AZURE CORPORATE TRAINING
TRAINER: VISHWANATH GOWDA H

Overall Benefits of This Training

- **End-to-End Azure Mastery → Covers infra, networking, security, PaaS, DevOps, data, and monitoring.**
- **Hands-On First Approach → Each topic includes live scenarios + labs for real skill-building.**
- **Progressive Learning Path → Moves from Basic → Intermediate → Advanced with real-time outcomes.**
- **Enterprise-Ready Skills → Learners gain knowledge aligned with Fortune 500 cloud practices.**
- **Identity & Governance Clarity → Strong foundation in subscriptions, RBAC, policies, compliance.**
- **Security by Design → Covers Key Vault, RBAC, WAF, encryption, zero-trust networking.**
- **Cloud-Native App Development → App Services, Functions, Service Bus, Event Hubs, API Management.**
- **DevOps + Automation → End-to-end DevOps pipelines, IaC, GitOps, CI/CD, and security scanning.**
- **Data & AI Integration → Azure SQL, Databricks, Event Hubs for analytics and ML workloads.**
- **Modern Work Enablement → AVD, hybrid networking, multi-cloud monitoring with Prometheus + Grafana.**
- **Disaster Recovery & Resilience → Snapshots, backups, geo-replication, failover groups, DR strategies.**
- **Cost Optimization Mindset → Covers Spot VMs, lifecycle rules, tagging, budget alerts, scaling.**
- **Multi-Cloud Awareness → Prometheus + Grafana and enterprise networking show cross-cloud integration.**
- **Industry Alignment → Content maps to job roles (Cloud Architect, DevOps Engineer, Security Engineer, Data Engineer).**
- **Corporate Relevance → Every service is tied to real-world enterprise scenarios (finance, healthcare, retail, remote workforce).**
- **Career Boost → Prepares for Microsoft certifications (AZ-104, AZ-305, AZ-400, DP-203, SC-100)**

---

1. Subscriptions & Resource Groups

◈ Concept

In Azure, everything is organized by **scopes**:

- **Management Group → Subscription → Resource Group → Resources**. Subscriptions define billing boundaries. Resource Groups (RGs) are logical containers for resources like VMs, Storage, etc.

◈ Why are we doing this?

- To **organize resources** logically (per department, project, or environment).
- To **control access** at the right level (subscription vs RG).
- To **optimize billing & governance** for enterprises.

◈ Tools

- **Azure Portal** (visual management)
- **Azure CLI** (az group create, az account list)
- **PowerShell** (New-AzResourceGroup)
- **Cost Management + Azure Policy** (for governance & billing)

◈ Useful in Real Time

- Large companies run **multiple subscriptions** for regions/departments.
- Resource Groups help in **lifecycle management** → delete RG to remove all resources in one go.
- Tags and policies enable **cost allocation** across teams.

□ Basic Level

Scenario 1: Organizing by Department

- **Story:** A company has HR, Finance, and IT teams. Each subscription is mapped to a department. Finance creates a Payroll-RG for salary-related VMs.
- **Lab:** Create a new RG Payroll-RG, deploy 1 VM inside. Delete the RG and observe all resources deleted.
- **Outcome:** Learners see how RGs simplify cleanup & organization.

Scenario 2: Separate Environments

- **Story:** Developers need separate environments for Dev & Prod. Create 2 RGs: Dev-RG and Prod-RG.

- **Lab:** Deploy a VM in each RG, show clear separation in portal.
- **Outcome:** Learners understand environment isolation using RGs.

□ Intermediate Level

Scenario 1: RBAC at Different Scopes

- **Story:** HR team should manage payroll VM but not production DB. Assign Contributor at RG scope, Reader at Subscription scope.
- **Lab:** Create user hruser@company.com, assign roles, test access.
- **Outcome:** Learners practice fine-grained RBAC.

Scenario 2: Governance with Management Groups

- **Story:** Org has 3 subscriptions → Dev, Test, Prod. Add them under a Management Group for centralized governance.
- **Lab:** Create Management Group, link subscriptions.
- **Outcome:** Learners see consolidated control at scale.

Scenario 3: Tagging & Cost Analysis

- **Story:** Apply tags Dept=Finance, Env=Prod. Run cost analysis by tag.
- **Lab:** Add tags to RGs & resources. Use Cost Analysis in portal.
- **Outcome:** Learners understand tracking cost per department/project.

⬤ Advanced Level

Scenario 1: Multi-Subscription Billing Strategy

- **Story:** A multinational org has 10 subscriptions in different regions. Consolidate billing to one invoice.
- **Lab:** Use Cost Management to link subscriptions & analyze combined billing.
- **Outcome:** Learners see how enterprises simplify cost visibility.

Scenario 2: Azure Policy Enforcement

- **Story:** Company enforces that all resources must be created in East US. Any other region is blocked.
- **Lab:** Create a policy restricting region, assign it at subscription. Try creating resource in West US.
- **Outcome:** Learners understand compliance enforcement.

Scenario 3: Resource Locks

- **Story:** A critical Production RG should not be deleted. Add a Delete Lock.
- **Lab:** Apply lock, try deleting RG → fails.

- **Outcome:** Learners learn protection against accidental deletion.

✅ **Final Outcomes for This Module**

- Understand scopes (Subscription → RG → Resource).
- Assign RBAC roles at correct levels.
- Apply tags, locks, and policies for governance.
- Manage billing strategies in enterprise setups.

2. Entra ID (Azure Active Directory)

🔹 **Concept**

Microsoft Entra ID (Azure AD) is Azure's identity and access management service. It controls authentication, authorization, and secure access to apps/resources.

🔹 **Why we are doing this**

- Manage **users, groups, roles** across cloud resources.
- Enforce **security (MFA, Conditional Access)**.
- Enable **SSO & B2B collaboration**.

