

How to use New-ADUser cmdlet in PowerShell

 lazyadmin.nl/powershell/new-aduser-cmdlet

July 13, 2024

Creating new users in the Active Directory is a common task for system administrators. You can do this manually in the ADUC, but a better and faster way is to use the New-ADUser cmdlet in PowerShell.

This cmdlet allows you to create a new user with all the required properties. I prefer to use an onboarding script to create new users. This allows you not only to create the user but also assign groups and licenses for example.

In this article, we will look at how to use the New-ADUser cmdlet and I will explain how you can create your own PowerShell script to create new users, **including a free script** to start with.

New-ADUser cmdlet

To create a new AD user with PowerShell you will need to have the Active Directory PowerShell module installed. This module is installed by default on the domain controllers, but you can also install it on your own workstation. Check out this article for more information on how to install it.

The New-ADUser cmdlet comes with a lot of options. We can basically set every property of the user while we create a new account. Now you don't need to specify every property, you could even create a new user account by just specifying the name:

```
New-ADUser -name "Z.Vance"
```

But this will create the user account in the default Users OU, without any attributes. So this is not really a realistic option.

The minimum information needed to create a usable new user account is:

- Given name
- Surname
- Name
- SamAccountName
- UserPrincipalName
- Path (OU)
- Password (AccountPassword)
- Enabled – True
- ChangePasswordAtLogon – True

In the example below, we have hardcoded the information, but we can also use a Read-Host cmdlet to make it a bit more dynamic of course. I recommend using a hashtable and splatting to make the code a bit more readable:

```

$firstName = "Zoe"
$lastName = "Vance"
$accountName = "zoevance"
$dnsroot = '@' + (Get-ADDomain).dnsroot
$OU = "OU=Users,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=nl"
$password = "lazyPass123"
$userDetails = @{
    GivenName = $firstName
    Surname = $lastName
    Name = "$firstName $lastName"
    SamAccountName = $accountName
    UserPrincipalName = $accountName + $dnsroot
    Path = $OU
    AccountPassword = (ConvertTo-SecureString -AsPlainText $password -Force)
    Enabled = $true
    ChangePasswordAtLogon = $true
}
New-ADUser @userDetails

```

Normally when you create a user account, you also want to enter details like contact information, which can be used in email signatures, for example, the manager of the user and probably group memberships.

Create User Account with Properties

The **New-ADUser** cmdlet allows you to specify all the properties that you need when creating a user account. You can for example enter all the address details and phone numbers of the user by adding the related parameters:

```

$userDetails = @{
    GivenName = $firstName
    Surname = $lastName
    Name = "$firstName $lastName"
    SamAccountName = $accountName
    UserPrincipalName = $accountName + $dnsroot
    Path = $OU
    AccountPassword = (ConvertTo-SecureString -AsPlainText $password -Force)
    Enabled = $true
    ChangePasswordAtLogon = $true
    StreetAddress = "Karl Johans gate 1"
    PostalCode = "0010"
    City = "Oslo"
    OfficePhone = "010 000 458 456"
}
New-ADUser @userDetails

```

The cmdlet allows you to enter the commonly used parameters directly, but a user account in the Active Directory also can have extension or custom attributes. These can also be populated through PowerShell, by using the `OtherAttributes` parameter.

To enter information in the `extensionattribute1`, for example, we can add the following parameter:

```
-OtherAttributes @{extensionattribute1="Special"}
```

Adding the Manager

Adding the manager to a user account in the Active Directory allows you, but also your users, to see a person's manager and the manager's direct reports in organization views for example. To add a manager to a new user account we first need to get the `SamAccountName` from the manager.

The easiest option is to find the manager based on the full name. This way we can simply add the `SamAccountName` to the manager parameter when creating a new account:

```
$manager = Get-AdUser -Filter {name -like "Adel Bowen"}  
New-ADUser "<Add other properties>" -Manager $manager.SamAccountName
```

Using a Template

Instead of supplying all the properties when creating a new user account, we can also use an existing user account as a template. This way we only need to supply the properties that are changing and don't need to reenter the address information or group membership for example.

