Microsoft Azure Fundamentals AZ-900 Exam Guide

(What, Where and How)!!!

First, understand basic Definition, what is **CLOUD**? what is **Cloud Computing**?

Cloud: The cloud refers to a network of remote servers hosted on the internet that store, manages, and process data. Instead of using a local computer or server, you access resources like storage, applications, or services over the internet.

Cloud Computing: Cloud computing is the delivery of computing services—like storage, processing power, databases, software, or applications—over the internet ("the cloud"). It allows users to access and use these resources on-demand without needing to own or maintain physical hardware.

think of the cloud as a shared pool of resources you can rent and use online, and cloud computing is the way you use those resources to do tasks like storing files, running apps, or analyzing data, all without managing the underlying servers yourself.

==========

Cloud Concepts

What is Cloud Computing? Why are we using this? What are the Benefits?

Cloud computing is the on-demand delivery of computing services, including *servers, storage, databases, networking, software, analytics, and intelligence*—over the internet ("the cloud") to offer faster innovation, flexible resources, and economies of scale. Instead of owning and maintaining your own computing infrastructure, you can access these services from a third-party cloud provider and pay only for what you use.

How Cloud Computing Works?

Cloud computing service models are based on the concept of sharing on-demand computing resources, software, and information over the internet. Companies or individuals pay to access a virtual pool of shared resources, including compute, storage, and networking services, which are located on remote servers that are owned and managed by cloud service providers *(like Amazon Web Services (AWS), Microsoft Azure, Google Cloud, etc.)*. A central server handles all communication between client devices and the cloud servers to facilitate data exchange.

The core idea is that you're no longer responsible for buying, installing, and maintaining physical hardware and software. The **cloud provider takes care** of all that "undifferentiated heavy lifting," allowing you to focus on your core business activities.

Why are We Use Cloud Computing?

- To Reduce Capital Expenditures (CapEx) and Shift to Operational Expenditures (OpEx)
  Meaning- **Traditional IT (CapEx):** Requires significant upfront investment in hardware, software licenses, data center space, and power. This can be a huge barrier for startups and a drain on cash flow for established businesses.
  Cloud (OpEx): You pay for resources as you consume them, like a utility bill. This transforms large, infrequent capital expenses into smaller, predictable operational expenses, freeing up capital for other business investments.

- To Achieve Scalability and Elasticity
- To Enhance Agility and Speed to Market
- To Improve Reliability and Business Continuity
- To Strengthen Security Posture
- To Focus on Core Business, Not IT Management
- To Access Advanced Technologies
- To Enable Global Reach
- To Facilitate Collaboration and Remote Work

Benefits-

a) **Cost Savings:** Reduce capital expenditure, pay-as-you-go pricing.
b) **Scalability & Elasticity:** Easily scale resources up or down to match demand.
c) **Agility & Speed:** Faster deployment of applications and services.

d) **Reliability & Disaster Recovery:** High availability and robust data protection.
e) **Enhanced Security:** Leveraging the extensive security expertise and infrastructure of cloud providers.
f) **Focus on Core Business:** Offload IT management to experts.
g) **Access to Innovation:** Leverage cutting-edge technologies like AI and ML.
h) **Global Reach:** Deploy applications worldwide with ease.
i) **Improved Collaboration:** Enable remote work and seamless team interaction.

How Many Clouds are available in the marketplace?

Cloud Service providers:

Major- [market leaders / Giant]- Amazon Web Services (AWS), Microsoft Azure and Google Cloud Platform (GCP).

Other vendors in Market: - Alibaba Cloud, Oracle Cloud, IBM Cloud, Tencent Cloud, Huawei Cloud and Cisco Cloud.

Understand the Difference between Public Cloud, Private Cloud and Hybrid Cloud.

Public Cloud

Public cloud services are provided by third-party cloud service providers (CSPs) over the public internet. The infrastructure (servers, storage, networking, software) is owned and managed by CSP and is shared among multiple customers (tenants). *Example- AWS, GCP and Azure.*

Private Cloud

A private cloud refers to cloud computing resources used exclusively by a single organization. It can be physically located on the company's on-site data center (on-premises private cloud) or hosted by a third-party service provider (hosted private cloud).

Example- An organization building and managing its own cloud infrastructure within its data center.

Hybrid Cloud

A hybrid cloud is a computing environment that combines two or more distinct cloud infrastructures (private, public, or even other hybrid clouds) into a single, unified architecture. It allows data and applications to be shared and orchestrated between these environments.

Example- A financial institution stores customer transaction data in its on-premises private cloud but uses a public cloud for its customer-facing website and mobile banking app.

Types Of Azure Exams/Certifications

For beginners- AZ-900 [Azure Fundamentals]- - -> *Entry level Exam.*

For developers-

- Azure Developer Associate (AZ-204)
- Azure DevOps Engineer Expert (AZ-400)
- Azure Administrator Associate (AZ-104)
- Azure AI Engineer Associate (AI-102)
- Azure Data Engineer Associate (DP-203)
- Azure Cosmos DB Developer Specialty (DP-420)
- Azure IoT Developer Specialty (AZ-220)

For Network Engineers-

- Azure Network Engineer Associate (AZ-700)
- Azure Administrator Associate (AZ-104)
- Azure Security Engineer Associate (AZ-500)
- Azure Solutions Architect Expert (AZ-305)

Other Miscellaneous Certifications:

- Azure Security, Compliance, and Identity Fundamentals (SC-900)
- Microsoft Cybersecurity Architect Expert (SC-100)
- Microsoft Security Operations Analyst (SC-200)
- Microsoft Identity and Access Administrator (SC-300)

Cloud Services Models

Often referred as “Cloud computing stack”, because they build on top of each other.

Also called as [Function as a service]. [mainly 3 Types]-

1. **Infrastructure as a Service (IaaS)-** Where a provider delivers virtualized computing resources—such as servers, storage, networking, and operating systems—over the internet.

IaaS = Rent IT infrastructure instead of buying it.

Components: This typically includes: -

- **Virtual Machines (VMs):** Virtualized servers that you can provision and configure.
- **Storage:** Block storage (like virtual hard drives), file storage, and object storage.
- **Networking:** Virtual networks, IP addresses, routers, load balancers, and firewalls.
- **Operating Systems:** You install and manage the operating system of your choice on the VMs. *(MacOS, Linux, Windows.)*

Your/User responsibility: You are responsible for managing the operating systems, applications, data, runtime, and middleware.

Provider's responsibility: The cloud provider manages the underlying infrastructure, including the physical servers, virtualization software, networking hardware, and data center facilities.

Analogy: Think of it like renting an empty apartment building. You get the structure, plumbing, and electricity, but you're responsible for everything inside: furniture, appliances, painting, and maintenance of your specific apartment units.

Use Cases:

  - Hosting websites and web applications.
  - Development and testing environments.
  - Data backup, recovery, and archival.
  - High-performance computing (HPC) workloads.

- **Examples:** Amazon EC2 (Elastic Compute Cloud), Azure Virtual Machines, Google Compute Engine.

2. **Platform as a Service (PaaS)-** Where a provider delivers a **ready-to-use platform** that includes everything you need to **develop, run, and manage applications**—without managing the underlying infrastructure (like servers, storage, or OS).

PaaS = You build and run your app; the cloud provider manages everything else.

Components: Includes all the IaaS components, plus:

- **Operating systems** (managed by provider).
- Programming language execution environments.
- **Databases.**
- **Web servers.**
- Development tools, middleware, and frameworks.

User/Your responsibility: You primarily focus on developing, deploying, and managing your applications and data.

Provider's responsibility: The cloud provider manages all the underlying infrastructure, operating systems, and platform software.

Analogy: This is like renting a fully furnished apartment. You just move in and start living; you don't worry about the building's structure or utility lines, but you still manage your personal belongings. 

Use Cases:

- Rapid development and deployment of web applications.
- Building and deploying APIs.
- Analytics and business intelligence.
- Microservices development.

Examples: AWS Elastic Beanstalk, Azure App Service, Google App Engine, Heroku.

3. **Software as a Service (SaaS)-** delivers a complete, fully functional software application over the internet on a subscription basis. Users simply access the application via a web browser or mobile app.

SaaS = Use the software via browser or app — no setup, no servers.

Components: The cloud provider manages the entire application stack: *the application itself, runtime, middleware, operating systems, servers, storage, and networking.*

User/Your responsibility: You are typically only responsible for using the software and managing your own data within the application.

Provider's responsibility: The cloud provider handles everything from infrastructure to application maintenance, updates, security, and bug fixes.

Analogy: This is like using a hotel room. Everything is provided and managed for you – furniture, utilities, cleaning. You just use the space for your stay.

Use Cases:

- Email services (e.g., Gmail, Outlook.com).
- Customer Relationship Management (CRM) (e.g., Salesforce).
- Productivity suites (e.g., Microsoft 365, Google Workspace).
- Human Resources (HR) software, accounting software.

Examples: Salesforce, Dropbox, Microsoft 365, Google Workspace, Zoom.

Nowadays, a new term/Service is Introduced- Serverless Computing-
Named- Function as a Service. [Faas]

FaaS = Write your code, deploy it, and the cloud runs it only when triggered. No servers to manage.

Components: The provider manages the servers, operating systems, runtimes, and scaling.

User/Your responsibility: You only write and manage your code (functions).

Provider's responsibility: The cloud provider handles all server management, scaling, patching, and provisioning. You don't see or manage any servers.

Analogy: This is like a public utility that delivers exactly what you need when you need it, and you only pay for the exact consumption (e.g., electricity by the kilowatt-hour).

Use Cases:

- Event-driven applications (e.g., image processing after upload, real-time data processing).
- Building serverless APIs and microservices.
- Chatbots and IoT backends.
- Scheduled tasks (e.g., running a daily report).

Examples: AWS Lambda, Azure Functions, Google Cloud Functions.

Share responsibility Model [Role of Cloud provider and User]

The "Shared Responsibility Model" is a fundamental concept in cloud computing, and it's absolutely critical to understand when using Azure (or any other cloud provider like AWS or Google Cloud). It clearly defines who is responsible for what security tasks between the cloud service provider (Microsoft, in Azure's case) and the customer (you or your organization).

In-Short: Microsoft is responsible for the security *of* the cloud.

You (the customer) are responsible for security *in* the cloud.

==========

The exact division of responsibility shifts based on *the type of cloud service model you're using:* **IaaS**, **PaaS**, or **SaaS**.

The Core Idea: It's a Partnership, not a Hand-off

Many people mistakenly believe that by moving to the cloud, all security responsibilities are transferred to the cloud provider. This is incorrect and can lead to significant security gaps. The shared responsibility model highlights that **security is a joint effort**.

Microsoft secures the underlying infrastructure, and you secure your data and applications running on that infrastructure. (v.v.v Important).

Now, why this Model is Important?

A. **Clarity of Responsibilities:** It removes ambiguity about who does what, preventing security gaps where both parties might assume the other is handling a task.
B. **Risk Management:** By understanding your responsibilities, you can perform proper risk assessments and implement appropriate controls for the parts of the cloud you manage.
C. **Compliance:** Many regulatory frameworks require organizations to demonstrate control over their data and applications, even in the cloud. This model helps define those boundaries.
D. **Effective Security Strategy:** It enables you to leverage Microsoft's massive security investments in the underlying infrastructure, while focusing your own security efforts on the areas you control and where your unique business logic and sensitive data reside.

Azure and Other Public Clouds are Consumption Based Model [pay-as-You-go]

Meaning- Simple, you are billed only for the computing resources you actually use, and only for the duration you use them. (important!!) ⟸

Imagine a scenario- Your electricity bill or water bill. You don't buy a power plant or a water reservoir to supply your home. Instead, you plug into the utility grid, consume electricity/water as needed, and receive a bill at the end of the month based on your exact usage (kilowatt-hours, liters). Azure works similarly for IT resources.

Deeper Explanation- ⬎

Granular Metering:

- Azure "meticulously" ***(अति सावधानी से) (extra attentive)*** tracks the usage of every single service and resource you provision. This tracking is done at a very granular level, often down to the second, minute, gigabyte, or transaction.
- **Examples of Consumption Metrics:**
    - **Compute (Virtual Machines, Functions):** Billed by the hour or even by the second that a VM is running. For serverless functions (like Azure Functions), it's often based on the number of executions and the memory/CPU consumed during those executions (e.g., GB-seconds).

- **Storage:** Billed by the amount of data stored (per GB-month), the type of storage (hot, cool, archive), and the number of read/write operations (transactions).
- **Networking:** Billed by the amount of data transferred out of Azure data centers (inbound data transfer is often free or very cheap), public IP addresses used, and specific networking service usage (e.g., load balancer rules, VPN gateway hours).
- **Databases:** Billed by database size, provisioned throughput (RU/s for Cosmos DB), number of transactions, and backup storage.
- **AI/ML Services:** Often billed per transaction, per API call, or based on compute time for training models.

No Upfront Capital Expenditure (CapEx):

- In traditional IT, you'd incur a large CapEx to buy servers, storage arrays, networking gear, and build data centers. This ties up significant capital.
- With Azure's consumption model, there's **no need for large upfront investments in hardware.** You simply sign up, and you can start provisioning resources immediately. This frees up capital for other business initiatives.

Shift from CapEx to Operational Expenditure (OpEx):

- Instead of capital investments, your IT costs become operational expenses. You budget for and pay for Azure services as an ongoing operating cost, similar to rent or utilities. This can significantly improve cash flow and financial flexibility.

Elasticity and Scalability:

- This is where the consumption model truly shines. Your business needs can fluctuate wildly.
  - **Spikes in Demand:** During peak seasons, marketing campaigns, or unexpected traffic surges, you can instantly scale up your resources (e.g., add more VMs, increase database throughput). You only pay for these extra resources during the period of high demand.
  - **Lulls in Demand:** When demand drops, you can scale down or even deallocate resources. You stop paying for those resources when they are no longer in use. This prevents waste and unnecessary costs.
- This dynamic scaling ensures optimal performance when needed, without over-provisioning and incurring costs for idle resources.

- **Potential Cost Optimization:**

While the basic model is pay-as-you-go, Azure offers various mechanisms to optimize costs within this model:

  - **Resource Deallocation/Stopping:** For VMs, stopping them (deallocating) means you only pay for storage, not compute. This is crucial for dev/test environments.
  - **Right-sizing:** Continuously monitoring your resource usage to ensure you're using the right size of VM, database, or other service, avoiding over-provisioning.
  - **Azure Advisor:** Provides personalized recommendations to optimize costs, performance, security, and reliability.
  - **Azure Cost Management + Billing:** Tools to track, analyze, and optimize your cloud spending.
  - **Azure Monitor:** Provides insights into resource utilization, allowing you to fine-tune scaling policies.

Discounts and Commitment Options (Layered on top of Consumption):

- While the default is pay-as-you-go, for predictable, long-running workloads, Azure offers ways to get discounts by making commitments:
  - **Azure Reservations:** By committing to a 1-year or 3-year term for specific compute resources (like VMs), you can get significant discounts (e.g., up to 72%). You still pay per hour/second, but at a much lower reserved rate.
  - **Azure Savings Plans for Compute:** A more flexible commitment plan where you commit to a consistent hourly spend on compute services for 1 or 3 years, gaining discounts on eligible usage across various compute services.
  - **Azure Hybrid Benefit:** Leverage your existing Windows Server or SQL Server licenses with Software Assurance to get reduced rates on Azure VMs and Azure SQL Database.
  - **Spot Instances:** For fault-tolerant workloads, you can bid for unused Azure capacity at heavily discounted rates. These instances can be preempted (stopped) if Azure needs the capacity back.

*** AWS and GCP also work in Consumption based Model ***

Vertical Scaling and Horizontal Scaling in Azure

In Azure (and other cloud platforms), **scaling** refers to the ability to increase or decrease the computing resources allocated to an application or service to handle changes in demand. There are two primary types of scaling: **Vertical Scaling (Scaling Up/Down)** and **Horizontal Scaling (Scaling Out/In)**.

Vertical Scaling (Scaling Up / Scaling Down)-

Definition: Vertical scaling, often called "scaling up" or "scaling down," involves **increasing or decreasing the resources (CPU, RAM, storage, network bandwidth) of a *single* existing instance** of a service or application. You make the current machine more powerful.

Scenario: Imagine you have one person doing a task, and they are becoming overwhelmed. Vertical scaling is like giving that *same person* more tools, more desk space, a faster computer, or a bigger brain so they can handle more work individually.

Example in Azure:

Azure Virtual Machine (VM): You have a web server running on a *Standard_D2s_v3 VM (2 vCPUs, 8 GB RAM)*. As your website traffic grows, you notice VM's CPU utilization constantly hitting 90%. To handle the increased load, you decide to **vertically scale up** the VM to a *Standard_D4s_v3 (4 vCPUs, 16 GB RAM).* This involves stopping the VM, changing its size to the Azure portal, and then restarting it with the new, more powerful configuration. If traffic later decreases, you might scale it back down.

Horizontal Scaling (Scaling Out / Scaling In)-

Definition: Horizontal scaling, often called "scaling out" or "scaling in," involves adding or removing more instances (or nodes) of a service or application to distribute the workload across multiple machines.

Scenario: Using the same example, if one person is overwhelmed, horizontal scaling is like hiring *more people* to work alongside them, so each person handles a smaller part of the total task.

Example in Azure:

- **Azure App Service (for web applications**): You have a web application hosted on Azure App Service. During a flash sale, your website experiences a massive surge in traffic. You can horizontally scale out your App Service Plan from 2 instances to 10 instances. Azure's load balancer automatically distributes incoming web requests across all 10 instances, ensuring the website remains responsive. When the sale ends and traffic subsides, the App Service can scale in automatically (if autoscaling is configured) back to 2 instances to save costs.
- **Azure Virtual Machine Scale Sets (VMSS):** You have a backend processing application running on a VMSS. As the number of incoming messages to be processed increases in a queue (e.g., Azure Service Bus queue), you can configure an autoscaling rule to scale out the VMSS by adding more VM instances (e.g., from 3 to 7 VMs). Each new VM instance runs the application, pulling messages from the queue and processing them. When the queue length drops, the VMSS can scale in to reduce costs.

Policy Enforcement in Azure

Policy Enforcement in Azure, primarily through **Azure Policy**, is a robust governance service that "allows organizations to create, assign, and manage policies that enforce rules and effects over their Azure resources". Its main goal is to ensure compliance with corporate standards, regulatory requirements, security best practices, and cost controls across your Azure environment at scale.

Think of Azure Policy as a guardrail that ensures your Azure resources stay within defined boundaries and configurations.

How Policy Enforcement Works?

1. **Evaluation Triggers:** Azure Policy evaluates resources at specific times:
    - When a resource is **created or updated**.
    - When a new policy or initiative is **assigned** to a scope.
    - When an existing policy or initiative assignment is **updated**.

        - During a **standard compliance evaluation cycle** (which occurs every 24 hours).

2. **Compliance Dashboard:**
    - Azure Policy provides a centralized compliance dashboard where you can see the compliance state of your resources against all assigned policies and initiatives. This helps identify non-compliant resources and track remediation efforts.
3. **Remediation Tasks:**
    - For policies with DeployIfNotExists or Modify effects, you can create **remediation tasks** to apply the policy to *existing* non-compliant resources. This helps bring your current environment into compliance.

Benefits of Policy Enforcement in Azure:

- **Standardization & Consistency:** Ensures all resources adhere to predefined configurations, naming conventions, and best practices.
- **Security Posture:** Enforces security controls like mandatory encryption, specific network configurations, and disabling insecure protocols.
- **Cost Control:** Prevents the deployment of expensive or unapproved resource types or sizes.
- **Regulatory Compliance:** Helps meet internal and external regulatory requirements (e.g., HIPAA, GDPR, PCI DSS) by ensuring specific configurations are in place.
- **Automated Governance:** Reduces manual effort in enforcing rules, leading to more efficient operations.
- **Visibility:** Provides a clear overview of your organization's compliance status.
- **Prevents "Configuration Drift":** Actively ensures resources remain in their desired state, even if manual changes are attempted.

Compliance in Azure

In the context of Azure, **compliance** refers to the adherence of your Azure cloud environment and the *services you consume within it to a set of predefined rules, standards, regulations, and industry best practices.* These standards can be:

i. **External Regulatory Standards:** Laws and regulations imposed by governments or regulatory bodies *(e.g., HIPAA for healthcare in the US, GDPR for data privacy in the EU, PCI DSS for payment card data, ISO 27001 for information security management).*

ii. **Internal Corporate Policies:** Rules and guidelines established by your own organization to ensure security, governance, operational efficiency, and cost control *(e.g., "all VMs must have a 'CostCenter' tag," "no public IP addresses allowed on production databases," "only approved VM sizes can be deployed".)*

iii. **Industry Best Practices:** Recommended configurations and operational procedures for optimal security, performance, and reliability within your industry or within cloud computing in general.

In-Short: Compliance in Azure is the *continuous process of ensuring* that your cloud environment and operations meet defined security, privacy, and regulatory standards, leveraging both Microsoft's platform compliance and your organization's diligent application of controls using Azure's governance tools

Azure Architecture and Services

Azure Resources List. (resource Group) -> What are these? In what ways can we use them?

Definition- In Azure, a resource is a manageable item that is available through Azure. Essentially, any service or component you create and use in Azure is considered a resource.

Think of an Azure resource as a single, distinct building block or a specific instance of a service. *Every time you provision something in Azure, you're creating one or more resources.*

Common Examples of Azure Resources:

Compute:

- **Virtual Machines (VMs):** Linux VMs, Windows VMs.
- **App Services:** Web Apps, API Apps, Mobile Apps, Function Apps (Serverless).
- **Container Instances:** Azure Container Instances (ACI).
- **Kubernetes Service (AKS):** Kubernetes clusters.
- **Virtual Machine Scale Sets (VMSS):** Groups of identical, load-balanced VMs.

Networking:

- **Virtual Networks (VNets):** Isolated network spaces.
- **Subnets:** Divisions within a VNet.
- **Network Interfaces (NICs):** Connect VMs to a VNet.
- **Public IP Addresses:** For internet accessibility.
- **Load Balancers:** Distribute traffic.
- **Application Gateways:** Web traffic load balancer with WAF.
- **VPN Gateways:** Connect on-premises networks to Azure.
- **ExpressRoute Circuits:** Private, dedicated connections.
- **Network Security Groups (NSGs):** Firewall rules for network traffic.

Storage:

- **Storage Accounts:** A single account that can contain various storage types:
  - **Blob Storage:** For unstructured data (images, videos, files).
  - **File Storage:** SMB file shares in the cloud.
  - **Queue Storage:** For message queuing.

    - **Table Storage:** NoSQL key-value store.

- **Managed Disks:** Virtual hard disks for VMs.
- **Azure Data Lake Storage:** Massively scalable data lake.

Databases:

- **Azure SQL Database:** Managed relational database (PaaS).
- **Azure Database for MySQL/PostgreSQL/MariaDB:** Managed open-source relational databases.
- **Azure Cosmos DB:** Globally distributed, multi-model NoSQL database.
- **Azure Cache for Redis:** In-memory data store.

Identity & Security:

- **Azure Active Directory (now Microsoft Entra ID):** Users, groups, applications, managed identities are all resources within Entra ID.
- **Azure Key Vault:** Store secrets, keys, and certificates.
- **Azure Firewall:** Managed network firewall service.

Analytics:

- **Azure Synapse Analytics:** Unified analytics platform.
- **Azure Databricks:** Apache Spark-based analytics platform.
- **Azure Stream Analytics:** Real-time data stream processing.

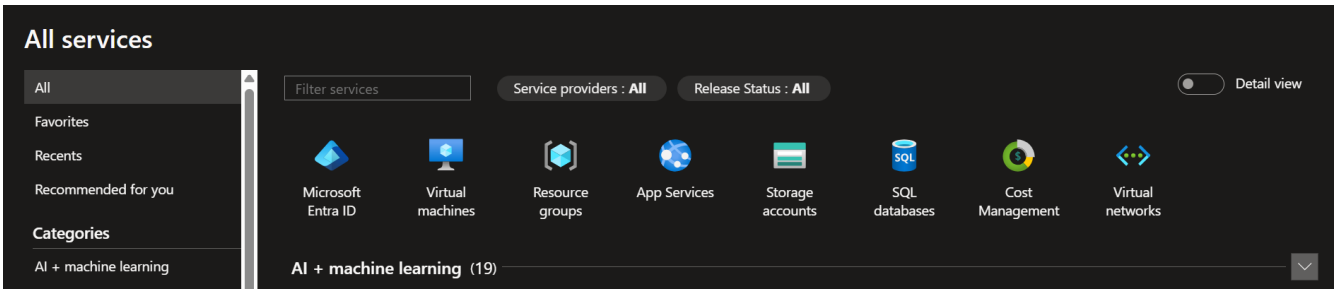
Internet of Things (IoT):

- **Azure IoT Hub:** Connect, monitor, and manage IoT devices.
- **Azure IoT Edge:** Edge computing services.

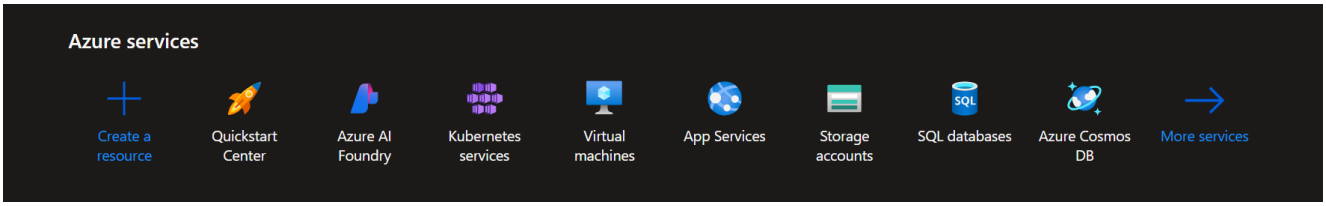
Management & Governance:

- **Azure Monitor:** Collects, analyzes, and acts on telemetry data.
- **Azure Policy:** Defines and enforces rules.
- **Azure Blueprints:** Deploys compliant environments.
- **Azure Cost Management + Billing:** Tools for cost analysis.

*** (We will See and Look major resources later in the guide) ***



Example-snapshot from Azure Portal [Azure-Services]*



What is an Azure Resource Group?

An **Azure Resource Group (RG)** is a **logical container** that "holds related Azure resources" for an Azure solution. It's a fundamental organizational unit in Azure's Resource Manager deployment model.