🔹 **Tools**

- Azure Portal → Identity blade
- Azure AD PowerShell / MS Graph
- Azure CLI → az ad user create
- Conditional Access & PIM

🔹 **Useful in Real Time**

- Centralized identity for employees, partners, apps.
- Secure login policies (MFA, passwordless).
- Lifecycle management for enterprise users.

 Basic (2 scenarios)

1. **Create a User & Assign Role**
    - Lab: Create user john@company.com, assign Reader at RG level. Login test.
    - Outcome: Learners see account provisioning & RBAC in action.
2. **Group-Based Access**

- Lab: Create “DevTeam” group, add users, assign Contributor role at RG.
- Outcome: Learners learn to manage at scale using groups.

□ Intermediate (3 scenarios)

1. **Conditional Access Policy (MFA)**
   - Lab: Enforce MFA for Finance users. Try login → MFA prompt.
   - Outcome: Learners practice securing logins.
2. **Privileged Identity Management (PIM)**
   - Lab: Activate Global Admin role for 1 hour. Show time-bound permissions.
   - Outcome: Learners understand least privilege access.
3. **Passwordless Authentication**
   - Lab: Enable Authenticator app login. Sign in using phone.
   - Outcome: Learners experience modern passwordless login.

● Advanced (3 scenarios)

1. **B2B Guest Access**
   - Lab: Invite Gmail user to tenant. Assign app access.
   - Outcome: Learners know external collaboration setup.
2. **Identity Protection**
   - Lab: Create risky login policy. Simulate suspicious login attempt.
   - Outcome: Learners understand risk-based access.
3. **Automated Provisioning**
   - Lab: Connect Azure AD with SaaS app (ServiceNow). Auto-provision/de-provision user.
   - Outcome: Learners see lifecycle automation.

✅ Final Outcomes: Learners can **manage identities, secure access, enable collaboration, and enforce governance**.

---

3. Networking (VNets, Subnets, NSGs, Peering)

◈ Concept

Networking is the foundation of Azure architecture. VNets provide isolation, Subnets divide workloads, NSGs secure traffic, and Peering connects networks.

◈ Why we are doing this

- To design secure, scalable network topologies.
- To control inbound/outbound traffic.

- To connect workloads across regions.

🔹 Tools

- Azure Portal (VNet creation)
- Azure CLI (az network vnet create)
- Azure Network Watcher (diagnostics)

🔹 Useful in Real Time

- Multi-tier apps (web subnet + DB subnet).
- Hybrid connectivity with on-prem networks.
- Global enterprise networks with Hub-Spoke.

🟢 Basic (2 scenarios)

1. **VNet with 2 Subnets**
    - Lab: Create VNet CorpVNet → Subnets Web, DB. Deploy 2 VMs, show connectivity.
    - Outcome: Learners understand subnet isolation.
2. **NSG Rule Block**
    - Lab: Apply NSG to block RDP. Try login → fails. Then allow → success.
    - Outcome: Learners see firewall effect.

🟡 Intermediate (3 scenarios)

1. **VNet Peering**
    - Lab: Peer VNet-A and VNet-B. Ping VMs across VNets.
    - Outcome: Learners connect cross-VNet workloads.
2. **VPN Gateway Hybrid**
    - Lab: Create VPN gateway. Connect to on-prem emulator (local VM).
    - Outcome: Learners experience hybrid networking.
3. **Priority NSG Rules**
    - Lab: Allow HTTP, deny SSH. Test both.
    - Outcome: Learners apply layered security.

🔴 Advanced (3 scenarios)

1. **Hub-Spoke Architecture**
    - Lab: Create Hub VNet with firewall, connect 2 Spoke VNets. Route all through hub.

- Outcome: Learners see enterprise design.

2. **Private Endpoints**
   - Lab: Secure Storage with private endpoint. Try public access → denied.
   - Outcome: Learners implement zero-trust networking.
3. **Service Endpoints**
   - Lab: Restrict SQL DB to only accept traffic from VNet.
   - Outcome: Learners secure PaaS resources.

✅ Final Outcomes: Learners can **design secure, scalable, hybrid-ready Azure networks**.

---

4. Virtual Machines (VMs)

🔹 **Concept**

VMs are Azure's IaaS compute. They simulate physical machines in the cloud.

🔹 **Why we are doing this**

- To host apps where full OS control is needed.
- To run workloads not suited for PaaS.
- For DR, testing, legacy app migration.

🔹 **Tools**

- Azure Portal
- CLI (az vm create)
- PowerShell
- Azure Backup, Monitor

🔹 **Useful in Real Time**

- Host business apps.
- Scale using VM Scale Sets.
- Backup & restore for critical workloads.

□ Basic (2 scenarios)

1. **Create & Connect VM**
   - Lab: Deploy Linux VM, SSH, install Apache, test webpage.
   - Outcome: Learners deploy & manage VM.
2. **Stop VM for Cost Savings**

- Lab: Stop VM → check billing.
- Outcome: Learners understand cost optimization.

□ Intermediate (3 scenarios)

1. **Availability Set**
   - Lab: Create 2 VMs in Availability Set. Simulate host failure.
   - Outcome: Learners understand redundancy.
2. **Scale Set**
   - Lab: Auto-scale VM when CPU > 70%. Stress test CPU.
   - Outcome: Learners see elasticity.
3. **Backup & Restore**
   - Lab: Enable backup, delete a file, restore.
   - Outcome: Learners see DR in action.

⬤ Advanced (3 scenarios)

1. **Spot VM**
   - Lab: Deploy Spot VM, simulate eviction.
   - Outcome: Learners test cost-efficient compute.
2. **Custom Image**
   - Lab: Create VM image → deploy 3 identical VMs.
   - Outcome: Learners understand standardization.
3. **Proximity Placement Group**
   - Lab: Deploy 2 VMs in PPG, test latency.
   - Outcome: Learners optimize latency-sensitive apps.