The first step is to get the user account that you want to use as a template. You will need to specify which properties you want to copy by using the `-Properties` parameter.

```
$templateUser = Get-ADUser -Identity a.bowen -Properties  
City,StreetAddress,PostalCode
```

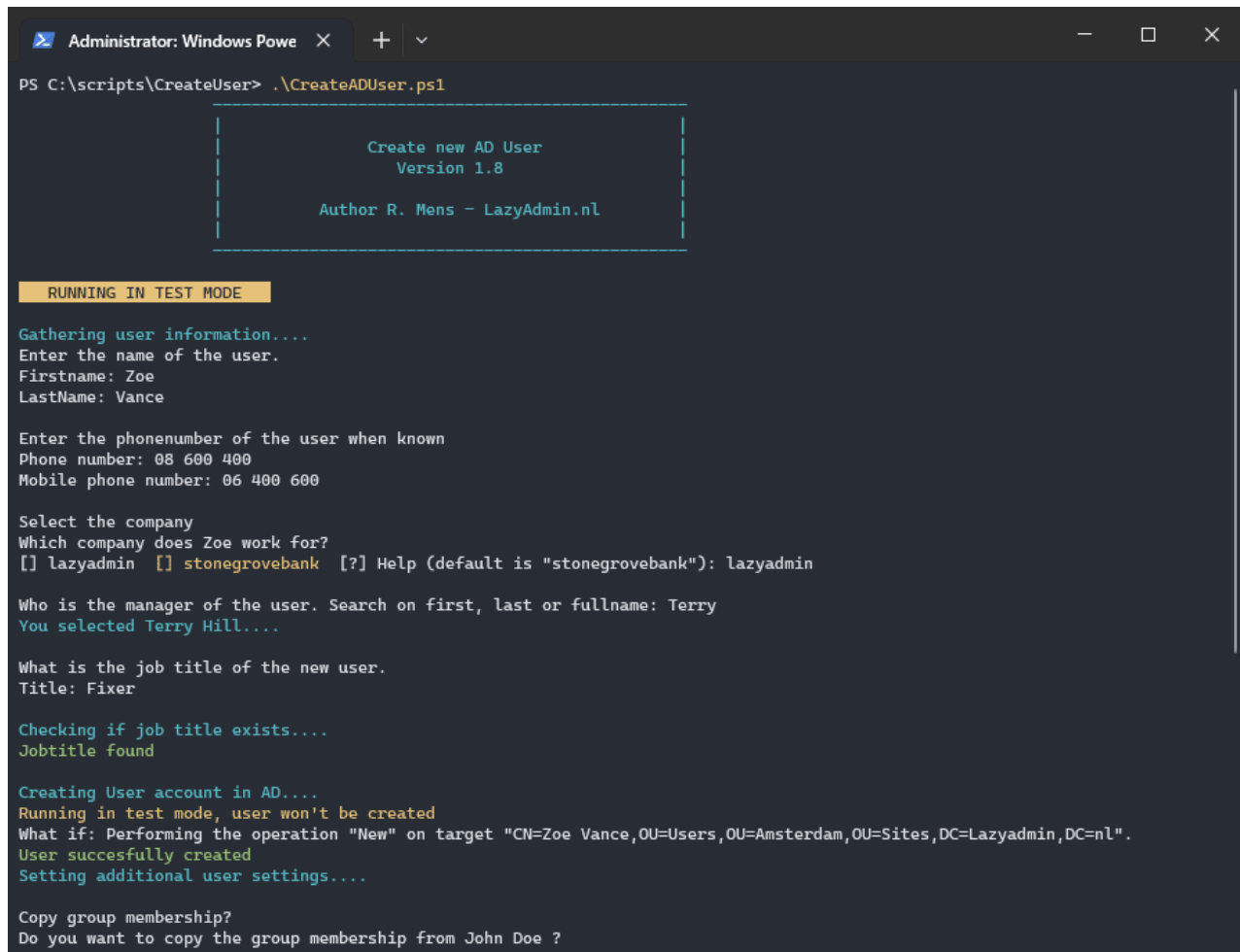
Next, we are going to create a new user account. We will need to supply the minimum information that I mentioned before in the article, including the OU (Path). You can then use the `-Instance` parameter to specify the template user where you want to copy the information from:

```
New-ADUser -Instance $templateUser -SamAccountName $accountName -Name  
$fullName -otherproperties
```

