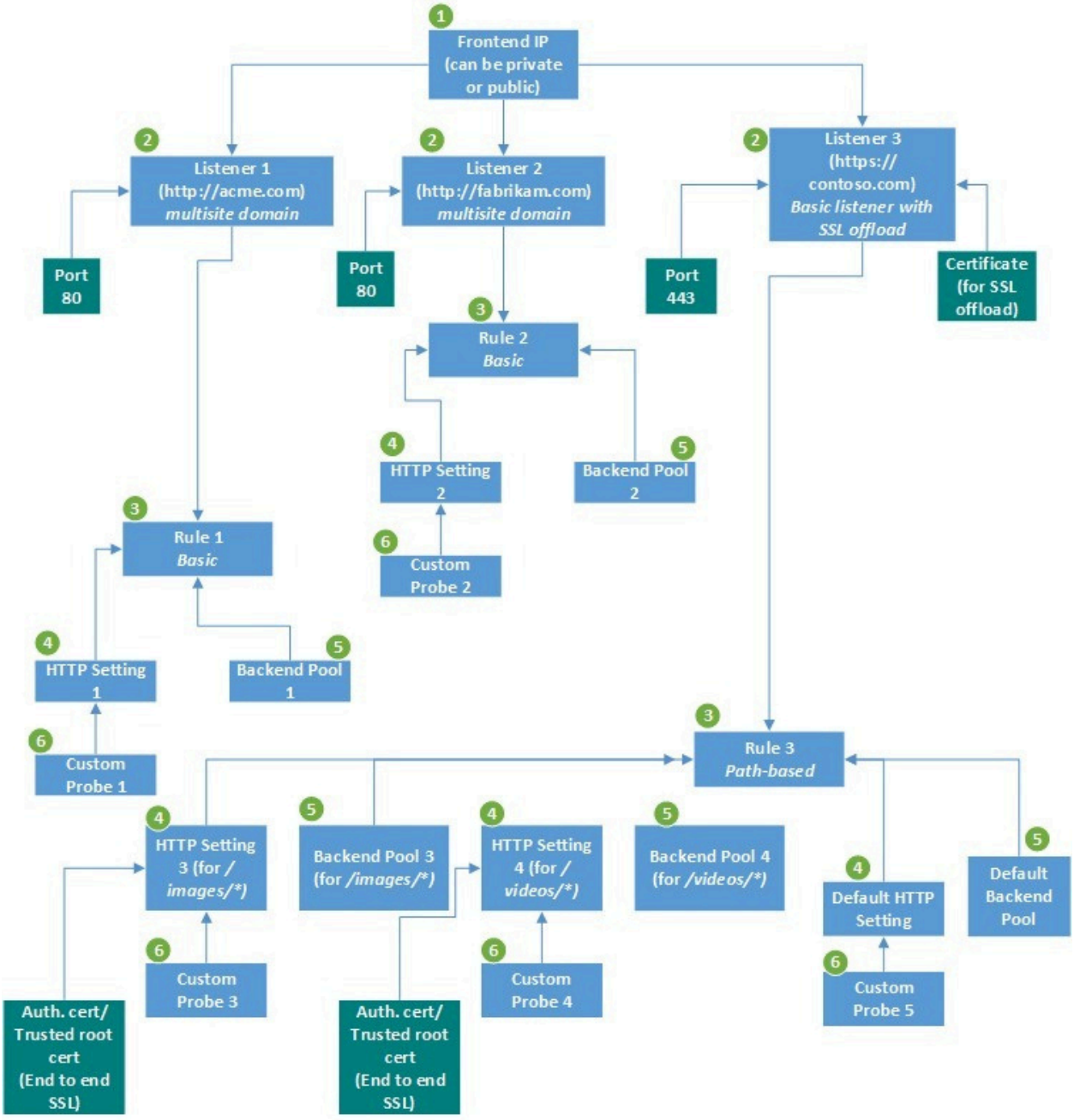
Type	**Description**
Standard	Basic Layer 7 load balancing with URL-based routing and health probes.
Standard v2	Improved performance, autoscaling, zone redundancy, and HTTPS-based health probes.
Web Application Firewall (WAF)	Includes protection against OWASP Top 10 threats, with customizable security rules.
WAF v2	Advanced WAF with autoscaling, custom rules, and better security analytics.

Comparison: Azure Load Balancer vs. Application Gateway

Feature	Azure Load Balancer (L4)	Azure Application Gateway (L7)
Layer	Layer 4 (Transport)	Layer 7 (Application)
Traffic Routing	Based on IP and port	Based on URLs, hostnames, and HTTP headers
Protocol Support	TCP, UDP	HTTP, HTTPS, WebSockets
Health Probes	Checks connections (TCP/UDP supported)	Checks HTTP responses
SSL Termination	Not supported	Supported
Path-Based Routing	No built-in security	Supported
Security		Web Application Firewall (WAF) available
Use Case	Load balancing for VMs, databases, applications	Load balancing for web applications, APIs, microservices

Azure Application Gateway consists of several components that you can configure in various ways for different scenarios

Below image illustrates an application that has three listeners. The first two are multi-site listeners for http://acme.com/* and http://fabrikam.com/*, respectively. Both listen on port 80. The third is a basic listener that has end-to-end Transport Layer Security (TLS) termination, previously known as Secure Sockets Layer (SSL) termination.


1
Frontend IP (can be private or public)
2
Listener 1 (http://acme.com) multisite domain
2
Listener 2 (http://fabrikam.com) multisite domain
2
Listener 3 (https://contoso.com) Basic listener with SSL offload
Port 80
Port 80
Port 443
Certificate (for SSL offload)
3
Rule 2 Basic
4
HTTP Setting 2
5
Backend Pool 2
6
Custom Probe 2
3
Rule 1 Basic
4
HTTP Setting 1
5
Backend Pool 1
6
Custom Probe 1
3
Rule 3 Path-based
4
HTTP Setting 3 (for /images/*)
5
Backend Pool 3 (for /images/*)
4
HTTP Setting 4 (for /videos/*)
5
Backend Pool 4 (for /videos/*)
4
Default HTTP Setting
5
Default Backend Pool
6
Custom Probe 3
6
Custom Probe 4
6
Custom Probe 5
Auth. cert/ Trusted root cert (End to end SSL)
Auth. cert/ Trusted root cert (End to end SSL)

Real-World Use Cases

1. E-commerce Website Traffic Management

A retail company with an e-commerce website uses **Azure Application Gateway** to:

- **Route traffic intelligently:** Requests to /shop go to an online store backend, while /auth goes to an authentication service.
- **Enhance security: Web Application Firewall (WAF)** blocks SQL injection and cross-site scripting (XSS) attacks.
- **Enable SSL Offloading:** Reduces server CPU load by decrypting SSL at the gateway.

2. Multi-Site Hosting for Enterprises

A multinational corporation runs multiple websites and applications under a single **Application Gateway**.

- **Website 1:** www.company.com → Routes traffic to backend pool **A**.
- **Website 2:** www.portal.company.com → Routes traffic to backend pool **B**.
- **Separate SSL certificates for each domain** to ensure secure communication.

3. Secure API Gateway

A **banking application** hosts multiple API services.

- Requests to /api/customers go to **Customer API Service**.
- Requests to /api/payments go to **Payment API Service**.
- **Rate limiting** and **WAF rules** prevent API abuse.

LAB: Deploying an Azure Application Gateway

This lab will guide you through the process of **creating an Azure Application Gateway (Layer 7 Load Balancer)** and configuring it to manage web traffic efficiently with HTTP-based routing.

Step 1: Create a Resource Group

1. Go to **Azure Portal** → Search for **Resource Groups** → Click **Create**.
2. Provide a **Resource Group Name** (e.g., MyAppGatewayRG).
3. Choose a **Region** (e.g., East US) and click **Review + Create** → **Create**.

Step 2: Create Virtual Machines (Backend Web Servers)

1. Go to **Azure Portal** → Search for **Virtual Machines** → Click **Create**.
2. Select the **Resource Group** created earlier.
3. Provide a **VM Name** (e.g., WebVM1 and WebVM2).
4. Choose an **Image** (e.g., Ubuntu Server or Windows Server).
5. Configure **Networking**:
    - Select **Create a new Virtual Network (VNet)**.
    - Create a **Subnet** (e.g., AppGatewaySubnet).
6. Allow **RDP (for Windows) or SSH (for Linux)** for remote access.
7. Click **Review + Create** → **Create**.
8. Repeat the steps to create **another VM (WebVM2)** in the same **VNet and Subnet**.

Step 3: Install Web Server (For Testing Routing)

1. Connect to both **WebVM1** and **WebVM2** using RDP/SSH.
2. Run the following commands:

For Linux (Ubuntu):

```
sudo apt update -y
sudo apt install apache2 -y
echo "Welcome to Web Server 1" | sudo tee /var/www/html/index.html
sudo systemctl restart apache2
```

(On WebVM2, change "Welcome to Web Server 1" to "Welcome to Web Server 2".)

For Windows Server (IIS Web Server):

1. Open **PowerShell as Administrator**.
2. Run the command:

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

3. Open C:\inetpub\wwwroot\index.html and add:

   ```
   Welcome to Web Server 1
   ```

(Modify the message on WebVM2 to "Welcome to Web Server 2".)

Step 4: Create an Azure Application Gateway

1. Go to **Azure Portal** → Search for **Application Gateway** → Click **Create**.
2. Choose the **Resource Group** created earlier.
3. Provide a **Name** (e.g., MyAppGateway).
4. Choose **Region** (e.g., East US).
5. Under **Virtual Network**, select the existing **VNet** from Step 2.
6. Create a new **Subnet** (e.g., AppGatewaySubnet).
7. Under **Frontend IP Configuration**, select **Public IP** → **Create a new public IP** → Name it (e.g., AppGatewayPublicIP).
8. Under **Backend Pool Configuration**, create a **Backend Pool** and add **WebVM1** and **WebVM2**.
9. Click **Review + Create** → **Create**.

Step 5: Create an HTTP Listener

1. Open **MyAppGateway** → Click **Listeners** → **Add Listener**.
2. Provide a **Name** (e.g., HTTP-Listener).
3. Choose **Frontend IP** as AppGatewayPublicIP.
4. Set **Protocol** to HTTP, and **Port** to 80.
5. Click **Add**.

Step 6: Configure Routing Rules

1. Open **MyAppGateway** → Click **Rules** → **Add Rule**.
2. Provide a **Rule Name** (e.g., HTTP-Rule).
3. Select **Listener:** HTTP-Listener.
4. Under **Backend Target**, select **Backend Pool:** WebVM-Pool.
5. Choose **HTTP Settings** → Click **Add New HTTP Setting** → Name it HTTP-Setting.
6. Set **Port** to 80, Enable **Cookie-Based Affinity**, and Enable **Health Probes**.
7. Click **Add**.

Step 7: Create a Health Probe

1. Open **MyAppGateway** → Click **Health Probes** → **Add Health Probe**.
2. Provide a **Name** (e.g., HTTP-Probe).
3. Select **Protocol: HTTP**, **Port: 80**, and **Path: /**.
4. Click **OK**.

Step 8: Testing the Application Gateway

1. Copy the **Public IP** of the Application Gateway (AppGatewayPublicIP).

2. Open a browser and go to:

   http://<AppGateway-Public-IP>

3. Refresh the page multiple times.
4. You should see **requests being routed** between WebVM1 and WebVM2.

Expected Output

- If the request goes to **WebVM1**, you will see "Welcome to Web Server 1".
- If the request goes to **WebVM2**, you will see "Welcome to Web Server 2".

Azure Traffic Manager – Global Traffic Routing Solution

Introduction

Azure Traffic Manager is a **DNS-based traffic routing solution** that distributes user traffic across multiple global endpoints. Unlike **Azure Load Balancer (Layer 4)** or **Application Gateway (Layer 7)**, Traffic Manager operates at the **DNS level (Layer 3)** to direct users to the nearest or best-performing application instance based on **routing policies**.

Traffic Manager **does not handle actual traffic** like a load balancer but helps clients resolve the best possible endpoint using **intelligent DNS resolution**.

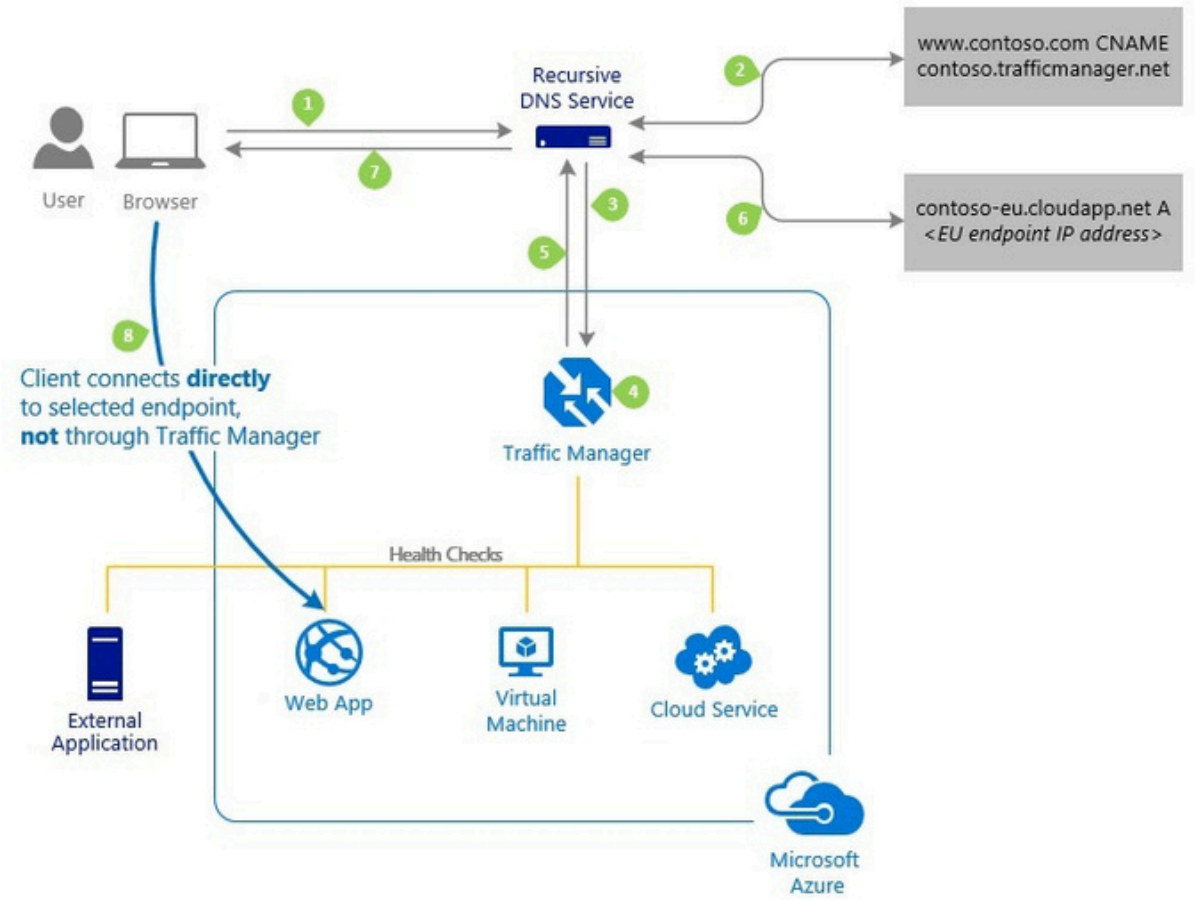
Key Features of Azure Traffic Manager

Feature	Description
Global Traffic Distribution	Routes users to the nearest Azure region for lower latency.
DNS-Based Load Balancing	Works at the DNS layer and does not act as a traditional load balancer.