*Read and understand carefully *

Key Characteristics and Purpose of Resource Groups:

Logical Grouping:

- Resource groups allow you to organize resources that belong together. This grouping is entirely up to you and how you want to manage your environment.
- **Common Grouping Strategies:**
  - **By Application:** All resources for a specific application (e.g., a web app, its database, storage) are in one RG.
  - **By Environment:** Resources for "development," "testing," "staging," and "production" environments are in separate RGs, even if they're for the same application. (e.g., MyApp-Prod-RG, MyApp-Dev-RG).
  - **By Department/Project:** Resources used by a specific department or project team.

- The goal is to bring together resources that share a common lifecycle or management needs.

Shared Lifecycle Management:

- The most crucial aspect of a resource group is that resources within it ideally **share the same lifecycle**. This means "you deploy, update, and delete them together."

- **Example:** If you're deploying a web application that includes a Web App, an Azure SQL Database, and a Storage Account, putting them all in one resource group means you can delete the entire application solution by simply deleting that single resource group. This simplifies cleanup and ensures no orphaned resources are left behind.

Unit of Management:

- Resource groups are a scope for applying **management actions**:
    - **Azure Role-Based Access Control (RBAC):** You can assign permissions at the resource group level. For instance, granting a developer "Contributor" access to a specific resource group means they can manage all resources *within* that group, but not resources outside it.
    - **Azure Policy:** You can assign Azure Policies to a resource group to enforce rules on all resources within it (e.g., "deny the creation of VMs larger than D4s_v3 in this resource group").
    - **Resource Locks:** You can apply locks (read-only or delete) to a resource group to prevent accidental deletion or modification of all resources within it.
    - **Tagging:** While tags can be applied to individual resources, applying them to a resource group can help categorize and organize resources effectively. (Note: resources *do not* automatically inherit tags from their resource group).

Billing and Cost Management:

- While billing happens at the subscription level, resource groups provide a great way to **track and analyze costs** associated with a particular application, environment, or project. Azure Cost Management tools allow you to filter and view costs by a resource group.

Location of Resource Group Metadata:

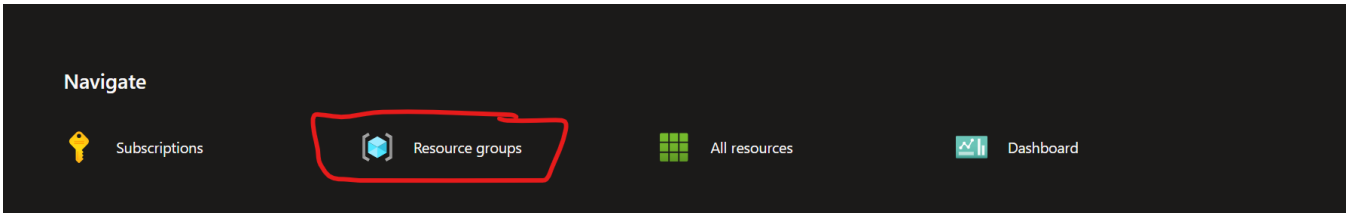
- A resource group itself has a **location** (region) associated with it. This location specifies where the metadata *(information about the resources, not the resources themselves)* for that resource group is stored.

- **Important:** The resources *within* a resource group can be located in different Azure regions than the resource group's metadata location. For example, you could have a resource group metadata in "East US" but contain VMs in "West US" and a SQL Database in "Europe North."

**[We will Study/see Zones & Regions later in the Guide] **

Every Resource Must Belong to a Resource Group:

- When you create any resource in Azure, you **must** place it into an existing resource group or create a new one for it. A resource cannot exist outside of a resource group.
- A resource can only belong to **one** resource group at a time, but it can be moved between resource groups (with certain limitations).



Here you can find all the resources which any user has created.

Azure Availability Zones

[Azure Region, Azure Sovereign Region, Azure Region Pair and Disaster Recovery]

Availability Zones Meaning- An Azure Availability Zone is a **physically and logically separate datacenter** within an Azure region. Each zone is designed to be isolated from failures in other Availability Zones.

Here's what that "isolation" means:

- **Independent Infrastructure:** Each Availability Zone has its own independent:
  - **Power Source:** Separate power grids and backup generators.

- **Network:** Distinct networking infrastructure.
- **Cooling:** Independent cooling systems.

What is **Azure Region?-**

An **Azure Region** is a fundamental geographical unit within Microsoft's global Azure infrastructure. It's a specific geographical area on the planet where Microsoft has deployed a cluster of one or more physical datacenters.

Think of an Azure Region as a large, self-contained cloud computing facility designed to provide reliable, low-latency access to Azure services for customers in that general area.

*"Below is the table of "Regions" where Microsoft has Deployed their **PHYSICAL** data Centers"*

Region	Availability Zones	Paired Region	Physical Location	Geography
Australia Central	No	Australia Central 2	Canberra	Australia
Australia Central 2	No	Australia Central	Canberra	Australia
Australia East	No	Australia Southeast	New South Wales	Australia
Australia Southeast	No	Australia East	Victoria	Australia
Austria East	Yes	n/a	Vienna	Austria
Brazil South	Yes	South Central US	São Paulo State	Brazil
Brazil Southeast	No	Brazil South	Rio de Janeiro State	Brazil
Canada Central	Yes	Canada East	Toronto	Canada
Canada East	No	Canada Central	Quebec	Canada
Central India	Yes	South India	Pune	India
Central US	Yes	East US 2	Iowa	United States
Chile Central	Yes	n/a	Santiago	Chile
East Asia	Yes	Southeast Asia	Hong Kong SAR	Asia Pacific
East US	Yes	West US	Virginia	United States
East US 2	Yes	Central US	Virginia	United States
France Central	Yes	France South	Paris	France
France South	No	France Central	Marseille	France

Germany North	No	Germany West Central	Berlin	Germany
Germany West Central	Yes	Germany North	Frankfurt	Germany
Indonesia Central	Yes	n/a	Jakarta	Indonesia
Israel Central	Yes	n/a	(Israel)	Israel
Italy North	Yes	n/a	Milan	Italy
Japan East	Yes	Japan West	Tokyo/Saitama	Japan
Japan West	No	Japan East	Osaka	Japan
Korea Central	Yes	Korea South	Seoul	Korea
Korea South	No	Korea Central	Busan	Korea
Malaysia West	Yes	n/a	(Malaysia)	Malaysia
Mexico Central	Yes	n/a	Querétaro State	Mexico
New Zealand North	Yes	n/a	Auckland	New Zealand
North Central US	No	South Central US	Illinois	United States
North Europe	Yes	West Europe	Ireland	Europe
Norway East	No	Norway West	(Norway)	Norway
Norway West	No	Norway East	(Norway)	Norway
Poland Central	Yes	n/a	Warsaw	Poland
Qatar Central	Yes	n/a	Doha	Qatar
South Africa North	Yes	South Africa West	Johannesburg	South Africa
South Africa West	No	South Africa North	Cape Town	South Africa
South Central US	Yes	North Central US	Texas	United States
South India	No	Central India	Chennai	India
Southeast Asia	Yes	East Asia	Singapore	Asia Pacific
Spain Central	Yes	n/a	Madrid	Spain
Sweden Central	Yes	Sweden South	Gävle	Sweden
Switzerland North	Yes	Switzerland West	Zurich	Switzerland
Switzerland West	No	Switzerland North	Geneva	Switzerland
UAE Central	No	UAE North	Abu Dhabi	UAE
UAE North	Yes	UAE Central	Dubai	UAE
UK South	Yes	UK West	London	United Kingdom
UK West	No	UK South	Cardiff	United Kingdom

West Central US	No	West US 2	Wyoming	United States
West Europe	Yes	North Europe	Netherlands	Europe
West India	No	South India	Mumbai	India
West US	No	East US	California	United States
West US 2	Yes	West Central US	Washington State	United States
West US 3	Yes	East US	Arizona	United States

In India-

Azure Region	Location (Approximate)	Availability Zones	Status
Central India	Pune, Maharashtra	Yes	Operational
South India	Chennai, Tamil Nadu	No	Operational
West India	Mumbai, Maharashtra	No	Operational

Now Understand-

Purpose of Availability Zones

They are designed for **high availability** and **resiliency**.
So if one zone (data center) fails, the other zones in that region **continue running**.

So, in Pune- (Central India), Microsoft Azure has deployed **at least 3 physically separate data centers**, known as **Availability Zones**.

So, what does this mean?

Each Availability Zone in Pune is:

- Hosted in a **separate building or data center**
- With **independent**:
  - Power supply
  - Cooling systems
  - Networking
- Connected through **high-speed, low-latency fiber links.**

Why this matter?

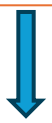
With **3 Availability Zones in Pune**, you can:

- Deploy apps with **high availability (99.99% SLA)**
- Use **Zone-redundant storage (ZRS)**
- Create **multi-zone load-balanced services** (e.g., in AKS, VM Scale Sets, SQL MI).

Now, what are Region Pair in Azure?

An Azure Region Pair is a "linked set of two Azure regions" within the same geography (e.g., India, US, Europe) that are chosen and maintained by Microsoft.

In Simple Terms:
"Region Pair = 2 Azure regions automatically linked for resiliency and disaster recovery"



If one region goes down *(due to natural disaster, outage, or planned update),* Azure services can **fail over** to the paired region.

Example-

Region	Paired Region	Geography
Central India (Pune)	South India (Chennai)	India
East US (Virginia)	West US (California)	United States
North Europe (Ireland)	West Europe (Netherlands)	Europe
Japan East (Tokyo)	Japan West (Osaka)	Japan

Meaning of example- If the Central India (Pune) Azure region fails, services can fail over to South India (Chennai) — but it depends on the service and how you've configured it.

Some Services automatically fail over and some don't.

Azure Sovereign Region?

An Azure Sovereign Region is a **special set of Azure data centers** designed to meet **specific national, legal, regulatory, or compliance requirements** of a country or government.
These regions are **isolated** from Microsoft's global Azure cloud and often operated **in partnership with or controlled by local governments or organizations**.

In-Short=> "" Azure Sovereign Cloud = A country-specific, isolated Azure environment built for government, defense, or regulated industries.""

Use Cases-

- National governments need **strict control over citizen data.**
- Defense or intelligence operations.
- Financial, telecom, or healthcare sectors with **strong compliance needs**
- Countries with **data localization** laws (e.g., India, Russia, China).

Disaster Recovery in Azure-

Disaster Recovery (DR) in Azure refers to the strategy, tools, and services used to **ensure your applications and data remain available** — or can be **quickly restored** — in the event of a **failure**, such as:

- Natural disaster (earthquake, flood)
- Data center outage
- Ransomware or cyberattack
- Human error (accidental deletion or misconfiguration)
- Hardware failure

When Does Disaster Recovery Come into Play?

Disaster Recovery is triggered when your primary Azure region, resource, or service becomes unavailable, and you need to fail over to a secondary region or backup environment to continue business operations.

Azure Services That Support Disaster Recovery:

Service	DR Feature / Support
Azure Site Recovery (ASR)	Real-time replication and failover of **VMs, servers**
Azure Backup	Snapshot and vault-based **file/system backup**
Azure SQL Database	Active geo-replication to other regions
Azure Storage (GRS/RA-GRS)	Geo-redundant storage replication
Azure Kubernetes (AKS)	Backed up with Azure Backup or redeployed via ARM
Azure App Service	Use Traffic Manager / Front Door for regional failover
Traffic Manager / Front Door	Global DNS-based routing and failover

Fun fact- *All the Azure Data Centers across the Globe are Interconnected with High Fiber Optics Backbone Cables.*

This backbone is not the public internet — it's Microsoft's own private global network.

It enables fast, secure, and reliable data transfer between regions, continents, and services.

Example:

You deploy your app in Central India (Pune) and replicate the database to East US (Virginia). That replication and all data movement happen over Microsoft's private backbone, not the open internet.

How Big Is Microsoft's Global Network?

200,000+ miles (320,000+ km) of subsea, terrestrial fiber.

60+ Countries Covered, 180+ Edge Nodes Globally.

Speed- 100+ Gbps

Management Groups in Azure

Azure Management Groups are containers used to organize and manage access, policies, and compliance across multiple Azure subscriptions in a hierarchical structure.

In-Short=> Think of **Management Groups** like folders in a computer.
You use them to **group multiple Azure subscriptions** together so you can **apply policies or RBAC** (Role-Based Access Control) to all of them **at once**.

Why Use Management Groups?

Use Case	Benefit
Have 5+ subscriptions	Organize them logically (e.g., by department, environment)
Apply governance policies (e.g., no public IPs)	Apply once at MG level, it flows down
Role assignments for many subscriptions	Assign once at MG, no need to do it 10 times
Centralized billing or compliance	Easier tracking and enforcement

*** We will learn about "containers" in later sections in this Guide****

Azure Compute and Networking Services

Azure Virtual machines-

Azure Virtual Machines (VMs) are one of the most fundamental and widely used services in Microsoft Azure. They provide on-demand, scalable computing resources in the cloud.

In essence, an Azure VM is a **"virtualized computer"** that you can create and run within Microsoft's global datacenter infrastructure. It behaves much like a physical computer you'd have in your office or a traditional datacenter, but all the underlying physical hardware is managed by Microsoft.

It is an Infrastructure as a service.

Basically, it is "Virtual" Computer, which User can Rent and do his/her work by selecting favorite specification. [Like RAM, CPU Cores, Storage etc.]

Now what does **VIRTUAL** mean?

“This means it's not a physical computer you can touch. Instead, it's a software-based version of a computer running on a giant, powerful physical computer in one of Microsoft's huge data centers (warehouses full of computers).”

Key Concepts and what exactly does it offers:

Infrastructure as a Service (IaaS): Azure VMs fall under the IaaS category of cloud computing. This means Microsoft provides and manages the underlying infrastructure (servers, networking, storage, virtualization layer), while you are responsible for:

- The **operating system** (Windows, Linux distributions like Ubuntu, Red Hat, CentOS, etc.)
- Any **applications** you install
- **Data**
- **Security configuration** within the OS
- **Networking configuration** (though Azure provides the virtual network infrastructure)

Full Control and Flexibility:

- **Operating System Choice:** You can choose from a vast marketplace of pre-configured OS images or even upload your own custom images.
- **Hardware Customization (Virtual):** You select the VM "size," which defines its virtual hardware specifications (number of vCPUs, amount of RAM, disk types/sizes, network performance). Azure offers various VM series optimized for different workloads (e.g., general purpose, compute-optimized, memory-optimized, storage-optimized, GPU-enabled, high-performance computing).
- **Networking:** You define its network configuration, including virtual networks, subnets, public IP addresses, network security groups (NSGs) for firewall rules, etc.
- **Storage:** You attach virtual disks (OS disk, data disks) for persistent storage. Azure Managed Disks simplify this by handling the underlying storage account management.

Scalability and Elasticity:

- **Scale Up/Down:** You can easily change the size of your VM (e.g., add more CPU or RAM) to meet changing performance demands without needing to migrate to new physical hardware.

- **Scale Out/In:** With **Virtual Machine Scale Sets**, you can create and manage a group of identical VMs that can automatically increase or decrease in number based on demand or a schedule. This is perfect for handling fluctuating traffic.
- **Pay-as-you-go:** You only pay for the compute resources you consume, typically by the minute or hour, reducing upfront capital expenditure. Reserved Instances can offer significant cost savings for predictable, long-term workloads.

High Availability and Resilience:

- **Availability Zones:** You can deploy VMs across multiple Availability Zones within an Azure region to protect against datacenter-level failures. For maximum resilience, you typically deploy multiple VMs in different zones behind a load balancer.
- **Availability Sets:** For regions without Availability Zones, or for grouping VMs within a single zone, Availability Sets can distribute your VMs across different fault domains (racks with independent power/network) and update domains (groups of VMs that are updated together) to maximize availability during planned or unplanned maintenance.
- **Managed Disks:** Azure manages the underlying storage accounts for your VM disks, providing better scalability and availability than unmanaged disks.

Integration with Azure Ecosystem:

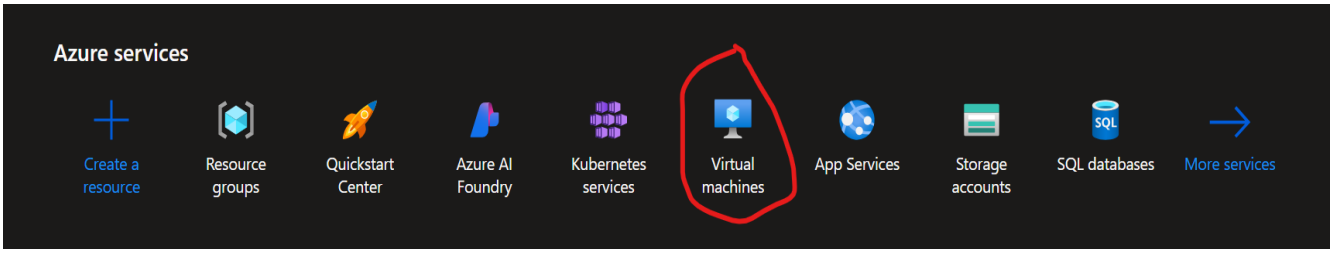
- VMs seamlessly integrate with other Azure services like Azure Networking (Virtual Networks, Load Balancers, VPN Gateway), Azure Storage, Azure Monitor (for monitoring and alerts), Azure Backup (for data protection), Azure Site Recovery (for disaster recovery), Azure Security Center, and Azure Active Directory (now Microsoft Entra ID).

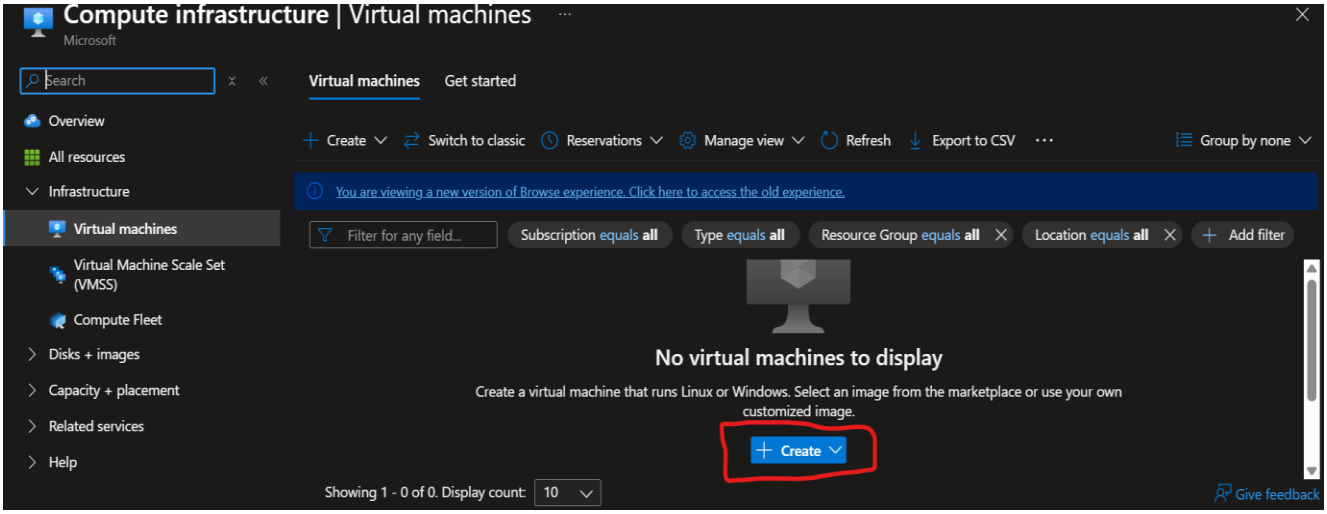
See the Common Use Cases, you will Understand and Importance of VM

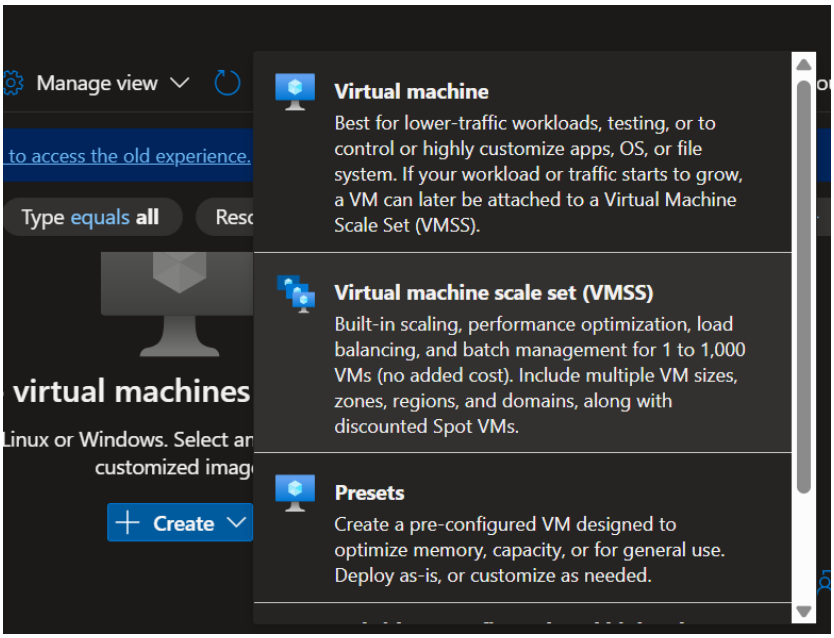
Use Case	Description	Why Azure VM?
1. Web/Application Hosting	Hosting enterprise web apps, APIs, or services (IIS, .NET, Java, etc.)	Full control over OS, scaling, and deployment stacks
2. Database Hosting	Running SQL Server, Oracle, MySQL, PostgreSQL, etc.	High IOPS disks, memory-intensive VM sizes, hybrid licensing support

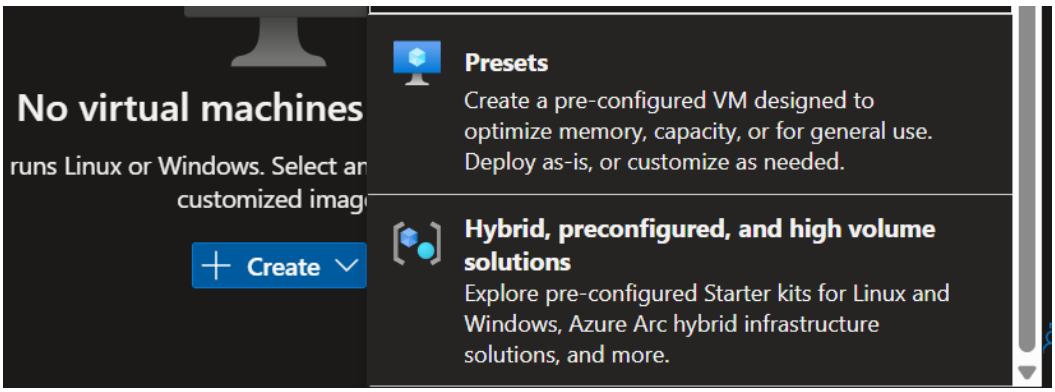
3. SAP Workloads	Running SAP HANA, NetWeaver, S/4HANA workloads	Certified SAP HANA infrastructure (E-series, M-series VMs)
4. Dev/Test Environments	Quick provisioning of environments for development or QA	Fast setup, low-cost VMs, and automation via ARM templates
5. Lift-and-Shift Migration	Moving on-prem legacy workloads to the cloud (rehosting)	Mimics on-prem setup with minimal refactoring
6. Backup & Disaster Recovery (DR)	Cold standby environments or replicated VMs using Azure Site Recovery	Reliable failover, pay-only-when-used VMs
7. Remote Desktop Services (RDS)	Hosting Windows Virtual Desktop or RDS workloads	Scalable remote workforce solutions with GPU-enabled VM options
8. Batch Processing / HPC	Running scheduled tasks, analytics jobs, simulations, etc.	Burstable compute, spot VMs, HPC-optimized VM types
9. Network Appliances	Deploying firewalls, routers, proxies (e.g., FortiGate, Cisco, Palo Alto)	VM Marketplace images with BYOL or pay-as-you-go models
10. Custom Containers / Docker Hosts	Hosting custom container workloads without AKS	VMs give OS-level control for custom container runtimes
11. Application Servers for ERP/CRM	Hosting backend servers for apps like Dynamics, Oracle ERP	Flexible scaling, hybrid network integration
12. File & Print Servers	Hosting traditional AD file shares, SMB/NFS mounts, or print queues	Legacy compatibility with existing enterprise infra
13. Domain Controllers	Deploying Active Directory domain services (ADDS) in hybrid mode	Full control over AD forests and replication
14. Security & Forensics Labs	Isolated environments for testing malware, EDR, SIEM tools	Sandbox-style VMs with custom firewall rules

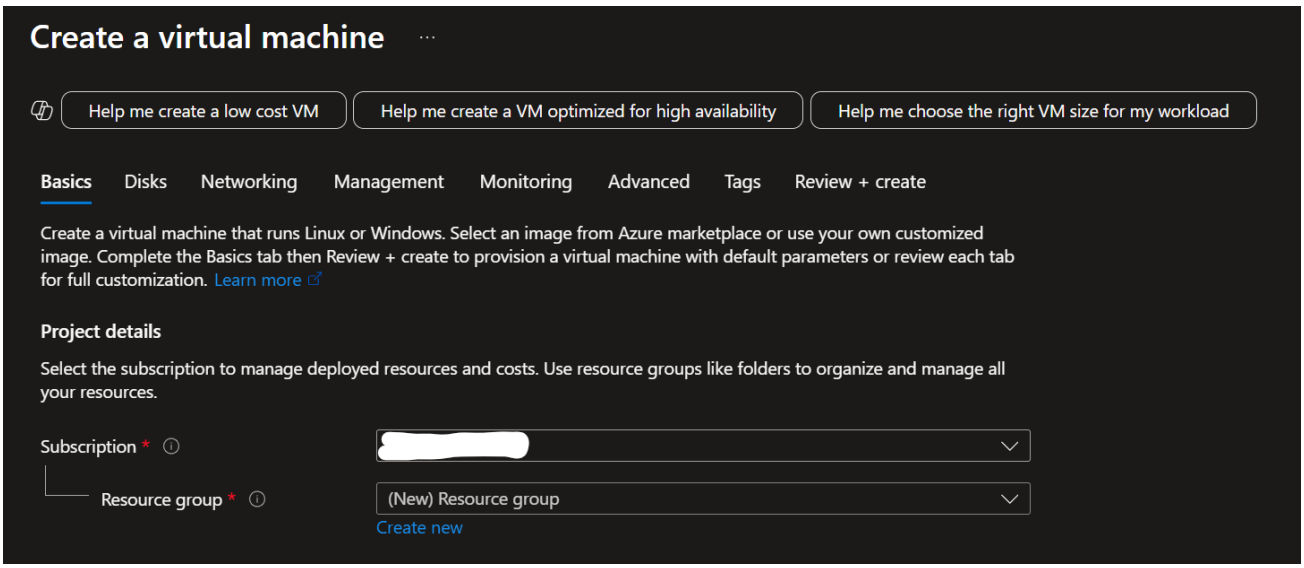
15. AI/ML Model Training	GPU-based VMs for training ML/DL models (N-series VMs)	Access to powerful GPU resources for compute-heavy training tasks
16. VPN or Bastion Hosts	Securely bridging on-prem and Azure via VPN or jump-boxes	Can be hardened and segmented for admin access only



