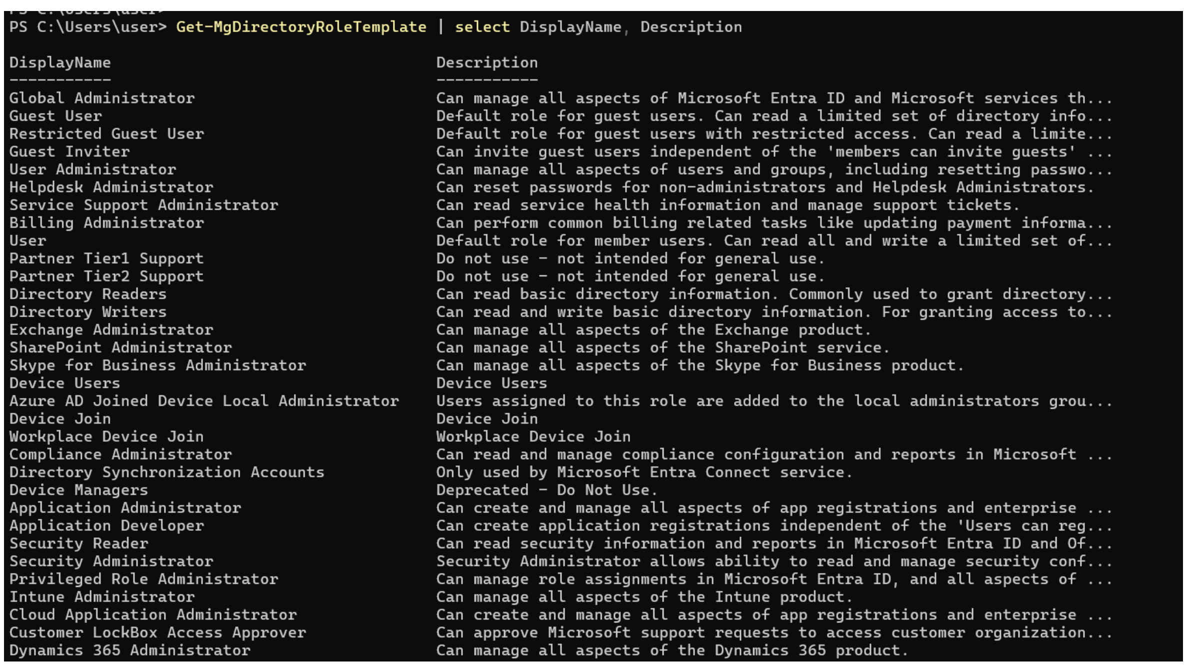
Lab role assignments and access control

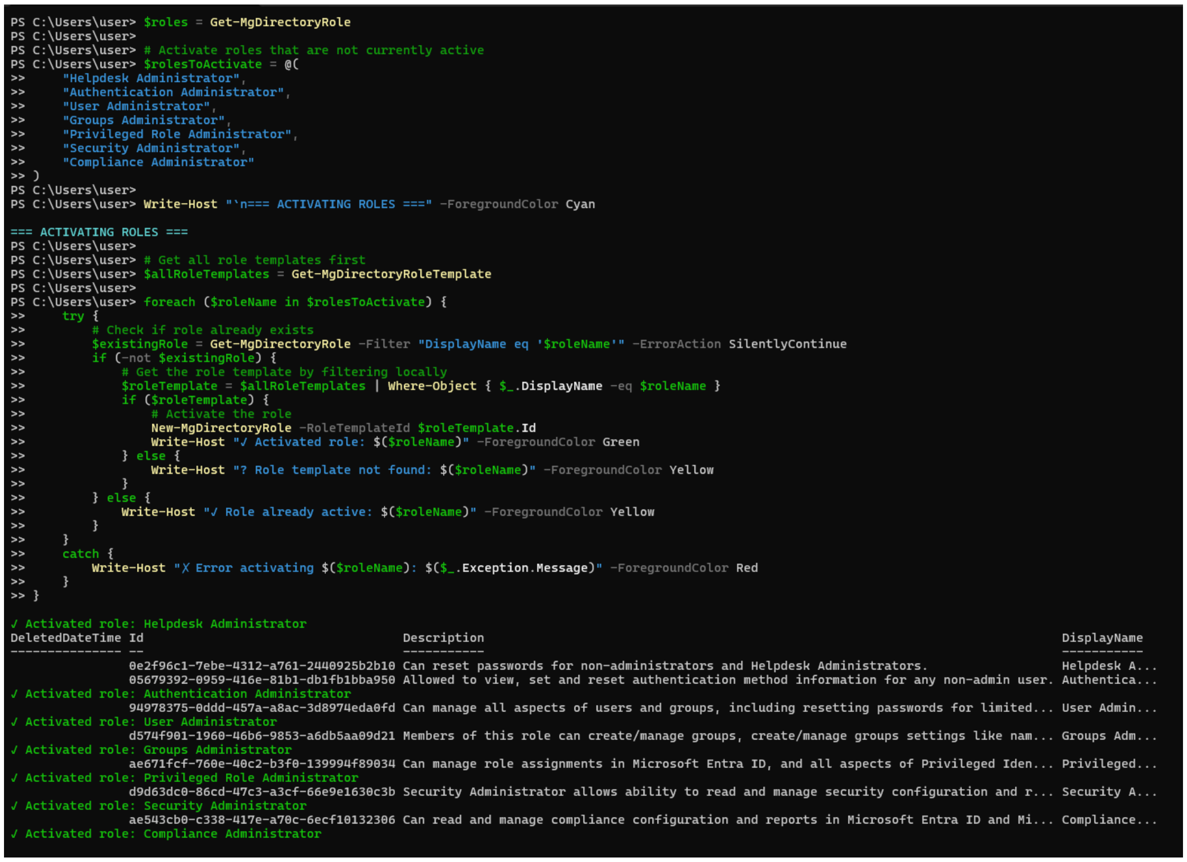
Objective, learn how to assign **Azure AD roles** to users and groups, practice the principle of least privilege, and configure a break-glass emergency account.

Part I. Review available roles



```
PS C:\Users\user> Get-MgDirectoryRoleTemplate | select DisplayName, Description

DisplayName                              Description
-----------                              -----------
Global Administrator                     Can manage all aspects of Microsoft Entra ID and Microsoft services th...
Guest User                               Default role for guest users. Can read a limited set of directory info...
Restricted Guest User                    Default role for guest users with restricted access. Can read a limite...
Guest Inviter                            Can invite guest users independent of the 'members can invite guests' ...
User Administrator                       Can manage all aspects of users and groups, including resetting passwo...
Helpdesk Administrator                   Can reset passwords for non-administrators and Helpdesk Administrators.
Service Support Administrator            Can read service health information and manage support tickets.
Billing Administrator                    Can perform common billing related tasks like updating payment informa...
User                                     Default role for member users. Can read all and write a limited set of...
Partner Tier1 Support                    Do not use - not intended for general use.
Partner Tier2 Support                    Do not use - not intended for general use.
Directory Readers                        Can read basic directory information. Commonly used to grant directory...
Directory Writers                        Can read and write basic directory information. For granting access to...
Exchange Administrator                   Can manage all aspects of the Exchange product.
SharePoint Administrator                 Can manage all aspects of the SharePoint service.
Skype for Business Administrator         Can manage all aspects of the Skype for Business product.
Device Users                             Device Users
Azure AD Joined Device Local Administrator  Users assigned to this role are added to the local administrators grou...
Device Join                              Device Join
Workplace Device Join                    Workplace Device Join
Compliance Administrator                 Can read and manage compliance configuration and reports in Microsoft ...
Directory Synchronization Accounts       Only used by Microsoft Entra Connect service.
Device Managers                          Deprecated - Do Not Use.
Application Administrator                Can create and manage all aspects of app registrations and enterprise ...
Application Developer                    Can create application registrations independent of the 'Users can reg...
Security Reader                          Can read security information and reports in Microsoft Entra ID and Of...
Security Administrator                   Security Administrator allows ability to read and manage security conf...
Privileged Role Administrator            Can manage role assignments in Microsoft Entra ID, and all aspects of ...
