Phase 1

⭐*Benefits of Using Cloud Services*

Capacity (Agility, Elasticity, Scalability)

- **Capacity** = How much your system can handle.
- **Agility** = How quickly you can adjust resources.
- **Elasticity** = Ability to **grow or shrink automatically** based on demand.
- **Scalability** = Ability to **handle more load** without breaking.

Example:
Imagine a website:

- On weekdays, 50 users visit → one small server is enough.
- On weekends, 5000 users visit → you add more servers automatically (elasticity).

⭐ *Availability*

- **Availability** = Your service stays **up and running**, even if something goes wrong.
- Think of it like a **backup plan** for failure.

Example:

- You have a website hosted in one data center. If that data center goes down, your site goes offline → low availability.
- If you have two data centers and traffic switches to the second when the first fails → high availability.

⭐ *Blast Radius*

- **Blast radius** = How much damage happens if something fails.
- Smaller blast radius → less impact when failure happens.

Example:

- One giant server crashes → your entire app goes down → huge blast radius.
- Many small servers → if one crashes, only part of your app is affected → small blast radius.

⭐ *Disaster Recovery*

- **Disaster recovery** = How quickly you can **restore your service after a big failure**.

Example:

- Your data center floods → you switch to another region and your website is back online.

⭐ *Vertical Scaling vs Horizontal Scaling*

Vertical Scaling (Scale Up)

- Add **more power to one server** (CPU, memory).
- **Problem:**
  - Often requires **downtime** to upgrade.
  - Not all apps can handle huge servers.
- **Example:**
  - Upgrade a small server from 2 CPUs → 16 CPUs → your server must restart.
  - If it crashes during upgrade → downtime.

Horizontal Scaling (Scale Out)

- Add **more servers** instead of making one bigger.
- **Better for cloud** → handles failure and traffic smoothly.
- **Example:**
  - Your website has 2 small servers → traffic increases → add 3 more servers automatically.
  - Traffic decreases → remove 1 server.
  - You **never go below 2 servers** to ensure availability.

Rule of thumb in cloud: Horizontal scaling is safer, more flexible, and keeps your service always available.

✅ Simple Analogy

- **Vertical Scaling** = One big pizza → hard to eat, and if it burns, you lose all.
- **Horizontal Scaling** = Many small pizzas → easy to share, if one burns, others are still fine.

⭐ *Capital Expenditure (CapEx)*

- **What it is:** Buying physical resources upfront, which you own and depreciate over time.
- **Example in the real world:** Buying servers, storage devices, or networking equipment for your company's on-premises data center.
- **Simple analogy:** Like buying a car—you pay upfront, and it's yours for years.

In cloud context (Phase 1 / Azure labs):

- On-premises equivalent: If you wanted to practice Azure security but instead bought your own physical servers, firewalls, and networking gear to test labs.
- You spend **a lot upfront**, but you **own the hardware**.

⭐ *Operational Expenditure (OpEx)*

- **What it is:** Paying only for what you use, usually subscription or consumption-based. No big upfront cost.
- **Example in the real world:** Using a streaming service like Netflix—you pay monthly only for what you watch.

- **Simple analogy:** Like renting a car—you pay only when you drive it.

In cloud context (Phase 1 / Azure labs):

- Using Azure free-tier or pay-as-you-go services for VMs, Storage, Key Vault, and AKS in your labs.
- You **pay based on usage**, scale up or down, and don't need to buy servers or networking devices upfront.
- Ideal for learning: You can practice security labs **without spending thousands on hardware**.

Key difference (super simple)

CapEx	OpEx
Buy and own (servers, devices)	Rent or pay-as-you-go (Azure services)
High upfront cost	Low/no upfront cost
Fixed capacity (limited by hardware)	Flexible capacity (scale up/down anytime)
Depreciates over time	Costs treated as ongoing expense

⭐ *What does "responsibility" mean in cloud computing?*

In cloud computing, **responsibility** means:

👉 *Which tasks you (the customer) must manage*
vs.
👉 *Which tasks the cloud provider (Azure, AWS, GCP) manages*

This is called the **shared responsibility model**.