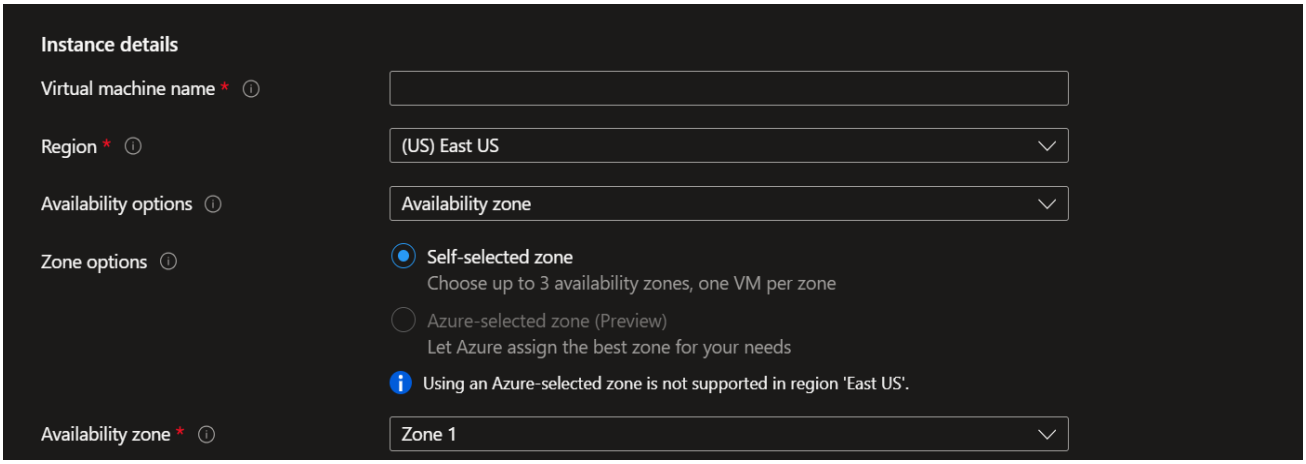


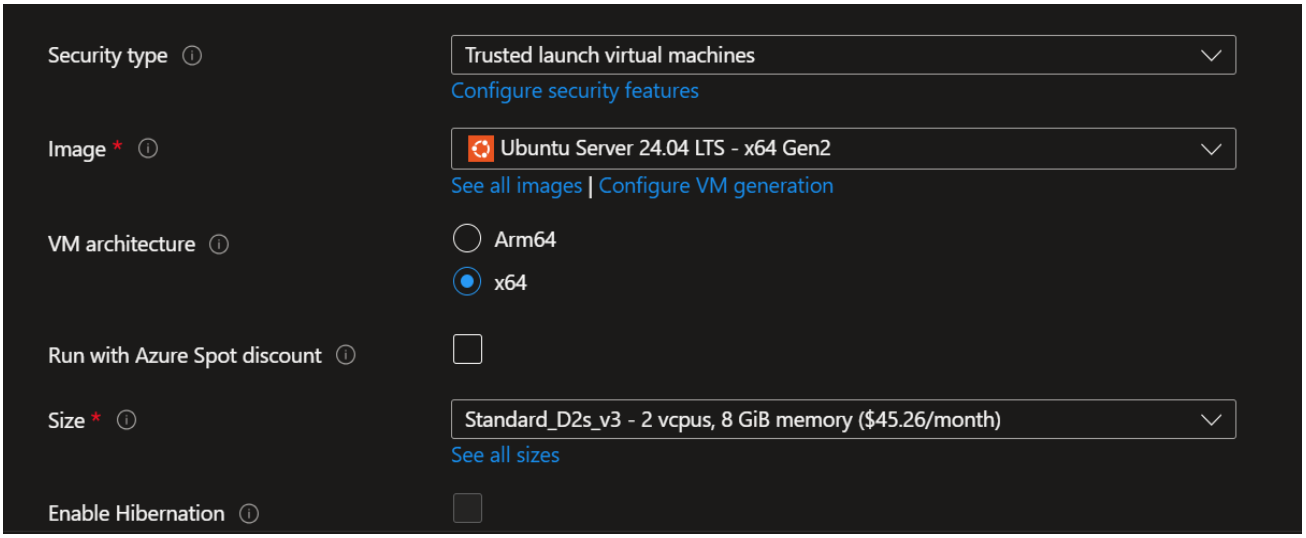


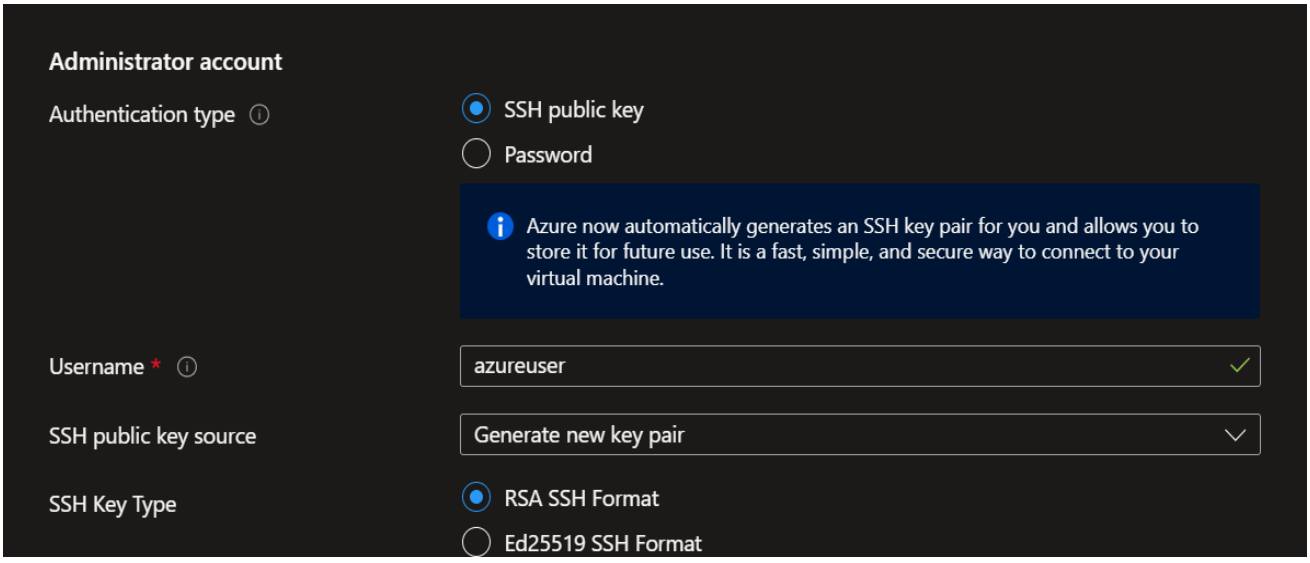


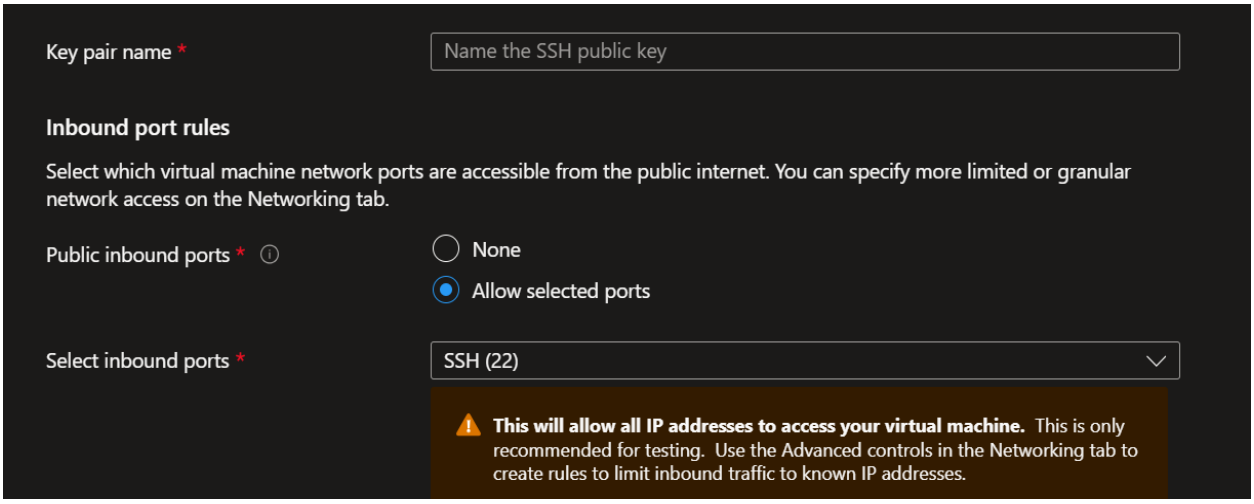


https://www.linkedin.com/in/abhishek-chaudhary-48997510b/


Instance details
Virtual machine name *
Region *
(US) East US
Availability options
Availability zone
Zone options
Self-selected zone
Choose up to 3 availability zones, one VM per zone
Azure-selected zone (Preview)
Let Azure assign the best zone for your needs
Using an Azure-selected zone is not supported in region 'East US'.
Availability zone *
Zone 1


Security type
Trusted launch virtual machines
Configure security features
Image *
Ubuntu Server 24.04 LTS - x64 Gen2
See all images | Configure VM generation
VM architecture
Arm64
x64
Run with Azure Spot discount
Size *
Standard_D2s_v3 - 2 vcpus, 8 GiB memory ($45.26/month)
See all sizes
Enable Hibernation


Administrator account
Authentication type
SSH public key
Password
Azure now automatically generates an SSH key pair for you and allows you to store it for future use. It is a fast, simple, and secure way to connect to your virtual machine.
Username *
azureuser
SSH public key source
Generate new key pair
SSH Key Type
RSA SSH Format
Ed25519 SSH Format



📎 **Note-** While Creating the VM, there is an Option for **"Azure Subscription".**

What is this Azure Subscription? Is it project Specific or Enterprise Specific.

Think of an **Azure Subscription** as your **contract or agreement with Microsoft for using their cloud services.**

It's the fundamental **billing and access control boundary** for all the resources you create in Azure.

Project Specific or Enterprise Specific?

The answer is **both**, and it largely depends on the **size and needs of the organization**.

Enterprise-Specific (Often the starting point):

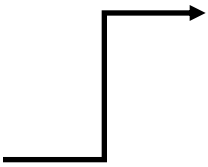
- **Overall Agreement:** For a large company, the Azure subscription often begins as part of a larger **Enterprise Agreement (EA)** with Microsoft. This is a master agreement that covers all of Microsoft's software and cloud services for the entire organization, often with discounted rates.
- **Centralized Billing:** In many enterprises, there's a central IT or finance department that manages the primary billing account, and then they create multiple subscriptions under that account.
- **Top-Level Governance:** Policies and security standards are often set at the highest level (Management Groups) which then trickle down to all subscriptions within an enterprise's Azure environment.

2. Project-Specific (Common for organization and management):

While a large organization might have an overarching Enterprise Agreement, it's very common and **best practice** to then create **multiple subscriptions** for different purposes within that enterprise. This is where "project-specific" comes in:

- **By Environment:**
  - Subscription-Dev (for all development projects)
  - Subscription-Test (for all testing/QA)
  - Subscription-Prod (for all production workloads)
  - This provides strong isolation, prevents developers from accidentally impacting production, and allows different budget allocations.
- **By Department/Business Unit:**
  - Subscription-Marketing
  - Subscription-Finance
  - Subscription-HR
  - This helps with chargebacks and budget accountability.
- **By Application/Workload:**
  - Subscription-CRM-App
  - Subscription-ERP-System
  - This makes it clear which costs and resources belong to which major application.
- **To Manage Quotas/Limits:** Azure has certain service limits (quotas) per subscription. For very large deployments, you might create multiple subscriptions simply to get around these limits and scale out more effectively.
- **For Specific Billing Offers:** You might have a "Dev/Test" subscription type which offers discounted rates for non-production workloads, separate from your "Pay-As-You-Go" or "Enterprise Agreement" production subscriptions.
- **For Isolation and Security:** High-security workloads (e.g., those handling sensitive financial data) might be placed in their own isolated subscription with very strict access controls and policies.

Now, understand the Hierarchy in Azure.



1- **Management Groups (Optional, for large enterprises):** The highest level. Used to group multiple subscriptions together for centralized governance and policy application.
2- **Subscriptions:** The core billing and access control boundary.
3- **Resource Groups:** Logical containers within a subscription for related resources of a single application/project.
4- **Resources:** The actual services (VMs, databases, etc.).

So, when you create a VM, you *must* choose which Subscription it belongs to, because that choice determines:

1. **Who gets billed** for that VM's usage.
2. **Who has access** to manage that VM (based on permissions at the subscription level or inherited from a Management Group).
3. **Which policies** apply to that VM.

Virtual machine Scale Set (VMSS) and Presets

Imagine you have a popular website or an app that gets a lot of visitors. Sometimes you have few visitors, but other times, you suddenly have thousands!

[Think of any e-commerce website/app during Diwali or any festival time, millions of users hit on the website] 

- **The Problem:** If you only have one virtual machine (VM) running your website, it might get overwhelmed and slow down or crash when too many people visit. If you buy a super powerful VM just in case, you're paying for power you don't always need.

Now the Solution is VMSS!!

VMSS is like having a **smart, automated team of identical virtual machines.**

Working of VMSS-

1. **Identical Copies:** You create a blueprint for your perfect VM (what operating system, what software, how much memory, etc.). The VMSS then creates *many identical copies* of that VM.
2. **Automatic Scaling (Auto-scaling):** This is the magic part!
    - You set rules: "If the CPU usage on my VMs goes above 70% for 5 minutes, add 2 more VMs to the team."
    - You also set rules for when demand goes down: "If CPU usage drops below 30% for 10 minutes, remove 1 VM from the team."
    - The VMSS automatically adds or removes VMs based on these rules, or even on a schedule (e.g., add more at 9 AM when everyone starts work, remove them at 5 PM).
3. **Load Balancing:** The VMSS works with an Azure Load Balancer. This is like a traffic controller that makes sure incoming requests (like website visitors) are spread evenly across all the VMs in your team, so no single VM gets overloaded.
4. **Easy Management:** Instead of managing 10 or 100 individual VMs, you manage the *single VMSS*. When you need to update software or change a setting, you update the VMSS, and it applies that change to all the VMs in the set automatically.
5. **High Availability:** If one VM in the set fails or needs maintenance, the others are still running, ensuring your application stays available to users. The load balancer just stops sending traffic to the unhealthy VM.

When to use VMSS:

- Web servers that experience fluctuating traffic.
- Batch processing jobs.
- Container orchestration (like Azure Kubernetes Service often uses VMSS for its worker nodes).
- Any stateless application (where user data isn't stored directly on the VM itself, but perhaps in a database or central storage).

*** We will study Kubernetes later in the guide. ***

Presets

"Presets" in the context of Azure VM creation primarily refer to predefined "VM Sizes" or "Series" that bundle together specific amounts of vCPUs (virtual processors), RAM (memory), and other capabilities.

Microsoft has thousands of different VM sizes to choose from, each optimized for different kinds of work. To make it easier, they group these sizes into families or "presets" based on their typical use cases:

When you select one of these "presets" or "families" in the Azure Portal, it then narrows down the specific VM sizes you can choose from within that category *(e.g., "D2s_v3" which might have 2 vCPUs and 8GB RAM, or "D4s_v3" with 4 vCPUs and 16GB RAM).*

So, "presets" just help you quickly find the right type of VM for your workload without having to become an expert on every single VM size available.

Azure Virtual Desktop: -

As the name suggests- It is a comprehensive **desktop and application virtualization service** that runs entirely on Microsoft Azure. In simple terms, it allows organizations to deliver full **Windows desktop experience and specific applications to users from anywhere, on any device**, securely over the internet.

Imagine- Think of it as having your work computer, with all its programs and files, available to *you on a tablet at home, a laptop at a coffee shop, or even just a web browser* – but the actual "computer" is running securely in Microsoft's cloud.

So, what problem does AVD solve?

Traditionally, providing remote access to desktops and applications was complex, costly, and resource intensive. It involved setting up and managing:

- Physical servers for Virtual Desktop Infrastructure (VDI).
- Licensing for various Windows and Remote Desktop Services components
- Gateways, brokers, load balancers, and monitoring tools

- Security and disaster recovery for all these components

AVD simplifies this by **offloading much of the infrastructure management to Microsoft**, while giving organizations robust, scalable, and secure remote desktop capabilities.

Key Features and Concepts (Deeper Dive-)-

1. **Windows 10/11 Enterprise Multi-Session:**
    - This is a groundbreaking feature unique to AVD. Traditionally, a standard Windows 10/11 desktop could only be used by one person at a time (single session).
    - AVD allows **multiple users to concurrently connect to and use the same Windows 10 or 11 virtual machine** (VM). This significantly reduces the number of VMs and thus costs, making it more efficient than traditional VDI.
    - It offers the full desktop experience of Windows 10/11, unlike Windows Server-based remote desktops which might have compatibility issues with some applications designed for desktop OS.

2. **Host Pools:**
    - A host pool is a collection of identical Azure Virtual Machines (VMs) that act as **session hosts** (the actual virtual desktops or app hosts that users connect to).
    - You define the image for these VMs (e.g., Windows 11 with specific apps pre-installed).
    - **Pooled (or shared) Desktops:** This is the most common and cost-efficient type. Users are assigned to any available VM in the pool, and their sessions are typically non-persistent (changes are discarded after logout, though user profiles are managed separately). This leverages multi-session Windows 10/11.
    - **Personal (or dedicated) Desktops:** Each user is assigned a specific VM that they always connect to. This allows for customization and persistence, similar to a physical desktop. Ideal for developers or users with unique software needs.

3. **App Groups:**
    - Within a host pool, you can create "app groups" to publish specific applications (RemoteApps) instead of full desktop. For example, you can publish just Microsoft Word, a specific CRM application, or a legacy app.
    - Users see these applications as if they were running locally on their device.

4. **Workspaces:**
    - A workspace is a logical grouping of one or more host pools and their associated app groups.
    - Users subscribe to a workspace and then see all the desktops and applications available to them within that workspace in their AVD client feed.

5. **Microsoft-Managed Infrastructure:**
    - This is a huge benefit. Microsoft manages the "control plane" components that are traditionally complex in VDI environments. This includes:
        - **Web Access:** The web portal users connect to.
        - **Gateway:** Securely connects remote users to session hosts using "reverse connect" technology, meaning no inbound firewall ports are needed on your VMs.
        - **Connection Broker:** Assigns users to available session hosts.
        - **Diagnostics:** Tools for monitoring and troubleshooting the service.
        - **Extensibility Components:** APIs for automation.
    - You, as the customer, only manage the **session host VMs themselves, user identities, and application data**. This significantly reduces IT overhead.

6. **FSLogix Profile Containers:**
    - A critical component for managing user profiles in pooled desktop environments.
    - FSLogix redirects entire user profiles (including user data, app settings, and cached Outlook data) to a virtual disk file (VHD/VHDX) stored on a central shared location (like Azure Files or Azure NetApp Files).
    - When a user logs in, their profile disk is dynamically attached to whichever session host VM they land on. This provides “persistent" user experience even on "non-persistent" pooled desktops.

7. **Integration with Microsoft Entra ID (formerly Azure Active Directory):**
    - AVD uses Microsoft Entra ID for identity and access management. This enables:
        - **Single Sign-On (SSO):** Users can access their virtual desktops and apps seamlessly after logging in once.
        - **Multi-Factor Authentication (MFA):** Enhances security by requiring more than just a password.
        - **Conditional Access:** Define policies to control access based on user location, device compliance, etc.
8. **Broad Client Support:**

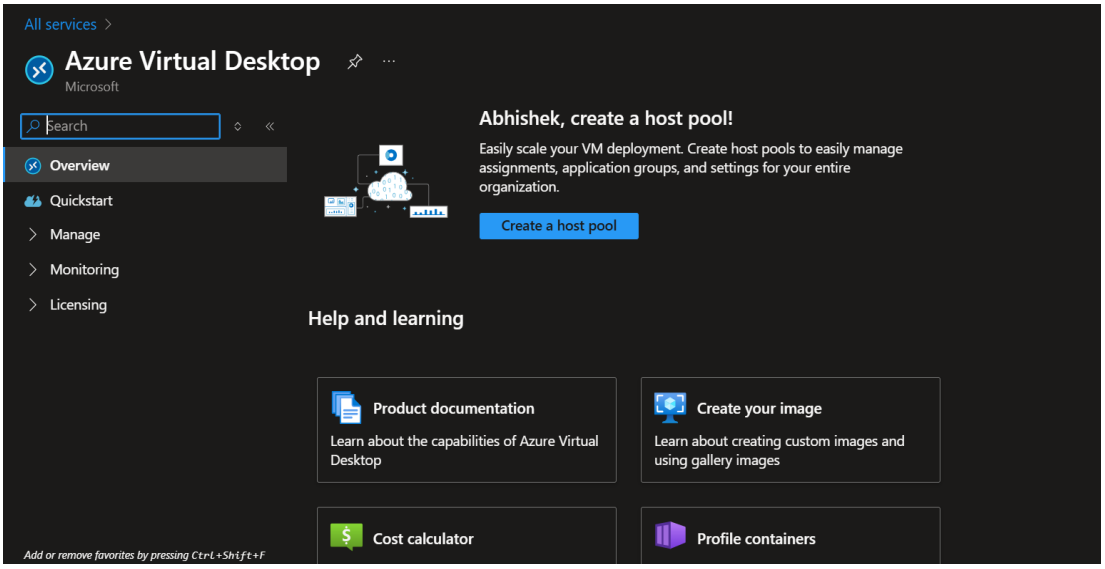
    Users can access AVD desktops and apps from almost any device:
    - Windows PCs (native Remote Desktop client, Windows app)
    - macOS
    - iOS (iPhones, iPads)
    - Android devices
    - Linux
    - Any HTML5-compatible web browser (no client installation needed)
9. **Security:**
    - Leverages Azure's robust security capabilities.
    - **Reverse Connect:** As mentioned, inbound ports aren't needed, reducing attack surface.
    - Integration with Microsoft Entra ID for identity security.
    - Data remains in the Azure cloud, not on endpoint devices, reducing data leakage risk.
    - Compliance with various industry regulations.
10. **Cost Optimization:**
    - **Multi-session Windows 10/11:** Reduces VM count and associated compute costs.
    - **Flexible VM sizes:** Choose the right VM size for your workload.
    - **Autoscaling:** Integrate with Azure Automation or third-party tools to automatically start/stop/scale session hosts based on user demand, saving compute costs during off-peak hours.

- **Licensing:** User access rights for AVD are included with eligible Microsoft 365 or Windows licenses (e.g., Microsoft 365 E3/E5, Windows 10/11 Enterprise E3/E5). You only pay for the underlying Azure compute, storage, and networking resources consumed by your session hosts.

Architecture Overview:

At a high level, the AVD architecture involves:

- **User Devices:** PCs, Macs, tablets, phones accessing through the AVD client or web browser.
- **Azure Virtual Desktop Service (Microsoft Managed):** This is the control plane that includes Web Access, Gateway, Broker, and Diagnostics services.
- **Customer-Managed Components in Azure:**
  - **Session Hosts:** Azure VMs running Windows 10/11 Enterprise multi-session, Windows 10/11 single-session, or Windows Server.
  - **Virtual Network:** Connects session hosts to your corporate network (on-premises via VPN/ExpressRoute or other Azure resources).
  - **Storage for FSLogix Profiles:** Azure Files (often with Azure AD DS or AD DS authentication) or Azure NetApp Files.
  - **Identity Service:** Microsoft Entra ID, typically synchronized with on-premises Active Directory Domain Services (AD DS) using Microsoft Entra Connect



Common Use cases of AVD-

Use Case	Description	Why AVD?
1. Remote Workforce Enablement	Securely provide full Windows desktops to remote employees or contractors	Access from anywhere, centralized control, secure data
2. Bring Your Own Device (BYOD)	Let employees use personal devices without storing data locally	Data stays in Azure, not on unmanaged devices
3. App Virtualization	Deliver specific apps (e.g., ERP, HRMS, custom tools) without full desktops	Publish apps only, reduce overhead and licensing
4. Compliance-Sensitive Workloads	Industries like finance, healthcare, or government requiring tight data control	Keeps sensitive data off endpoints; supports HIPAA, ISO, GDPR, etc.
5. Temporary Workers or Projects	Provision desktops for interns, temps, or short-term consultants	Easy to scale up/down without buying laptops or licenses
6. Call Centers or BPO Operations	Agents access a unified desktop environment with softphone, CRM, etc.	Standardized desktops with session persistence
7. Education and Labs	Provide student labs or training desktops remotely	Uniform desktop environments, easy reset between sessions

8. Developer or Test Environments	Give dev teams Windows/Linux desktops with specific tools or versions	Isolated environments, image versioning, no hardware dependency
9. Legacy App Hosting	Run older Windows apps that aren't cloud-native	Centralized access via AVD, even from modern devices
10. Disaster Recovery for Desktops	Failover user environments in case of local IT failures	Cloud-based desktops remain accessible even if local infra fails

Containers *[Beautiful and Fascinated Concept]* ❤️

Ok, imagine you're shipping different kinds of cargo around the world: "electronics, furniture, perishable food," etc. Each need specific handling but dealing with each item individually would be a nightmare.

Traditional method: Like putting loose items directly on a truck. Every truck might be different, and you have to worry about how each item will fit and if it will get damaged.

