Windows Server Deployment Guide on AWS Cloud

Sou Chanrojame, Orn Pheakdey, Long Neron, Then Sivthean, Le Sreyma

December 19, 2025

Abstract

This document provides a comprehensive step-by-step guide for deploying various Windows Server roles and services on the AWS Cloud infrastructure, including configuration details for EC2 instances, security groups, and storage optimization.

Contents

Prerequisites

AWS Account Setup

- Active AWS account with appropriate permissions
- VPC configured with public and private subnets
- Security groups properly configured
- Key pairs created for RDP access
- IAM roles for EC2 instances

General Windows Server Launch Steps

1. Navigate to EC2 Dashboard in AWS Console
2. Click “Launch Instance”
3. Select Windows Server AMI (2019/2022 recommended)
4. Choose instance type based on workload
5. Configure instance details (VPC, subnet, IAM role)
6. Add storage as needed
7. Configure security groups
8. Review and launch with key pair

1 File Server

1.1 AWS Configuration

Instance Name: File Server
Instance Type: t3.medium or larger
Storage: EBS volumes with provisioned IOPS for performance
Security Group Ports: tcp 445 (SMB), tcp 3389 (RDP)

1.2 Implementation

```
Install-WindowsFeature -Name FS-FileServer
New-Item -Path "C:\FileShare" -ItemType Directory -Force
New-SmbShare -Name "PublicShare" -Path "C:\FileShare" -FullAccess "Everyone"
echo hello > C:\FileShare\hello.txt
Write-Host "File Server setup complete."
```

1.3 Usage

- connect to Proxy Server or other windows server
- open file explorer and `\\<public dns of File Server>`
- enter credential

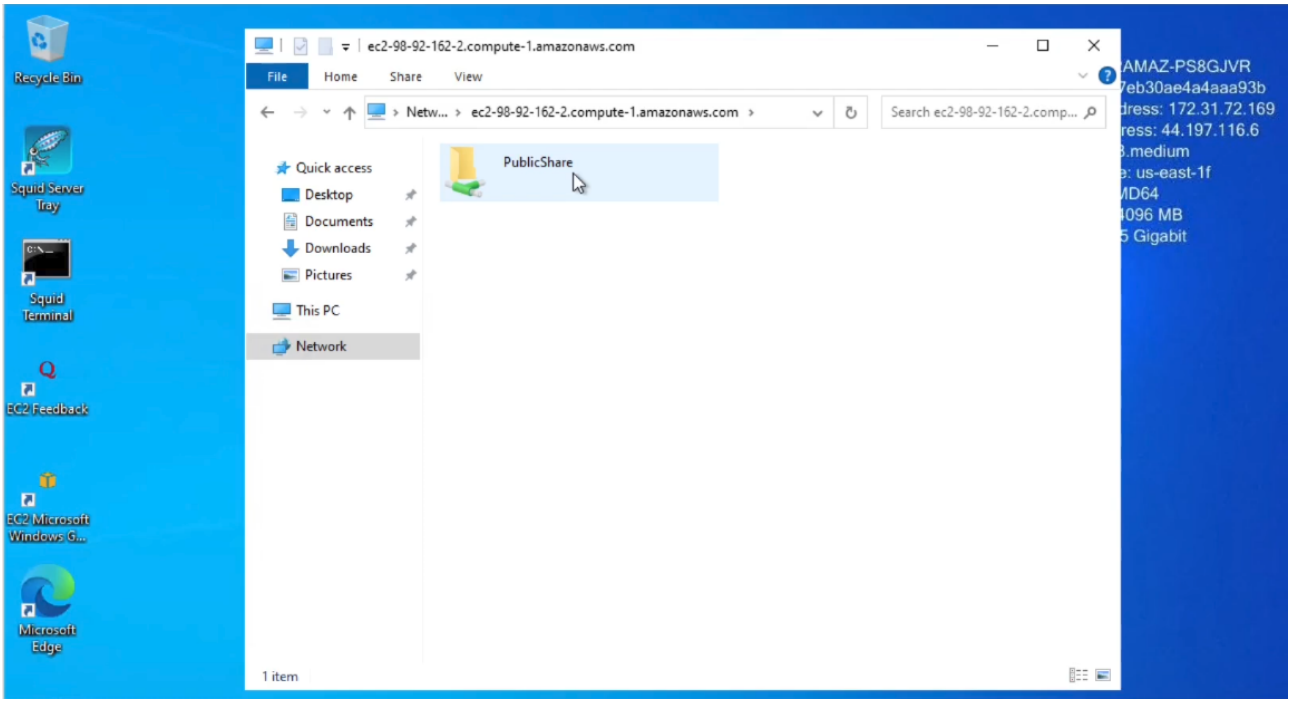

Figure 1: Successfully connected to file server

2 Proxy Server (Caching, Control Access)

2.1 AWS Configuration

Instance Name: Proxy Server
Instance Type: t3.medium
Security Group Ports: tcp 3128, tcp 3389 (RDP)

2.2 Implementation

```
Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::
    SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex
    ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/
    install.ps1'))
choco install squid -y
Invoke-WebRequest https://raw.githubusercontent.com/james5635/windows_server_assignment/
    refs/heads/main/config/squid.conf -OutFile "C:\Squid\etc\squid\squid.conf"
Restart-Service squidsrv
Write-Host "Proxy Server setup complete."
```

2.3 Usage

- connect to File Server or other windows server
- change proxy address to <private ip of Proxy Server> with port 3128
- open browser
- visit youtube.com => allow
- visit facebook.com => blocked