Simple explanation with examples

1. Example: Using a Virtual Machine (VM)

Your responsibility:

- Install updates
- Secure passwords
- Install antivirus
- Configure firewall rules
- Manage who can log in

Cloud provider's responsibility:

- Physical servers
- Data center buildings
- Power, cooling, hardware

✔ If your VM gets hacked because you didn't update it → **your responsibility**
✔ If a physical server in Azure breaks → **Azure's responsibility**

---

2. Example: Using cloud storage

Your responsibility:

- Decide who can access the files

- Encrypt sensitive data if needed
- Set access policies correctly

Cloud provider's responsibility:

- Keep storage hardware running
- Ensure data is replicated/redundant
- Fix hardware failures

✔ If you accidentally share a storage container with "Public access = ON" → **your responsibility**
✔ If a disk in the data center physically breaks → **cloud provider's responsibility**

---

3. Example: Using cloud networking

Your responsibility:

- Create secure network rules
- Close unused ports
- Create strong access policies

Cloud provider's responsibility:

- Maintain routers, switches, cables in the data center
- Ensure global network connectivity

✔ If you leave port 22 open to the internet → **your responsibility**
✔ If an internet cable inside the Azure data center fails → **Azure handles it**

---

4. **Example: Using databases in the cloud**

If you use a fully managed database (like Azure SQL):

Your responsibility:

- Protect your data
- Control who can access the database
- Configure firewalls

Cloud provider's responsibility:

- Patch the database engine
- Maintain servers
- Handle automatic backups (depending on service)

✔ If someone steals your data because permissions were too open → **your responsibility**

Super simple summary

Responsibility in cloud computing = what YOU must secure, configure, and manage vs. what the provider handles.

Cloud providers handle the **physical stuff**.
You handle the **things you run inside the cloud**.

⭐ *Differences Between Cloud Services*

IaaS(Infrastructure-as-a-Service)

When you use **IaaS**, you are renting the *infrastructure* from Azure — like servers, storage, and networking — but **you are still responsible for what happens inside your virtual machine and inside your cloud environment**.

Think of it like renting an empty apartment:

- Azure = landlord (manages the building)
- You = tenant (manage everything inside the apartment)

Here are the responsibilities in **simple words + examples**:

✅ What YOU manage (your responsibility)

1. Operating System

You decide:

- Windows or Linux
- Update it
- Patch vulnerabilities

Example:
If your Linux VM has a security update available, *you must install it*.

---

2. Applications

You control:

- What applications you install
- How they run
- Their security settings

Example:
If you install NGINX on your VM, *you configure it, secure it, and update it*.

---

3. Security settings

You must set:

- Firewalls inside the VM
- Anti-malware
- Correct permissions

Example:
If someone can SSH into your VM because your password is weak, **that's your responsibility**, not Azure's.

---

4. Network rules

You configure:

- NSGs (Network Security Groups)
- Subnets
- Routing rules

Example:
If port 3389 (RDP) is accidentally left open to the world, **you fix it**.

---

5. Identity and Access

You handle:

- Who can access your VM

- Role assignments
- Permissions

Example:
If you give admin rights to someone who shouldn't have them, **that's on you**, not Azure.

---

6. Monitoring

You must set up:

- Logs
- Alerts
- Health checks

Example:
If you want alerts when someone logs into your VM at 3 AM, **you must configure that** through Monitor or Sentinel.

BUT EVEN THOUGH these are all my responsibility i am not alone cz Azure will help me or provide my tools which i can use to do all these works as well

---

❌ What Azure manages (NOT your responsibility)

Azure takes care of:

- Physical servers
- Data center buildings
- Power & cooling
- Internet connections in data centers

- Disk failures (hardware)
- Racks, cables, network devices

Example:
If a physical server in Azure breaks, **Azure replaces it**, not you.

⭐ *PaaS (Platform as a Service)*

👉 Simple meaning:

PaaS gives you a **ready-made platform** to build and run your applications **without managing servers, OS, or hardware**.

You only focus on your **code**.
The provider handles everything else like OS updates, patches, runtime, and scaling.