With Shipping Containers: You put all the necessary items for one type of cargo into a **standardized, sealed shipping container**. ←

- This container has everything the cargo needs (like temperature control for food).
- The **container itself is the same size and shape**, no matter what's inside.
- It can be easily loaded onto any truck, train, or ship that's designed for these standard containers.

Software Containers are exactly like this!

- **Your application (code):** This is your cargo (the electronics, the furniture).
- **Dependencies (libraries, settings):** These are the special instructions or tools the cargo needs (like the temperature control system for food).
- **A Container:** This is a **standardized, lightweight, self-contained package** that bundles your application code **along with everything it needs to run** (libraries + dependencies + configuration).

What are Containers in Azure? 

A **container** in Azure is a **lightweight, portable, and isolated environment** used to run applications and their dependencies — similar to virtual machines, but **without the overhead of a full OS**.

Key Characteristics:

- Lightweight and fast to start.
- Portable across environments (dev, test, prod).
- Runs the same regardless of where it's deployed.
- Ideal for **microservices, APIs, and isolated workloads.**

Understand the Need for Containers with the below Example !

Real-Life Example: Financial Services Company Using Containers in Azure

- The company manages a **customer-facing loan application portal**.
- Originally, it was hosted as a **monolithic application on Azure VMs**.
- Any small change (e.g., updating the loan interest calculator) requires **re-deploying the entire application**, causing **downtime** and **deployment risk**.

Now, After Implementing Containers

- The development team **containerized** the application using **Docker**.
- They **broke the monolith** into individual services:

- 📊 **Loan Calculator**
- 🔐 **User Authentication**
- 📁 **Document Upload Service**
- 📧 **Notification Services**

Each service was packaged in its **own container** with required runtime, libraries, and tools.

Deployment in Azure

- Containers were deployed using **Azure Kubernetes Service (AKS)**:

  - Ensured **auto-scaling**, **self-healing**, and **zero-downtime deployments**.
  - Allowed **independent scaling** of services *(e.g., more instances of the upload service during peak load).*
- Container images were stored in **Azure Container Registry (ACR)**:
  - Integrated with **CI/CD pipeline** (e.g., GitHub Actions or Azure DevOps).
  - Automated builds pushed updated images to ACR after code changes.

Key Results

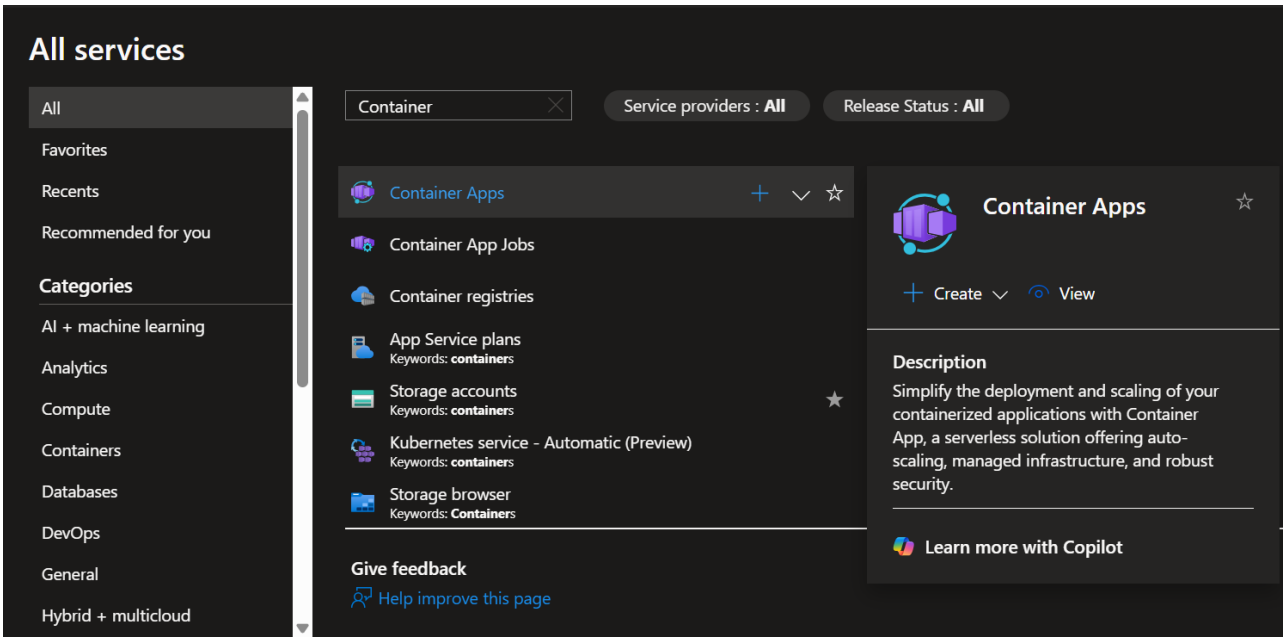
- Faster deployments with **no need to restart the whole app**.
- Improved uptime and **high availability**.
- **Better resource usage** — scale only what is needed.
- **Consistent environments** across dev, test, and production.
- Better **security** and **isolation** of microservices.

Quick Doubt- Then what is the difference between Virtual machine & Containers.

Feature	Azure Virtual Machine (VM)	Azure Container
Definition	Full emulation of a physical machine in Azure	Lightweight, portable package with app + dependencies
Boot Time	Minutes	Seconds
Operating System	Full OS (Windows or Linux)	Shares host OS kernel; no full OS
Resource Overhead	High — each VM includes full OS	Low — shares host OS, minimal overhead
Use Case Fit	Traditional apps, legacy workloads, databases	Microservices, APIs, stateless apps
Isolation Level	Strong — each VM runs separately	Medium — isolated user space but shared kernel
Scaling	Slower — requires provisioning of full VM	Fast — containers can be scaled rapidly
Deployment Speed	Slow (OS and environment setup required)	Fast (prebuilt images can be deployed instantly)

Portability	Lower — VMs are tied to platform/hardware	High — container runs anywhere Docker/Kubernetes is supported
Cost Efficiency	Less efficient — higher compute cost due to full OS	More efficient — optimized use of compute resources
Management Complexity	Higher — needs patching, updates, OS management	Lower — minimal OS management, easier automation
Storage Requirements	GBs (OS + App)	MBs (only app + dependencies)
Examples in Azure	Azure VMs, Azure VM Scale Sets	Azure Kubernetes Service (AKS), Azure Container Instances (ACI)
Best Suited For	Legacy apps, databases, desktop apps, Windows services	Cloud-native apps, microservices, CI/CD, dev/test workloads

"VM = Full Office ||| Container = Boxed Desk with Just What You Need"



Azure Container Instances (ACI)

Imagine you have an application packaged inside a tiny, self-contained box called a "container" (like a Docker container). You want to run this box in the cloud, but you don't want to deal with setting up and managing a whole computer (virtual machine) just for that one box.

ACI is like a magic button that lets you instantly run your containerized application in Azure without having to worry about servers, virtual machines, or complex orchestration tools. You simply tell Azure what container image to use, how much CPU and memory it needs, and ACI handles all the underlying infrastructure for you. You only pay for the resources your container uses, down to the second.

It's perfect for quickly deploying small, bursty workloads, running short-lived tasks, or testing new containerized applications without any overhead.

Technical Definition-

- Azure Container Instances (ACI) is a **serverless Platform-as-a-Service (PaaS)** offering from Microsoft Azure that enables the rapid deployment and execution of isolated Docker containers on-demand.
- It provides a lightweight and efficient way to run containers without the need to provision, manage, or scale underlying virtual machines or orchestrators like Kubernetes.

Deeper Breakdown->

- ✓ **Serverless:** This means you don't manage any servers or virtual machines. Azure abstracts away the underlying infrastructure, allowing you to focus purely on your application code and container image.
- ✓ **Container Group as the Primary Resource:** The fundamental deployment unit in ACI is a **container group**. A container group is a collection of containers that share the same host, network, storage, and lifecycle. This is analogous to Kubernetes Pod.
- ✓ **Hypervisor Isolation:** ACI provides strong security and isolation by running each container group with hypervisor-level isolation. This ensures that your containers are as isolated as they would be in a virtual machine, preventing them from impacting other tenants' workloads.
- ✓ **On-demand Compute:** ACI offers fast startup times (seconds) and elastic scaling. You can specify precise CPU cores and memory allocations for your containers, and Azure dynamically provisions these resources.

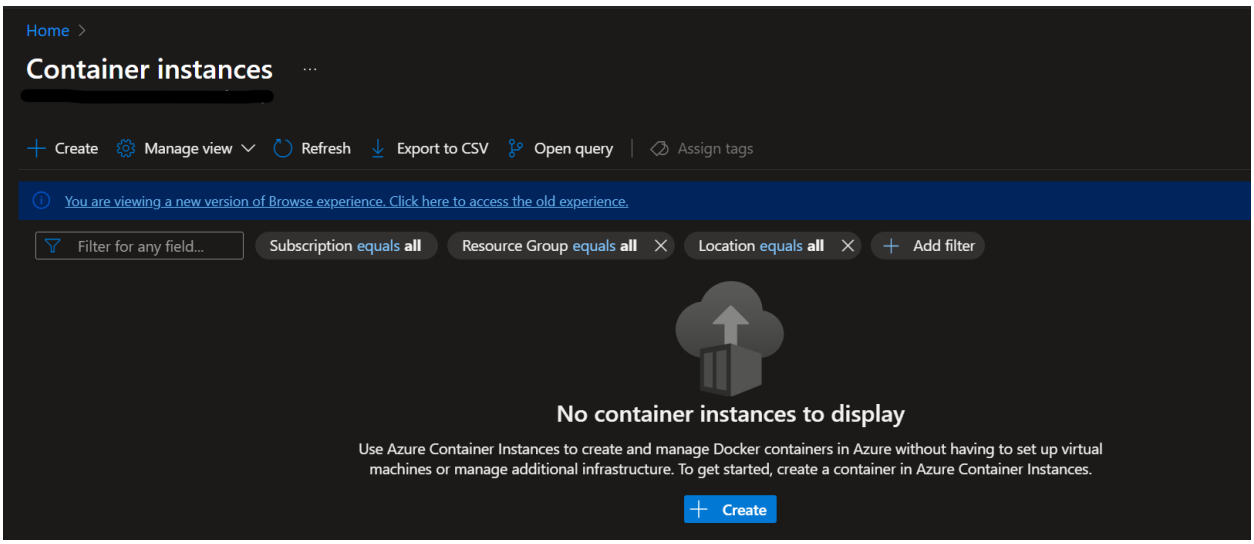
- ✓ **Consumption-Based Billing:** Billing for ACI is granular and based on the actual CPU and memory resources consumed by your container groups, billed by the second. This makes it cost-effective for intermittent or variable workloads.
- ✓ **Support for Linux and Windows Containers:** ACI supports both Linux and Windows-based Docker container images.
- ✓ **Integration with Azure Services:** ACI seamlessly integrates with other Azure services, such as Azure Container Registry (for storing container images), Azure Virtual Networks (for secure private networking), Azure Files (for persistent storage), and Azure Monitor (for logging and metrics).

Common Use Cases-

Use Case	Description	Why ACI is a Good Fit
1. Quick App Prototyping	Deploy apps quickly without setting up infrastructure	No VM or Kubernetes cluster needed — run containers instantly
2. Serverless Batch Jobs	Run background tasks like file conversions, ETL, report generation	Pay-per-second billing; container runs and shuts down after job completes
3. API/Microservice Hosting	Host lightweight APIs or stateless microservices temporarily	Great for temporary, isolated workloads without full cluster complexity
4. Event-Driven Tasks	Trigger containers based on events (e.g., blob upload, queue message)	Easily integrates with Logic Apps, Event Grid, or Azure Functions
5. CI/CD Pipeline Execution	Use containers to perform build/test steps as part of DevOps workflows	Isolated, consistent build environments across pipeline runs
6. Isolated Sandbox Testing	Run code in a safe, throwaway environment for testing or security analysis	Spin up an ACI container in seconds with no long-term resources
7. Lightweight Web Apps	Host basic frontend apps, dashboards, or portals temporarily	Fast to deploy, no VM management
8. Disaster Recovery Scripts	Run DR tools or scripts after an outage	Deploy instantly when needed, without idle costs
9. Data Processing Pipelines	Run a container to clean, validate, or transform data on a schedule	Integrates well with Azure Data Factory and Logic Apps
10. Multi-Tenant App Isolation	Deploy separate containers per user/customer in SaaS apps	ACI provides isolated runtime per instance without managing Kubernetes

Key Strengths of ACI:

- No infrastructure to manage (fully serverless)
- Instant startup (within seconds)
- Billing per second
- Secure and isolated runtime
- Supports both Linux and Windows containers



Azure Container Apps-

Azure Container Apps (ACA) is a fully managed serverless platform designed for running containerized applications, particularly those built around microservices, event-driven architectures, and background tasks.

While it is built on top of Kubernetes, it **abstracts away the complexities of Kubernetes management**, allowing developers to focus on writing code rather than managing infrastructure.

Features-

- **Serverless and Scalable:** ACA automatically scales your containerized applications up and down based on various triggers like HTTP traffic, events (using KEDA - Kubernetes Event-driven Autoscaling), CPU/memory load, or custom metrics. It can even scale down to zero instances when there's no traffic, minimizing costs.

- **Microservices-Oriented:** ACA is optimized for microservices. It has built-in support for Dapr (Distributed Application Runtime), which simplifies building resilient, portable microservices by providing capabilities like service-to-service communication, state management, pub/sub messaging, and more.
- **Managed Environment:** You deploy your container apps into a "Container Apps environment," which provides a secure perimeter for your applications. This environment handles networking, logging, monitoring, and security for all the container apps within it.
- **Revision Management:** ACA supports a revision model, meaning that every time you make a configuration change to your container app, a new immutable revision is created. This allows for easy rollbacks, traffic splitting (e.g., A/B testing, blue/green deployments) across different revisions, and managing the application lifecycle.
- **Ingress and Traffic Management:** It offers built-in Layer 7 ingress, allowing you to easily expose your applications to the internet. It also provides features for traffic splitting, enabling you to route incoming requests to different revisions of your application.
- **Integrated Observability:** ACA integrates with *Azure Monitor and Log Analytics* for comprehensive logging, metrics, and monitoring of your containerized applications.
- **Support for Various Workloads:** While great for microservices, ACA can also host simple web applications, API endpoints, and long-running background jobs.
- **Supported Container Images:** It supports Linux-based x86-64 (Linux/amd64) container images from any public or private registry (like Azure Container Registry or Docker Hub).

Now, How Container Instances and Container Apps are related?

Think of ACA as a smarter, auto-managing layer built on top of core capabilities like ACI.

Meaning-> ACI gives you the fundamental power to run a container, ACA provides a highly advanced, automated, and feature-rich platform that leverages that fundamental power to deliver a complete, resilient, and scalable solution for modern cloud-native applications, without requiring you to become a Kubernetes expert or manage the underlying infrastructure.

Understand with Example->

Scenario: Banking Application

A bank wants to build two systems:
{

Use Case 1: Simple Daily Report Processor

- Runs once a day, processes customer data, generates PDF.
- **Container image** is built and uploaded to ACR.
- Needs to run **only once a day**, no user interaction.

Best Fit: **Azure Container Instances (ACI)**

- They schedule it via Azure Logic App or Automation to start the container.
- Once the job is done, the container shuts down — no idle cost.

}

Use Case 2: Customer Loan API Service

- Exposed via HTTP to customers applying for loans.
- Needs to be **highly available**, **scale up during traffic surges**, and support **rolling updates**.

Best Fit: **Azure Container Apps (ACA)**

- They deploy the app container to ACA.
- ACA auto-scales based on HTTP traffic.
- Provides HTTPS endpoint, auto-revisions, built-in traffic splitting.

Azure Kubernetes Services

First Understand the term- "Kubernetes"-> It is an **open-source container orchestration system** that automates the deployment, scaling, and management of containerized applications.

Feature	Description
Fully Managed Control Plane	Azure handles the Kubernetes master nodes, upgrades, and patching
Auto-Scaling (Cluster + Pod)	Automatically adds/removes nodes and scales app pods based on demand
Integrated Monitoring	Built-in with **Azure Monitor**, **Log Analytics**, and **Container Insights**
RBAC & Azure AD Integration	Role-based access control, integrated with Azure Active Directory
DevOps Support	Integrates with **Azure DevOps**, **GitHub Actions**, **Helm**, **Kustomize**
Networking	Supports Azure CNI, custom DNS, VNet integration, private clusters
CI/CD Pipelines	Automate builds and deployments using pipelines
Secrets & Config Management	Manage app secrets using **Azure Key Vault** or Kubernetes secrets
Persistent Storage	Attach Azure Disks, Files, or Blob storage to containers
GPU Support	Run ML/DL workloads using GPU-powered nodes (N-series VMs)
Compliance & Security	SOC, HIPAA, ISO certified; supports Pod Security Policies, Network Policies

Key Concepts in Kubernetes:

Pods: The smallest deployable unit in Kubernetes. A Pod typically contains one or more tightly coupled containers that share the same network namespace and storage.

Deployments: A higher-level object that manages the desired state of your Pods. It handles rolling updates, rollbacks, and ensuring a specified number of Pods are running.

Services: An abstraction that defines a logical set of Pods and a policy by which to access them *(e.g., a stable IP address and DNS name for your web application, even if the underlying Pods change).*

Ingress: Manages external access to services within the cluster, providing HTTP/S routing.

Volumes: Used to provide persistent storage to containers.

How Kubernetes Works:

- **Clusters:** Kubernetes organizes a group of physical or virtual machines into a "cluster." These machines are called "nodes."
    - **Master Node(s) (Control Plane):** This is the brain of the cluster. It manages the overall state, schedules containers, handles API requests, and monitors the cluster.
    - **Worker Nodes:** These are the machines where your actual containerized applications run. Each worker node runs a kubelet (an agent that communicates with the master) and a container runtime (like Docker or containerd).
- **Declarative Configuration:** You tell Kubernetes your "desired state" using configuration files (often YAML). For example, you might say: *"I want 3 instances of my web application, using this container image, exposed on this port, with access to this storage."*
- **Kubernetes' Job:** Kubernetes then continuously works to match the current state of your cluster to your desired state.
    - If a container crashes, it restarts it.
    - If a node fails, it moves containers to healthy nodes.
    - If traffic increases, it automatically scales up the number of container instances.
    - It provides internal networking and load balancing so your application components can find each other and users can access your services

Real-Life scenarios-

Scenario	Why AKS Is Ideal
Microservices-based Applications	Run independent services in containers, auto-scale based on demand
High-Traffic Web Apps	Deploy large-scale web apps with horizontal scaling and load balancing
Machine Learning Workloads	Use GPU nodes and run containerized training/inference jobs
CI/CD Deployment Pipelines	Automate building, testing, and deploying containers using DevOps pipelines
IoT Data Processing	Run distributed processing on real-time telemetry streams

Gaming Backend Services	Manage player sessions, matchmaking, game state with high scalability
API Gateways & Backend Services	Run scalable backend microservices that handle API traffic
Hybrid Cloud Deployments	Combine on-prem and Azure workloads using AKS and Azure Arc
Dev/Test Environments	Quickly spin up environments per developer or team in isolated namespaces
Enterprise SaaS Platforms	Multi-tenant architecture support with namespace and resource isolation

Now Understand the differences between Container Instances vs Container Apps and Kubernetes

Aspect	Azure Kubernetes Service (AKS)	Azure Container Instances (ACI)	Azure Container Apps (ACA)
What It Is	Managed Kubernetes (full orchestration)	Lightweight container runtime (serverless)	Serverless container service with app-centric features
Use Case Type	Complex, large-scale workloads	Short-lived, simple jobs	Web apps, APIs, event-driven apps, microservices
Infrastructure Control	High — full control over nodes and networking	Low — no VM or infra management	Medium — no node management but built on Kubernetes
Scaling	Manual & auto-scaling (cluster + pods)	No autoscaling	Built-in autoscaling based on traffic, events, CPU
Startup Speed	Slower (VM-based nodes)	Fast (seconds)	Fast (seconds)
App Complexity	Best for complex, distributed microservices	Best for quick, standalone tasks	Ideal for scalable apps with routing and versioning
Multi-container support	Yes (pods)	No (only one container per instance)	Yes (multi-container apps)
Orchestration	Yes (Kubernetes)	No	Yes (built-in based on Kubernetes)
Ingress & Routing	Manual setup via Ingress Controllers	No built-in routing	Built-in HTTPs ingress, traffic splitting
Event-driven Scaling	Requires custom setup or KEDA	Not supported directly	Native support via KEDA for queues, HTTP, timers
CI/CD Integration	Full DevOps pipelines	Build-and-run model	Integrated via GitHub, Azure DevOps

Billing	Pay for VMs + K8s control plane (limited free)	Pay-per-second (serverless)	Pay for resources used (CPU/RAM only)
Learning Curve	High — requires knowledge of Kubernetes	Very low	Low — YAML or portal-based deployments
Ideal For	Enterprises with complex production workloads	Dev/test, jobs, one-off scripts	App developers need scalability + routing without Kubernetes
Example Scenario	Multi-service ecommerce platform with scaling + CI/CD	Scheduled data cleanup job or PDF generator	REST API with autoscaling, internal microservices

Azure Functions-

key component of Microsoft Azure's serverless computing platform.

Azure Function is a **small, single-purpose piece of code** that executes in response to a specific event or "trigger." The defining characteristic is that you, as the developer, **don't manage the underlying servers or infrastructure.** You simply write your code (in languages like C#, JavaScript, Python, Java, PowerShell, etc.), define what triggers it, and Azure takes care of all the provisioning, scaling, and maintenance.

Key Concepts-

Serverless: This is the most crucial aspect. You don't provision or manage virtual machines, operating systems, or web servers. Azure automatically handles the infrastructure. You focus 100% on your code.

Event-Driven: Functions are designed to respond to events. These events can come from a wide variety of sources, such as:

- **HTTP requests:** A web request to an API endpoint.
- **Timers:** Executing code on a schedule (e.g., every 5 minutes, daily at midnight).
- **Database changes:** When a new item is added to a Cosmos DB collection or a row is updated in SQL.
- **Queue messages:** When a message arrives in an Azure Storage Queue or Service Bus.
- **Blob storage changes:** When a file is uploaded or deleted in Azure Blob Storage.
- **Event Hubs:** Processing real-time streams of data.

- **IoT Hub:** Responding to messages from IoT devices.

Pay-per-execution (Consumption Plan): In the most common hosting plan (Consumption Plan), you only pay for the compute resources consumed *when your function is actually running*. If your function isn't triggered, you pay nothing. This makes it incredibly cost-effective for intermittent or unpredictable workloads.

Automatic Scaling: Azure automatically scales your functions up or down based on the incoming event load. If there's a sudden surge in HTTP requests, Azure will create more instances of your function to handle the load and then scale them down when the load subsides.

Bindings: Functions have a powerful concept called "bindings." These allow you to easily connect your function to other Azure services (like storage accounts, databases, messaging queues) without writing boilerplate code for connection strings, SDKs, or data parsing. You simply declare input and output bindings, and Azure handles the data flow.

Stateless by default (but can be stateful with Durable Functions): Generally, functions are designed to be stateless, meaning each execution is independent. However, Azure Durable Functions is an extension that allows you to write stateful workflows in a serverless environment, enabling long-running orchestrations, human interaction, and fan-out/fan-in patterns.

Where and When are Azure Functions Used?

1. **Event-Driven:** They shine when your code needs to react to something happening.
2. **Short-Lived (generally):** While Durable Functions allow for longer orchestrations, individual function executions on the Consumption plan have a time limit (typically 10 minutes, but configurable for Premium plans).
3. **Stateless or managing state externally:** Best for functions that don't need to maintain state between invocations or can manage state through external services like databases or storage.
4. **Cost-Sensitive for Intermittent Workloads:** The pay-per-execution model makes them very economical for tasks that run infrequently or have unpredictable usage patterns.

See the Common use cases where these Azure Functions are used.

- **Event-Driven Automation**
  - Triggered by events in Azure *(e.g., Blob storage uploads, queue messages, database changes)*
  - Example: Resize an image when it's uploaded to Azure Blob Storage
- **Queue Processing**
  - Process tasks from Azure Storage Queue, Service Bus, or Event Hub
  - Example: Handle incoming orders or background jobs asynchronously
- **Lightweight APIs & Webhooks**
  - Serve as backend logic for microservices or HTTP-triggered endpoints
  - Example: Create serverless REST APIs for a mobile or web app
- **Scheduled Jobs (Cron)**
  - Run on a defined time schedule using Timer Triggers
  - Example: Send daily reports or clean up logs every night
- **Identity & Security Workflows**
  - Handle events from Azure AD or API Management
  - Example: Custom authentication/authorization checks
- **Dev/Test Utilities**
  - Quickly test code logic or run ad-hoc serverless functions
  - Example: Mocking services or temporary endpoints for testing
- **Integration with Other Systems**
  - Bridge between Azure and third-party systems (e.g., Salesforce, SAP)
  - Example: Post updates to Microsoft Teams or Slack when a record is added
- **Data Cleanup and Transformation**

- Transform, enrich, or clean data before storing or moving it
- Example: Format telemetry data before inserting into a database

❖ **Notification Dispatch**

- Trigger notifications via email, SMS, or push
- Example: Alert users when certain thresholds are crossed

❖ **Serverless Backend for Low-Traffic Apps**

- Cost-effective backend for apps with irregular or low request volume
- Example: Backend logic for a calculator or contact form

Azure App Services

Azure App Services is a fully managed Platform-as-a-Service (PaaS) offering from Microsoft Azure for building, deploying, and scaling web applications, mobile backends, and RESTful APIs.

Think of it as a highly productive and comprehensive web hosting service in the cloud. Instead of managing servers, operating systems, and web server software (like IIS or Nginx), you simply deploy your code, and Azure handles all the underlying infrastructure, patching, and maintenance.

Key Features and Concepts-

Fully Managed PaaS: This is the core benefit. Azure handles the servers, OS patching, runtime updates, and infrastructure scaling, allowing you to focus purely on your application code.

Multiple Language and Framework Support: App Service supports a wide range of popular programming languages and frameworks, including:

- .NET, .NET Core
- Java (including Tomcat and JBoss)
- Node.js
- Python
- PHP
- Ruby
- You can also deploy custom Docker containers for maximum flexibility (this is where the relationship with containers comes in).

