Windows Server Deployment Guide on AWS Cloud

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

December 20, 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

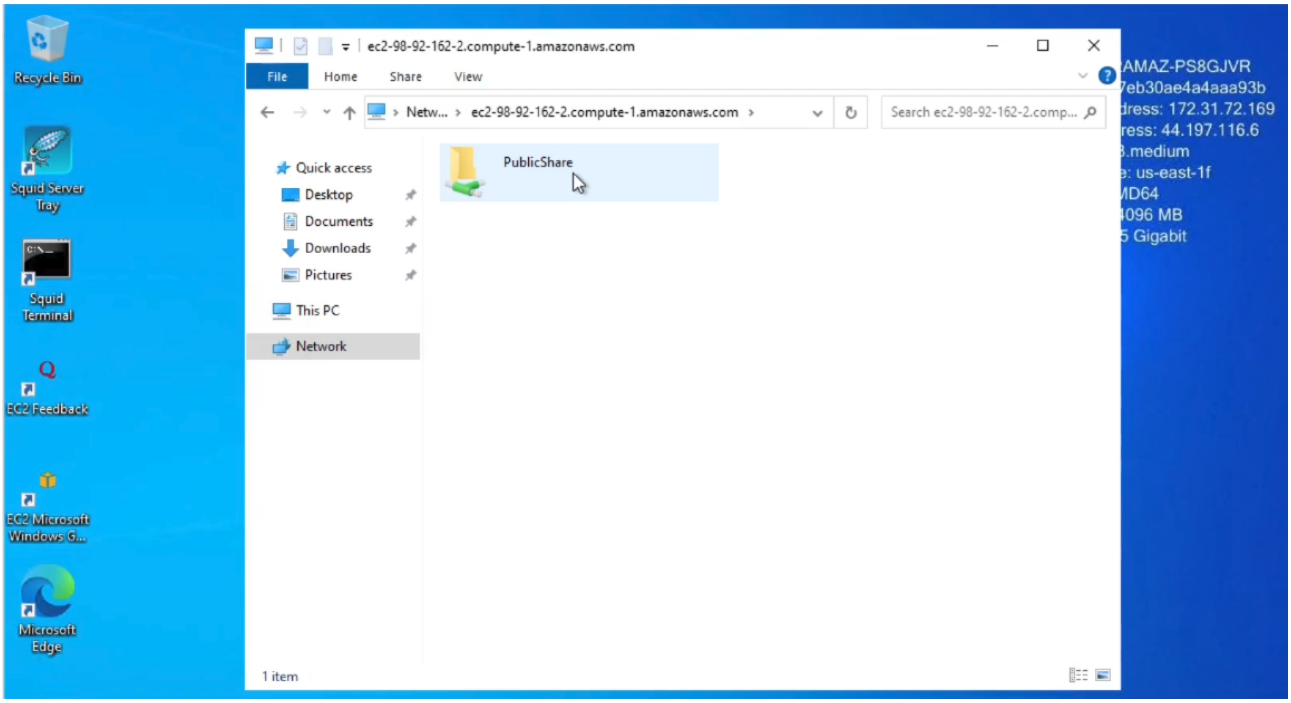

Figure 1: Successfully connected to file server

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

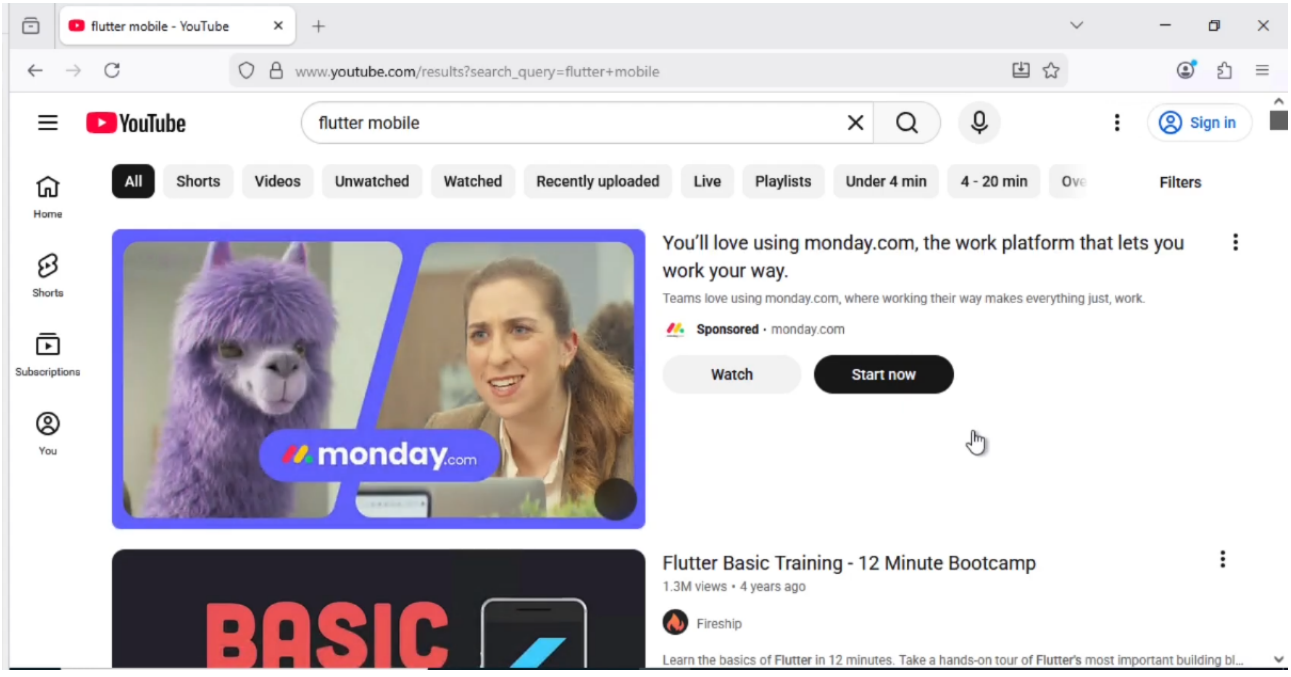

Figure 2: Allow access to youtube

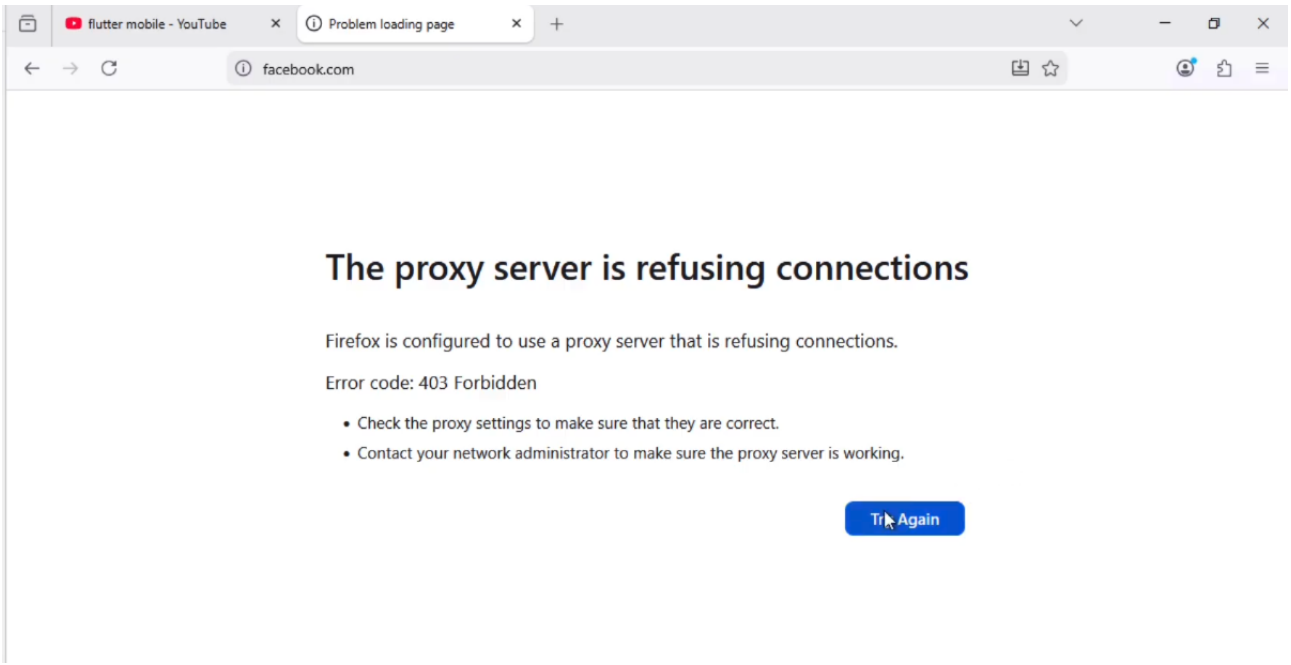

Figure 3: Block access to facebook

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>‘

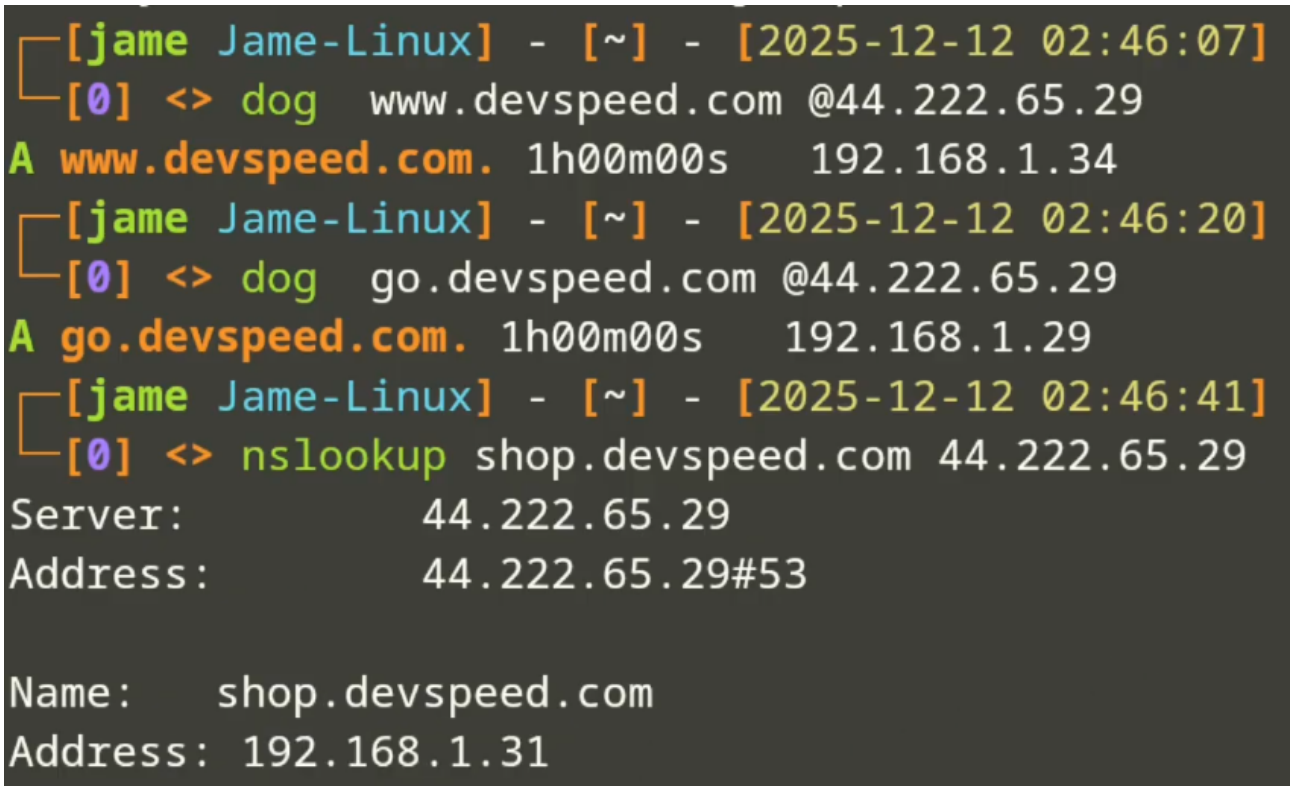

Figure 4: Dns server response with respective IP address

4 DHCP Server

4.1 AWS Configuration

Instance Name: DHCP Server
Instance Type: t3.medium
Security Group Ports: udp 67, udp 68, tcp 3389 (RDP)

> **Note**
>
> AWS VPC provides DHCP by default; custom DHCP server is optional.
> Is the dhcp server in ec2 instance works in AWS, becuase ec2 instance already used VPC DHCP?