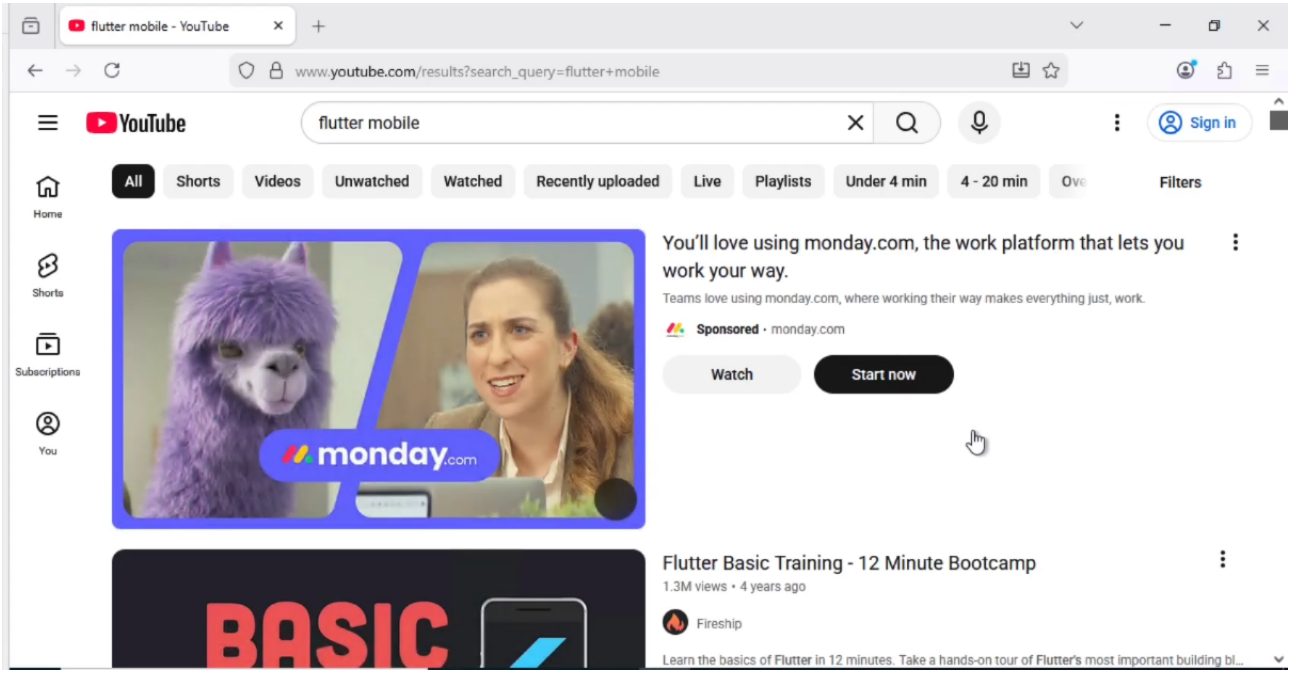

Figure 2: Allow access to youtube

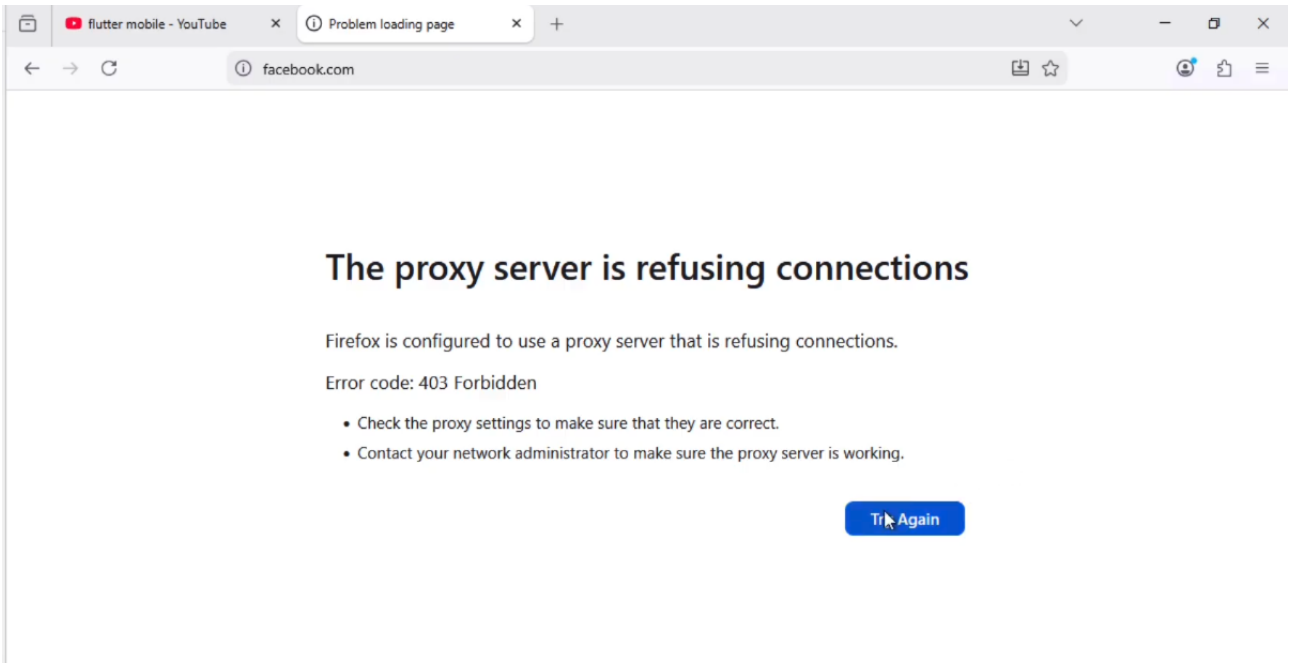

Figure 3: Block access to facebook

3 DNS Server

3.1 AWS Configuration

Instance Name: DNS Server
Instance Type: t3.medium
Security Group Ports: tcp/udp 53, tcp 3389 (RDP)

3.2 Implementation

```
Install-WindowsFeature DNS -IncludeManagementTools

Add-DnsServerPrimaryZone -Name "example.internal" -ZoneFile "example.internal.dns"
Add-DnsServerPrimaryZone -Name "devspeed.com" -ZoneFile "devspeed.com.dns"

Add-DnsServerResourceRecordA -Name 'server1' -ZoneName 'example.internal' -IPv4Address '10.0.1.10'
Add-DnsServerResourceRecordA -Name 'console' -ZoneName 'devspeed.com' -IPv4Address '192.168.1.11'
Add-DnsServerResourceRecordA -Name 'go' -ZoneName 'devspeed.com' -IPv4Address '192.168.1.29'
Add-DnsServerResourceRecordA -Name 'blog' -ZoneName 'devspeed.com' -IPv4Address '192.168.1.30'
Add-DnsServerResourceRecordA -Name 'shop' -ZoneName 'devspeed.com' -IPv4Address '192.168.1.31'
Add-DnsServerResourceRecordA -Name 'support' -ZoneName 'devspeed.com' -IPv4Address '192.168.1.32'
Add-DnsServerResourceRecordA -Name 'mail' -ZoneName 'devspeed.com' -IPv4Address '192.168.1.33'
Add-DnsServerResourceRecordA -Name 'www' -ZoneName 'devspeed.com' -IPv4Address '192.168.1.34'
Add-DnsServerResourceRecordA -Name 'www2' -ZoneName 'devspeed.com' -IPv4Address '192.168.1.100'

# For example, forwarding to Google DNS:
# Add-DnsServerForwarder -IPAddress "8.8.8.8" -PassThru
# Or forwarding to AWS VPC DNS (recommended for EC2). This is the Amazon-provided DNS resolver for VPCs.
# Add-DnsServerForwarder -IPAddress "169.254.169.253" -PassThru

New-NetFirewallRule -DisplayName "Allow DNS TCP 53" -Direction Inbound -Protocol TCP -LocalPort 53 -Action Allow
New-NetFirewallRule -DisplayName "Allow DNS UDP 53" -Direction Inbound -Protocol UDP -LocalPort 53 -Action Allow
```

3.3 Usage

- open Terminal
- dog www.devspeed.com ‘@<public ip of dns server>‘
- dog go.devspeed.com ‘@<public ip of dns server>‘
- nslookup shop.devspeed.com ‘<public ip of dns server>‘
- nslookup mail.devspeed.com ‘<public ip of dns server>‘