✔ Example (everyday example)

Imagine you want to make a cake:

- You don't grow wheat
- You don't make the oven
- You don't clean the kitchen

You just:
✔ bring your ingredients
✔ bake the cake

PaaS = a ready kitchen. You cook, they manage the kitchen.

✔ Real cloud examples:

- **Azure App Service** (you deploy your web app, platform manages servers)

- **Azure SQL Database** (you use the database without managing SQL Server)
- **Azure Kubernetes Service (AKS)**
- **Google App Engine**
- **AWS Elastic Beanstalk**

⭐ *SaaS (Software as a Service)*

👉 Simple meaning:

SaaS is **complete software** that you simply **use** through a browser or app.

You don’t:

- install it
- update it
- manage servers
- manage the application

You just **log in and use it**.

✔ Everyday example:

You use **Netflix**:

- you don’t install servers
- you don’t manage videos
- you don’t update anything

You just click and watch.

SaaS = ready-to-eat food. Just consume it.

✔ Real cloud examples:

- **Microsoft 365**
- **Google Workspace**
- **Salesforce**
- **Dropbox**
- **Zoom**

⭐ *Serverless Computing*

👉 Simple meaning:

Serverless means **you write small pieces of code**, and the cloud runs them **automatically only when needed**.

- No servers to manage
- No infrastructure planning
- No paying for idle time
- It scales instantly

You **only pay when your code runs** (per request).

✔ Everyday Example:

Imagine a **vending machine**:

- You don’t keep a cashier
- Machine only works when someone inserts money
- You only pay for usage

Serverless = vending machine for code. Runs only when triggered.

✔ Where Serverless is used:

Serverless is used for **event-driven** tasks:

Use Case	Example
Running code when something happens	When a file is uploaded → run a function
APIs	Quick backend API services
Automation	Clean logs every night
IoT	When a sensor reports data
Chatbots	Run when user sends messages
Data processing	Resize images, process logs

✔ Serverless real examples:

- **Azure Functions**
- **AWS Lambda**
- **Google Cloud Functions**

⭐ Super Simple Summary

Model	What It Means	You Manage	They Manage	Example

PaaS	A platform to build apps	Code	Servers, OS, runtime	Azure App Service
SaaS	Ready-made software	Nothing	Everything	Netflix, Microsoft 365
Serverless	Code that runs only when needed	Tiny code functions	All infrastructure	Azure Functions

● Differences overall

1. The Core Idea (Super Simple)

Think of it like **owning, renting, or just using something already built**:

Model	Like…	Meaning
IaaS	Renting an empty house	You manage most things, provider gives basic infrastructure.
PaaS	Renting a house with kitchen & furniture	You only manage your app/code.
SaaS	Booking a hotel	Everything is done for you. Just use it.
Serverless	Uber	Used **only when needed**. You don’t manage the vehicle.

2. Detailed but Simple Difference

🔵 IaaS (Infrastructure as a Service)

Cloud gives you **virtual machines, storage, networks**, etc.
You control: OS, runtime, apps.

Example:

- Azure Virtual Machines
- AWS EC2
- Google Compute Engine

Simple example:
You rent an empty house.
You bring furniture, cook, clean, maintain.

🟣 PaaS (Platform as a Service)

Cloud gives you **platform + runtime**, so you only write/deploy code.

Provider manages OS, servers, patches.

Example:

- Azure App Service
- Azure SQL Database
- AWS Elastic Beanstalk

Simple example:
You rent a house *with kitchen* + *furniture*.
You only cook and live.
No cleaning the building or fixing plumbing.

🟢 SaaS (Software as a Service)

You get a **ready-made application**.
Just log in and use it.

Example:

- Office 365
- Gmail
- Salesforce
- Netflix

Simple example:
Like staying in a hotel.
You do NOTHING—just use the room.

🟡 Serverless

You write **tiny code functions**, and the cloud runs them **only when triggered**.
No servers.
No scaling.
No idle cost.

Example:

- Azure Functions
- AWS Lambda
- Google Cloud Functions

Simple example:
Using **Uber**:
The car only comes when you request it.
You pay only for the ride.