> No — a normal DHCP server inside an EC2 instance does NOT work for assigning IPs to other EC2 instances.
> Because:
> 👉 AWS VPC already provides DHCP
> 👉 DHCP broadcast is blocked in AWS

4.2 Implementation

```
# Install DHCP Role
Install-WindowsFeature DHCP -IncludeManagementTools

# If in a domain (skip if standalone)
# Add-DhcpServerInDC -DnsName "dhcpserver.example.internal" -IpAddress "10.0.1.10"

# Create scope
Add-DhcpServerv4Scope `
    -Name "MainScope" `
    -StartRange 10.0.1.50 `
    -EndRange 10.0.1.150 `
    -SubnetMask 255.255.255.0 `
    -State Active

# Configure DHCP options
Set-DhcpServerv4OptionValue -ScopeId 10.0.1.0 -Router 10.0.1.1
Set-DhcpServerv4OptionValue -ScopeId 10.0.1.0 -DnsServer 10.0.1.10
Set-DhcpServerv4OptionValue -ScopeId 10.0.1.0 -DnsDomain "example.internal"

New-NetFirewallRule -DisplayName "Allow DHCP UDP 67" -Direction Inbound -Protocol UDP -
    LocalPort 67 -Action Allow
New-NetFirewallRule -DisplayName "Allow DHCP UDP 68" -Direction Inbound -Protocol UDP -
    LocalPort 68 -Action Allow
```