Intune Administrator                     Can manage all aspects of the Intune product.
Cloud Application Administrator          Can create and manage all aspects of app registrations and enterprise ...
Customer LockBox Access Approver         Can approve Microsoft support requests to access customer organization...
Dynamics 365 Administrator               Can manage all aspects of the Dynamics 365 product.
```

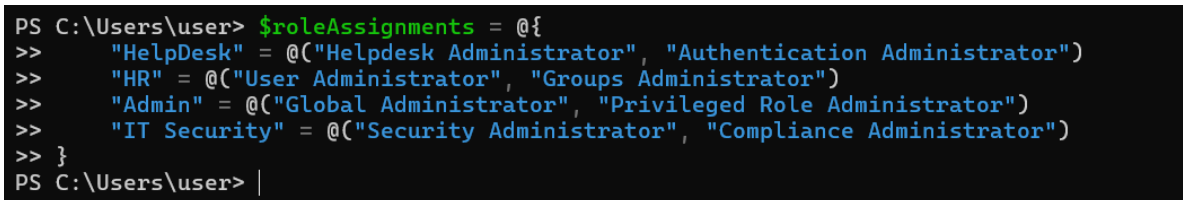
Next step we activate the roles that we want to give to the groups



```
PS C:\Users\user> $roles = Get-MgDirectoryRole
PS C:\Users\user>
PS C:\Users\user> # Activate roles that are not currently active
PS C:\Users\user> $rolesToActivate = @(
>>     "Helpdesk Administrator",
>>     "Authentication Administrator",
>>     "User Administrator",
>>     "Groups Administrator",
>>     "Privileged Role Administrator",
>>     "Security Administrator",
>>     "Compliance Administrator"
>> )
PS C:\Users\user>
PS C:\Users\user> Write-Host "`n=== ACTIVATING ROLES ===" -ForegroundColor Cyan

=== ACTIVATING ROLES ===
PS C:\Users\user>
PS C:\Users\user> # Get all role templates first
PS C:\Users\user> $allRoleTemplates = Get-MgDirectoryRoleTemplate
PS C:\Users\user>
PS C:\Users\user> foreach ($roleName in $rolesToActivate) {
>>     try {
>>         # Check if role already exists
>>         $existingRole = Get-MgDirectoryRole -Filter "DisplayName eq '$roleName'" -ErrorAction SilentlyContinue
>>         if (-not $existingRole) {
>>             # Get the role template by filtering locally
>>             $roleTemplate = $allRoleTemplates | Where-Object { $_.DisplayName -eq $roleName }
>>             if ($roleTemplate) {
>>                 # Activate the role
>>                 New-MgDirectoryRole -RoleTemplateId $roleTemplate.Id
>>                 Write-Host "√ Activated role: $($roleName)" -ForegroundColor Green
>>             } else {
>>                 Write-Host "? Role template not found: $($roleName)" -ForegroundColor Yellow
>>             }
>>         } else {
>>             Write-Host "√ Role already active: $($roleName)" -ForegroundColor Yellow
>>         }
>>     }
>>     catch {
>>         Write-Host "X Error activating $($roleName): $($_.Exception.Message)" -ForegroundColor Red
>>     }
>> }

√ Activated role: Helpdesk Administrator
DeletedDateTime Id                                   Description                                                                       DisplayName
--------------- --                                   -----------                                                                       -----------
                0e2f96c1-7ebe-4312-a761-2440925b2b10 Can reset passwords for non-administrators and Helpdesk Administrators.            Helpdesk A...
                05679392-0959-416e-81b1-db1fb1bba950 Allowed to view, set and reset authentication method information for any non-admin user. Authentica...
√ Activated role: Authentication Administrator
                94978375-0ddd-457a-a8ac-3d8974eda0fd Can manage all aspects of users and groups, including resetting passwords for limited... User Admin...
√ Activated role: User Administrator
                d574f901-1960-46b6-9853-a6db5aa09d21 Members of this role can create/manage groups, create/manage groups settings like nam... Groups Adm...
