



AUTOMATED AD USER CREATION USING POWERSHELL AND CSV

Streamlining user onboarding in Windows Server
environments

Summary

This project demonstrates how to automate the creation of multiple Active Directory users using a PowerShell script and a CSV file. The solution simplifies IT onboarding by generating user accounts, placing them in the correct Organizational Units (OUs), and assigning them to relevant Security Groups, all without manual setup.

Tebogo Matseding

Intro

Set up Network

1st Login

Change the Server IP Address

Change the Server IP Address 2

Verify IP Address is changed

Renaming Server Name

Verify Name Has Changed

Install Active Directory Domain Services
(AD DS) role

Promote the server to a Domain
Controller (Get-Command -Module
addsdeployment)

Promote the server to a Domain
Controller (Install-ADDSForestInstallation)

Verify Installations

Import-Module ActiveDirectory

Creating Organizational Unit

Verify that OUs have been created

Creating Security Groups

Verify that SGs have been created

Create .csv file with users

Create script that automates user
creation & assign them OUs & SGs

Execute Script

Verify users have been created

Filtering users

Login into Office-PC01

Rename Office-PC01

Changing IP Address and DNS

Joining Office-PC01 to domain

Office-PC01 joined the domain

Login in with user

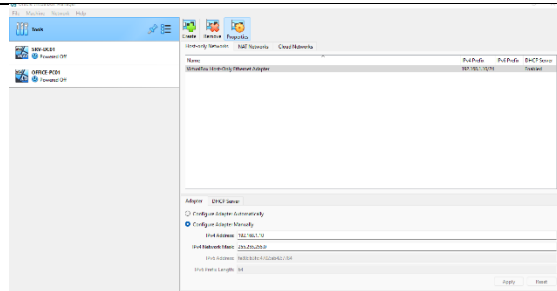
START

Intro

In this project, we set up a Windows Server 2022 environment from scratch and automate the process of creating Active Directory (AD) users using a PowerShell script and a CSV file. This simulates real-world IT workflows, especially useful in organizations onboarding many users at once.

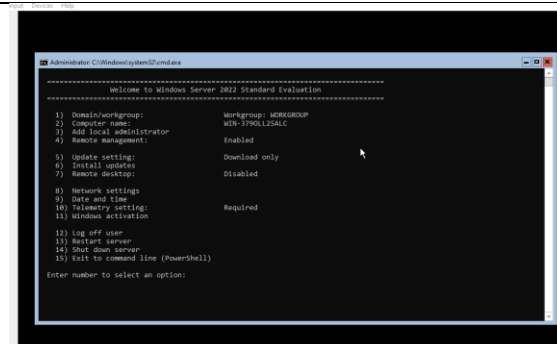
Set up Network

We start by ensuring our server is connected to a reliable and isolated network, either through a virtual switch (in VirtualBox or Hyper-V) or a real network environment. A working network connection is crucial for domain communications.



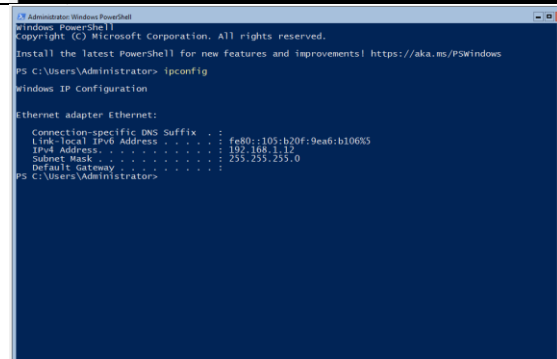
1st Login

After installing Windows Server, we log in for the first time using the local Administrator account to begin configuring the server.



Change the Server IP Address

We assign a static IP address to the server. Static IPs are important for servers so that other devices on the network can consistently find and communicate with them.