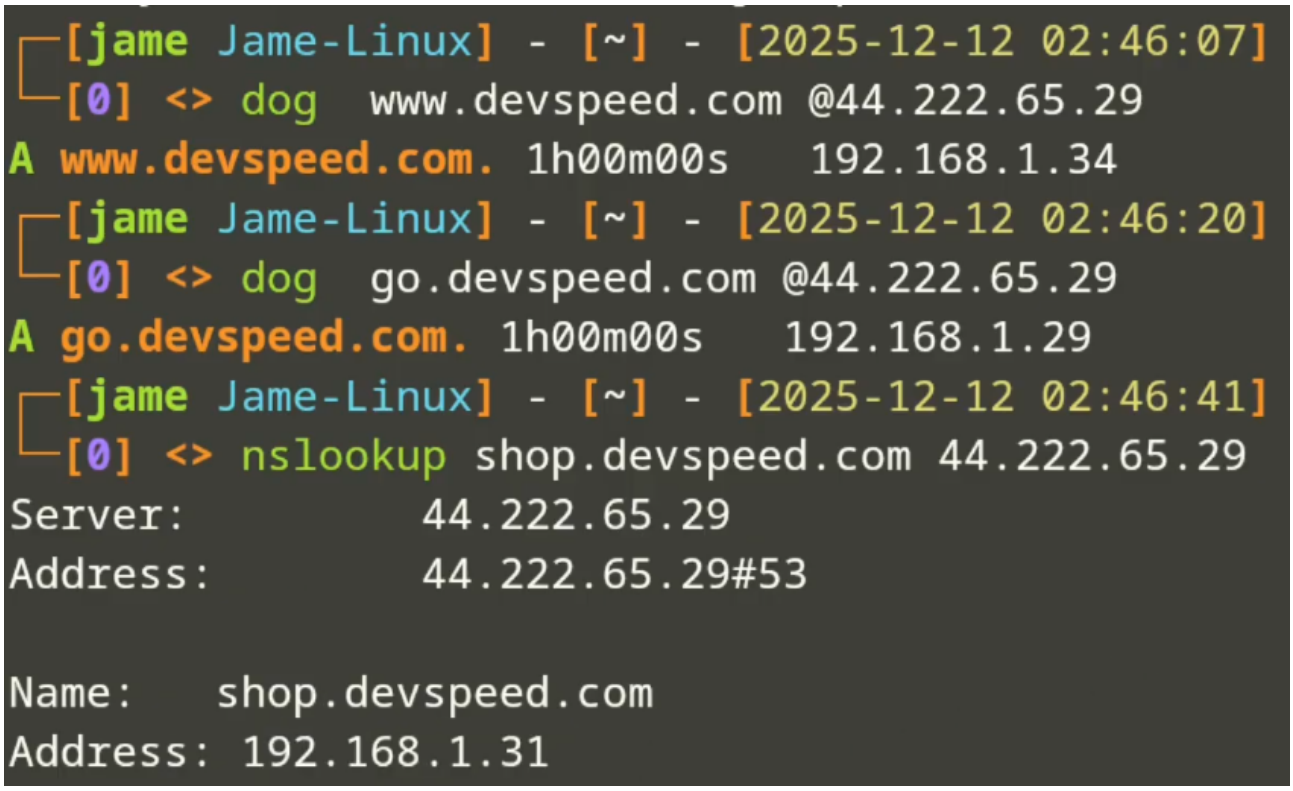

Figure 4: Dns server response with respective IP address

4 DHCP Server

4.1 AWS Configuration

Instance Type: t3.small
Note: AWS VPC provides DHCP by default; custom DHCP server is optional.

4.2 Implementation Steps

1. **Launch Windows Server Instance**

2. **Install DHCP Server Role**

```
Install-WindowsFeature -Name DHCP -IncludeManagementTools
Add-DhcpServerInDC -DnsName "dhcp.yourdomain.local"
```

3. **Configure DHCP Scope**

```
Add-DhcpServerv4Scope -Name "Internal Network" -StartRange 10.0.1.100 -EndRange
    10.0.1.200 -SubnetMask 255.255.255.0

Set-DhcpServerv4OptionValue -ScopeId 10.0.1.0 -Router 10.0.1.1
Set-DhcpServerv4OptionValue -ScopeId 10.0.1.0 -DnsServer 10.0.1.10
```

4. **Configure Reservations**

```
Add-DhcpServerv4Reservation -ScopeId 10.0.1.0 -IPAddress 10.0.1.50 -ClientId "
    00-11-22-33-44-55" -Description "Print Server"
```

5. **Authorize DHCP Server**

```
Add-DhcpServerInDC -DnsName "dhcp.yourdomain.local" -IPAddress 10.0.1.10
```

4.3 Best Practices

- Consider using AWS-provided DHCP for simplicity
- Deploy DHCP failover for redundancy
- Use DHCP policies for different device types
- Monitor DHCP lease utilization

5 VPN Server

5.1 AWS Configuration

Instance Name: VPN Server
Instance Type: t3.small to t3.medium
Security Group Ports: udp 1194, tcp 3389 (RDP)
Elastic IP: Required for consistent endpoint

5.2 Implementation

```

Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::
    SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex
    ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/
    install.ps1'))
choco install openvpn -y  --package-parameters=" /AddToDesktop /Gui /GuiOnLogon /EasyRsa
    /DcoDriver /TapDriver /WintunDriver /OpenSSL /Service "
choco install caddy 7zip -y

$cfg = "C:\Program Files\OpenVPN\config"

Invoke-WebRequest https://raw.githubusercontent.com/james5635/windows_server_assignment/
    refs/heads/main/config/openvpn/server/dh.pem -OutFile "$cfg\dh.pem"
Invoke-WebRequest https://raw.githubusercontent.com/james5635/windows_server_assignment/
    refs/heads/main/config/openvpn/server/server.crt -OutFile "$cfg\server.crt"
Invoke-WebRequest https://raw.githubusercontent.com/james5635/windows_server_assignment/
    refs/heads/main/config/openvpn/server/server.key -OutFile "$cfg\server.key"
Invoke-WebRequest https://raw.githubusercontent.com/james5635/windows_server_assignment/
    refs/heads/main/config/openvpn/server/ca.crt -OutFile "$cfg\ca.crt"
Invoke-WebRequest https://raw.githubusercontent.com/james5635/windows_server_assignment/
    refs/heads/main/config/openvpn/server/server.ovpn -OutFile "$cfg\server.ovpn"

Set-Service -Name OpenVPNService -StartupType Automatic
Start-Service -Name OpenVPNService

New-NetFirewallRule -DisplayName "OpenVPN" -Direction Inbound -Protocol UDP -LocalPort
    1194 -Action Allow
New-NetFirewallRule -DisplayName "ShareFileOpenVPN" -Direction Inbound -Protocol TCP -
    LocalPort 80 -Action Allow

# don't know why it doesn't working. (The server doesn't run)
# sleep -Seconds 5 # still not working
# Start-Process "C:\Program Files\OpenVPN\bin\openvpn-gui.exe"

# # for client to openvpn server
# $cfg = "C:\Program Files\OpenVPN\config"
# Invoke-WebRequest <url> "$cfg\client1.ovpn"
# Invoke-WebRequest <url> "$cfg\ca.crt"
# Invoke-WebRequest <url> "$cfg\client1.crt"
# Invoke-WebRequest <url> "$cfg\client1.key"
```

5.3 Usage

- connect to VPN Server
- Open OpenVPN GUI to start the server
- connect to File Server

- change `YOUR_PUBLIC_IP` in `C:\Program Files\OpenVPN\config\client1.ovpn` to public ip of the VPN Server
- Open `OpenVPN GUI` to connect to the server
- Open powershell and type `ipconfig` and will see something like:

```
Unknown adapter OpenVPN Data Channel Offload:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::f729:5f67:58f2:7253%17
   IPv4 Address. . . . . . . . . . . : 10.8.0.6
   Subnet Mask . . . . . . . . . . . : 255.255.255.252
   Default Gateway . . . . . . . . . :
```