✅ Final Outcomes: Students can **deploy, scale, and optimize VMs for enterprise workloads**.

---

5. Azure Snapshots

◈ Concept

Snapshots are point-in-time backups of Azure VM disks.

◈ Why we are doing this

- Protect VMs before risky changes.
- Recover quickly from corruption.
- Clone environments.

◈ Tools

- Azure Portal → Snapshots
- CLI: az snapshot create
- Azure Backup integration

◈ Useful in Real Time

- Rollback before patching.
- Disaster recovery test.
- Creating golden image VMs.

□ Basic (2 scenarios)

1. **Snapshot & Restore VM**
    - Lab: Snapshot OS disk, delete VM, restore.
    - Outcome: Learners see quick recovery.
2. **Snapshot Before Patching**
    - Lab: Take snapshot before OS update. Rollback if failure.
    - Outcome: Learners practice safe updates.

□ Intermediate (3 scenarios)

1. **Daily Snapshot Automation**
    - Lab: Configure policy for daily snapshot.
    - Outcome: Learners see automation.
2. **Cross-Subscription Restore**
    - Lab: Copy snapshot, create VM in another subscription.
    - Outcome: Learners migrate across boundaries.
3. **Managed Disk Creation**
    - Lab: Convert snapshot → managed disk → attach to new VM.
    - Outcome: Learners repurpose snapshots.

● Advanced (3 scenarios)

1. **Geo-Replication**
    - Lab: Copy snapshot to secondary region.
    - Outcome: Learners build DR-ready infra.
2. **Integration with Backup Vault**
    - Lab: Manage snapshots centrally in Backup vault.

- Outcome: Learners unify DR strategy.

3. **Automated Cleanup**
    - Lab: Use CLI to auto-delete snapshots older than 30 days.
    - Outcome: Learners optimize storage costs.

✅ Final Outcomes: Learners can **protect, restore, and automate VM disk snapshots for DR and operations**.

6. Storage Accounts

🔹 Concept

Azure Storage Accounts provide highly available, durable, and scalable storage services: Blob, File, Queue, and Table.

🔹 Why we are doing this

- Store unstructured (Blob) or structured data (Tables).
- Host files (File shares), manage messages (Queue).
- Control redundancy (LRS, GRS, ZRS).

🔹 Tools

- Azure Portal
- Azure Storage Explorer
- CLI (az storage account create)

🔹 Useful in Real Time

- Web apps store images/videos in Blob.
- Shared drives using File Share.
- Queue messages between microservices.

🟢 Basic (2 scenarios)

1. **Upload to Blob Storage**
    - Lab: Upload invoice.pdf, generate SAS URL, access externally.
    - Outcome: Learners practice blob basics.
2. **File Share Mount**
    - Lab: Create File Share, map it to Windows VM.
    - Outcome: Learners simulate SMB-based shared storage.

🟡 Intermediate (3 scenarios)

1. **Soft Delete**

- Lab: Enable soft delete, delete a blob, restore it.
- Outcome: Learners recover data easily.

2. **Replication Choice**
    - Lab: Switch redundancy from LRS to GRS.
    - Outcome: Learners understand replication trade-offs.
3. **Queue Storage Messaging**
    - Lab: Send/receive queue messages between 2 apps.
    - Outcome: Learners connect microservices.

🔴 **Advanced (3 scenarios)**

1. **Lifecycle Management**
    - Lab: Configure rule → move files to archive after 30 days.
    - Outcome: Learners optimize storage cost.
2. **Private Endpoint**
    - Lab: Secure blob access via VNet only.
    - Outcome: Learners implement zero-trust storage.
3. **Data Lake Gen2**
    - Lab: Enable hierarchical namespace, query logs.
    - Outcome: Learners handle big data storage.

✅ Final Outcomes: Master **blob, file, queue, redundancy, security, and automation**.

---

7. Key Vault

🔹 **Concept**

Azure Key Vault securely stores **secrets, keys, and certificates**.

🔹 **Why we are doing this**

- Centralize secret management.
- Enforce key rotation policies.
- Secure applications (no hardcoded passwords).

🔹 **Tools**

- Portal → Key Vault
- CLI (az keyvault secret set)
- Managed Identity integration

🔹 **Useful in Real Time**

- Store DB passwords securely.
- Auto-fetch secrets in Function Apps.
- Manage SSL certificates for web apps.

□ Basic (2 scenarios)

1. **Store Secret**
    - Lab: Store DB password. Retrieve via portal.
    - Outcome: Learners understand secure storage.
2. **Access Secret from App**
    - Lab: Use App Service to read Key Vault secret.
    - Outcome: Learners see app-secret integration.

□ Intermediate (3 scenarios)

1. **RBAC vs. Access Policy**
    - Lab: Assign Reader vs. Secret Officer role.
    - Outcome: Learners compare access models.
2. **Key Rotation**
    - Lab: Auto-rotate key every 90 days.
    - Outcome: Learners enforce compliance.
3. **Audit Access Logs**
    - Lab: Enable logging → check who accessed secrets.
    - Outcome: Learners monitor key usage.

● Advanced (3 scenarios)

1. **Managed Identity Integration**
    - Lab: VM fetches secret without credentials.
    - Outcome: Learners avoid hardcoding credentials.
2. **HSM-protected Keys**
    - Lab: Create key in HSM mode.
    - Outcome: Learners see highest security model.
3. **Integration with AKS**
    - Lab: Use CSI driver to mount Key Vault secrets to pods.
    - Outcome: Learners secure containerized workloads.

✅ Final Outcomes: Learners can **securely manage secrets, enforce policies, and integrate apps with Key Vault**.