4.3 Usage

- AWS VPC DHCP is used in EC2 instance

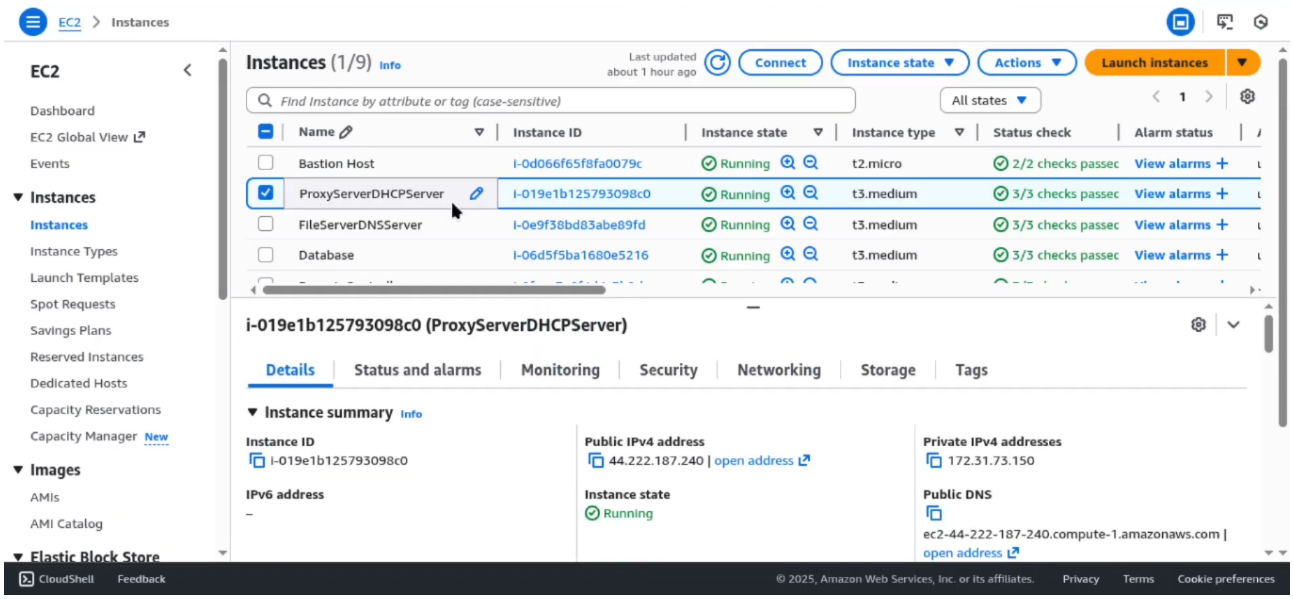

Figure 5: A dhcp server instance

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 . . . . . . . . . :
```

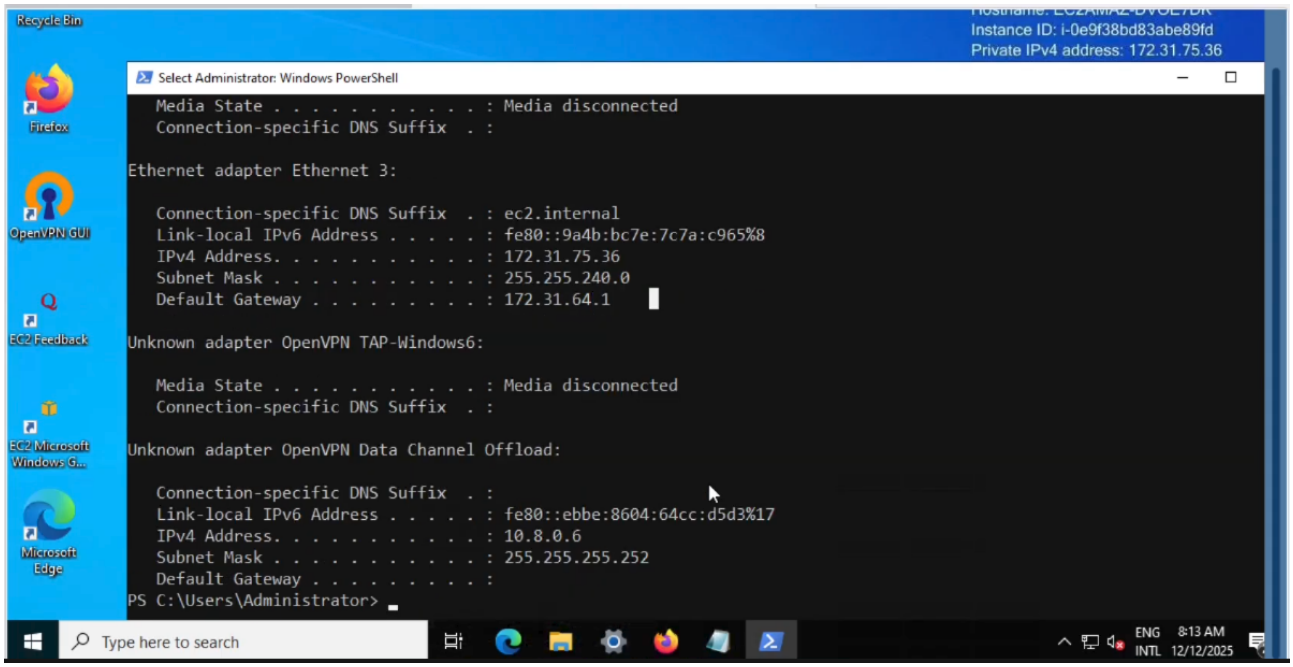

Figure 6: vpn client successfully connect to VPN server

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

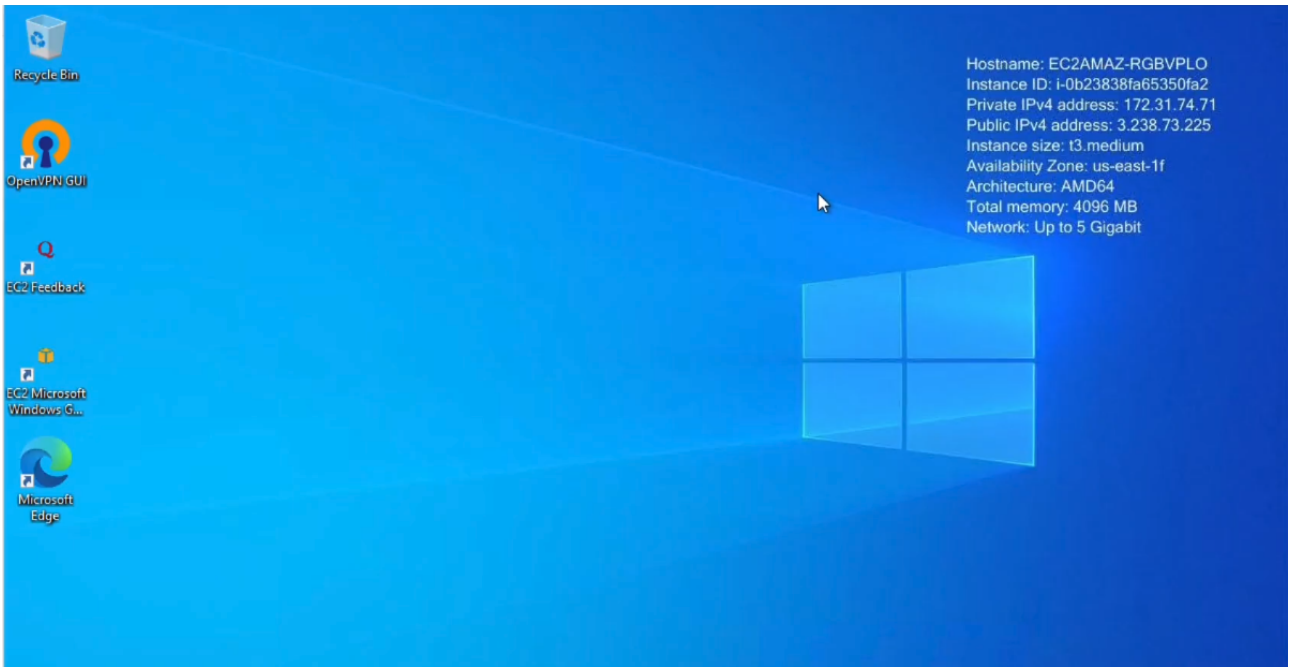

Figure 7: Successfully connect to terminal server

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

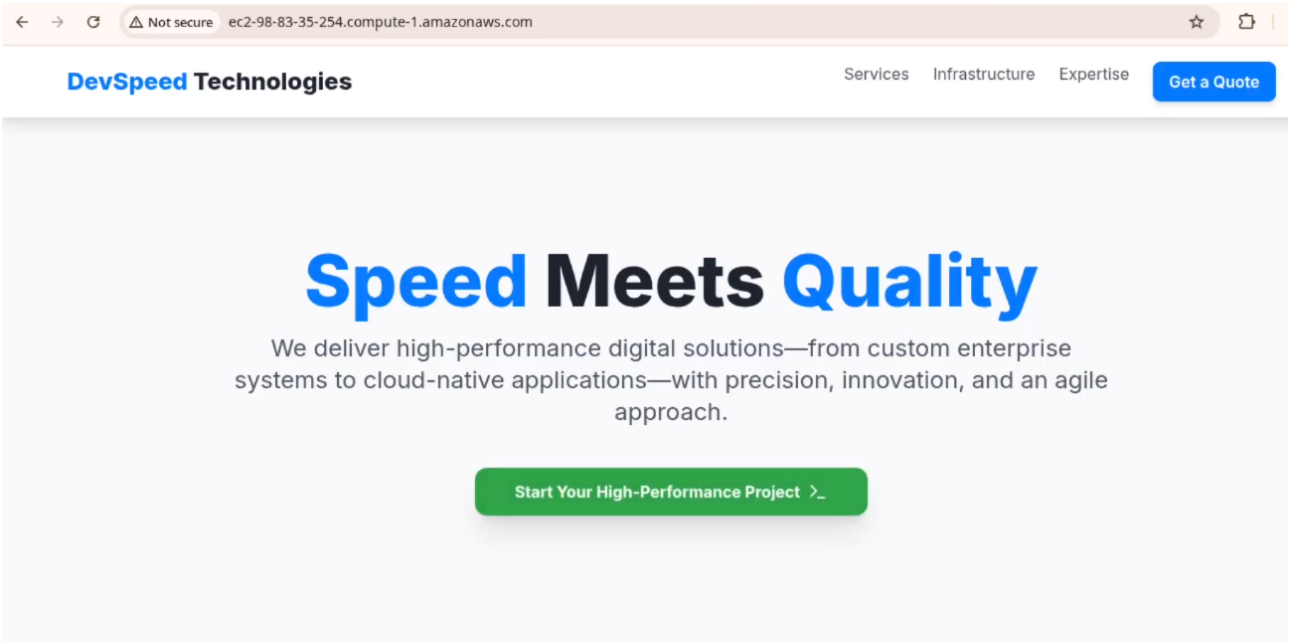

Figure 8: visiting the web page

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

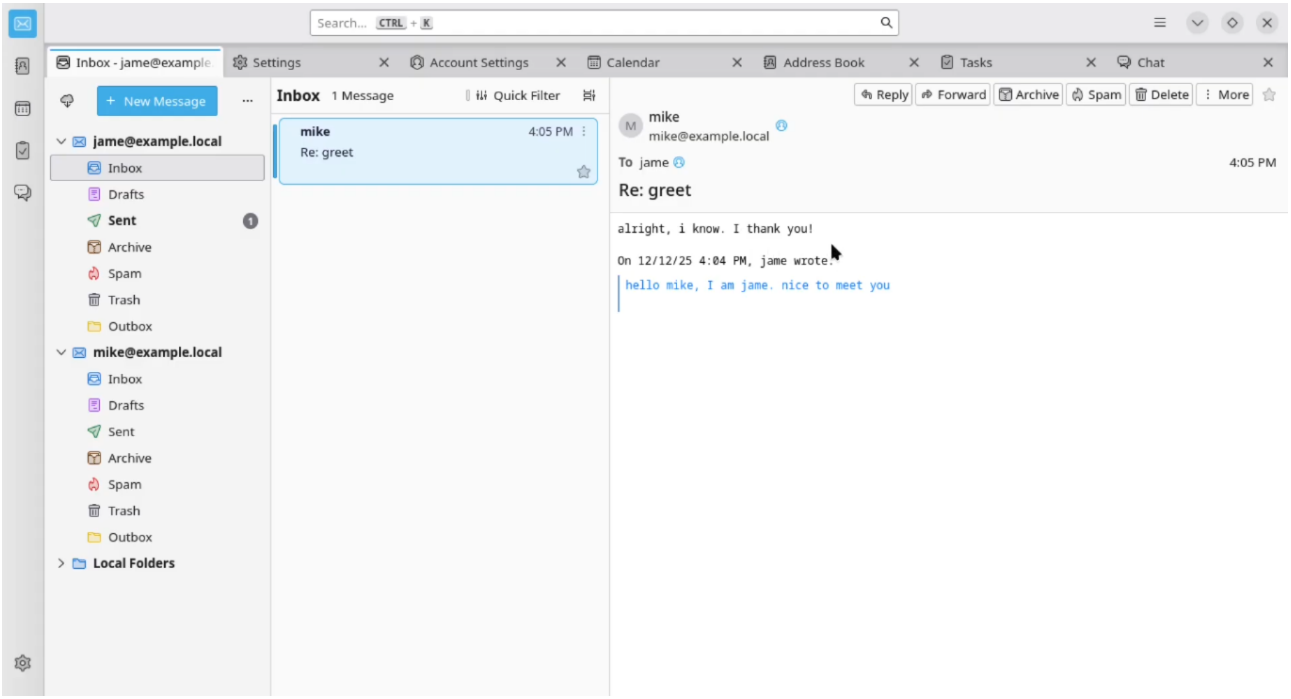

Figure 9: Two users sending messages to each other

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 10: 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 11: client using sql server

Figure 12: client using mongodb server

10 Backup Server

10.1 AWS Configuration

Instance Type: t3.medium
Security Group Ports: tcp 3389 (RDP)
Storage: Large EBS volumes or S3 integration
IAM Role: Permissions for S3, EBS snapshots

10.2 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 awscli -y

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

$bucket = (New-Guid).ToString()
echo $bucket > "C:\bucket.txt"

aws s3 mb "s3://$bucket"

Invoke-WebRequest https://raw.githubusercontent.com/james5635/windows_server_assignment/
    refs/heads/main/scripts/S3Backup.ps1 -OutFile "C:\S3Backup.ps1"
Invoke-WebRequest https://raw.githubusercontent.com/james5635/windows_server_assignment/
    refs/heads/main/scripts/S3BackupSchedule.ps1 -OutFile "C:\S3BackupSchedule.ps1"
Invoke-WebRequest https://raw.githubusercontent.com/james5635/windows_server_assignment/
    refs/heads/main/scripts/S3BackupScheduleNightly.ps1 -OutFile "C:\
    S3BackupScheduleNightly.ps1"

& "C:\S3BackupSchedule.ps1"
& "C:\S3BackupScheduleNightly.ps1"
```

