Azure Multi-Web App Setup Guide

Overview

This guide shows how to host multiple web applications on a single Azure VM with different access levels:

- **Public App**: Accessible from anywhere on the internet
- **Internal App**: Accessible only from internal network/VPN

Prerequisites

- Azure subscription
- Basic understanding of Linux/Windows Server
- Domain name (optional but recommended)

Step 1: Create Azure Virtual Machine

1.1 Create VM through Azure Portal

```
# Or use Azure CLI
az vm create \
  --resource-group myResourceGroup \
  --name myWebServerVM \
  --image Ubuntu2204 \
  --admin-username azureuser \
  --generate-ssh-keys \
  --size Standard_B2s
```

1.2 VM Configuration

- **Size**: Standard_B2s (2 vCPUs, 4 GB RAM) - good for learning

- **OS**: Ubuntu 22.04 LTS (recommended for students)
- **Authentication**: SSH public key
- **Ports**: Open 22 (SSH), 80 (HTTP), 443 (HTTPS)

Step 2: Configure Network Security Groups (NSG)

2.1 Create Custom NSG Rules

```
{
  "Public Web App Rules": [
    {
      "name": "Allow-HTTP-Public",
      "priority": 100,
      "access": "Allow",
      "protocol": "TCP",
      "direction": "Inbound",
      "sourceAddressPrefix": "*",
      "destinationPortRange": "8080"
    },
    {
      "name": "Allow-HTTPS-Public",
      "priority": 101,
      "access": "Allow",
      "protocol": "TCP",
      "direction": "Inbound",
      "sourceAddressPrefix": "*",
      "destinationPortRange": "8443"
    }
  ],
  "Internal Web App Rules": [
    {
      "name": "Allow-HTTP-Internal",
      "priority": 200,
      "access": "Allow",
      "protocol": "TCP",
```

```
      "direction": "Inbound",
      "sourceAddressPrefix": "10.0.0.0/8",
      "destinationPortRange": "9080"
    },
    {
      "name": "Allow-VPN-Access",
      "priority": 201,
      "access": "Allow",
      "protocol": "TCP",
      "direction": "Inbound",
      "sourceAddressPrefix": "192.168.1.0/24",
      "destinationPortRange": "9080"
    }
  ]
}
```

Step 3: Install and Configure Web Server

3.1 Install Nginx

```
# Update system
sudo apt update && sudo apt upgrade -y

# Install Nginx
sudo apt install nginx -y

# Install Node.js (for demo apps)
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt install nodejs -y

# Install PM2 for process management
sudo npm install -g pm2
```

3.2 Configure Nginx Virtual Hosts

```
# /etc/nginx/sites-available/public-app
server {
    listen 8080;
    server_name your-domain.com;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_cache_bypass $http_upgrade;
    }
}

# /etc/nginx/sites-available/internal-app
server {
    listen 9080;
    server_name internal.your-domain.com;

    # Restrict access to internal networks only
    allow 10.0.0.0/8;
    allow 192.168.0.0/16;
    allow 172.16.0.0/12;
    deny all;

    location / {
        proxy_pass http://localhost:3001;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
```

```
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_cache_bypass $http_upgrade;
    }
}
```

3.3 Enable Sites

```
# Enable the sites
sudo ln -s /etc/nginx/sites-available/public-app /etc/nginx/sites-enabled/
sudo ln -s /etc/nginx/sites-available/internal-app /etc/nginx/sites-enabled/

# Test configuration
sudo nginx -t

# Restart Nginx
sudo systemctl restart nginx
```

Step 4: Create Sample Web Applications

4.1 Public Web App (Port 3000)

```
// /home/azureuser/public-app/app.js
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {
    res.send(`
        <h1>Public Web Application</h1>
        <p>This app is accessible from anywhere on the internet!</p>
        <p>Server Time: ${new Date()}</p>
        <p>Your IP: ${req.ip}</p>
```

```
  `);
});

app.listen(port, 'localhost', () => {
  console.log(`Public app running on http://localhost:${port}`);
});
```

