

CLOUD-AZURE

DEVOPS:

DevOps is a **combination of**

- **tools,**
- **practices, and**
- **cultural philosophy**

that helps in delivering high-quality software **faster** and **more reliably** through **automation** and **collaboration**.

Gate Management in DevOps:

- Acts as a **control checkpoint**
- Used to decide when to allow a process or **deployment to move forward**
- Helps in **improving quality** by catching issues early and reducing technical debt

DevOps is a combination of 3 things:

1 Tools

Used for automation, integration, and tracking:

- **Ansible** – Configuration management
 - **Jenkins** – CI/CD automation
 - **Jira** – Project and issue tracking
-

2 Practices (8)

Key DevOps lifecycle stages:

Plan → Code → Build → Test → Release → Deploy → Operate → Monitor

Other Important Practices:



- **CI/CD** – Continuous Integration & Delivery
- **IaC (Infrastructure as Code)** – Automate infrastructure setup (e.g., using Terraform, ARM templates)
- **Certificate Management** – Secure communication
- **Security Checks** – Integration of tools to catch vulnerabilities early


3 Cultural Philosophy

- **Collaboration** between Development, Operations, QA, and Security
- Encourages **shared responsibility**, **faster feedback**, and **continuous improvement**

Why "Shift Left" Is Important

Moving things to the **left side of the pipeline (earlier)** helps reduce cost and risk.

 Where Bug is Found	 Cost to Fix
After release (prod)	Very High
In testing stage	Moderate
During coding	Low

 **Goal:** Catch defects early → Less rework → Lower cost → Faster delivery

DevSecOps:

- DevSecOps = Development + Security + Operations
- It integrates security into every step of the DevOps pipeline
- **SAST** – Static Application Security Testing
 - Scans source code for vulnerabilities
 - **Should be done during the integration phase** (not after deployment)
 - Helps catch security issues early in the build process

DNS & Cloud-Based Product :

1. DNS (Domain Name System) – Cost:

- DNS is typically a **monthly-paid service**.
- You pay as long as you use the domain, usually **low cost (e.g., ₹4/month)**.
- Essential for making your cloud product accessible over the internet (e.g., www.myproduct.com).

Q. Product in Cloud – Why It's a Success?

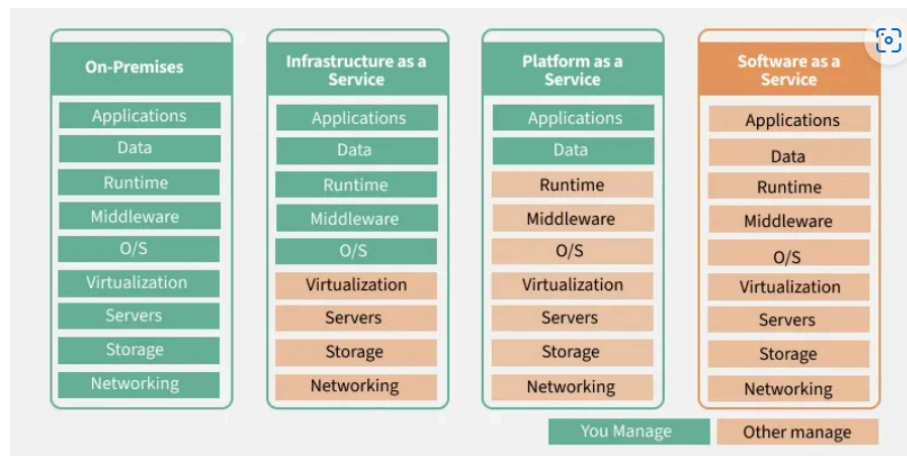
Ans:

- Cloud allows you to **host and scale** your product globally.
- No need to maintain **physical servers**.
- Helps in faster go-to-market and **availability**.

💰 Upfront Cost Comparison

Type	Upfront Cost?	Notes
Traditional Hosting	✗ No upfront cost	But may have limitations & hidden costs
Cloud	✓ Yes, minimal upfront	Small initial cost, but scalable and flexible







CLOUD SERVICE MODELS:



IaaS, PaaS, SaaS

1. **IaaS:** Cloud model where you **rent complete infrastructure** (like virtual machines, storage, networks) from a provider (e.g., Azure), and you manage everything on top of it.
 - eg: Working from home with **no laptop and no salary** = You have **no infrastructure** to work.
 - If **UST provides you a laptop, network, and setup**, now you have complete infra = That's IaaS.