Even though this method does sound nice, I don't really like it. You still need to supply the main information and select every field you want to copy. It's better to spend some time creating a proper script that will do the most work for you.

Creating a New User PowerShell Script

When you need to create new user accounts quite often, it's best to spend some time and create a good PowerShell script. The advantage of using a script is that you can automate every step that needs to be taken to fully onboard a new user.



```
Administrator: Windows Powe X + v
PS C:\scripts\CreateUser> .\CreateADUser.ps1

Create new AD User
Version 1.8

Author R. Mens - LazyAdmin.nl

RUNNING IN TEST MODE

Gathering user information....
Enter the name of the user.
Firstname: Zoe
LastName: Vance

Enter the phonenumber of the user when known
Phone number: 08 600 400
Mobile phone number: 06 400 600

Select the company
Which company does Zoe work for?
[] lazyadmin [] stonegrovebank [?] Help (default is "stonegrovebank"): lazyadmin

Who is the manager of the user. Search on first, last or fullname: Terry
You selected Terry Hill....

What is the job title of the new user.
Title: Fixer

Checking if job title exists....
Jobtitle found

Creating User account in AD....
Running in test mode, user won't be created
What if: Performing the operation "New" on target "CN=Zoe Vance,OU=Users,OU=Amsterdam,OU=Sites,DC=Lazyadmin,DC=nl".
User succesfully created
Setting additional user settings....

Copy group membership?
Do you want to copy the group membership from John Doe ?
```

Every organization is different, so I don't think my script will work for you out of the box. This means that you will need to modify the script to your needs. That's why I will first explain what the script exactly does, and explain some of the important steps. You can then [download the script](#) and modify it.

So when creating a new user account, we need the following to be done:

- Get the personal details of the new user. We only need the name, phone number, job title, company (in case of multiple offices) and manager
- Get address information etc from a config file
- Create the user account
- Copy the group membership from an existing user that has the same job title, and works for the same company
- Force the Microsoft Entra Sync (if you have a hybrid environment)
- Send an email to the manager that the account has been created

As you can see in the steps above, we are using a config file. This is nothing more than a JSON file with all the static information. It allows us to define multiple branch offices, each with its own address information. We also use it to define some of the settings, like

passwords.

```
{
  "SMTP": {
    "address": "lazydev-onmicrosoft-com.mail.protection.outlook.com",
    "from": "Lazy IT Desk <it@lazyadmin.nl>",
    "serviceDesk": "it@lazyadmin.nl",
    "subject": "User Account details for {{user.fullname}}",
    "attachment": "\\la-srv-dc01\\scripts\\assets\\new user manual.pdf"
  },
  "Settings": {
    "mailTemplateManager": "MailTemplateUserCreated.html",
    "mailTemplateServiceDesk": "MailTemplateServiceDesk.html",
    "password": "lazyPass123",
    "AccountSkuld": "lazydev:DEVELOPERPACK_E5"
  },
  "Companies": {
    "lazyadmin": {
      "City": "Amsterdam",
      "Name": "LazyAdmin",
      "OU": "OU=Users,OU=Amsterdam,OU=Sites,DC=Lazyadmin,DC=nl",
      "Phone": "088 11 33 33 77",
      "PostalCode": "1337 AB",
      "WebSite": "lazyadmin.nl",
      "StreetAddress": "Handelsplein 1"
    }
  }
}
```

As you can see in the JSON, we are also defining two mail templates and an attachment. The attachments are instructions on how to get started, what they need to know, etc.