---

3. What You Manage vs What Cloud Manages

The best visual way to learn this:

```
┌───────────────────────────┐
│ Business Requirements     │  <--- You always manage this
└───────────────────────────┘

IaaS

┌────────────────┬──────────────────────────────────┐
│ You Manage     │ OS, runtime, apps                │
│ Provider Manages│ VM, network, storage            │
└────────────────┴──────────────────────────────────┘

PaaS

┌────────────────┬──────────────────────────────────┐
│ You Manage     │ Code & application               │
│ Provider Manages│ Runtime, OS, servers            │
└────────────────┴──────────────────────────────────┘

SaaS

┌────────────────┬──────────────────────────────────┐
│ You Manage     │ Nothing                          │
│ Provider Manages│ Entire application              │
└────────────────┴──────────────────────────────────┘

Serverless

┌────────────────┬──────────────────────────────────┐
│ You Manage     │ Only small pieces of code        │
│ Provider Manages│ Everything else                 │
└────────────────┴──────────────────────────────────┘
```

---

4. Fastest Way to Remember Them

Model	What You Do	What It Feels Like	Example
IaaS	Manage VMs	Buying raw materials to build something	Azure VM
PaaS	Write code	Kitchen ready → You only cook	Azure App Service
SaaS	Use app	Eating food served to you	Gmail
Serverless	Write function	Taxi — used only when needed	Azure Functions

Yes — Serverless is considered a specialized part *of* PaaS, but more automated and more managed.

BUT…

Serverless is **not exactly the same as PaaS**.
It is **PaaS taken to the next level** → more automation, more scaling, less management.

⭐ Long Answer (Simple Explanation)

Think of Cloud Services like this:

```
SaaS  → fully managed apps
PaaS  → platform to run apps
Serverless → automatic, event-driven PaaS
IaaS → raw infrastructure (VMs)
```

So:

🔵 **PaaS = You write code, cloud runs it, but app is long-running**

Examples:

- Azure App Service
- Azure SQL Database

🟡 Serverless = You write tiny functions, cloud runs only when triggered

Examples:

- Azure Functions
- AWS Lambda

⭐ Why serverless is considered a part of PaaS

Because:

- You **don’t manage servers**
- You **deploy code**, not infrastructure
- Platform automatically handles runtime
- Cloud does scaling for you

This is exactly what PaaS does — **Serverless just automates it more**.

⭐ Difference in Simple Words

PaaS

Your app is always running.

Example:
You deploy a website to **Azure App Service** → it runs 24/7.

You pay even if nobody visits it.

Serverless

Your code is *not running* unless triggered.

Example:
Azure Function runs **only when you receive an event**.

You pay **only when it runs**.

⭐ Final Summary

Feature	PaaS	Serverless
Always running?	Yes	No
Pay when idle?	Yes	No
Event-driven?	Not always	Yes
Scaling	Automatic but limited	Fully automatic
Server management	None	None

