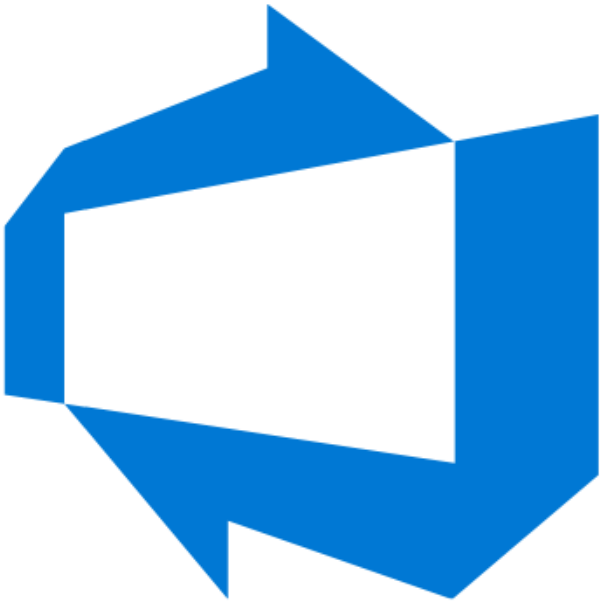

Azure DevOps

☁️ Cloud Computing ke 3 Main Types:

1. IaaS – Infrastructure as a Service 🌐

Kya hota hai?
IaaS mein tumhe **basic infrastructure** milta hai — jaise ki **servers**, **storage**, **networking**, aur **virtual machines (VMs)** — sab kuch cloud par, bina apna physical hardware setup kiye.

Example:

- **Azure Virtual Machines (VMs)**: Tum apne servers ko cloud par launch kar sakte ho, bina physical server lagaye.
- **Amazon EC2** (AWS ka equivalent): Yeh bhi virtual machines ko provide karta hai.

Kaise kaam karta hai?

- Tumhe sirf hardware ka management nahi karna padta.
- Tum apni applications ko install kar sakte ho, manage kar sakte ho, aur scale kar sakte ho.

Use Case:

- Agar tumhe apne applications ko quickly scale up/down karna ho.
- Large infrastructure manage karne ki need ho par apne servers nahi chahiye ho.

---

2. PaaS – Platform as a Service 🚀

Kya hota hai?
PaaS mein tumhe ek **ready-made platform** milta hai jahan tum apne **applications** ko **develop** aur **deploy** kar sakte ho — bina backend infrastructure ki tension ke. Yeh server, storage, networking ka management khud cloud provider kar leta hai.

Example:

- **Azure App Services**: Tum apne web applications ya APIs ko easily deploy kar sakte ho bina server setup kiye.
- **Google App Engine**: PaaS service jo applications ko deploy aur scale karne ke liye use hoti hai.

Kaise kaam karta hai?

- Tum apna code likhte ho aur directly cloud platform par deploy kar dete ho.
- Cloud provider automatically tumhare app ko scale karta hai, updates provide karta hai, aur runtime environment manage karta hai.

Use Case:

- Agar tumhe apne code ko quickly deploy karna hai aur infrastructure ka sochna nahi hai.
- Web apps ya APIs ko scale karna hai.

---

3. SaaS – Software as a Service 👨‍💻

Kya hota hai?
SaaS ek **ready-made software** hai jo internet par available hota hai. Tumhe bas apne browser ya app se login karna hota hai, aur tumhare paas software ka full access hota hai.

Example:

- **Office 365** (Word, Excel, etc.): Tumhare paas ready-made software available hai jo tum directly use kar sakte ho.
- **Salesforce**: A popular CRM software.

Kaise kaam karta hai?

- Tum software ko directly internet ke through use karte ho. Cloud provider sab kuch manage karta hai — software updates, security, storage, etc.
- Tumhe apna software install karne ki ya infrastructure manage karne ki zarurat nahi hoti.

Use Case:

- Agar tumhe ready-made software chahiye jo tum directly use kar sako, jaise email services, document editing, etc.
- Personal ya business applications ke liye jahan only software ka access chahiye, infrastructure ka nahi.

Azure Regions 🌍

What is a Region?
Azure mein ek **region** ek geographical location hota hai jahan Microsoft apne data centers ko set up karta hai. Har region multiple **availability zones** ho sakte hain.

Why Regions Matter?

- **Latency**: Tumhare users ke nazdeek region choose karne se latency kam hoti hai.
- **Compliance**: Kuch regions me data store karne ki specific rules hoti hain (like GDPR in Europe).
- **Disaster Recovery**: Different regions mein infrastructure hona disaster recovery strategies mein madad karta hai.

Example of Regions:

- **East US**
- **West US**
- **North Europe**
- **Southeast Asia**

Azure Availability Zones 🌐

What are Availability Zones (AZs)?
Availability Zones ek physical data center hote hain jo ek hi region mein hote hain. Har region mein 3 ya zyada availability zones ho sakte hain (region ke size par depend karta hai). In ka main purpose high availability aur fault tolerance provide karna hai.

Why Availability Zones Matter?

- **Redundancy**: Agar ek zone mein kuch issue ho, dusre zones se services chalu rehti hain.
- **Fault Isolation**: Agar ek zone fail ho jata hai (power failure, hardware issue), dusre zones pe impact nahi hota.
- **Business Continuity**: Inse tumhara application zyada reliable aur available rehta hai.

Example of Availability Zones:

- **East US** region mein 3 AZs ho sakte hain.
- **West Europe** region mein bhi 3 AZs.

Regions aur Availability Zones ka Relationship:

- **Region** ek broad geographical area hai (e.g., East US, West Europe).
- **Availability Zones** us region ke andar **independent** data centers hain jo power, cooling, aur networking mein isolated hote hain.

Key Benefits of Using Multiple Availability Zones:

1. **High Availability**: Agar ek availability zone mein koi problem hota hai, tumhare services automatically dusre zones mein switch ho jaati hain.
2. **Disaster Recovery**: Tum apni app ko multiple zones mein deploy kar sakte ho, taaki agar ek zone down ho jaye toh dusra zone kaam kare.
3. **Better SLA**: Microsoft tumhe **99.99% uptime guarantee** deta hai agar tum apni services ko multiple AZs mein deploy karte ho.

Example Use Case:

- **Web Application**: Agar tumhare paas ek web app hai, toh tumhare app ka front-end ek AZ mein ho sakta hai, aur back-end database doosre AZ mein, taaki agar kisi zone mein failure ho, tumhara web app continue rahe.

What is an Azure Virtual Machine (VM)? 🖥️

Azure VM ek **on-demand, scalable compute resource** hai jo tumhe cloud mein apne workloads ko run karne ka environment provide karta hai. Tum Azure ke data centers mein virtualized hardware ko use karte ho bina apna physical server manage kiye.

Types of Virtual Machines in Azure 🎛️

Azure mein **multiple types** of virtual machines available hain, jo specific workloads ke liye optimized hote hain. Inme se kuch main categories hain:

1. **General Purpose VMs (B, D-Series)**
    - **Use Case**: Balanced CPU, memory, and storage requirements for most common applications like small to medium databases, web servers, etc.
    - Example: **B-series**, **D-series**
2. **Compute Optimized VMs (F-series)**
    - **Use Case**: CPU-intensive applications. Good for high-performance computing tasks and batch processing, Gaming.
    - Example: **F-series**
3. **Memory Optimized VMs (E, M-series)**
    - **Use Case**: Applications that require a lot of memory, such as large databases or in-memory analytics.
    - Example: **E-series**, **M-series**

4. **Storage Optimized VMs (L-series)**
    - **Use Case**: Workloads that require high disk throughput, such as big data or database applications.
    - Example: **L-series**
5. **GPU Optimized VMs (NV, NC-series)**
    - **Use Case**: Graphic processing tasks like AI, machine learning, and rendering.
    - Example: **NC-series**, **NV-series**
6. **High Performance Compute VMs (H-series)**
    - **Use Case**: HPC (High-Performance Computing) workloads like simulations and scientific calculations.
    - Example: **H-series**

VM Configuration Options ⚙️

- **Operating Systems**: Tum apne VM mein Windows ya Linux OS install kar sakte ho.
- **Disk Types**: Azure VMs mein **standard HDD, SSD** ya **premium SSD** storage available hota hai.
- **Size**: Azure VM ka size depend karta hai tumhare requirements par — kitni CPU cores, RAM aur storage chahiye.
- **Networking**: VM ko **virtual network** ke saath connect karna hota hai, taaki tumhare VM securely communicate kar sake.

What is Azure Resource Manager (ARM)? 🛠️

Azure Resource Manager (ARM) ek management tool hai jo tumhare **Azure resources** ko organize aur manage karta hai, jaise VM, storage, aur databases, sab ek **resource group** ke andar. Yeh tumhe tumhare resources ko deploy, configure, aur control karne mein madad karta hai**. Easy monitoring, Easy cost analysis, Easy deletion, Easy access control**

Key Concepts:

1. **Resource Group**:
   Ek container hai jisme tumhare resources store hote hain. Agar resource group delete hota hai, toh uske saare resources bhi delete ho jaate hain.
2. **Resources**:
   Individual items jaise **VMs**, **databases**, **storage** jo tum deploy karte ho.
3. **Role-based Access Control (RBAC)**:
   Tum users ko specific permissions de sakte ho, jaise read, write, ya owner access.
4. **ARM Templates**:
   Yeh **JSON** files hain jo tumhare resources ko code ki tarah define karti hain. Isse tum easily infrastructure ko automate aur replicate kar sakte ho.

NOTES: Most of the resources like Virtual machine sb issi ke ander se bnega. Iske ander se sb track kr skte ho tum kya resources bne hai ek group ke under

Q) How are you managing your Azure resources in your company?