10.3 Usage

- automatically backup "C:\inetpub" to s3 bucket every 1 minute and every night at 2 a.m.
- check out the s3 bucket

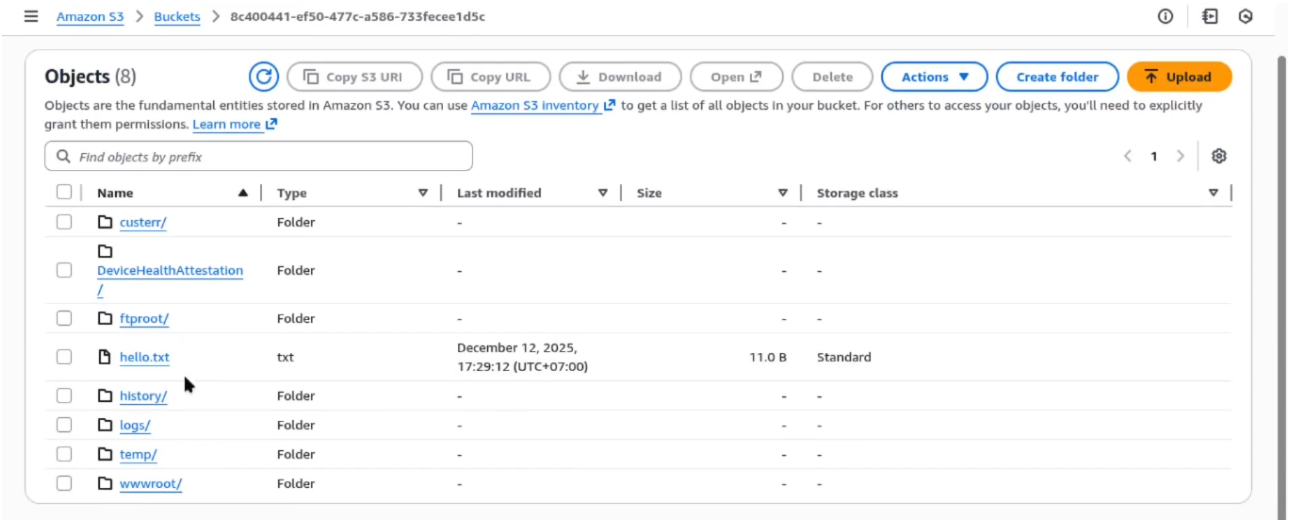

Figure 13: automatically backup to s3 bucket

11 Load Balancing

11.1 AWS Configuration

Service: Application Load Balancer (ALB)
Target Group: Multiple Windows Server instances
Instance Type: t3.medium
Security Group Ports: tcp 3389 (RDP)

11.2 Implementation Steps

11.2.1 Server 1

```
Start-Transcript -Path "C:\WebServer.log" -Append
Install-WindowsFeature -Name Web-Server
# echo "Server 1" > C:\inetpub\wwwroot\index.html
```

11.2.2 Server 2

```
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"
```

11.3 Usage

- manually configure load balancer (application load balancer)
- name: devspeedlb
- select all availability Zones and subnets
- create target group (name: web)
- - include WebServerMailServer & Docker
- choose 'web' as target group
- add tcp (port 80) to inbound rule of the load balancer's security group

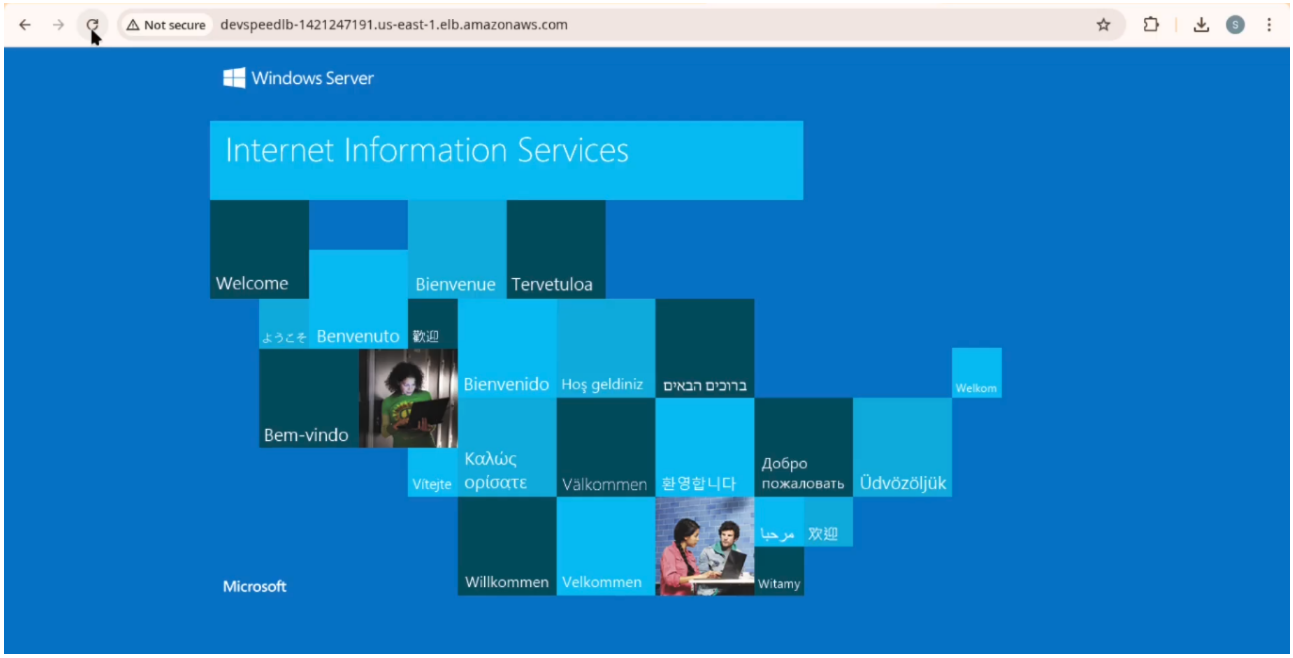

Figure 14: load balance to server 1

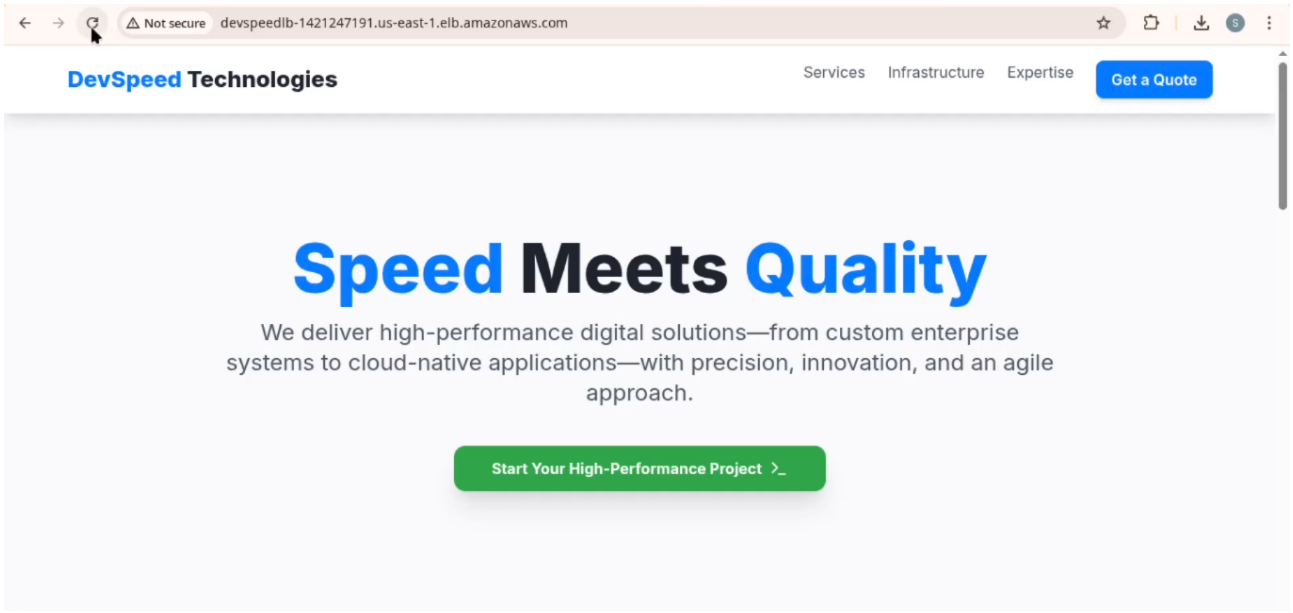

Figure 15: load balance to server 2

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 Name: FTP Server
Instance Type: t3.small to t3.medium
Security Group Ports: tcp 21, tcp 50000-51000, tcp 3389 (RDP)