✔ Serverless = Advanced PaaS

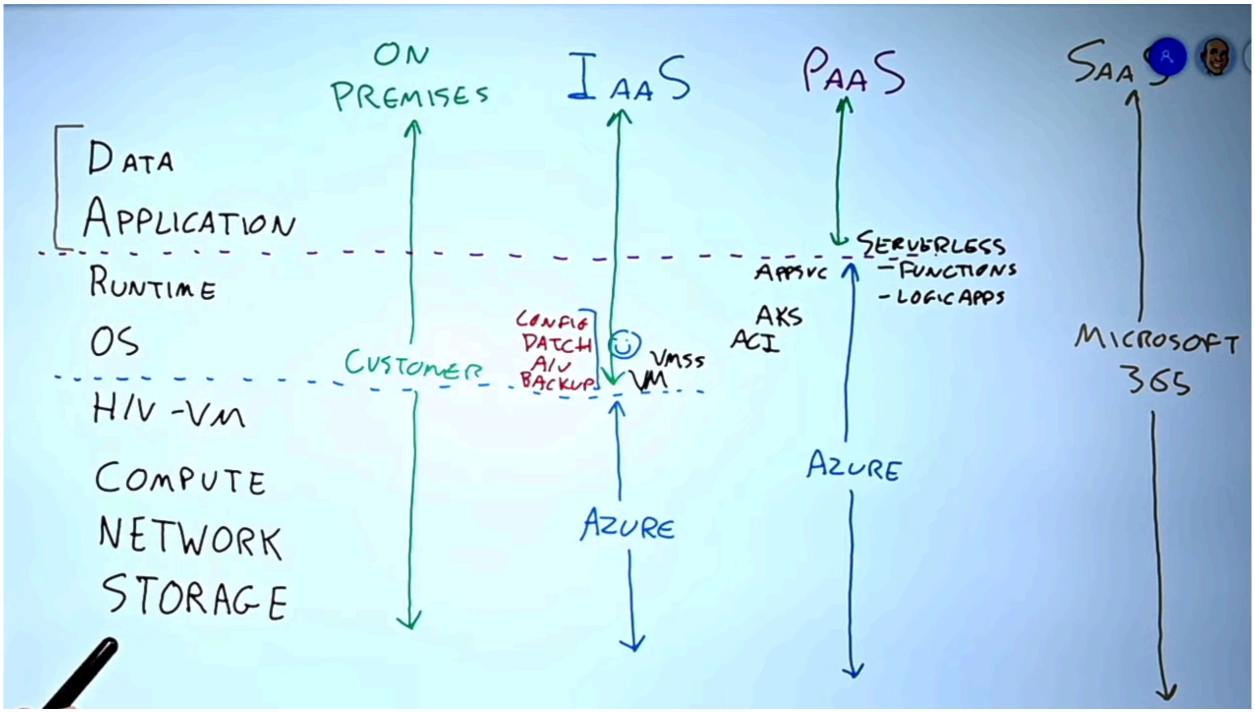
✔ More automated

✔ More scalable

✔ More cost-efficient

👉 **The key difference is whether the code runs 24/7 or only when triggered.**

So, in paas the code i wrote is running 24/7 but in serverless the code is not running or inactive but only when triggered the code will run.



⭐ *Cloud Properties*

1. Resource Pooling

Cloud providers like Azure, AWS, and GCP have **huge shared data centers** with:

- Thousands of servers
- Huge storage units
- Massive networks

These resources are **pooled together** and shared among customers **dynamically**.

✔ What it means:

You don’t get a fixed physical server.
You get a **virtual slice** of the big pool.

The cloud automatically allocates resources where needed.

✔ Simple Example:

Imagine a giant water tank.
Everyone gets water when they turn on the tap, but they don't know which exact pipe it came from.

Cloud is the same:
You request a VM, storage, or database → cloud picks available resources from the shared pool.

✔ Azure Example:

You create a VM in Azure.
Azure finds free CPU, memory, and network inside their giant infrastructure and gives it to you.

You don’t need to know **which exact machine** your VM is on.

---

2. Self-Service (with quotas & policies)

Cloud allows users to create resources **instantly on their own**.

You don’t need to call IT or wait 3 days for approval.

✔ What it means:

You open Azure Portal → Create VM → Done.
Just like ordering food from an app.

✔ Example:

You want a database → click “Create Database” → ready in minutes.

⭐ Self-Service has two controls:

A. Quotas (Limits)

To prevent accidental overspending.

- Max number of VMs
- Max CPU cores
- Max storage capacity

Example:

Your Azure subscription may limit you to **10 vCPUs**.
You cannot create 100 VMs accidentally.

B. Policies (Rules)

Prevent users from breaking security guidelines.

Example:

Azure Policy may enforce:

- All storage must be encrypted
- No VMs in unapproved regions
- No public IP on sensitive servers

These rules protect the organization.

---

3. Showback / Chargeback

These are financial features in cloud usage.

⭐ Showback

Cloud **shows** how much each team consumed, but does not charge them money.

Example:

Team A used: $200
Team B used: $320

Management sees the cost, but nobody is billed internally.

⭐ Chargeback

Teams are **directly billed** based on their cloud usage.

Example:

Team A gets an invoice for $200.
Team B gets an invoice for $320.

This encourages teams to use cloud responsibly.

⭐ *Types of Cloud Computing*

1. Public Cloud

Cloud services provided to the **general public** over the internet.

✔ Who owns it?

