Azure Infrastructure Workshop: Manual Deployment Guide

Workshop Overview

This hands-on workshop guides you through manually creating a complete Azure infrastructure including Virtual Network, Virtual Machines, Web Application deployment, Load Balancer, and Backup configuration using Storage Blobs.

Duration: 3-4 hours
Skill Level: Beginner to Intermediate
Prerequisites: Active Azure subscription with Contributor access

---

Architecture Overview

You will build:

- Virtual Network with subnets
- 2 Virtual Machines running web applications
- Azure Load Balancer for traffic distribution
- Storage Account for backups
- Network Security Groups for security

Part 1: Create Virtual Network (VNet)

Step 1.1: Navigate to Virtual Networks

1. Sign in to Azure Portal (portal.azure.com)
2. Click **Create a resource** (+ icon in top-left)
3. Search for **Virtual Network**
4. Click **Create**

Step 1.2: Configure Basic Settings

1. **Subscription:** Select your subscription
2. **Resource Group:** Click **Create new**
   - Name: `rg-workshop-demo`
   - Click **OK**

3. **Name:** `vnet-workshop`
4. **Region:** Select `East US` (or your preferred region)
5. Click **Next: IP Addresses**

Step 1.3: Configure IP Address Space

1. **IPv4 address space:** Keep default `10.0.0.0/16`
2. Click **+ Add subnet**
   - **Subnet name:** `subnet-web`
   - **Subnet address range:** `10.0.1.0/24`
   - Click **Add**
3. Click **+ Add subnet** again
   - **Subnet name:** `subnet-backend`
   - **Subnet address range:** `10.0.2.0/24`
   - Click **Add**
4. Click **Next: Security**

Step 1.4: Security Settings

1. Leave all security features as default (disabled for workshop)
2. Click **Review + create**
3. Review settings and click **Create**
4. Wait for deployment to complete (1-2 minutes)

Part 2: Create Network Security Group (NSG)

Step 2.1: Create NSG

1. Click **Create a resource**
2. Search for **Network Security Group**
3. Click **Create**

Step 2.2: Configure NSG

1. **Subscription:** Your subscription
2. **Resource Group:** `rg-workshop-demo`

3. **Name:** `nsg-web-servers`
4. **Region:** Same as VNet (`East US`)
5. Click **Review + create**
6. Click **Create**

Step 2.3: Add Inbound Security Rules

1. After deployment, click **Go to resource**
2. In left menu, click **Inbound security rules**
3. Click + **Add**

Rule 1 - Allow HTTP:

- **Source:** Any
- **Source port ranges:** *
- **Destination:** Any
- **Service:** HTTP
- **Destination port ranges:** 80
- **Protocol:** TCP
- **Action:** Allow
- **Priority:** 100
- **Name:** `Allow-HTTP`
- Click **Add**

Rule 2 - Allow SSH: 4. Click + **Add** again

- **Source:** Any
- **Source port ranges:** *
- **Destination:** Any
- **Service:** SSH
- **Destination port ranges:** 22
- **Protocol:** TCP
- **Action:** Allow
- **Priority:** 110
- **Name:** `Allow-SSH`

- Click **Add**

Step 2.4: Associate NSG with Subnet

1. In left menu, click **Subnets**
2. Click **+ Associate**
3. **Virtual network:** Select `vnet-workshop`
4. **Subnet:** Select `subnet-web`
5. Click **OK**

Part 3: Create First Virtual Machine (Web Server 1)

Step 3.1: Start VM Creation

1. Click **Create a resource**
2. Search for **Virtual Machine**
3. Click **Create**

Step 3.2: Basic Configuration

1. **Subscription:** Your subscription
2. **Resource Group:** `rg-workshop-demo`
3. **Virtual machine name:** `vm-web-01`
4. **Region:** Same as VNet (`East US`)
5. **Availability options:** No infrastructure redundancy required
6. **Security type:** Standard
7. **Image:** Ubuntu Server 22.04 LTS - Gen2
8. **Size:** Click **See all sizes**
    - Select **B2s** (2 vCPUs, 4 GB RAM)
    - Click **Select**

Step 3.3: Administrator Account

1. **Authentication type:** Password
2. **Username:** `azureuser`
3. **Password:** Create a strong password (save this!)