Automatic Scaling: An App Service can automatically scale your application up (more CPU/memory) or out (more instances) based on demand. You can configure auto-scaling rules based on metrics like CPU usage, memory, HTTP queue length, or schedule.

High Availability: Built-in load balancing and redundancy across Azure's global data centers ensure your application remains available even during unexpected outages. Azure provides a robust SLA (Service Level Agreement) for uptime.

DevOps Optimization:

- **Continuous Deployment (CI/CD):** Seamless integration with popular source control systems like GitHub, Azure DevOps, Bitbucket, and local Git for automated deployments whenever code changes are pushed.
- **Deployment Slots:** Allows you to deploy a new version of your application to a "staging" slot for testing before swapping it into "production" with zero downtime. This is crucial for blue/green deployments and A/B testing.

Security and Compliance:

- Built-in authentication and authorization using Azure Active Directory, Google, Facebook, Twitter, and Microsoft accounts ("Easy Auth").
- SSL/TLS certificates for secure communication (including free managed certificates).
- Integration with Azure Key Vault for secure secrets management.
- Compliance with various industry standards (ISO, SOC, PCI).

Integrated Monitoring and Diagnostics: Integrates with Azure Monitor and Application Insights, providing real-time insights into application performance, health, logs, and metrics.

Custom Domains and SSL: Easily map your custom domain names to your App Service app and secure them with SSL certificates.

Hybrid Connectivity: Securely connect your App Service apps to on-premises resources using features like Hybrid Connections and Azure Virtual Network Integration.

Cost-Effectiveness: Offers various pricing tiers (Free, Shared, Basic, Standard, Premium, Isolated) to match different workload requirements and budgets. You pay for the compute resources provisioned in your "App Service Plan."

Where & When to Use Azure App Services

Web Application Hosting

- Host full-stack **web apps (HTML, .NET, Java, Node.js, Python, PHP)** with minimal setup.
- Great for business websites, customer portals, dashboards, etc.
- **When to use**: You need a **fully managed web hosting** platform with auto-scaling and high availability.

RESTful API Hosting

- Deploy secure, scalable **API backends** for mobile or web applications.
- Supports OpenAPI/Swagger, versioning, throttling, and CORS.
- **When to use**: Building **lightweight microservices** or public/private APIs.

CI/CD Integration

- Connects easily with **Azure DevOps, GitHub, Bitbucket** for auto-deployment pipelines.
- Supports staging environments and slot swapping.
- **When to use**: Automating deployments and enabling **zero-downtime releases**.

Business Applications & SaaS Platforms

- Host **line-of-business (LOB)** applications like CRMs, HRMS, or internal portals.
- Easily integrate with on-premises data via **Hybrid Connections**.
- **When to use**: Developing **enterprise-grade apps** with secure user access.

Multi-Tenant Web Apps (SaaS)

- Create apps that serve multiple customers/organizations from a single codebase.
- Built-in support for **authentication, scaling, and SSL**.
- **When to use**: You're building a **SaaS platform** with separate user bases.

Mobile App Backend Services

- Provide **backend logic, APIs, or databases** for mobile apps using REST endpoints.
- Supports **offline sync, authentication**, and push notifications.
- **When to use**: A mobile app needs a scalable, secure backend without managing infra.

Serverless Web App Alternatives

- Combine **App Service with Azure Functions or Logic Apps** for serverless architectures.
- **When to use**: You want event-driven scalability but also need traditional web app support.

Custom Domain & SSL Hosting

- Host apps with **custom domains**, **HTTPS**, and **automatic certificate management**.
- **When to use**: Public-facing apps requiring secure, branded URLs.

Authentication & Identity Integration

- Supports **Azure AD, Facebook, Google, Twitter, GitHub** authentication out-of-the-box.
- **When to use**: You need **secure login and access control** without building it yourself.

Azure Virtual Network/Networking

It is the **fundamental building block for your private network in Azure**. It's a service that allows you to create your own isolated and logically segmented network in the cloud, much like a traditional network you'd operate in your own data center, but with the added benefits of Azure's scale, availability, and global reach.

It Provides a **secure and customizable network environment** where you can deploy and connect various Azure resources *(like Virtual Machines, Azure App Services, Azure Functions Premium Plan, Azure Kubernetes Service, databases, etc.)* and connect them to your on-premises networks.

Deeper Breakdown-

Logical Isolation and Private IP Addressing:

- **Your Own Private Space:** When you create a VNet, you define a private IP address space using CIDR (Classless Inter-Domain Routing) notation (e.g., 10.0.0.0/16, 172.16.0.0/12, 192.168.0.0/16). This IP address range is unique to your VNet and is not directly accessible from the public internet, providing inherent isolation.

- RFC 1918 Compliance: You typically use RFC 1918 private IP address ranges within your VNet. Azure ensures that these private IP addresses are only used within your virtual network and its connected networks (via peering or hybrid connectivity), preventing IP address conflicts with other Azure customers.
- Internal Communication: Resources within the same VNet communicate with each other using their private IP addresses, making communication fast and secure.

2. **Subnets for Segmentation:**

- Network Segmentation: A VNet can be divided into one or more subnets. Each subnet is a logical division of the VNet's IP address space.
- Organization and Security: Subnets allow you to segment your network based on application tiers (e.g., a "WebSubnet" for front-end servers, an "AppSubnet" for application logic, and a "DbSubnet" for databases). This improves organization and enables granular security controls.
- Network Security Groups (NSGs): NSGs are like virtual firewalls that you can associate with subnets or individual network interfaces (NICs). They contain a set of inbound and outbound security rules that allow or deny traffic based on source/destination IP address, port, and protocol. By applying NSGs to subnets, you can enforce security policies between different application tiers (e.g., only allow traffic from the WebSubnet to the AppSubnet, and from the AppSubnet to the DbSubnet).

3. **Routing and Traffic Flow Control:**

- Default Routing: By default, Azure VNets provide basic routing:
  - Traffic within the same VNet (and between peered VNets)
  - Outbound internet access (with a public IP or NAT Gateway)
- User-Defined Routes (UDRs) / Route Tables: You can create custom route tables and associate them with subnets to override Azure's default routing. This allows you to:
  - Force traffic to flow through a Network Virtual Appliance (NVA) like a firewall or a proxy server for inspection.
  - Direct traffic to specific gateways for hybrid connectivity.
- Service Endpoints: Extend your VNet's private address space to specific Azure PaaS services (e.g., Azure Storage, Azure SQL Database, Azure Cosmos DB). This allows resources within your VNet to access these services directly over the Azure backbone network, enhancing security by eliminating public internet exposure.

- Azure Private Link / Private Endpoint: A more recent and powerful feature that allows you to connect privately and securely to Azure PaaS services (and even your own services hosted in Azure) using a *private IP address* from your VNet. This brings the service directly into your VNet, providing the highest level of network isolation and preventing data exfiltration.

4. **Connectivity Options (Hybrid and Inter-VNet):**

- Virtual Network Peering: Allows you to seamlessly connect two or more Azure VNets. Once peered, resources in either VNet can communicate with each other using private IP addresses as if they were in the same network.
    - Regional VNet Peering: Connects VNets within the same Azure region.
    - Global VNet Peering: Connects VNets across different Azure regions. Traffic travels over Microsoft's high-speed global backbone network.
- VPN Gateway: Enables secure, encrypted connectivity between your Azure VNet and your on-premises network over the public internet (IPsec VPN tunnel).
    - Site-to-Site VPN: Connects your on-premises VPN device to an Azure VPN Gateway, establishing a persistent tunnel.
    - Point-to-Site VPN: Allows individual client computers to establish a secure connection to your VNet.
- Azure ExpressRoute: Provides a dedicated, private, and high-bandwidth connection between your on-premises network and Azure. Traffic *does not* traverse the public internet, offering higher security, reliability, and lower latency than VPNs.
- Virtual WAN: A managed networking service that provides optimized and automated branch connectivity to Azure and inter-VNet connectivity.

5. **Integration with Other Azure Services:**

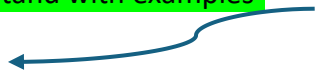
- Compute Resources: Virtual Machines (VMs), Azure Kubernetes Service (AKS), Azure Container Instances (if deployed into a VNet), Azure App Service Environment (isolated tier of App Service).
- Databases: Azure SQL Database (via Private Link), Azure Cosmos DB (via Service Endpoints/Private Link), Azure Database for MySQL/PostgreSQL/MariaDB.
- Storage: Azure Storage Accounts (via Service Endpoints/Private Link).

- Networking Services: Integrates with Azure Load Balancer, Application Gateway, Azure Firewall, Azure DNS, Azure DDoS Protection, Azure Bastion.

6. **Benefits of Using Azure Virtual Network:**

- Security: Provides network isolation, granular traffic control with NSGs, secure hybrid connectivity, and private access to PaaS services.
- Scalability & Flexibility: Easily expand your network, add subnets, and scale resources within the VNet as your needs grow.
- Control: Full control over your IP address space, routing, and security policies.
- Hybrid Cloud Scenarios: Seamlessly extend your on-premises network to Azure, creating a unified hybrid environment.
- High Availability: Built-in redundancy and integration with other Azure services for highly available application architectures.
- Compliance: Helps meet regulatory and compliance requirements by providing a secure and isolated network environment.

Understand with examples-



1. Secure Communication Between Azure Resources

- **Scenario**: Connect Azure VMs, App Services, and databases privately within the same network.
- **Example**: A 3-tier web application (Web VM → App VM → SQL Database) communicates internally via VNet without public exposure.

2. Hybrid Cloud Networking (On-Prem to Azure)

- **Scenario**: Extend your on-premises network to Azure using VPN or ExpressRoute.
- **Example**: A manufacturing company connects its datacenter to Azure to back up SQL servers using a site-to-site VPN.

3. Isolated and Private App Hosting

- **Scenario**: Run internal applications without internet access using VNet integration.

- **Example**: An HR application hosted on Azure App Service is accessible only within the corporate VNet.

4. Virtual Machines and Bastion Access

- **Scenario**: Place VMs in subnets and access them securely via Bastion without using public IPs.
- **Example**: An enterprise manages Linux VMs via Azure Bastion inside a secure subnet for daily operations.

5. Application Gateway + Load Balancer for Web Apps

- **Scenario**: Host public-facing apps with Application Gateway and internal apps with Load Balancer in a VNet.
- **Example**: An e-commerce site routes external traffic via Azure Application Gateway and balances traffic across multiple VM instances.

6. Integration with Azure Services

- **Scenario**: Integrate services like Azure App Service, Azure Kubernetes Service (AKS), and Azure Functions with VNets.
- **Example**: A company secures traffic from AKS pods to an Azure SQL Database using VNet service endpoints.

7. VNet Peering Across Regions/Departments

- **Scenario**: Connect multiple VNets across regions or teams for shared services and communication.
- **Example**: A multinational firm peers a Europe VNet with an India VNet to sync regional backup services.

8. Custom DNS & Name Resolution

- **Scenario**: Use custom DNS servers within VNets for internal name resolution.
- **Example**: A DevOps team uses a private DNS zone to resolve internal service names for microservices across subnets.

9. NSG and Route Table Configurations

- **Scenario**: Control and secure traffic with Network Security Groups and route tables within a VNet.
- **Example**: A finance department isolates sensitive workloads in a subnet with strict NSG rules and custom routes for logging.

10. Private Link & Service Endpoints

- **Scenario**: Connect securely to Azure PaaS services without going over the public internet.
- **Example**: A banking app connects to Azure Blob Storage via Private Link from within its VNet for compliance.

11. Multi-tier Application Architecture

- **Scenario**: Design subnets for each tier (web, app, database) within a VNet.
- **Example**: A logistics company runs its web tier in a public subnet, app tier in a private subnet, and database tier in an isolated subnet.

12. Disaster Recovery Across Regions

- **Scenario**: Use VNet + VPN or peering to sync backups or failover services to a secondary region.
- **Example**: A healthcare provider replicates patient data to a DR site in a different Azure region via peered VNets.

13. Hosting Containers with Network Policies

- **Scenario**: Deploy containerized apps (AKS or ACI) within VNets for policy enforcement.
- **Example**: A security firm uses Kubernetes network policies in AKS to restrict traffic between microservices.

Azure VPN Gateway

It is a service within Microsoft Azure that allows you to establish secure, encrypted connections between various networks over the public internet. It functions as a "virtual VPN device" within your Azure Virtual Network. (VNet)

The primary purpose of Azure VPN Gateway is to enable **hybrid connectivity** – seamlessly connecting your on-premises networks *(your corporate data center, branch offices, or even individual remote users)* to your Azure Virtual Networks. It also facilitates secure communication **between Azure Virtual Networks** themselves.

Key Concepts and Types of Azure VPN Gateway Connections:

Mainly 2 Types-

Site-to-Site (S2S) VPN:

- Connects your entire on-premises network (e.g., your corporate office or data center) to an Azure Virtual Network. It's like extending your on-premises network into Azure.
- **How it works:** An IPsec/IKE (Internet Protocol Security/Internet Key Exchange) VPN tunnel is established over the public internet between your on-premises VPN device (hardware or software firewall/router) and an Azure VPN Gateway in your VNet.
- **Traffic:** All traffic between your on-premises network and your Azure VNet is encrypted and sent through this tunnel. Resources on your on-premises network can communicate with resources in your Azure VNet as if they were on the same network (assuming proper routing).
- **Use Cases:**
  - **Hybrid Cloud:** Extending your data center to the cloud, allowing on-premises applications to access resources in Azure and vice versa.
  - **Disaster Recovery:** Replicating data or deploying disaster recovery sites in Azure.
  - **Consolidating IT Infrastructure:** Moving some workloads to Azure while keeping other on-premises.
  - **Multi-branch Connectivity:** Connecting multiple on-premises branch offices to a centralized VNet in Azure (Multi-Site VPN).

Point-to-Site (P2S) VPN:

- Allows individual client computers *(e.g., a remote employee's laptop at home, a developer's workstation)* to securely connect to an Azure Virtual Network over the internet.
- **How it works:** The connection is initiated from the client computer using a VPN client *(e.g., Azure VPN Client, native Windows VPN client, OpenVPN client).* It uses protocols like OpenVPN, IKEv2, or SSTP (Secure Socket Tunneling Protocol) to create an encrypted tunnel. Unlike S2S, it does not require a dedicated VPN device on the client side.
- **Traffic:** The individual client can then access resources within the Azure VNet using private IP addresses.
- **Use Cases:**
  - **Remote Access:** Enabling remote employees or contractors to securely access corporate resources in Azure.
  - **Developer Access:** Allowing developers to securely connect to development and testing environments in Azure.
  - **Auditor/Consultant Access:** Providing temporary, secure access to specific Azure resources for external parties.

VNet-to-VNet Connections:

- Securely connect two or more Azure Virtual Networks together. This is essentially a Site-to-Site VPN but entirely within Azure, across different VNets (which can be in the same or different regions, and even different subscriptions).
- **How it works:** An IPsec/IKE VPN tunnel is established between two Azure VPN Gateways. Traffic between the connected VNets traverses the encrypted tunnel over the Microsoft backbone network.
- **Use Cases:**
  - **Multi-tier Applications:** Deploying different application tiers into separate VNets for better isolation and security but allowing them to communicate securely.
  - **Regional Redundancy:** Connecting VNets in different regions for disaster recovery or global load balancing.
  - **Centralized Services:** Connecting spoke VNets to a hub VNet where shared services (e.g., Active Directory, shared databases, monitoring tools) reside.

Azure Express Route

- Azure ExpressRoute is a dedicated, private network connection service that allows you to extend your on-premises networks into Microsoft cloud.
- Unlike VPN Gateway which uses the public internet for encrypted tunnels, ExpressRoute connections **do not traverse the public internet**.
- Instead, they establish a direct, high-bandwidth, low-latency, and highly reliable connection through a third-party connectivity provider.

Meaning

It uses a dedicated physical connection/Circuits provided through a third-party network provider. (TATA, Airtel and Equinix)

Easy/Simple-> A Private, Fiber-Based Fast Lane to Azure, leased and managed through a third-party provider, but enabled/configured as a service within your Azure subscription.

It offers higher bandwidth, lower latency, and more consistent performance than VPN over the internet.

Step-by Step Working-

1. Establish a Physical Connection

- You work with a **service provider** (ExpressRoute partner) to create a **dedicated MPLS circuit** between your site and Microsoft's peering location (ExpressRoute meet point).
- Your **Edge router (CPE)** connects to the provider's router, which connects to Microsoft's **Edge router**.

2. Create an ExpressRoute Circuit in Azure

- You create the circuit in Azure Portal or CLI, specifying:
  - **Location** (peering location, e.g., Mumbai, Chennai)
  - **Bandwidth** (50 Mbps to 100 Gbps)
  - **SKU** (Standard or Premium)
- This circuit is assigned a **Service Key (S-Key)**, which you share with your provider.

3. **Configure Peering**

There are three types of **BGP peering** sessions you can configure:

Peering Type	Description
Private Peering	Connect to Azure VNets and private services like VMs, SQL, AKS
Microsoft Peering	Connect to Microsoft SaaS services like Office 365, Dynamics 365, etc.
MSEE Peering (ER Gateway)	Through Microsoft Enterprise Edge if used in certain enterprise models

4- Link the Circuit to Your VNet

- Deploy a **Virtual Network Gateway (ExpressRoute Gateway)** in Azure.
- Link it to the ExpressRoute circuit.
- Once BGP is up, your **on-prem routers exchange routes** with Azure over the private connection.

5. Traffic Flow Begins

- Your traffic now **flows through the dedicated circuit**, bypassing the public internet.
- You can route traffic to:
    - Azure VNets (Private IPs)
    - PaaS services (via Service Endpoints or Private Link)
    - Microsoft services like Azure Storage, Microsoft 365 (via Microsoft Peering)

Use Cases of Azure ExpressRoute

- **Banking & Financials**: Secure data exchange with regulatory compliance.
- **Manufacturing**: Real-time telemetry & production control systems.
- **Enterprises**: SAP, Oracle workloads, large file transfers, VDI
- **Disaster Recovery**: Replicate VMs, databases with high throughput.

Azure DNS

- It is a **highly available and scalable Domain Name System (DNS) hosting** service provided by Microsoft Azure.
- It allows you to host your DNS domains and manage your DNS records using Azure's global infrastructure.

Specialty about Azure DNS-

1. Native Azure Integration

- Seamlessly integrates with other Azure services (e.g., App Services, VMs, Azure CDN).
- Use **RBAC (Role-Based Access Control)** for secure DNS management.
- Use **ARM templates, Bicep, Azure CLI, or PowerShell** to automate DNS zone creation and record updates.

2. Ultra-Low Latency + High Availability

- Hosted on Azure's **global anycast network**.
- DNS queries are routed to the **nearest DNS server automatically**.
- Built-in **high availability and DDoS protection** as part of Azure's backbone.

3. Security & Compliance

- **Built-in DNSSEC support** (currently in preview).
- Access control via **Azure AD and RBAC**.
- Complies with **industry-grade certifications** (ISO, SOC, etc.).

4. Centralized DNS Management for Enterprises

- Manage DNS zones for **internal (private)** and **external (public)** domains in one place.
- Combine with **Private DNS Zones** for internal name resolution within VNets.

5. Dynamic, Programmable DNS

- Fully API-driven – supports **infrastructure-as-code** for DevOps use.
- Easily used in **CI/CD pipelines** to update DNS records automatically on deployment.

6. DNS Alias Records

- Supports **alias records** that can **point to Azure resources** (like Traffic Manager, Public IPs, CDN endpoints) auto-updates when IP or resource changes.

Beauty of Azure DNS-> If someone is starting a **company or startup**, he/she can **fully host all their websites' DNS records using Azure DNS.**

Beauty lies here→ it is not mandatory that the Entire Company should be Hosted on Azure environment, rest of the company's infrastructure can be **completely physical and can be hosted elsewhere, if you only want Azure DNS to be used, you can Use.**

Now, understand this-

Azure DNS as a Standalone Service – Use Case

Many organizations use **Azure DNS only for domain name resolution**, even when:

- Their servers are **on-premises.**
- Their apps are hosted on **AWS, Google Cloud, or a private data center.**
- Their email runs on **Google Workspace or Zoho.**
- They have no other Azure resources at all.

Example Scenario: ←

A logistics company runs:

- All applications on **on-prem Linux servers.**
- Email via **Google Workspace.**
- ERP on a **private cloud.**
- But they host their DNS on **Azure DNS.**

Why? Because they want:

- Centralized, secure, API-driven DNS
- Avoid manual work of managing records via GoDaddy panel.
- Use PowerShell or CLI to auto-update DNS during app deployments

Important Note- Azure DNS **only handles domain name resolution** — it **does not care where the actual servers or services are hosted**.

You just need to:

- Register your domain (via any domain registrar)
- Point its **nameservers to Azure DNS**
- Create A, CNAME, MX, TXT, etc. records pointing to your physical/public infrastructure

Azure Storage Accounts/Services

What is this?

An **Azure Storage Account** is the foundational service in Azure that provides a **single, secure, and highly scalable container for all your Azure Storage data objects.** It's like a central umbrella under which you can store various types of data in the cloud.

It is a Cloud-based storage solution.

Ok, what types of data can you store?

Azure Blobs (Blob Storage):

- A massively scalable object stores unstructured data. Think of it as a place to store files of any type (text, binary) without a fixed structure.
- Use Cases:
  - Serving images, videos, audio, and documents directly to web browsers.
  - Storing files for distributed access (e.g., application logs, telemetry data).
  - Backup and disaster recovery data.
  - Storing data for big data analytics (integrated with Azure Data Lake Storage Gen2).
  - Cold data archiving.
- Blob Types:
  - Block Blobs: Ideal for digital media, documents, log files, backups. Optimized for streaming and random access. Most common type.
  - Append Blobs: Optimized for append operations, like logging data from IoT devices or web server logs.
  - Page Blobs: Optimized for random read/write operations and used as the basis for Azure Virtual Machine OS and data disks.

Azure Files (File Shares):

- Fully managed file shares in the cloud that are accessible via the industry-standard Server Message Block (SMB) protocol or Network File System (NFS) protocol. This allows you to "lift and shift" traditional file shares from on-premises to Azure without rewriting applications.
- Use Cases:
  - Shared application settings or configuration files.
  - File shares for legacy applications that rely on file system APIs.
  - User home directories.
  - Development tools and utilities.

Azure Queues (Queue Storage):

- A service for storing large numbers of messages. It's a simple, REST-based queueing service for reliable messaging between application components.
- Use Cases:
  - Decoupling application components for greater scalability and resilience.
  - Building asynchronous messaging systems.
  - Buffering requests when processing a large burst of data (e.g., offloading web tasks to a backend worker).

Azure Tables (Table Storage):

- A NoSQL (schema-less) key-attribute store that allows you to store large amounts of structured, non-relational data in the cloud.
- Use Cases:
  - Storing flexible datasets like user data for web applications, address books, device information, or other types of metadata.
  - Large-scale data storage where complex joints or relational queries are not required.

Azure Disks (Managed Disks):

- These are durable, block-level storage volumes for Azure Virtual Machines. While technically they reside within storage accounts (Page Blobs), Managed Disks abstract away the complexity of managing individual storage accounts for your VM disks. Azure automatically places the disk in a storage account behind the scenes for you.
- Use Cases:
  - Operating system disks for Azure VMs.
  - Data disks for Azure VMs.
  - Disks for database servers running on VMs.

Now Understand what's happening Physically in Azure data Centers

Your data is stored on SSD (Solid State Drives) and HDD (Hard Disk Drives) just like in traditional infrastructure, but on an enterprise-grade hyperscale level.

Storage Type	Purpose in Azure	Backed by
Premium Storage	High IOPS workloads (VM disks, databases)	NVMe & SSD-based storage
Standard Storage	General-purpose workloads (backup, archive, logs)	Spinning disks (HDD) & SSDs
Archive Storage	Infrequently accessed, long-term cold storage	Low-cost, high-capacity HDDs
Ultra Disk Storage	Mission-critical apps needing extreme performance	High-throughput NVMe SSDs

*** HIGH IOPS-> Input/Output Operations Per second ***

Azure Data Redundancy **-> *Storing Multiple copies of your data. ***

When you create a storage account, you choose a redundancy strategy, which applies to all data stored within that account. This determines how many copies of your data are maintained and where they are stored.

You can Store your data either within the same data center, across availability zones, or even across entire geographic regions — to **prevent data loss** and ensure **continuous availability**.

Why Redundancy Matters??

- Hardware can fail (disks, servers).
- Entire racks or zones can go down (power outage, flood, etc.)
- You need durable backups even if a data center is lost.

Redundancy protects you against these risks.