Change the Server IP Address 2

Using this exact command **New-NetIPAddress -InterfaceAlias "Ethernet" -IPAddress 192.168.1.30 -PrefixLength 24 -DefaultGateway 192.168.1.10**

```
PS C:\Users\Administrator> Get-NetAdapter

Name                InterfaceDescription      IfIndex Status      MacAddress
-----                -
Ethernet            Intel(R) PRO/1000 MT Desktop Adapter      5 Up          08-00-27-...

PS C:\Users\Administrator> New-NetIPAddress -InterfaceIndex 4 -IPAddress 192.168.1.30 -PrefixLength 24 -DefaultGateway 192.168.1.10
```

Verify IP Address is changed

We use PowerShell ipconfig to make sure our static IP and DNS settings are correctly in place.

```
PS C:\Users\Administrator> ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::105:b20f:9ea6:b106%5
    IPv4 Address. . . . . : 192.168.1.30
    Subnet Mask . . . . . : 240.0.0.0
    Default Gateway . . . . . : 192.168.1.10

PS C:\Users\Administrator>
```

Renaming Server Name

The default name (like WIN-XXXX) is changed to SRV-DC01 to reflect its role as a domain controller.

```
PS C:\Users\Administrator> hostname
WIN-3790LL25ALC
PS C:\Users\Administrator> Rename-Computer -NewName SRV-DC01
WARNING: The changes will take effect after you restart the computer WIN-3790LL25ALC.
PS C:\Users\Administrator>
```

Verify Name Has Changed

After restarting the server, we confirm that the hostname has been updated using hostname

```
PS C:\Users\Administrator> hostname
SRV-DC01
PS C:\Users\Administrator>
```

Install Active Directory Domain Services (AD DS) role

We install the AD DS role using PowerShell. This adds the necessary features for managing users, computers, and security in a domain environment.

```
PS C:\Users\Administrator> Install-WindowsFeature -Name ad-domain-services -IncludeManagementTools

Start-Installation...
24%
[Progress bar showing installation progress]
```

Promote the server to a Domain Controller

(Get-Command -Module addsdeployment)

We explore the addsdeployment module to find commands that can promote our server to a domain controller, such as Install-ADDSForest.

```
PS C:\Users\Administrator> Install-WindowsFeature -Name ad-domain-services -IncludeManagementTools
Success Restart Needed Exit Code      Feature Result
-----
True    No          Success      [Active Directory Domain Services, Group P...

PS C:\Users\Administrator> Get-Command -Module addsdeployment

CommandType      Name                                           Version      Source
-----
Cmdlet            Add-ADDSReadonlyDomainControllerAccount      1.0.0.0      addsdeployment
Cmdlet            Install-ADDSDomain                           1.0.0.0      addsdeployment
Cmdlet            Install-ADDSForest                           1.0.0.0      addsdeployment
Cmdlet            Test-ADSDomainControllerInstallation         1.0.0.0      addsdeployment
Cmdlet            Test-ADSDomainControllerUninstallation       1.0.0.0      addsdeployment
Cmdlet            Test-ADSDomainInstallation                  1.0.0.0      addsdeployment
Cmdlet            Test-ADSDomainControllerAccountCreation     1.0.0.0      addsdeployment
Cmdlet            Uninstall-ADSDomainController                1.0.0.0      addsdeployment

PS C:\Users\Administrator>
```

Promote the server to a Domain Controller (Install-ADDSForestInstallation)

Using PowerShell, we promote our server to become the first Domain Controller in a new forest. This is the backbone of our domain environment.

```
PS C:\Users\Administrator> Get-Command -Module addsdeployment

CommandType      Name                                           Version      Source
-----
Cmdlet            Add-ADDSReadonlyDomainControllerAccount      1.0.0.0      addsdeployment
Cmdlet            Install-ADDSDomain                           1.0.0.0      addsdeployment
Cmdlet            Install-ADDSForest                           1.0.0.0      addsdeployment
Cmdlet            Test-ADSDomainControllerInstallation         1.0.0.0      addsdeployment
Cmdlet            Test-ADSDomainControllerUninstallation       1.0.0.0      addsdeployment
Cmdlet            Test-ADSDomainInstallation                  1.0.0.0      addsdeployment
Cmdlet            Test-ADSDomainControllerAccountCreation     1.0.0.0      addsdeployment
Cmdlet            Uninstall-ADSDomainController                1.0.0.0      addsdeployment

PS C:\Users\Administrator> Install-ADDSForest

cmdlet Install-ADDSForest at command pipeline position 1
Supply values for the following parameters:
DomainName: visionmedia.local
SafeModeAdministratorPassword: *****
Confirm SafeModeAdministratorPassword: *****
Input and confirm input for SafeModeAdministratorPassword do not match. Please try again.
Confirm SafeModeAdministratorPassword: *****
The target server will be configured as a domain controller and restarted when this operation is complete.
Do you want to continue with this operation?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):
```