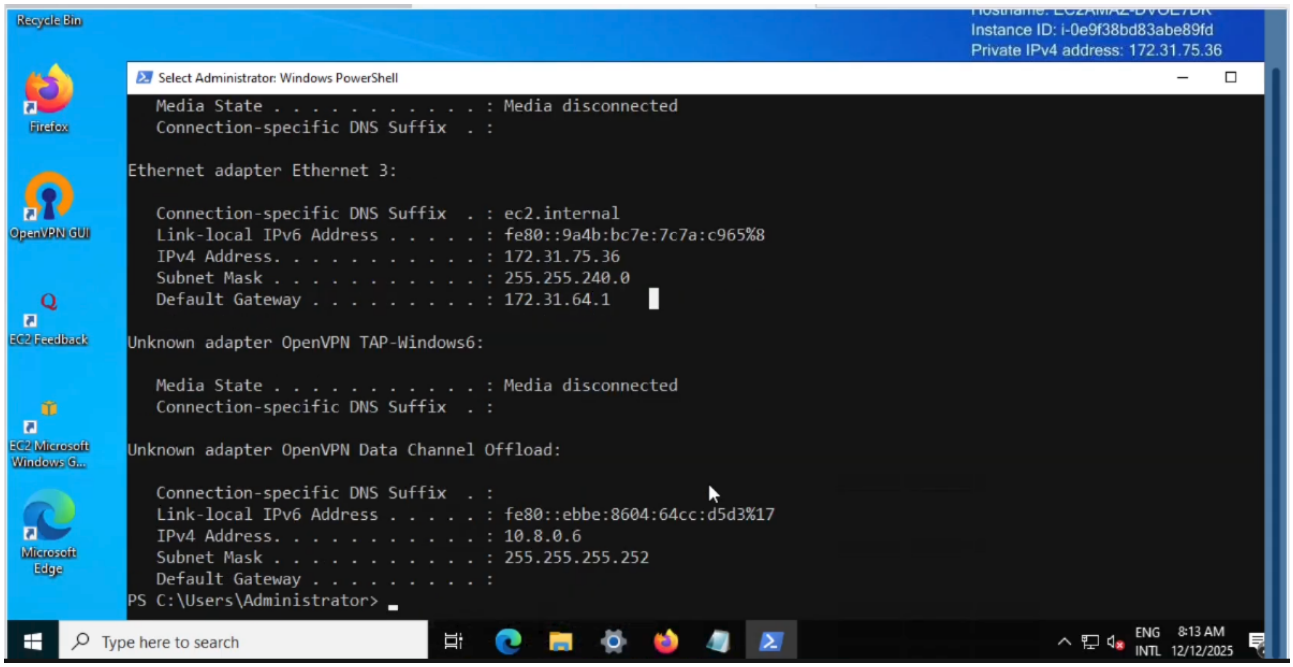

Figure 5: vpn client successfully connect to VPN server

6 Terminal Server (Thin Clients)

6.1 AWS Configuration

Instance Name: Terminal Server
Instance Type: t3.medium
Security Group Ports: tcp 3389 (RDP)

6.2 Implementation

```
# Install Remote Desktop Session Host Role
Install-WindowsFeature RDS-RD-Server
# Enable Multiple Sessions
Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server" -Name
    fSingleSessionPerUser -Value 0
# Allow RDP Firewall Rule
Enable-NetFirewallRule -DisplayGroup "Remote Desktop"
Write-Host "Terminal Server (RDS Session Host) setup complete."
```

6.3 Usage

- connect to the server with RDP

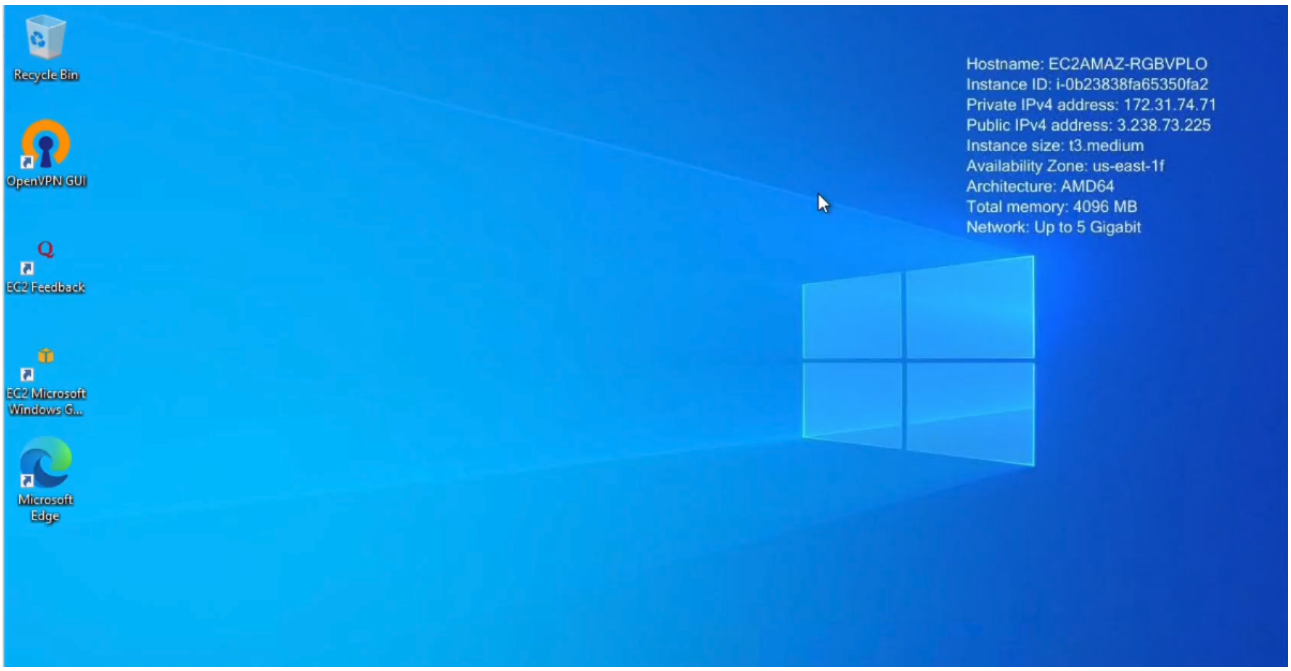

Figure 6: Successfully connect to terminal server

7 Web Server

7.1 AWS Configuration

Instance Name: Web Server
Instance Type: t3.medium
Security Group Ports: tcp 80 (HTTP), tcp 443 (HTTPS), tcp 3389 (RDP)

7.2 Implementation

```
Install-WindowsFeature -Name Web-Server -IncludeManagementTools
# Install-WindowsFeature -Name Web-Asp-Net45, Web-Net-Ext45
# Install-WindowsFeature -Name Web-Mgmt-Console
Invoke-WebRequest https://raw.githubusercontent.com/james5635/windows_server_assignment/
    refs/heads/main/static/devspeed.html -OutFile "C:\inetpub\wwwroot\index.html"
```

7.3 Usage

- visit public dns of the server

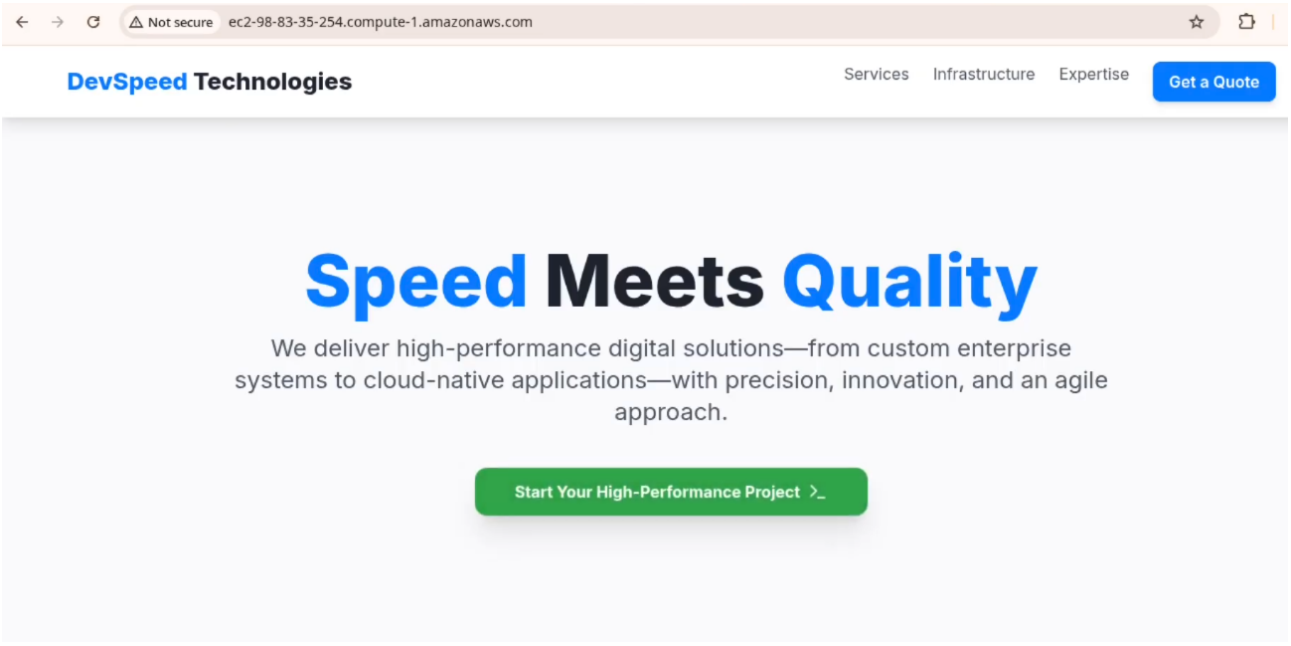

Figure 7: visiting the web page

8 Mail Server

8.1 AWS Configuration

Instance Type: t3.medium
Security Group Ports: 25 (SMTP), 110 (POP3), 143 (IMAP), 587 (Submission), 993 (IMAPS), 995 (POP3S)
Elastic IP: Required
Note: AWS blocks port 25 by default; request removal.