Microsoft (Azure), Amazon (AWS), Google (GCP).

✔ Customers?

Anyone: individuals, startups, enterprises.

✔ Characteristics:

- Pay-as-you-go
- No hardware to manage
- Very scalable
- Shared infrastructure but isolated logically

Example:

You create a VM in Azure → Azure owns the data center.

---

2. Private Cloud

A cloud environment dedicated to **one single organization**.

✔ Who owns it?

Could be:

- The company itself (on-prem)
- A vendor offering a dedicated private cloud

✔ Characteristics:

- Higher control
- More secure
- Expensive
- Customized for the organization

Example:

A bank or government hosts servers in their own data center using VMware or Azure Stack.

Only **their** employees use it—no one else.

---

3. Hybrid Cloud

A combination of **public cloud + private cloud** working together.

✔ Why hybrid?

Some data must remain private (security/business rules), but public cloud is better for scalability.

Example:

A hospital stores patient data in a **private cloud**, but uses Azure public cloud for:

- Backups
- Analytics
- Large-scale computing

Both environments share data securely.

⭐ Summary Table

Concept	Meaning (Detailed)	Simple Example
Resource Pooling	Many users share the provider’s huge resources; cloud assigns automatically	Like many homes using the same huge water tank
Self Service	Create cloud resources instantly by yourself	Click “create VM” and it's ready

Quotas	Limits on usage to avoid overspending	Max 10 VMs allowed
Policies	Security rules enforced automatically	Cannot create VM without encryption
Showback	Shows how much cost each team generated	IT says: Team A spent $200
Chargeback	Teams are billed for what they use	Finance charges Team A $200
Public Cloud	Shared cloud for everyone	Azure, AWS
Private Cloud	Cloud for one organization only	Company's own data center
Hybrid Cloud	Mix of public + private cloud	Bank uses private for data, public for analytics

EXTRA Info

1. Multi-Cloud (Simple + Detailed + Example)

Simple meaning

Using **more than one cloud provider** at the same time.

Slightly detailed

You purposely choose **multiple cloud platforms** (like Azure + AWS + GCP) to use each for what they are best at.
You are *not* mixing them together in one system; you just use different clouds for different parts of your business.

Example

A company chooses:

- **Azure** → Identity & access (Azure AD), Windows workloads
- **AWS** → Data analytics (because they like AWS Athena, Redshift)
- **GCP** → Machine learning (because of Google AI tools)

So the company operates in **three clouds** simultaneously → **multi-cloud**.

Why companies use it

- Avoid becoming dependent on one vendor (vendor lock-in)
- Use best features from multiple clouds
- Global performance / redundancy

---

2. Community Cloud (Simple + Detailed + Example)

Simple meaning

A cloud **shared by organizations with similar needs**, usually managed privately.

Slightly detailed

It is not public like AWS/Azure, and not private for one company.
It is a cloud environment created for a **group of organizations that share the same standards, security needs, or regulations.**

They share:

- Infrastructure
- Security controls

- Compliance
- Costs

Examples

1. **Hospitals in one country** sharing a cloud for patient records (HIPAA compliance).
2. **Banks** sharing a cloud built by a financial regulatory authority.
3. **Universities** sharing cloud resources for research, data centers, and software.

Why use community cloud

- Similar compliance rules
- Reduced cost because groups share the infrastructure
- Increased security due to shared standards

---

3. Public vs Private vs Hybrid (Interview-Level Explanation with Examples)

Public Cloud

Cloud services offered to *anyone* on the internet, pay-as-you-go.

Examples:

- Azure

- AWS
- GCP

Real example:

A startup hosts their mobile app backend on Azure because it’s cheap and scalable.

Key point:

You don't own the infrastructure — the provider manages everything.

Private Cloud

A cloud that is used **exclusively by one organization**.

Can be:

- On-prem (your own data center)
- Hosted by a vendor but dedicated only to you

Example:

A government builds its own private cloud to store citizen data.

Key point:

High security, high cost, full control.

Hybrid Cloud

Mix of **private cloud + public cloud**, working together.