Verify Installations

We check that the domain controller promotion completed successfully by verifying DNS zones, domain info, and system logs.

```
PS C:\Users\Administrator> Get-WindowsFeature AD-Domain-Services, RSAT-AD-PowerShell

Display Name      Name      Install State
-----
[X] Active Directory Domain Services      AD-Domain-Services      Installed
[X] Active Directory module for Windows ... RSAT-AD-PowerShell      Installed

PS C:\Users\Administrator>
```

Import-Module ActiveDirectory

Before managing users and groups, we load the Active Directory module in PowerShell. This module contains the commands needed for AD management.

```
PS C:\Users\Administrator> Import-Module ActiveDirectory
PS C:\Users\Administrator>
```

Creating Organizational Units

We use PowerShell to create Organizational Units (OUs) like "Sales", "IT", or "HR" to logically organize users and apply group policies later.

```
PS C:\Users\Administrator> New-ADOrganizationalUnit -Name Sales -Path "DC=visionmedia,DC=local"
PS C:\Users\Administrator> New-ADOrganizationalUnit -Name "IT" -Path "DC=visionmedia,DC=local"
PS C:\Users\Administrator> New-ADOrganizationalUnit -Name "HR" -Path "DC=visionmedia,DC=local"
PS C:\Users\Administrator> New-ADOrganizationalUnit -Name "Finance" -Path "DC=visionmedia,DC=local"
PS C:\Users\Administrator>
```

Verify that OUs have been created

We confirm the OUs exist using commands like Get-ADOrganizationalUnit or by listing the directory structure.

```
PS C:\Users\Administrator> Get-ADOrganizationalUnit -Filter * | Select Name, DistinguishedName

Name                DistinguishedName
-----
Domain Controllers  OU=Domain Controllers,DC=visionmedia,DC=local
Sales                OU=Sales,DC=visionmedia,DC=local
IT                  OU=IT,DC=visionmedia,DC=local
HR                  OU=HR,DC=visionmedia,DC=local
Finance             OU=Finance,DC=visionmedia,DC=local
```

Creating Security Groups

We create Security Groups (SGs) within the OUs. These groups help manage permissions and access to network resources.

```
PS C:\Users\Administrator> New-ADGroup -Name SalesGroup -GroupScope Global -GroupCategory Security
PS C:\Users\Administrator> New-ADGroup -Name ITGroup -GroupScope Global -GroupCategory Security
PS C:\Users\Administrator> New-ADGroup -Name HRGroup -GroupScope Global -GroupCategory Security
PS C:\Users\Administrator> New-ADGroup -Name FinanceGroup -GroupScope Global -GroupCategory Security
```