In our company, we manage Azure resources primarily using Azure Resource Manager (ARM). We organize all services into resource groups based on environment (like **dev, staging, and prod).**

For access control, we use Role-Based Access Control (RBAC) to ensure that only authorized users can perform specific actions.

We also use ARM Templates and Bicep for infrastructure-as-code to automate deployments. This helps maintain consistency across environments.

Additionally, we use tags for cost tracking and resource classification, and Azure Policy for enforcing governance like region restrictions or naming conventions.

Monitoring and security are handled using Azure Monitor, Log Analytics, and Azure Security Center.

How to run VM on Azure using Pem Key and Git Bash

✅ **Step 1: Prerequisites**

- Azure VM **already running** (Ubuntu/RedHat etc.)
- Public IP address of the VM (**52.188.185.120**)
- .pem file (SSH private key) (**Example: Anuj.pem**)

✅ **Step 2: Set Permissions on .pem File**

Git Bash me:

chmod 600 mykey.pem → **Chmod 600 Anuj.pem**

✅ **Step 3: Connect to Azure VM Using SSH**

ssh -i Location of pem.file azureuser@<Public-IP> → **ssh -I "C:\Users\syala\Downloads\new_key.pem" azureuser@52.188.185.120**

🔐 **chmod 400 – Read-only for owner**

- 👤 Owner: ✅ read
- 👥 Group: ❌ no access
- 🌍 Others: ❌ no access

🧠 **Use-case**: This is perfect for **SSH private key files** (like .pem) where **only read access** is needed and no one (not even you) should modify it.

🔐 **chmod 600 – Read + write for owner**

- 👤 Owner: ✅ read & write
- 👥 Group: ❌ no access
- 🌍 Others: ❌ no access

🧠 **Use-case**:
Also secure, but gives **write access to the owner**, which is okay for private files like .pem **if** you want to allow yourself to modify the file.

If I want to install Jenkins in this server: The steps are same as the AWS but yhn bhi security ground mein inbound rule mein jake port 8080 ko add krna hoga so azure mein.

Go to Azure Portal → VM → **Networking** → Add Inbound Port Rule:

- **Port**: 8080
- **Protocol**: TCP
- **Allow**

VMSS = Virtual Machine Scale Set (AWS Auto Scaling Group = Azure VMSS**)**

📘 **Definition:**

VMSS ek **Azure service** hai jo automatically **multiple VMs ko create, manage aur scale** karta hai, based on **demand**.

🔑 **Key Points:**

Feature	**Explanation**
🔁 **Auto Scaling**	Traffic badhta hai to zyada VMs ban jaate hain, kam hota hai to ghatt jaate hain
👥 **Uniformity**	Sab VMs same image/configuration se bante hain
⚙️ **Load Balancer ke sath aata hai**	Traffic automatically sab VMs me distribute hota hai
🧱 **Infrastructure as Code** support	ARM, Bicep, Terraform se bana sakta hai
🔐 **Integration with Azure Monitor/Alerts**	Performance ke basis pe scale kar sakta hai

🎯 **Use-case Example:**

Soch le tu ek **streaming app** deploy kar raha hai:

- Jab match chal raha hota hai, traffic high → VMSS **auto-scale** karke 10 VMs chala deta hai
- Jab raat ko koi nahi dekh raha → sirf 1-2 VM active
- Tu tension free – Azure sambhalta hai

Azure VNet (Virtual Network) – AWS VPC

🔹 **Kya hota hai VNet?**

Azure VNet ek **logically isolated network** hai jo tu Azure me banata hai, jisme tu apne **VMs, databases, containers, app services** ko daal sakta hai — ye sab ek doosre se **securely communicate** kar sakte hain.

VNet (Virtual Network) isliye introduce kiya gaya Azure me taaki tu apne **cloud resources** ko ek **secure, isolated aur manageable network** ke andar rakh sake — **bilkul waise jaise tu apne office ka LAN banata hai**.

Jaise ki tu ek society banata hai with:

- Ghar (VMs)
- Roads (Subnets)
- Security gates (NSG)
- Rules (Route Tables)

🔐 **Kya-kya control milta hai VNet me?**

Feature	Explanation
🏠 **Subnets**	VNet ke andar tu logical networks banata hai (like gharon ke blocks)
🔐 **NSG (Network Security Group)**	Tu har subnet/VM pe firewall rules laga sakta hai
🚦 **Route Table**	Kaunsa traffic kahan jayega, tu control karta hai
🌐 **Public/Private IPs**	Internal aur external communication ke liye
🔗 **VNet Peering**	Do alag-alag VNet ko connect karne ka tareeka
🔌 **VPN Gateway / ExpressRoute**	On-premise data center ko Azure se jodne ke liye

🧠 **Subnet – Simple Definition:** "Subnet ek logical portion hai Azure Virtual Network ka, jisme tu specific resources (jaise VMs, DBs) daal ke unka traffic aur access control manage karta hai." **Soch le VNet = bada network, aur Subnet = uske andar chhote logical tukde. Jaise ek city ke andar sectors ya colonies hote hain — waise hi VNet ke andar Subnets.**

Kyun use hota hai Subnet?

Reason	Explanation
✅ **Segregation**	Alag-alag tier/role ke resources ko alag subnets me daalna (web, DB, app)
🔒 **Security**	Har subnet pe tu **alag NSG (firewall)** rules laga sakta hai
📊 **Traffic Management**	Routing rules alag ho sakti hain per subnet
🧱 **Isolation**	Web tier ka subnet internet-facing ho sakta hai, DB ka subnet **internal-only** ho sakta hai

Example:

Subnet Name	CIDR Range	Resources	Internet Access
web-subnet	10.0.1.0/24	Web VMs, App Gateway	✅ Yes
app-subnet	10.0.2.0/24	App layer VMs	❌ No (only internal)
db-subnet	10.0.3.0/24	Azure Database / MySQL	❌ No (private endpoint)

🎯 Real-Life Use Case Example:

Soch:

- Ek company ka web app Azure me deploy hai
- Ek VNet banaya: 10.0.0.0/16
    - Subnet1: Web VMs (10.0.1.0/24)
    - Subnet2: Database VMs (10.0.2.0/24)
- NSG rule:
    - Subnet1 → Internet allowed (HTTP/HTTPS)
    - Subnet2 → Sirf Subnet1 se access allowed (no internet)

🧠 Isse:

- Web server public access se baat karta hai
- Database **internally secure** rehta hai
- Performance fast, security strong

🔍 Why Azure VNet Was Introduced? 🎯 Main Use: Secure aur Controlled Networking in the Cloud

Cloud me sab kuch public nahi hota — agar tu VM banata hai ya database, to sabko **internet se directly access nahi chahiye hota**.

Tu chahta hai:

- Private access
- Firewall control
- Internal-only communication
- VPN ya data center se secure connection

Ye sab **possible nahi hota** bina ek proper **networking layer** ke — isliye **Azure ne VNet introduce kiya.**

🧠 VNet ka Use Kya Hai?

1. ✅ Isolated Private Network

- Tera **VM, DB, container, app** sab ek **private network** me honge
- Baaki Azure customers se **completely separate**
- Like apna office ka LAN

2. 🔒 Security & Access Control

- Tu define karega:
    - Kaunsa subnet internet se baat kare
    - Kaunsa sirf internal communication kare
- **NSG (Network Security Group)** laga ke tu firewall rules define kar sakta hai

3. 🔗 Hybrid Connectivity

- Apne on-premises data center se Azure VNet ko **VPN ya ExpressRoute** se jod sakta hai
- Soch le: tera local server aur Azure ka server ek hi network me baat kar rahe

4. 📶 Service Communication

- Azure services (App Service, Key Vault, Storage, etc.) ko tu **Private Endpoint** ke through VNet me la sakta hai
- Internal traffic → secure → fast → cost-effective

5. ⚙️ Subnets, Routing & Traffic Control

- Large VNet ko tu **subnets** me tod ke divide kar sakta hai
- Har subnet ka apna role: Web, DB, App Layer
- **Custom routing** laga ke traffic ko specific direction me bhej sakta hai

6. 🌐 Load Balancer & DNS Support

- Tu VNet ke andar hi **Azure Load Balancer** ya **Application Gateway** use kar sakta hai
- Tu apna **custom DNS server** bhi configure kar sakta hai

🚨 **NSG ka Main Function:** NSG (Network Security Group) ek firewall jaise kaam karta hai jo Azure ke Virtual Network (VNet) ke andar traffic ko control karta hai. Yeh inbound (incoming) aur outbound (outgoing) traffic ko allow ya deny karta hai based on defined rules.

- Secure karna tera network traffic by controlling kaunsa traffic allow ho aur kaunsa deny ho, ek VM ya subnet ke liye.

🔑 **NSG ke Key Features:**

1. **Inbound Rules**:
    - Yeh define karta hai ki kaunsa traffic **allow** hoga jab woh kisi VM ya subnet ke andar aayega.
2. **Outbound Rules**:
    - Yeh define karta hai ki kaunsa traffic **allow** hoga jab woh kisi VM ya subnet ke andar se **bahar** jayega.
3. **Priority**:
    - NSG rules ki ek **priority** hoti hai — jo **lowest number** ka rule hota hai woh pehle apply hota hai.
4. **Allow & Deny**:
    - Tu specific ports (jaise 22 for SSH, 80 for HTTP, 443 for HTTPS) ko allow ya deny kar sakta hai.
    - Tu specific **IP addresses** ya **IP ranges** ko bhi allow ya deny kar sakta hai.

🛡 **NSG ka Real-Life Analogy:**

Soch, **NSG** ek **security guard** hai jo **gate par khada hota hai** aur **decide karta hai** ki kaun andar aa sakta hai aur kaun nahi.
Gate = Network connection
Guard = NSG rules

🧠 **NSG ka Example:**

1. **Scenario 1**: Tere paas ek **Web Server** hai jo **public internet** se access ho sakta hai.
    - Inbound rule: Allow **Port 80/443** (HTTP/HTTPS) from anywhere.
    - Outbound rule: Allow all traffic (VM can reach internet for updates).
2. **Scenario 2**: Tere paas ek **Database Server** hai jo **sirf internal app se** communicate karega.
    - Inbound rule: Allow **Port 1433** (SQL) **only from app subnet**.
    - Outbound rule: Deny all traffic (private connection).