---

8. Load Balancers (Layer 4)

🔹 Concept

Azure Load Balancer distributes traffic at **network (L4) level**.

🔹 Why we are doing this

- Improve app availability.
- Scale workloads horizontally.
- Route traffic to healthy endpoints.

🔹 Tools

- Azure Portal
- CLI (az network lb create)

🔹 Useful in Real Time

- Distribute user traffic to VM pool.
- HA for internal business apps.
- DR with regional load balancers.

🟢 Basic (2 scenarios)

1. **Internal LB**
    - Lab: Create internal LB for 2 VMs in backend pool. Test traffic flow.
    - Outcome: Learners see traffic balancing.
2. **Public LB**
    - Lab: Create LB with public IP. Access app externally.
    - Outcome: Learners publish services securely.

🟡 Intermediate (3 scenarios)

1. **Health Probe**
    - Lab: Configure probe on port 80. Stop IIS on one VM. Traffic shifts.
    - Outcome: Learners validate failover.
2. **Inbound NAT**
    - Lab: Configure RDP NAT for each VM.
    - Outcome: Learners practice admin connectivity.

3. **HA Ports**
    - Lab: Enable HA ports for all traffic.
    - Outcome: Learners understand advanced balancing.

🔴 **Advanced (3 scenarios)**

1. **Cross-Region LB**
    - Lab: Configure global LB. Simulate region failover.
    - Outcome: Learners ensure global HA.
2. **Dual Stack LB**
    - Lab: Configure IPv6 support.
    - Outcome: Learners support modern networks.
3. **Integration with VNets**
    - Lab: Secure backend with private LB.
    - Outcome: Learners deploy zero-trust infra.

✅ Final Outcomes: Learners can **design internal/external, scalable, resilient load balancing**.

---

9. Application Gateway (Layer 7 + WAF)

🔷 **Concept**

App Gateway is a **Layer 7 reverse proxy** with Web Application Firewall (WAF).

🔷 **Why we are doing this**

- Smart routing (path-based, host-based).
- Protect apps from OWASP attacks.
- SSL termination.

🔷 **Tools**

- Azure Portal
- CLI (az network application-gateway create)

🔷 **Useful in Real Time**

- Route /api traffic to APIs, /app to web app.
- Protect against SQLi, XSS attacks.
- Host multiple domains.

🟢 **Basic (2 scenarios)**

1. **Path-based Routing**
    - Lab: Route /api → API VM, /app → Web VM.
    - Outcome: Learners understand L7 routing.
2. **Multi-Site Hosting**
    - Lab: Host app1.com and app2.com.
    - Outcome: Learners host multiple apps.

□ Intermediate (3 scenarios)

1. **Enable WAF**
    - Lab: Run SQL injection test. Blocked by WAF.
    - Outcome: Learners see app protection.
2. **End-to-End SSL**
    - Lab: Configure SSL certificate.
    - Outcome: Learners understand SSL offloading.
3. **Custom Error Pages**
    - Lab: Create branded error page.
    - Outcome: Learners improve UX.

⬤ Advanced (3 scenarios)

1. **Autoscaling Gateway**
    - Lab: Enable autoscaling on gateway. Stress test.
    - Outcome: Learners validate elasticity.
2. **Rewrite HTTP Headers**
    - Lab: Rewrite headers before sending to backend.
    - Outcome: Learners control traffic metadata.
3. **Hybrid Gateway**
    - Lab: Route on-prem + Azure apps together.
    - Outcome: Learners integrate hybrid infra.

✅ Final Outcomes: Learners can **design secure, intelligent, and scalable web gateways with WAF**.

---

10. Monitoring Tools (Azure Monitor, Log Analytics, Application Insights)

◈ Concept

Azure monitoring stack provides **observability** for infra, apps, and logs.

◈ Why we are doing this

- Detect issues proactively.
- Enable root cause analysis.
- Optimize performance & reliability.

◈ Tools

- Azure Monitor
- Log Analytics (KQL queries)
- Application Insights
- Alerts & Dashboards

◈ Useful in Real Time

- Alerting on high CPU.
- Analyzing failed logins.
- End-to-end tracing of apps.

□ Basic (2 scenarios)

1. **Enable VM Monitoring**
    - Lab: Enable monitoring, view CPU metrics.
    - Outcome: Learners track infra health.
2. **Alert on CPU Usage**
    - Lab: Create alert for CPU > 80%.
    - Outcome: Learners practice basic alerting.

□ Intermediate (3 scenarios)

1. **Log Analytics Queries**
    - Lab: Query failed logins from VM logs.
    - Outcome: Learners detect anomalies.
2. **Dashboard Creation**
    - Lab: Build dashboard with CPU/memory graphs.
    - Outcome: Learners visualize health.
3. **Alert Groups**
    - Lab: Configure alert group → email + Teams.
    - Outcome: Learners escalate alerts.

● Advanced (3 scenarios)

1. **Application Insights**
    - Lab: Enable App Insights on App Service. View request traces.
    - Outcome: Learners trace user journey.
2. **Distributed Tracing**
    - Lab: Enable tracing across 2 microservices.
    - Outcome: Learners understand full request path.
3. **Integration with SIEM**
    - Lab: Forward logs to Sentinel.
    - Outcome: Learners connect monitoring with security.

✅ Final Outcomes: **implement monitoring, alerting, logging, and tracing across Azure infra & apps**.

11. App Services

🔹 **Concept**

Azure App Service is a **PaaS (Platform-as-a-Service)** for hosting web apps, APIs, and mobile backends without managing servers.

🔹 **Why we are doing this**

- Faster deployment (no infra worries).
- Autoscaling and high availability.
- Built-in integration with DevOps.