Getting the Personal Information

The first step is to get the personal information of the new user. The script is intended to create a single user at a time, but you could also modify it for bulk creating, using the Import-CSV cmdlet.

We are using a couple of Read-Host cmdlets, and using the Write-Host cmdlets to give some guides:

```
Write-Host "Enter the name of the user."
$user.givenName = Read-Host "Firstname"
$user.surName = Read-Host "LastName"
Write-Host "Enter the phonenumber of the user when known"
$user.telephoneNumber = Read-Host "Phone number"
$user.mobilePhone = Read-Host "Mobile phone number"
```

```
$user.fullName = ($user.givenName + ' ' + $user.surName)
```

Selecting the Company

If you are working in a large environment you often have to deal with different branch offices or departments. In the JSON file, we can define multiple companies, each with their own address details, etc.

We want to select one of those companies. To do this, we are building a choice menu that allows the user (you), to select one of the options.

```
If ($companyList.Count -gt 1) {  
    $title = "Select the company"  
    $message = "Which company does $usersName work for?"  
    # Build the choices menu  
    $choices = @()  
    For ($index = 0; $index -lt $companyList.Count; $index++) {  
        $choices += New-Object System.Management.Automation.Host.ChoiceDescription  
        ($companyList[$index]), ($companyList[$index])  
    }  
    $options = [System.Management.Automation.Host.ChoiceDescription[]]$choices  
    $result = $host.ui.PromptForChoice($title, $message, $options, 1)  
    $company = $companies.($companyList[$result])  
}
```

Getting the Manager

The last piece of information that we need is the manager of the user. We select the manager by simply searching for the manager's name. This can be the first name, last name, or full name. Now there is always a chance that we find multiple users with the same name.

```
$managerName = Read-Host "Who is the manager of the user. Search on first, last or  
fullname"
```

```
$manager = Get-Manager -name $managerName
```

We deal with this problem in the function [Get-Manager](#). I am not going to show the full code of the function here, you can check that here on Github, but if we find multiple users, then the script will create another choice menu, so you can select the correct manager.

Creating the SamAccountName and UserPrincipalName

With the information that we now have, we can create the SamAccountName and the UserPrincipalName. The SamAccountName has some requirements that we need to keep in mind. It can't be longer than 20 characters for example, and you want to define your format.

I prefer to keep the logon name (SamAccountName) the same format as in the email addresses. The current principal for the name is `firstname + lastname`. So for example:

- John Doe > johndoe
- Klaas de Vries > klaasdevries

The SamAccountName also has to be unique within the Active Directory. If the name already exists, it will create the following alternative:

- John Doe > jdoe
- Klaas de Vries > kdevries

Now you can change this of course to match your environment.

If the SamAccountName is created, we can use the name to create the `UserPrincipalName`. This is SamAccountName + the company domain name.

Job title

The job title might look like a simple property, but we need to make sure that the job title exists within the OU that we are going to create the user. The reason for this is, that we are going to copy the group memberships later on based on the job title and OU of the user.

```
write-host "What is the job title of the new user."
```

```
$title = Read-Host "Title"
```

```
write-host "`n"
```

```
write-host "Checking if job title exists...." -ForegroundColor Cyan
```

```
$user.title = Get-JobTitle -title $title -company $company
```

So we ask for the job title in the script and check if it exists in the function `Get-JobTitle`. This also prevents multiple variations (and typos) of the same title. If the script can't find a matching job title, then it will return all job titles from the users within the same OU and allow you to select one.