8.2 Implementation

```
# mail server
# - apache james (smpt server [port: 25], imap server [port: 143])
# - swaks (smpt client)
# - thunderbird (email client)

Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))
choco install openjdk strawberryperl thunderbird notepadplusplus telnet -y
Invoke-WebRequest https://www.jetmore.org/john/code/swaks/files/swaks-20240103.0/swaks -OutFile swaks.pl
Invoke-WebRequest https://dlcdn.apache.org/james/server/3.9.0/james-server-spring-app-3.9.0-app.zip -OutFile james-server-spring-app-3.9.0-app.zip
Expand-Archive james-server-spring-app-3.9.0-app.zip -DestinationPath james-server-spring-app-3.9.0-app

Invoke-WebRequest https://raw.githubusercontent.com/james5635/windows_server_assignment/refs/heads/main/config/smtpserver.xml -OutFile "james-server-spring-app-3.9.0-app\james-server-spring-app-3.9.0\conf\smtpserver.xml"
Invoke-WebRequest https://raw.githubusercontent.com/james5635/windows_server_assignment/refs/heads/main/config/imapserver.xml -OutFile "james-server-spring-app-3.9.0-app\james-server-spring-app-3.9.0\conf\imapserver.xml"

Import-Module $env:ChocolateyInstall\helpers\chocolateyProfile.psm1
refreshenv

# cd "james-server-spring-app-3.9.0-app\james-server-spring-app-3.9.0\bin\"

$james = "C:\Windows\System32\james-server-spring-app-3.9.0-app\james-server-spring-app-3.9.0\bin\james.bat"
$james_cli = "C:\Windows\System32\james-server-spring-app-3.9.0-app\james-server-spring-app-3.9.0\bin\james-cli.ps1"

Invoke-WebRequest https://raw.githubusercontent.com/james5635/windows_server_assignment/refs/heads/main/scripts/james-cli.ps1 -OutFile $james_cli

function Wait-JamesReady {
    param(
        [int]$Port = 9999,
        [string]$Url = "localhost",
        [int]$TimeoutSeconds = 120
    )

    $start = Get-Date

    Write-Host "Waiting for James server to open JMX on port $Port..."

```

```
    while ((Get-Date) -lt $start.AddSeconds($TimeoutSeconds)) {
        if ((Test-NetConnection -ComputerName $Url -Port $Port -WarningAction
    SilentlyContinue).TcpTestSucceeded) {
            Write-Host "James server is ready!"
            return
        }

        Start-Sleep -Seconds 2
    }

    throw "Timeout: James server did not open JMX port $Port"
}

& $james install
& $james start

# need to wait james server completely ready
Wait-JamesReady

& $james_cli -username james-admin -password changeme adddomain example.local
& $james_cli -username james-admin -password changeme adduser mike@example.local mike
& $james_cli -username james-admin -password changeme adduser joe@example.local joe
& $james_cli -username james-admin -password changeme adduser leng@example.local leng
& $james_cli -username james-admin -password changeme adduser jame@example.local jame

New-NetFirewallRule -DisplayName "Allow smtp 25" -Direction Inbound -Protocol TCP -
    LocalPort 25 -Action Allow
New-NetFirewallRule -DisplayName "Allow imap 143" -Direction Inbound -Protocol TCP -
    LocalPort 143 -Action Allow
```

8.3 Usage

- Open Thunderbird
- Username: `jame@example.local`
- Password: `jame`
- Username: `mike@example.local`
- Password: `mike`
- IMAP
    - Hostname: `<public-ip-of-mail-server>`
    - Port: 143
- SMTP
    - Hostname: `<public-ip-of-mail-server>`
    - Port: 25

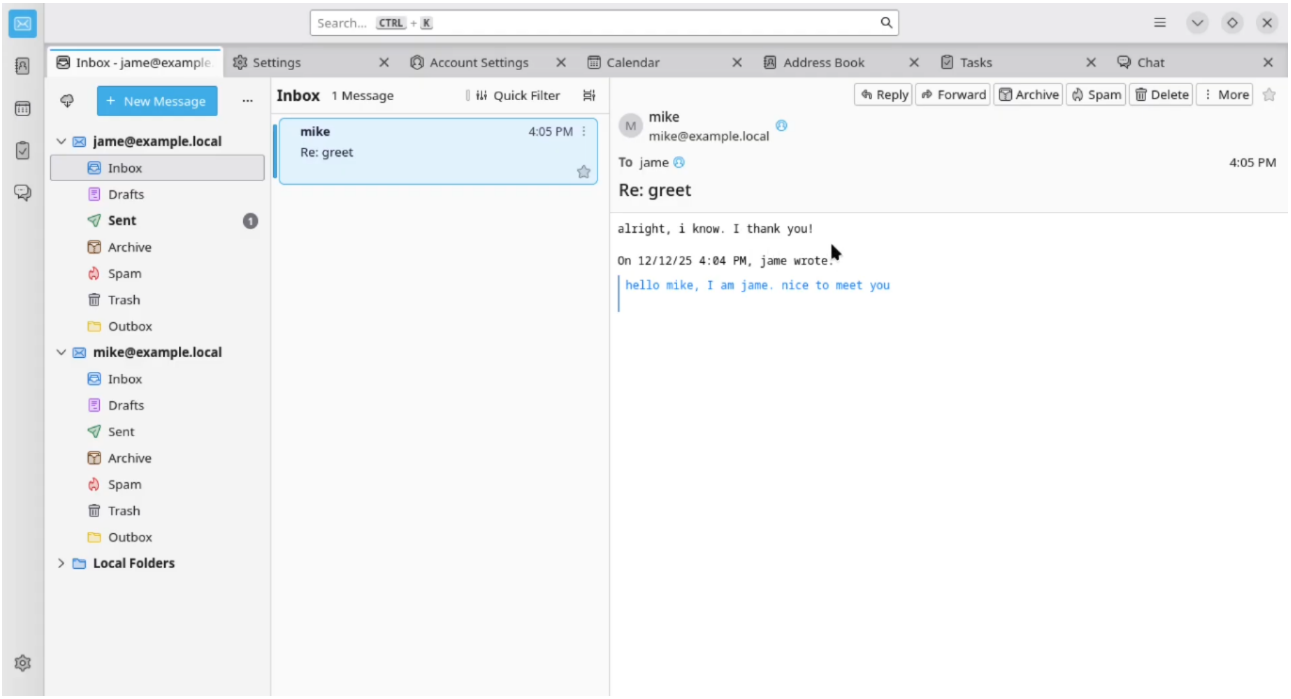

Figure 8: Two users sending messages to each other

9 Database Server

9.1 AWS Configuration

Instance Name: Database Server
Instance Type: t3.medium or larger (memory-optimized)
Storage: EBS with provisioned IOPS or io2
Security Group Ports:

- PostgreSQL: tcp 5432
- SQL Server: tcp 1433
- MongoDB: tcp 27017
- RDP: tcp 3389

9.2 Implementation

9.2.1 PostgreSQL

```
# postgresql
Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::
   SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex
   ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/
   install.ps1'))
# username postgres
choco install postgresql11 --params '/Password:test /AllowRemote' -y
Restart-Service postgresql-x64-11
New-NetFirewallRule -DisplayName "Allow PostgreSQL 5432" `
  -Direction Inbound `
  -Protocol TCP `
  -LocalPort 5432 `
  -Action Allow
```

9.2.2 SQL Server

```
# sql server
Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::
   SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex
   ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/
   install.ps1'))
choco install mssqlserver2014express -y
choco install sqlserver-cmdlineutils -y

# Get access to SqlWmiManagement DLL on the machine with SQL
# we are on, which is where SQL Server was installed.
# Note: This is installed in the GAC by SQL Server Setup.

# Load the WMI assembly
[System.Reflection.Assembly]::LoadWithPartialName('Microsoft.SqlServer.SqlWmiManagement'
   ) | Out-Null

# Connect to local SQL Server machine
$wmi = New-Object 'Microsoft.SqlServer.Management.Smo.Wmi.ManagedComputer' 'localhost'

# Get TCP protocol for SQLEXPRESS
$tcp = $wmi.ServerInstances['SQLEXPRESS'].ServerProtocols['Tcp']
```

```
$tcp.IsEnabled = $true

# Disable dynamic ports on each IP entry
foreach ($ip in $tcp.IPAddresses) {
    $ip.IPAddressProperties["TcpDynamicPorts"].Value = ""
}

# Set static port 1433 on ALL IPs (including IPAll)
foreach ($ip in $tcp.IPAddresses) {
    $ip.IPAddressProperties["TcpPort"].Value = "1433"
}

# Apply configuration
$tcp.Alter()

# You need to restart SQL Server for the change to persist
# -Force takes care of any dependent services, like SQL Agent.
# Note: If the instance is named, replace MSSQLSERVER with MSSQL$ followed by
# the name of the instance (e.g., MSSQL$MYINSTANCE)

Restart-Service -Name 'MSSQL$SQLEXPRESS' -Force

& "C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\110\Tools\Binn\SQLCMD.EXE"  -S
   .\SQLEXPRESS -Q "ALTER LOGIN sa ENABLE; ALTER LOGIN sa WITH PASSWORD = '
   mypassword@2025';"
Set-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Microsoft SQL Server\MSSQL12.SQLEXPRESS
   \MSSQLServer" -Name LoginMode -Value 2
Restart-Service -Name 'MSSQL$SQLEXPRESS' -Force

New-NetFirewallRule -DisplayName "Allow SQL Server 1433" `
  -Direction Inbound `
  -Protocol TCP `
  -LocalPort 1433 `
  -Action Allow
```

9.2.3 MongoDB

```
# mongodb
Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::
   SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex
   ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/
   install.ps1'))
choco install mongodb -y
choco install mongodb-shell -y
Invoke-WebRequest https://raw.githubusercontent.com/james5635/windows_server_assignment/
   refs/heads/main/config/mongod.cfg -OutFile "C:\Program Files\MongoDB\Server\8.2\bin\
   mongod.cfg"
Restart-Service MongoDB
New-NetFirewallRule -DisplayName "Allow MongoDB 27017" `
  -Direction Inbound `
  -Protocol TCP `
  -LocalPort 27017 `
  -Action Allow
```

9.3 Usage

- MongoDB
    - mongosh <public ip of mongodb server>

- SQL Server
    - /opt/mssql-tools18/bin/sqlcmd -S <public ip of sql server> -C -U sa -P mypassword@2025
- PostgreSQL
    - psql -h <public ip of postgresql server> -U postgres
    - password: test

```
3b4840fd2100:/# psql -h 98.92.51.253 -U postgres
Password for user postgres:
psql (17.6, server 11.22)
Type "help" for help.

postgres=# CREATE TABLE person(id INT, name VARCHAR(100));
CREATE TABLE
postgres=# INSERT INTO person VALUES(1, 'james');
INSERT 0 1
postgres=# SELECT * FROM person;
 id | name
----+-------
  1 | james
(1 row)
```

Figure 9: Client using postgresql server

```
mssql@5f1c740a77e7:/$ /opt/mssql-tools18/bin/sqlcmd -S 98.92.51.253 -C -U sa -P
mypassword@2025
1> CREATE TABLE person(id int, name varchar(100))
2> GO
1> INSERT INTO person VALUES(1, 'seakleng')
2> GO

(1 rows affected)
1> INSERT INTO person VALUES(2, 'vanthorn')
2> GO

(1 rows affected)
1> SELECT * FROM person
```

Figure 10: client using sql server

Figure 11: client using mongodb server

10 Backup Server

10.1 AWS Configuration

Instance Type: t3.medium
Storage: Large EBS volumes or S3 integration
IAM Role: Permissions for S3, EBS snapshots

10.2 Implementation Steps

1. **Launch Windows Server Instance**
2. **Install Windows Server Backup**

```
Install-WindowsFeature -Name Windows-Server-Backup -IncludeManagementTools
```

3. **Configure AWS Backup**
   - Set up AWS Backup service
   - Create backup plans
   - Assign resources to backup plans
4. **Install Third-Party Backup Software**
   - **Option A: Veeam Backup**
     Download Veeam Backup & Replication; Install and configure; Set up backup jobs to S3.
   - **Option B: Windows Server Backup to S3**

```
# Create backup policy
$Policy = New-WBPolicy
$Target = New-WBBackupTarget -VolumePath "D:"
Add-WBBackupTarget -Policy $Policy -Target $Target
Add-WBVolume -Policy $Policy -Volume (Get-WBVolume -VolumePath "C:")
Set-WBSchedule -Policy $Policy -Schedule 02:00
Set-WBPolicy -Policy $Policy
```

5. **Configure S3 Lifecycle Policies**
   - Transition to S3 Glacier for long-term retention
   - Set expiration policies
6. **Set Up EBS Snapshot Automation**

```
# Using AWS PowerShell
New-EC2Snapshot -VolumeId vol-12345678 -Description "Daily Backup"
```

10.3 Best Practices

- Use AWS Backup for centralized backup management
- Store backups in S3 with versioning enabled
- Implement 3-2-1 backup strategy

- Test backup restoration regularly
- Use S3 Glacier for long-term archival
- Enable cross-region backup replication

11 Load Balancing

11.1 AWS Configuration

Service: Application Load Balancer (ALB) or Network Load Balancer (NLB)
Target Group: Multiple Windows Server instances

11.2 Implementation Steps

1. **Launch Multiple Windows Server Instances**
   - Deploy identical servers in different availability zones
   - Install and configure web application on all instances
2. **Create Target Group**
   - Navigate to EC2 > Target Groups
   - Create target group with health check settings

```
Protocol: HTTP/HTTPS
Port: 80/443
Health Check Path: /health
Health Check Interval: 30 seconds
Healthy Threshold: 2
Unhealthy Threshold: 2
```

3. **Create Application Load Balancer**
   - Choose ALB for HTTP/HTTPS traffic
   - Select availability zones
   - Configure security groups
   - Add listener rules
   - Register target group
4. **Configure Session Persistence**
   - Enable sticky sessions if needed
   - Configure duration
5. **Set Up Auto Scaling Group**

```
# Using AWS CLI or CloudFormation
# Define launch template
# Create auto-scaling group
# Configure scaling policies
```