Multiple Method Routing	Offers **Priority, Performance, Weighted, Geographic, Multivalue, and Subnet routing**.
Automatic Failover	Detects unhealthy endpoints and redirects traffic to healthy instances.
Multi-Cloud and Hybrid Support	Can route traffic to Azure, on-premises, or other cloud providers.
Traffic Analytics	Provides insights into traffic distribution, endpoint health, and failover events.

Types of Routing Methods in Traffic Manager

Routing Method	Description	Use Case
Priority	Routes all traffic to primary endpoint; fails over to the next in case of failure.	Disaster recovery setups.
Performance	Routes users to the closest and lowest latency endpoint.	Global applications with multiple data centers.
Weighted	Distributes traffic based on assigned weights to different endpoints.	A/B testing, gradual rollouts.
Geographic	Routes users based on their geographical location.	Compliance-based applications, regional data sovereignty.
Multivalue	Returns multiple healthy endpoints to the client for load balancing.	Resilient DNS-based failover.
Subnet	Routes traffic based on the client's IP address subnet.	Custom traffic segmentation by subnet.



1. The client sends a DNS query to its configured recursive DNS service to resolve the name 'partners.contoso.com'. A recursive DNS service, sometimes called a 'local DNS' service, does not host DNS domains directly. Rather, the client off-loads the work of contacting the various authoritative DNS services across the Internet needed to resolve a DNS name.
2. To resolve the DNS name, the recursive DNS service finds the name servers for the the 'contoso.com' domain. It then contacts those name servers to request 'partners.contoso.com' DNS record. The contoso.com DNS servers return the CNAME record that points to contoso.trafficmanager.net.
3. Next, the recursive DNS service finds the name servers for the 'trafficmanager.net' domain, which are provided by the Azure Traffic Manager service. It then sends a request for the 'contoso.trafficmanager.net' DNS record to those DNS servers.
4. The Traffic Manager name servers receive the request. They choose an endpoint based on:

    - The configured state of each endpoint (disabled endpoints are not returned)

- The current health of each endpoint, as determined by the Traffic Manager health checks. For more information,
- The chosen traffic-routing method. For more information

2. The chosen endpoint is returned as another DNS CNAME record. In this case, let us suppose contoso-eu.cloudapp.net is returned. Next, the recursive DNS service finds the name servers
3. for the 'cloudapp.net' domain. It contacts those name servers to request the 'contoso-eu.cloudapp.net' DNS record. A DNS 'A' record containing the IP address of the EU-based service endpoint is returned. The recursive DNS service consolidates the results and returns
4. a single DNS response to the client. The client receives the DNS results and connects to the given IP address. The client connects to the application service endpoint directly, not through
5. Traffic Manager. Since it is an HTTPS endpoint, the client performs the necessary SSL/TLS handshake, and then makes an HTTP GET request for the '/login.aspx' page.

Real-World Use Cases

1. Global Web Application with Low Latency Routing

A company has **multiple Azure Web Apps** deployed in **USA, Europe, and Asia**.

- **Traffic Manager (Performance Routing)** ensures users in **Asia** access the Asian data center, USA users access the US data center, and so on.
- This **improves application response time** by directing users to the nearest available instance.

2. Disaster Recovery & High Availability

A **financial services company** has a **primary application in the East US** and a **backup in the West US**.

- Using **Priority Routing**, all traffic is directed to **East US**.
- If the **East US** region goes down, **Traffic Manager redirects** traffic to **West US** automatically.

3. Multi-Cloud & Hybrid Deployment

A **media streaming service** runs its primary application in **Azure** but maintains backup infrastructure in **AWS**.

- **Traffic Manager ensures seamless failover** between Azure and AWS based on endpoint health.
- Users experience **zero downtime** in case of an Azure outage.

Lab: Deploying Azure Traffic Manager for Global Routing

Step 1: Create a Traffic Manager Profile

1. **Go to Azure Portal** → Search for **Traffic Manager Profiles** → Click **Create**.
2. Enter **Profile Name** and select **Routing Method** (Priority, Performance, Weighted, etc.).
3. Choose **Subscription** and **Resource Group**, then click **Create**.

Step 2: Add Endpoints

1. Inside the **Traffic Manager Profile**, go to **Endpoints** → Click **Add Endpoint**.
2. Select **Target Resource Type** (Azure Web App, Public IP, External Endpoint, etc.).
3. Choose **Region and Endpoint Priority/Weight** based on routing policy.
4. Repeat the process to add **multiple endpoints** across different locations.

Step 3: Configure Health Probes

1. **Navigate to Health Probes** → Set up **Protocol (HTTP/HTTPS)** and **Port**.
2. Define **Health Check Interval** and **Tolerated Failures** to ensure traffic is only routed to healthy endpoints.

Step 4: Test the Traffic Routing

1. Copy the **Traffic Manager DNS Name** and open it in a browser.
2. Traffic will be routed based on the chosen **routing method**.
3. Simulate failures by stopping an endpoint and check **automatic failover behavior**.

Azure Front Door – Global Load Balancer & Web Application Accelerator

Introduction

Azure Front Door is a **global, application-level load balancer and content delivery network (CDN)** that ensures high availability, low latency, and security for web applications. Unlike **Azure Load Balancer (L4)** and **Application Gateway (L7)**, which operate **regionally**, Front Door is designed to **route global HTTP/HTTPS traffic** efficiently.

Azure Front Door is Microsoft's advanced cloud Content Delivery Network (CDN) designed to provide fast, reliable, and secure access to your applications' static and dynamic web content globally. By using Microsoft's extensive global edge network, Azure Front Door

Front Door uses **Microsoft's global edge network** to accelerate application performance and enhance security with **Web Application Firewall (WAF)** capabilities.

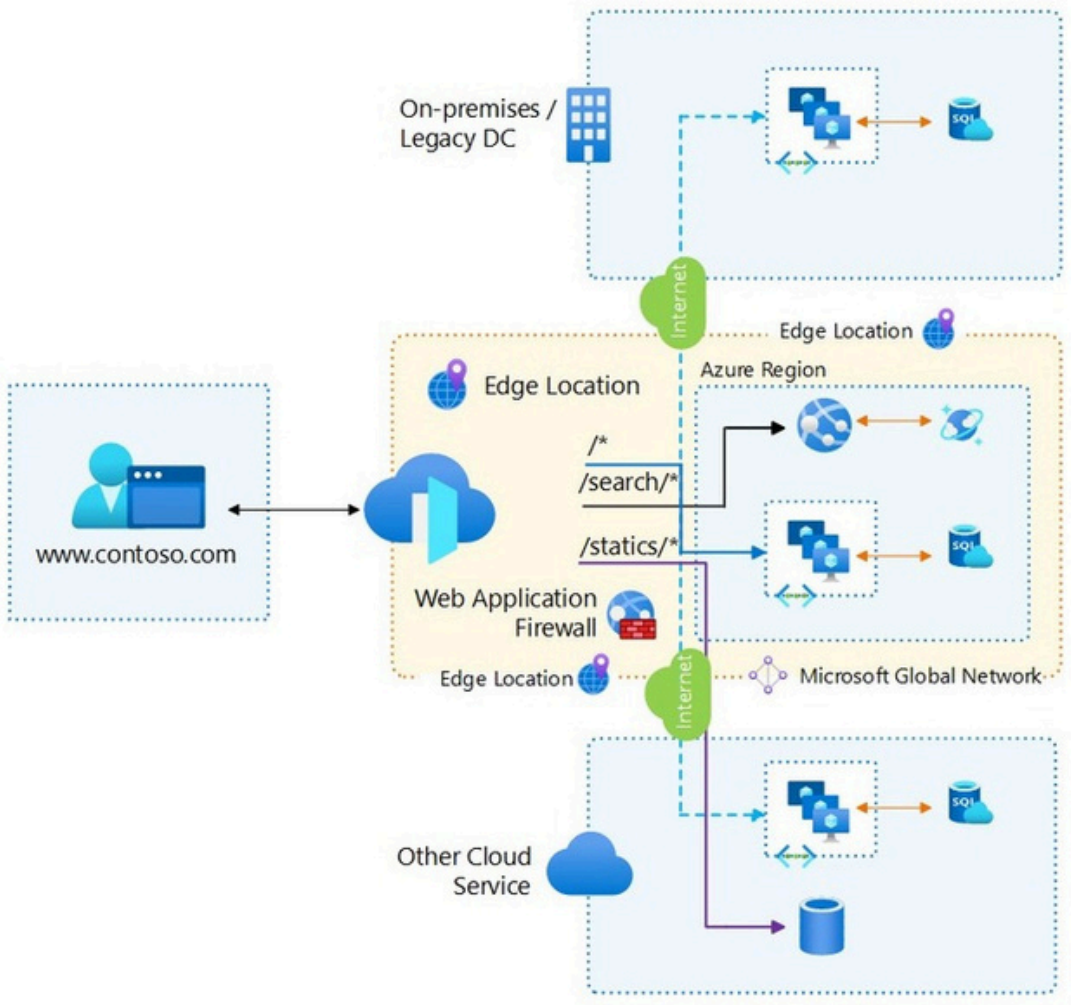
Key Features of Azure Front Door:

Feature	Description
Global HTTP Load Balancing	Routes traffic intelligently across multiple global endpoints.
Intelligent Routing	Uses latency-based routing and session affinity for optimal user experience.
Caching & CDN Integration	Delivers content faster by caching at the edge locations.
Web Application Firewall (WAF)	Protects against **DDoS, SQL injection, and cross-site scripting attacks**.
SSL/TLS Offloading	Terminates SSL connections at the edge for improved performance.
Zero-Downtime Deployment	Supports A/B testing, blue-green deployments, and canary releases.
Multi-Cloud and Hybrid Support	Can route traffic to Azure, on-premises, and even AWS or GCP endpoints.

Why Use Azure Front Door?

- **Faster Application Performance** – Content caching reduces latency.
- **Global Failover & High Availability** – Ensures zero downtime.

- **Advanced Security** – Protects against DDoS, SQL injection, and other threats.
- **Multi-Cloud & Hybrid Capabilities** – Works across Azure, AWS, and on-premises environments.
- **Zero-Downtime Deployments** – Supports blue-green and canary releases.



Real-World Use Cases

1. E-commerce Website with Global Customers

A retail company with customers worldwide wants to ensure:

- **Faster page load times** for users across different continents.
- **Global failover** in case of a regional outage.
- **Protection against cyberattacks** like **DDoS** and **SQL injection**.

Solution:

- **Azure Front Door** routes customers to the nearest online store based on **latency-based routing**.
- **CDN caching** improves website loading speed.
- **WAF rules** block malicious attacks automatically.

2. Multi-Cloud Deployment with Azure & AWS

A media streaming company uses Azure for video encoding but **stores content in AWS S3**.

- **Azure Front Door directs users** to the closest server (Azure or AWS) dynamically.
- **SSL offloading at edge locations** improves security and reduces latency.
- **Automatic failover** ensures that users experience **zero downtime**.

3. Blue-Green Deployment for a Banking App

A **banking application** wants to **gradually roll out** a new feature:

- **Front Door splits traffic** between the **old (Blue) and new (Green) versions**.
- **A/B testing** helps analyze user experience before full deployment.
- If issues arise, **traffic is switched back** to the stable version instantly.

Lab: Deploying Azure Front Door for Global Web Traffic Management

Step 1: Create an Azure Front Door Profile

1. **Go to Azure Portal** → Search for **Front Door** → Click **Create**.
2. Choose **Subscription** and **Resource Group**, then enter a **Front Door Name**.
3. Select **Pricing Tier** based on your requirements.

Step 2: Configure Front Door Routing

1. Inside the **Front Door Profile**, go to **Routing Rules** → Click **Add Rule**.
2. Set up **Frontends** (Custom Domain or Azure-provided DNS).
3. Define **Backend Pools** (Add web apps, storage accounts, or external services like AWS).
4. Choose **Load Balancing Method** (Latency-based, Priority-based, or Weighted).

Step 3: Enable Security & Performance Features

1. **Enable Web Application Firewall (WAF)** to protect against threats.
2. **Enable Caching & CDN** for better performance.
3. **Configure Session Affinity** to maintain user sessions across requests.

Step 4: Test Global Load Balancing

1. Copy the **Front Door URL** and access it from different locations.
2. Verify that traffic is routed based on the closest or best-performing endpoint.
3. Simulate failure by **stopping a backend service** and observe automatic failover.

Feature	**Azure Front Door (L7, Global)**	**Traffic Manager (DNS-based)**	**Application Gateway (L7, Regional)**	**Azure Load Balancer (L4, Regional)**
Layer	Layer 7 (Application)	Layer 3 (DNS)	Layer 7 (Application)	Layer 4 (Transport)
Load Balancing Scope	Global HTTP-based routing	Global DNS-based routing	Regional HTTP-based routing	Regional VM-level balancing
Traffic Handling	HTTP(S) routing & caching	DNS-based endpoint selection	Web traffic routing with WAF	Network-level VM distribution
Performance Acceleration	Yes (via Edge Caching & CDN)	No	No	No
Multi-Cloud & Hybrid	Yes	Yes	No	No

Best Use Case	Global web acceleration & security	Disaster recovery & global traffic routing	Regional web application management	Load balancing for VMs & microservices

Azure Storage Account – Scalable, Secure & Highly Available Cloud Storage

An **Azure Storage Account** is a **highly available, durable, and scalable** cloud storage solution for storing various types of data, including files, blobs, tables, and queues. It provides multiple **redundancy options, security features, and integration with Azure services** for cost-effective storage management.

Azure Storage is **designed to handle massive-scale workloads**, including **Big Data analytics, backups, IoT applications, and enterprise solutions**.

Storage account in Azure is a method of creating a storage service for storing data in it. It contains Blob, queue, tables, and files with disk images. It uniquely provides namespace and service access to functions of storage.

It is durable, highly available, and scalable. By using Azure storage account services, we don't need to worry about space because it will be scaled upon our demand. The Azure storage account is a container that groups a set of Azure storage services together

Azure Storage Account అనేద **హై అవైలబ్లలిటీ, డోర్బుల్ట మరియు స్తకలబుల్ట ◌ౌడ్ స్ప్రుజజ్** సొల్యేషన, ఇద వివిధ ర్కాల డేటాను (ఫైల్స్, క్లడోబ్, టేబుల్స్, కూూప్) భత్తపర్చడానికి ఉపయోగంచబడుతంద.

Key Features of Azure Storage Account

Feature	Description
Durability & High Availability	Data is automatically replicated across Azure data centers.
Security & Compliance	Encryption, role-based access control (RBAC), and firewall rules ensure data protection.
Scalability	Automatically scales with demand, supporting petabytes of data.
Integration with Azure Services	Works with Azure Functions, Virtual Machines, App Services, and Kubernetes.
Cost-Effective	Pay-as-you-go model with different redundancy and performance tiers.
Multi-Protocol Access	Supports REST API, SMB, NFS, and more for seamless data access.