4. **Confirm password:** Re-enter password

Step 3.4: Inbound Port Rules

1. **Public inbound ports:** Allow selected ports
2. **Select inbound ports:** Check both **HTTP (80)** and **SSH (22)**

Step 3.5: Disk Configuration

1. Click **Next: Disks**
2. **OS disk type:** Standard SSD
3. Click **Next: Networking**

Step 3.6: Networking Configuration

1. **Virtual network:** Select `vnet-workshop`
2. **Subnet:** Select `subnet-web (10.0.1.0/24)`
3. **Public IP:** (new) vm-web-01-ip
4. **NIC network security group:** Basic
5. Click **Next: Management**

Step 3.7: Management Settings

1. Scroll down to **Boot diagnostics**
2. Enable **Boot diagnostics**
3. Click **Review + create**
4. Review all settings
5. Click **Create**

Step 3.8: Wait for Deployment

Wait 3-5 minutes for VM to deploy. Click **Go to resource** when complete.

Part 4: Install Web Application on VM1

Step 4.1: Connect to VM

1. On the VM overview page, note the **Public IP address**

2. Open your SSH client (Terminal on Mac/Linux, PuTTY on Windows)
3. Connect using: `ssh azureuser@<PUBLIC_IP>`
4. Enter your password when prompted
5. Type `yes` when asked about host authenticity

Step 4.2: Update System

bash

```
sudo apt update && sudo apt upgrade -y
```

Step 4.3: Install Nginx Web Server

bash

```
sudo apt install nginx -y
```

Step 4.4: Create Custom Web Page

bash

```
sudo bash -c 'cat > /var/www/html/index.html << EOF
<!DOCTYPE html>
<html>
<head>
    <title>Web Server 1</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            padding: 50px;
            background-color: #4CAF50;
            color: white;
        }
    </style>
</head>
<body>
    <h1>Web Server 1</h1>
    <h2>Workshop Demo Application</h2>
    <p>Server: vm-web-01</p>
</body>
</html>
EOF'
```

Step 4.5: Start Nginx

```bash
sudo systemctl start nginx
sudo systemctl enable nginx
```

Step 4.6: Verify Installation

1. Open a web browser
2. Navigate to: `http://<VM_PUBLIC_IP>`
3. You should see your custom web page

Part 5: Create Second Virtual Machine (Web Server 2)

Step 5.1: Repeat VM Creation

Follow the same steps as Part 3, but with these changes:

- **Virtual machine name:** `vm-web-02`
- **Public IP:** (new) vm-web-02-ip
- All other settings remain the same

Step 5.2: Install Web Application on VM2

1. SSH to the second VM: `ssh azureuser@<VM2_PUBLIC_IP>`
2. Run the same update and Nginx installation commands
3. Create a different web page:

```bash
```

```
sudo bash -c 'cat > /var/www/html/index.html << EOF
<!DOCTYPE html>
<html>
<head>
    <title>Web Server 2</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            padding: 50px;
            background-color: #2196F3;
            color: white;
        }
    </style>
</head>
<body>
    <h1>Web Server 2</h1>
    <h2>Workshop Demo Application</h2>
    <p>Server: vm-web-02</p>
</body>
</html>
EOF'
```

4. Start Nginx:

```bash
sudo systemctl start nginx
sudo systemctl enable nginx
```

Part 6: Create Azure Load Balancer

Step 6.1: Start Load Balancer Creation

1. In Azure Portal, click **Create a resource**
2. Search for **Load Balancer**
3. Click **Create**

Step 6.2: Basic Configuration

1. **Subscription:** Your subscription
2. **Resource Group:** `rg-workshop-demo`

3. **Name:** `lb-workshop`
4. **Region:** Same as VMs (`East US`)
5. **SKU:** Standard
6. **Type:** Public
7. **Tier:** Regional