6. **Install and Configure IIS ARR (Alternative)**

```
# For Windows-based load balancing
Install-WindowsFeature Web-Server -IncludeManagementTools
# Install Application Request Routing
# Configure server farms
```

11.3 Best Practices

- Use ALB for HTTP/HTTPS traffic
- Use NLB for TCP/UDP traffic or ultra-low latency
- Deploy instances across multiple availability zones
- Configure proper health checks
- Enable access logs for troubleshooting
- Use CloudWatch for monitoring
- Implement auto-scaling based on metrics

12 Failover Cluster

12.1 AWS Configuration

Instance Type: r5.xlarge or larger
Storage: Shared storage using FSx for Windows or S3
Network: Placement groups for low latency
Security Group: Allow cluster communication ports

12.2 Implementation Steps

1. **Launch Multiple Windows Server Instances**
   - Deploy in same VPC, different availability zones
   - Use placement group for low latency
2. **Install Failover Clustering Feature**

```
Install-WindowsFeature -Name Failover-Clustering -IncludeManagementTools
```

3. **Configure Shared Storage**
   - **Option A: Amazon FSx for Windows File Server**
     Create FSx file system; Mount on all cluster nodes.
   - **Option B: EBS Multi-Attach (io2 only)**
     Attach same EBS volume to multiple instances; Initialize as cluster shared volume.
4. **Create Failover Cluster**

```
# Validate cluster configuration
Test-Cluster -Node "Node1", "Node2"

# Create cluster
New-Cluster -Name "MyCluster" -Node "Node1", "Node2" -StaticAddress "10.0.1.100" -
    NoStorage
```

5. **Configure Cluster Quorum**

```
Set-ClusterQuorum -NodeAndFileShareMajority "\\FSx\Witness"
```

6. **Add Clustered Role**

```
# For SQL Server
Add-ClusterServerRole -Name "SQL-Cluster" -Storage "Cluster Disk 1"
```

7. **Configure Secondary Private IP**
   - Assign secondary private IP to ENI
   - Configure in cluster as virtual IP

12.3 Common Cluster Types in AWS

SQL Server Failover Cluster Use FSx for shared storage; Configure SQL Server on cluster nodes; Set up availability group for database replication.

File Server Cluster Use FSx or S3 for storage; Configure highly available file shares.

12.4 Best Practices

- Use Amazon FSx for Windows File Server for shared storage
- Deploy cluster nodes in different availability zones
- Use Elastic IP or Network Load Balancer for client access
- Monitor cluster health with CloudWatch
- Regular testing of failover scenarios
- Consider Amazon RDS Multi-AZ for database clustering

13 FTP Server

13.1 AWS Configuration

Instance Type: t3.small to t3.medium
Security Group Ports: 21 (FTP Control), 20 (FTP Data), 990 (FTPS), Range for Passive Mode (e.g., 50000-50100)
Elastic IP: Required for consistent access

13.2 Implementation Steps

1. **Launch Windows Server Instance with Elastic IP**

2. **Install FTP Server Role**

```
Install-WindowsFeature -Name Web-Ftp-Server -IncludeManagementTools
Install-WindowsFeature -Name Web-Ftp-Service
```

3. **Configure FTP Site**

```
# Create FTP site
New-WebFtpSite -Name "FTP Site" -Port 21 -PhysicalPath "D:\FTP"

# Configure authentication
Set-WebConfigurationProperty -Filter /system.ftpServer/security/authentication/
    basicAuthentication -PSPath IIS:\ -Location "FTP Site" -Name enabled -Value
    $true
```

4. **Configure Passive Mode**

```
# Set passive port range
Set-WebConfigurationProperty -Filter /system.ftpServer/firewallSupport -PSPath IIS:\
     -Name lowDataChannelPort -Value 50000
Set-WebConfigurationProperty -Filter /system.ftpServer/firewallSupport -PSPath IIS:\
     -Name highDataChannelPort -Value 50100

# Set external IP
Set-WebConfigurationProperty -Filter /system.ftpServer/firewallSupport -PSPath IIS:\
     -Name externalIp4Address -Value "YOUR_ELASTIC_IP"
```

5. **Enable FTPS (FTP over SSL)**

```
# Import SSL certificate
$cert = New-SelfSignedCertificate -DnsName "ftp.yourdomain.com" -CertStoreLocation
    cert:\LocalMachine\My

# Bind certificate to FTP site
Set-WebConfigurationProperty -Filter /system.ftpServer/security/ssl -PSPath IIS:\ -
    Location "FTP Site" -Name serverCertHash -Value $cert.Thumbprint
Set-WebConfigurationProperty -Filter /system.ftpServer/security/ssl -PSPath IIS:\ -
    Location "FTP Site" -Name ssl128 -Value $true
```

6. **Configure User Access**

```
# Create FTP user
New-LocalUser -Name "ftpuser" -Password (ConvertTo-SecureString "Password123!" -
    AsPlainText -Force)

# Set folder permissions
```

```
$acl = Get-Acl "D:\FTP"
$permission = "ftpuser","FullControl","Allow"
$accessRule = New-Object System.Security.AccessControl.FileSystemAccessRule
    $permission
$acl.SetAccessRule($accessRule)
Set-Acl "D:\FTP" $acl
```

7. **Configure Security Group**
    - Allow port 21 (control)
    - Allow passive port range (50000-50100)
    - Restrict source IPs if possible

13.3 Alternative: AWS Transfer Family

Consider using AWS Transfer Family (SFTP, FTPS, FTP) for fully managed file transfer service with S3 backend.

13.4 Best Practices

- Use FTPS or SFTP instead of plain FTP
- Use AWS Transfer Family for managed solution
- Store files on S3 via AWS Transfer Family
- Limit source IP addresses in security groups
- Use separate EBS volume for FTP data
- Monitor with CloudWatch logs
- Regular security audits

14 Container (Docker)

14.1 AWS Configuration

Instance Type: t3.medium or larger
Operating System: Windows Server 2019/2022 with Containers
Security Group Ports: Custom ports based on containerized applications

14.2 Implementation Steps

1. **Launch Windows Server Instance**
   - Choose Windows Server 2019/2022
   - Select version with container support

2. **Install Docker**

```
# Install Docker provider
Install-Module -Name DockerMsftProvider -Repository PSGallery -Force

# Install Docker
Install-Package -Name docker -ProviderName DockerMsftProvider -Force

# Restart computer
Restart-Computer -Force
```

3. **Verify Docker Installation**

```
docker version
docker info
```

4. **Pull Windows Container Images**

```
# Pull Windows Server Core base image
docker pull mcr.microsoft.com/windows/servercore:ltsc2022

# Pull .NET Framework image
docker pull mcr.microsoft.com/dotnet/framework/aspnet:4.8
```

5. **Create Dockerfile**

```
FROM mcr.microsoft.com/dotnet/framework/aspnet:4.8
WORKDIR /inetpub/wwwroot
COPY ./app .
EXPOSE 80
```

6. **Build and Run Container**

```
# Build image
docker build -t mywebapp:v1 .

# Run container
docker run -d -p 80:80 --name webapp mywebapp:v1

# View running containers
docker ps
```

7. **Push to Amazon ECR**

```
# Authenticate to ECR
aws ecr get-login-password --region us-east-1 | docker login --username AWS --
    password-stdin ACCOUNT_ID.dkr.ecr.us-east-1.amazonaws.com

# Tag image
docker tag mywebapp:v1 ACCOUNT_ID.dkr.ecr.us-east-1.amazonaws.com/mywebapp:v1

# Push image
docker push ACCOUNT_ID.dkr.ecr.us-east-1.amazonaws.com/mywebapp:v1
```

14.3 Alternative: Amazon ECS for Windows Containers

Use Amazon Elastic Container Service (ECS) or Amazon Elastic Kubernetes Service (EKS) with Windows support for orchestrated container deployments.