🔹 **Tools**

- Azure Portal → App Services
- CLI (az webapp create)
- Deployment Center (GitHub, DevOps, FTP)

🔹 **Useful in Real Time**

- Hosting corporate websites.
- Running APIs without VM management.
- Blue-Green deployments for updates.

□ Basic (2 scenarios)

1. **Deploy Web App**

- Lab: Deploy sample .NET app to App Service.
- Outcome: Learners see easy app hosting.

2. **Custom Domain**
   - Lab: Add custom domain to App Service.
   - Outcome: Learners map apps to business URLs.

□ Intermediate (3 scenarios)

1. **Deployment Slots**
   - Lab: Create staging slot, deploy update, swap slots.
   - Outcome: Learners do zero-downtime deployments.
2. **Scaling App Service**
   - Lab: Configure scale-out to 3 instances.
   - Outcome: Learners test load distribution.
3. **Managed Identity**
   - Lab: Enable managed identity, access Key Vault secret.
   - Outcome: Learners integrate with secure identity.

⬤ **Advanced (3 scenarios)**

1. **Hybrid Connections**
   - Lab: Connect App Service to on-prem SQL DB.
   - Outcome: Learners integrate hybrid infra.
2. **VNet Integration**
   - Lab: Restrict App Service to private VNet.
   - Outcome: Learners implement private hosting.
3. **App Service Environment (ASE)**
   - Lab: Deploy ASE for isolated high-security apps.
   - Outcome: Learners understand enterprise-grade PaaS.

✅ Final Outcomes: Learners can **deploy, scale, secure, and integrate apps with Azure App Services**.

---

12. Function Apps (Serverless)

◈ **Concept**

Azure Functions enable **serverless compute** — run code on demand without provisioning servers.

◈ **Why we are doing this**

- Pay only per execution.
- Event-driven automation.
- Easily integrate with Azure services.

◈ **Tools**

- Portal → Function Apps
- Azure Functions Core Tools
- CLI (az functionapp create)

◈ **Useful in Real Time**

- Auto-process files when uploaded.
- Scheduled background jobs.
- Event-driven pipelines.

□ Basic (2 scenarios)

1. **HTTP Trigger**
    - Lab: Create HTTP function returning “Hello Azure!”.
    - Outcome: Learners understand triggers.
2. **Timer Trigger**
    - Lab: Run function every 5 minutes.
    - Outcome: Learners automate tasks.

□ Intermediate (3 scenarios)

1. **Blob Trigger**
    - Lab: Auto-run function when file uploaded to blob.
    - Outcome: Learners see event-driven compute.
2. **Queue Trigger**
    - Lab: Function reads messages from Storage Queue.
    - Outcome: Learners connect messaging pipelines.
3. **Bindings**
    - Lab: Bind function to input/output resources.
    - Outcome: Learners reduce boilerplate code.

● **Advanced (3 scenarios)**

1. **Durable Functions**

- Lab: Create workflow with multiple steps (approval process).
- Outcome: Learners orchestrate serverless workflows.

2. **API Integration**
    - Lab: Function triggered by Event Grid → process event → call API.
    - Outcome: Learners integrate serverless with APIs.
3. **Hybrid Function (On-Prem)**
    - Lab: Deploy function with Hybrid Connection to on-prem DB.
    - Outcome: Learners bridge hybrid workloads.

✅ Final Outcomes: Learners can **build serverless apps triggered by events, scale automatically, and integrate with services**.

---

13. Service Bus

🔹 Concept

Service Bus is an **enterprise messaging system** with queues and topics for reliable communication.

🔹 Why we are doing this

- Decouple services.
- Enable asynchronous processing.
- Ensure reliable message delivery.

🔹 Tools

- Azure Portal → Service Bus
- CLI (az servicebus namespace create)
- SDKs (.NET, Java, Python)

🔹 Useful in Real Time

- Order processing in e-commerce.
- Event-driven workflows.
- Guaranteed message delivery.

□ Basic (2 scenarios)

1. **Queue Messaging**
    - Lab: Send message to queue, receive via app.
    - Outcome: Learners understand queue basics.

2. **Dead Letter Queue**
    - Lab: Force message delivery failure → message moves to DLQ.
    - Outcome: Learners handle failed messages.

□ Intermediate (3 scenarios)

1. **Topics & Subscriptions**
    - Lab: Publish message to topic, multiple subscribers receive.
    - Outcome: Learners implement pub-sub.
2. **Session-Based Queue**
    - Lab: Enable sessions to order messages.
    - Outcome: Learners handle sequential workloads.
3. **Scheduled Messages**
    - Lab: Schedule message for 5 mins later.
    - Outcome: Learners practice delayed delivery.

⬤ Advanced (3 scenarios)

1. **Geo-DR**
    - Lab: Configure geo-disaster recovery namespace.
    - Outcome: Learners ensure messaging resilience.
2. **Message Deferral**
    - Lab: Defer processing of specific message.
    - Outcome: Learners manage workflow exceptions.
3. **Hybrid Relay**
    - Lab: Use Service Bus Relay to connect on-prem API securely.
    - Outcome: Learners integrate hybrid apps.

✅ Final Outcomes: Learners can **design reliable, decoupled, enterprise-scale messaging systems**.

---

14. Event Hubs

◈ Concept

Event Hubs is a **big data streaming platform** (Kafka-compatible).

◈ Why we are doing this

- Ingest millions of events per second.
- Real-time analytics and telemetry.

- Integrate with big data pipelines.

🔹 Tools

- Azure Portal → Event Hubs
- CLI (az eventhubs namespace create)
- SDKs, Kafka endpoints

🔹 Useful in Real Time

- IoT telemetry ingestion.
- Real-time clickstream analysis.
- Streaming into Databricks/Synapse.