Step 6.3: Frontend IP Configuration

1. Click **Next: Frontend IP configuration**
2. Click + **Add a frontend IP**
3. **Name:** `lb-frontend`
4. **IP type:** IP address
5. **Public IP address:** Click **Create new**
    - **Name:** `lb-public-ip`
    - **SKU:** Standard
    - **Assignment:** Static
    - Click **OK**
6. Click **Add**

Step 6.4: Backend Pool Configuration

1. Click **Next: Backend pools**
2. Click + **Add a backend pool**
3. **Name:** `lb-backend-pool`
4. **Virtual network:** Select `vnet-workshop`
5. **Backend Pool Configuration:** NIC
6. Click + **Add** under IP configurations

Add VM1:

- **Virtual machine:** Select `vm-web-01`
- **Network IP configuration:** Select the IP config
- Click **Add**

Add VM2: 7. Click + **Add** again

- **Virtual machine:** Select `vm-web-02`
- **Network IP configuration:** Select the IP config
- Click **Add**

8. Click **Save**

Step 6.5: Health Probe Configuration

1. Click **Next: Inbound rules**
2. Under **Health probes**, click **+ Add a health probe**
3. **Name:** `http-health-probe`
4. **Protocol:** HTTP
5. **Port:** 80
6. **Path:** /
7. **Interval:** 5 seconds
8. **Unhealthy threshold:** 2
9. Click **Add**

Step 6.6: Load Balancing Rule

1. Under **Load balancing rules**, click **+ Add a load balancing rule**
2. **Name:** `http-lb-rule`
3. **IP Version:** IPv4
4. **Frontend IP address:** Select `lb-frontend`
5. **Backend pool:** Select `lb-backend-pool`
6. **Protocol:** TCP
7. **Port:** 80
8. **Backend port:** 80
9. **Health probe:** Select `http-health-probe`
10. **Session persistence:** None
11. **Idle timeout:** 4 minutes
12. Click **Add**

Step 6.7: Complete Creation

1. Click **Review + create**
2. Review all settings
3. Click **Create**
4. Wait for deployment (2-3 minutes)

Step 6.8: Test Load Balancer

1. Click **Go to resource**
2. Note the **Frontend public IP address**
3. Open browser and navigate to: `http://<LOAD_BALANCER_IP>`
4. Refresh the page multiple times
5. You should see traffic alternating between Web Server 1 (green) and Web Server 2 (blue)

Part 7: Create Storage Account for Backups

Step 7.1: Start Storage Account Creation

1. Click **Create a resource**
2. Search for **Storage account**
3. Click **Create**

Step 7.2: Basic Configuration

1. **Subscription:** Your subscription
2. **Resource Group:** `rg-workshop-demo`
3. **Storage account name:** `saworkshopbackup` + random numbers (must be globally unique, use lowercase letters and numbers only)
4. **Region:** Same as other resources (`East US`)
5. **Performance:** Standard
6. **Redundancy:** Locally-redundant storage (LRS)

Step 7.3: Advanced Settings

1. Click **Next: Advanced**

2. Keep all default settings
3. Click **Next: Networking**

Step 7.4: Networking Settings

1. **Network access:** Enable public access from all networks
2. Click **Next: Data protection**

Step 7.5: Data Protection

1. Keep default settings
2. Click **Review + create**
3. Click **Create**
4. Wait for deployment

Part 8: Create Blob Container for Backups

Step 8.1: Access Storage Account

1. After deployment, click **Go to resource**
2. In left menu, click **Containers** (under Data storage)

Step 8.2: Create Container

1. Click **+ Container**
2. **Name:** `vm-backups`
3. **Public access level:** Private (no anonymous access)
4. Click **Create**