Redundancy Options in Azure-

Type	Stands For	Where Data is Replicated	Use Case / Protection Level
LRS	Locally Redundant Storage	**3 copies** within a **single data center**	Protects against **disk/rack failure**
ZRS	Zone-Redundant Storage	**3 copies across 3 Availability Zones** in a region	Protects against **zone failure**
GRS	Geo-Redundant Storage	**3 copies locally (LRS) + 3 in secondary region**	Protects against **the entire region loss**
RA-GRS	Read-Access GRS	Same as GRS + **read access** to secondary region	Use for **read-heavy apps** or DR
GZRS	Geo-Zone-Redundant Storage	Combines **ZRS in primary** + **GRS to secondary**	Best of **ZRS + GRS** (high + geo)

How It Works (Behind the Scenes)

LRS (Locally Redundant Storage)

- Data is synchronously copied 3 times within a single physical data center.
- Protected from disk, server, or rack failure
- Not protected from data center failure

Best for: Cost-sensitive apps where high availability isn't critical

ZRS (Zone-Redundant Storage)

- Data is synchronously copied across 3 separate Availability Zones.
- These zones are physically isolated with independent power, cooling, and networking
- Handles zone-wide failures automatically

Best for: High availability apps (like websites, APIs, live data apps)

GRS (Geo-Redundant Storage)

- Combines LRS + Geo-replication to another paired region
- *Example:* If your storage is in East US, the backup copy goes to West US
- Data is replicated asynchronously (there's a slight lag)
- In case of a regional outage, Microsoft can initiate a failover to secondary region

Best for: Disaster recovery, compliance, regulatory scenarios

RA-GRS (Read-Access Geo-Redundant Storage)

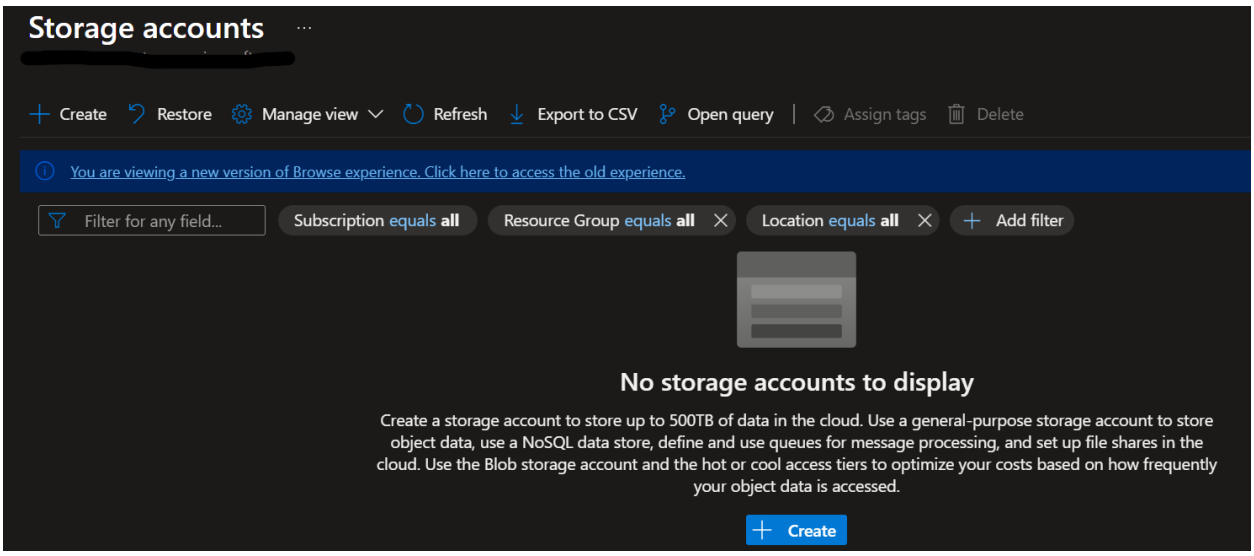
- Same as GRS but you get read access to the secondary region
- Useful for read-only analytics, backups, or dashboards in case of disaster

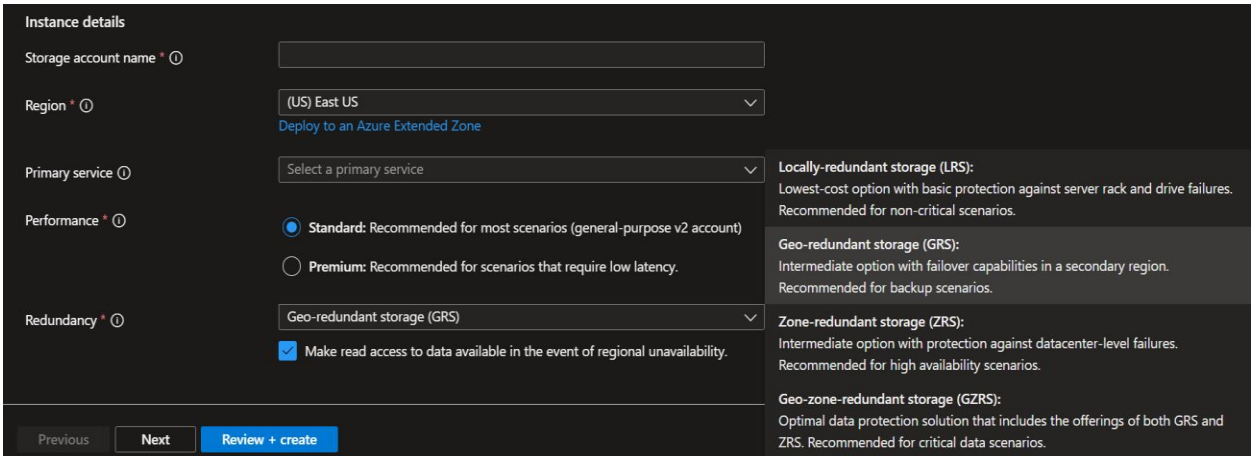
Best for: Read-heavy DR-ready systems

GZRS (Geo-Zone Redundant Storage)

- Combines ZRS across zones + GRS to another region
- Best protection: Handles zone and regional outages
- Azure automatically manages failover and replication end-to-end

Best for: Mission-critical workloads needing max durability + HA + DR





Scenario time- A company wants to:

- Store **primary data in Azure Pune**
- Keep a **redundant (replicated) copy in Azure Chennai**
- Use **Geo-Redundant Storage (GRS)**

How Data Replication Works Between Pune and Chennai?

1. **Primary Write (Pune - Central India)**

- You upload or update data (blob/file/table/queue) to the Pune storage account.
- Azure immediately stores 3 copies of that data within Pune (LRS style).
    - This protects against local disk or rack failures.

2. Asynchronous Replication to Secondary (Chennai - South India)

- After the data is securely written in Pune, Azure asynchronously starts replicating that data to Chennai.
- 3 additional copies are created and stored in the Chennai region (again, LRS-style there).

3. Data Sync Delay (RPO impact)

- The replication is not instant — it usually takes seconds to a few minutes, depending on:
    - File size

- Network conditions
- Azure replication engine performance

This delay is called the Recovery Point Objective (RPO) --the maximum amount of data you could lose if a region fails before replication completes.

4. Failover Scenario (When Pune Goes Down)

- By default, you don't access the secondary (Chennai) directly.
- In case of a complete region-wide failure in Pune:
  - Microsoft initiates a failover to Chennai (only they can do this).
  - After failover, Chennai becomes your new primary region.
  - From then on, your data writes and reads happen in Chennai.

Note: During normal operation, you cannot read or write directly to Chennai unless you use RA-GRS (read-only access to Chennai).

So, can we Store Data in Mumbai Region?

Yes, but we must do it Manually.

Option 1: Custom Geo-Replication (Manual Sync)

1. Primary Storage Account: Pune (Central India)
2. Secondary Storage Account: Mumbai (West India)
3. Use Azure Data Factory, AzCopy, or Azure Functions to:
   - Automatically replicate new or changed files
   - Maintain synchronization on a schedule (near real-time if needed)

Change Flow: ↘

- Any changes in Pune → Triggered or scheduled → Replicated to Mumbai
- You control the logic (overwrite, append, etc.)

Option 2: Use Azure GRS (Chennai) + Manual Copy to Mumbai

- Use **Azure GRS** (Pune → Chennai auto-handled)
- Add a **third Storage Account in Mumbai**
- Build a **custom replication pipeline** from Pune → Mumbai

This gives:

- **Auto failover to Chennai (via GRS)**
- **Manual access copy in Mumbai**

Important Concept* **How Companies and Companies Data moves from On-Prem to Azure Cloud?**

*** Comes Under Azure Migrate Services***

What Does "On-Prem to Azure" Mean?

It means migrating files, databases, virtual machines, apps, backups, or entire workloads from company-owned physical servers (on-premises) to Microsoft Azure's cloud infrastructure.

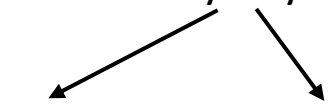
What Exactly Companies Migrate to Azure?

- File servers (NAS/SAN storage)
- SQL / Oracle / MySQL databases
- Virtual Machines (VMware, Hyper-V, bare metal)
- Line-of-business apps (ERP, HRMS, CRM)
- Web servers, APIs, backend applications
- Active Directory (hybrid setup)
- Backups, disaster recovery solutions

Common Migration Methods

Migration Phase	Purpose	Azure Tools Used
1. **Assessment**	Analyze existing infra, dependencies	**Azure Migrate**, Azure TCO Calculator
2. **Planning**	Choose what to move, when, and how	Azure Landing Zones, CAF (Cloud Adoption Framework)
3. **Preparation**	Set up Azure infra	Azure Virtual Network, Storage, VMs, NSGs, etc.
4. **Data Migration**	Transfer files, databases, and app data	**AzCopy, Azure Data Box, Azure File Sync, DMS**
5. **Application Migration**	Move apps/VMs to cloud	**Azure Migrate: Server Migration**, Site Recovery
6. **Testing**	Validate performance, security	Monitoring tools, App Insights
7. **Cutover & Go Live**	Switch production traffic to Azure	DNS update, decommission on-prem
8. **Optimization**	Cost, performance, auto-scaling	Azure Monitor, Advisor, Cost Management

How Is Data Physically Transferred?



Online **Offline**

Online Over Internet

- Tools: AzCopy, Azure Migrate, Azure File Sync, Data Factory, Database Migration Service (DMS)
- Used when:
  - You have high-speed internet
  - Data size is manageable (up to few TBs)

Offline – Azure Data Box

- Azure **sends a physical storage appliance** to your site (like a secure external hard drive)
- You **copy data locally**, ship it back to Microsoft.

- Microsoft **uploads it to your Azure Storage Account.**

Used when:

- **Data is massive** (tens of TBs or more)
- **Internet bandwidth is limited**
- Strict **data compliance** required

Data Box
100 TB

Data Box Disk
40 TB

Azure Copy-

AzCopy is a command-line utility by Microsoft that enables you to upload, download, and copy data to and from:

- Azure Blob Storage
- Azure File Shares
- Azure Data Lake Storage

It is lightweight, fast, secure, and optimized for bulk data transfer.

Azure Copy is Powerful tool, but why? -> because

Feature	Benefit
High Performance	Multi-threaded, parallel transfers for faster uploads/downloads
Secure Transfers	Uses **HTTPS**, supports **SAS tokens**, **Azure AD**, **OAuth**
Cross-platform	Works on **Windows, Linux, and macOS**
Smart Retry Mechanism	Automatically retries failed chunks to ensure integrity
Resumable Transfers	Resume partially completed jobs without starting over
Recursive Directory Support	Transfer entire folder structures with one command
Server-side Copy	Copy data **within Azure** without download/upload (fast + no egress)
No Extra Cost	AzCopy itself is **free to use** — you only pay for Azure data usage

Azure Identity, Access, and Security

Microsoft Entra ID *(Formerly Azure Active Directory)*

when we talk about Microsoft Entra ID in the context of Azure, we are talking about the core identity provider for accessing and managing virtually everything within your Azure environment, as well as many other Microsoft cloud services and third-party applications.

- It is a **cloud-based identity and access management (IAM) service** from Microsoft.
- It helps organizations securely manage **user identities, authentication**, and **access to Azure resources, apps, and services** — both in the cloud and on-premises.

Now, Understand the Role of Microsoft Entra ID within Azure:

Identity Provider for the Azure Portal and Azure Services:

- When you log in to the Azure portal, you are authenticating against Microsoft Entra ID. Your user account and its associated permissions *(defined through Azure Role-Based Access Control - RBAC)* are managed in Entra ID.
- Microsoft Entra ID provides the identities for users, groups, applications, and managed identities that interact with Azure resources (Virtual Machines, Storage Accounts, Databases, App Services, etc.).
- It ensures that only authorized individuals and services can access your Azure subscriptions and the resources within them.

Centralized Identity and Access Management (IAM):

- **User Management:** Create, manage, and provision user accounts *(both internal employees and external guests/collaborators*).
- **Group Management:** Organize users into groups to simplify permission assignments.
- **Application Access:** Control which users and groups can access specific applications, whether they are:
  - **Azure-hosted applications:** Apps running on App Service, VMs, Azure Functions, etc.
  - **Microsoft 365 services:** (formerly Office 365) like Exchange Online, SharePoint Online, Teams.
  - **SaaS applications:** Thousands of pre-integrated third-party SaaS apps like Salesforce, ServiceNow, Workday, Google Workspace, etc., through Single Sign-On (SSO).
  - **On-premises applications:** Securely expose on-premises web apps to external users via Microsoft Entra application proxy.

Authentication and Authorization:

- **Authentication:** Verifies the identity of users. Microsoft Entra ID supports a wide range of modern authentication methods:
  - Password-based

- Multi-Factor Authentication (MFA): Requires two or more forms of verification (e.g., password + phone notification, biometric).
- Passwordless: FIDO2 security keys, Windows Hello for Business, Microsoft Authenticator app.
- Certificate-based

- **Authorization:** Determines what an authenticated user or service is allowed to do.
  - **Azure RBAC:** Directly tied to Entra ID. You assign *roles (e.g., Reader, Contributor, Owner, Storage Blob Data Contributor*) to users, groups, or service principals defined in Entra ID, granting them specific permissions over Azure resources (subscriptions, resource groups, individual resources).
  - **Conditional Access:** A powerful feature that enforces policies based on real-time signals (user location, device compliance, sign-in risk, application being accessed) to grant or deny access, or require additional authentication (like MFA). This is a cornerstone of Zero Trust security.

Hybrid Identity (Connecting On-Premises Active Directory to Azure):

- **Microsoft Entra Connect:** This synchronization tool allows you to replicate user accounts, groups, and password hashes (or use Pass-through Authentication/Federation) from your on-premises Windows Server Active Directory Domain Services (AD DS) to Microsoft Entra ID.
- **Unified Identity:** This creates a "hybrid identity" environment where users have a single identity for accessing both on-premises resources and cloud resources (Azure, Microsoft 365, SaaS apps). They use the same username and password.

Managed Identities for Azure Resources:

- A critical security feature within Azure. Managed identities allow Azure services (like Azure VMs, Azure App Services, Azure Functions) to authenticate to other Azure services (like Azure Key Vault, Azure Storage, Azure SQL Database) without needing to manage credentials (connection strings, secrets) in your code.
- These identities are managed by Microsoft Entra ID. When an Azure service uses a managed identity, Entra ID automatically handles the authentication token exchange.

Security and Compliance:

- **Identity Protection:** Detects and remediates identity-based risks (e.g., impossible travel, leaked credentials).

- **Privileged Identity Management (PIM):** Manages, controls, and monitors access to important resources in Microsoft Entra ID, Azure, and other Microsoft online services. It provides just-in-time access and approval workflows for elevated privileges.
- **Security Defaults:** A baseline set of security policies for all tenants to improve security posture immediately.
- **Audit Logs:** Detailed logs of all identity-related activities (sign-ins, changes to users/groups/roles).

So, why this is Crucial? 

1. **Foundation of Security:** It's the primary gatekeeper for who can access your Azure environment and what they can do. Without a robust identity solution, your Azure resources are vulnerable.
2. **Scalability for Modern Workloads:** Designed to handle vast numbers of users and authentication requests, essential for large cloud deployments and distributed applications.
3. **Simplifies Access Management:** Provides a centralized hub to manage access across all your cloud resources and SaaS applications, reducing administrative overhead.
4. **Enables Hybrid Scenarios:** Bridges the gap between on-premises and cloud environments, allowing for seamless access and management across both.
5. **Empowers Developers:** Provides APIs and SDKs for applications to integrate with Entra ID for authentication and authorization, simplifying secure app development

Authentication Methods in Azure Environment

1. **Password:**

   The most traditional method, where users provide a string of characters (password) that they know. While still widely used, it's considered the least secure primary authentication method due to common vulnerabilities like phishing, brute-force attacks, and password reuse.

Use Case: Basic access for users, often combined with MFA for enhanced security. It's the fallback method and cannot be fully disabled in Microsoft Entra ID.

2. **Microsoft Authenticator App:**

   A mobile application that provides secure, passwordless, and multi-factor authentication. Users can receive push notifications on their phone to approve sign-in attempts or generate time-based one-time passwords (TOTP) codes.

   Use Case: Highly recommended for both primary passwordless sign-in and as a robust second factor for MFA, offering excellent security and user experience.

3. **FIDO2 Security Keys (Passkeys):**

   Physical hardware devices *(e.g., USB keys, often with fingerprint readers)* that provide strong, phishing-resistant, and passwordless authentication. They use public-key cryptography to verify a user's identity.

   Fido Token Device

   Use Case: Ideal for high-security environments, critical roles, and users who frequently access sensitive data. They offer a superior level of security by being device-bound and resistant to phishing.

4. **Windows Hello for Business:**

   A biometric-based (face or fingerprint recognition) or PIN-based authentication method built into Windows devices. It allows users to sign in to their Windows device and automatically authenticate to Microsoft Entra ID without a password.

   Use Case: Excellent for corporate-managed Windows devices, providing a convenient and secure passwordless experience for employees.

5. **OATH Tokens (Hardware and Software):**

   Open-standard solutions for generating one-time passwords (OTPs).

   a. **Hardware OATH Tokens:** Small physical devices that display an OTP that change every 30 or 60 seconds.

b. **Software OATH Tokens:** Authenticator applications (like Microsoft Authenticator, Google Authenticator, etc.) that generate OTPs on a smartphone.

Use Case: Primarily used as a second factor for MFA, especially when push notifications might not be reliable or for users who prefer a code-based method.

6. **SMS (Text Message) Verification:**

   A verification code is sent to the user's registered mobile phone number via SMS. The user then enters this code into the sign-in interface to verify their identity.

   Use Case: A common and convenient second factor for MFA, especially for users who may not have a smartphone with an authenticator app. Also supports SMS-based passwordless sign-in for front-line workers.

7. **Voice Call Verification:**

   An automated phone call is made to the user's registered phone number. The user is prompted to press a key (e.g., '#') on their phone keypad to approve the sign-in.

   Use Case: Another common second factor for MFA, providing an alternative to SMS for users who prefer or require a voice call.

8. **Temporary Access Pass (TAP):**

   A time-limited passcode that can be used once or for a short period to sign in or set up other authentication methods (like a FIDO2 key or Microsoft Authenticator). It's primarily for onboarding new users, recovering accounts, or providing temporary access.

   Use Case: Great for helpdesk scenarios, new employee onboarding, or when a user has lost all other authentication methods.

9. **Certificate-Based Authentication (CBA):**

   Users authenticate using a digital certificate stored on a smart card, a trusted device, or a virtual smart card. This provides a strong, phishing-resistant form of authentication.

   Use Case: Common in highly secure environments (e.g., government, defense) where organizations already leverage Public Key Infrastructure (PKI) and smart cards.

10. App Passwords:

Auto-generated, unique passwords designed for older, non-browser-based applications (like older versions of Outlook or other mail clients) that don't support modern authentication protocols (MFA or SSO).

Use Case: A legacy method used only when per-user MFA is enabled, and an application does not support modern authentication. Generally discouraged in favor of modern protocols.

Single Sign ON (SSO)

Allows users to sign in once with a single set of credentials and then gain access to multiple independent applications and services without having to re-enter their credentials for each one.

Think of it as having **one master key** that unlocks many doors. Instead of having a different key (username/password) for your email, your CRM, your HR system, and your Azure portal, you use just one key.

Now, How SSO works with Microsoft Entra ID? 

Microsoft Entra ID acts as the **Identity Provider (IdP)** in the SSO flow. Applications that use SSO are called **Service Providers (SPs)**. The core process generally involves:

1. **User Initiates Access:** A user tries to access an application (Service Provider), like Salesforce, Microsoft 365 (e.g., Outlook), or a custom web app hosted in Azure.
2. **Redirection to IdP:** The application detects that the user isn't authenticated and redirects the user's browser to Microsoft Entra ID (the Identity Provider) for authentication.
3. **Authentication with Entra ID:**
    - If the user is already signed in to Microsoft Entra ID (perhaps they just logged into their Windows machine, or accessed another Azure service), Microsoft Entra ID recognizes their active session.

- If not, Microsoft Entra ID prompts the user for their credentials (username, password, MFA, etc.).

4. **Token Issuance:** Upon successful authentication, Microsoft Entra ID generates a **security token** (e.g., SAML assertion, OAuth 2.0 access token, OpenID Connect ID token). This token contains information about the authenticated user and their identity.
5. **Redirection Back to SP with Token:** Microsoft Entra ID redirects the user's browser back to the original application (Service Provider), including the newly issued security token.
6. **Token Validation and Access:** The application (Service Provider) receives and validates this token using a pre-established trust relationship with Microsoft Entra ID (e.g., a shared certificate). Once validated, the application grants the user access without requiring them to enter credentials again.

Key Protocols Involves in the SSO Process.

SAML 2.0 (Security Assertion Markup Language): Widely used for enterprise applications. It's an XML-based standard for exchanging authentication and authorization data between an identity provider and a service provider. Many SaaS applications support SAML SSO.

OAuth 2.0 and OpenID Connect (OIDC): OAuth 2.0 is an authorization framework that allows third-party applications to obtain limited access to user accounts on an HTTP service. OpenID Connect is an authentication layer built on top of OAuth 2.0, providing identity verification and basic profile information. These are widely used for modern web and mobile applications.

WS-Federation: An older protocol often used for integrating with on-premises Active Directory Federation Services (AD FS) or other Microsoft technologies.

Password-based SSO (Password Vaulting): For applications that don't support modern federation protocols. Microsoft Entra ID securely stores and "replays" the user's credentials to the application's login page on their behalf. This is less secure than federated SSO and typically used as a fallback.

Seamless Single Sign-On (Seamless SSO): A feature that works specifically for users on corporate devices connected to the corporate network (or VPN). It leverages Kerberos to provide a truly seamless experience where users might not even need to type their username for cloud-based apps.

Now, the Question is which protocol is widely and Most Used in SSO -Azure Environment?

OpenID Connect (OIDC) Most Common & Modern

Because- It's built on top of OAuth 2.0 and supports secure **authentication** (not just authorization) for **web, mobile, and cloud-native apps**.

Others are also used, but less preferred

Conditional Access

- Microsoft's powerful, policy-driven security engine that allows organizations to enforce dynamic access controls based on real-time signals. At its core, a Conditional Access policy is an "if-then" statement:
- **IF** certain conditions are met (e.g., a user is trying to access from an untrusted location, or their sign-in is deemed risky).
- **THEN** apply specific access controls (e.g., require Multi-Factor Authentication, block access entirely, or require a compliant device).

Fundamental Elements of Conditional access: -

Users and Groups (Assignments - Who?):

- You define *who* the policy applies to. This can be specific users, security groups, or even all users.
- Crucially, you should always **exclude emergency access ("break-glass") accounts** from Conditional Access policies to prevent accidental lockout of all administrators.

Cloud Apps or Actions (Assignments - What?):

- You define *what* resources or applications the policy applies to. This can be:
  - All cloud apps (a common starting point).
  - Specific Microsoft cloud apps (e.g., Microsoft 365, Azure Management, Exchange Online, SharePoint Online).
  - Specific third-party SaaS applications integrated with Microsoft Entra ID.
  - On-premises applications published through Microsoft Entra Application Proxy.

- Specific user actions, like "Register security information" (for MFA setup) or "Register or join devices."

Conditions (If - When? Where? How?):

- These are the signals and attributes that trigger the policy. This is where the "intelligence" of Conditional Access comes in. Common conditions include:
  - **User risk:** Assessed by Microsoft Entra ID Protection (e.g., impossible travel, leaked credentials, suspicious IP address).
  - **Sign-in risk:** Assessed in real-time during a sign-in attempt (e.g., user signing in from an unfamiliar location or device).
  - **Device platforms:** Windows, macOS, iOS, Android, Linux.
  - **Locations:** Specific IP ranges (e.g., your corporate network), named locations (e.g., "trusted offices"), or specific countries/regions.
  - **Client apps:** Browser, mobile apps, desktop apps, legacy authentication clients (e.g., ActiveSync, IMAP).
  - **Device state:** Whether the device is marked as compliant (from Intune/Microsoft Endpoint Manager), or hybrid Microsoft Entra joined (joined to both on-premises AD and Microsoft Entra ID).
  - **Filter for devices (Preview):** Target policies to specific devices (e.g., privileged access workstations).

Access Controls (Then - What to do?):

- These are the actions that are enforced when the conditions are met.
  - **Block access:** Deny access entirely.
  - **Grant access:** Allow access, but with additional requirements (one or more can be selected):
    - **Require Multi-Factor Authentication (MFA):** The most common control.
    - **Require device to be marked as compliant:** Device must meet security standards defined in Intune.
    - **Require Microsoft Entra hybrid joined device:** Device must be joined to your on-premises AD and Microsoft Entra ID.

- **Require approved client app:** Only allow access from specific, approved mobile apps (e.g., Outlook Mobile, Teams app).
- **Require app protection policy:** Apply mobile application management (MAM) policies to protect data within the app.
- **Require password change:** Force the user to reset their password (often used for high-risk users).
- **Require Terms of Use:** Force the user to accept terms of use before accessing.

○ **Session Controls:**

- **Use app enforced restrictions:** Integrates with Microsoft Defender for Cloud Apps to provide real-time monitoring and control over sessions (e.g., block downloads of sensitive data).
- **Use Conditional Access App Control:** Similar to above.
- **Sign-in frequency:** Control how often users are prompted for authentication (e.g., re-authenticate every hour).
- **Persistent browser session:** Control whether users remain signed in after closing their browser.

Policies Under conditional Access-:

Policy Component	Description
Users or Groups	Specifies **who the policy applies to** – individual users, groups, or roles
Cloud Apps / Actions	Targets **which apps or actions** the policy applies to (e.g., SharePoint, Teams, sign-in)
Conditions	Defines **when the policy applies** based on multiple signal types
→ **Sign-in Risk**	Applies if Microsoft Entra ID detects a risky sign-in
→ **User Risk**	Applies if user's identity is flagged as risky
→ **Device Platform**	Target based on OS (Windows, iOS, Android, etc.)
→ **Locations**	Apply based on IP location (e.g., block access outside corporate IP range)
→ **Client Apps**	Specify browser, mobile apps, legacy clients

→ **Device State**	Apply based on whether device is compliant or hybrid Azure AD joined
Access Controls	Define what happens when policy is triggered
→ **Grant Controls**	Allow access but require MFA, compliant device, or hybrid AD join
→ **Block Access**	Deny sign-in if conditions are not met
→ **Session Controls**	Limit user session (e.g., limit access to download or set timeouts)
Enable Policy	Enable, disable, or set as report-only

Role based access Control

Role-Based Access Control (RBAC) in Azure is a security model that allows you to control who can access specific Azure resources, what they can do with those resources, and at what scope.

In simple terms:

RBAC defines "who can do what, where" in your Azure environment.

Key Concepts: -

Concept	Description
Security Principal (Who)	A **user**, **group**, **service principal**, or **managed identity** that needs access
Role Definition (What)	A set of **permissions** like read, write, delete — e.g., Reader, Contributor, Owner
Scope (Where)	The level at which access is applied — **Subscription**, **Resource Group**, or **Resource**

Common Built-in Roles in Azure RBAC.

