Azure Availability Sets vs VMSS - Hands-On Workshop

Workshop Overview

Duration: 90 minutes
Level: Intermediate
Prerequisites: Azure subscription, basic VM knowledge

Lab 1: VM Scale Set with Autoscaling (Azure Portal)

Objectives

- Create a VMSS using Azure Portal
- Configure autoscaling rules
- Test load balancing
- Monitor scaling events

Part 1: Create Virtual Machine Scale Set (20 min)

Step 1: Navigate to VMSS Creation

1. Sign in to **Azure Portal** (portal.azure.com)
2. Click **"Create a resource"** (top left)
3. Search for **"Virtual machine scale set"**
4. Click **"Create"**

Step 2: Configure Basics Tab

- **Subscription:** Select your subscription
- **Resource group:** Click "Create new" → Enter `rg-vmss-workshop`
- **Scale set name:** `vmss-web-app`
- **Region:** `East US`
- **Availability zone:** Select `Zones 1, 2, 3` (for 99.99% SLA)
- **Orchestration mode:** `Uniform`
- **Image:** `Ubuntu Server 22.04 LTS - x64 Gen2`
- **Size:** Click "See all sizes" → Select `B2s` → Click "Select"

- **Username:** `azureuser`
- **Authentication type:** `Password`
- **Password:** Create a strong password (note it down)
- Click **"Next: Disks >"**

Step 3: Configure Disks Tab

- **OS disk type:** `Standard SSD (locally redundant storage)`
- Click **"Next: Networking >"**

Step 4: Configure Networking Tab

- **Virtual network:** Click "Create new"
  - Name: `vnet-vmss`
  - Address space: Keep default (10.0.0.0/16)
  - Subnet name: `subnet-vmss`
  - Subnet address range: Keep default
  - Click "OK"
- **Network interface:** Keep default settings
- **Load balancing:** Select `Azure load balancer`
- **Select a load balancer:** Click "Create new"
  - Name: `lb-vmss-web`
  - Click "OK"
- **Public IP address:** Keep default
- Click **"Next: Scaling >"**

Step 5: Configure Scaling Tab

- **Initial instance count:** `2`
- **Scaling policy:** Select `Custom`
- **Scale out:**
  - CPU threshold: `70%`
  - Duration (minutes): `5`
  - Number of instances to increase by: `2`
- **Scale in:**

    - CPU threshold: `30%`
    - Duration (minutes): `5`
    - Number of instances to decrease by: `1`
- **Instance limits:**
    - Minimum: `2`
    - Maximum: `10`
    - Default: `2`
- Click **"Next: Management >"**

Step 6: Configure Management Tab

- **Upgrade mode:** `Rolling`
- **Enable application health monitoring:** Check this box
- **Automatic repairs:** Check this box
- Keep other defaults
- Click **"Next: Health >"**

Step 7: Configure Health Tab

- **Enable application health monitoring:** Should already be checked
- **Protocol:** `HTTP`
- **Port:** `80`
- **Path:** `/`
- Click **"Review + create"**

Step 8: Review and Create

- Review all settings
- Click **"Create"**
- Wait for deployment (5-10 minutes)

Part 2: Install Web Server on VMSS (10 min)

Step 1: Navigate to VMSS

1. Go to **"Resource groups"** → Click `rg-vmss-workshop`

2. Click on `vmss-web-app`

Step 2: Run Custom Script Extension

1. In left menu, scroll to **"Settings"** section
2. Click **"Extensions + applications"**
3. Click **"+ Add"**
4. Select **"Custom Script For Linux"**
5. Click **"Next"**
6. In **"Script files"** section, click **"Browse"** but we'll enter directly
7. Instead, paste this in **"Command"** field:

```bash
apt-get update && apt-get install -y nginx && echo "<h1>Hello from $(hostname)</h1>" > /var/www/html/index.html && s
```

8. Click **"Review + create"** → Click **"Create"**
9. Wait for extension deployment (~5 minutes)

Step 3: Upgrade Instances

1. In left menu, click **"Instances"**
2. Select **all instances** (check the box at top)
3. Click **"Upgrade"** button at top
4. Click **"Yes"** to confirm
5. Wait for instances to upgrade (they'll restart one by one)

Part 3: Test Load Balancer and Autoscaling (15 min)

Step 1: Get Load Balancer Public IP

1. Go back to **"Resource groups"** → `rg-vmss-workshop`
2. Click on the **Public IP address** (starts with `vmss-web-app`)
3. Copy the **IP address**
4. Open new browser tab
5. Navigate to `http://<YOUR_IP_ADDRESS>`

6. Refresh multiple times - you should see different hostnames

Step 2: Generate Load to Trigger Scaling

1. In Azure Portal, click **Cloud Shell** icon (top right, looks like `>_`)
2. Select **Bash**
3. Run this command (replace with your IP):

```bash
for i in {1..10000}; do curl -s http://<YOUR_IP_ADDRESS> > /dev/null & done
```