Verify that SGs have been created

We verify the groups were created using Get-ADGroup and ensure they are placed in the correct OUs.

```
PS C:\Users\Administrator> Get-ADGroup -SearchBase "OU=IT,DC=visionmedia,DC=local" -Filter *

DistinguishedName : CN=ITGroup,OU=IT,DC=visionmedia,DC=local
GroupCategory      : Security
GroupScope         : Global
Name               : ITGroup
ObjectClass        : group
ObjectGUID         : 62a5d4c-d8b0-40d9-a7b3-39b66a18667d
SamAccountName     : ITGroup
SID                : S-1-3-21-2199384527-1268843624-1072581727-1104

PS C:\Users\Administrator> Get-ADGroup -SearchBase "OU=Finance,DC=visionmedia,DC=local" -Filter *

DistinguishedName : CN=FinanceGroup,OU=Finance,DC=visionmedia,DC=local
GroupCategory      : Security
GroupScope         : Global
Name               : FinanceGroup
ObjectClass        : group
ObjectGUID         : 63b57691-4f16-4435-8751-58a79deeee06
SamAccountName     : FinanceGroup
SID                : S-1-3-21-2199384527-1268843624-1072581727-1106

PS C:\Users\Administrator> Get-ADGroup -SearchBase "OU=HR,DC=visionmedia,DC=local" -Filter *

DistinguishedName : CN=HRGroup,OU=HR,DC=visionmedia,DC=local
GroupCategory      : Security
GroupScope         : Global
Name               : HRGroup
ObjectClass        : group
ObjectGUID         : 62a5d4c-d8b0-40d9-a7b3-39b66a18667d
```

Create .csv file with users

A CSV file is created with user information such as First Name, Last Name, Password, Department and Group. This file will serve as input for our script.

```
First Name, Last Name, Password, Department, Group
Samantha, Smith, P@ssw0rd1, IT, ITGroup
Kajal, Nair, P@ssw0rd2, Finance, FinanceGroup
Liam, Jacobs, P@ssw0rd3, HR, HRGroup
Lerato, Ngwenya, P@ssw0rd4, Sales, SalesGroup
Matthew, Vanderhorst, P@ssw0rd5, Finance, FinanceGroup
Ayisha, Abrahams, P@ssw0rd6, HR, HRGroup
Sipho, Zwane, P@ssw0rd7, IT, ITGroup
Chad, Fortune, P@ssw0rd8, Sales, SalesGroup
Nemutla, Botha, P@ssw0rd9, Finance, FinanceGroup
Devon, Pillay, P@ssw0rd10, IT, ITGroup
Taym, September, P@ssw0rd11, HR, HRGroup
Tehpo, Makhaba, P@ssw0rd12, Sales, SalesGroup
Megan, Williams, P@ssw0rd13, Finance, FinanceGroup
Fatima, Khan, P@ssw0rd14, IT, ITGroup
Dylan, Abels, P@ssw0rd15, HR, HRGroup
Zanele, Mlongo, P@ssw0rd16, Sales, SalesGroup
Shawn, Cassim, P@ssw0rd17, Finance, FinanceGroup
Anastasi, Olantoni, P@ssw0rd18, IT, ITGroup
Jade, Jules, P@ssw0rd19, HR, HRGroup
Desiree, Williams, P@ssw0rd20, IT, ITGroup
Tanaka, Mokuena, P@ssw0rd21, Sales, SalesGroup
Tebogo, Hodges, P@ssw0rd22, Finance, FinanceGroup
```

Create script that automates user creation & assign them OUs & SGs

We write a PowerShell script that reads the CSV file and automatically creates each user, placing them in the correct OU and adding them to the appropriate group.

```
# Create-users.ps1
# Load AD module
Import-Module ActiveDirectory

# Set CSV path
$CsvPath = "C:\Users\Administrator\Desktop\test_4"

# Read users from CSV
$Users = Import-Csv -Path $CsvPath

# Loop through each user and create AD account
foreach ($User in $Users) {
    $Surname = ($User.Firstname.Substring(0,1)) +
    $OU = "OU=($User.Department),DC=visionmedia,DC=local"

    # Create the user
    New-ADUser -Name "$($User.Firstname) $($User.Lastname)" -GivenName $User.Firstname -Surname $User.Lastname -SamAccountName $User.Firstname -UserPrincipalName "$($User.Firstname)@$($User.Department).visionmedia.local" -Enabled $true -Path $OU -Department $User.Department

    # Add to group
    Add-ADGroupMember -Identity $User.Group -Members $User.Surname

    # Confirmation message
    Write-Host "[+] Created user: $User.Surname and added to group: $($User.Group)" -ForegroundColor Green
}
```