🧠 **ASG (Application Security Group):** ASG (Application Security Group) bhi ek network security feature hai Azure me, jo NSG se thoda different hai. ASG ka primary role dynamic scaling aur application-tier isolation ko simplify karna hai, jabki NSG ka kaam traffic filtering pe focus karta hai. ASG basically ek logical grouping hai tujhse related Azure resources ko. Iska main use traffic ko organize aur manage karne me hota hai, especially jab tere paas large-scale applications ho jahan pe VM ya services ko dynamically group karna ho.

🎯 **ASG vs NSG: Difference**

Feature	NSG (Network Security Group)	ASG (Application Security Group)
Purpose	Filters inbound/outbound traffic based on rules	Logical group for managing security at application layer
Use Case	Define specific traffic rules for individual VMs or subnets	Grouping VMs together for easy management in large apps
Control	Allows or denies traffic based on **IP, port, and protocol**	Groups VMs for **easier security rule application**
Granularity	More granular (IP and port-based)	More abstracted (grouped by application)

🔑 **Key Features of ASG:**

1. **Application Layer Security**:
    - ASG resources are grouped based on their application role (e.g., **Web Tier, App Tier, DB Tier**).
    - For example, all web VMs can be in an ASG, and all database VMs in another ASG.
2. **Simplified Rule Management**:
    - Agar tu 50 VMs ko **web app** bana ke **publicly accessible** rakhna chahta hai, toh tu unhe ek ASG me daal dega aur **NSG** ke rules se unko manage kar sakta hai. Tu **IP addresses** ko directly manage nahi karega.
3. **Dynamic Scaling**:
    - Agar VM dynamically scale ho rahi hai (e.g., scale-up or scale-out), toh woh **same ASG** ke andar rahegi. Jab resources scale karte hain, tu **ASG ko change nahi karna padta**.
4. **Ease of Management**:
    - Instead of managing rules for each individual VM, tu **entire group** ko ek hi time pe manage kar sakta hai.

- **Easy to apply**: Traffic rules ko ek hi jagah apply karna as compared to applying rules on every individual VM.

🎯 **Real-Life Example:**

Soch le tu ek **3-tier architecture** bana raha hai:

- **Web Tier** (public access)
- **App Tier** (internal communication)
- **DB Tier** (private access)

Instead of managing individual VMs, tu unko **ASG** me group kar sakta hai:

ASG Name	VMs (Assigned)	Role
Web-AG	VM1, VM2	Web Tier (Public)
App-AG	VM3, VM4	Application Tier
DB-AG	VM5, VM6	Database Tier

Ab, tu **NSG rules** ko **ASG** par apply karega:

1. **Web-AG** → Public traffic allowed (HTTP, HTTPS)
2. **App-AG** → Only communication allowed from **Web-AG**
3. **DB-AG** → Only communication allowed from **App-AG**

🔁 **Scenario:**

- Tere paas **2 subnets** hain:
    1. user-subnet → jisme users ya VMs hain
    2. db-subnet → jisme Database hai
- Ab tu chahta hai:
    - **Kuch users/VMs** (in user-subnet) **DB ko access kar saken**
    - Aur **kuch users/VMs** DB ko **access na kar saken**

⚙️ **To yahan pe NSG aur ASG ka kaam kaise aata hai?**

Role	Explanation
NSG	**Traffic control engine** hai. Yeh decide karega: "Yeh traffic allow hai ya deny?"
ASG	**Tagging/grouping mechanism** hai. Tu VMs ko logically group karta hai jaise “TrustedUsers”, “NonTrustedUsers”, etc.

✅ **Solution: Use NSG + ASG Together**

1. Pehle, tu user-subnet ke andar kuch VMs ko **ASG me daalega**:
    - Example:
        - VM1 → ASG: TrustedUsers
        - VM2 → ASG: NonTrustedUsers
2. Fir tu db-subnet pe ek **NSG rule** lagayega:

NSG Rule on DB Subnet:

- ✅ Allow inbound traffic on port 1433 (SQL) **from ASG TrustedUsers**
- ❌ Implicitly deny traffic from others (including NonTrustedUsers)

🔐 **Result:**

VM	Subnet	ASG	DB Access
VM1	user-subnet	TrustedUsers	✅ Allowed
VM2	user-subnet	NonTrustedUsers	❌ Denied

🔁 **Internally Kya Ho Raha Hai:**

- NSG **IP based** filtering karta hai normally.
- Lekin jab tu ASG use karta hai, to tu IPs ki jagah **ASG name** use karta hai — which is more dynamic and scalable.

NSG + ASG = AWS Security Group + Tags / Separate Security Groups

Route 53 of AWS = Azure DNS ek service hai jo teri **custom domain names** ke liye **name resolution** provide karti hai — jaise ki "myapp.com" ko uske actual **IP address** se map karna. Yeh fully managed service hai jo **DNS zone management**, **records creation**, aur **fast DNS resolution** allow karti hai using Azure infrastructure.

Azure Load Balancer aur **Azure Application Gateway** dono hi load distribution ke liye use hote hain, lekin **use-case, layer, aur features** alag-alag hote hain.

🔁 **1. Azure Load Balancer (ALB)** (Internet ko face krne ke liye)

📍 **Layer: OSI Layer 4 (Transport Layer)**

(Works on IP + Port → TCP/UDP level)

🎯 **Use When:** Tu **VMs, Virtual Machine Scale Sets**, etc. ke beech **TCP/UDP traffic distribute** karna chahta hai

- Jaise:
    - Distribute traffic across multiple backend VMs
    - Game servers, low-level apps
    - SQL Server clusters

🔧 **Features:**

Feature	Description
Protocol	TCP/UDP only
Health probe	Basic (TCP/HTTP)
Rules	Port-level
Performance	Ultra-low latency (good for infra-level load balance)
Public + Internal	Dono types ka LB available

🛠 **Example:**

- Public Load Balancer distributes incoming traffic on port 80 to:
    - VM1 (10.0.0.4:80)
    - VM2 (10.0.0.5:80)

🌐 **2. Azure Application Gateway (Internal Traffic)**

📍 **Layer: OSI Layer 7 (Application Layer)**

(Works on HTTP/HTTPS level, jaise browser traffic)

🎯 **Use When:**

- Tu **web apps** ke liye intelligent routing chahta hai
- Jaise:
    - Route traffic to /login → backend A
    - Route /api → backend B
    - Handle SSL termination, web application firewall (WAF)

🔧 **Features:**

Feature	Description
Protocol	HTTP/HTTPS
Smart Routing	URL/path-based, host-based
SSL Termination	Yes (decrypts HTTPS)
WAF	Built-in Web Application Firewall
Cookie-based session affinity	Yes
Autoscaling	Yes

🛠 **Example:**

- Traffic to:
    - myapp.com/login → AppService-1
    - myapp.com/api → AppService-2

🤔 **Toh Tere Use-case ke liye Kaunsa?**

- **Web App, APIs, WAF, SSL chahiye?**
  → ✅ Use **Application Gateway**
- **Simple TCP Load Balancing (VMs, ports)?**
  → ✅ Use **Azure Load Balancer**

WAF (Web Application Firewall) Overview: Yeh ek security measure hai jo web applications ko malicious traffic se protect karta hai.

Azure me Azure Web Application Firewall (WAF) ek managed service hai jo web applications ko common threats se protect karta hai, jaise SQL injection, cross-site scripting (XSS), aur other OWASP top 10 vulnerabilities.

WAF, primarily web applications ko protect karta hai by filtering, monitoring, and blocking HTTP traffic to and from a web application. Yeh bad requests aur malicious requests ko identify karta hai aur unhe block karta hai.

Azure me WAF typically use hota hai with services like Azure Application Gateway or Azure Front Door. Yeh security policies ko define karne mein madad karta hai jo specific traffic patterns ko allow ya deny karti hain.

Key Features of Azure WAF:

1. **Protection against OWASP Top 10:**
    - WAF helps in protecting web applications from the OWASP Top 10 threats like SQL injection, Cross-Site Scripting (XSS), and others.
2. **Custom Rules:**
    - Azure WAF allows you to define custom security rules tailored to your application's needs, beyond the default OWASP ruleset.
3. **Bot Protection:**
    - Azure WAF can identify and block malicious bots trying to exploit the web application.
4. **Logging and Monitoring:**
    - WAF integrates with Azure Monitor and provides logs and metrics to track potential security events.
5. **Real-time Protection:**
    - WAF provides real-time protection by blocking requests based on the configured security policies. It can automatically block suspicious or harmful requests.
6. **Managed Rules:**
    - Azure offers built-in managed rule sets (e.g., from the OWASP Top 10, Microsoft-specific rules) to quickly set up protection against common threats.

7. **Geographical Restriction:**
    - You can restrict or allow traffic from certain geographic locations, adding an extra layer of security.
8. **Rate Limiting:**
    - You can configure WAF to limit the rate of requests to prevent Denial of Service (DoS) attacks.

Azure WAF Deployment Options:

1. **Azure Application Gateway with WAF:**
    - Application Gateway ek web traffic load balancer hai, aur usme built-in WAF hota hai. It can be configured to protect web applications from a variety of attacks.
    - WAF policies can be applied to an Application Gateway to filter incoming traffic.
2. **Azure Front Door with WAF:**
    - Azure Front Door provides global HTTP/HTTPS load balancing and fast content delivery. With WAF integration, it can protect your applications from threats while ensuring high availability.