Creating a New User Account

With all the information collected, we can create the new user account. The script supports a test mode (WhatIf), so you can try it out without creating an actual user.

```
Write-Host 'Creating User account in AD....' -ForegroundColor Cyan
```

```
if ($whatIf) {
```

```
Write-Host "Running in test mode, user won't be created" -ForegroundColor Yellow  
}
```

```
$userCreated = New-DomainUserAccount -user $user -manager $manager -company  
$company -whatIf:$whatIf
```

```
# Only continue when account is created or when running in whatif (test) mode
```

```
If ($userCreated -or $whatIf -eq $true) {
```

```
Write-Host 'User succesfully created' -ForegroundColor Green
```

The function `New-DomainUserAccount` is a custom function, but it does nothing special. It uses a splatted hashtable, as I have shown you at the beginning of the article, and the `New-ADUser` cmdlet to create the user.

Copying Group membership

With the user account created, we need to set the group membership of the new user. The best way to do this is to copy it from an existing user which has the same job title and works for the same company (department/branch office).

The first step is to get the newly created user account and the user to copy from. The function `Get-UserToCopyGroupsFrom` will search the Active Directory for all users based on the job title and in the same OU. It will of course exclude the newly created user.

```
# Copy Group Membership
```

```
$createdUser = Get-AdUser -Identity $user.SamAccountName -Properties *
```

```
# Find user to copy group membership from
```

```
$userToCopyFrom = Get-UserToCopyGroupsFrom -user $createdUser -company  
$company
```

It only returns one user, selecting the last created account and asking for confirmation before it copies the group membership.

```
If ($userToCopyFrom) {
```

```
# Copy group membership from user?
```

```
$title = "Copy group membership?"
```

```
$message = "Do you want to copy the group membership from " +
```

```
$userToCopyFrom.name + " ?"
```

```
$yes = New-Object System.Management.Automation.Host.ChoiceDescription "&Yes",  
"Yes"
```

```
$no = New-Object System.Management.Automation.Host.ChoiceDescription "&No", "No"
```

```
$options = [System.Management.Automation.Host.ChoiceDescription[]]($yes, $no)
```

```
$copyMembership = $host.ui.PromptForChoice($title, $message, $options, 0)
```

```
if ($copyMembership -eq 0 -and $whatIf -eq $false) {
```

```
Set-GroupMemberShip -user $user.SamAccountName -copyFrom
```

```
$userToCopyFrom.SamAccountName -whatIf $whatIf
```

```
}
```

```
if ($whatIf) {
```

```
Write-Host "Copy group memberships from $userToCopyFrom.SamAccountName"
```

```
}
```

```
}
```

Sending Emails

A part of the process of creating a new user account is also to inform the manager that the account is created. You can expand this to even the HR department and helpdesk if needed.

To send the emails, we are using an email template, which is a HTML file with placeholders in it. We can get the contents of the HTML file and by using the string replace function in PowerShell, we can modify the content with the details of the new user account.

```
$emailBody = Get-EmailTemplate -user $user -Manager $manager  
Send-MailtoManager -user $user -manager $manager -EmailBody $emailBody -whatIf  
$whatIf
```

Force Entra Connect Sync

The last step is to force the Entra Connect Sync so that the new user account is immediately synced to Microsoft 365. Depending on where you are running the script, we can use the Start-ADSyncSyncCycle cmdlet or use the Invoke-Command to run the command on the domain controller.

```
Write-Host "Syncing Azure AD Connect...." -ForegroundColor Cyan  
# Run command on local domain controller  
Start-ADSyncSyncCycle -PolicyType Delta  
# Run sync command on remote domain controller  
# Invoke-Command -ComputerName lazy-srv-dc02 -ScriptBlock {Start-ADSyncSyncCycle  
-PolicyType Delta}
```

Wrapping Up

Creating a new user account with the New ADUser cmdlet is pretty easy, but it does require entering a lot of information. Therefore the true power of this cmdlet comes when using it in an onboarding script.

You can use my script as a starting point for your own environment. Make sure that you run it in test mode the first couple of times. The script will output what it does, so you can check every step.

Hope you liked this article, if you have any questions, just drop a comment below.

Did you **Liked** this **Article**?

Get the latest articles like this **in your mailbox**
or share this article

I hate spam to, so you can unsubscribe at any time.