Step 3: Monitor Autoscaling

1. Go back to your VMSS → `vmss-web-app`
2. In left menu, click **"Scaling"**
3. Watch the **"Current instance count"** - it should increase
4. In left menu, click **"Instances"** to see new instances being created
5. In left menu, click **"Metrics"**
6. Click **"Add metric"**
    - Metric: `Percentage CPU`
    - Aggregation: `Average`
7. Observe CPU spike and correlation with scaling

Step 4: View Autoscale History

1. Go to **"Resource groups"** → `rg-vmss-workshop`
2. Click on **"Autoscale setting"** (starts with `vmss-web-app-autoscale`)
3. Click **"Run history"** tab
4. View scale-out events with timestamps and reasons

Part 4: Test Rolling Upgrades (10 min)

Step 1: Simulate Application Update

1. Navigate back to `vmss-web-app`
2. Click **"Extensions + applications"**

3. Select the **Custom Script** extension
4. Click **"Update"**
5. Change the command to:

```bash
apt-get install -y nginx && echo "<h1>Hello from $(hostname) - VERSION 2</h1>" > /var/www/html/index.html
```

6. Click **"Review + create"** → **"Create"**

Step 2: Monitor Rolling Update

1. Click **"Instances"** in left menu
2. Watch as instances update one by one (shows "Updating" status)
3. Keep refreshing your browser at the load balancer IP
4. Notice zero downtime - some instances show V2 while others still show V1

Part 5: Cleanup (5 min)

1. Go to **"Resource groups"**
2. Click `rg-vmss-workshop`
3. Click **"Delete resource group"** at top
4. Type the resource group name to confirm
5. Click **"Delete"**

Lab 2: Availability Set Deployment (Azure CLI)

Objectives

- Create availability set with fault/update domains using CLI
- Deploy multiple VMs into the set
- Verify fault domain distribution
- Compare with VMSS approach

Part 1: Setup and Create Availability Set (15 min)

Step 1: Open Cloud Shell

```bash
# Click Cloud Shell icon in Azure Portal (top right)
# Or use local Azure CLI if installed
```

Step 2: Create Resource Group

```bash
az group create \
  --name rg-availset-workshop \
  --location eastus
```

Step 3: Create Virtual Network

```bash
az network vnet create \
  --resource-group rg-availset-workshop \
  --name vnet-availset \
  --address-prefix 10.1.0.0/16 \
  --subnet-name subnet-web \
  --subnet-prefix 10.1.1.0/24
```

Step 4: Create Availability Set

```bash
az vm availability-set create \
  --resource-group rg-availset-workshop \
  --name avset-web \
  --platform-fault-domain-count 2 \
  --platform-update-domain-count 5 \
  --location eastus
```

What this creates:

- 2 Fault Domains = VMs distributed across 2 physical racks
- 5 Update Domains = During maintenance, only 1/5 VMs restart at a time

Part 2: Deploy VMs into Availability Set (15 min)

Step 1: Create First VM

bash

```bash
az vm create \
  --resource-group rg-availset-workshop \
  --name vm-web-01 \
  --availability-set avset-web \
  --vnet-name vnet-availset \
  --subnet subnet-web \
  --image Ubuntu2204 \
  --size Standard_B2s \
  --admin-username azureuser \
  --admin-password 'YourStrongPassword123!' \
  --public-ip-sku Standard \
  --nsg-rule SSH
```

Step 2: Install Web Server on VM1

bash

```bash
# Get public IP of vm-web-01
VM1_IP=$(az vm show -d \
  --resource-group rg-availset-workshop \
  --name vm-web-01 \
  --query publicIps -o tsv)

echo "VM1 Public IP: $VM1_IP"

# SSH and install nginx (enter password when prompted)
ssh azureuser@$VM1_IP "sudo apt-get update && sudo apt-get install -y nginx && echo '<h1>VM-WEB-01</h1>' | sudo te
```

Step 3: Create Second VM

bash

```
az vm create \
  --resource-group rg-availset-workshop \
  --name vm-web-02 \
  --availability-set avset-web \
  --vnet-name vnet-availset \
  --subnet subnet-web \
  --image Ubuntu2204 \
  --size Standard_B2s \
  --admin-username azureuser \
  --admin-password 'YourStrongPassword123!' \
  --public-ip-sku Standard \
  --nsg-rule SSH
```

Step 4: Install Web Server on VM2

```bash
# Get public IP of vm-web-02
VM2_IP=$(az vm show -d \
  --resource-group rg-availset-workshop \
  --name vm-web-02 \
  --query publicIps -o tsv)

echo "VM2 Public IP: $VM2_IP"

# SSH and install nginx
ssh azureuser@$VM2_IP "sudo apt-get update && sudo apt-get install -y nginx && echo '<h1>VM-WEB-02</h1>' | sudo te
```

Part 3: Add Load Balancer (10 min)