Azure WAF Modes:

1. **Prevention Mode:**
    - In this mode, WAF blocks malicious requests based on configured rules (blocking mode).
2. **Detection Mode:**
    - In this mode, WAF detects and logs malicious requests without blocking them. It's more of a monitoring and logging feature.

---

1. VNet Peering:

VNet Peering allow karta hai two virtual networks ko connect karne ke liye, chahe woh same region mein ho ya different region mein (Global VNet Peering). Isse networks ke beech traffic pass ho sakta hai jaise woh ek hi network mein ho.

Key points about VNet Peering:

- **Same region peering** aur **Global VNet Peering** available hai.
- VNet Peering mein traffic directly, securely, aur efficiently route hota hai dono networks ke beech.
- No need for VPN gateway or public IP addresses.
- **Private IP addressing** ka use hota hai, jo internal communication ko secure rakhta hai.
- Peered VNets ke beech services access karna possible hota hai.

Example use cases:

- Different VNets ko same region mein connect karna.
- Remote office locations ko ek VNet mein integrate karna.
- Multi-tier architecture (e.g., frontend VNet aur backend VNet ko connect karna).

2. VNet Gateway:

VNet Gateway ka use Azure Virtual Network ko on-premises network se connect karne ke liye hota hai, ya phir ek VNet ko dusre VNet se connect karne ke liye, typically using VPN connections or ExpressRoute.

Key points about VNet Gateway:

- **VPN Gateway** ke through secure connection create hota hai between Azure VNet aur on-premises network.
- **ExpressRoute** ka use karte hue VNet ko dedicated, private connection ke through on-premises data centers se connect kar sakte hain.
- VNet Gateway **Internet-based** ya **private** connections ke liye configure kiya ja sakta hai.
- **Gateway Subnet** ki requirement hoti hai, jo VNet mein create hota hai aur usme VNet Gateway resources deploy hoti hain.

Example use cases:

- On-premises network ko Azure VNet se connect karna (VPN).
- Multiple VNets ko on-premises network se securely connect karna.
- Hybrid cloud setup create karna.

In summary:

- **VNet Peering**: Same region or different region ke VNets ko connect karta hai using private IPs.
- **VNet Gateway**: Azure VNet ko on-premises ya dusre VNets se securely connect karne ke liye use hota hai, typically VPN or ExpressRoute ke through.

3. VPN Gateway Overview:

Azure VPN Gateway, VPN connections ko manage karta hai between Azure VNet aur remote locations. Yeh encrypted tunnels (IPSec or IKE) ke through communication establish karta hai.

Types of VPN Gateway Connections:

1. **Site-to-Site VPN**:
    - **Site-to-Site (S2S)** VPN connection use hota hai jab aap apne on-premises network ko Azure VNet se connect karna chahte hain.
    - Yeh connection ek **VPN Gateway** use karta hai on both ends (on-premises aur Azure side).
    - Usually **IPSec/IKE (Internet Protocol Security/Internet Key Exchange)** protocols use hote hain for secure data transmission.

Use case: Agar apne on-premises data center ko Azure VNet se securely connect karna hai.

2. **Point-to-Site VPN**:
    - **Point-to-Site (P2S)** VPN connection individual clients ke liye hota hai. Yeh connection users ke devices ko directly Azure VNet se connect karta hai.
    - Yeh **SSL VPN** ya **IKEv2** protocols us` 1 e karta hai.
    - Typically **remote workers** ya **developers** use karte hain jab unhe Azure resources access karne hon.

Use case: Remote workers ko Azure resources access karne ke liye VPN connection establish karna.

3. **VNet-to-VNet VPN**:
    - Yeh connection use hota hai jab do Azure Virtual Networks ko securely connect karna ho.
    - VNet-to-VNet connection, VPN Gateway ka use karta hai to establish the tunnel between two VNets.

Use case: Agar aapko multiple VNets ko interconnect karna ho securely, like different environment networks (e.g., dev and prod VNets).

Key Features of VPN Gateway:

- **Secure communication** using IPSec and IKE protocols.
- Supports both **Static** and **Dynamic routing**.
- Supports high availability and **resilient connectivity** with multiple gateway types.
- Can be connected to **on-premises** networks, other VNets, or even internet-based clients (P2S).
- VPN Gateway mein **Gateway Subnet** hona zaroori hota hai, jahan pe VPN Gateway resource deploy kiya jata hai.

VPN Gateway vs. VNet Peering:

- **VPN Gateway** typically external networks (on-premises or another VNet) se Azure VNet ko connect karta hai.
- **VNet Peering** usually internal networks ko connect karta hai, i.e., two VNets within Azure (no external connection).

Example use cases for VPN Gateway:

- On-premises office ko Azure VNet se securely connect karna via Site-to-Site VPN.
- Remote employees ko Azure VNet ke resources access dena via Point-to-Site VPN.
- Two VNets ko interconnect karna (within Azure) using VNet-to-VNet connection.

🔐 **What is Azure Bastion?**

Azure Bastion is a **PaaS (Platform as a Service)** that provides **secure RDP/SSH access** to your Azure VMs directly from the **Azure portal** over **SSL (port 443)** — without needing public IPs on the VMs.

Normally agar aap kisi Azure VM pe SSH (Linux) ya RDP (Windows) karna chahte ho, toh aapko:

- Public IP assign karna padta hai
- Port 22 (SSH) ya 3389 (RDP) kholna padta hai

Yeh risky hai, kyunki internet se directly exposed hota hai. **Azure Bastion** aapko yeh sabse bachata hai — VM ke andar jaane ke liye **portal se hi connect kar sakte ho**, bina public IP ke.

🔍 **Key Features:**

Feature	Description
🔒 **No public IP required**	VMs secure rehte hain, sirf private IPs hoti hain
🌐 **Access via browser**	SSH/RDP access directly from Azure Portal
🔐 **Uses SSL (443)**	Firewall-friendly, koi open inbound ports nahi
📜 **Audit & Logging**	Centralized logging ke liye integrate kar sakte ho
🌍 **VNet-wide access**	Bastion once setup ho gaya toh us VNet ke sabhi VMs ko access kar sakte ho

✅ Steps to Create a Virtual Network (VNet) in Azure:

- **Step 1: Login to Azure Portal**
  - Go to: https://portal.azure.com
  - Sign in with your Azure credentials.
- **Step 2: Search for "Virtual Network"**
  - On the top search bar, type: Virtual Network
  - Click on **"Virtual Networks"** under **Services**.
- **Step 3: Click on "Create"**
  - Then click **+ Create** or **+ Add** to start the VNet creation wizard.
- **Step 4: Basics Tab**

Fill in the basic details:

- **Subscription**: Choose your Azure subscription.
- **Resource Group**: Select existing or click "Create new".
- **Name**: Give your VNet a name, e.g., myVNet.
- **Region**: Select the region where you want to create this VNet (e.g., East US, Central India).

- **Step 5: IP Addresses Tab**

This is important – define your **Address Space** and **Subnets**:

- **IPv4 address space**: e.g., 10.0.0.0/16
- Click **+ Add subnet**:
  - **Subnet name**: e.g., default
  - **Subnet address range**: e.g., 10.0.1.0/24

(Note: You can add more subnets later for app-tier, db-tier, etc.)

- **Step 6: (Optional) Security Tab**
  - You can enable **BastionHost**, **Firewall**, or **DDoS protection** if needed.
  - For learning, you can skip this or keep defaults.
- **Step 7: (Optional) Tags Tab**
  - Add tags if you want (for cost tracking, project ID, etc.)
- **Step 8: Review + Create**
  - Azure will validate your settings.
  - If everything looks good, click **Create**.

Bhai, ab jab tu ne Virtual Network (VNet) create kar liya, ab uske andar Virtual Machine (VM) create karna simple hai.

✅ Steps to Create a VM inside a Virtual Network (VNet):

- **Step 1: Go to Azure Portal:** Visit: https://portal.azure.com
- **Step 2: Search for "Virtual Machines"**
  - Top search bar mein type kar: Virtual Machines
  - Click on the **Virtual Machines** service.
- **Step 3: Click on + Create**
  - Click **+ Create** > **Azure Virtual Machine**
- **Step 4: Basics Tab**

Fill out the required info:

- **Subscription**: Select your Azure subscription.
- **Resource Group**: Select the same group jisme VNet hai.
- **Virtual Machine name**: e.g., myVM
- **Region**: Select the same region as your VNet.
- **Availability options**: Keep default unless needed.
- **Image**: Choose OS (e.g., Ubuntu 20.04 LTS or Windows Server)
- **Size**: Choose appropriate size (e.g., B1s for testing)
- **Authentication type**:
  - **SSH public key** (Linux) ya
  - **Password** (Windows)
- **Username/Password**: Enter as needed.

- **Step 5: Disks Tab**
  - Keep default (Standard SSD) or choose as per your need.
- **Step 6: Networking Tab (IMPORTANT!)**

Here tu VNet select karega jisme VM create karni hai:

- **Virtual Network**: Select your existing VNet (e.g., myVNet)
- **Subnet**: Choose subnet (e.g., default)
- **Public IP**: Select **None** if you're using **Bastion**, ya select "new" if direct access chahiye.
- **NIC network security group**: You can choose basic and allow SSH (Linux) or RDP (Windows).

- **Step 7: (Optional) Management, Monitoring, Tags**
  - Enable/disable monitoring, auto-shutdown, etc. as per need.
- **Step 8: Review + Create**
  - Azure will validate your configuration.
  - Click **Create**.

🔐 **How to Connect to the VM?**

Option 1: If you selected Public IP:

- Go to VM → Click **Connect** → Choose **SSH (for Linux)** or **RDP (for Windows)**
- Use given IP to access from your local machine.

Option 2: If you selected No Public IP:

- Use **Azure Bastion** to connect securely via browser:
    - VM → Connect → Bastion → Enter credentials → Access via browser.