Storage Account types

Storage type defines the methodology for storing data in Azure infrastructure. it gives the solution to the question of what type and how to store data in Azure.

- **Accessible via REST API**: Queue, table, Blob
- **Designed for Microsoft Azure Virtual machines**: File storage, Disk storage

Key Features of Azure Storage Account:

- **Durability**: Provides high durability by storing data across multiple locations.
- **Scalability**: Scalable storage solutions for both small and large data.
- **Security**: Encryption at rest and in transit, access control via Azure Active Directory (Azure AD).
- **Performance**: Optimized for high throughput and low latency access to data.
- **Multiple Storage Types**: Supports Blob, File, Table, Queue, and Disk storage.

Storage Type	Purpose	Common Use Cases
Blob Storage	Stores unstructured object data.	Images, videos, backups, logs.
File Storage (Azure Files)	Managed file shares accessible via SMB/NFS.	Shared drives for applications, lift-and-shift migrations.
Table Storage	NoSQL key-value storage.	Storing structured, non-relational data.
Queue Storage	Message-based communication between components.	Decoupling microservices, event-driven applications.
Disk Storage	Managed virtual hard disks (VHDs) for VMs.	Persistent storage for virtual machines.

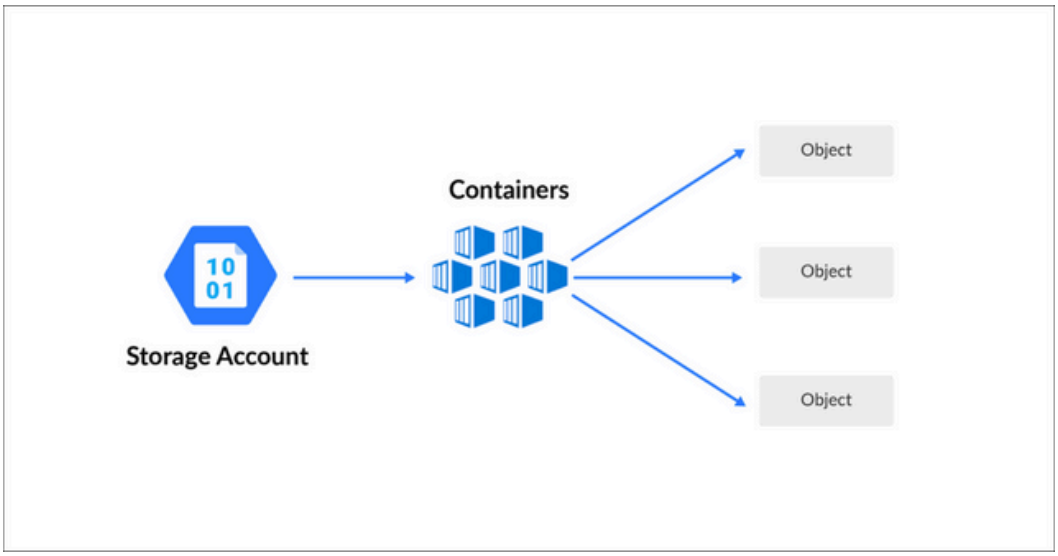
1. Azure Blob Storage Azure Blob Storage is a Microsoft Azure cloud-based object storage solution. It is intended to store and manage unstructured data at scale, such as text or binary data such as photos, videos, documents, and other file formats.

- It is an object storage solution in Azure.
- It is used to store unstructured data.
- This is ideal when you have storage solutions for files, videos, log files, and images
- It has different tier levels:

Hot storage tier: It is ideal for objects that are accessed frequently

Cool storage tier: It is optimized for data that are infrequently accessed. This is a less expensive option than the hot storage tier

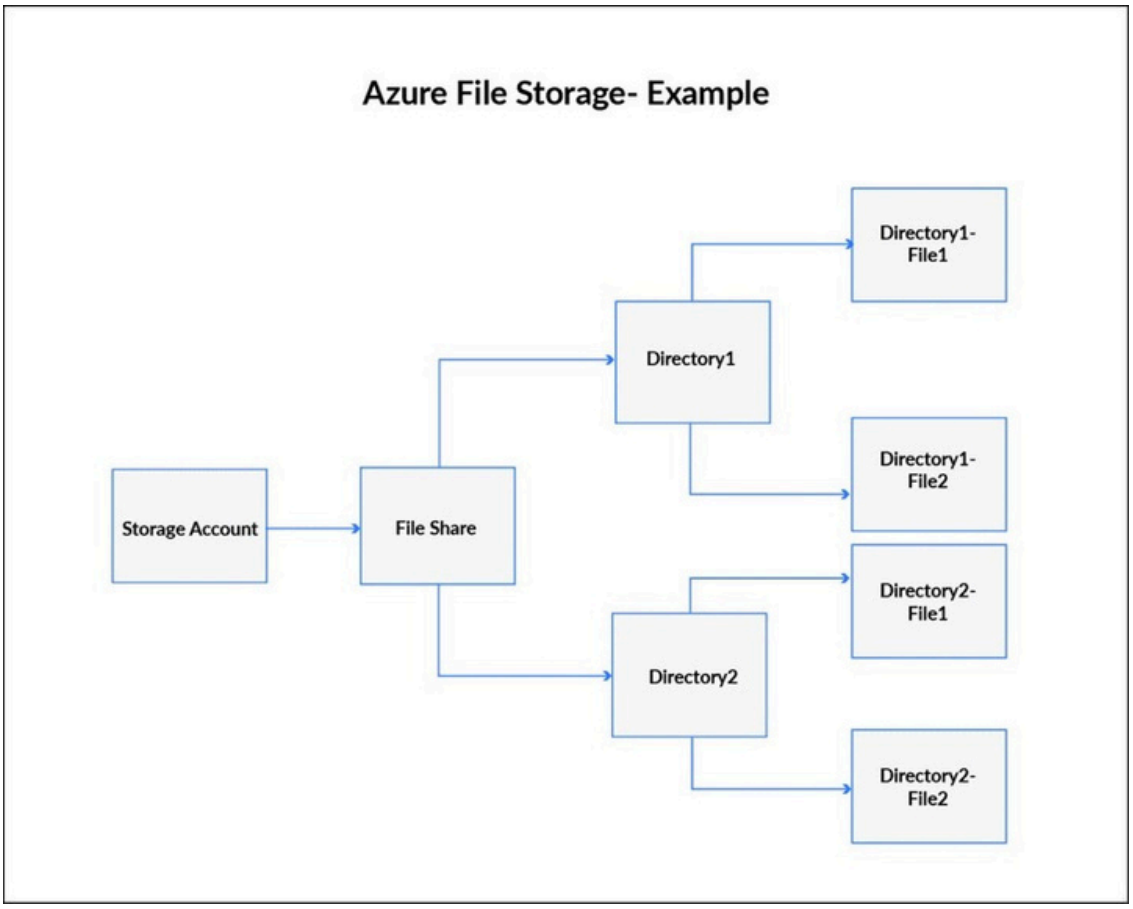
1. **Achieve storage tier:** It is optimized for data that is rarely accessed. Mostly used for archiving or backup data. It is the least expensive service



2. Azure File Storage

Microsoft Azure File storage is a type of Azure service that was designed to support the needs of the Azure VM environment. That storage is, in essence, a network share. You can store files there that can be accessed from different Virtual Machines. It is similar to Amazon EFS and is its direct competitor.

- It allows for the retrieval of files via the server message block protocol
- Using file storage, you can mount file shares on Windows, Linux, and Mac-based machines
- Here you don't need to manage file servers



3. Azure Queue Storage

Queue Storage is a type of storage that is built to connect components of your application. It allows you to build flexible applications with decoupled and independent components that rely on asynchronous message queuing.

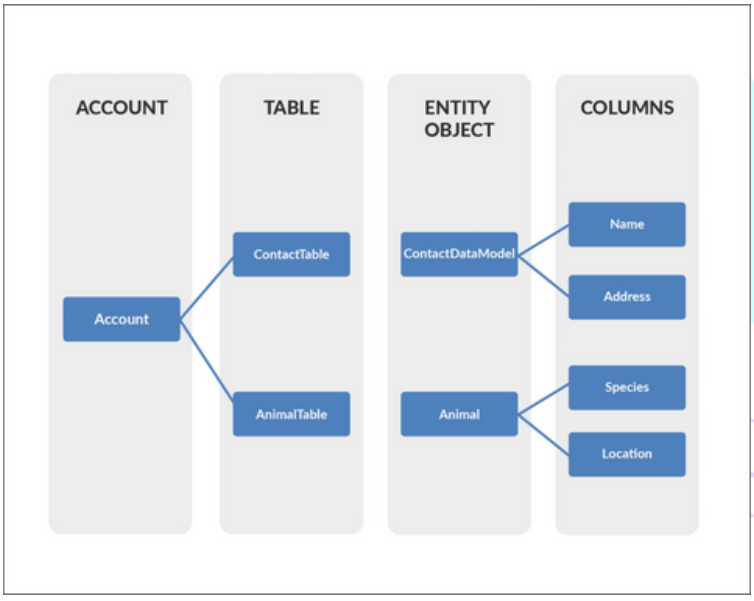
- This service used for the storage and retrieval of messages
- This service is good when you want to decouple components of an application
- A single message in the queue can be up to 64kb in size
- You can store millions of messages in the queue

4. Azure Table Storage

Microsoft Azure Table Storage was made to store structured NoSQL data. The storage is very scalable and, at the same time, very cheap to keep data in. However, it set off more expensive when you access files frequently.

- It is used for storing structured NoSQL data
- It is a key attribute store
- It is a cost-effective option for the storage of table-like data for applications

Instead of using SQL Database to store data, you can use Azure table storage in a more cost-effective manner



Azure Storage Redundancy Options

Azure Storage provides multiple **replication strategies** to ensure **data availability and protection** against failures.

Replication Type	Description	Best Use Case
LRS (Locally Redundant Storage)	Data is stored within a single data center in a region.	Cost-effective but no protection from regional failures.
ZRS (Zone-Redundant Storage)	Data is replicated across multiple availability zones in a region.	High availability within a region.
GRS (Geo-Redundant Storage)	Data is copied to a secondary region for disaster recovery.	Ensures regional failover in case of a disaster.
RA-GRS (Read-Access GRS)	GRS with read-access to the secondary region.	Best for applications requiring failover readiness.
GZRS (Geo-Zone-Redundant Storage)	Combines ZRS and GRS for maximum availability.	Best for critical workloads requiring zone and geo-redundancy.

Real-World Use Cases

1. Storing and Serving Media Content (Blob Storage)

A **video streaming platform** wants to store **high-quality video files** and serve them efficiently to users.

- **Solution:** Store videos in **Azure Blob Storage**, use **Azure CDN** for faster global delivery.
- **Benefit:** High scalability, automatic backups, and low latency for video playback.

2. Lift-and-Shift File Shares to the Cloud (Azure Files)

A company running **on-premises file servers** wants to migrate to the cloud.

- **Solution:** Use **Azure Files** to create **SMB/NFS file shares**, eliminating hardware dependencies.
- **Benefit:** Seamless integration with Windows/Linux systems, reducing IT maintenance costs.

3. Processing IoT Data with Azure Table Storage

An **IoT solution** collects millions of sensor readings per minute.

- **Solution:** Store structured data in **Azure Table Storage** for fast retrieval.
- **Benefit:** NoSQL scalability, low-cost storage, and easy querying.

4. Decoupling Microservices with Azure Queue Storage

An **e-commerce platform** wants to ensure **reliable order processing** without overloading its backend.

- **Solution:** Use **Azure Queue Storage** to manage asynchronous processing between services.
- **Benefit:** Ensures **event-driven architecture**, prevents system failures due to peak loads.

Lab: Creating an Azure Storage Account

Step 1: Create a Storage Account

1. **Go to Azure Portal** → Search for **Storage Accounts** → Click **Create**.
2. Select **Subscription** and **Resource Group**.
3. Enter **Storage Account Name** (must be globally unique).
4. Choose **Region** and **Performance Tier** (Standard or Premium).
5. Select **Redundancy Option** (LRS, ZRS, GRS, etc.).
6. Click **Review + Create** → Wait for Deployment to Complete.

Step 2: Create a Blob Container

1. Open **Storage Account** → Go to **Containers** → Click **+ Container**.
2. Enter **Container Name**, choose **Public or Private Access**.
3. Click **Create**.

Step 3: Upload and Access a Blob File

1. Inside the container, click **Upload** and select a file.
2. Once uploaded, copy the **Blob URL** and open it in a browser.
3. Set **Access Policies** and use **Shared Access Signatures (SAS)** for secure access.

Step 4: Enable Azure Files for Shared Storage

1. Go to **File Shares** → Click **+ File Share** → Set **Quota**.
2. Use **Storage Explorer** to map it as a network drive on Windows/Linux.

Azure SQL Database – Managed Cloud Database Service

Azure SQL Database is a **fully managed, intelligent, and scalable relational database service** built on Microsoft SQL Server technologies. It offers **high availability, security, performance tuning, and automatic scaling** without the need for manual maintenance.

- **Fully Managed (PaaS)** – Microsoft handles updates, backups, and scaling.
- **AI-powered Performance Optimization** – Automatic tuning, index recommendations.
- **Built-in High Availability** – 99.99% SLA uptime.
- **Scalability** – Compute and storage auto-scaling.
- **Advanced Security** – TDE, Always Encrypted, and threat detection.

Key Features of Azure SQL Database

Feature	Description
Fully Managed PaaS	Microsoft manages backups, patching, and scaling.
Intelligent Performance Tuning	AI-driven recommendations for indexing, query optimization.
Automatic Scaling	Adjusts compute power based on demand.
Geo-Replication & High Availability	Built-in disaster recovery across regions.
Security & Compliance	Encryption, firewall rules, private endpoints, and auditing.

Azure SQL Database Deployment Models

Deployment Model	Purpose	UseCases Small business apps, development/testing.
Single Database	Isolated database with dedicated resources.	
Elastic Pool	Multiple databases share resources.	SaaS applications, cost optimization.
Managed Instance	Full SQL Server compatibility.	Enterprise applications needing SQL Server features.

Real-World Use Cases of Azure SQL Database

1. E-commerce Platform – Scalable Database Solution

- **Use Case:** A growing online store needs a **scalable** database to handle seasonal traffic.
- **Solution:** Deploy **Azure SQL Database (Single Database)** with **auto-scaling** enabled.
- **Benefit:** Ensures smooth shopping experience during sales.

2. SaaS Product – Multi-Tenant Database (Elastic Pool)

- **Use Case:** A SaaS startup needs a **multi-tenant** database solution to handle multiple customers efficiently.
- **Solution:** Use **Azure SQL Database Elastic Pool** for cost-effective resource sharing.
- **Benefit:** Each tenant gets a **separate database**, optimizing performance while reducing cost.

3. Enterprise Data Migration – On-Prem SQL to Cloud

- **Use Case:** A financial company wants to migrate their on-prem SQL Server database to the cloud.
- **Solution:** Use **Azure SQL Managed Instance** for **seamless SQL Server migration**.
- **Benefit:** Full SQL Server compatibility with **low downtime**.