Step 1: Create Load Balancer

```bash
```

```
# Create public IP for load balancer
az network public-ip create \
  --resource-group rg-availset-workshop \
  --name pip-lb \
  --sku Standard

# Create load balancer
az network lb create \
  --resource-group rg-availset-workshop \
  --name lb-web \
  --sku Standard \
  --public-ip-address pip-lb \
  --frontend-ip-name frontend-lb \
  --backend-pool-name backend-pool
```

Step 2: Create Health Probe

bash

```
az network lb probe create \
  --resource-group rg-availset-workshop \
  --lb-name lb-web \
  --name health-probe \
  --protocol tcp \
  --port 80
```

Step 3: Create Load Balancing Rule

bash

```
az network lb rule create \
  --resource-group rg-availset-workshop \
  --lb-name lb-web \
  --name lb-rule-web \
  --protocol tcp \
  --frontend-port 80 \
  --backend-port 80 \
  --frontend-ip-name frontend-lb \
  --backend-pool-name backend-pool \
  --probe-name health-probe
```

Step 4: Add VMs to Backend Pool

bash

```
# Get NIC IDs
NIC1_ID=$(az vm show \
  --resource-group rg-availset-workshop \
  --name vm-web-01 \
  --query 'networkProfile.networkInterfaces[0].id' -o tsv)

NIC2_ID=$(az vm show \
  --resource-group rg-availset-workshop \
  --name vm-web-02 \
  --query 'networkProfile.networkInterfaces[0].id' -o tsv)

# Add to backend pool
az network nic ip-config address-pool add \
  --resource-group rg-availset-workshop \
  --nic-name $(basename $NIC1_ID) \
  --ip-config-name ipconfig1 \
  --lb-name lb-web \
  --address-pool backend-pool

az network nic ip-config address-pool add \
  --resource-group rg-availset-workshop \
  --nic-name $(basename $NIC2_ID) \
  --ip-config-name ipconfig1 \
  --lb-name lb-web \
  --address-pool backend-pool
```

Step 5: Open HTTP Port on NSG

```bash
az network nsg rule create \
  --resource-group rg-availset-workshop \
  --nsg-name vm-web-01NSG \
  --name Allow-HTTP \
  --priority 100 \
  --destination-port-ranges 80 \
  --protocol Tcp

az network nsg rule create \
  --resource-group rg-availset-workshop \
  --nsg-name vm-web-02NSG \
  --name Allow-HTTP \
  --priority 100 \
  --destination-port-ranges 80 \
  --protocol Tcp
```

Part 4: Verify Configuration (10 min)

Step 1: Check Fault Domain Distribution

```bash
az vm availability-set show \
  --resource-group rg-availset-workshop \
  --name avset-web \
  --query "virtualMachines[].{Name:id}" -o table

# Get detailed instance view
az vm get-instance-view \
  --resource-group rg-availset-workshop \
  --name vm-web-01 \
  --query "platformFaultDomain"

az vm get-instance-view \
  --resource-group rg-availset-workshop \
  --name vm-web-02 \
  --query "platformFaultDomain"
```

Expected: VMs should be in different fault domains (0 and 1)

Step 2: Test Load Balancer

```bash
# Get load balancer public IP
LB_IP=$(az network public-ip show \
  --resource-group rg-availset-workshop \
  --name pip-lb \
  --query ipAddress -o tsv)

echo "Load Balancer IP: http://$LB_IP"

# Test load balancing
for i in {1..10}; do
  curl http://$LB_IP
  echo ""
done
```

Expected: You should see responses alternating between VM-WEB-01 and VM-WEB-02

Step 3: Compare with VMSS

```bash
# List all resources to see the difference
echo "=== Availability Set Resources ==="
az resource list \
  --resource-group rg-availset-workshop \
  --query "[].{Name:name, Type:type}" -o table
```

Part 5: Cleanup (5 min)

```bash
# Delete resource group
az group delete \
  --name rg-availset-workshop \
  --yes \
  --no-wait
```

Key Differences Summary

Aspect	Availability Set (Lab 2)	VMSS (Lab 1)
Creation	Manual - each VM created separately	Automatic - one config creates all
Scaling	Manual - add/remove VMs yourself	Automatic - based on rules
Load Balancer	Configure separately	Built-in integration
Management	Individual VM management	Centralized updates
Updates	Manual for each VM	Rolling upgrades built-in
Best For	Traditional apps, full control	Modern apps, elastic workloads

---

Workshop Completion Checklist

□ Created VMSS with autoscaling via Portal
□ Tested load balancing across instances
□ Triggered autoscaling with load test
□ Performed rolling upgrade with zero downtime

□ Created availability set with fault domains via CLI

□ Deployed and configured multiple VMs

□ Added manual load balancer configuration

□ Verified fault domain distribution

□ Understood key differences between approaches

□ Cleaned up all resources

Additional Practice

Challenge: Create a VMSS using Azure CLI with the same configuration as Lab 1

- Hint: Use `az vmss create` command
- Add autoscale settings with `az monitor autoscale`
- Compare the number of commands needed vs Portal approach