🟢 **Azure Bastion = Easy + Secure + No maintenance**
🟡 **AWS Bastion = Manual setup + More control but more work**

✅ **Azure Bastion:**

- **Fully managed service** (PaaS) by Azure
- **Browser se direct SSH/RDP**
- **No public IP** needed on VMs
- **Auto-scaled, secure, no maintenance**

✅ **AWS Bastion Host:**

- **EC2 instance** tu khud banata hai
- SSH/RDP ke liye **public IP** lagta hai Bastion pe
- Tu hi **patching, security, scaling** handle karta hai
- **Extra setup** & manual management

Azure Interview Questions:

✅ **How to Block Access from a Particular Subnet:**

🔧 **Using NSG (Network Security Group)**

1. **Go to NSG** (attached to your subnet or NIC)
2. Add a **Deny rule** in **Inbound security rules**:
    - **Source**: The subnet you want to block (CIDR format, e.g., 10.1.0.0/24)
    - **Source Port Range**: *
    - **Destination**: Your VM or subnet (default is Any)
    - **Destination Port Range**: 22 (SSH) or 3389 (RDP) or * for all
    - **Protocol**: Any
    - **Action**: Deny
    - **Priority**: Lower number than any existing "Allow" rule (e.g., 100) (Least number has highest priority and least number we can choose is only 100)
3. Click **Save**.

✅ is NSG stateful or stateless?

NSG (Network Security Group) in **Azure is *stateful***.

✅ **What does *stateful* mean?**

- Jab **inbound connection** allow hoti hai, toh **return outbound traffic** automatically allow ho jaata hai.
- Aur agar **outbound allow** kiya hai, toh response ke liye **inbound** bhi allowed hota hai — bina explicitly likhe. Meaning you don't have to open the port in outbound rules to send response back.

⚠️ **Opposite:** *Stateless* hota hai Azure Firewall rules, Route Tables, etc., jahan tu explicitly har direction define karta hai.

✅ **What is Custom Data in Azure or AWS user data ?**

Tu VM create karte waqt ek script ya config file de sakta hai (bash script, cloud-init, PowerShell etc.) Ye script **VM ke first boot** par run hoti hai (one time)

Use hota hai:

- Packages install karne ke liye
- App configure karne ke liye
- Environment setup karne ke liye

✅ **Azure Front Door:** Azure Front Door ek global, scalable, and secure entry point hai for your web applications. Ye mainly web traffic (HTTP/HTTPS) ke liye hota hai, aur features deta hai jaise global load balancing, SSL termination, caching, and web application firewall (WAF).

"Ye duniya bhar ke users ke liye fastest, secure, aur smart route choose karta hai to reach your web app."

OSI Layer - Layer 7 (HTTP/HTTPS)
Built-in WAF Support
Caching is present
Global hota hai

Assume your company is using all the ideal azure networking setup and your app is deployed in the web subnet, what is the traffic flow to your app?

🌐 **Traffic Flow to Your App in Web Subnet (Ideal Setup)**

📥 **1. Client/User → Azure Front Door / CDN**

- Global edge location pe request aati hai
- Front Door handles global routing, SSL termination, caching, etc.
- Fastest path choose karta hai

🔁 **2. Front Door → Azure Application Gateway (WAF Enabled)**

- Front Door se request jaati hai Application Gateway ko
- WAF (Web Application Firewall) filters malicious traffic (XSS, SQLi, etc.)
- SSL offloading bhi yahin ho sakta hai

🧠 **3. Application Gateway → Azure Load Balancer (Optional)**

- If backend mein multiple tiers/services hain (e.g., API, DB), traffic can go via Internal Load Balancer
- Layer 4 (TCP/UDP) load balancing

🌐 **4. Load Balancer → Web Subnet → App VM / App Service**

- Request reaches your actual App hosted in Web Subnet
    - Could be: VM Scale Set, App Service Environment (ASE), AKS Node, etc.
- App processes the request and responds

📩 **5. Response Path:**

- App → Load Balancer → App Gateway → Front Door → Client

🔐 **Other Networking Components in Ideal Setup:**

- NSGs (Network Security Groups):
  Restrict traffic between subnets (e.g., only App Gateway can talk to Web Subnet)
- UDRs (User Defined Routes):
  For routing traffic via NVA (firewall) or Azure Firewall
- Azure Bastion:
  Secure RDP/SSH access to VMs without public IPs
- Private DNS Zones + VNet Peering:
  For internal resolution & communication

AZURE Storage Services

Azure Storage Microsoft ka ek cloud-based storage solution hai jo tumhare data ko securely store karne, access karne, aur manage karne ke liye use hota hai. Ye highly scalable, durable, and secure hota hai.

🔸 **Types of Azure Storage Services**

1. Blob Storage also Container (Binary Large Object) (Cheap) → Azure Blob Storage = Amazon S3

Use: Images, videos, documents, backups, big files, unstructured data.

- Types of blobs:
    - Block blob (large files like images/videos)
    - Append blob (log files, etc.)
    - Page blob (used for virtual hard disks - VHDs)

📦 **Example use case:** Tumhare app ke users ne profile pics upload kari, wo blob storage me store hongi.

---

2. File Storage → Azure File Storage = Amazon EFS

- Use: File shares jo SMB protocol ke through accessible hote hain.
- Network file share jaise kaam karta hai jo tum Windows/Linux system me mount kar sakte ho.

📁 **Example use case:** Tumhare company ka shared drive jaha documents rakhe gaye hain.

3. Queue Storage

- Use: Messaging system for decoupling components.
- Ek component message bhejta hai, doosra later process karta hai.

📩 Example **use case:** User ne image upload kari, ek message queue me gaya ki “process this image”, aur background service usse process karti hai.

4. Table Storage

- Use: NoSQL key-value store.
- Structured data store karne ke liye use hota hai (fast access, scalable).

📊 Example use case**:** App ke logs ya user data like scores, preferences, etc.

5. Disk Storage → Azure Disk Storage = Amazon EBS

- Use: Virtual machines ke liye hard disk.
- Ye VHDs (virtual hard disks) hoti hain jo tumhare Azure VM ke OS ya data disk ke roop me use hoti hain.

💿 **Example use case:** Tumhare Azure VM ke liye ek SSD disk.

Note: Pehle Stroage Account bnao, phir uske ander Storage bnane ke option aaenge, choose kro aur storage bnao.

Azure CLI: Azure CLI (Command-Line Interface) is a cross-platform command-line tool used to manage Azure resources. You can use it on Windows, macOS, and Linux to create, manage, and configure Azure services without using the Azure portal.

🔧 **How to Install Azure CLI:**

 # macOS: brew install azure-cli

 # Ubuntu: curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash

🧰 **Common Azure CLI Commands:**

 # Create a resource group: az group create --name myResourceGroup --location eastus

 # Delete a resource group: az group delete --name myResourceGroup --yes

Virtual Machines

 # Create a VM: az vm create --resource-group myResourceGroup --name myVM --image UbuntuLTS --admin-username azureuser --generate-ssh-keys

 # Start/Stop/Delete a VM

az vm start --name myVM --resource-group myResourceGroup

az vm stop --name myVM --resource-group myResourceGroup

az vm delete --name myVM --resource-group myResourceGroup --yes

```
# Create a storage account

az storage account create --name mystorageaccount --resource-group myResourceGroup --location eastus --sku Standard_LRS
```

Azure CLI is used because it provides a **fast, efficient, and scriptable way** to manage Azure resources without relying on the Azure Portal UI. Here's a breakdown of **why it's widely used**:

🚀 1. Automation

- You can write **scripts** to automate repetitive tasks (e.g., provisioning VMs, setting up networks, deploying applications).
- Easily integrates into **CI/CD pipelines** for DevOps workflows.

⌨️ 2. Command-Line Access

- Direct control from terminal or command prompt.
- Useful for developers, sysadmins, and DevOps engineers who prefer keyboard over GUI.

🔄 3. Consistency Across Platforms

- Azure CLI works on **Windows, macOS, and Linux**.
- Behavior is consistent, so commands are the same on all platforms.

🧰 4. Supports All Azure Services

- Almost every service in Azure (VMs, Storage, AKS, App Services, Key Vault, etc.) can be managed via CLI.
- Keep infrastructure **as code** using CLI commands in scripts.

🧪 5. Faster Testing & Troubleshooting

- Quickly spin up resources for testing.
- Retrieve logs, metrics, and resource states directly in terminal.

📁 6. Simplified Resource Management

- Easier to create/delete/update resources using az commands compared to navigating the Azure Portal.

🌐 7. Ideal for Headless/Remote Environments

- When using a **remote VM** or **cloud shell**, GUI may not be available — CLI works everywhere.

👷 Use Case Examples

Task	CLI Command Example
Create Resource Group	az group create --name myRG --location eastus
Launch VM	az vm create ...
Setup Kubernetes Cluster (AKS)	az aks create ...
Deploy App	az webapp up --name myapp ...
Automate Backup or Scale Operations	az vm backup ... or az vm scale .

Azure Resource Manager (ARM) Templates: Bhai, Azure Resource Manager (ARM) Templates ek tarika hai Azure resources ko infrastructure as code (IaC) ke through manage karne ka. Matlab tu code likh ke hi pura infrastructure bana sakta hai — bina Azure portal khole.

🧠 Kya hota hai ARM Template?

ARM Template ek JSON file hoti hai jo batati hai:

- Kaunse resources banane hain (VM, storage, VNet, etc.)
- Kis location pe banane hain
- Kis configuration ke saath banane hain

Note: **Bicep** is a **domain-specific language (DSL)** developed by Microsoft to simplify the process of deploying Azure resources.