Lab: Deploying Azure SQL Database

Step 1: Create an Azure SQL Database

1. **Login to Azure Portal** – https://portal.azure.com
2. **Search for "SQL Database"** and click **+ Create**.
3. Select **Subscription & Resource Group**.
4. Enter **Database Name** (e.g., MySQLDB).
5. Select **Server** → Click **Create New**.
    - Enter **Server Name** (e.g., my-sql-server).
    - Choose **Region** and Authentication Type (SQL/AD).
    - Set **Admin Username & Password**.
6. Choose **Compute + Storage** (e.g., Standard S2 for medium workloads).
7. Click **Review + Create** → Click **Create**.

Step 2: Configure Firewall for Access

1. Navigate to **SQL Database** → **Networking**.
2. Click **Allow Azure services and resources to access this server**.
3. Add **Client IP** to firewall rules for local connection.
4. Click **Save**.

Step 3: Connect to Azure SQL Database using SSMS

1. Open **SQL Server Management Studio (SSMS)**.
2. Click **Connect > Database Engine**.
3. Enter **Server Name** (from Azure portal).
4. Choose **SQL Authentication** → Enter **Username & Password**.
5. Click **Connect**.

✅ **You are now connected to Azure SQL Database!**

Step 4: Create a Table & Insert Data

sql

CopyEdit

```
CREATE TABLE Customers (
   ID INT PRIMARY KEY IDENTITY(1,1),
   Name NVARCHAR(100),
   Email NVARCHAR(100),
   City NVARCHAR(50)
);

INSERT INTO Customers (Name, Email, City)
VALUES ('John Doe', 'john@example.com', 'New York'),
      ('Jane Smith', 'jane@example.com', 'San Francisco');


SELECT * FROM Customers;
```

Step 5: Enable Automatic Performance Tuning

1. Go to **SQL Database → Intelligent Performance**.
2. Enable **Automatic Indexing** and **Query Store**.
3. Click **Save**.

DevOps

DevOps is a combination of **Development (Dev)** and **Operations (Ops)** aimed at automating and integrating software development and IT operations. It enhances collaboration, reduces deployment time, and ensures continuous delivery of high-quality software.

Key Benefits of DevOps

- **Faster Development & Deployment:** Automated pipelines reduce manual effort.
- **Improved Collaboration:** Developers and IT teams work together seamlessly.
- **Continuous Integration & Delivery (CI/CD):** Enables frequent releases.
- **Scalability & Security:** Infrastructure as Code (IaC) ensures efficient resource management.

Traditional IT vs. DevOps:

Feature	Traditional IT	DevOps
Manual Deployments	Yes	Automated
Separate Teams	Yes	Collaborative
Slow Release Cycles	Yes	Faster, Continuous
Less Feedback & Monitoring	Yes	Continuous Feedback
Reactive Maintenance	Yes	Proactive Monitoring

Why DevOps?

DevOps solves common software development problems such as:

- Slow deployment cycles
- Bugs in production due to lack of testing
- Inefficiencies in managing infrastructure
- Lack of collaboration between teams

DevOps Principles

DevOps is based on **5 key principles**, often referred to as **CALMS**:

1. **Culture** – Encourages collaboration between development and operations teams.
2. **Automation** – Uses tools to automate repetitive tasks like testing and deployments.
3. **Lean** – Focuses on eliminating waste and delivering value faster.
4. **Measurement** – Uses metrics to track performance and improvements.
5. **Sharing** – Encourages knowledge sharing across teams.

DevOps Lifecycle:

The DevOps process follows a continuous loop, integrating development, testing, deployment, and monitoring.

1. **Plan** – Agile project planning, backlog management (Azure Boards, Jira).
2. **Develop** – Writing code, using version control systems (Git, Azure Repos).
3. **Build** – Continuous integration (CI) using Jenkins, Azure Pipelines.
4. **Test** – Automated testing to ensure quality (Selenium, JUnit).
5. **Release** – Deploying applications (Terraform, Azure DevOps).
6. **Operate** – Monitoring and managing infrastructure (Azure Monitor, Prometheus).
7. **Monitor** – Gathering logs and feedback for improvements.

Different tools serve different purposes in the DevOps lifecycle. Let's compare **Jenkins, GitHub, and Azure DevOps**.

Feature	Jenkins	GitHub Actions	Azure DevOps
Type	Open-source CI/CD tool	Built-in automation for GitHub	Full DevOps suite by Microsoft
Primary Use	Continuous Integration & Deployment (CI/CD)	CI/CD for GitHub repositories	End-to-end DevOps (Repos, Pipelines, Boards, etc.)
Scalability	Highly customizable but requires plugins	Scales well with GitHub repositories	Enterprise-ready with cloud integration
Integration	Supports multiple tools via plugins	Natively integrates with GitHub	Deep integration with Azure Cloud

Ease of Use	Requires setup & maintenance	Simple YAML-based workflows	UI-based configuration with YAML support
Best Use Case	Organizations needing high flexibility	Developers using GitHub	Enterprises using Azure

Frequently Used DevOps Tools:

In DevOps, tools play a crucial role in automating workflows. Below are the **most widely used DevOps tools** in different categories.

Version Control Systems (VCS)

Version Control Systems (VCS) help teams track changes in code.

- **Git** – Most popular distributed version control system.
- **Azure Repos** – Microsoft's cloud-based Git repository.
- **GitHub** – A popular cloud-hosted Git repository.
- **GitLab & Bitbucket** – Alternative Git hosting platforms.

Real-World Example

A software company develops a new feature. Developers push code to **GitHub**, where automated CI/CD pipelines test and deploy it.

Continuous Integration (CI) Tools

CI tools help integrate code changes frequently, reducing conflicts.

- **Jenkins** – Open-source automation tool for CI/CD.
- **Azure DevOps Pipelines** – Microsoft's CI/CD automation tool.
- **GitHub Actions** – CI/CD integrated with GitHub.
- **GitLab CI/CD** – Provides a built-in pipeline feature.

Real-World Example

A developer pushes code to a Git repository. **Jenkins** automatically builds and tests the application. If successful, it deploys the app to a test server.

2.3 Continuous Deployment (CD) Tools

CD tools help automate software releases.

- **Azure DevOps Pipelines** – Supports automated deployments.
- **Spinnaker** – Open-source CD tool from Netflix.
- **ArgoCD** – Kubernetes-native GitOps tool.

Real-World Example

A DevOps team uses **Azure Pipelines** to deploy an app to Azure Kubernetes Service (AKS) whenever a change is pushed to the repo.

Infrastructure as Code (IaC) Tools

IaC automates infrastructure provisioning and configuration.

- **Terraform** – Cloud-agnostic infrastructure automation.
- **Bicep** – Microsoft's IaC tool for Azure.
- **Ansible** – Automates configuration management.

Real-World Example

A DevOps engineer writes a **Terraform** script to create multiple **Azure VMs** and deploy applications automatically.

Containerization & Orchestration

Containers package applications with their dependencies, and orchestration tools manage them.

- **Docker** – Containerizes applications.
- **Kubernetes (K8s)** – Manages containerized applications.
- **Azure Kubernetes Service (AKS)** – Microsoft's managed Kubernetes service.

Real-World Example

A company runs its microservices in **Docker** containers. They use **AKS** to deploy and scale their application across multiple nodes.

Monitoring & Logging Tools

Monitoring tools track performance, while logging tools collect system logs.

- **Azure Monitor & Log Analytics** – Monitors Azure workloads.
- **Prometheus & Grafana** – Collects and visualizes metrics.
- **ELK Stack (Elasticsearch, Logstash, Kibana)** – Centralized logging solution.

Real-World Example

An e-commerce company uses **Azure Monitor** to detect CPU spikes on its web servers and scales up instances automatically.

Security & Compliance Tools

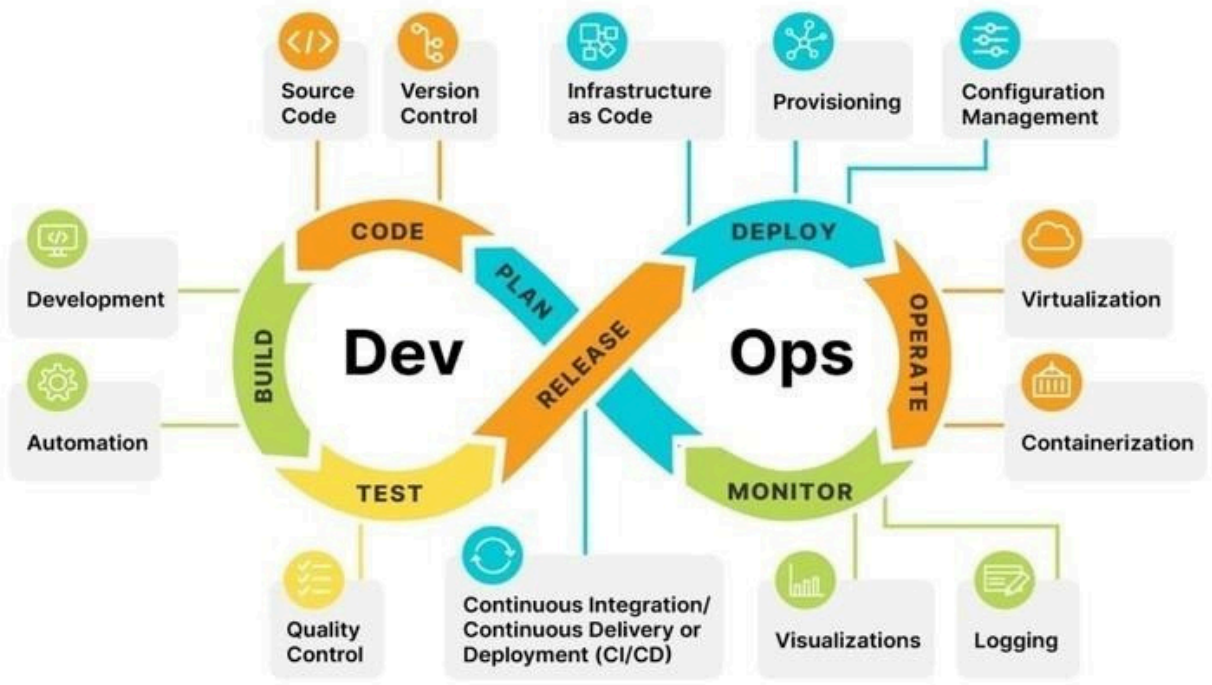
Security tools ensure compliance and protect applications.

- **Azure Security Center & Defender for Cloud** – Cloud security tools.
- **HashiCorp Vault** – Manages secrets securely.

Real-World Example

A bank stores API keys and passwords in **HashiCorp Vault** to prevent security leaks.

DevOps Architecture:



Key Components of DevOps Architecture

FollowingarethekeycomponentsofDevOps Architecture −

Continuous Integration (CI) − CI is a practice where we automatically combine code changes from many contributors into a shared place several times a day. This process helps us find and fix problems early. It makes our software better.

Continuous Delivery (CD) − CD builds on CI. It makes sure that the combined code is always ready to be used. This way, we can release new features and fixes quickly and reliably. It helps us get fast feedback and improve.

Infrastructure as Code (IaC) − IaC is about managing and setting up our infrastructure through code. We dont do it by hand. This way, we can set up our infrastructure automatically, consistently, and repeatedly. It reduces mistakes and speeds up the process.

Microservices Architecture − This design means breaking our applications into smaller services. These services are not tightly connected. We can develop, deploy, and scale them on their own. Microservices give us more flexibility. They let us work on different parts without stopping the whole system. **Automation Tools** − Automation is very important in DevOps architecture. We use
tools like Jenkins, Ansible, and Terraform to automate tasks. These include testing, deployment, and managing setups. This lets our teams focus on more important work. Monitoring and Logging − We need to monitor our applications and infrastructure all the time. This is key to keeping our performance and reliability. Monitoring tools gather data and metrics. Logging tools collect detailed event information. This helps us find and fix issues quickly. Collaboration and Communication − Good teamwork between development and operations is key for DevOps architecture to work well. We use tools like Slack, Jira, and Confluence to help us communicate and manage projects. This way, everyone stays aligned and informed during the development process. **Security (DevSecOps**) − We add security into the DevOps process through DevSecOps. This makes sure security steps are part of every stage of development. This way, we can find problems early and stay compliant. We do this without slowing down our delivery.

Git

Git is a **distributed version controlsystem (VCS)** that allows multiple developers to collaborate on aproject efficiently. It tracks changesinfiles and enables rollbacks if needed.

Why Use Git?

- Version Control – Keeps track of every change.
- Collaboration – Multiple developers can work on the same project without conflicts.
- Branching & Merging – Developers can work on different features separately and merge them later.
- Backup & Recovery – Ensures code is never lost.

Git Workflow

1. **Clone** – Download an existing repository.
2. **Create & Switch Branch** – Work on a separate feature branch.
3. **Stage Changes** – Add modified files to the staging area.
4. **Commit Changes** – Save changes with a message.
5. **Push Changes** – Upload changes to the remote repository.
6. **Pull Changes** – Sync with the latest code from the repository.
7. **Merge** – Combine code from different branches.