Example:

- A bank keeps customer data in its **private cloud** (security reasons).
- The bank runs AI/analytics workloads in **public Azure** for cost efficiency.
- Both environments are connected securely.

Key point:

You choose which workloads stay private and which move to public.

Summary Table

Cloud Type	Who Uses It	Who Owns It	Example
Public	Anyone	Cloud provider (Azure/AWS/GCP)	Deploying a website on Azure App Service
Private	One organization only	That organization	Bank's private data center
Hybrid	Organization using both private + public	Shared	Keeping sensitive data on-prem, running apps on Azure
Multi-Cloud	Organizations using multiple cloud providers	Many providers	Azure + AWS + GCP together
Community Cloud	Group with same goals	Shared among the group	Hospitals sharing a health cloud

---

⭐ *Some Fundamental Properties*

1. Reliability

Meaning:
Your system keeps working even if something goes wrong.
Cloud services are designed so that if part of your system fails, the whole application doesn't go down.

(a) Auto-Healing

Simple:
The system fixes itself automatically.

Example:
If one virtual machine crashes, Azure automatically replaces it with a healthy one.

(b) Storage Reliability

Simple:
Your data is kept safe even if disks fail.

Example:
Azure Storage stores **multiple copies** of your data (locally redundant, zone redundant), so even if one copy is lost, your data remains available.

(c) Auto-Scale

Simple:
Your app automatically increases or decreases the number of servers depending on traffic.

Example:
If a website suddenly gets 10× more users, Azure App Service scales up automatically so the site doesn't crash.

(d) SLA (Service Level Agreement)

Simple:
The cloud provider promises a certain level of uptime (like 99.9%).

Example:
Azure VM has 99.9% uptime SLA → your application is expected to be available almost all the time.

(e) Design for Failure

Simple:
Assume things *will* break, and build your system so it still works.

Example:

- Deploy apps in multiple regions
- Use load balancers to distribute traffic
- Store backups in a separate location

So even if one region goes down, your app stays online.

(f) Monitoring

Simple:
Checking continuously if your system is healthy or having problems.

Example:
Azure Monitor sends alerts when CPU usage is high or an application crashes.

---

2. Predictability

Meaning:
Your system behaves in a consistent way, and you can predict:

- **performance**
- **costs**
- **deployment behavior**

Predictability helps avoid surprises.

(a) SKUs / ACUs

SKUs (Stock Keeping Units)

Simple:
Different sizes or versions of cloud resources.

Example:
VM sizes like B2s, D4s, E8s → each has predictable performance and cost.

ACUs (Azure Compute Units)

Simple:
A performance rating that helps compare VM power.

Example:
A VM with 200 ACUs is twice as powerful as one with 100 ACUs.

Why this helps:
You can predict performance before deploying.

(b) Predictable Behavior

Simple:
Your system behaves the same way every time you deploy or change something.

Example:
Using the same VM SKU ensures the same performance everywhere.

(c) Use Templates

Simple:
Create resources using **ARM templates**, **Bicep**, or **Terraform** so everything is built the same way every time.

Example:
If you deploy 10 VMs using a template, all 10 VMs will have identical configuration.

(d) Automation

Simple:
Reduce manual work — let scripts or pipelines do repetitive tasks.

Example:
Auto-deploying code using Azure DevOps or GitHub Actions ensures consistent releases.

(e) DevOps

Simple:
A practice combining development + operations to automate and standardize processes.

Example:
Developers push code → pipeline tests it → pipeline deploys it → monitoring checks health.
This makes systems **predictable** and **reliable**.

Quick Summary Table

Concept	Meaning	Example
Auto-Healing	System fixes itself	VM replaced automatically
Auto-Scale	Grows/shrinks based on demand	App scales out during traffic spike
SLA	Uptime guarantee	99.9% uptime
Monitoring	Watch system health	Alerts for failures
Design for Failure	Expect failures and prepare	Multi-region deployment
SKUs	Resource sizes	B2s vs D4s VM
ACUs	Performance rating	100 ACU vs 200 ACU
Templates	Consistent deployment	ARM/Bicep/Terraform
Automation	Remove manual steps	CI/CD pipeline
DevOps	Continuous, predictable process	Automated testing + deployment