🔍 Bicep vs ARM Templates

Feature	Bicep	ARM Templates
Language	Declarative DSL	JSON
Readability	High (clean, concise syntax)	Low (verbose and complex)
Tooling	Great IntelliSense support in VS Code	Less developer-friendly
Compilation	Compiled into ARM templates	Native JSON
Looping/Modular	Easier and more intuitive	Possible, but complicated

ARM Template for Storage Account:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
"contentVersion": "1.0.0.0",
 "parameters": {
  "storageAccountName": {
    "type": "string"
   }
 },
 "resources": [
 {
   "type": "Microsoft.Storage/storageAccounts",
  "apiVersion": "2021-09-01",
   "name": "[parameters('storageAccountName')]",
   "location": "[resourceGroup().location]",
   "sku":
{
   "name": "Standard_LRS"
        },
  "kind": "StorageV2",
  "properties": {}
  }
 ]
}
```

🔧 Deploy Command (Azure CLI)

```
az deployment group create \

 --resource-group myResourceGroup \

 --template-file simple-storage.json \

 --parameters storageAccountName=mystorageanuj123
```

📄 Full Explanation of the Storage ARM Template

```
"$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
```

🔸 **$schema**:
Yeh batata hai ki yeh JSON kis template standard ko follow karta hai.
Azure use karta hai isse validate karne ke liye ki tera format sahi hai ya nahi.

```
"contentVersion": "1.0.0.0",
```

🔸 **contentVersion**:
Template ka version hai. Tera versioning track rakhne ke kaam aata hai.
Agar multiple versions bana raha ho toh helpful hota hai.

```
 "parameters": {

  "storageAccountName": {

   "type": "string"

  }

 }
```

🔸 **parameters**:
Yahan user se input liya jata hai.
Is case mein, user storageAccountName input karega jab template deploy karega.

```
--parameters storageAccountName=mystorageanuj123
```

```
 "resources": [

  {

   "type": "Microsoft.Storage/storageAccounts",
```

🔸 **type**:
Yeh batata hai ki kis type ka Azure resource banana hai.
Microsoft.Storage/storageAccounts = storage account banana hai.

```
"apiVersion": "2021-09-01",
```

- **apiVersion**:

Yeh version Azure API ka hai jisko use karke resource create hoga.
(Always try to use latest version unless you have a reason.)

```
"name": "[parameters('storageAccountName')]",
```

- **name**:

Yeh storage account ka naam hai jo user ne parameters mein diya hoga.

```
"location": "[resourceGroup().location]",
```

- **location**:

Yeh location us resource group ki location le leta hai jahan tu deploy kar raha hai.
Matlab user ko manually location dene ki zarurat nahi.

```
"sku": {
  "name": "Standard_LRS"
},
```

- **sku.name**:

SKU ka matlab hai pricing tier.
Standard_LRS = Locally Redundant Storage, sasta aur reliable hai.

```
"kind": "StorageV2",
```

- **kind**:

Storage ka type define karta hai.
StorageV2 = latest and recommended kind, support karta hai blob, file, queue, table etc.

```
"properties": {}
```

- **properties**:

Khaali chhoda gaya hai. Agar tujhe kuch extra setting add karni ho (like encryption, accessTier), toh yahaan likhta.

Access tier (Hot, Cool):

Access Tier ka matlab hota hai: "Tera data kitni frequently access hoga aur uske hisaab se pricing aur speed kaise honi chahiye."

Azure Storage mein mainly 2 Access Tiers hote hain:

🔥 **1. Hot Tier**

- Use case: Jab data frequently access hota hai (roz ka kaam jaise images, logs, reports).
- Read/Write cost: Low
- Storage cost: High
- Speed: Fast access

📦 **Example:**
Website ki daily images, active project data, real-time logs.

🧊 **2. Cool Tier**

- Use case: Jab data kam access hota hai (once in a while, backup ya archive).
- Read/Write cost: High
- Storage cost: Low
- Speed: Thoda slower hota hai Hot tier se.

📦 **Example:**
Old reports, old logs, backup data jo kabhi hi chahiye hota hai.

📝 **Storage Account Banate Time Set Kar Sakte Ho:**

ARM Template mein aise likhte hain:

```
"properties": {

 "accessTier": "Hot"

}
```

Ya phir:

```
"accessTier": "Cool"
```

Azure IAM (dentity and Access Management): Yeh Azure ka ek system hai jo control karta hai:

- Kaun Azure resources ko access kar sakta hai

  Kya access kar sakta hai
- Kitna access hai (read, write, delete, etc.)

🔐 IAM Azure Mein Kya Karta Hai?

🔹 1. Authentication

Kaun ho tum? (Login karne wale ki pehchaan)

- Azure AD ke users, apps, groups ya external users

🔹 2. Authorization

Tumhe kya karne ka right hai? (Permission control)

- Read, Write, Contributor, Owner, etc.

✅ Roles

Azure pre-defined roles deta hai:

Role	Kya kar sakta hai
Owner	Full control (RBAC + permissions)
Contributor	Create/Edit kar sakta hai, delete bhi
Reader	Sirf read kar sakta hai
Storage Blob Data Reader	Blob data dekh sakta hai

🔐 Microsoft Entra ID (formerly Azure Active Directory)

Ye ek identity and access management (IAM) service hai jo:

- Users ko authenticate karta hai (login check)
- Access control deta hai Azure, Microsoft 365, aur custom apps ke liye
- Single Sign-On (SSO), MFA, RBAC jaisi cheeze handle karta hai

🔑 Kaam Kya Karta Hai?

Feature	Description
🔑 Authentication	Users, apps, devices ko login karwata hai securely
🔐 Authorization	Kis resource pe kya karne ka permission hai (RBAC)
👤 User Management	Users, groups, guest users handle karta hai
🔁 SSO (Single Sign-On)	Ek hi login se multiple apps access
📱 MFA	Multi-Factor Authentication ke options deta hai
📜 Audit Logs	Kaun login hua, kya kiya — sabka record
🔄 Hybrid Identity	On-prem AD + Azure sync karne ke options

Service Principal aur **Managed Identity** — dono Azure mein automation aur app access ke liye use hote hain, lekin dono ka purpose aur handling alag hai.

👨‍💼 1. What is a Service Principal?

Ek **Service Principal (SP)** ek **identity hoti hai kisi app ke liye** jo Azure mein login kar sakti hai — jaise ek robot user.

🛠 Use Case:

- Tu jab Jenkins, GitHub, Terraform ya Ansible se Azure ke resources manage karta hai — tu ek SP banata hai jisko permissions de sakta hai.

🔑 SP ke pass hota hai:

- **Client ID** (username jaisa)
- **Client Secret** (password jaisa)
- **Tenant ID**
- **Subscription ID**

🔐 Example:

```
az ad sp create-for-rbac --name "myApp" --role contributor
```

🤖 2. What is a Managed Identity?

Azure ka **built-in way** hai to give identity to Azure resources (VM, Function, Logic App, etc.) **without passwords or secrets**.

🛠 Use Case:

- Jaise tu Azure VM se Storage access karna chahta hai — to VM ke liye Managed Identity enable karke permission de.
- No secret management needed — Azure khud handle karta hai.

🧩 Types:

Type	Use
System-assigned	Ek resource ke sath banta, delete hote hi identity bhi delete
User-assigned	Reusable identity, multiple resources me use ho sakta hai

🔧 Enable Karna:

```
az identity create --name myID --resource-group myRG
```

🔄 Service Principal vs Managed Identity

Feature	Service Principal	Managed Identity
Secret rotation	Manual (needs automation)	Automatic by Azure
Setup	Manual	One-click in portal or CLI
Use with external apps	✅ Yes (Jenkins, GitHub, Terraform etc.)	❌ No (only Azure internal apps)
Resource binding	Not linked to Azure resource	Linked to Azure resource (VM, Function etc.)
Best for	External CI/CD tools	Internal Azure service access

🎯 Kab kya use kare?

Situation	Use
Terraform se Azure infra banani hai	**Service Principal**
Azure VM ko Storage ya Key Vault access chahiye	**Managed Identity**
GitHub Actions se Azure deploy karna hai	**Service Principal**
Azure Function ko DB access dena hai	**Managed Identity**

Azure DevOps

Azure DevOps ek **DevOps toolchain** hai jo **Microsoft** ka product hai. Ye developers aur DevOps teams ko **planning**, **development**, **testing**, aur **deployment** process automate karne aur manage karne mein help karta hai. Ye ek SaaS (Software as a Service) platform hai jo end-to-end DevOps lifecycle cover karta hai.

🔧 Azure DevOps ke Major Components (Services):

Ye 5 main parts ya services provide karta hai:

1. Azure Repos

- Git repo milta hai jahan tum apna code rakhte ho.
- Pull request, branch policies, aur code review ka support milta hai.

2. Azure Pipelines

- **CI/CD pipelines** banane ke liye use hota hai.
- Tum apna code build, test aur deploy kar sakte ho automatically.
- Linux, Windows, macOS — sabko support karta hai.
- Docker aur Kubernetes ke sath bhi integrate hota hai.

3. Azure Boards

- Ye ek project management tool hai.
- Jira jaisa hi hai — jisme tum tasks, bugs, user stories track kar sakte ho.
- Sprint planning, Kanban boards, backlog etc. sab milega.

4. Azure Test Plans

- Manual aur exploratory testing ke liye use hota hai.
- Test cases banakar assign kar sakte ho QA team ko.

5. Azure Artifacts

- NuGet, npm, Maven jaise packages ko manage karne ke liye.
- Tumhare build ke packages ko store kar sakta hai securely.

Azure Kubernetes Service: Yeh Microsoft Azure ka managed Kubernetes service hai, jisme tu bina headache ke apne containerized apps ko deploy, manage aur scale kar sakta hai.

AKS Kya Hai?

Azure AKS ek **Managed Kubernetes Cluster** hai. Matlab:

- Azure tere liye **control plane (master nodes)** ko manage karta hai (free of cost).
- Tu sirf **worker nodes** (jaha tera actual app chalega) ka paisa deta hai.
- Deployment, scaling, monitoring sab kaafi easy ho jata hai.

Feature	Fayda
Managed Service	Master node ka load Azure uthata
Scaling	Automatically scale ho sakta hai
Security	Azure AD, RBAC, network policies support karta hai
Integration	Azure Monitor, Log Analytics, DevOps pipelines ke saath aasani se integrate hota hai
CI/CD Friendly	GitHub Actions, Azure DevOps, ArgoCD easily integrate ho jata

AKS vs Self-Managed Kubernetes

🔥 **1. Cluster Setup aur Management**

Feature	AKS (Azure Kubernetes Service)	Self-Managed Kubernetes
Setup	Azure khud setup karta hai – control plane, networking sab kuch	Khud sab kuch setup karna padta hai (kubeadm, kubelet, kube-proxy, etc.)
Control Plane	Azure manage karta hai – tu dekh bhi nahi sakta	Khud manage karta hai (Master node setup + etcd + kube-apiserver, etc.)
Maintenance	Azure handle karta hai (patching, updates)	Tu khud updates, monitoring aur backup karega

🔥 **2. Cost (Paison ki baat hai bhai 😅)**

Feature	AKS	Self-Managed
Master Node	Free (Azure khud manage karta hai)	Tere master nodes bhi tere hi infra pe chalenge, unka bhi paisa lagega

Feature	AKS	Self-Managed
Worker Node	Tu khud choose karta hai – uska paisa lagta hai	Same, but tu cloud/VM bare metal kuch bhi use kar sakta hai

🔥 3. Scaling & Upgrades

Feature	AKS	Self-Managed
Auto-scaling	Built-in node pool scaling + HPA supported	Khud implement karna padta hai (cluster autoscaler + HPA setup)
Upgrades	Azure CLI se 1 command: az aks upgrade	Khud rolling update plan karna padta hai, risky bhi ho sakta hai

🔥 4. Security & Monitoring

Feature	AKS	Self-Managed
Security	Azure AD, RBAC, Azure Policy se easy integration	Khud AD ya LDAP ya RBAC setup karna padega
Monitoring	Azure Monitor + Log Analytics	Tu Prometheus + Grafana ya 3rd party tools lagayega manually

🔥 5. Learning & Flexibility

Feature	AKS	Self-Managed
Learning Curve	Easy to medium (Kubernetes + Azure ecosystem)	Tough (Pure Kubernetes from scratch + infra ka knowledge bhi chahiye)
Flexibility	Azure ke ecosystem mein bound rahega	Full control – jo chahe install kar, jaise chahe run kar

🔧 Problem without AKS:

Agar tu khud se Kubernetes cluster setup karega (jo kaafi log self-managed karte hain), toh tujhe har cheez ka khayal rakhna padta hai:

- Cluster installation
- Node setup
- Load balancing
- Updates
- Monitoring
- Security patches

Create effective Monitoring System on Azure

🔍 Monitoring Kya Hai?

Monitoring ka matlab hota hai system, application, aur infrastructure ki health aur performance ko continuously dekhna — taki agar koi dikkat aaye, to turant pata chal jaye.

🤖 Monitoring se kya pata chalta hai?

- Server ka CPU, Memory, Disk usage
- Application ka response time aur error rate
- Website up hai ya down?
- Network latency ya failure
- Log files mein kuch unusual activity
- Custom business metrics (jaise: kitne log login kar rahe hain)

🛠️ Common Monitoring Tools:

Tool Name	Use Case
Prometheus (There is Azure managed Prometheus also)	Metrics collect karne ke liye (especially infra)
Grafana	Visualize karne ke liye (dashboards)
ELK Stack	Logs monitor karne ke liye
Datadog	SaaS-based all-in-one tool
Nagios	Old-school monitoring tool
New Relic	App performance monitoring (APM)
Zabbix	Open-source infra monitoring
Azure Monitor	Azure cloud ka native tool
CloudWatch	AWS ka native tool

🔄 Types of Monitoring:

1. **Infrastructure Monitoring** – Servers, VMs, CPU, Memory
2. **Application Monitoring (APM)** – App ka performance
3. **Log Monitoring** – Log files mein kya chal raha
4. **Network Monitoring** – Traffic, packet loss, speed
5. **User Monitoring –** End-user experience

📈 Why It's Important?

- Downtime ka pata turant chal jata hai
- Root cause analysis easy hota hai
- Performance tuning mein help karta hai
- Alerts set kar sakte ho (email, Slack, SMS)
- Business reputation bachi rehti hai

Azure Monitor: Azure Monitor ek built-in monitoring service hai Microsoft Azure ka, jo tujhe teri cloud-based (aur on-prem) applications, infrastructure, aur network ka health, performance, aur logs track karne mein madad karta hai.

💡 Azure Monitor Kya Karta Hai?

- **Data collect** karta hai (metrics, logs, events)
- **Visualize** karta hai (dashboards, charts)
- **Alert** bhejta hai jab koi issue hota hai
- **Automate** kar deta hai response (jaise auto-scale, restart, etc.)

📊 Kya-Kya Monitor Kar Sakta Hai?

Monitoring Type	Example
Infra Monitoring	VM ka CPU, RAM, Disk
Application Monitoring	Web App response time, errors
Container Monitoring	AKS pods ki health
Network Monitoring	Traffic, latency
Custom Metrics	Apne business metrics (login counts, etc.)

🔗 Azure Monitor Ke Components:

Component	Kya Karta Hai?
Metrics	Numeric data (jaise CPU%, Memory%)
Logs (Log Analytics)	Detailed event info, queries likh ke data nikal sakte ho
Alerts	Jab threshold cross ho to notification mile
Dashboards	Charts/Graphs banake sab visualize karo
Application Insights	App ka performance monitor karta hai
Network Watcher	Network traffic aur connectivity ka scene
Azure Monitor Agent	Data collect karne wala agent jo VM pe install hota hai

🛠️ Azure Monitor Use Karne Ka Flow:

1. **Enable Monitoring** on resource (like VM, AKS, App Service)
2. **Data Collect** hota hai automatically (via agent or built-in telemetry)
3. Use **Log Analytics** to write queries (KQL – Kusto Query Language)
4. Setup **Alerts** (email, SMS, Teams, etc.)
5. Create **Dashboard** for visualization

📘 Example Use Case:

Tu chah raha hai ki agar VM ka CPU 80% se upar chala jaye to alert aaye:

- Metric Alert banayega on Percentage CPU
- Condition: > 80%
- Action Group: Email ya Teams message bhejna

🔥 Bonus Tip:

Agar tu **Application Insights** bhi saath use karega (jo Azure Monitor ka part hai), to tu trace kar sakta hai:

- Kaunse API slow hai
- Kaunse requests fail ho rahe
- Kaunse region se zyada traffic aa raha

🔐 Azure Key Vault – Kya hai?

Azure Key Vault ek cloud service hai Microsoft Azure ki taraf se, jo secrets (jaise passwords, API keys, certificates, connection strings, etc.) ko securely store aur access karne ke liye use hoti hai. Azure Key Vault ek **locker** ki tarah hai cloud mein, jisme sensitive information ko **safe aur encrypted** rakhte ho.

🔧 Isme kya-kya store kar sakte ho?

1. **Secrets**: passwords, API keys, connection strings
2. **Keys**: encryption keys (jaise RSA, HSM-based keys)
3. **Certificates**: SSL certificates
4. **Secrets rotation**: automatically update passwords etc.

🎯 Real-life Example:

Tere paas ek web app hai jo SQL DB use karta hai. Instead of DB password code mein likhne ke, tu password ko Key Vault mein store karta hai Aur app waha se fetch karta hai securely runtime pe.

Bhai, AWS mein **Azure Key Vault** jaisa service hai:

🔑 **AWS Secrets Manager**
🔐 **AWS KMS (Key Management Service)**

Azure AD (Active Directory) kya hai?: Ye Microsoft ki **identity and access management (IAM)** service hai. Matlab: Users, groups, permissions, login access sab handle karne ke liye use hoti hai.

🤓 **Simple words mein:**

Samajh le tera office ka ek login system hai – jaha sab employees apne **username-password** se login karte hain, apps ka access milta hai, roles assign hote hain.
Ye poora system **Azure Active Directory (Azure AD)** handle karta hai.

🛠️ Azure AD kya-kya karta hai?

Feature	Kya karta hai
👤 **User Management**	Users create, delete, manage kar sakte ho
🔐 **Authentication**	Login/Logout, MFA, password reset
🔑 **Authorization**	Kis user ko kya access milega
🌐 **SSO (Single Sign-On)**	Ek baar login, multiple apps ka access

🎯 **Real Life Example:**

Tere paas ek app hai jo Azure pe host hai.
Tu chaahta hai ki sirf tera team login kare app mein.
