Virtual Machine Scale Sets (VMSS)

What is Scale Sets?

VM Scale Sets are intended to manage a group of identical, load-balanced VMs.

Automatically scales the number of VMs based on demand or schedule.

Ensures high availability and redundancy.

Key Features:

Autoscaling: Automatically scales VM instances on the basis of CPU, memory, or custom metrics.

Load Balancing: It integrates with Azure Load Balancer or Application Gateway.

Fault Tolerance: Spreads VMs across fault and update domains

Integration: Seamless integration with Azure Monitor and Azure DevOps.

Practical Example: **Deploying a Web App on Scale Sets.**

Create a Resource Group:

```
az group create
--name ScaleSetRG
--location "East US"
```

Create a Virtual Machine Scale Set:

```
az vmss create \
--resource-group ScaleSetRG \
 --name MyScaleSet \
--image UbuntuLTS \
--admin-username azureuser \
--generate-ssh-keys \
--instance-count 2 \
 --upgrade-policy-mode Automatic \
--load-balancer ""
```

Enable Autoscaling:

```
az monitor autoscale create \
    --resource-group ScaleSetRG \
    --resource MyScaleSet \
    --resource-type
Microsoft.Compute/virtualMachineScaleSets \
    --name MyAutoScaleRule \
    --min-count 2 \
    --max-count 10 \
    --count 3
```

Add a Scaling Rule: Scale out when CPU usage exceeds 70%:

```
az monitor autoscale rule create \
    --resource-group ScaleSetRG \
--. autoscale-name MyAutoScaleRule \
    --condition "Percentage CPU > 70 avg 5m"  \
    --scale out 1
```

Test the Setup:

Apply high traffic load on VMs by running a CPU intensive task.

Monitor scaling through the Azure Portal or CLI:

```
az vmss list-instances
--resource-group ScaleSetRG
--name MyScaleSet
--output table
```

Access the Application:

Get the public IP of the Load Balancer:

```
az network public-ip show
-resource-group ScaleSetRG
--name MyScaleSetLBPublicIP
--query ipAddress -o tsv
```

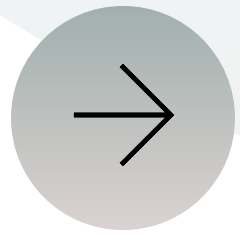
Open the IP address in a browser to view the hosted application.

2. Azure App Services

What are App Services?

A PaaS service offering of hosting web apps, REST APIs, and mobile backends.

A fully managed service with built-in scaling, security, and CI/CD capabilities.



Key Features

Language Support:
.NET, Java, Python, PHP, Node.js, and Ruby.

Autoscaling: Autoscale based on usage.

Custom Domains & SSL: Add secure connection with ease.

DevOps Integration: Integration with Azure DevOps, GitHub, and Bitbucket for CI/CD.

Deployment Slots: Test new versions in staging environments before production.

Practical Example: Deploying a Web App

Create a Resource Group:

```
az group create
--name AppServiceRG
--location "East US"
```

Create an App Service Plan:

The plan determines the compute resources for the app.

```
az appservice plan create \
    --name MyAppServicePlan \
    --resource-group AppServiceRG \
    --sku B1 \
    --is-linux
```

Create a Web App:

```
  az webapp create \
    --resource-group AppServiceRG \
    --plan MyAppServicePlan\
--name MyWebApp \
    --runtime "PYTHON|3.8"
```

Deploy Your Code:

Use GitHub for CI/CD integration or deploy manually:

```
az webapp deployment source config \
 --name MyWebApp \
 --resource-group AppServiceRG \
 --repo-url https://github.com/<your-repo-url> \
 --branch main \
 --manual-integration
```

Access the Web App:

The default URL will be: https://<MyWebApp>.azurewebsites.net

Enable Scaling:

```
az monitor autoscale create
--resource-group AppServiceRG \
 --resource MyAppServicePlan \
 --resource-type Microsoft.Web/serverFarms \
 --name AppAutoScaleRule \
 --min-count 1 \
 --max-count 5 \
 --count 2
```

VM Scale Sets:
Use when you need full control over the underlying infrastructure, or for compute-heavy applications like data processing pipelines.

App Services:
Use for web apps, APIs, or lightweight applications where simplicity, scaling, and integration are the priority.