Frequently Used Git Commands

1. Configuring Git

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
git config --list # Check configured settings
```

2. Initializing & Cloning a Repository

```
git init # Initialize a new repository
git clone <repository_url> # Clone an existing repository
```

3. Checking Status & Logs

```
git status # Check the status of your working directory
git log # View commit history
git log --oneline --graph # Compact commit history with a graph
```

4. Staging & Committing Changes

```
git add <file> # Stage a specific file
git add . # Stage all files
git commit -m "Commit message" # Commit staged changes
git commit --amend -m "Updated message" # Edit last commit message
```

5. Working with Branches

```
git branch # List all branches
```

git branch <new-branch> # Create a new branch
git checkout <branch-name> # Switch to a branch
git checkout -b <new-branch> # Create and switch to a new branch
git merge <branch-name> # Merge a branch into the current branch

6. Pushing & Pulling Changes

git push origin <branch-name> # Push changes to a remote branch
git pull origin <branch-name> # Pull the latest changes from the remote

7. Undoing Changes

git reset --soft HEAD~1 # Undo last commit but keep changes staged
git reset --hard HEAD~1 # Undo last commit and delete changes
git revert <commit-hash> # Create a new commit that reverses a previous commit

AzureDevOps:

Azure DevOps is a complete suite of tools offered by **Microsoft** to help organizations implement DevOps practices efficiently. It provides services for **source code management, continuous integration (CI), continuous delivery (CD), infrastructure automation, monitoring, and security**.

Key Components of Azure DevOps

Azure DevOps consists of **five core services**, each designed to support different stages of the DevOps lifecycle:

Component	Purpose	Key Features
Azure Repos	Version Control System	Git repositories, Branching & Merging, Pull Requests
Azure Pipelines	Continuous Integration & Deployment	Build automation, CI/CD pipelines, Multi-stage releases
Azure Boards	Agile Project Management	Work tracking, Kanban boards, Sprint planning
Azure Test Plans	Automated & Manual Testing	Test cases, Load testing, Exploratory testing

Azure Artifacts	Package Management	Secure storage for NuGet, npm, Maven, and Python packages

How Azure DevOps Works in a Real-World Scenario

Let's say a **software company** is developing a new web application. Here's how Azure DevOps helps streamline the process:

1. **Code Management (Azure Repos)**
   - Developers write code and push it to **Azure Repos (Git-based version control)**.
   - Branching strategies ensure safe feature development.
2. **CI/CD Automation (Azure Pipelines)**
   - Every code change is **automatically built and tested** using **CI/CD pipelines**.
   - If the build passes, it is **automatically deployed** to the staging environment.
3. **Agile Project Tracking (Azure Boards)**
   - Teams use **Azure Boards** to track tasks, sprints, and bugs.
   - Product managers assign user stories to developers.
4. **Testing & Quality Assurance (Azure Test Plans)**
   - QA teams use **Azure Test Plans** to create test cases and run automation tests.
   - Bugs are tracked and fixed before production release.
5. **Secure Package Management (Azure Artifacts)**
   - Developers use **Azure Artifacts** to store and share **pre-built packages** securely.

Security Best Practices in Azure DevOps

Security is a critical aspect of DevOps, ensuring that applications and infrastructure remain protected throughout the software development lifecycle (SDLC). **Azure DevOps** provides multiple built-in security features to enforce **access control, compliance, and secure coding practices**.

1. Identity and Access Management (IAM)

Best Practice: Implement **Role-Based Access Control (RBAC)** to restrict access based on the principle of **least privilege**.

- Use **Azure AD (Active Directory) integration** for centralized authentication.
- Assign **roles** at the **organization, project, and repository level** in Azure DevOps.
- Use **Personal Access Tokens (PATs)** for automation instead of using passwords.
- Enforce **Multi-Factor Authentication (MFA)** for all users.

Example Use Case:

A **DevOps Engineer** should have access to **pipelines** but not modify **security settings**. RBAC ensures that permissions are restricted to their job role.

2. Secure Code Management with Azure Repos

Best Practice: Protect source code from unauthorized access and enforce security policies.

- Enable **Branch Policies** (e.g., Require Pull Request (PR) reviews before merging).
- Use **Code Scanning Tools** like **SonarCloud** for security vulnerabilities.
- Configure **Git repository permissions** to prevent accidental code deletion.
- Use **Signed Commits** to verify code authenticity.

Example Use Case:

A development team ensures that **every commit** is scanned for vulnerabilities before merging into the **main branch**.

3. Secure CI/CD Pipelines in Azure Pipelines

Best Practice: Prevent unauthorized code deployment and enforce compliance.

- **Use Service Connections securely**: Store secrets in **Azure Key Vault**, not in pipeline YAML files.
- Implement **Pipeline Approvals**: Use manual approval gates before production deployment.
- Use **Agent Pools**: Restrict pipelines to run only on **trusted** agents.
- Enable **artifact signing**: Ensure that only verified builds are deployed.
- Scan **container images and dependencies** for vulnerabilities before deployment.

Example Use Case:
A **Finance App Deployment** pipeline requires **manual approval** before deploying updates to production.

4. Secure Package Management with Azure Artifacts

Best Practice: Store and manage dependencies securely.

- **Enable scoped feeds** to prevent external users from accessing internal packages.
- Scan dependencies for vulnerabilities using **Microsoft Defender for DevOps**.
- Use **Immutable Artifacts** to prevent tampering.

Example Use Case:
A **Node.js application** team stores **private npm packages** in **Azure Artifacts**, ensuring secure package distribution.

---

5. Secure Testing with Azure Test Plans

Best Practice: Identify security flaws early in the development lifecycle.

- Implement **penetration testing** on application builds.
- Use **load testing** to check for **DDoS vulnerabilities**.
- Perform **security compliance checks** using **Microsoft Defender for DevOps**.

Example Use Case:
Before a **government application** is deployed, it must pass **security compliance tests** to meet industry regulations.

6. Auditing & Compliance in Azure DevOps

Best Practice: Continuously monitor and log all DevOps activities.

- Enable **Azure Monitor & Log Analytics** to track **pipeline executions and deployments**.
- Configure **Audit Logs** for tracking repository access and code changes.

- Set up **Microsoft Defender for Cloud** to detect security misconfigurations.
- Use **Azure Policy** to enforce compliance rules (e.g., restrict external repo access).

Example Use Case:
A **healthcare company** must comply with **HIPAA regulations**, so all DevOps activities are logged and monitored for security compliance.

Lab 1: Implement Role-Based Access Control (RBAC) in Azure DevOps

Step 1: Configure Project-Level Security

1. **Go to Azure DevOps** (https://dev.azure.com).
2. Select your **Organization** and open a **Project**.
3. Navigate to **Project Settings** > **Permissions**.
4. Click **Add a Member** and assign one of the following roles:
    - **Project Reader:** View-only access.
    - **Contributor:** Can push code and create pipelines.
    - **Project Admin:** Full project control.
5. Click **Save** and verify access by logging in with the assigned user.

Expected Outcome: The user can only perform **actions** based on the **assigned role**.

Lab 2: Secure CI/CD Pipelines in Azure DevOps

Step 1: Enable Secure Pipeline Approvals

1. **Go to Pipelines** > **Select a pipeline**.
2. Click **Edit** and go to **Environments**.
3. Click **Pre-deployment Approvals** and enable **manual approvals**.
4. Assign an **Approver (e.g., Manager or Security Engineer)**.
5. Click **Save**.

Expected Outcome: Deployments will be paused until an authorized approver confirms the release.

Step 2: Secure Secrets with Azure Key Vault

1. Go to **Azure Portal** > **Create a Key Vault**.
2. Add a **new secret** (e.g., SQL_PASSWORD).
3. In **Azure DevOps**, go to **Library** > **Secure Files**.
4. Link **Azure Key Vault** to DevOps pipelines.
5. In the pipeline YAML, replace secrets with:

```
variables:
 sql Password: $(sql_password)
```

Expected Outcome: Secrets will be securely fetched from **Azure Key Vault** instead of being hardcoded in YAML files.

Lab 3: Secure Package Management with Azure Artifacts

Step 1: Configure Scoped Feeds

1. **Go to Azure DevOps** > **Artifacts**.
2. Click **Create a Feed** and select **Private** visibility.
3. Upload a sample **NuGet/NPM package**.
4. Go to **Feed Settings** > **Permissions**.
5. Assign only **specific users** access to the package feed.

Expected Outcome: Only authorized users can download the package, securing package distribution.

Lab 4: Audit & Compliance in Azure DevOps

Step 1: Enable Audit Logs

1. **Go to Organization Settings** > **Security**.
2. Navigate to **Auditing** and click **Enable Audit Logs**.
3. Monitor **who accessed repositories, modified pipelines, or triggered builds**.

Expected Outcome: All security events will be **logged** for compliance tracking.

Introduction to YAML Pipeline

- YAML: Yet Another Markup Language
- The pipeline is versioned with your code and follows the same branching structure. You get validation of your changes through code reviews in pull requests and branch build policies. ● Every branch you use can modify the build policy by modifying the azure-pipelines.yml file. ● A change to the build process might cause a break or result in an unexpected outcome. Because the change is in version control with the rest of your codebase, you can more easily identify the issue.

Continuous Integration using YAML Pipelines

- Introduction to YAML Pipeline
- Building Azure DevOps Pipeline using YAML
- Publishing results to Artifacts
- Triggering Continuous Integration in YAML
- Filtering Tasks based on branch being built
- Using Templates to Build Multiple Configurations
- Build on Multi-Platform pipeline

Mini Project: Deploy a Web Application using Azure DevOps & Azure Resources

In this mini-project, we will deploy a **sample web application** to an **Azure Web App** using **Azure DevOps CI/CD pipelines**.

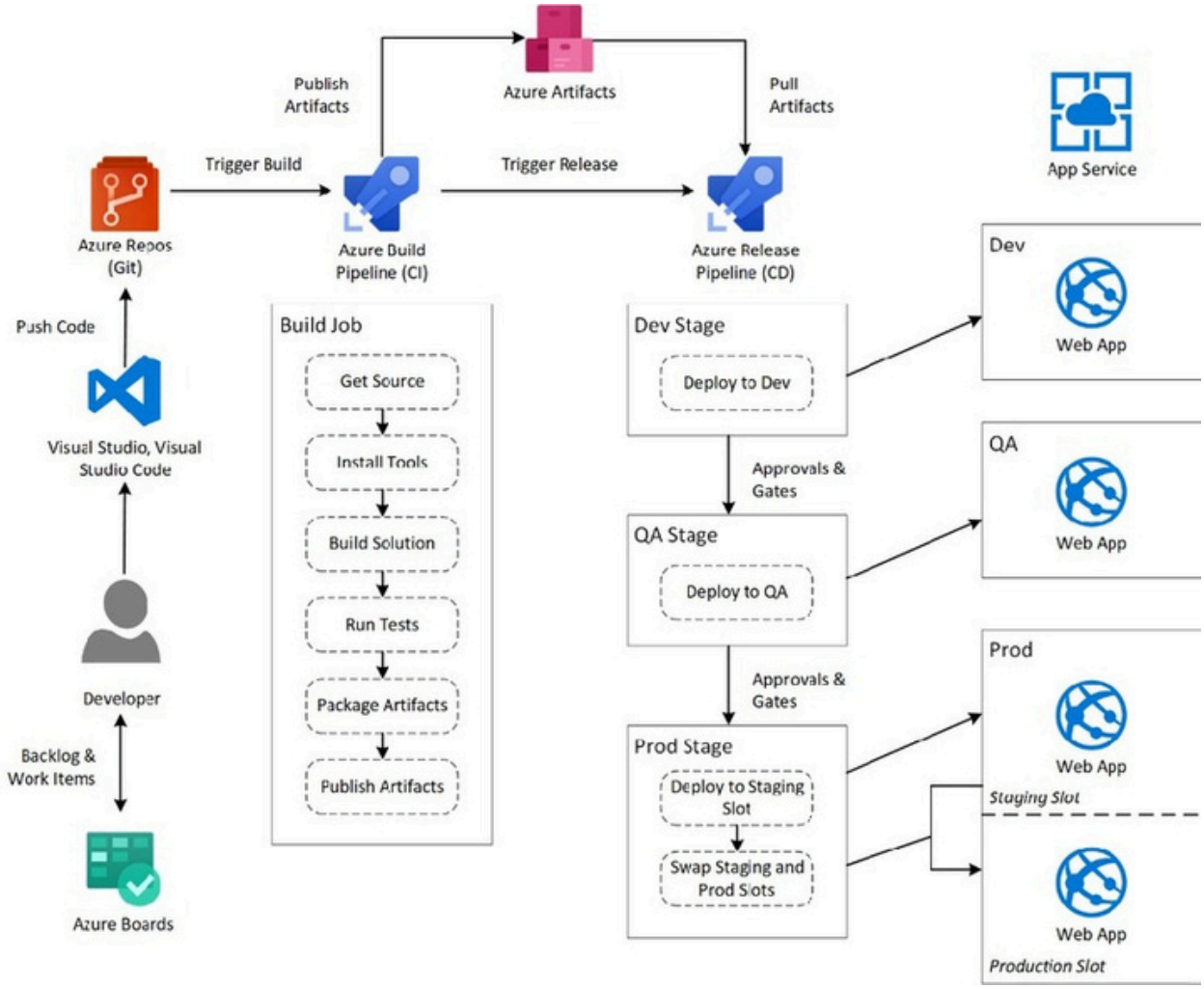
Prerequisites

✓ **Azure Subscription** (for creating Azure resources)

✓ **Azure DevOps Account** (for setting up CI/CD pipeline)

✓ **Basic knowledge of Git, Azure Web Apps, and DevOps**

High Level- Architecture Diagram:



Step1: Create a Web App in Azure Portal

- o Login to Azure Portal, https://portal.azure.com/
- o Azure Portal More Services Web App + Add
- o Select Web Apps Create
- o Name = "DssDemoWebApp", Subscription = "Free Trail" Resource Group="DemoRG", (Name=Standard Plan, App Service plan/Location=Create New Plan Location=Central US, Pricing tier=S1 Standard.
- o Application Insights=Off

- Create

Follow the above steps and create the below AppServices

- 1. Create an App Service = **DssDemoApp-Dev**
- 2. Create an App Service = **DssDemoApp-QA**
- 3. Create an App Service = **DssDemoApp / Slot = Staging**

Expected Outcome: A Web App (mywebapp-demo.azurewebsites.net) is ready.

Step2: How does Azure Pipelines connect to Azure?

To deploy your app to an Azure resource, such as a virtual machine or App Service, you need a *service connection*.

Create an Azure Resource Manager service connection Manually

Create Service Principal and assign Contributor role to it for a subscription.

1. Login to Azure Portal as a Global Administrator and Subscription Owner.

2. To Create a Service Principal: Azure Active Directory App Registrations **+ New registration**
   a. Name = **AzureDevOpsServicePrincipal.**
   b. Select Accounts in this organizational directory only (Microsoft)
   c. Register.

3. The Service Principal should be assigned owner rights for the Subscription
   a. Seach and goto Azure Subscription
   b. Access Control (IAM) Add role Assignment
   c. Role = **Contributor**, Select Service Principal (**AzureDevOpsServicePrincipal**) Assgin.

4. Copy Application (Client) ID and Secret
   a. To go AzureDevOpsServicePrincipal Certificates and secrets + New Client secret Copy the secret
   b. Overview Tab and Copy Application ID and Tenant ID

5. Go to Subscription and Copy Subscription ID and Name.

Client ID = 9cccb41d-a0fa-4304-9ea9-999402037270

Secret = Az_aWFo~fwi-91evSos-eu5qJK1.M2Cs.e

Tenant ID = 3217245f-90ec-4ef7-b6c1-909be73fa1eb

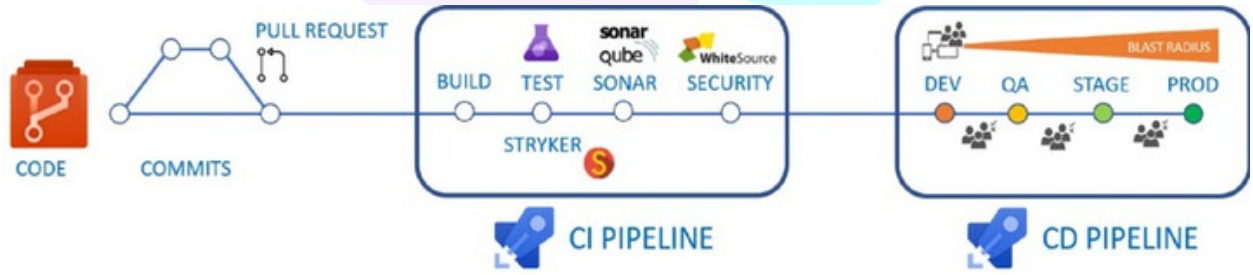
Subscription ID = 51081bf2-da0d-4998-9462-b59b512f8690

Subscription Name = Visual Studio Enterprise – SS2

Step3:Create a DevOps Service Connection to Azure Subscription:

- **Switch to Azure DevOps** Project Settings (Pipeline Section) **Service Connections** Create a Service Connection **Azure Resource Manager** Next
- Service Principal (Manual) Next
- Provide the values copied earlier.
- Verify and Create a Service Connection.

Azure DevOps Pipeline



Push the Sample Application to Repo

- DevOps Portal Create a Project (HelloWorldApp)
- Project => **Repo** => Files => Copy Clone URL to Clipboard
- D:\DevOpsDemos>git clone <Paste the URL>
- D:\DevOpsDemos>dotnet new sln -o HelloWorldApp
- D:\DevOpsDemos> cd HelloWorldApp

- D:\DevOpsDemos\HelloWorldApp> dotnet new mvc -n HelloWorldApp.Web
- D:\DevOpsDemos\HelloWorldApp> dotnet sln HelloWorldApp.sln add HelloWorldApp.Web\HelloWorldApp.Web.csproj
- git add .
- git commit -m "Initial Commit"
- git push origin master
- Switch back to the web portal and View the Committed Files.

Continuous Integration using Azure Build Pipelines

- About Azure DevOps Pipeline
- Understanding the Build Process
- Create a Pipeline using Classic Editor
- Enable Continuous Triggers for Build Pipeline
- Add a status badge to Repository
- Working with Task Groups
- Validate Pull Request based on Build Pipeline result
- Add a Widget to Dashboard

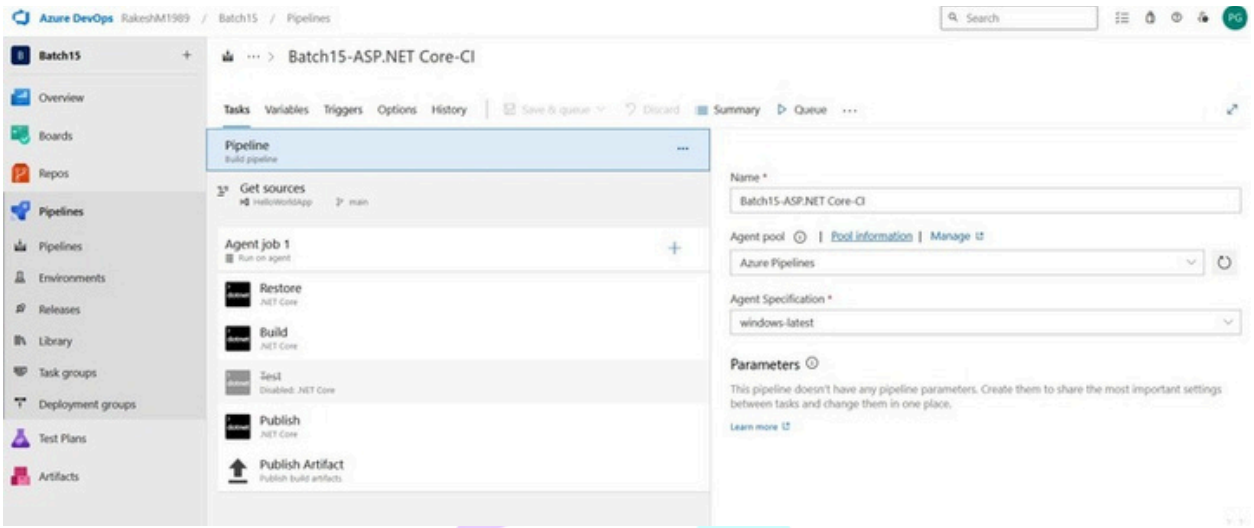
Step4: Create a Pipeline using Classic Editor

- Pipeline New Pipeline **Use the classic editor**
- Select your Source Code Repository and Branch Continue

- Select a Template: Select **ASP.NET Core** Apply
- This generates all the Tasks in a Job.
- (Optional) Add Task: **File Creator** to create a file
- **File path** = ./HelloWorldApp.Web/wwwroot/buildinfo.txt,
- **Content** = "$(Build.DefinitionName), $(Build.BuildId), $(Build.BuildNumber)"
- (Optional) Add a Task: **Command line** Add

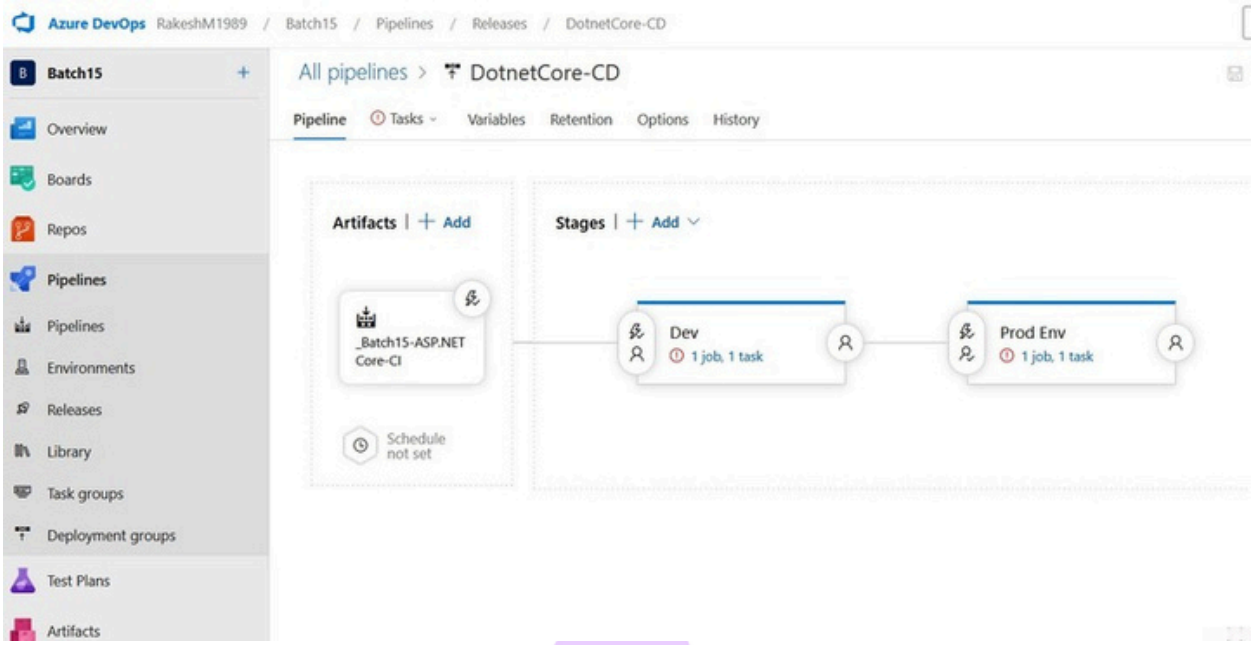
- **Script** = echo "$(Build.DefinitionName), $(Build.BuildId), $(Build.BuildNumber)"
- Click on **Save and queue**

Validation: Once the Pipelines completes the task – Check the Artifacts or check for the error.



Step5: Create a release to deploy your app

- Pipelines => Releases => New => New release pipeline => Select Template Azure App Service Deployment => Apply Change Stage name = Dev Stage View Stage tasks: Click on 1 job, task, Select Azure Subscription =
- <Subscription created before>, App type = Web App on Linux, App service name = Dssdemoapp Check
- Deploy to App Service, App Service Name = Dev



Create a release to deploy your app

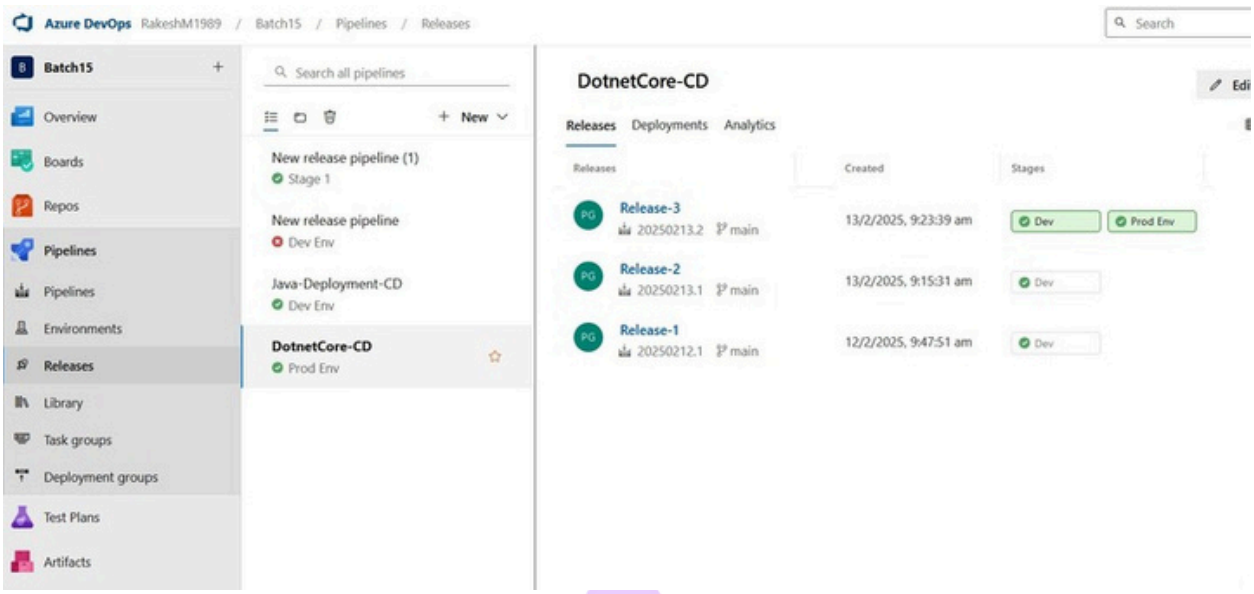
- Choose + Release and select Create a release.
- In the Create a new release panel, check that the artifact version you want to use is selected and choose Create.
- Choose the release link in the information bar message. For example: "Release Release-1 has been created".
- In the pipeline view, choose the status link in the stages of the pipeline to see the logs and agent output.
- After the release is complete, navigate to your app and verify its contents.

Step6: Adding Another Stage (Production)

- Edit the Release Pipeline
- Create a copy of existing Dev stage
- Change Name to QA Stage => Go to Tasks tab => Check Deploy to App Service = QA

Create a release to deploy your app.

- Pipelines => Releases => Select the Pipeline => Click Create release Create
- Choose the release link in the information bar message. For example: "Release Release-2 has been created".
- After the release is complete, navigate to your app and verify its contents.


Azure DevOps RakeshM1989 / Batch15 / Pipelines / Releases
Search
Batch15
Search all pipelines
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Releases
Library
Task groups
Deployment groups
Test Plans
Artifacts
New
New release pipeline (1)
Stage 1
New release pipeline
Dev Env
Java-Deployment-CD
Dev Env
DotnetCore-CD
Prod Env
DotnetCore-CD
Releases
Deployments
Analytics
Releases
Created
Stages
Release-3
20250213.2 main
13/2/2025, 9:23:39 am
Dev
Prod Env
Release-2
20250213.1 main
13/2/2025, 9:15:31 am
Dev
Release-1
20250212.1 main
12/2/2025, 9:47:51 am
Dev

Fundamentals of Docker & AKS (Azure Kubernetes Service):

Orchestration tools like **Kubernetes** (K8s) and **Azure Kubernetes Service (AKS)** have become increasingly popular for managing containerized applications. Here are the main reasons why organizations are migrating to orchestration platforms:

Docker Fundamentals:

Docker is a **containerization platform** that enables developers to package applications and their dependencies into **lightweight, portable containers**.

Key Benefits of Docker:

✅ **Portability** – Run containers on any system with Docker installed.

✅ **Isolation** – Each container runs independently with its own libraries and dependencies.

✅ **Scalability** – Easily scale applications across multiple containers.

✅ **Fast Deployment** – Containers start up quickly compared to VMs.

Docker Setup and Installation:

Download and Install the Docker Desktop using below link

https://www.docker.com/get-started

Basic Docker Commands

1. Pull an Image from Docker Hub

docker pull nginx

This pulls the **nginx** web server image.

2. Run a Docker Container

docker run -d -p 8080:80 nginx

This runs an **nginx** web server in a container and maps port **8080** of your system to port **80** of the container.

3.View Running Containers

docker ps

4.Stop and Remove Containers

docker stop <container_id>
docker rm <container_id>

5.Create a Custom Docker Image (Dockerfile Example)

1. Create a file named **Dockerfile**:

FROM nginx
COPY index.html /usr/share/nginx/html
EXPOSE 80

2. Build the Docker image:

docker build -t my-nginx-app .

3. Run the container:

docker run -d -p 8080:80 my-nginx-app

Azure Kubernetes Service (AKS)

Kubernetes (K8s) is an **orchestration tool** that automates container deployment, scaling, and management.

Why Use Kubernetes?

✅ **Automatic Scaling** – Scale applications based on demand.

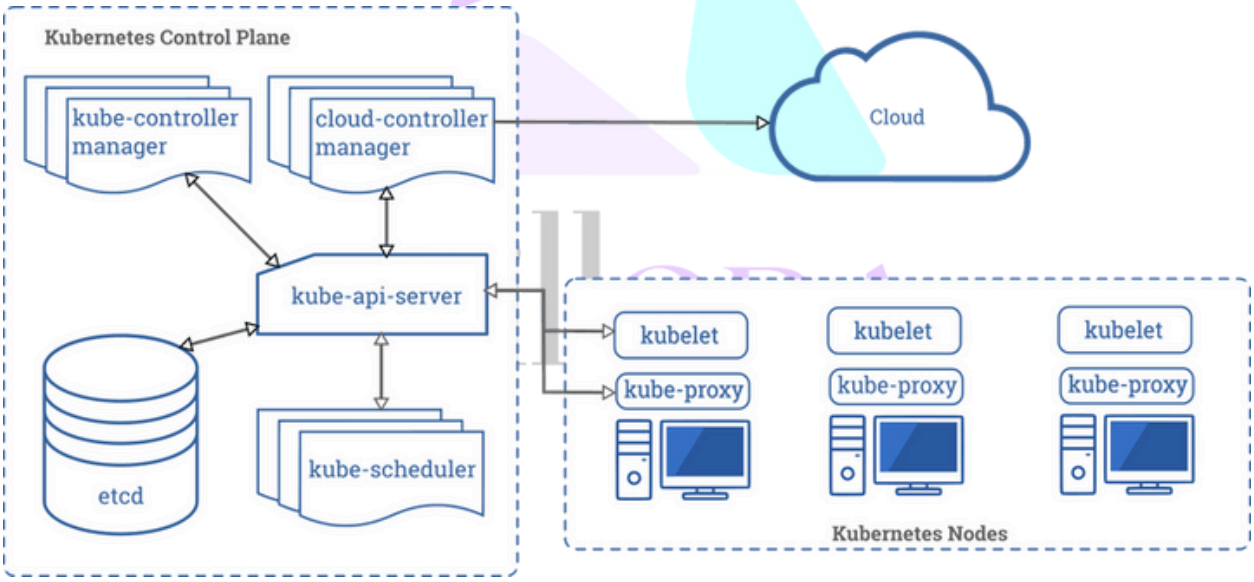
✅ **Self-Healing** – Restart failed containers automatically.

✅ **Load Balancing** – Distributes traffic across multiple instances.

✅ **Rolling Updates** – Deploy updates with zero downtime.

AKS (Azure Kubernetes Service) is a **managed Kubernetes service** in Azure, which simplifies deployment and scaling of containerized applications.

AKS Architecture Overview:



Kubernetes Master Node In Kubernetes (k8s), a master node is the **control plane component** responsible for **managing the cluster**. It coordinates and schedules tasks, maintains cluster state, and monitors node health. It

includes components like API server, scheduler, and controller manager, ensuring overall cluster functionality and orchestration of containerized applications.

Master Node Components:

1) Kube API server handles administrative tasks on the master node. Users send REST commands in YAML/JSON to the API server, which processes and executes them. The Kube API server acts as the front end of the Kubernetes control plane.

2) etcd, a distributed key-value store, maintains the cluster state and configuration details like subnets and config maps in Kubernetes' database. It's where Kubernetes stores its information.
Read More: Kubernetes for Testers.

3) Kube-scheduler assigns tasks to worker nodes and manages new requests from the API Server, ensuring they are directed to healthy nodes.

4) Kube Controller Manager task is to retrieve the desired state from the API Server. If the desired state does not match the current state of the object, corrective steps are taken by the control loop to align the current state with the desired state. There are different types of control manager in Kubernetes architecture:

- **Node Manager:** It oversees nodes, creating new ones in case of unavailability or destruction.
- **Replication Controller:** It ensures the desired container count is maintained within the replication group.
- **Endpoints Controller:** This controller populates the endpoints object, connecting Services & Pods.

Kubernetes Worker Node

Worker nodes in a cluster are machines or servers running applications, controlled by the Kubernetes master. Multiple nodes connect to the master. On each node, multiple pods and containers operate.

Components of Worker Nodes:

1) Kubelet, an agent on each node, communicates with the master. It ensures pod containers' health, executing tasks like deploying or destroying containers, reporting back to the Master.

2) Kube-proxy enables worker node communication, managing network rules. It ensures rules are set for containers to communicate across nodes.

3) A **Kubernetes pod** is a set of containers on a single host, sharing storage and network. It includes specifications for container execution, enabling easy inter-container communication.

4) Container Runtime, responsible for container execution, supports multiple runtimes: Docker, containers.

AKS Core Components

1. **Containers**
    - A **lightweight, portable unit** containing an application and its dependencies.
    - Runs inside a **Pod** in Kubernetes.
    - Uses Docker or any container runtime (e.g., containerd).
2. **Pods**
    - The **smallest and simplest Kubernetes object**.
    - Can contain **one or multiple** containers.
    - All containers within a pod share **networking and storage**.
    - Pods can be **scaled up/down** automatically.
3. **Nodes (Worker Nodes)**
    - Virtual machines (VMs) where pods run.
    - Managed by Azure (via AKS).
    - Each node has **Kubelet** (agent to communicate with Kubernetes).
4. **Deployments**
    - Defines **how applications are deployed** in AKS.
    - Helps in **rolling updates and rollbacks**.
5. **ReplicaSets**
    - Ensures that a specific number of **pod replicas** are running.

- Provides **auto-scaling capabilities**.

6. **Kubernetes Services**
   - **Exposes** applications to internal or external users.
   - Common service types:
   - **ClusterIP** – Internal communication (within AKS cluster).
   - **NodePort** – Exposes service via node IP.
   - **LoadBalancer** – Provides external access via Azure Load Balancer.
   - **Ingress Controller** – Acts as a traffic router (Nginx, Traefik).

Mini Project – Deploying an Application to AKS

We will deploy a **sample Nginx web application** in **two different approaches**:

1. **Manual Deployment using kubectl**
2. **Automated CI/CD Deployment using Azure DevOps Pipelines**

Prerequisites

Before starting, ensure you have:

✅ An **Azure subscription**

✅ An **AKS cluster** (already created)

✅ **kubectl** installed

✅ **Azure CLI** installed

✅ **Azure DevOps account**

Approach 1: Manual Deployment using kubectl

Step 1: Create a Kubernetes Deployment YAML

Create a file named nginx-deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
```

```
    app: nginx
  spec:
   containers:
   -name: nginx
    image: nginx:latest
    ports:
    -containerPort: 80
```