Advantages of IaaS:

-  **Complete control** of resources
-  **Access to any type of cloud resource** (VM, Storage, Networking, etc.)
-  **Pay-as-you-go** model – pay only for what you use
-  **Scalability** – Scale from 1 VM to 100+ (ask for quota increase if needed)
-  **Elastic IP** – Assign fixed public IP to a resource (quota-restricted)
-  **Self-Service Provisioning** – Create resources yourself without contacting support

Tools for Self-Service Provisioning:

Tool/Method	Description
Azure ARM Template	Earlier method for provisioning (JSON-based)
Azure Bicep	Latest from Microsoft; faster, secure, standard-based
Terraform	3rd-party tool by HashiCorp, works with multiple clouds
Azure Portal	UI-based, easy and self-explanatory





Self service provisioning:

- create almost any kind of services(instance,cluster,acr,roles,acs,azure files,storages,devops pipeline,network,subnet)
- portal - self explanatory(dont need trainer)
- dis: time,consistency






Q. why so many options like Azure ARM template,Azure Bicep,Terraform?


Ans:

1. ARM Templates (Azure Resource Manager)






-  Written in JSON
-  Oldest Azure-native tool
-  Works well, but harder to read and maintain
-  Forms the base for newer tools like Bicep

2. Bicep





-  New and improved version of ARM Templates
-  More secure, easier to write, and modular
-  Designed and maintained by Microsoft
-  Supports all new Azure features
-  If you're fully using Azure – Bicep is the best choice

 Think of Bicep as a simpler, modern language that internally converts to ARM.

3. Terraform

-  Created by HashiCorp (a third-party company)
- **Hashicorp** - Other company, not from azure and microsoft
- - terraform ,vagrant,vault
- deploy an app -> artifact is docker images -> deploy docker images
- Use terraform and automatically create **provision infrastructure**.
- U dont have to switch to diff system , be in azure platform
- Terraform can **integrate** with any cloud.
- not tightly bound to azure .
-  Not just for Azure – supports multi-cloud (AWS, GCP, Azure, etc.)
-  Best if you are working across different cloud providers
-  More flexible – can connect to other tools like Vault, Consul
-  Great for companies who don't want to be locked into one cloud

4. Azure Portal

-  Graphical interface (UI) – no coding
-  Easy for beginners or one-time tasks
-  Not great for repeatable or large-scale automation
-  May lack consistency and take more time

Tool	Best For	Pros	Cons
ARM	Legacy support	Native, structured	Hard to manage manually
Bicep	Azure-only, modern setups	Simple, secure, all new Azure features	Azure-specific
Terraform	Multi-cloud environments	Cloud-agnostic, powerful integrations	Third-party, not Azure-native
Portal	Quick, manual setups	Visual, user-friendly	Not scalable or reusable

Q. Virtualization?

- ★ Multiple VMs run on a single physical machine.
- ★ Each vm have its own OS,CPU,RAM,Disk,Network.
- ★ Each VM is **isolated** – one tenant cannot access another's data
- ★ Helps in **cost reduction** and efficient **resource utilization**

Q. Multi-Tenancy?

- ★ One physical server is shared by many customers, but securely isolated.
- ★ 1 vm can be used by multiple users
- ★ u should not be able to see data and any kind with other tenants,by the same tym u r sharing all the resources
- ★ cost reduces
- ★ Can lead to issues in industry-specific high-security needs
- ★ Some company needs higher security, Solution-> companies coming up together -> still fails.



Q. How does IaaS work?

- ★ IaaS (Infrastructure as a Service) provides virtual infrastructure (like VMs, storage, network) but you manage everything above it — like the OS, middleware, and applications.

What You Need to Maintain in IaaS:

Layer	Responsibility	Example/Explanation
 OS	✓ Your Responsibility	You install, update, patch Linux/Windows
 Middleware	✓ Your Responsibility	e.g., Apache Tomcat – you install and upgrade
 Runtime	✓ Your Responsibility	e.g., Java, .NET – setup and manage
 Application & Data	✓ Your Responsibility	You write code, deploy app, and handle data


- ★ **Data** - data inside services (like VMs) is not automatically persisted.
- ★ Services are mostly **stateless** (won't persist data). Why?
- ★ To **enable scalability** — easily add/remove instances without losing data.
- ★ So you must **externalize the data**
- ★ external database store in Redis, Memcached (no SQL) - key-value pair DB
- ★ if you're storing by yourself -> you have to take care of that

External Storage	Purpose
 Azure Blob, S3, Disks	Persistent storage
 Redis, Memcached	Fast, in-memory storage (key-value)
 External Databases	MySQL, PostgreSQL, Cosmos DB, etc.