Execute Script

We run the script on the domain controller. PowerShell processes the CSV and creates all users in one go it is fast, efficient, and repeatable.

```
PS C:\Users\Administrator\Desktop> ls

Directory: C:\Users\Administrator\Desktop

Mode                LastWriteTime         Length Name
----                -
-a----          7/24/2025 1:02 PM             585 all_test.csv
-a----          7/24/2025 12:40 PM             596 created_users.csv
-a----          7/24/2025 12:31 PM            1001 create_users.ps1
-a----          7/24/2025 12:33 PM             250 test_user.csv

PS C:\Users\Administrator\Desktop> .\create_users.ps1
Created user: Thabho and added to group: Marketing
Created user: gmatseiding and added to group: FinanceGroup
Created user: bathethwa and added to group: ITGroup
Created user: tshepo and added to group: ITGroup
```

Verify users have been created

We check that the users exist using Get-ADUser, and confirm they are in the right OUs and SGs by checking their properties.

```
PS C:\Users\Administrator> Get-ADUser -Filter * | Select-Object Name, SamAccountName, DistinguishedName

Name                SamAccountName DistinguishedName
-----
Administrator       Administrator    CN=Administrator,CN=Users,DC=visionmedia,DC=local
Guest               Guest          CN=Guest,CN=Users,DC=visionmedia,DC=local
krbtgt              krbtgt         CN=krbtgt,CN=Users,DC=visionmedia,DC=local
Thabho Mokena       thabho         CN=Thabho Mokena,OU=Sales,DC=visionmedia,DC=local
Lehako Mashilane   lemashilane   CN=Lehako Mashilane,OU=Sales,DC=visionmedia,DC=local
Jordi Nsenga       jnsenga       CN=Jordi Nsenga,OU=Sales,DC=visionmedia,DC=local
Seabelo Moutlong   smoutlong     CN=Seabelo Moutlong,OU=IT,DC=visionmedia,DC=local
Owethu Typu        otypu         CN=Owethu Typu,OU=Finance,DC=visionmedia,DC=local
Chutlong Masha     cmasha        CN=Chutlong Masha,OU=HR,DC=visionmedia,DC=local
Samantha Smith     smith         CN=Samantha Smith,OU=IT,DC=visionmedia,DC=local
Kajal Naidoo       knaidoo       CN=Kajal Naidoo,OU=Finance,DC=visionmedia,DC=local
Nkiam Jacobs       njacobs       CN=Nkiam Jacobs,OU=HR,DC=visionmedia,DC=local
Lerato Ngwenya     lngwenya      CN=Lerato Ngwenya,OU=Sales,DC=visionmedia,DC=local
Matthew Vandermerwe mrvandermerwe CN=Matthew Vandermerwe,OU=Finance,DC=visionmedia,DC=local
Ayisha Abraham    aabraham     CN=Ayisha Abraham,OU=Sales,DC=visionmedia,DC=local
Sipho Zwane        szwane       CN=Sipho Zwane,OU=IT,DC=visionmedia,DC=local
Chad Fortune       cfortune     CN=Chad Fortune,OU=Sales,DC=visionmedia,DC=local
Nomavola Buthelesi nbutheseles CN=Nomavola Buthelesi,OU=Finance,DC=visionmedia,DC=local
Devan Pillay       dpillay      CN=Devan Pillay,OU=IT,DC=visionmedia,DC=local
Tayari September  tshepo       CN=Tayari September,OU=Sales,DC=visionmedia,DC=local
Tshepo Mashaba    tmashaba     CN=Tshepo Mashaba,OU=Sales,DC=visionmedia,DC=local
Nkegan Devillers  ndevillers   CN=Nkegan Devillers,OU=Finance,DC=visionmedia,DC=local
Fatima Khan       fkhan        CN=Fatima Khan,OU=IT,DC=visionmedia,DC=local
Oryan Abels       oabels       CN=Oryan Abels,OU=HR,DC=visionmedia,DC=local
Zanele Mhlongo    zmhlongo     CN=Zanele Mhlongo,OU=Sales,DC=visionmedia,DC=local
Shuan Cassie     scassie     CN=Shuan Cassie,OU=Finance,DC=visionmedia,DC=local
Anathi Dlamini   adlamini     CN=Anathi Dlamini,OU=IT,DC=visionmedia,DC=local
Jade Jules       djules       CN=Jade Jules,OU=HR,DC=visionmedia,DC=local
Desiree Williams  dwilliams    CN=Desiree Williams,OU=IT,DC=visionmedia,DC=local
Tanaka Mkwena     tmkwena      CN=Tanaka Mkwena,OU=Sales,DC=visionmedia,DC=local
Tshepo Hodges    thodges     CN=Tshepo Hodges,OU=Finance,DC=visionmedia,DC=local
Lerato Nkoola    lnkoola     CN=Lerato Nkoola,OU=HR,DC=visionmedia,DC=local
Gary Matseding   gmatseding   CN=Gary Matseding,OU=Finance,DC=visionmedia,DC=local
Pedrick William  pwilliam     CN=Pedrick William,OU=IT,DC=visionmedia,DC=local
```