13.2 Implementation

```
#Install IIS Feature
Install-WindowsFeature -Name Web-Server -IncludeManagementTools

#Install FTP feature
Install-WindowsFeature -Name Web-Ftp-Server -IncludeAllSubFeature -IncludeManagementTools
     -Verbose

#Creating new FTP site
$SiteName = "Demo FTP Site"
$RootFolderpath = "C:\inetpub\ftproot"
$PortNumber = 21
$FTPUserGroupName = "Demo FTP Users Group"
$FTPUserName = "jame"
$FTPPassword = ConvertTo-SecureString "Mypassword@2025" -AsPlainText -Force

if (!(Test-Path $RootFolderpath)) {
    # if the folder doesn't exist
    New-Item -Path $RootFolderpath -ItemType Directory # create the folder
}

New-WebFtpSite -Name $SiteName -PhysicalPath $RootFolderpath -Port $PortNumber -Verbose -
    Force

#Creating the local Windows group
if (!(Get-LocalGroup $FTPUserGroupName  -ErrorAction SilentlyContinue)) {
    #if the group doesn't exist
    New-LocalGroup -Name $FTPUserGroupName `
        -Description "Members of this group can connect to FTP server" #create the group
}

# Creating an FTP user
If (!(Get-LocalUser $FTPUserName -ErrorAction SilentlyContinue)) {
    New-LocalUser -Name $FTPUserName -Password $FTPPassword `
        -Description "User account to access FTP server" `
        -UserMayNotChangePassword
}

# Add the created FTP user to the group Demo FTP Users Group
Add-LocalGroupMember -Name $FTPUserGroupName -Member $FTPUserName -ErrorAction
    SilentlyContinue

# Enabling basic authentication on the FTP site
$param = @{
    Path    = 'IIS:\Sites\Demo FTP Site'
    Name    = 'ftpserver.security.authentication.basicauthentication.enabled'
    Value   = $true
    Verbose = $True
}
Set-ItemProperty @param

```

```
# Adding authorization rule to allow FTP users
# in the FTP group to access the FTP site
$param = @{
    PSPath   = 'IIS:\'
    Location = $SiteName
    Filter   = '/system.ftpserver/security/authorization'
#    Value    = @{ accesstype = 'Allow'; roles = $FTPUserGroupName; permissions = 1 }
    Value    = @{ accesstype = 'Allow'; roles = $FTPUserGroupName; permissions = 3 }
}

Add-WebConfiguration @param

# Changing SSL policy of the FTP site
'ftpServer.security.ssl.controlChannelPolicy', 'ftpServer.security.ssl.dataChannelPolicy'
    |
ForEach-Object {
    Set-ItemProperty -Path "IIS:\Sites\Demo FTP Site" -Name $_ -Value $false
}

# can be 'ReadAndExecute', 'Modify'
$ACLObject = Get-Acl -Path $RootFolderpath
$ACLObject.SetAccessRule(
    ( # Access rule object
        New-Object System.Security.AccessControl.FileSystemAccessRule(
            $FTPUserGroupName,
            'FullControl',
            'ContainerInherit,ObjectInherit',
            'None',
            'Allow'
        )
    )
)
Set-Acl -Path $RootFolderpath -AclObject $ACLObject

# Checking the NTFS permissions on the FTP root folder
Get-Acl -Path $RootFolderpath | ForEach-Object Access

# Test FTP Port and FTP access
Test-NetConnection -ComputerName localhost -Port 21


Set-WebConfigurationProperty -pspath 'MACHINE/WEBROOT/APPHOST'  -filter "system.ftpServer
    /firewallSupport" -name "lowDataChannelPort" -value 50000
Set-WebConfigurationProperty -pspath 'MACHINE/WEBROOT/APPHOST'  -filter "system.ftpServer
    /firewallSupport" -name "highDataChannelPort" -value 51000

Set-WebConfigurationProperty -pspath 'MACHINE/WEBROOT/APPHOST' -filter "system.
    applicationHost/sites/site[@name='Demo FTP Site']/ftpServer/firewallSupport" -name "
    externalIp4Address" -value "${MyEIP}"

# Control port
New-NetFirewallRule -DisplayName "Allow FTP 21" -Direction Inbound -Protocol TCP -
    LocalPort 21 -Action Allow

# Passive ports
New-NetFirewallRule -DisplayName "Allow FTP Passive Ports" -Direction Inbound -Protocol
    TCP -LocalPort 50000-51000 -Action Allow

Restart-Service FTPSVC

# C:\Windows\System32\config\systemprofile\AppData\Local\Temp\EC2Launch380802110\
    UserScript.ps1
# apple
```

```
103
104 echo hello > "C:\inetpub\ftproot\hello.txt"
```

13.3 Usage

- use filezilla (port 21)
- username: jame
- password: Mypassword@2025

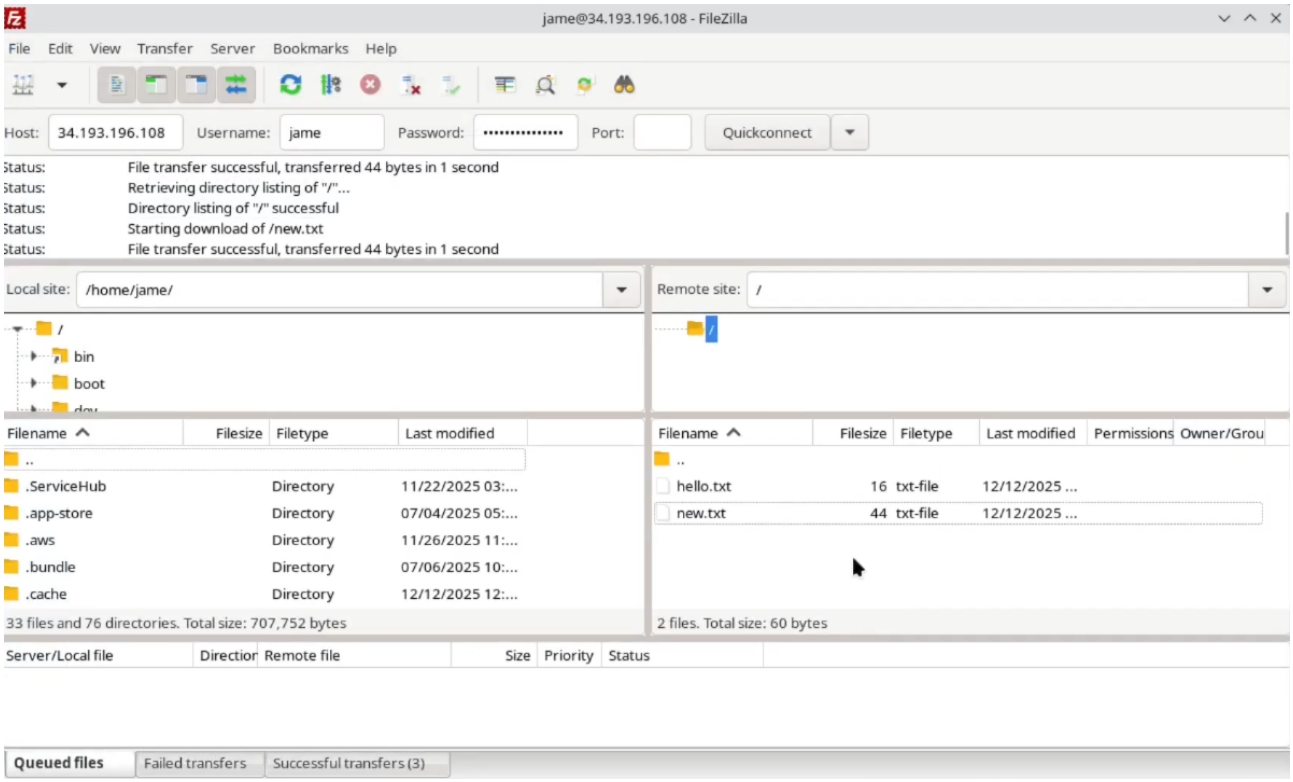

Figure 16: FileZilla successfully connected to ftp server

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

```
Invoke-WebRequest -UseBasicParsing "https://raw.githubusercontent.com/microsoft/Windows-Containers/Main/helpful_tools/Install-DockerCE/install-docker-ce.ps1" -o install-docker-ce.ps1
.\install-docker-ce.ps1
# will restart the machine
```