Q. Why Services Don't Store Data by Themselves?

- To **scale easily** and avoid **data loss**.
- Example: During an Amazon flash sale, 10,000+ people may add laptops to their cart, and backend servers need to scale instantly. If services store data inside themselves, new servers start fresh and old data is lost, but if they're stateless and use external databases, all instances share the same data, enabling seamless scaling and consistent user experience.

Q. Benefits of IaaS?

-  **Cost Reduction**
No need to invest in physical infrastructure. Pay-as-you-go model helps cut upfront costs.

★ ⚡ **Faster Deployment (Automated)**

Automation tools enable faster provisioning, testing, and releasing — reduces time-to-market.

★ 🔄 **High Availability & Reliability**

Use of AMIs (Amazon Machine Images) ensures reliable replication and fast recovery in case of failures.

★ 🌐 **Global Reach**

Easily deploy applications in any region around the world to serve users with low latency.

★ 🗝️ **Data Security and Compliance**

Data can be encrypted and moved to specific regions to comply with local regulations and standards.

Q. Challenges of IaaS?

- **Security & Vulnerabilities**

Cloud platforms may face vulnerabilities — providers send alerts, but you must apply patches yourself (especially in IaaS).

- **Compliance Requirements**

Industries like banking and healthcare (e.g., HIPAA in the US) require strict data confidentiality.

- In SaaS, data is with the provider.
- In IaaS, you are responsible for compliance and data protection.

- **Integration with Existing Systems**

Old systems (legacy apps) may not work smoothly with new cloud or automation tools — so extra effort or changes may be needed to make them work together.

- **Dependency on Cloud Provider**

Service interruptions, pricing changes, or policy shifts by the provider can impact your business.

Q. categories of cloud?

★ **Public** - Shared among multiple users

★ **Private** - Used by single organization

★ **Hybrid** - Mix of public and private clouds

Q. How u can trust the VM provided is secure and no one have ownership?

- ➔ **Limited Access:** Less than 2% of employees, even from outside the company, have access to the data centers. Only authorized people can access the VMs.
- ➔ **Data Protection:** When data is stored, it's split and stored securely. No one outside the organization has the private encryption keys (PEM files).

- **Secure despite Attacks:** While attacks can happen (like software vulnerabilities), the **cloud ensures data security**. Most attacks are common software-related ones, not specific to the cloud.

2. PaaS (Platform as a Service): web based

- **Scalability:**
PaaS allows you to scale your applications without worrying about the underlying infrastructure.
Example: Azure App Service — It automatically knows where to deploy your code based on the language you select (e.g., Java, Python).
- **Deployment & Control:**
With PaaS, you don't need to manage the server or hardware — just deploy your application (artifact).
Scaling is easy:
 - If your app receives 1000 requests tomorrow, you pay for the resources used.
 - The next day, if the demand reduces, you don't pay as much.
- **Ownership and Control:**
You have full control over the application, but the cloud provider manages the infrastructure (servers, network, etc.).
- **Data & Application Responsibility:**
While the platform manages scaling, you are still responsible for your app and data.
- **Lift & Shift:**
You can move your existing app (without major changes) to PaaS for better scalability and management.
- **AKS (Azure Kubernetes Service):**
AKS is a PaaS platform for managing and scaling containerized applications using Kubernetes.
- **Real-World Example:**
MakeMyTrip — A PaaS product that allows businesses to build and customize their services while the platform manages the infrastructure.

Q. Key characteristics of PaaS?

- **Integrated DB and Backend Services:** All parts of the system (e.g., database, APIs, and services) work together seamlessly.

Q. Benefits of PaaS?

- **Non-Functional Requirements:**

- ◆ Solutions span across all modules, ensuring uniform performance, security, and scalability.
- ◆ Means: "solutions span across all modules," it means that the design of the system ensures these concerns (performance, security, scalability) are applied equally to all parts of the system, not just isolated to one area. This leads to a more reliable and effective system overall.

Functional requirements:

- **Authentication:** Users can log in securely across all parts of the system.
- **Example (Amazon):** When buying a product, backend services handle actions like verifying login, updating the database, and processing payment.