Step2: Deploy the Application to AKS

Runthe following command to deploy:

kubectl apply -f nginx-deployment.yaml

Step 3: Expose the Deployment as a Service

Create a file named nginx-service.yaml:

```
apiVersion: v1
kind: Service
metadata:
 name: nginx-service
spec:
 type: LoadBalancer
 selector:
  app: nginx
 ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

Apply the service:

kubectl apply -f nginx-service.yaml

Step 4: Get the External IP

Run:

kubectl get svc nginx-service

Validation: **Once you get the** External IP**, open it in a browser to see the Nginx page**

Approach 2: Deploy Spring Boot Application, Automated CI/CD using Azure DevOps Pipelines

Prerequisites:

- **Azure Subscription**
- **Azure DevOps Account**
- **AKS Cluster**
- **Azure Container Registry (ACR)**
- **Sample Spring Boot Application**
- Create a Name Space: springboot-app
- Create a RBACK with below code:

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: sprint-boot-frontend
  name: service-reader
rules:
- apiGroups: [""] # "" indicates the core API group
  resources: ["services"]
  verbs:["get", "watch", "list"]
```

Pipeline Flow

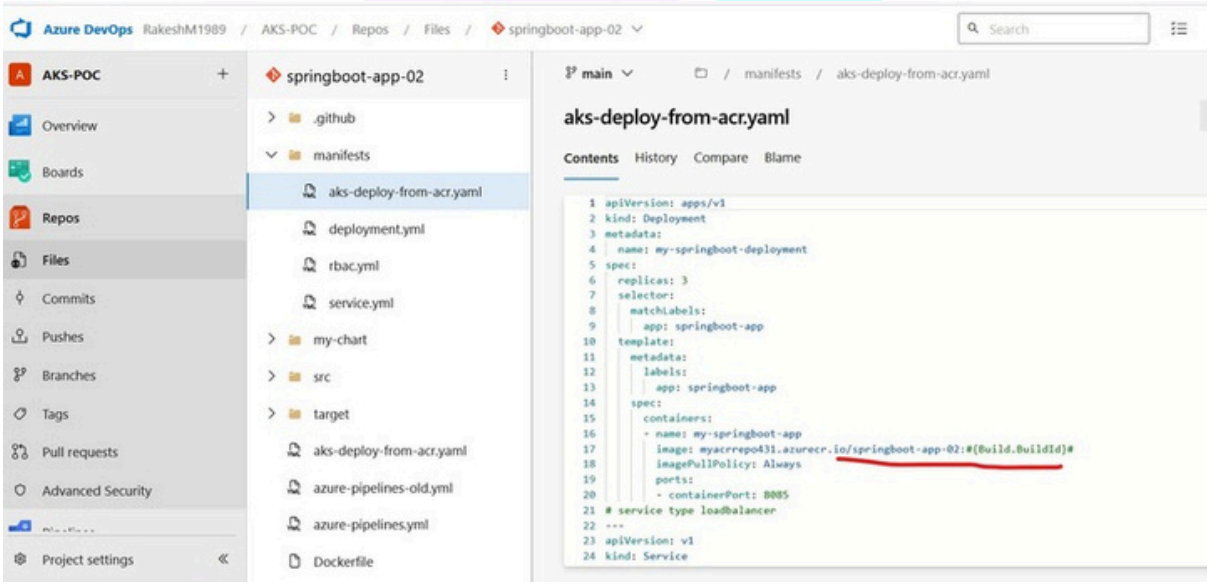
1. **CI - Continuous Integration**
    - Build the Spring Boot Application (Maven)
    - Create a Docker image
    - Push the Docker image to Azure Container Registry (ACR)
2. **CD - Continuous Deployment**
    - Deploy the application from ACR to AKS using kubectl
    - Expose the application using a Kubernetes Service

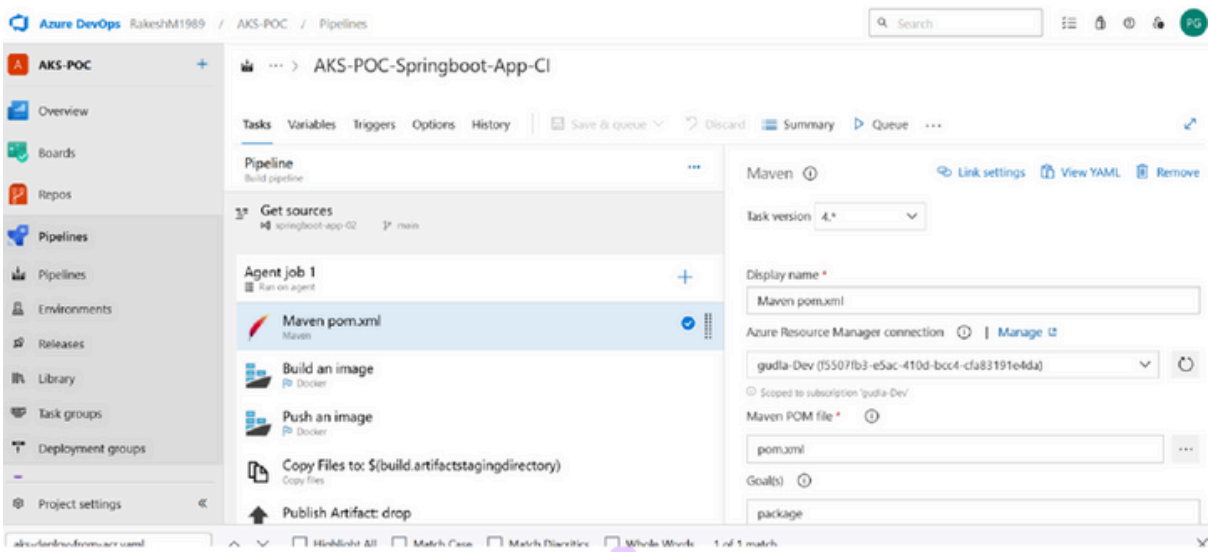
deployment.yml

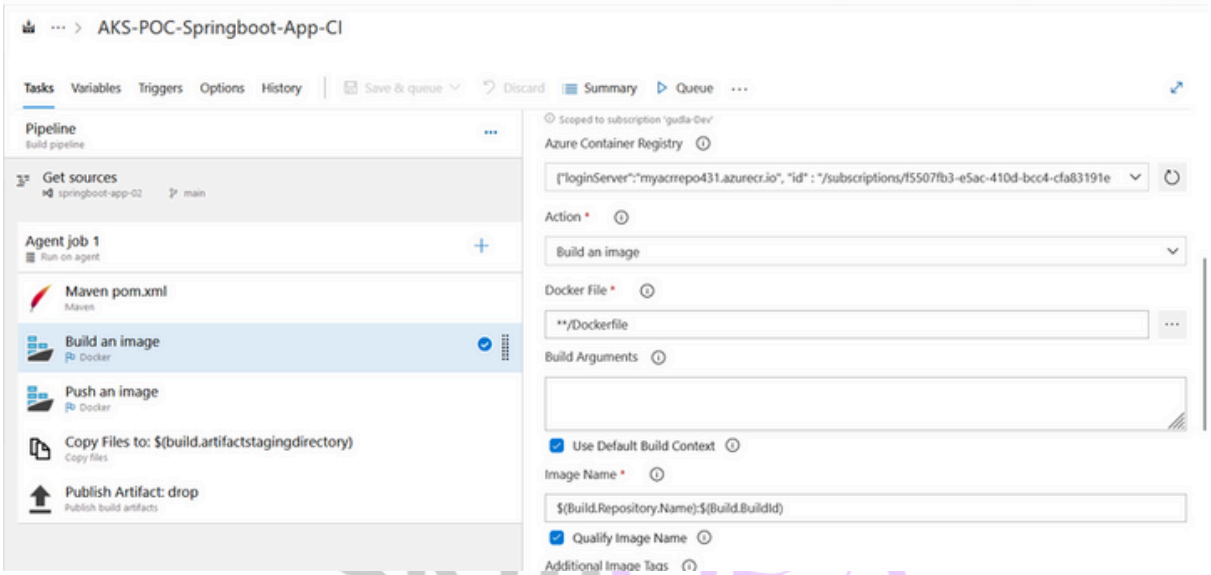
```
apiVersion : apps/v1
kind: Deployment
metadata:
 name: akannanspringbootapp
spec:
 replicas: 1
 selector:
  matchLabels:
   app: akannanspringbootapp
 template:
  metadata:
   labels:
    app: akannanspringbootapp
  spec:
   containers:
    - name: akannanspringbootapp
      image: testacr1989.azurecr.io/akannanspringbootapp
      ports:
      - containerPort: 80
```