14.3 Usage

- connect to the Docker ec2 instance
- docker pull mcr.microsoft.com/windows/nanoserver:ltsc2022
- docker pull mcr.microsoft.com/windows/servercore:ltsc2022 (optional)
- docker run -it <image>

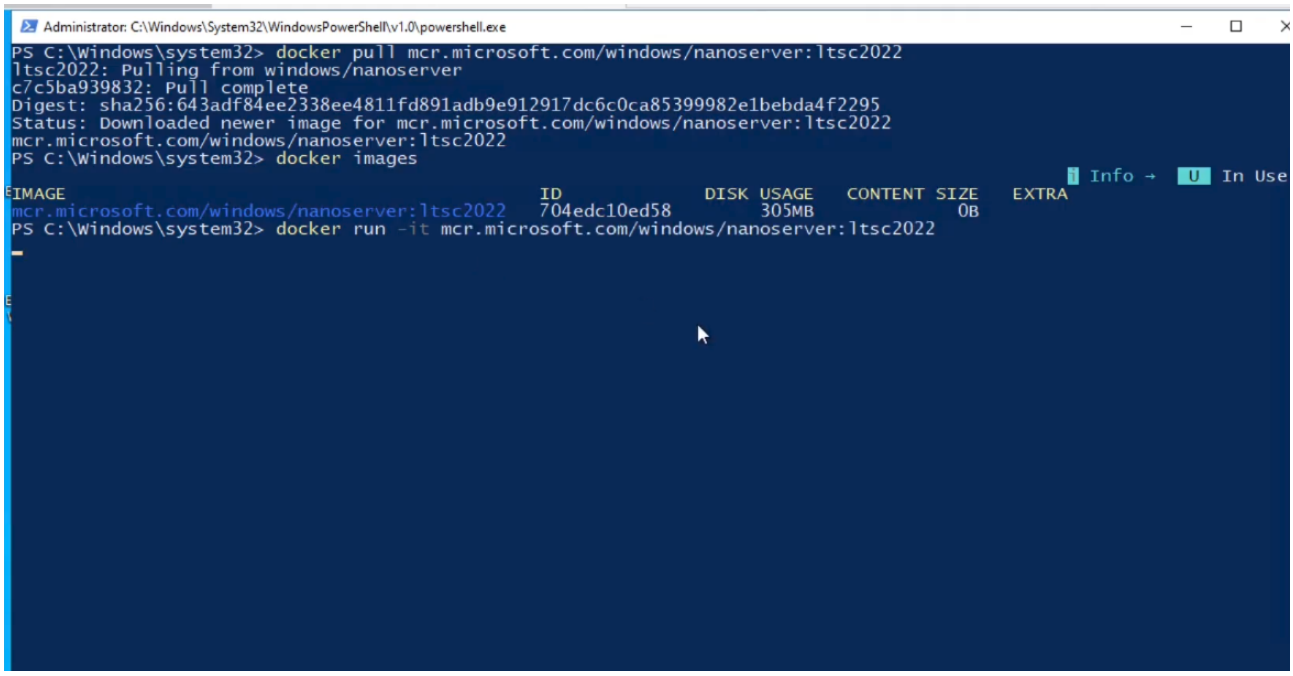

Figure 17: Running windows server container

15 Domain Controller

15.1 AWS Configuration

Instance Name: Domain Controller
Instance Type: t3.medium or larger
Security Group Ports:

- tcp 80
- tcp 3389 (RDP)
- tcp 0-65535 (All tcp ports)
- udp 0-65535 (all udp ports)
- icmp -1 (all icmp)

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

15.2 Implementation

```
Install-WindowsFeature -Name AD-Domain-Services -IncludeManagementTools
Install-ADDSForest -DomainName "example.local" -InstallDNS -SafeModeAdministratorPassword
    (ConvertTo-SecureString "Mypassword@2025" -AsPlainText -Force) -NoRebootOnCompletion
    -Force
# Get-Service adws,kdc,netlogon,dns
Restart-Computer
```

15.3 Usage

- connect to File Server or other windows server
- change dns server address to <private ip of domain controller>
- join the domain 'example.local'