□ Basic (2 scenarios)

1. **Stream Data**
    - Lab: Producer app sends data, consumer reads in real time.
    - Outcome: Learners understand ingestion.
2. **Capture to Blob**
    - Lab: Enable Capture, events auto-saved to Blob.
    - Outcome: Learners persist raw data.

□ Intermediate (3 scenarios)

1. **Consumer Groups**
    - Lab: Add consumer group, connect Power BI for visualization.
    - Outcome: Learners enable multiple readers.
2. **Scaling with Partitions**
    - Lab: Create 4 partitions, distribute load.
    - Outcome: Learners scale ingestion.
3. **Throughput Units**
    - Lab: Increase throughput units. Show capacity change.
    - Outcome: Learners optimize performance.

⬤ Advanced (3 scenarios)

1. **Kafka Integration**
    - Lab: Publish events via Kafka producer.
    - Outcome: Learners use Kafka skills in Azure.
2. **Streaming Analytics**

- Lab: Connect Event Hub → Stream Analytics → SQL DB.
- Outcome: Learners see real-time pipelines.

3. **Geo-DR for Event Hubs**
    - Lab: Failover namespace to another region.
    - Outcome: Learners ensure global continuity.

✅ Final Outcomes: Learners can **design real-time data ingestion pipelines for big data workloads**.

---

15. API Management (APIM)

🔹 **Concept**

API Management is a secure **API gateway** that publishes, secures, and monitors APIs.

🔹 **Why we are doing this**

- Centralize API publishing.
- Apply policies (throttling, transformation).
- Expose APIs to partners securely.

🔹 **Tools**

- Azure Portal → API Management
- CLI (az apim create)
- Dev Portal (for consumers)

🔹 **Useful in Real Time**

- Secure backend APIs.
- Rate-limit free users.
- Version control for APIs.

🟢 Basic (2 scenarios)

1. **Publish API**
    - Lab: Import backend API, expose via APIM.
    - Outcome: Learners publish API securely.
2. **Developer Portal**
    - Lab: Explore API docs in Dev Portal.
    - Outcome: Learners see consumer experience.

🟡 Intermediate (3 scenarios)

1. **Policy: Rate Limiting**
    - Lab: Limit free tier to 100 calls/day.
    - Outcome: Learners apply usage plans.
2. **Transform Response**
    - Lab: Mask sensitive fields in response.
    - Outcome: Learners enforce data protection.
3. **Mock API**
    - Lab: Create mock API returning sample response.
    - Outcome: Learners test integration.

🔴 **Advanced (3 scenarios)**

1. **Multi-Region APIM**
    - Lab: Deploy gateway in US & EU. Failover traffic.
    - Outcome: Learners ensure global presence.
2. **Custom Identity Provider**
    - Lab: Enable OAuth2 for APIs.
    - Outcome: Learners secure APIs with tokens.
3. **APIM with VNet**
    - Lab: Restrict API access to internal VNet.
    - Outcome: Learners implement private APIs.

✅ Final Outcomes: Learners can **publish, secure, monitor, and scale APIs with enterprise-grade policies**.

16. Azure SQL Database

🔷 **Concept**

Azure SQL Database is a **fully managed relational database** service.

🔷 **Why we are doing this**

- No patching, backups, or infra management.
- Built-in HA, scaling, and security.
- Support for modern and legacy apps.

🔷 **Tools**

- Azure Portal → SQL Database
- CLI (az sql db create)

- SSMS / Azure Data Studio

🔷 Useful in Real Time

- Hosting transactional apps.
- Analytics with secure relational DB.
- DR-ready databases.

□ Basic (2 scenarios)

1. **Create DB & Connect**
    - Lab: Create SQL DB, connect with SSMS.
    - Outcome: Learners provision DB.
2. **Firewall Rules**
    - Lab: Allow client IP, test connection.
    - Outcome: Learners secure DB endpoints.

□ Intermediate (3 scenarios)

1. **Transparent Data Encryption (TDE)**
    - Lab: Enable TDE. View encryption status.
    - Outcome: Learners secure storage.
2. **Geo-Replication**
    - Lab: Replicate DB to another region.
    - Outcome: Learners build DR solution.
3. **Auditing**
    - Lab: Enable auditing → check login attempts.
    - Outcome: Learners enable compliance.

⬤ Advanced (3 scenarios)

1. **Hyperscale**
    - Lab: Scale DB from 10GB to 1TB.
    - Outcome: Learners test scalability.
2. **Elastic Pools**
    - Lab: Run multiple DBs in pool.
    - Outcome: Learners optimize cost.
3. **Failover Group**
    - Lab: Configure auto-failover between regions.

- Outcome: Learners validate global HA.

✅ Final Outcomes: Learners can **deploy, secure, scale, and replicate managed SQL databases**.

---

17. Azure Databricks

🔹 Concept

Azure Databricks is a **data engineering, ML, and analytics platform** powered by Apache Spark.

🔹 Why we are doing this

- Handle big data workloads.
- Unified data lakehouse architecture.
- AI/ML at scale.

🔹 Tools

- Azure Portal → Databricks workspace
- Notebooks (Python, Scala, SQL)
- MLflow, Delta Lake

🔹 Useful in Real Time

- ETL pipelines.
- Streaming analytics.
- Machine learning workflows.

🟢 Basic (2 scenarios)

1. **Notebook Demo**
   - Lab: Word count using PySpark.
   - Outcome: Learners run first Spark job.
2. **Delta Table**
   - Lab: Create Delta Lake table from blob data.
   - Outcome: Learners handle structured data.

🟡 Intermediate (3 scenarios)

1. **Streaming Ingestion**
   - Lab: Ingest streaming sales data → Delta Lake.
   - Outcome: Learners analyze real-time data.