Part 9: Manual Backup Configuration

Step 9.1: Create Backup Script on VM1

1. SSH to VM1: `ssh azureuser@<VM1_PUBLIC_IP>`
2. Create a backup directory:

```
bash
```

```bash
mkdir -p ~/backups
```

3. Create backup script:

```bash
nano ~/backup-web.sh
```

4. Paste this script:

```bash
#!/bin/bash
DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_FILE="website_backup_$DATE.tar.gz"
tar -czf ~/backups/$BACKUP_FILE /var/www/html /etc/nginx
echo "Backup created: $BACKUP_FILE"
ls -lh ~/backups/$BACKUP_FILE
```

5. Save file (Ctrl+O, Enter, Ctrl+X)
6. Make executable:

```bash
chmod +x ~/backup-web.sh
```

Step 9.2: Test Backup Creation

```bash
~/backup-web.sh
```

Step 9.3: Install Azure CLI

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

Step 9.4: Upload Backup to Azure Storage

1. In Azure Portal, go to your Storage Account
2. Click **Access keys** in left menu

3. Click **Show** next to key1
4. Copy the **Connection string**
5. Back in SSH session, configure Azure Storage:

```
bash

export AZURE_STORAGE_CONNECTION_STRING="<paste_connection_string_here>"
```

6. Upload backup:

```
bash

az storage blob upload \
  --container-name vm-backups \
  --name website_backup_$(date +%Y%m%d).tar.gz \
  --file ~/backups/*.tar.gz
```

Step 9.5: Verify Upload

1. In Azure Portal, go to Storage Account
2. Click **Containers**
3. Click **vm-backups**
4. You should see your backup file listed

Part 10: Verify Complete Architecture

Verification Checklist

Virtual Network:

□ VNet `vnet-workshop` exists with address space 10.0.0.0/16
□ Subnet `subnet-web` exists with 10.0.1.0/24
□ Subnet `subnet-backend` exists with 10.0.2.0/24

Network Security:

□ NSG `nsg-web-servers` allows HTTP (port 80)
□ NSG `nsg-web-servers` allows SSH (port 22)
□ NSG is associated with `subnet-web`

Virtual Machines:

□ VM1 `vm-web-01` is running
□ VM2 `vm-web-02` is running
□ Both VMs have Nginx installed and running
□ Both VMs display different web pages

Load Balancer:

□ Load Balancer `lb-workshop` exists
□ Backend pool contains both VMs
□ Health probe is configured for port 80
□ Load balancing rule distributes HTTP traffic
□ Accessing LB IP shows alternating servers

Storage and Backup:

□ Storage Account exists
□ Container `vm-backups` exists
□ Backup file uploaded successfully

Part 11: Clean Up Resources (Optional)

When finished with the workshop:

1. Go to **Resource Groups**
2. Click `rg-workshop-demo`
3. Click **Delete resource group**
4. Type the resource group name to confirm
5. Click **Delete**

This removes all resources and stops billing.

Troubleshooting Tips

Cannot connect to VM via SSH:

- Verify NSG rules allow port 22
- Check VM is running
- Verify correct public IP address
- Ensure you're using correct username/password

Web page not loading:

- Check Nginx status: `sudo systemctl status nginx`
- Verify NSG allows port 80
- Check VM network interface has public IP

Load Balancer not distributing traffic:

- Verify both VMs are in backend pool
- Check health probe status in LB overview
- Ensure both VMs have Nginx running
- Wait 2-3 minutes for health probes to initialize

Backup upload fails:

- Verify connection string is correct
- Check storage account access keys
- Ensure container name matches exactly
- Verify Azure CLI is installed correctly

Workshop Summary

Congratulations! You have successfully:

- Created a Virtual Network with multiple subnets
- Deployed Network Security Groups with custom rules
- Created two Virtual Machines running web applications
- Configured an Azure Load Balancer for traffic distribution
- Set up Azure Storage for backups
- Manually backed up application data to blob storage

This architecture demonstrates a basic but complete Azure infrastructure setup suitable for small web applications with redundancy and backup capabilities.