Azure RBAC comes with over 100 built-in roles, which cover common scenarios. These roles define a collection of permissions, specifying actions that can be performed (like read, write, delete) and "not actions" (actions that are explicitly denied).

See the common Built-in Roles below: -

1. **Owner:**
    - This is the highest level of access within a given scope. An Owner has full access to manage all resources, and critically, can **assign roles** to other users, groups, service principals, or managed identities.
    - **Use Case:** Typically assigned only to a very small number of individuals who need full control over a subscription or resource group, such as lead architects, subscription administrators, or highly trusted team leads. It should be used sparingly due to its broad permissions.
2. **Contributor:**
    - A Contributor can create, manage, and delete all types of Azure resources within a given scope. However, unlike an Owner, a Contributor **cannot assign roles** to other users or manage access permissions (they don't have Microsoft. Authorization/* permissions).
    - **Use Case:** Ideal for developers, DevOps engineers, or IT operations teams who need to deploy and manage resources (e.g., virtual machines, storage accounts, databases, web apps) but should not have the ability to change who has access.
3. **Reader:**
    - A Reader can view all Azure resources within a given scope but **cannot make any changes** to them. This role grants only read access.
    - **Use Case:** Suitable for monitoring, auditing, or reporting purposes. For example, a finance team member might need Reader access to view resource costs, or a security auditor might need to review configurations without modifying them.
4. **User Access Administrator:**
    - This role is specifically designed to manage user access to Azure resources. A User Access Administrator can **manage role assignments** for other users, groups, and service principals. They also have */read permissions.
    - **Use Case:** Often assigned to security administrators or identity administrators who are responsible for granting and revoking access permissions to various teams or individuals within the Azure environment. They don't manage the resources themselves, only who can access them.

Other Important Built-in Roles (often more specialized):

Beyond the fundamental four, there are many other built-in roles tailored for specific services or functions:

5. **Network Contributor:**
    - can manage all networking resources, such as virtual networks, network interfaces, load balancers, and network security groups.
    - **Use Case:** Ideal for network administrators or teams responsible for configuring and maintaining Azure networking infrastructure.
6. **Storage Blob Data Contributor / Storage Blob Data Reader / Storage Blob Data Owner:**
    - These roles provide specific permissions to interact with **data** within Azure Storage Blobs (containers and blobs), rather than just managing the storage account itself.
        - **Contributor:** Read, write, and delete blob data.
        - **Reader:** Read blob data.
        - **Owner:** Full access to blob data, including assigning blob data roles.
    - **Use Case:** Used for applications or users who need to read from or write to specific blob containers (e.g., a web app uploading images to a storage container). These are distinct from the Contributor role on the storage account, which manages the account properties but not necessarily the data inside.
7. **Virtual Machine Contributor:**
    - Can create and manage virtual machines but cannot manage the virtual network or storage account to which the VM connects (unless they also have permissions on those resources).
    - **Use Case:** For IT teams dedicated to VM deployment and management.
8. **SQL DB Contributor:**
    - Can create and manage SQL databases, but not their security-related policies or access.
    - **Use Case:** For database administrators who need to manage SQL databases within Azure.

9. **Key Vault Contributor / Key Vault Reader / Key Vault Secrets User / Key Vault Crypto Service Encryption User, etc.:**
    - Various roles for managing Azure Key Vault and the secrets, keys, and certificates stored within it. Permissions are very granular for Key Vault.
    - **Use Case:** Security teams managing secrets, developers needing to retrieve secrets, or applications requiring access to cryptographic keys.

*well-organized **summary table***

Role Name	Definition	Typical Use Case
Owner	Full control over all resources, including the ability to assign roles	Lead architects, subscription admins, or highly trusted team leads
Contributor	Can create, manage, and delete resources; **cannot assign roles**	DevOps, developers, IT ops who manage Azure resources
Reader	View-only access to all resources within scope	Finance, auditors, security reviewers
User Access Administrator	Can assign roles to users, groups, and service principals; **no resource management**	Security or identity administrators managing access rights
Network Contributor	Manage all network resources (VNETs, NICs, NSGs, LB, etc.)	Network engineers managing Azure network setup
VM Contributor	Manage virtual machines only; **not the network or storage linked to VM**	IT teams deploying/managing Azure VMs
Storage Blob Data Contributor	Read, write, delete blob data (not storage account settings)	Apps or users uploading, processing blob content
Storage Blob Data Reader	Read blob data only	Web apps serving static content from blob storage
Storage Blob Data Owner	Full blob data access + assign blob roles	Data owners managing sensitive file systems on Azure
SQL DB Contributor	Create and manage SQL databases; **limited security control**	DBAs managing SQL resources in Azure
Key Vault Contributor	Manage Key Vaults, secrets, keys, and certificates	Security admins managing secret lifecycle

Key Vault Reader	Read metadata of Key Vaults only (not secrets or keys)	Monitoring teams or inventory audits
Key Vault Secrets User	Retrieve secrets from Key Vault	Developers or apps that need access to secrets (like connection strings)
Key Vault Crypto Service Encryption User	Use cryptographic keys in vault for encryption/decryption	Apps or systems performing data encryption/decryption using Azure Key Vault

Understand RBAC with example-

A Company Hosting a Web Application in Azure

Company: Abhishek and Sons Private Limited

They've built a cloud-based **e-commerce website** hosted entirely in Azure. The architecture includes:

- Azure Web Apps for the frontend
- Azure SQL Database for backend
- Azure Storage Blob for product images
- Azure Virtual Machines for custom backend processing
- Azure Key Vault for storing API keys and DB credentials
- Azure Virtual Network for secure traffic routing

Team Roles/Assignments-

Team Member / Role	Assigned Azure Role	Why This Role?
Raj (Cloud Architect)	**Owner** of the Subscription	Needs full access to manage all services + assign permissions to team
Priya (DevOps Engineer)	**Contributor** on Resource Group WebAppRG	Can deploy/update resources like Web Apps, VM, Storage but **not assign roles**
Anjali (Network Admin)	**Network Contributor** on VNet + NSG RG	Manages virtual network, VPN, NSGs — no access to VMs or apps

Mohit (Database Admin)	**SQL DB Contributor** on SQL Resource	Manages Azure SQL DB: tables, indexes, queries — no access to app or infra
Aarav (Security Admin)	**User Access Administrator** on subscription	Assigns/removes user access and roles as per audit or policy requirements
Frontend Dev Team	**Reader** on entire WebAppRG	Can monitor settings, logs, configs — **no permission to edit** or deploy
Image Uploader Service	**Storage Blob Data Contributor**	Uploads/deletes product images to Azure Blob Storage
Web App Code	**Key Vault Secrets User**	Retrieves DB connection string securely during app startup

What This Achieves:

- **Security**: Least privilege model — no one has more access than needed.
- **Compliance**: Security team handles access; developers don't touch secrets.
- **Auditability**: Clear separation of roles, traceable permissions.
- **Scalability**: Easily extendable if team grows or services expand.

Azure Security /Defense in Depth

***("Defense in Depth** is a cybersecurity strategy that applies a layered approach to security.")*

"Defense in Depth is a foundational principle that guides how Microsoft designs and operates its cloud infrastructure, and how organizations should secure their own workloads deployed on Azure."

Why is it Crucial?

- **No Single Point of Failure:** No single security control is 100% foolproof. By layering controls, if one fails or is bypassed, another layer can still stop the attack.
- **Reduced Risk:** It significantly reduces the overall risk of a successful cyberattack by increasing complexity for attackers.

- **Comprehensive Protection:** It addresses various attack vectors and vulnerabilities across different aspects of an IT environment.
- **Shared Responsibility Model:** In the cloud, security is a shared responsibility between Microsoft and the customer. Defense in Depth applies to both sides of this model. (Microsoft implements it for the underlying cloud infrastructure, and customers implement it for their data and applications running on that infrastructure.) ➔ v.v Important

Now, there are **7** Layers in this Defense in depth Model- (let's see!)

Layer	Description
1. Physical Security	Microsoft secures its **datacenter facilities** through fencing, guards, biometric access, etc.
2. Perimeter Security	Protects the **network boundary** using DDoS protection, firewalls, and network security services
3. Network Security	Uses **NSGs**, **Azure Firewall**, **routing**, **VPNs**, **private endpoints**
4. Compute Security	Security for **VMs**, **containers**, and **App Services** — including patching, and endpoint protection
5. Application Security	Secure your apps using **secure coding practices**, **WAF**, **API management**, **authentication**
6. Identity & Access	Use of **Microsoft Entra ID**, **RBAC**, **MFA**, **Conditional Access**
7. Data Security	Encryption at rest & in transit, **Azure Key Vault**, **Access policies**

WHY? 7 Layers

1. Physical Security

- Azure datacenters are **certified ISO 27001**
- Security staff, surveillance, limited personnel access
- **Importance:** First line of defense. If physical infrastructure is compromised, all other layers may be at risk.

2. Perimeter Security

- **Azure DDoS Protection**, perimeter firewalls, traffic monitoring
- **Importance:** Prevents large-scale external attacks from reaching internal systems

3. Network Security

- Configure **NSGs (Network Security Groups)** to allow/deny traffic
- Use **VNET peering**, **Azure Firewall**, **Private Endpoints**
- **Importance:** Prevents **east-west traffic attacks** (lateral movement), and secures internal communications

4. Compute Security

- Apply **updates**, **anti-malware**, **just-in-time VM access**
- Secure containers using **Azure Kubernetes Service (AKS)** security practices
- **Importance:** Secures operating systems and workloads from exploits and malware

5. Application Security

- Implement **Azure Web Application Firewall (WAF)**, **API Management**, input validation
- Use **Azure Defender for App Services**
- **Importance:** Prevents attacks like SQL injections, cross-site scripting, and session hijacking

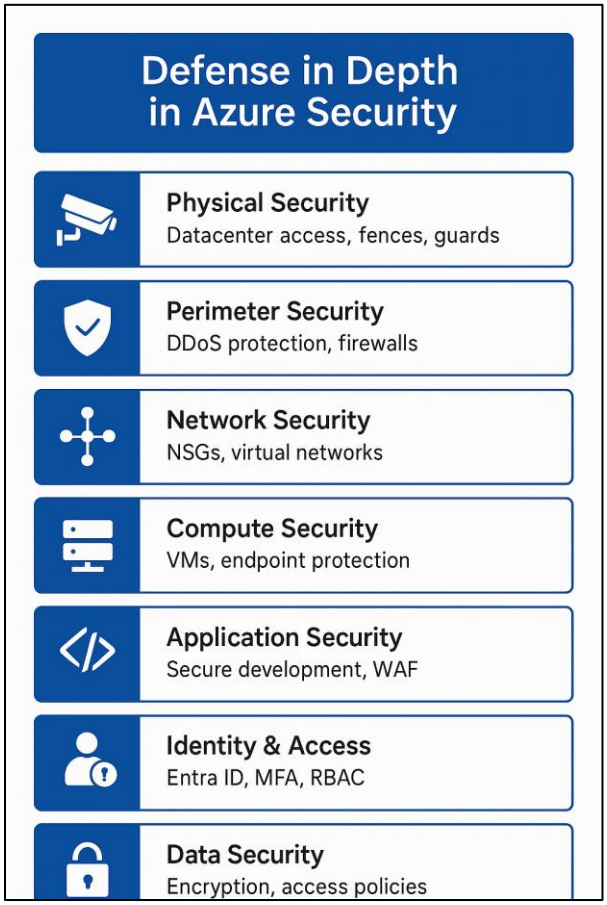
6. Identity and Access Management

- Use **Azure AD (Entra ID)** for managing users and app identities

- Enforce **Multi-Factor Authentication (MFA)** and **Conditional Access**
- **Importance:** Prevents unauthorized access and enforces the principle of **least privilege**

7. Data Security

- Data encrypted with **Azure Storage encryption**, **Disk encryption**, **TLS**
- Secure access using **Key Vault**, **RBAC**, **data masking**
- **Importance:** Data is the **ultimate target** in most breaches — securing it is crucial



Zero trust Policy-

- Zero Trust is a modern security framework that replaces the outdated perimeter-based model with a more dynamic, risk-aware approach.
- Unlike the traditional "trust but verify" mindset—where anything inside the corporate network was automatically considered safe, Zero Trust follows the principle of "never trust, always verify."
- This strategy assumes that security breaches are inevitable and that threats can arise from both external sources and within the organization. "As a result, every access request, whether from a user, device, or application—must be explicitly authenticated" and authorized before being granted, regardless of its origin or destination.

How Zero Trust Works in Azure (Using Microsoft Entra ID)?

*Combination of (**identity** + **device** + **application** and **network signals**) = [enforced through policies and services.]*

Pillar	How It's Enforced in Azure
Verify Explicitly	Every user or device must prove who/what it is before gaining access
Least Privilege Access	Users only get access to what they absolutely need (enforced via RBAC, PIM)
Assume Breach	Act as though attackers are already inside the network – limit lateral movement

Key Technologies Used in Azure Zero Trust

Capability	Azure Service or Feature	Purpose

Identity Management	Microsoft Entra ID (Azure AD)	Central identity platform; handles authentication & SSO
Conditional Access	Entra Conditional Access Policies	Allows or blocks access based on conditions like device, location
Multi-Factor Authentication (MFA)	Microsoft Entra MFA	Requires second verification step (e.g., mobile code)
Device Compliance	Microsoft Intune + Entra ID device registration	Ensures only trusted, compliant devices are allowed
Privileged Identity Management	Azure PIM	Just-in-time access to sensitive roles or resources
App Protection	Microsoft Defender for Cloud Apps	Monitors and governs user actions within SaaS apps
Network Controls	Azure Network Security Groups (NSG), Azure Firewall, Private Endpoints	Limits access paths and exposure within the network
Data Protection	Azure Information Protection, Key Vault	Encrypts and classifies sensitive data
Threat Detection	Microsoft Defender for Cloud, Sentinel (SIEM)	Detects and responds to suspicious activities or threats

Real-Life Example of a Zero Trust Policy in Azure: 

Let's consider a company that stores highly sensitive customer data in an Azure SQL Database and uses Microsoft 365 for daily operations. They have a mix of company-managed laptops and employee-owned mobile devices (BYOD). Their employees sometimes work from the office, sometimes from home, and occasionally from public Wi-Fi.

Here's how a Zero Trust policy, built with Azure capabilities, would work:

Scenario: An employee, Sarah, needs to access the sensitive customer database.

Traditional Security Model:

- If Sarah is in the office, connected to the corporate network, she's automatically trusted. Her device might not be heavily scrutinized. If an attacker compromises her laptop inside the network, they might have relatively free rein.

Zero Trust Policy in Azure:

1. **Verify Explicitly (Identity & Device):**
    - **Authentication (Microsoft Entra ID + MFA):** When Sarah tries to access the SQL Database (or any sensitive app), Microsoft Entra ID **always** requires her to perform Multi-Factor Authentication (MFA), even if she's in the office. No implicit trust based on network location.
    - **Device Compliance (Microsoft Intune + Conditional Access):**
        - Azure Conditional Access policy is configured*: "If user accesses the 'Sensitive Customer Database' app, THEN require device to be marked as compliant."*
        - Sarah's laptop (company-managed) is enrolled in Intune. Intune continuously checks if the laptop has the latest OS updates, antivirus definitions, disk encryption, and no known vulnerabilities. If it's compliant, it's marked as "compliant" in Microsoft Entra ID.
        - If Sarah tries to access the database from her personal mobile phone, the Conditional Access policy would block access if the phone isn't enrolled in Intune and compliant.
2. **Verify Explicitly (Context - Location & Risk):**
    - **Location-based Conditional Access:** Another Conditional Access policy is in place: "If user accesses 'Sensitive Customer Database' from an untrusted country (e.g., a known high-risk nation) OR from a public/unknown IP address, THEN block access."
    - **Sign-in Risk Detection (Microsoft Entra ID Protection):** If Sarah's sign-in behavior is deemed "risky" by Microsoft Entra ID Protection (e.g., she just signed in from New Delhi and now, 5 minutes later, there's an attempt from Paris, or her credentials have been detected in a public data breach), a Conditional Access policy could trigger: "If sign-in risk is 'High', THEN block access OR require password reset immediately and re-authenticate with MFA."

3. **Least Privilege Access (Azure RBAC & JIT):**
    - **Azure RBAC:** Sarah is granted the "SQL DB Contributor" role *only* on the specific Resource Group containing the Sensitive Customer Database. She doesn't have "Owner" or "Contributor" on the entire subscription.
    - **Privileged Identity Management (PIM):** If Sarah needs *elevated* access (e.g., "SQL DB Owner" to change database settings), she must explicitly request it through PIM. This access is granted for a limited time (e.g., 1 hour) and requires approval from her manager, and possibly MFA again. After 1 hour, her elevated permissions are automatically revoked.
4. **Assume Breach (Monitoring & Segmentation):**
    - **Network Micro-segmentation (NSGs, Azure Firewall):** The Azure SQL Database is placed in a dedicated subnet, separated from other applications and the internet by Network Security Groups (NSGs) and Azure Firewall. Only necessary traffic (e.g., from the application server, specific jump boxes) is allowed to reach the database, blocking lateral movement from other compromised resources.
    - **Continuous Monitoring (Azure Monitor, Microsoft Sentinel):** All access attempts to the database, successful or failed, along with user and device health signals, are logged and fed into Azure Monitor and Microsoft Sentinel (SIEM).
    - **Anomaly Detection:** Microsoft Sentinel uses AI and analytics to detect unusual behavior (e.g., Sarah trying to export an unusually large amount of data or accessing the database at 3 AM from an unfamiliar country).
    - **Automated Response:** If a severe anomaly is detected, an automated playbook might trigger, temporarily blocking Sarah's access or isolating her device until an investigation can be performed.

Outcome: Under a Zero Trust model, Sarah's access to the sensitive database is continuously evaluated. Her identity, device, location, and the perceived risk of her activity are all factored into the access decision *for every request*. No matter if she's inside or outside the traditional network perimeter, she's never implicitly trusted. This significantly reduces the attack surface and minimizes the impact of a potential breach, as an attacker would face multiple, dynamic hurdles at every step.

Azure Management and Governance

Cost management in Azure

Azure operates primarily on a consumption-based (Pay-As-You-Go) model, where you only pay for the resources, you consume. However, there are various pricing options and levels:

- **Pay-As-You-Go:** You're billed for actual usage *(compute hours, storage capacity, data transfer, transactions, etc.)* without upfront commitment. Offers maximum flexibility.
- **Reserved Instances (RIs):** For predictable, steady-state workloads *(e.g., Virtual Machines, SQL Database, Cosmos DB).* You commit to a 1-year or 3-year term for a specific resource type and region, receiving significant discounts (up to 72% or more). Payment can be upfront or monthly.
- **Azure Savings Plans for Compute:** A newer, more flexible commitment model *(1-year or 3-year term)*. You commit to spending a fixed hourly amount on compute services across various regions, instance types, and even different compute services. Offers up to 65% savings. Ideal for dynamic workloads where your specific resource needs might change but overall compute usage is consistent.
- **Spot Instances/VMs:** Leverage unused Azure compute capacity at significantly reduced prices (up to 90% off Pay-As-You-Go). However, these instances can be interrupted by Azure with short notice if capacity is needed back. Suitable for fault-tolerant workloads, batch processing, or development/testing.
- **Azure Hybrid Benefit:** Allows you to reuse your on-premises Windows Server and SQL Server licenses (with active Software Assurance) on Azure VMs and Azure SQL Database, significantly reducing costs for the software component. Also applies to Red Hat and SUSE Linux subscriptions.
- **Free Services:** Azure offers a range of services that are free indefinitely, for 12 months, or for 30 days, or up to a certain usage threshold.

Azure Cost Management + Billing (The Core Tool)

This is the primary service in the Azure portal for monitoring, analyzing, and optimizing your cloud costs.

Key Features:

- Cost Analysis:
  - **Detailed Views:** Provides interactive charts and tables to visualize your spending across various dimensions (services, resources, resource groups, tags, locations, time periods).
  - **Filtering and Grouping:** Allows you to drill down into costs by specific criteria (e.g., cost by resource group, cost by tag for a specific department).
  - **Forecasting:** Predicts future spending based on historical usage patterns, helping you anticipate potential overruns.
  - **Actual vs. Amortized Costs**: Shows your real-time consumption vs. how reserved instance benefits are applied over time.
- **Budgets**:
  - **Set Spending Limits:** Define custom spending thresholds for subscriptions, resource groups, or management groups (higher level for organizational hierarchy).
  - **Alerts:** Configure alerts to notify you via email, SMS, or webhooks when your actual or forecasted spending approaches or exceeds a defined percentage of your budget (e.g., 80%, 100%).
  - **Action Groups:** Integrate with Azure Action Groups to trigger automated actions (e.g., scale down VMs, send notifications to Teams/Slack) when budget thresholds are crossed.
  - Time Grain: Budgets can be set for monthly, quarterly, or annual periods.
- **Exports:**
  - **Automated Data Export:** Schedule exports of your detailed cost and usage data to an Azure Storage account. This is crucial for integrating cost data with external financial systems, data warehouses, or BI tools (like Power BI).
  - **Cost Details API:** Programmatic access to your cost and usage data for custom reporting and integration.
- **Cost Allocation**:
  - **Chargeback/Showback:** Helps allocate shared costs (e.g., networking, shared services) to specific departments, projects, or cost centers using rules and tag inheritance. This enables better financial accountability within an organization.

- **Invoices and Billing Profiles:**
    - While distinct from "Cost Management," the "Billing" section manages your billing accounts, payment methods, and provides access to your monthly invoices and billing history.

Azure Cost Optimization Best Practices & Tools

Beyond the core Cost Management + Billing features, effective cost optimization involves a strategic approach:

- **Right-sizing Resources:**
    - **Azure Advisor:** This free personalized cloud consultant analyzes your Azure usage and configuration and provides actionable recommendations to optimize costs (e.g., identify idle VMs, recommend right-sizing underutilized VMs, suggest purchasing Reserved Instances). This is often the first stop for optimization.
    - **Monitor Utilization:** Use Azure Monitor to track CPU, memory, and disk utilization of your VMs to identify over-provisioned resources.
- **Leverage Discounts:**
    - **Azure Reservations and Savings Plans:** Actively identify workloads suitable for RIs or Savings Plans.
    - **Azure Hybrid Benefit:** Ensure you're utilizing existing Windows Server and SQL Server licenses.
    - **Dev/Test Pricing:** For development and testing environments, leverage special pricing available for certain Azure services.
- **Elasticity and Automation:**
    - **Autoscaling:** Implement autoscaling for Virtual Machine Scale Sets, App Services, and other services to automatically adjust resource capacity based on demand, ensuring you only pay for what you need.
    - **Automated Start/Stop Schedules:** For non-production environments (dev, test, staging), implement schedules to automatically shut down VMs or other resources during off-hours (evenings, weekends) when not in use.
    - **Serverless Computing (Azure Functions, Azure Logic Apps):** Pay only for the execution time and resources consumed by your code, ideal for event-driven or intermittent workloads.

- **Storage Optimization:**
  - **Storage Tiers:** Utilize different storage tiers (Hot, Cool, Archive) in Azure Blob Storage based on data access frequency. Move less frequently accessed data to cooler or archive tiers to reduce costs.
  - **Lifecycle Management:** Implement policies to automatically transition data between tiers or delete old data.
- **Tagging and Resource Organization:**
  - **Comprehensive Tagging Strategy:** Apply consistent tags *(e.g., environment:production, project:newapp, department:finance, owner:john.doe)* to all your resources. This is crucial for accurate cost analysis, chargeback, and reporting, allowing you to slice and dice costs by business unit or project.
  - **Resource Groups and Management Groups:** Organize resources logically into resource groups and then organize subscriptions into management groups for hierarchical cost visibility and governance.
- **Clean Up Unused Resources:**
  - Regularly audit your environment for idle or unattached resources (e.g., unattached disks, old public IPs, unused network interfaces, stale snapshots) and delete them to stop incurring charges.
- **Network Cost Management:**
  - Monitor data transfer costs, especially egress (data leaving Azure). Optimize network architectures to minimize unnecessary data movement across regions or to the internet.
  - Use private endpoints to access Azure services privately, reducing public egress costs.
- **FinOps Culture:**
  - Foster a culture of FinOps (Cloud Financial Operations) within your organization. This involves collaboration between finance, IT operations, and development teams to manage cloud costs effectively, making cost optimization a shared responsibility.

General example of Costing of Major Azure Services:

Azure Service	Description	Approx. Cost (USD/hr)	Pricing Notes
Virtual Machines (VMs)	Compute (VM types per category)	Memory-Optimized: ~$0.126/hr	Varies by region, OS, licensing
App Service Plan (Standard S2)	Managed web hosting	~$0.20/hr per instance	Will double cost if scaled out
Azure Functions (Consumption Plan)	Serverless compute, pay-per-execution	~$0.000016 per GB-s + $0.20 per million executions	Ideal for event-based workloads
Azure Functions Premium V2	Dedicated compute for Functions	$0.532/hr (4 vCPU, 14 GB)	Always-ready, VNET/GPU support
Azure Container Instances (ACI)	Serverless containers	Approx. $0.0025–0.01/hr per vCPU & GB mem*	Depends on CPU & memory allocated
Azure Kubernetes Service (AKS) – nodes	Container orchestration via VM nodes	Same as VMs (plus free control plane)	Autoscaling charges vary
Azure Blob Storage (Hot tier, LRS)	Object storage	~$0.0184 per GB-month → ≈ $0.000025/hr/GB	Billing for storage + operations
Azure SQL Database (DTU model)	Managed SQL	S2: ~$0.11/hr (100 DTUs)	VCore model varies
Azure VPN Gateway (VpnGw1)	Site-to-site connectivity	~$0.05/hr + data transfer	Pricing increases with SKU & bandwidth

Azure Pricing Calculator and the Total Cost of Ownership (TCO) Calculator.

Azure Pricing Calculator

The Azure Pricing Calculator is a **free, web-based tool** that allows you to generate **estimated monthly costs for specific Azure services** based on your anticipated configurations and usage. It's designed for granular, service-by-service cost estimation for your cloud solutions.