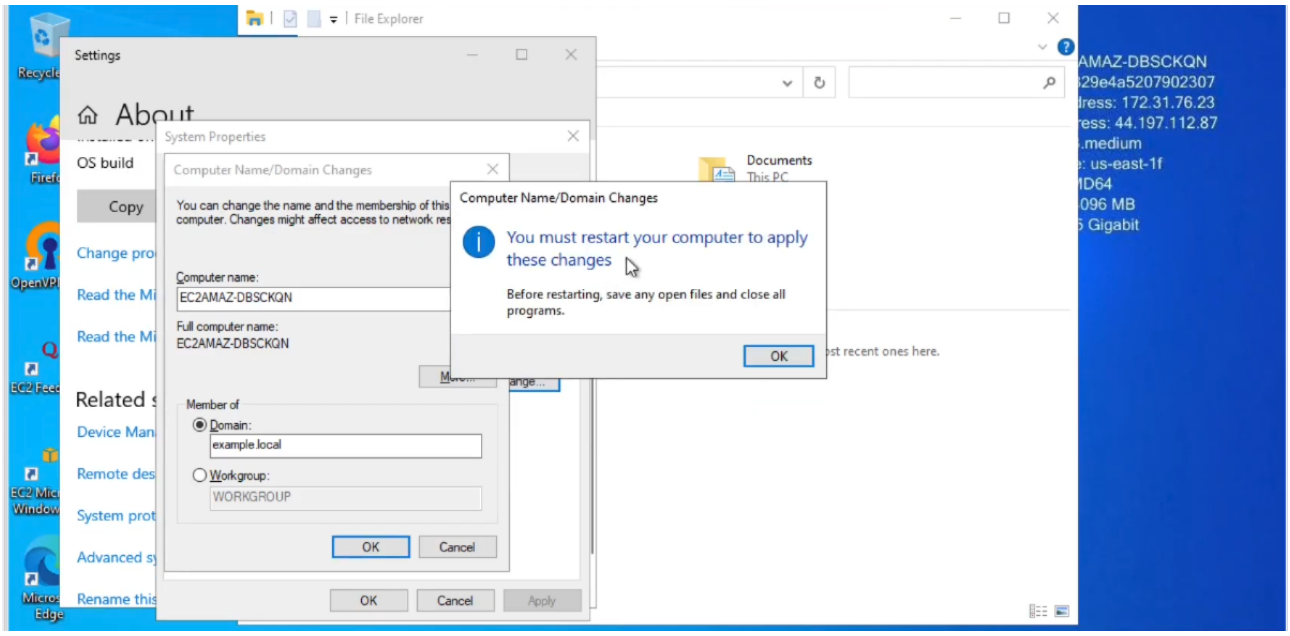

Figure 18: Successfully join the domain

16 DevSpeedLLM

16.1 Configuration

GPU: dual NVIDIA T4 GPUs
OS: Linux

16.2 Implementation

```
from datasets import load_dataset

# train_dataset = load_dataset("trl-lib/Capybara", split="train")
def make_chatml(example):
    return {
        "messages": [ [ {"role": "user", "content": "Summarize the following text:\n\n" + content },
                            {"role": "assistant", "content":  summary }  ]  +
                            [x for qa in qas for x in (
                            {"role": "user", "content": qa["question"] if qa['question'] != None else ""}, # There are 2 question=None in the dataset
                            {"role": "assistant", "content": qa["answer"]},
                            )] for content, summary, qas in zip(example['content'], example['summary'], example["QAs"])
        ]
    }
dataset = load_dataset("james56352025/devspeedllm-datasets", "rupp_enhanced", split = "train")
train_dataset = dataset.map(make_chatml, batched=True, remove_columns=['summary', 'qa_pairs', 'content', 'QAs'])


from transformers import AutoModelForCausalLM, BitsAndBytesConfig
import torch

# model_id = "Qwen/Qwen2.5-1.5B-Instruct"
# model_id = "mistralai/Mistral-7B-Instruct-v0.1"
# model_id = "google/gemma-3-270m-it"
model_id = "meta-llama/Llama-3.2-3B-Instruct"
output_dir = "lora-out"

four_bit = BitsAndBytesConfig(
        load_in_4bit=True,                      # Load the model in 4-bit precision to save memory
        bnb_4bit_compute_dtype=torch.float16,   # Data type used for internal computations in quantization
        bnb_4bit_use_double_quant=True,         # Use double quantization to improve accuracy
        bnb_4bit_quant_type="nf4"               # Type of quantization. "nf4" is recommended for recent LLMs
)

model = AutoModelForCausalLM.from_pretrained(
    model_id,
    attn_implementation="sdpa",                 # Change to Flash Attention if GPU has support
    dtype=torch.float16,                        # Change to bfloat16 if GPU has support
    use_cache=True,                             # Whether to cache attention outputs to speed up inference
    # quantization_config=four_bit,
)

```

```python
from peft import LoraConfig

# You may need to update `target_modules` depending on the architecture of your chosen model.
# For example, different LLMs might have different attention/projection layer names.
peft_config = LoraConfig(
    r=32,
    lora_alpha=32,
    target_modules = ["q_proj", "k_proj", "v_proj", "o_proj", "gate_proj", "up_proj", "down_proj",],
)

import shutil, os, glob
from transformers.trainer_callback import TrainerCallback

class ZipCallback(TrainerCallback):
    # Required empty methods
    def on_init_end(self, args, state, control, **kwargs):
        pass
    def on_train_begin(self, args, state, control, **kwargs):
        pass
    def on_train_end(self, args, state, control, **kwargs):
        pass
    def on_epoch_end(self, args, state, control, **kwargs):
        pass
    # THE ONE WE USE
    def on_save(self, args, state, control, **kwargs):
        # Find latest checkpoint
        # ckpts = sorted(glob.glob(os.path.join(args.output_dir, "checkpoint-*")))
        ckpts = glob.glob(os.path.join(args.output_dir, "checkpoint-*"))
        if not ckpts:
            return control
        # latest = ckpts[-1]
        latest = max(ckpts, key=lambda x: int(x.split("-")[-1]))
        zip_path = os.path.join(os.path.dirname(args.output_dir), os.path.basename(latest) + ".zip")
        if os.path.exists(zip_path):
            print("\n" + f"Already zipped: {zip_path}")
            return control
        shutil.make_archive(os.path.splitext(zip_path)[0], "zip", latest)
        print()
        print(f"Zipped: {zip_path}")
        return control


from trl import SFTConfig, SFTTrainer

training_args = SFTConfig(
    # Training schedule / optimization
    per_device_train_batch_size = 1,      # Batch size per GPU
    gradient_accumulation_steps = 4,      # Gradients are accumulated over multiple steps → effective batch size = 2 * 8 = 16
    warmup_steps = 5,
    num_train_epochs = 5,                 # Number of full dataset passes. For shorter training, use `max_steps` instead (this case)
    # max_steps = 2,
    learning_rate = 2e-4,                 # Learning rate for the optimizer
    optim = "paged_adamw_8bit",           # Optimizer
    # Logging / reporting
    logging_steps=1,                      # Log training metrics every N steps
    report_to="trackio",                  # Experiment tracking tool
    trackio_space_id=output_dir,          # HF Space where the experiment tracking will be saved
```

```
    output_dir=output_dir,                  # Where to save model checkpoints and logs
    # max_length=1024,                        # Maximum input sequence length
    use_liger_kernel=True,                  # Enable Liger kernel optimizations for faster training
    activation_offloading=True,             # Offload activations to CPU to reduce GPU memory usage
    gradient_checkpointing=True,            # Save memory by re-computing activations during backpropagation
    # Hub integration
    push_to_hub=False,                      # Automatically push the trained model to the Hugging Face Hub
                                            # The model will be saved under your Hub account in the repository named `output_dir`
    gradient_checkpointing_kwargs={"use_reentrant": False}, # To prevent warning message
)

trainer = SFTTrainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    peft_config=peft_config,
    callbacks = [ZipCallback()],
)

trainer_stats = trainer.train()
trainer.save_model(output_dir)
```

16.3 Usage

- ask ”tell me about RUPP”
- ask ”list academic staffs of computer science department at RUPP with name”
- ask ”teach me flutter and give example”

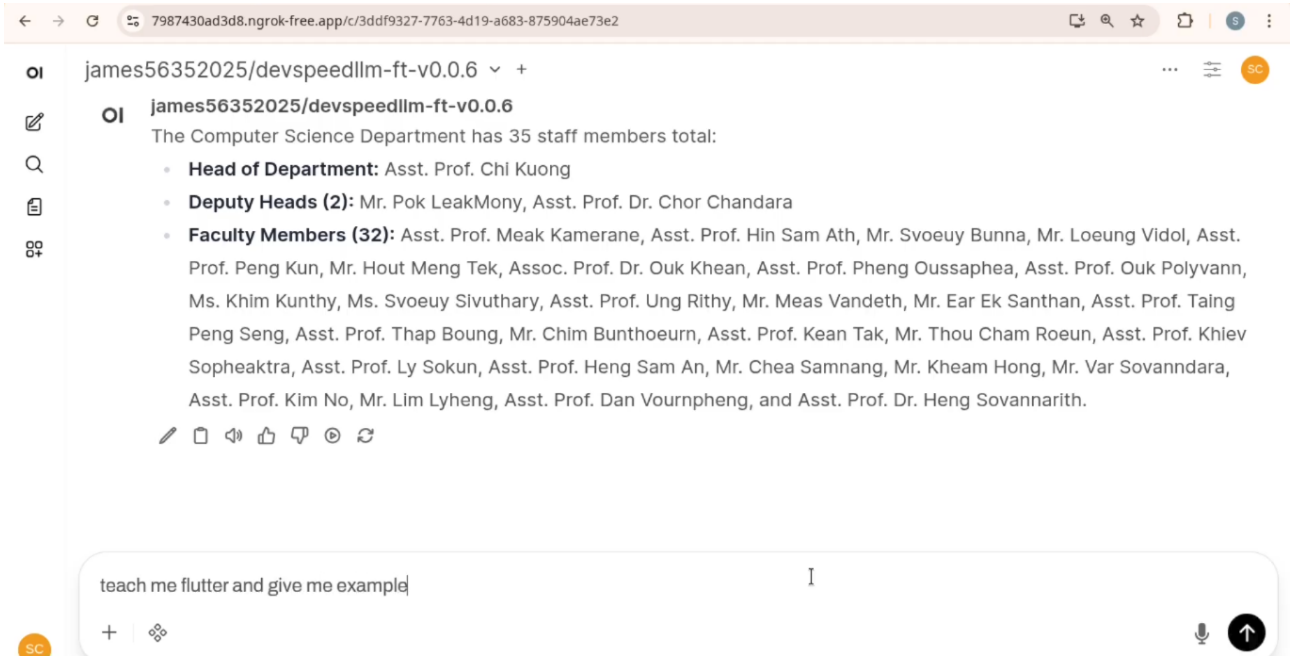

Figure 19: Chatting with DevSpeedLLM