In service-based companies, after winning a project through an RFP and proposal, you gather both **functional** and **non-functional requirements**. While customers mostly focus on **features** (functional requirements), they also expect things like **performance**, **reliability**, and **quality** (non-functional requirements). Customers rarely mention these non-functional needs, but they expect them to be met.

RFP - document to submit proposals for a specific project

A **proposal** is a detailed document submitted by a vendor in response to an RFP.

Benefits of PaaS:

1. **Accelerated Dev Cycles:**
PaaS provides pre-configured tools, speeding up development and deployment.
 2. **Reduced Complexity:**
No need to manage infrastructure, letting developers focus on the application.
 3. **Lower Cost:**
You pay only for what you use, reducing the need for expensive hardware.
 4. **Scalability and High Availability:**
Easily scale your app with built-in redundancy and uptime.
 5. **Support Diverse Technologies:**
PaaS supports various programming languages and tools, offering flexibility.
 6. **Improved Collaboration:**
Teams can work together in real-time with shared resources.
 7. **Faster Time to Market:**
Streamlined processes help deliver products to market more quickly.
-

Challenges of PaaS:

1. **Vendor Lock-In:**
Migrating from one PaaS to another can be difficult and costly.
 2. **Security:**
You have less control over security, relying on the provider's measures.
 3. **Integration with Existing Systems:**
It can be complex to connect PaaS with existing on-premise or other cloud systems.
 4. **Customization Limitations:**
Some PaaS solutions have restrictions on how much you can customize the environment.
 5. **Performance Issues:**
Performance may vary, and you lack control over the infrastructure.
 6. **Cost Management:**
Costs can increase unexpectedly if usage is not carefully monitored.
-

22/04/25

Strategy vs Tactics

- **Strategy** → Long-term plan (vision-level)
- **Tactics** → Short-term steps to solve specific problems (especially NFRs)
- NFR(Non Functional Requirements) -
 - ◆ Example: Incoming event → system processes → outgoing response
- large number of events(like req) - cause performance issues
- solving NFR issues have tactics
- tactics for solving performance and tactics for solving Scalability is diff
- **Tactic for Performance:** Use Redis caching, enable GZIP compression.
- **Tactic for Scalability:** Use Kubernetes HPA + Azure Load Balancer.

Scalability

- Ability of the system to handle increasing load gracefully
- **Load increases** → more users, requests, events
- **Resources can become constrained** → CPU, memory, DB connections
- Can cause **latency**, **timeouts**, or **crashes** if not handled
- **Solution** : horizontal scaling(more machines),**Auto-scaling**(based on CPU, memory, request count),distribute load equally by load balancer.

Q. why iaas is more vulnerable than paas and saas?

You **manage** the entire **OS, patches, security groups**, firewall rules, etc. A small misconfiguration (like leaving **SSH open** to the world) can expose your system. Attackers exploit **weak IAM policies**, open buckets, default credentials, etc.

Buildtime vs Runtime

- **Build time** : ok to take more time to build
- **Build Time** →
 - ◆ Time taken to **create Docker image**
 - ◆ Okay if it takes more time (once per deployment)
 - ◆ **Image Caching** helps here (layers reused)
 - ◆ image caching helps in build time not run time
- **Solution:**
- Use **Multi-stage Builds**
 - ◆ Only copy the required final binaries into the production image
 - ◆ Means - After building your app, just **take the output file** (like a `.jar`, `.pyc`, or built folder) and put **only that** into the final Docker image.
 - ◆ Don't include: Source code, Build tools, Temporary files
 - ◆ **Final binaries** - These are the files that are ready to **run**.
- Use **small base image**
 - ◆ (alpine -they have networking)
 - ◆ So that attack surfaces reduces
- **Run time**: Time when the **app is running in production**
 - ◆ Affects **performance** and **user experience**
 - ◆ Image caching doesn't help here

3. SAAS(software as a service)

- You access applications via a web browser
- You **don't need to install or manage** the app
- Eg: gmail, googledocs
- **Key features:**
 - ◆ **Web-based access** (via browser)
 - ◆ **Multi-tenant architecture** (one app, many users/companies)
 - ◆ **Accessible from anywhere** (just need internet)
- **Benefits of SaaS:**
 - ◆ **Lower upfront cost** - No need to buy or install software
 - ◆ **Reduced IT overhead** - No need to manage infrastructure or updates
 - ◆ **Scalability** - Easily increase/decrease usage as needed
 - ◆ **Flexibility (Pay-as-you-go)**- Pay only for what you use(**only for what u use**)
 - ◆ **Accessibility**- Use it anytime, anywhere on any device
 - ◆ **Usability**- learn how to use it easily and continue to remember it ,then the product is usable..