2. **ML Model Training**
    - Lab: Train simple ML model with MLflow.
    - Outcome: Learners deploy ML pipeline.
3. **Job Scheduling**
    - Lab: Schedule nightly ETL job.
    - Outcome: Learners automate data processing.

🔴 **Advanced (3 scenarios)**

1. **Integration with Synapse**
    - Lab: Query Databricks table from Synapse.
    - Outcome: Learners integrate BI.
2. **Advanced ML**
    - Lab: Train deep learning model on GPU cluster.
    - Outcome: Learners scale AI workloads.
3. **Unity Catalog**
    - Lab: Manage data governance with Unity Catalog.
    - Outcome: Learners enforce enterprise data governance.

✅ Final Outcomes: Learners can **process, analyze, and model large-scale data with Databricks**.

---

18. Azure DevOps (Repos, Boards, Artifacts)

🔹 **Concept**

Azure DevOps provides an **end-to-end DevOps suite** for code, planning, artifacts, and CI/CD.

🔹 **Why we are doing this**

- Version control with Git.
- Agile planning with Boards.
- Artifact/package management.

🔹 **Tools**

- Azure DevOps Portal
- Git CLI
- Pipelines, Boards, Repos, Artifacts

🔹 **Useful in Real Time**

- Agile project tracking.
- Secure package feeds.
- Unified DevOps workflows.

□ Basic (2 scenarios)

1. **Repo Creation**
    - Lab: Create Git repo, push sample code.
    - Outcome: Learners practice version control.
2. **Work Item Tracking**
    - Lab: Create user story & task in Boards.
    - Outcome: Learners link tasks to code.

□ Intermediate (3 scenarios)

1. **Branch Policies**
    - Lab: Enforce PR review for master branch.
    - Outcome: Learners ensure code quality.
2. **Artifact Feed**
    - Lab: Publish NuGet package to Artifacts.
    - Outcome: Learners share secure packages.
3. **Agile Dashboard**
    - Lab: Create dashboard with sprint burndown.
    - Outcome: Learners visualize project health.

⬤ **Advanced (3 scenarios)**

1. **Service Connections**
    - Lab: Connect Azure DevOps with Azure subscription.
    - Outcome: Learners integrate cloud infra.
2. **End-to-End DevOps Workflow**
    - Lab: Plan → Code → Build → Release via DevOps suite.
    - Outcome: Learners execute complete DevOps lifecycle.
3. **Compliance Reports**
    - Lab: Generate change tracking reports.
    - Outcome: Learners meet audit needs.

☑ Final Outcomes: Learners can **manage projects, code, and artifacts within DevOps**.

---

19. Azure DevOps Pipelines

◈ Concept

Pipelines provide **CI/CD automation** for apps and infra.

◈ Why we are doing this

- Automate build & deployment.
- Ensure faster delivery.
- Enforce quality gates.

◈ Tools

- Azure DevOps Pipelines (YAML/Classic)
- GitHub Actions (alternative)
- CLI

◈ Useful in Real Time

- Automating web app deployments.
- Infrastructure as Code.
- Secure DevSecOps pipelines.

□ Basic (2 scenarios)

1. **Build Pipeline**
    - Lab: Create pipeline to compile .NET app.
    - Outcome: Learners build code automatically.
2. **Release Pipeline**
    - Lab: Deploy app to App Service.
    - Outcome: Learners automate deployment.

□ Intermediate (3 scenarios)

1. **Multi-Stage Pipeline**
    - Lab: Deploy to Dev → Prod.
    - Outcome: Learners enforce environments.
2. **Pipeline Variables & Secrets**
    - Lab: Secure secrets in pipeline.
    - Outcome: Learners secure automation.
3. **Infrastructure Deployment**

- Lab: Deploy VM with ARM template pipeline.
- Outcome: Learners practice IaC.

🔴 **Advanced (3 scenarios)**

1. **Integrate Security Scans**
   - Lab: Add OWASP scan step.
   - Outcome: Learners build DevSecOps.
2. **Container CI/CD**
   - Lab: Build Docker image, push to ACR, deploy to AKS.
   - Outcome: Learners manage container lifecycle.
3. **Approval Gates**
   - Lab: Add manual approval for Prod stage.
   - Outcome: Learners implement governance.

✅ Final Outcomes: Learners can **design CI/CD pipelines for apps, infra, and security**.

---

20. AKS (Azure Kubernetes Service)

🔷 **Concept**

AKS is Azure's managed **Kubernetes orchestration service**.

🔷 **Why we are doing this**

- Deploy containers at scale.
- Automate scaling & upgrades.
- Manage microservices architectures.

🔷 **Tools**

- Azure Portal → AKS
- CLI (az aks create)
- kubectl / Helm

🔷 **Useful in Real Time**

- Host microservices apps.
- Automate deployments via Helm.
- Hybrid + GitOps clusters.

🟢 Basic (2 scenarios)

1. **Deploy Pod**
    - Lab: Deploy nginx pod → access public IP.
    - Outcome: Learners deploy container.
2. **Scale Pods**
    - Lab: Scale nginx from 1 → 3 replicas.
    - Outcome: Learners understand scaling.

□ Intermediate (3 scenarios)

1. **Ingress Controller**
    - Lab: Route /app1 → Pod1, /app2 → Pod2.
    - Outcome: Learners practice routing.
2. **Secrets in AKS**
    - Lab: Store DB password in secret, mount in pod.
    - Outcome: Learners secure workloads.
3. **Rolling Update**
    - Lab: Update pod image, rollout gradually.
    - Outcome: Learners test zero-downtime deployments.

⬤ Advanced (3 scenarios)

1. **GitOps with Flux/ArgoCD**
    - Lab: Automate deployment via GitOps repo.
    - Outcome: Learners implement modern ops.
2. **Service Mesh**
    - Lab: Install Istio, enable traffic policy.
    - Outcome: Learners secure microservices traffic.
3. **Azure Policy for AKS**
    - Lab: Block privileged containers.
    - Outcome: Learners enforce compliance.