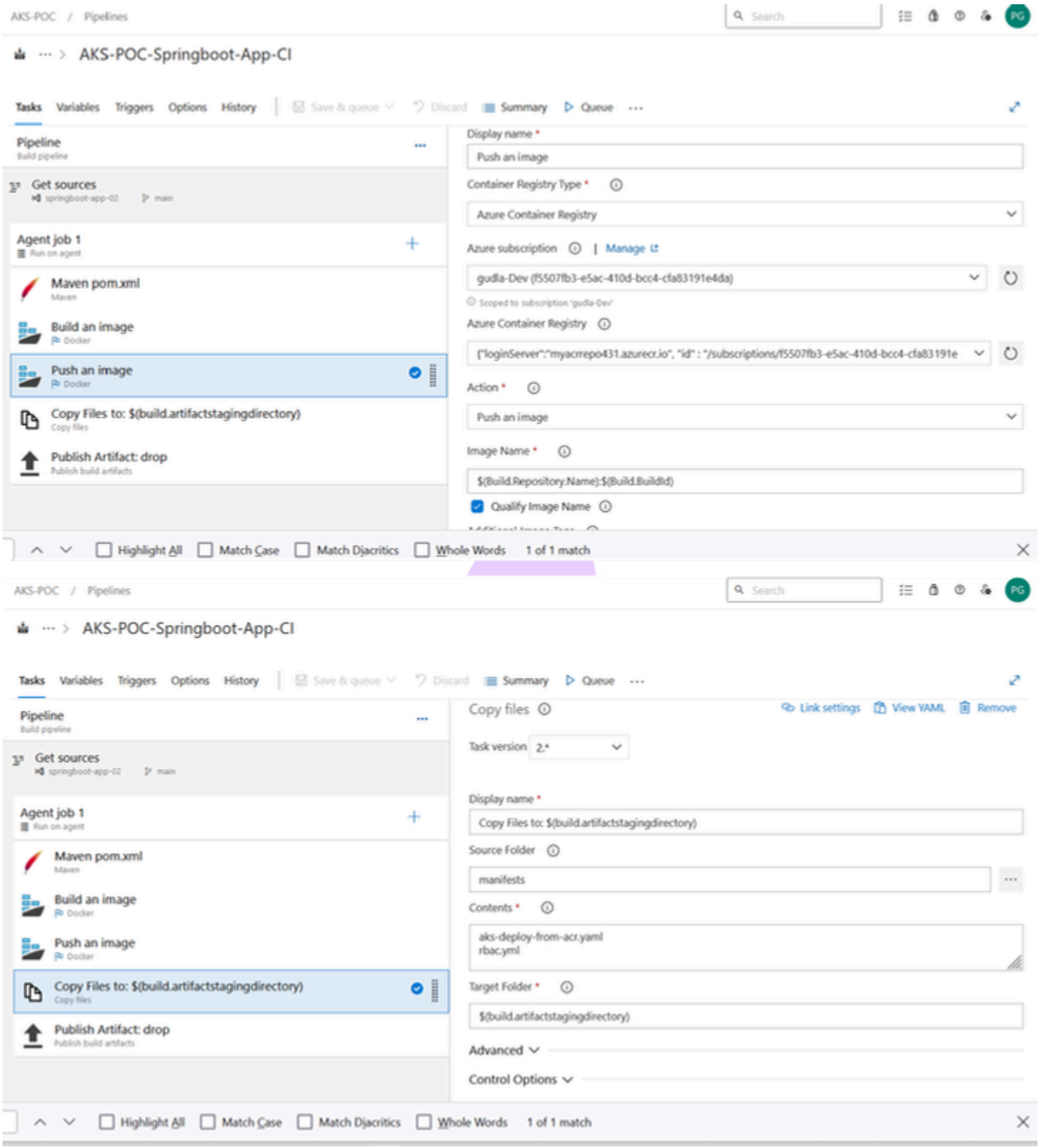
service.yml

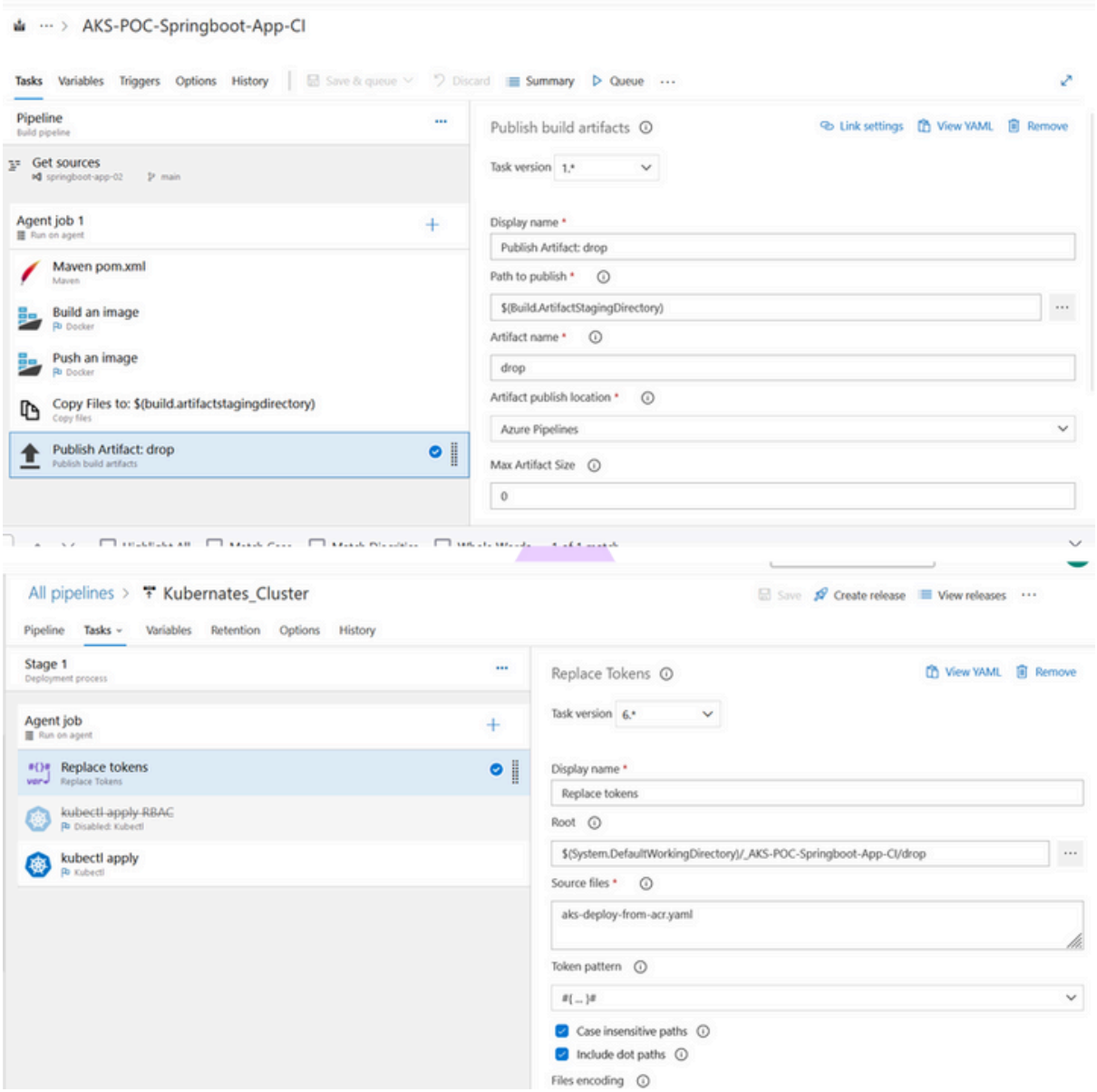
```
apiVersion: v1
kind: Service
metadata:
  name: akannanspringbootapp
spec:
  type: LoadBalancer
  ports:
  -port: 80
  selector:
    app: akannanspringbootapp
```

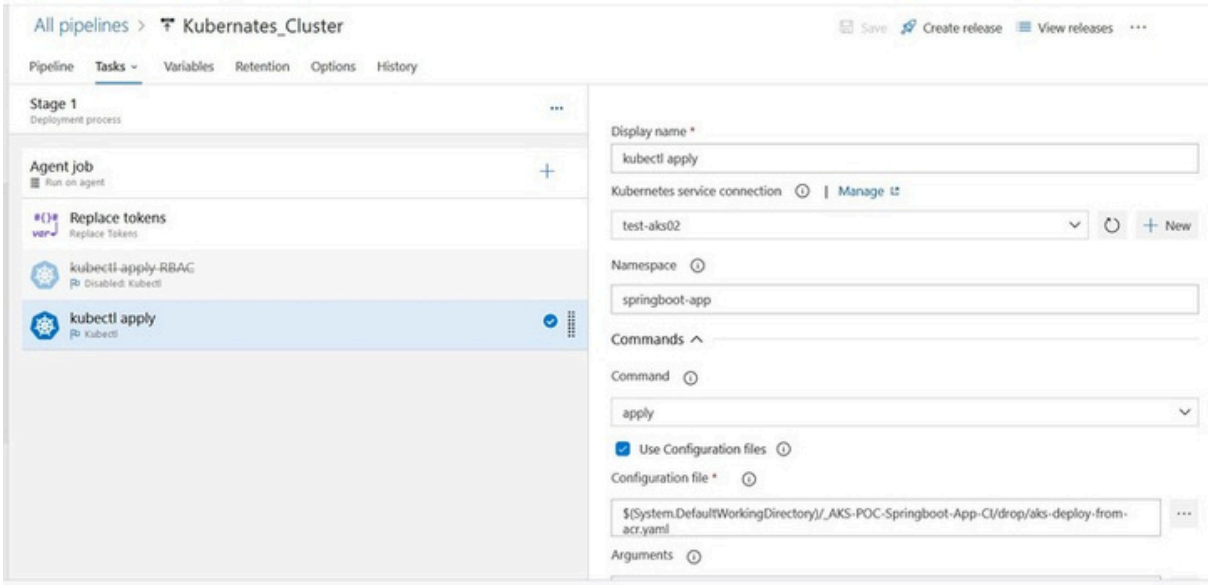



Azure DevOps
AKS-POC
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Releases
Library
Task groups
Deployment groups
Project settings
AKS-POC-Springboot-App-CI
Tasks
Variables
Triggers
Options
History
Save & queue
Discard
Summary
Queue
Pipeline
Get sources
Agent job 1
Maven pom.xml
Build an image
Push an image
Copy Files to: $(build.artifactstagingdirectory)
Publish Artifact: drop
Maven
Link settings
View YAML
Remove
Task version
Display name *
Maven pom.xml
Azure Resource Manager connection
Manage
gudla-Dev
Maven POM file *
pom.xml
Goal(s)
package


AKS-POC-Springboot-App-CI
Tasks
Variables
Triggers
Options
History
Save & queue
Discard
Summary
Queue
Pipeline
Get sources
Agent job 1
Maven pom.xml
Build an image
Push an image
Copy Files to: $(build.artifactstagingdirectory)
Publish Artifact: drop
Azure Container Registry
Action *
Build an image
Docker File *
**/Dockerfile
Build Arguments
Use Default Build Context
Image Name *
$(Build.Repository.Name):$(Build.BuildId)
Qualify Image Name
Additional Image Tags


AKS-POC / Pipelines
AKS-POC-Springboot-App-CI
Tasks Variables Triggers Options History
Save & queue Discard Summary Queue
Pipeline
Get sources
Agent job 1
Maven pom.xml
Build an image
Push an image
Copy Files to: $(build.artifactstagingdirectory)
Publish Artifact: drop
Display name *
Push an image
Container Registry Type *
Azure Container Registry
Azure subscription | Manage
gudla-Dev (f5507fb3-e5ac-410d-bcc4-cfa83191e4da)
Azure Container Registry
{"loginServer":"myacrrepo431.azurecr.io", "id" : "/subscriptions/f5507fb3-e5ac-410d-bcc4-cfa83191e
Action *
Push an image
Image Name *
$(Build.Repository.Name):$(Build.BuildId)
Qualify Image Name
Highlight All Match Case Match Diacritics Whole Words 1 of 1 match
AKS-POC / Pipelines
AKS-POC-Springboot-App-CI
Tasks Variables Triggers Options History
Save & queue Discard Summary Queue
Pipeline
Get sources
Agent job 1
Maven pom.xml
Build an image
Push an image
Copy Files to: $(build.artifactstagingdirectory)
Publish Artifact: drop
Copy files
Link settings View YAML Remove
Task version 2.*
Display name *
Copy Files to: $(build.artifactstagingdirectory)
Source Folder
manifests
Contents *
aks-deploy-from-acr.yaml
rbac.yml
Target Folder *
$(build.artifactstagingdirectory)
Advanced
Control Options
Highlight All Match Case Match Diacritics Whole Words 1 of 1 match


AKS-POC-Springboot-App-CI
Tasks
Variables
Triggers
Options
History
Save & queue
Discard
Summary
Queue
Pipeline
Build pipeline
Get sources
springboot-app-02
main
Agent job 1
Run on agent
Maven pom.xml
Maven
Build an image
Docker
Push an image
Docker
Copy Files to: $(build.artifactstagingdirectory)
Copy files
Publish Artifact: drop
Publish build artifacts
Publish build artifacts
Link settings
View YAML
Remove
Task version 1.*
Display name *
Publish Artifact: drop
Path to publish *
$(Build.ArtifactStagingDirectory)
Artifact name *
drop
Artifact publish location *
Azure Pipelines
Max Artifact Size
0
All pipelines
Kubernates_Cluster
Save
Create release
View releases
Pipeline
Tasks
Variables
Retention
Options
History
Stage 1
Deployment process
Agent job
Run on agent
Replace tokens
Replace Tokens
kubectl apply RBAC
Disabled: Kubectl
kubectl apply
Kubectl
Replace Tokens
View YAML
Remove
Task version 6.*
Display name *
Replace tokens
Root
$(System.DefaultWorkingDirectory)/_AKS-POC-Springboot-App-CI/drop
Source files *
aks-deploy-from-acr.yaml
Token pattern
#{ ... }#
Case insensitive paths
Include dot paths
Files encoding


All pipelines > Kubernates_Cluster
Save
Create release
View releases
Pipeline
Tasks
Variables
Retention
Options
History
Stage 1
Deployment process
Agent job
Run on agent
Replace tokens
Replace Tokens
kubectl apply RBAC
Disabled: Kubectl
kubectl apply
Kubectl
Display name *
kubectl apply
Kubernetes service connection
Manage
test-aks02
New
Namespace
springboot-app
Commands
Command
apply
Use Configuration files
Configuration file *
$(System.DefaultWorkingDirectory)/_AKS-POC-Springboot-App-CI/drop/aks-deploy-from-acr.yaml
Arguments

Conclusion – The Journey to Cloud & DevOps Excellence

Congratulations on reaching the end of this book! You have embarked on a transformative journey through **Cloud Computing, DevOps, Azure, Docker, Kubernetes, and more**. By now, you have built a **solid foundation in cloud technologies and DevOps practices**, gaining not just theoretical knowledge but also **real-world implementation skills** through hands-on exercises and projects.

What's Next?

Apply What You Learned – The best way to master Cloud & DevOps is through continuous practice. Implement real-world projects, contribute to open-source, and experiment with different architectures.

Stay Updated – Cloud and DevOps technologies **evolve rapidly**. Follow tech blogs, join community forums, and attend meetups or webinars to stay ahead.

Get Certified – If you're planning for certifications like **AZ-104, AZ-400, CKAD, or Terraform Associate**, now is the perfect time to validate your skills and boost your career.

Never Stop Learning – The IT world never stops innovating. Keep pushing yourself to explore **AI, Security, Automation**, and other emerging trends that integrate with Cloud & DevOps.

A Heartfelt Thank You

I sincerely **thank you** for choosing this book as your learning companion. Whether you're a **beginner stepping into the cloud** or an **experienced engineer refining your DevOps skills**, I hope this book has **empowered you with knowledge and confidence**.

I wish you **success, growth, and endless opportunities** in your **Cloud & DevOps career**. Keep learning, keep innovating, and most importantly, **keep pushing boundaries!**

Signing Off with a Powerful Thought

*"The cloud is not just about technology; it's about a **mindset shift**. The faster you embrace automation, scalability, and agility, the closer you get to becoming a true cloud professional."*

"అర్జునుడిలా సందేహ ంచవచు, కానీ కృష్ణని మార్గదర్శకత్వం ఉంటే... విజయం ఖాయం!"

భగవదగత్లలో చెపేపినటుట – కర్మ చేయడమేమన బాధ్యత్. ఫలితాలు సవయంగా వచ్చు చేర్తాయి. జాఞనం, ధైర్యం, కరమశిక్షణతో మ ందుకు సాగ దాం.

See you in the world of innovation!

Best Regards,

Praveen G

skillORA

Empowering IT Aspirants, Enabling Future-Professionals
SkillORA isn't just an institute — it's a mission.

K.K.Reddy - (CEO & Founder)

"At SKILLORA, we believe that talent is everywhere — but opportunity is not. Our mission is to break this barrier. Whether you're from a small town or a tech hub, we are here to unlock your potential with world-class training, mentorship, and real-time exposure. Let your dreams fly high — SKILLORA is your launchpad."

Praveen G - (MD & Co-Founder)

"Being from a Cloud & DevOps background myself, I understand the real struggles students face in transitioning from learning to earning. That's why SKILLORA is designed to replicate the IT industry's demands with hands-on labs, real-world scenarios, and career guidance. We don't just want to teach — we want to build future-ready professionals. Your success is our mission."

- Multi-Cloud & DevSecOps
- Data Science, AI & ML
- Full Stack Java Development
- Python Full Stack

Let your career take flight with SkillORA. Because the future doesn't wait — and neither should you.

FOR MORE INFORMATION CONTACT US

follow us on

www.skillora.co.in
VISIT OUR WEBSITE

+91 90109 27666
manager@skillora.co.in

"At SKILLORA, we don't just teach — we TRANSFORM careers."