Filtering users

We use PowerShell filters (e.g., Where-Object) to search for specific users based on department, group membership, or name. This helps validate the structure.

```
PS C:\Users\Administrator> Get-ADUser -Filter * -SearchBase "OU=IT,DC=visionmedia,DC=local" | Select-Object Name, SamAccountName

Name                SamAccountName
-----
Seabelo Moutlong    smoutlong
Samantha Smith      smith
Sipho Zwane         szwane
Devan Pillay        dpillay
Fatima Khan         fkhan
Anathi Dlamini     adlamini
Desiree Williams    dwilliams
Pedrick William    pwilliam
Tshepo Matseding   tmatseding

PS C:\Users\Administrator> Get-ADUser -Filter * -SearchBase "OU=Sales,DC=visionmedia,DC=local" | Select-Object Name, SamAccountName

Name                SamAccountName
-----
Thabho Mokena       thabho
Lehako Mashilane   lemashilane
Jordi Nsenga       jnsenga
Lerato Ngwenya     lngwenya
Chad Fortune       cfortune
Tshepo Mashaba    tmashaba
Zanele Mhlongo    zmhlongo
Tanaka Mkwena     tmkwena
Bafan Mthethwa    bmthethwa
Lerato Nkoola    lnkoola
Shusiso Mvelase   smvelase
Thamsanqa Gamede  tgamede

PS C:\Users\Administrator> Get-ADUser -Filter * -SearchBase "OU=HR,DC=visionmedia,DC=local" | Select-Object Name, SamAccountName
```

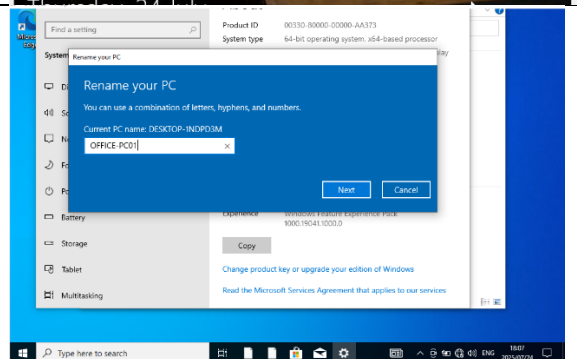
Login into Office-PC01

We now go to a client PC (Office-PC01) that will join the domain. We first log in using the local admin account.



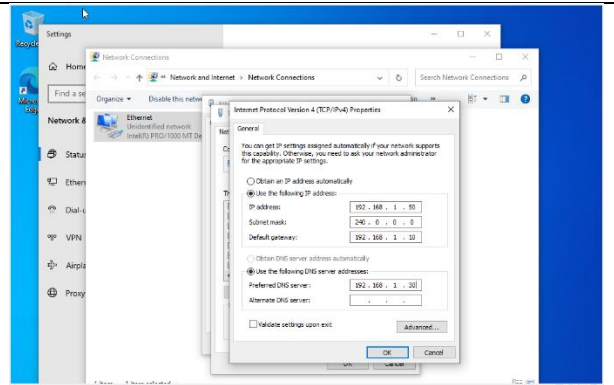
Rename Office-PC01

Just like the server, we rename the client machine to something recognizable (e.g., OfficePC01) for better network management.