14.3.1 ECS Windows Container Setup

1. **Create ECS Cluster**
   - Choose EC2 launch type with Windows AMI
   - Or use Fargate for Windows (when available)
2. **Create Task Definition**

```
{
  "family": "windows-webapp",
  "containerDefinitions": [
    {
      "name": "webapp",
      "image": "ACCOUNT_ID.dkr.ecr.us-east-1.amazonaws.com/mywebapp:v1",
      "memory": 2048,
      "cpu": 1024,
      "portMappings": [
        {
          "containerPort": 80,
          "protocol": "tcp"
        }
      ]
    }
  ],
  "requiresCompatibilities": ["EC2"],
  "networkMode": "awsvpc",
  "runtimePlatform": {
    "operatingSystemFamily": "WINDOWS_SERVER_2022_CORE"
  }
}
```

3. **Create ECS Service**
   - Deploy task definition
   - Configure load balancer
   - Set desired task count

14.4 Best Practices

- Use Amazon ECS or EKS for production container orchestration
- Store images in Amazon ECR
- Use Windows Server Core or Nano Server base images
- Implement CI/CD with AWS CodePipeline
- Monitor containers with CloudWatch Container Insights
- Use Task roles for AWS service access
- Regular image security scanning

15 Domain Controller

15.1 AWS Configuration

Instance Type: t3.medium or larger
Security Group Ports:

- 53 (DNS TCP/UDP)
- 88 (Kerberos)
- 135 (RPC)
- 139, 445 (SMB)
- 389, 636 (LDAP, LDAPS)
- 3268, 3269 (Global Catalog)
- 49152-65535 (Dynamic RPC)

Storage: Minimum 50GB SSD
Operating System: Windows Server 2019/2022

15.2 Installation Steps

1. **Prepare the Server**
    - Set a static IP address in Windows network settings
    - Configure DNS to point to itself (127.0.0.1) and a secondary DNS
    - Rename the server with a descriptive hostname
    - Ensure the system is fully updated
2. **Install Active Directory Domain Services**
    - Open Server Manager
    - Click “Add roles and features”
    - Select “Active Directory Domain Services” role
    - Include management tools when prompted
    - Complete the installation wizard
3. **Promote to Domain Controller**
    - Click the notification flag in Server Manager
    - Select “Promote this server to a domain controller”
    - Choose “Add a new forest” for a new domain or “Add a domain controller to an existing domain”
    - Specify the root domain name (e.g., company.local)
    - Set the Forest and Domain functional levels (Windows Server 2016 or higher recommended)
    - Configure DNS and Global Catalog options (typically enabled by default)
    - Set Directory Services Restore Mode (DSRM) password

- Review NetBIOS domain name
- Specify paths for AD database, log files, and SYSVOL
- Review settings and promote
- Server will restart automatically

4. **Post-Installation Configuration**
   - Verify DNS is functioning correctly
   - Create Organizational Units (OUs) for logical organization
   - Configure Group Policy Objects (GPOs) as needed
   - Set up additional domain controllers for redundancy
   - Configure Active Directory Sites and Services if multi-site
   - Implement backup strategy for system state and Active Directory
   - Configure time synchronization (PDC Emulator should sync with external source)
   - Enable and configure Active Directory Recycle Bin for easier object recovery

15.3 Installation Steps - PowerShell

1. **Set Static IP Address**

```
# View current network adapters
Get-NetAdapter

# Set static IP (adjust values for your environment)
New-NetIPAddress -InterfaceAlias "Ethernet" -IPAddress 10.0.1.10 -PrefixLength 24 -
    DefaultGateway 10.0.1.1

# Set DNS to localhost (127.0.0.1) and secondary
Set-DnsClientServerAddress -InterfaceAlias "Ethernet" -ServerAddresses
    127.0.0.1,8.8.8.8
```

2. **Rename Computer**

```
# Rename the server
Rename-Computer -NewName "DC01" -Restart
```

3. **Install AD DS Role**

```
# Install Active Directory Domain Services role with management tools
Install-WindowsFeature -Name AD-Domain-Services -IncludeManagementTools

# Verify installation
Get-WindowsFeature | Where-Object {$_.Name -eq "AD-Domain-Services"}
```

4. **Promote to Domain Controller (New Forest)**

```
# Import the AD DS Deployment module
Import-Module ADDSDeployment

# Create new forest and promote to DC
Install-ADDSForest `
    -DomainName "company.local" `
    -DomainNetbiosName "COMPANY" `
    -ForestMode "WinThreshold" `
    -DomainMode "WinThreshold" `
    -InstallDns:$true `
```

```
    -CreateDnsDelegation:$false `
    -DatabasePath "C:\Windows\NTDS" `
    -LogPath "C:\Windows\NTDS" `
    -SysvolPath "C:\Windows\SYSVOL" `
    -NoRebootOnCompletion:$false `
    -Force:$true
```

Note: You'll be prompted for the SafeModeAdministratorPassword (DSRM password)

5. **Promote Additional Domain Controller (Existing Domain)**

```
# Add DC to existing domain
Install-ADDSDomainController `
    -DomainName "company.local" `
    -InstallDns:$true `
    -Credential (Get-Credential "COMPANY\Administrator") `
    -DatabasePath "C:\Windows\NTDS" `
    -LogPath "C:\Windows\NTDS" `
    -SysvolPath "C:\Windows\SYSVOL" `
    -NoRebootOnCompletion:$false `
    -Force:$true
```

6. **Post-Installation Verification**

```
# Verify AD Web Services is running
Get-Service ADWS

# Check domain controller functionality
Get-ADDomainController

# Test AD replication (if multiple DCs)
repadmin /replsummary

# Verify DNS zones
Get-DnsServerZone

# Check SYSVOL replication
dfsrdiag replicationstate /all

# Verify FSMO roles
Get-ADDomain | Select-Object InfrastructureMaster, RIDMaster, PDCEmulator
Get-ADForest | Select-Object DomainNamingMaster, SchemaMaster
```

7. **Create Organizational Units**

```
# Create OUs for organization
New-ADOrganizationalUnit -Name "Users" -Path "DC=company,DC=local"
New-ADOrganizationalUnit -Name "Computers" -Path "DC=company,DC=local"
New-ADOrganizationalUnit -Name "Groups" -Path "DC=company,DC=local"
New-ADOrganizationalUnit -Name "Servers" -Path "DC=company,DC=local"
```

8. **Configure Active Directory Recycle Bin**

```
# Enable AD Recycle Bin (cannot be reversed)
Enable-ADOptionalFeature -Identity 'Recycle Bin Feature' `
    -Scope ForestOrConfigurationSet `
    -Target 'company.local' `
    -Confirm:$false
```

9. **Configure Time Synchronization (PDC Emulator)**

```
# Configure external time source on PDC Emulator
w32tm /config /manualpeerlist:"time.windows.com,0x8" /syncfromflags:manual /reliable
    :yes /update

# Restart Windows Time service
Restart-Service W32Time

# Force sync and check status
w32tm /resync
w32tm /query /status
```

10. **Set Password Policy**

```
# Configure default domain password policy
Set-ADDefaultDomainPasswordPolicy -Identity "company.local" `
    -ComplexityEnabled $true `
    -LockoutDuration "00:30:00" `
    -LockoutThreshold 5 `
    -MaxPasswordAge "90.00:00:00" `
    -MinPasswordAge "1.00:00:00" `
    -MinPasswordLength 12 `
    -PasswordHistoryCount 24
```

15.4 Security Best Practices

- Implement least privilege access for domain administrators
- Use separate administrative accounts for daily tasks vs. domain administration
- Enable and monitor security logs
- Regularly patch and update the domain controller
- Consider implementing tiered administrative model
- Use strong password policies and account lockout policies
- Utilize Advanced Threat Protection