Tu Azure AD mein users ko assign karega, unko roles dega, aur app ko secure kar lega.

💡 **Bonus:**

Azure Key Vault ko bhi secure karne ke liye **Azure AD** use hota hai.
Matlab: Kaun secret access karega, wo Azure AD decide karta hai.

Serverless using Azure Functions

Azure Serverless ek aisa model hai jahan tumhe servers manage karne ki tension nahi hoti. Tum sirf apna code likhte ho, aur Azure usko automatically run karta hai jab zarurat ho — pay only for what you use.

🔧 **Azure Serverless ke Key Services:**

1. **Azure Functions –** Event-driven code likhne ke liye (jaise Lambda in AWS)
2. **Azure Logic Apps –** Low-code workflows banane ke liye
3. **Azure Event Grid –** Events ko route karne ke liye
4. **Azure Service Bus** – Messaging service for decoupled communication
5. **Azure Container Apps –** Lightweight container-based serverless app deployment

✅ **Use Cases of Azure Serverless:**

Use Case	Explanation
1. Real-time File Processing	Jab bhi koi file Azure Blob Storage mein upload hoti hai, Azure Function usse turant process kar leta hai (e.g., image resize, virus scan).
2. API Backend	Serverless APIs bana sakte ho Azure Functions ke through, bina full backend setup kiye.
3. Scheduled Jobs / Cron	Jaise daily report generation, backups, ya data sync kaam schedule karke automatically chalana.

Use Case	Explanation
4. IoT Data Processing	IoT devices se data aata hai, usse Azure Event Hub + Azure Functions ke through process karke database mein store karna.
5. Chatbot or Voice Bot	Logic Apps + Cognitive Services se ek smart bot banaya ja sakta hai bina infrastructure manage kiye.
6. CI/CD Hooks	GitHub ya Azure DevOps ke events pe automatic validation/test/deploy triggers.
7. Lightweight Machine Learning Inference	Trained model ko Azure Function mein use karke predictions dena (e.g., sentiment analysis, fraud detection).

⚡ **Serverless Ka Faayda:**

- No server management (Azure sab handle karta hai)
- Auto-scaling (1 user ho ya 1 million, sab handle hota hai)
- Cost-effective (idle time ka paisa nahi lagta)
- Rapid development (microservice style development easily ho jata hai)

💥 **Disadvantages of Azure Serverless:**

❌ Problem	😬 Detail
1. Cold Start Time	Jab koi Azure Function kaafi time tak idle raha ho, aur fir trigger hota hai — toh usme **delay** hota hai (called **cold start**). Especially noticeable in high latency-sensitive apps.
2. Limited Execution Time	Azure Functions ka ek timeout hota hai (default: 5 mins for Consumption Plan). Long-running tasks ke liye best option nahi.
3. Debugging is Hard	Serverless environment ka debugging local ya traditional apps ke comparison mein thoda tricky hota hai.
4. Vendor Lock-in	Azure ke serverless ecosystem mein ghusne ke baad switch karna mushkil ho sakta hai (AWS ya GCP mein port karne mein mehnat lagegi).
5. Limited Control over Infrastructure	Serverless mein tum infra control nahi kar sakte. Agar tumhe memory size, CPU config, ya VMs ka tight control chahiye — toh yeh option limited hai.
6. Cold Start in Private VNet	Agar function kisi VNet ke andar hai (jaise private DB access ke liye), toh cold start aur bhi slow ho sakta hai.
7. Monitoring & Troubleshooting Overhead	Proper observability ke liye tumhe **Application Insights** ya external monitoring tools integrate karne padte hain. Default logs kabhi confuse kar dete hain.
8. Stateful Applications	Serverless architecture stateless hota hai. Agar app ko continuously state ya session maintain karna hai, toh kaafi jugad karni padti hai (Redis, DB etc.).

🧊 **Cold Start :** Jb tumhara Azure Function kaafi time tak idle (yaani koi use nahi hua) rehta hai, toh Azure usko sleep mode mein daal deta hai — taa ki cost kam ho.
Ab jab next baar koi request aati hai, toh pehle Azure ko function ko dubara wake-up karna padta hai, uska environment ready karna padta hai. Yeh process 2-10 seconds ya zyada bhi lag sakti hai = delay in response

🔥 **Warm Start :** Jab tumhara Azure Function pehle se hi active hai — ya recently koi request aayi thi — toh Azure ne uska environment shutdown nahi kiya hota.
Isliye jab naye request aati hai, toh directly fast response milta hai — bina kisi initialization delay ke.

☠️ **Dead State – Terminated:** Agar function ko bahut der tak koi hit nahi karta Azure ne usko terminate kar diya
Next request pe cold start lagega

⚡ Event-Driven Serverless Kya Hota Hai?

"Event-driven serverless" ka matlab hai: Jab **koi event hota hai**, tab hi tumhara **serverless code (function)** trigger hota hai — **on-demand**.

Tujhe **server setup nahi karna**, bas event aaya, function chala, kaam ho gaya ✅

🔔 Common Events:

Event Type	Example
HTTP Request	User ne API call ki
Blob Created	Koi file upload hui storage mein
Timer Trigger	Roz subah 8 baje function chalu ho
Queue Message	Queue mein koi message aaya
Event Grid	Resource create/update hua Azure mein
Database Trigger	Cosmos DB/SQL mein data change hua

🧠 ***Samjho Ek Example:***

Scenario:

- Tumhara ek Azure Function bana hua hai.
- Koi banda **image upload** karta hai Azure Blob Storage mein.
- **Event**: Blob Created
- Tumhara serverless function turant **trigger hota hai**.
- Wo image ko **resize** karta hai, phir compress karke **dusre blob container** mein daal deta hai.

Pure process mein tumne **koi server manage nahi kiya**, bas event aaya, kaam ho gaya. ✨

✅ Day-to-Day Activities of a DevOps Engineer

💬 "As a DevOps Engineer, my daily activities revolve around ensuring smooth CI/CD processes, infrastructure automation, and system reliability."

🌅 Morning (Planning & Coordination)

- Attend **daily stand-ups** to sync with developers, QA & PM teams.
- Check **overnight pipeline builds**, incident alerts, or failed deployments.
- Pick tasks from **JIRA board** and set goals for the day.

🔧 CI/CD Pipeline Management

- Maintain and enhance **CI/CD pipelines** using Jenkins/GitHub Actions.
- Integrate **unit tests, security scans (SonarQube), linting, and container builds**.
- Ensure **zero-downtime deployments** via blue-green or rolling updates.

☁️ Infrastructure Automation

- Use **Terraform/CloudFormation** to provision or update cloud infra (AWS/Azure).
- Manage **Kubernetes clusters**, perform pod monitoring and log checks.
- Maintain Dockerfiles, Helm charts, and **image versioning**.

🔍 Monitoring & Troubleshooting

- Monitor logs and dashboards (**Grafana, Prometheus, ELK**) for performance issues.
- Investigate infra-related bugs or **deployment failures**.
- Implement alerting via **Alertmanager or CloudWatch**.

🤝 Collaboration & Documentation

- Assist developers with **environment setup**, secrets, or config files.
- Write or update internal **runbooks, wikis, and postmortems**.
- Participate in **incident reviews** and suggest automation opportunities.

📝 JIRA Tickets – DevOps/Release Engineer ke Kaam Mein Kyu Zaroori Hai?

✅ Kab Tickets Raise/Handle Karte Hain:

1. **Infra Change Requests**
   Jaise: "Need to increase CPU/memory for a pod"
   ➤ Ticket banta hai → approval milta hai → fir apply karte hain.
2. **Deployment Requests**
   Jaise: "Release v1.2.3 to staging"
   ➤ PM ya QA raise karega → tu assign leke deploy karega → status update karega.
3. **Bug Fix Coordination**
   Agar deployment ke baad koi issue aaya, tu ticket raise karega with logs & analysis.

4. **Automation Task Tracking**
   Jaise: “Create a script for log rotation”
   ➤ Tere backlog mein ye ticket aayega, tu resolve karke close karega.

5. **Incident Reporting / RCA (Root Cause Analysis)**
   Jaise: “Site was down for 10 mins due to failed DB connection”
   ➤ Incident ticket banta hai, tu RCA likhega aur preventive steps suggest karega.

🔥 **Interview Mein Aise Bol Sakta Hai:**

“Yes, I actively work on JIRA — I raise tickets for infra changes, track automation tasks, and also update deployment tickets during releases. Post-deployment, I document RCA tickets in case of failures and link them with corresponding PRs or alerts.”

✅ AI in DevOps – Real-World Use Cases

🔹 1. Anomaly Detection in Monitoring

🔍 Problem: Log ya metrics ke through manual error detection time-consuming hota hai.
🤖 Solution: AI/ML models abnormal CPU, memory usage, traffic spikes detect kar lete hain.

Example:

- Tools: **Dynatrace, DataDog, Splunk AIOps**
- Use Case: “Automatically detect sudden drop in API response time and notify teams with RCA suggestions.”

🔹 2. Predictive Auto-Scaling

🔍 Problem: Traffic spike ke time pe system slow ho jaata hai.
🤖 Solution: AI past usage data se predict karta hai aur proactively scale up/down karta hai.

Example:

- AWS Auto Scaling with AI-based predictions
- Kubernetes + KEDA + AI models

🔹 3. Automated Incident Management (AIOps)

🔍 Problem: Incidents ko handle karne mein time lagta hai.
🤖 Solution: AI root cause analyze karke suggest karta hai kya fix karna hai.

Example:

- Tools: **Moogsoft, BigPanda, Opsgenie + ML**
- “500 error detect hua, AI ne logs analyze kiye aur suggest kiya DB connection timeout hua.”

🔹 4. Code Review & Quality Gate using AI

🔍 Problem: Manual code review mein time aur bias hota hai.
🤖 Solution: AI tools code ko scan karke vulnerability, duplication aur formatting errors batate hain.

Example:

- Tools: **Codacy, Amazon CodeGuru, DeepCode**
- Pull request ke sath automatic AI-based suggestions milte hain.

🔹 5. Intelligent Test Case Generation & Prioritization

🔍 Problem: Manual test case selection inefficient hoti hai.
🤖 Solution: AI most impacted code areas ko detect karke relevant test cases run karta hai.

Example:

- Tools: **Testim.io, Functionize, Launchable**
- Regression suite mein AI sirf wahi tests run karta hai jo naye code se impacted hain.

🔹 6. ChatOps with AI

🤖 AI chatbots like **OpsGPT**, **Azure Copilot**, ya **Slack bots** help karte hain:

- Build trigger karna
- Logs fetch karna
- Deployment status dena

🔹 7. Security Threat Detection

🔍 Problem: Real-time attack ya vulnerability detect karna mushkil hota hai.
🤖 Solution: AI-based threat detection tools continuously scan karte hain and alert dete hain.

Example:

- Tools: **Darktrace, IBM QRadar, AWS GuardDuty**