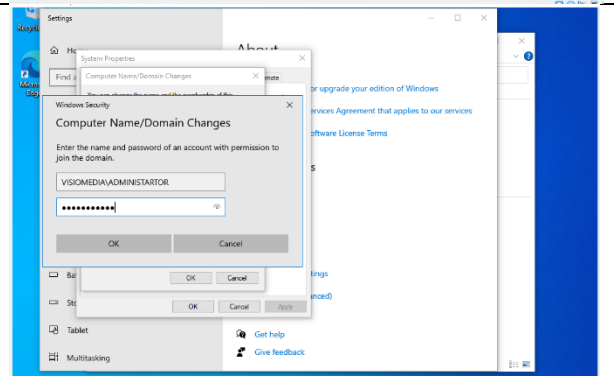
Changing IP Address and DNS

We set a static IP and assign the DNS server to the domain controller's IP address so the client can locate the domain.



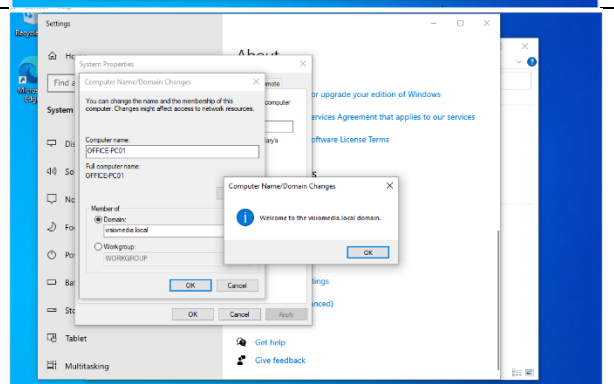
Joining Office-PC01 to domain

From Office-PC01, we joined the machine to the domain **visiomedia.local**, which is managed by the domain controller (our server at **192.168.1.30**). This step is important because it allows the client computer to become part of the domain environment and be managed centrally.



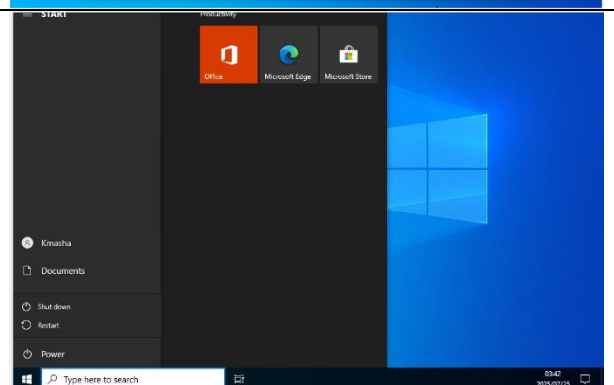
Office-PC01 joined the domain

We restart the machine and confirm that it successfully joined the domain.



Login in with user

Finally, we log in using one of the new AD user accounts created from the CSV. If everything works, we're now running in a fully domain-integrated environment.