How it works:

1. **Select Azure Services:** You start by adding the specific Azure services you plan to use *(e.g., Virtual Machines, Storage Accounts, SQL Databases, Azure Functions, Networking components, etc.).* There are hundreds of services to choose from.
2. **Configure Parameters:** For each selected service, you configure its specific parameters. This is where you define the size, type, region, redundancy, usage hours, number of instances, performance tiers, data transfer amounts, and other relevant metrics. For example:
    - **For a Virtual Machine:** OS type (Windows/Linux), VM size (series, vCPUs, RAM), region, operating hours per month (e.g., 730 hours for 24/7), pricing model (Pay-As-You-Go, 1-year Reserved Instance, 3-year Reserved Instance, Azure Savings Plan).
    - **For Storage:** Storage account type (Blob, File, Queue, Table), performance tier (Standard/Premium), access tier (Hot/Cool/Archive), redundancy (LRS, ZRS, GRS, RA-GRS), and estimated data capacity and transaction volume.
3. **Review Estimate:** As you configure services, the calculator automatically updates to display an estimated monthly cost for each service and a total estimated monthly cost. It often breaks down costs into upfront and monthly components if you choose reserved instances or savings plans.
4. **Export and Share:** You can export your estimate to an Excel file for detailed analysis, sharing with stakeholders, or saving for future reference. You can also save your estimate online if you're logged into an Azure account.

Key Uses:

- **Pre-deployment planning:** Get an up-front estimate of costs before deploying resources.
- **Budgeting:** Help establish budgets for new projects or existing workloads.

- **Cost comparison:** Compare costs of different service configurations (e.g., different VM sizes, storage tiers) or pricing models (Pay-As-You-Go vs. Reserved Instances).
- **Scenario analysis:** Model "what-if" scenarios for scaling up or down, or for different architectural choices.
- **Transparency:** Provides a detailed breakdown of costs for each component of your solution.

Azure Total Cost of Ownership (TCO) Calculator

Its primary purpose is to help you **compare the costs of running your workloads on-premises versus running them in Azure**. *[over a specified period (typically 1 to 5 years) and highlight potential savings]*

How it works:

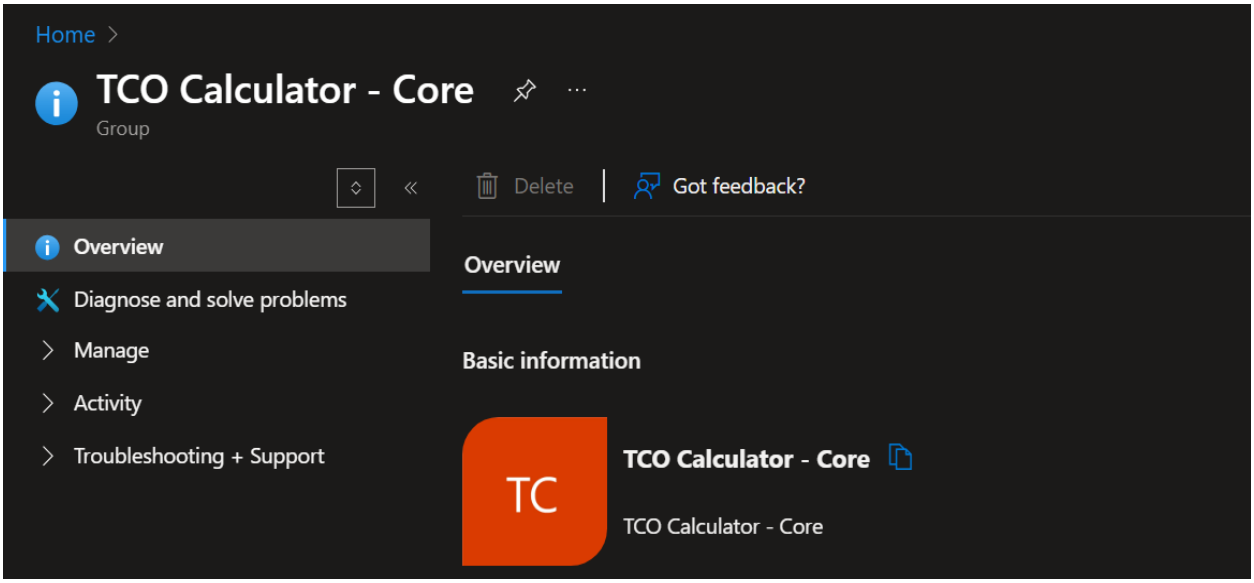
1. **Define Your Current On-Premises Infrastructure:** You input details about your current on-premises environment, including:
   - **Servers:** Number of physical or virtual servers, operating systems, CPU cores, RAM, virtualization technology.
   - **Storage:** Total storage capacity (TB), storage type (SAN, NAS, DAS), backup/archive requirements.
   - **Databases:** Number of database servers, database types (SQL Server, Oracle), licensing details.
   - **Networking:** Estimated network bandwidth consumption.
2. **Adjust Assumptions:** The calculator comes with default assumptions for various operational costs *(e.g., electricity cost per kWh, IT labor rates, data center space costs, network maintenance).* You can adjust these to reflect your organization's specific real-world costs for a more accurate comparison.
3. **View Report and Compare:** Based on your input, the TCO Calculator generates a detailed report that provides a side-by-side comparison of:
   - **On-premises costs:** Including hardware, software licenses, electricity, cooling, networking, storage, IT labor, and data center maintenance.

- **Azure equivalent costs:** Estimated costs for running those same workloads on Azure (converting your on-premises servers into equivalent Azure VMs, storage, databases, etc., and factoring in potential savings from Reserved Instances, Azure Hybrid Benefit, etc.).
- **Potential cost savings:** Highlights the financial benefits of migrating to Azure over the chosen period.
- **Visualizations:** Often include charts and graphs to make the comparison easy to understand.

4. **Save/Export:** You can typically download the report for internal review and presentation.

"Difference between Azure Pricing Calculator & Azure TCO Calculator"

Aspect	Azure Pricing Calculator	Azure TCO Calculator
Purpose	Estimates the **cost of Azure services** based on configuration	Estimates **cost savings** when migrating from on-prem to Azure
Target Audience	Current or prospective **Azure users**	Organizations planning a **cloud migration**
Use Case	Budgeting for Azure workloads or services	Comparing on-premises costs vs Azure cloud over time
Input Focus	Azure-specific inputs: region, VM size, storage, bandwidth	On-prem costs: servers, energy, IT labor, software licenses
Output Type	Hourly/monthly cost estimates for Azure resources	Estimated total cost savings and ROI of Azure migration
Granularity	Service-level, detailed cost breakdown	High-level cost estimation and savings analysis
Customizations Available	Customize service tiers, redundancy, licenses, OS, regions	Customize hardware type, quantity, power, maintenance cost
Best For	Estimating how much your **Azure solution will cost**	Justifying **cloud migration** with financial metrics
Integration with Azure Portal	Yes (can save/export estimates to Portal)	No direct integration
Export Options	Excel, PDF, shareable link	Excel, PDF reports



Tags in Azure

- **Tags in Azure** are metadata elements that you apply to your Azure resources, resource groups, and subscriptions.
- They are essentially **key-value pairs** that help you organize your resources based on criteria relevant to your organization.
- Think of them as labels you stick on your resources to categorize them.

For example, you could have a tag with the key "Environment" and the value "Production" *(Environment:Production),* or a tag with the key "Department" and the value "Finance" *(Department:Finance).*

Where We Use Tags in Azure ***[Common and in-general use cases]***

Cost Management and Allocation:

- **Purpose:** This is one of the most critical uses. Tags allow you to categorize and group your costs by specific criteria.
- **How it's used:**
  - In Azure Cost Management + Billing, you can filter and group your spending reports by tags. This lets you see, for example, the total cost for all resources belonging

to the "Marketing" department *(Department:Marketing)* or all resources in your "Production" environment *(Environment:Production).*

- It enables chargeback (billing departments for their actual cloud usage) and showback (showing departments their cloud consumption without direct billing), fostering cost accountability.
- You can set up budgets in Azure Cost Management that are scoped to specific tags, allowing you to monitor spending for particular projects or teams.

Resource Organization and Management:

- **Purpose:** To logically organize and easily find resources in large and complex Azure environments.
- **How it's used:**
  - Filtering and Searching: You can quickly find all resources associated with a specific project, owner, or application by filtering your Azure resources in the portal, PowerShell, or CLI based on their tags. For instance,

Get-AzResource -TagName "Project" -TagValue "Alpha"

  - **Grouping:** While resource groups provide structural grouping, tags offer a more flexible, cross-cutting way to group resources that might span different resource groups or subscriptions.
  - **Auditing and Inventory:** Tags can help track resource ownership, purpose, and lifecycle status, aiding in asset management.

Operations Management:

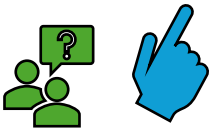
- **Purpose:** To provide operational context for resources, simplifying automation, monitoring, and incident response.
- **How it's used**:
  - **Automation:** Azure Automation runbooks or Azure Functions can be configured to perform actions on resources based on their tags. For example, a script could automatically shut down all VMs tagged with ShutdownTime:1900 (7 PM) or perform backups only on resources tagged *BackupPolicy:Daily.*
  - **Monitoring and Alerting:** You can configure Azure Monitor alerts to trigger for resources with specific tags, allowing you to focus on critical components (e.g., monitor only *Criticality:High resources).*

- **Patching and Maintenance:** Identify groups of resources that require specific patching schedules or maintenance windows based on tags like *PatchGroup:A or MaintenanceWindow:Weekend.*

Security and Compliance:

- **Purpose:** To enforce security policies and track compliance requirements across your Azure footprint.
- **How it's used:**
  - **Azure Policy:** You can create Azure Policies that enforce tagging standards. For example:
    - "Require a 'Department' tag on all new resources."
    - "Require a 'DataClassification' tag *(DataClassification:Confidential, DataClassification:Public)* on storage accounts to enforce specific security controls (e.g., encryption settings) via other policies.
    - "Prevent deployment of resources if they don't have mandatory tags."
    - "Inherit tags from the parent resource group or subscription."
  - **Security Audits:** Use tags to identify resources that fall under specific regulatory compliance requirements (e.g., *Compliance:GDPR, Compliance:HIPAA).*
  - **Access Control:** While RBAC is the primary for access, tags can indirectly inform access. For instance, a policy might dictate that only users from the "HR" department can access resources tagged *Department:HR and DataClassification:Sensitive*

Doubt- Where we exactly use TAGS, and what resources can be tagged, how can we implement it, via (GUI or CLI)? Does Azure offer built-in tags, or can the user make its own favorite tags?



tags can be applied to a wide range of entities, providing a flexible way to organize and manage your cloud environment:

1. **Individual Azure Resources (e.g., VMs, Storage Accounts, Web Apps, Databases, Networks):**
    - **Yes, on VMs!** You can apply tags directly to individual Virtual Machines (VMs) to denote their purpose *(Role:Web_Server), environment (Environment:Prod), owner (Owner:JohnDoe), or patch group (PatchGroup:Monthly).*
    - You also apply tags to other services like:
        - **Storage Accounts:** To identify the data type *(DataClassification:Confidential),* retention policy *(Retention:7Years),* or department.
        - **Azure SQL Databases:** To indicate the application, it supports *(Application:CRM),* performance tier *(Tier:Premium),* or compliance needs *(Compliance:HIPAA).*
        - **Azure Web Apps/App Services:** For the application name *(AppName:CustomerPortal),* development team *(Team:DevOps),* or deployment stage *(Stage:UAT*).
        - **Network resources:** Like Virtual Networks, Network Security Groups (NSGs), Public IP addresses, etc., to associate them with specific projects or environments.
        - Azure Functions, Logic Apps, Key Vaults, Kubernetes Service (AKS) clusters, and many more services.
2. **Resource Groups:**
    - **Yes, you can tag Resource Groups.** Applying tags at the resource group level helps categorize groups of resources that share a common lifecycle, purpose, or owner.
    - While tags applied to a resource group do **not automatically inherit** to the resources *within* that resource group by default, you can use **Azure Policy** to enforce tag inheritance or ensure resources within a group receive specific tags. This is a common and highly recommended practice.
3. **Subscriptions:**
    - **Yes, you can tag entire Azure Subscriptions.** This is useful for high-level organizational tagging, especially in large enterprises with multiple subscriptions

(e.g., one subscription per business unit, or per major environment like *"Production" vs. "Development"*).

- Tags on subscriptions can help with overall cost allocation at a broader level.

Where and How We Implement Tags:

We can use both GUI and CLI for creating, deleting and viewing tags.

We can use both built-in Azure tags and can define Custom tags also.

Governance and Compliance in Azure

- **Governance** refers to the **framework of rules, policies, and controls** that help organizations **manage, secure, and control** how Azure resources are deployed, used, and maintained.
- It ensures that **teams operate within organizational boundaries**, especially in large enterprises where multiple teams are provisioning and managing resources.

Key Azure Governance tools:

Feature	Purpose
Management Groups	Organize subscriptions into a hierarchy for centralized governance
Azure Policy	Enforce rules (e.g., only deploy in specific regions, or use specific VM sizes)
Role-Based Access Control (RBAC)	Control *who* can do *what* with *which* resource
Tags	Label resources for tracking, billing, or automation
Blueprints	Package policies, role assignments, and ARM templates into repeatable environments

- **Compliance** refers to ensuring your **Azure environment aligns with regulatory standards** like **ISO, GDPR, HIPAA, SOC, PCI-DSS**, etc. Azure helps you stay compliant with these by providing **certified services** and **auditable controls**.

Azure Compliance Offerings:

Component	Description
Microsoft Compliance Manager	Provides a compliance score and actionable recommendations
Azure Trust Center	Central hub for understanding compliance offerings and certifications
Service Trust Portal	Access audit reports, compliance guides, and certifications
Compliance Blueprints	Predefined policies and templates for regulated workloads (e.g., HIPAA, FedRAMP)

Doubt- What are these terms? ->> ISO, GDPR, HIPAA, SOC, PCI-DSS

What Does India Generally Follow? ?

These are the Global Compliance standards.

Acronym	Full Form	What It Covers	Applies To
ISO 27001	International Organization for Standardization (InfoSec)	International standard for **information security management systems (ISMS)**	All industries worldwide
GDPR	General Data Protection Regulation	Strict **data privacy** law in the **European Union** — governs how personal data is collected and handled	Businesses handling EU citizen data
HIPAA	Health Insurance Portability and Accountability Act	U.S. law that protects **healthcare information (PHI)**	Healthcare providers, insurers (U.S. specific)
SOC	System and Organization Controls (SOC 1, 2, 3)	**Auditing standards** for service organizations (focus on security, availability, confidentiality)	Tech & cloud service providers

PCI-DSS	Payment Card Industry Data Security Standard	Security standard for handling **credit/debit card transactions**	Any business processing payment cards

In India-

Standard	Followed in India?	Remarks
ISO 27001	Yes	Widely adopted by Indian IT companies, banks, data centers
GDPR	Partially	Not legally applicable *within* India, but **followed** by Indian firms handling **EU data**
HIPAA	No (U.S. only)	Not followed in India; Indian healthcare firms serving U.S. may comply
SOC 2 / SOC 3	Yes (voluntary)	Many Indian IT firms pursue this for global clients
PCI-DSS	Yes	Mandatory for Indian companies that **process credit/debit cards**

Data Governance in Azure

Data Governance in Azure refers to the strategic framework and tools used to ensure that your data is well-managed, secure, compliant, accurate, and used responsibly across your organization.

It helps you answer:

- *Where is my data?*
- *Who has access to it?*
- *Is it protected and compliant?*
- *Is the data reliable and trustworthy?*

Key Components of Azure Data Governance

Component	Purpose
Microsoft Purview	Unified data governance service — **discovery**, **cataloging**, **classification**, and **lineage** of data
Azure Information Protection (AIP)	Classifies, labels, and protects **sensitive information**
Azure Role-Based Access Control (RBAC)	Controls **who can access what data**
Azure Policy & Tags	Ensures **data resources follow governance rules**, such as geo-location or tagging
Azure Monitor / Sentinel	Enables **auditing**, monitoring, and **security oversight** on data usage
Data Loss Prevention (DLP)	Prevents accidental data leaks or sharing outside the organization

Real Life Examples-

a) A bank wants to ensure that customer financial data is only stored in the India region. Using Azure Policy, it blocks storage account creation outside India.

b) An enterprise wants to catalog and classify all its internal data. It uses Microsoft Purview to build a searchable, auto-classified data catalog.

c) A healthcare provider ensures patient records are encrypted and labeled using Azure Information Protection to meet HIPAA compliance.

Azure Purview -

- It is a unified data governance and compliance solution offered by Microsoft.
- It helps organizations **discover, classify, manage, protect, and govern data** across **Microsoft services, Azure**, and even **external (on-prem or multi-cloud) environments**.
- It is an **Azure-based tool**, but it integrates **across the Microsoft ecosystem** and beyond.

****Now** a days, Purview comes Pre-installed/Integrated with all Microsoft products along with Azure. **

Question-

Does Microsoft Purview Work Automatically or Manually?

Both — It's a Hybrid Approach

What's Automated in Microsoft Purview?

Feature	How It Works Automatically
Data Scanning & Discovery	Automatically scans data sources like Azure SQL, Blob, Power BI, M365, and builds metadata
Data Classification	Uses **built-in sensitive info types** (like PII, credit cards, Aadhaar, PAN, etc.) to auto-classify data
Data Lineage	Automatically tracks how data flows across services (e.g., SQL → Synapse → Power BI)
Compliance Assessments	Continuously evaluates compliance posture against standards like **ISO, GDPR, HIPAA**
Alerts & DLP	Automatically applies Data Loss Prevention (DLP) policies on M365 (Outlook, Teams, etc.)

Resource Lock in Azure

- Resource Lock in Azure is a protective feature that helps prevent accidental deletion or modification of your critical resources.
- Even users with high-level permissions (like Owner or Contributor) cannot delete or change a resource if it has a lock applied — unless they remove the lock first.

Types of Resource Locks; -

Lock Type	Effect
ReadOnly	Resources can be **read** but **not modified or deleted**.
CanNotDelete	Resources can be **read and modified** but **not deleted**.

***** You can only have one lock per resource at a time, but it can be inherited from the resource group or subscription level. *****

Where Can You Apply Resource Locks?

- Subscription
- Resource Group
- Individual Resource (e.g., VM, Storage Account, Public IP)

** Locks at higher levels are inherited by all child resources. **

** **Locks can be overridden** — but only by **removing them first**, which requires appropriate permissions. **

Azure ARC

It is a hybrid and multi-cloud management platform from Microsoft that allows you to extend Azure's management, governance, and security to:

- On-premises servers and VMs
- Other clouds (AWS, GCP)
- Kubernetes clusters
- Databases (like SQL Server, PostgreSQL)

It brings non-Azure infrastructure into Azure control, as if it were native Azure resources.

Exact meaning! -> Imagine **This:**

You have:

- Some computers and servers **in your office (on-premises).**
- Some running in **Amazon Cloud (AWS)**
- Some running in **Google Cloud (GCP)**
- And of course, some running in **Azure (Microsoft Cloud)**

Now, managing and monitoring all these systems separately is a **big headache**, right? (Obviously right).

Azure Arc is like a universal remote control 

It lets you **manage ALL your computers and systems — no matter where they are — from ONE place: Azure Portal**

Even if your servers are:

- In your company office

- In another cloud like AWS
- Running on a laptop in a branch office

You can still:

- See them in the **Azure portal**
- Apply **security rules**
- Monitor their **health**
- Control **access permissions**
- Keep everything **organized and compliant**

But 1 Question, **HOW?** How does this actually works?

- Azure Arc **doesn't take control** of AWS or GCP *like owning them* — instead, it **connects to resources you already own** using **agents, APIs, and registration tokens** — and brings their **metadata, status, and control hooks into Azure.**

Integration is Done via an Agent or Kubernetes Extension

For Servers (on-prem, AWS, GCP VMs):

1. You install the **Azure Connected Machine Agent** on the server (Windows or Linux).

   This agent:

- **Authenticates** with Azure (via a token during onboarding)
- **Registers the machine** into Azure Arc
- Periodically **syncs metadata**, such as OS version, location, policies

2. After that, the server **appears in Azure Portal** under "Arc-enabled servers"

It doesn't change how AWS or GCP work — it simply brings visibility and management from Azure.

For Kubernetes Clusters (anywhere):

1. You use an **Azure Arc K8s extension** (via CLI or portal)

   It:

- Deploys a **set of agents (pods)** in your K8s cluster
- Connects securely to Azure
- Reports cluster health, workloads, compliance, etc.

2. Now you can use Azure Monitor, Defender, and Policies **on that K8s cluster**

Security & Permissions

- Azure Arc doesn't break security of AWS/GCP — you're still in charge.
- You **control what is connected**, and **Azure uses secure tokens & certificates**.
- AWS/GCP just see it as "your VM" running a Microsoft agent — **they don't block it** because it's your environment.

Types of Azure Arc Connectors

Resource Type	Connector / Agent Used	Where It's Installed
Servers (Windows/Linux)	Azure Connected Machine Agent	Installed directly on the server or VM
Kubernetes Clusters	Azure Arc K8s Connect Agent (runs as pods)	Installed inside the Kubernetes cluster
Databases (SQL, Postgres)	Containerized Azure Arc data services	Runs on your infrastructure (on-prem/cloud)

The Azure Arc **connector is just a software agent**, like an app running in the background, that **helps Azure talk to and manage** your external or on-prem resources.

Azure Resource Manager (ARM)

- **Azure Resource Manager (ARM)** is the **core management layer** of Azure.
  It's **not a tool or button** you click — it's the **engine that runs behind the scenes** whenever you deploy, manage, or delete anything in Azure.
- ARM is the **brain of Azure** that understands and organizes all your resources — whether you're using the **Azure Portal**, **Azure CLI**, **PowerShell**, **ARM templates**, or **SDKs/APIs**.
- It's not a physical component like a CPU. Instead, it's a **global, distributed software platform** that **runs in Microsoft's Azure backend infrastructure**, across its data centers.
- You don't install ARM. It's **always running in Azure**, and **you interact with it via Azure Portal, CLI, PowerShell, or APIs.**

What exactly does ARM Do?

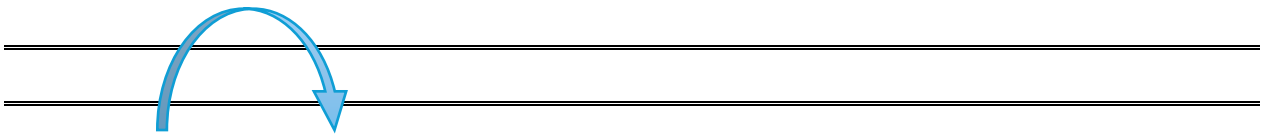
Function	Description
Resource Deployment	Handles creation, modification, and deletion of resources
Resource Grouping	Allows logical grouping of related resources (via Resource Groups)
Tagging & Organization	Supports tagging for cost, ownership, environment, etc.
Access Control (RBAC)	Enforces security roles and permissions
Policy Enforcement	Ensures resources comply with your organization's rules
Auditing & Logging	Connects to Azure Monitor, Logs, and Security Center for insights

Azure Monitoring Tools

Monitoring Tool	What It Monitors / Tracks
Azure Monitor	The **core monitoring platform** in Azure. Collects telemetry (metrics, logs, traces) from nearly **all Azure resources**, custom apps, and VMs.
Log Analytics	Part of Azure Monitor. Helps you **query and analyze log data** (from VMs, Azure resources, custom apps) using **Kusto Query Language (KQL)**.
Azure Application Insights	Focused on **application performance monitoring**. Tracks **requests, dependencies, failures, response times**, and user behavior for apps (web, mobile, APIs).
Azure Activity Log	Logs **management and control-plane events**, like resource creation, deletion, RBAC changes, and policy updates. Useful for **auditing**.

Azure Diagnostic Settings	Captures **resource-specific logs and metrics** (e.g., VM logs, storage access logs) and sends them to **Log Analytics, Event Hubs, or Storage Account**.
Network Watcher	Monitors **network-level metrics**, such as packet flow, connection issues, topology, and latency. Includes tools like **IP flow verify, connection monitor, NSG flow logs**.
Azure Metrics Explorer	Visual tool to plot **real-time performance metrics** for Azure services like CPU usage, memory, disk I/O, etc.
Azure Advisor	Monitors and **analyzes best practices** — gives recommendations for **cost optimization, high availability, performance, and security**.
Azure Service Health	Monitors the **health of Azure services** in your regions. Alerts you when **planned maintenance or outages** occur.
Azure Sentinel	A cloud-native **SIEM (Security Information and Event Management)** platform. Monitors **security-related events and threats** across Azure and external systems.
Azure Resource Health	Tracks the **health of individual Azure resources** (e.g., if a VM is down due to platform issues). Useful for troubleshooting.

*** All Tools are available in Azure GUI Portal ****



References-

https://learn.microsoft.com/en-us/training/courses/az-900t00

https://www.geeksforgeeks.org/devops/microsoft-azure/

https://azure.microsoft.com/en-in/products/databox/data/

https://azure.microsoft.com/en-us/blog/microsoft-azures-defense-in-depth-approach-to-cloud-vulnerabilities/

https://talibilat.medium.com/day-35-azure-az-900-defence-in-depth-0a351aad7bc0

https://learn.microsoft.com/en-us/entra/identity/conditional-access/concept-conditional-access-policies

https://learn.microsoft.com/en-us/entra/identity/conditional-access/policy-block-example

https://www.rezonate.io/blog/microsoft-entra-id-the-complete-guide-to-conditional-access-policies/

https://www.microsoft.com/en-us/security/business/zero-trust#:~:text=Zero%20Trust%20is%20the%20cornerstone,across%20eight%20key%20defense%20areas

https://www.nist.gov/blogs/taking-measure/zero-trust-cybersecurity-never-trust-always-verify#:~:text=Comments,the%20access%20to%20the%20system

https://www.twingate.com/blog/the-3-core-principles-of-zero-trust

https://learn.microsoft.com/en-us/security/zero-trust/copilots/zero-trust-microsoft-365-copilot

https://www.geeksforgeeks.org/devops/azure-virtual-machine-pricing/

https://www.geeksforgeeks.org/tag/azure/

https://azure.microsoft.com/en-us/pricing/calculator/#:~:text=Calculate%20your%20estimated%20hourly%20or,Get%20started%20with%20Azure

https://azure.microsoft.com/en-us/pricing/details/defender-easm/#product-pricing

https://azure.microsoft.com/en-us/pricing/tco/calculator/

https://www.elliottdavis.com/insights/strengthening-cybersecurity-part-3-peeling-back-the-layers-of-protection#:~:text=When%20properly%20implemented%2C%20these%20layers,breach%20or%20minimize%20its%20impact