✅ Final Outcomes: Learners can **deploy, secure, and scale containerized microservices on AKS**.

---

21. Azure Virtual Desktop (AVD)

◈ Concept

AVD delivers **Windows desktops and apps** from Azure to any device.

◈ **Why we are doing this**

- Remote workforce enablement.
- Centralized security & management.
- Cost-effective VDI solution.

◈ **Tools**

- Azure Portal → Virtual Desktop
- FSLogix for profiles
- RD Client

◈ **Useful in Real Time**

- Work from anywhere securely.
- Shared desktops for contractors.
- Compliance-driven industries.

□ Basic (2 scenarios)

1. **Publish Desktop**
    - Lab: Publish Windows desktop to user. Connect with RD client.
    - Outcome: Learners see remote access.
2. **Publish App Only**
    - Lab: Publish MS Word only.
    - Outcome: Learners deliver apps instead of full desktops.

□ Intermediate (3 scenarios)

1. **FSLogix Profile Container**
    - Lab: Configure FSLogix for user profiles.
    - Outcome: Learners enable roaming profiles.
2. **Scaling Host Pool**
    - Lab: Auto-scale desktops based on usage.
    - Outcome: Learners optimize cost.
3. **Multi-Session Host**
    - Lab: Configure multi-user Windows 10 host.
    - Outcome: Learners maximize density.

● **Advanced (3 scenarios)**

1. **Conditional Access for AVD**

- Lab: Enforce MFA for AVD login.
- Outcome: Learners secure desktops.

2. **Hybrid AVD**
    - Lab: Access on-prem file server via AVD.
    - Outcome: Learners integrate hybrid resources.
3. **Disaster Recovery AVD**
    - Lab: Replicate AVD setup to another region.
    - Outcome: Learners ensure DR.

✅ Final Outcomes: Learners can **deploy, scale, and secure virtual desktops/apps with AVD**.

---

22. Azure Managed Prometheus + Grafana

🔹 **Concept**

Azure offers **managed Prometheus and Grafana** for deep observability.

🔹 **Why we are doing this**

- Collect time-series metrics.
- Visualize infra/app performance.
- Set alerts for anomalies.

🔹 **Tools**

- Azure Monitor → Managed Prometheus
- Azure Managed Grafana
- KQL & PromQL queries

🔹 **Useful in Real Time**

- Monitor AKS workloads.
- Custom dashboards for SRE teams.
- Integrate with alerts & incident response.

□ Basic (2 scenarios)

1. **Default Dashboard**
    - Lab: View AKS metrics in Grafana.
    - Outcome: Learners explore observability.
2. **Simple Query**

- Lab: Run PromQL for CPU usage.
- Outcome: Learners fetch metrics.

□ Intermediate (3 scenarios)

1. **Custom Dashboard**
   - Lab: Create dashboard with CPU/memory charts.
   - Outcome: Learners customize views.
2. **Alert Rule**
   - Lab: Alert when CPU > 80%.
   - Outcome: Learners practice alerting.
3. **Log Correlation**
   - Lab: Correlate AKS metrics with app logs.
   - Outcome: Learners find root cause.

⬤ Advanced (3 scenarios)

1. **Multi-Cloud Monitoring**
   - Lab: Add AWS/GCP metrics to Grafana.
   - Outcome: Learners unify observability.
2. **Business KPIs**
   - Lab: Dashboard with revenue-per-minute from logs.
   - Outcome: Learners connect business + infra metrics.
3. **Integration with Incident Response**
   - Lab: Connect Grafana alerts to PagerDuty/Teams.
   - Outcome: Learners automate incident management.

✅ Final Outcomes: Learners can **monitor, visualize, and alert across infra and apps using managed Prometheus & Grafana**.

🚀 **Overall Benefits of This Training**

- **End-to-End Azure Mastery** → Covers infra, networking, security, PaaS, DevOps, data, and monitoring.
- **Hands-On First Approach** → Each topic includes live scenarios + labs for real skill-building.
- **Progressive Learning Path** → Moves from **Basic** → **Intermediate** → **Advanced** with real-time outcomes.

- **Enterprise-Ready Skills** → Learners gain knowledge aligned with **Fortune 500 cloud practices**.
- **Identity & Governance Clarity** → Strong foundation in subscriptions, RBAC, policies, compliance.
- **Security by Design** → Covers Key Vault, RBAC, WAF, encryption, zero-trust networking.
- **Cloud-Native App Development** → App Services, Functions, Service Bus, Event Hubs, API Management.
- **DevOps + Automation** → End-to-end DevOps pipelines, IaC, GitOps, CI/CD, and security scanning.
- **Data & AI Integration** → Azure SQL, Databricks, Event Hubs for analytics and ML workloads.
- **Modern Work Enablement** → AVD, hybrid networking, multi-cloud monitoring with Prometheus + Grafana.
- **Disaster Recovery & Resilience** → Snapshots, backups, geo-replication, failover groups, DR strategies.
- **Cost Optimization Mindset** → Covers Spot VMs, lifecycle rules, tagging, budget alerts, scaling.
- **Multi-Cloud Awareness** → Prometheus + Grafana and enterprise networking show cross-cloud integration.
- **Industry Alignment** → Content maps to **job roles (Cloud Architect, DevOps Engineer, Security Engineer, Data Engineer)**.
- **Corporate Relevance** → Every service is tied to **real-world enterprise scenarios** (finance, healthcare, retail, remote workforce).
- **Career Boost** → Prepares for **Microsoft certifications (AZ-104, AZ-305, AZ-400, DP-203, SC-100)**.