scaling in kubernetes:

- hpa - adding more pods
- vpa - increase resource of pods (memory,cpu)
- ca - adding more nodes in a cluster

Some imp terms

- **Bastion Host:** 1 machine in cloud ..from there u connect to other machines,ensure security from public cloud
 - A **simple VM** in the cloud used to **safely access other machines** (like app servers, DBs)
 - **Main purpose:** Secure admin access to VMs in **private networks**
 - Prevents exposing all machines to the public internet


Q. why use bastion host?

- No direct public access to critical servers
- Login only through Bastion Host = extra layer of security
- Think of it like:
 *"Gold in a locker — unless you know the full path and unlock the locker, you can't get it."*
- Notes: Port 443 → Used for HTTPS (secure web traffic)
 - ◆ Need to open specific ports when doing updates/upgrades (like SSH 22, HTTPS 443, etc.)
- **Vnet:**
 - VNet is like your own private network in the cloud
 - Used to **securely connect** your cloud resources like VMs, databases, and apps
 - Create subnet within the vnet
 - Subnet can be public and private
 - Inside subnet there will be vm
 - A **VNet spans the entire region, Ex: Central India region**
 - Includes **multiple Availability Zones (AZs)** like DataCenters in **Trivandrum (TVM) and Kochi**
- **Availability Zones:**
 - ◆ Each AZ = separate datacenter with independent power, cooling, backup, and infra
 - ◆ VM gets deployed in one of the AZs within a region
- **Subnet within the vnet:**
- You can create **subnets** for specific purposes (App, DB, Bastion, etc.)
- You can also **assign subnets to specific AZs** ➤ So **only allowed AZs can communicate** (controlled)

Q. VNet is Account Specific:

Each VNet is tied to your Azure account, You **cannot access someone else's VNet** unless it's **shared** with you

Q. Bastion Host with VNet:

- You can attach Bastion Host to a VNet
- You need:
 - ◆ Access to correct IP
 - ◆  .pem file or credentials
- If user tries to connect from unauthorized IP → Blocked

Q. VNet – Public and Private

Public VNet:

- Can be accessed from the internet
- Public IPs assigned to resources (VMs, Load Balancers, etc.)

Private VNet:

Cannot be accessed from the internet

Used for internal resources only

Bastion Host can be used within a private VNet to access internal resources securely

External users can't directly access the private VNet (must go through secure methods like Bastion Host)

Q. Region in Azure

- ❖ **its a geographical region where the azure has the datacenter**
- ❖ azure having datacenter in central india(bombay),south and west
- ❖ within region there will be availability zones- independent data centers(1 dc crashes other used..what should azure do?)
- ❖ **1 dc crashes other used..what should azure do?**
 - entire infrastructure(backup,power,network) system independent
 - u can have malware,terrorist attack
 - keeping dc soo away (1 is attacked then use other eg: earthquake also)
 - restart,os upgrade - group of machines they do it together

Q.vnet peering?

- Consider there are 2 vnets
- 1 from san francisco and 1 from india
- 2 vnets having diff range of ip where subnetting principle comes
- Default vnet ip range -same ,so diff range needed for both vnet to possible
- Soo if we need to connect the vnet ,s without using internet we can do vnet peering b/w the vnet
- After vnet peering the machines from the both vnet can talk using private ip

Q. bastion host helps to communicate using private ip within the vnet.how?

- Vm inside the vnet can communicate without the internet.
- Bastion Host sits inside your VNet.
- You connect to Bastion through the Azure portal.
- Bastion then connects to your VM **using its private IP, without needing internet** or a public IP.
- So, Bastion acts like a **secure door** to reach your VMs inside the private network.

Note:

You connect to the **Bastion Host** through **Azure Portal** using HTTPS (port 443).From the Bastion Host, it initiates **RDP/SSH connection** to the target VM using its private IP address.

Q. private cloud:

- protected
- u have better control for dc(people with right access)
- more security
- still vulnerable to various kinds of attacks
- install vpn

Q. scaleset - It is a service

- automatically scale no of vm
- create load balancer infront of scaleset
- If load increases,vm increases

Q. fault domain - distribute vm across multiple fault domains

- 5 vm created from 3 phy machines
- if 3 of the phy machines undergoes (restart,upgrade)-maintenance affected
- solution : fault domain

