Azure Monitoring and Scaling Practice Lab

1. Azure Monitoring

Azure Monitoring is all about **observing and analyzing** the health, performance, and availability of your applications, infrastructure, and resources in the Azure cloud.

- **Tool used**: Azure Monitor is the main service.
- **What it does**:
  - Collects **metrics** (e.g., CPU %, memory, request rates, latency).
  - Collects **logs** (e.g., errors, exceptions, activity logs, security logs).
  - Provides **alerts** (e.g., notify if CPU > 80% for 5 minutes).
  - **Dashboards & Insights** – helps visualize data in real-time.
  - Can integrate with **Application Insights** (for app-level monitoring) and **Log Analytics** (for deep querying).

👉 Example: If your Web API in Azure App Service suddenly slows down, Azure Monitor can alert you that **response time > 3 seconds**, helping you fix it before users complain.

---

2. Azure Scaling

Scaling in Azure means adjusting the **resources** (like CPU, memory, or instances of an app) to meet demand.
This ensures **performance** during high traffic and **cost savings** during low usage.

There are two main types:

a) Vertical Scaling (Scale Up/Down)

- Increasing or decreasing the **power of the resource**.
- Example: Changing your App Service Plan from **Basic (1 vCPU, 1.75 GB RAM)** to **Premium (4 vCPU, 14 GB RAM)**.

b) Horizontal Scaling (Scale Out/In)

- Increasing or decreasing the **number of resource instances**.
- Example: Running your API on **2 instances during low traffic** and automatically scaling to **10 instances during peak hours**.
- Often managed by **Azure Autoscale** rules (e.g., scale out when CPU > 70%).

👉 Scaling is what keeps apps **responsive** and **cost-effective**.

✅ **In short:**

- **Azure Monitoring** = "Keep an eye on everything, get alerts, analyze issues."
- **Azure Scaling** = "Automatically adjust resources to match demand."

Let's build a **hands-on lab exercise** where you'll practice both **Azure Monitoring** and **Auto Scaling** using an **Azure App Service** (since it's the simplest way to start).

🧪 Lab Exercise – Azure Monitoring & Scaling with App Service

Prerequisites

- An **Azure subscription**
- A **sample .NET Web API or Angular app** (or any app you can deploy to Azure App Service)
- Visual Studio / VS Code or Azure CLI

Step 1: Create a Resource Group

1. Go to **Azure Portal** → Search for **Resource Groups**.
2. Click **+ Create**.
3. Enter:
    - Resource Group name: `AppServiceLabRG`
    - Region: Select nearest to you (e.g., Central India).
4. Click **Review + Create** → **Create**.

Step 2: Create an App Service

1. In Azure Portal, search for **App Services** → **+ Create**.
2. Fill details:
    - Subscription: Your subscription
    - Resource Group: `AppServiceLabRG`
    - Name: `myappservice-lab` (must be unique)
    - Publish: Code
    - Runtime: `.NET 8 (LTS)` or Node.js/Python if you prefer
    - OS: Windows
    - Region: Same as Resource Group
3. Select **App Service Plan**:
    - Click **Create new** → Choose **B1 (Basic)** (allows scaling).

4. Click **Review + Create → Create**.

Step 3: Deploy Your Application

- From **Visual Studio**:
  - Right-click project → **Publish** → Select **Azure → App Service (Windows)**.
- Or use **Azure CLI**:
- `az webapp up --name myappservice-lab --resource-group AppServiceLabRG --runtime "DOTNET:8"`

Once deployed, open `https://myappservice-lab.azurewebsites.net` in browser.

Step 4: Enable Monitoring

1. Open your App Service → Left menu → **Application Insights**.
2. Click **Turn On Application Insights**.
3. Select **Create New** and link to your app.
4. Choose **Log Analytics Workspace** → Create a new one.
5. Once enabled, you'll get:
   - **Live Metrics** (requests/sec, response time, failures).
   - **Availability Tests** (ping app every 5 mins).
   - **Alerts** (CPU > 80%, app errors, etc.).

Step 5: Create an Alert Rule

1. Go to App Service → **Alerts** → **+ Create Alert Rule**.
2. Select **Scope**: Your App Service.
3. Select **Condition**:
   - Signal: `Percentage CPU > 70`
   - Operator: Greater than
   - Aggregation: Average over 5 minutes
4. Select **Action Group**:
   - Create new → Add your **Email ID**.
5. Click **Review + Create**.

✅ Now you'll get an **email alert** when CPU > 70%.

Step 6: Configure Auto Scaling

1. Go to App Service → **Scale out (App Service Plan)**.

2. Select **Custom Autoscale**.
3. Add Rule:
    - Condition: CPU % > 70 for 5 minutes → **Increase instance count by 1**.
    - Condition: CPU % < 30 for 10 minutes → **Decrease instance count by 1**.
4. Set **Min Instances = 1** and **Max Instances = 5**.
5. Save.

Step 7: Test Auto Scaling

- Use a load testing tool (like **Azure Load Testing**, JMeter, or `ab` command). Example:
- `ab -n 10000 -c 200 https://myappservice-lab.azurewebsites.net/`
- Watch metrics in **Application Insights → Live Metrics**.
- Once CPU > 70%, Azure will automatically add more instances.

Step 8: Review Logs & Metrics

- Go to **Log Analytics → Logs** and run a query:
- `requests`
- `| summarize count() by bin(timestamp, 1m), resultCode`
- View how many requests per minute were served.
- Check **Alerts** to confirm scaling & email notifications.

☑ End Result

- You deployed a Web App.
- Enabled **Application Insights Monitoring**.
- Set up **CPU Alerts**.
- Configured **Auto Scaling** rules.
- Tested with **load** to trigger scaling.