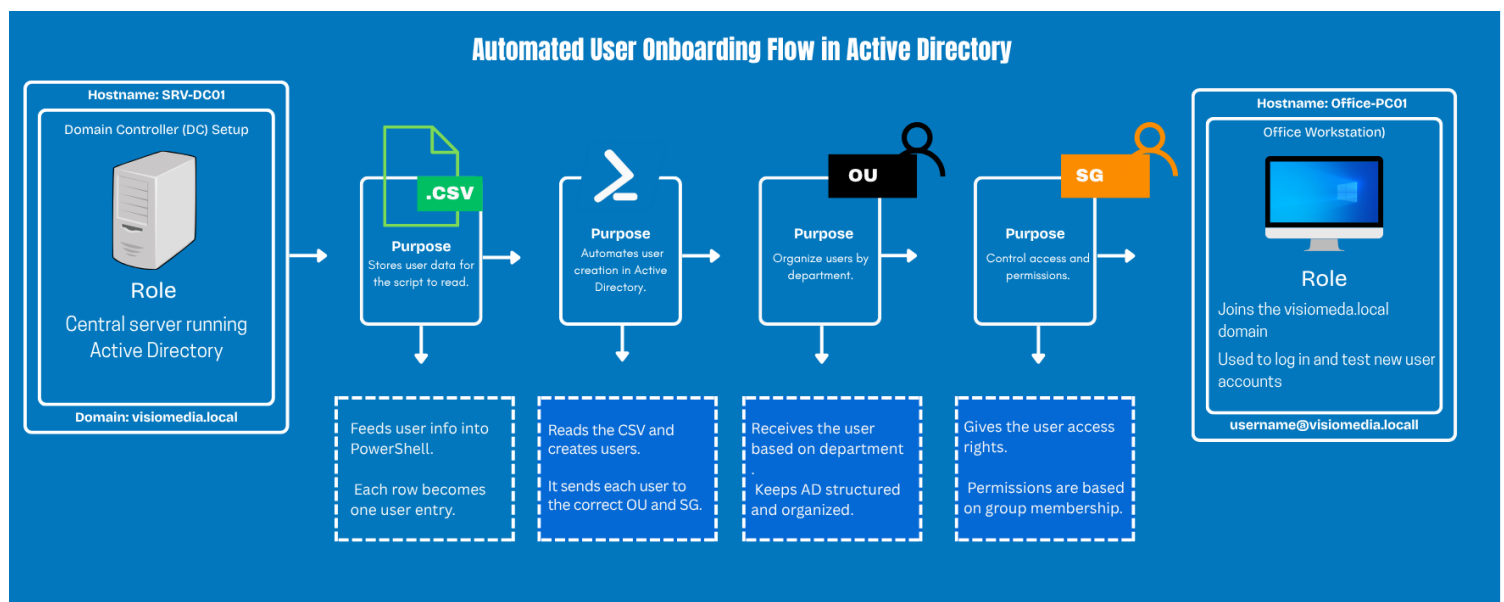


Project Summary

In this project, we set up a Windows Server 2022 environment and automated the creation of Active Directory (AD) users using PowerShell and a CSV file. We configured the domain controller, created organizational units (OUs) and security groups, and then used a script to bulk-add users and assign them to the correct groups and departments.

We also prepared a client machine (Office-PC01), configured its IP and DNS settings, and joined it to the domain directly from the PC itself. Finally, we verified that domain users could log in successfully, confirming that our automation and domain setup worked as intended.

This project demonstrates how scripting and proper AD structure can streamline user onboarding in a networked environment—an essential skill in IT administration.



Additional Improvements

Now that the core of the project is done, there are a few improvements I'd like to make to push it closer to a real-world setup:

Apply Group Policies (GPO)

I'd like to set up Group Policies to control user settings across the domain—things like enforcing strong passwords, limiting Control Panel access, or setting a default desktop wallpaper. It's a good way to maintain consistency and security.

Create Home Folders with Permissions

Another step would be to automatically create personal folders for each user on a shared drive and assign the right NTFS permissions. This is something you'd see in most companies and helps keep user files organized and private.

Automate Software Deployment

I'd also add a script or use GPO to push common apps like Chrome, 7-Zip, or antivirus software to all domain-joined PCs. It would save time and make sure every device has the tools it needs.

User Offboarding Script

As a follow-up to the onboarding script, I want to create a PowerShell script that can safely disable or remove users, unassign them from groups, and archive their folders when they leave the company.

Add Basic Security Auditing

Lastly, I'd enable basic security monitoring—like logging failed login attempts or tracking user activity with auditpol or Event Viewer. It's important for knowing what's happening on the network and responding to potential issues.