4.2 Internal Web App (Port 3001)

```
// /home/azureuser/internal-app/app.js
const express = require('express');
const app = express();
const port = 3001;

app.get('/', (req, res) => {
  res.send(`
    <h1>Internal Web Application</h1>
    <p>This app is only accessible from internal networks!</p>
    <p>Server Time: ${new Date()}</p>
    <p>Your IP: ${req.ip}</p>
    <p>🔒 Secure Internal Access Only</p>
  `);
});

app.get('/admin', (req, res) => {
  res.json({
    message: "Admin dashboard",
    users: ["admin", "user1", "user2"],
    serverStatus: "healthy"
  });
});

app.listen(port, 'localhost', () => {
```

```
    console.log(`Internal app running on http://localhost:${port}`);
});
```

4.3 Install Dependencies and Start Apps

```
# Create directories
mkdir -p /home/azureuser/public-app
mkdir -p /home/azureuser/internal-app

# Initialize package.json for both apps
cd /home/azureuser/public-app
npm init -y
npm install express

cd /home/azureuser/internal-app
npm init -y
npm install express

# Start apps with PM2
pm2 start /home/azureuser/public-app/app.js --name "public-app"
pm2 start /home/azureuser/internal-app/app.js --name "internal-app"

# Save PM2 configuration
pm2 save
pm2 startup
```

Step 5: Configure Firewall (UFW)

```
# Enable UFW
sudo ufw enable

# Allow SSH
sudo ufw allow 22
```

```
# Allow public app (accessible from anywhere)
sudo ufw allow 8080

# Allow internal app (will be restricted by Nginx)
sudo ufw allow 9080

# Check status
sudo ufw status
```

Step 6: SSL/TLS Configuration (Optional)

6.1 Install Certbot

```
sudo apt install certbot python3-certbot-nginx -y

# Get SSL certificate for public domain
sudo certbot --nginx -d your-domain.com

# Update Nginx config for HTTPS on port 8443
```

Step 7: Access Configuration

7.1 Public App Access

- **URL**: `http://your-vm-ip:8080` or `http://your-domain.com:8080`
- **Accessible from**: Anywhere on the internet
- **Use case**: Customer-facing website, blog, portfolio

7.2 Internal App Access

- **URL**: `http://your-vm-ip:9080` (only from internal networks)
- **Accessible from**:
  - Same VNet: `10.0.0.0/8`

    - VPN users: `192.168.1.0/24`
    - Corporate network: Define your IP ranges
- **Use case**: Admin panel, internal tools, company intranet

Step 8: Testing Access Controls

8.1 Test Public Access

```
# From any internet connection
curl http://your-vm-ip:8080
# Should work ✅
```

8.2 Test Internal Access

```
# From internet (should fail)
curl http://your-vm-ip:9080
# Should return 403 Forbidden ❌

# From internal network/VPN (should work)
curl http://internal-vm-ip:9080
# Should work ✅
```

Step 9: Monitoring and Logs

9.1 Check Application Status

```
# PM2 status
pm2 status

# Nginx logs
sudo tail -f /var/log/nginx/access.log
sudo tail -f /var/log/nginx/error.log
```

```
# Application logs
pm2 logs
```

9.2 Azure Monitoring

- Enable **Azure Monitor** for VM metrics
- Set up **Network Watcher** for traffic analysis
- Configure **Log Analytics** for centralized logging

Security Best Practices

1. **Regular Updates**: Keep OS and packages updated
2. **Strong Authentication**: Use SSH keys, disable password auth
3. **Least Privilege**: Only open necessary ports
4. **VPN Access**: Set up Azure VPN Gateway for internal access
5. **Backup Strategy**: Regular snapshots of VM disk
6. **Monitoring**: Set up alerts for unusual traffic patterns

Troubleshooting Common Issues

Port Not Accessible

- Check NSG rules in Azure portal
- Verify UFW firewall rules
- Confirm Nginx configuration
- Test with `telnet vm-ip port`

403 Forbidden on Internal App

- Verify client IP is in allowed ranges
- Check Nginx access logs for client IP
- Update allow/deny rules in Nginx config

App Not Starting

- Check PM2 logs: `pm2 logs`
- Verify Node.js app syntax
- Check port conflicts: `netstat -tulpn`