√ Activated role: Groups Administrator
                ae671fcf-760e-40c2-b3f0-139994f89034 Can manage role assignments in Microsoft Entra ID, and all aspects of Privileged Iden... Privileged...
√ Activated role: Privileged Role Administrator
                d9d63dc0-86cd-47c3-a3cf-66e9e1630c3b Security Administrator allows ability to read and manage security configuration and r... Security A...
√ Activated role: Security Administrator
                ae543cb0-c338-417e-a70c-6ecf10132306 Can read and manage compliance configuration and reports in Microsoft Entra ID and Mi... Compliance...
√ Activated role: Compliance Administrator
```

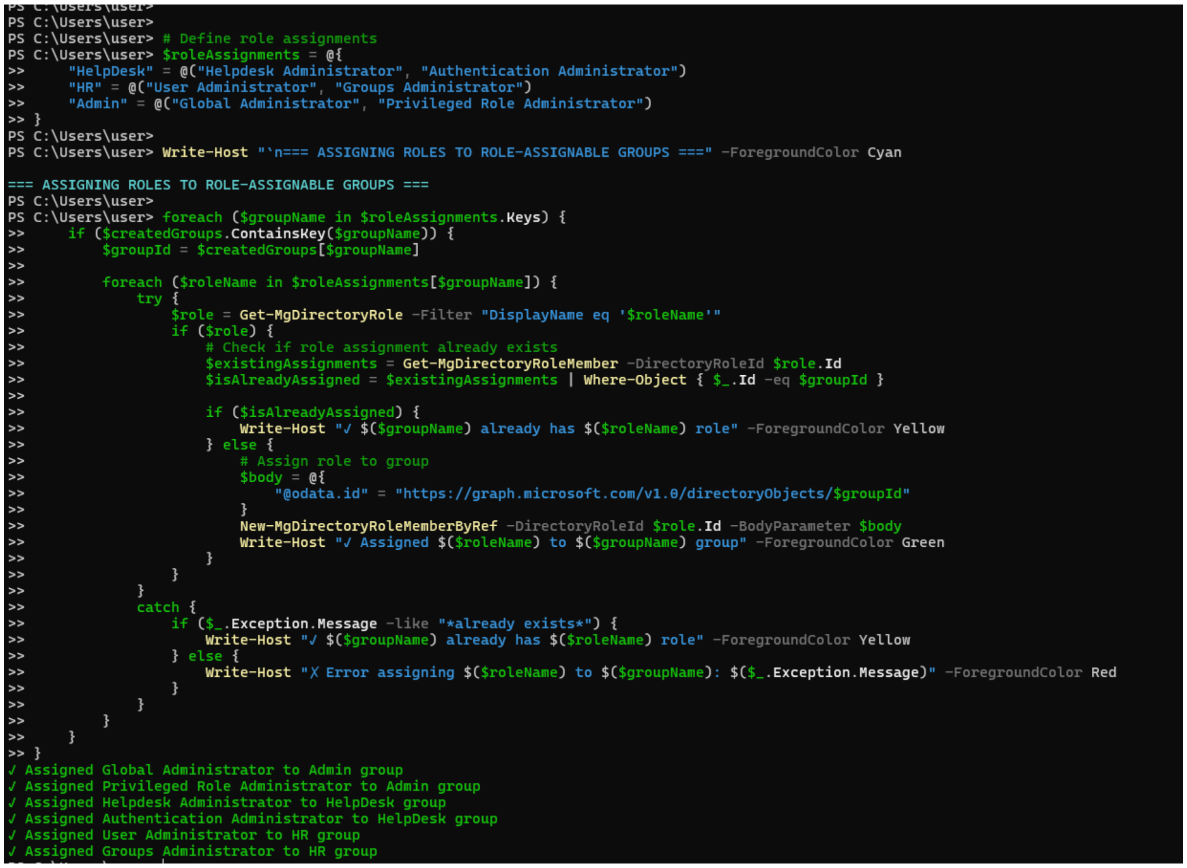
Then let's to define each role for group.

Group	Role 1	Role 2
Help Desk	HelpDesk administrator	Authentication admin
HR	User admin	Groups admin
Admin	Global admin	Groups admin



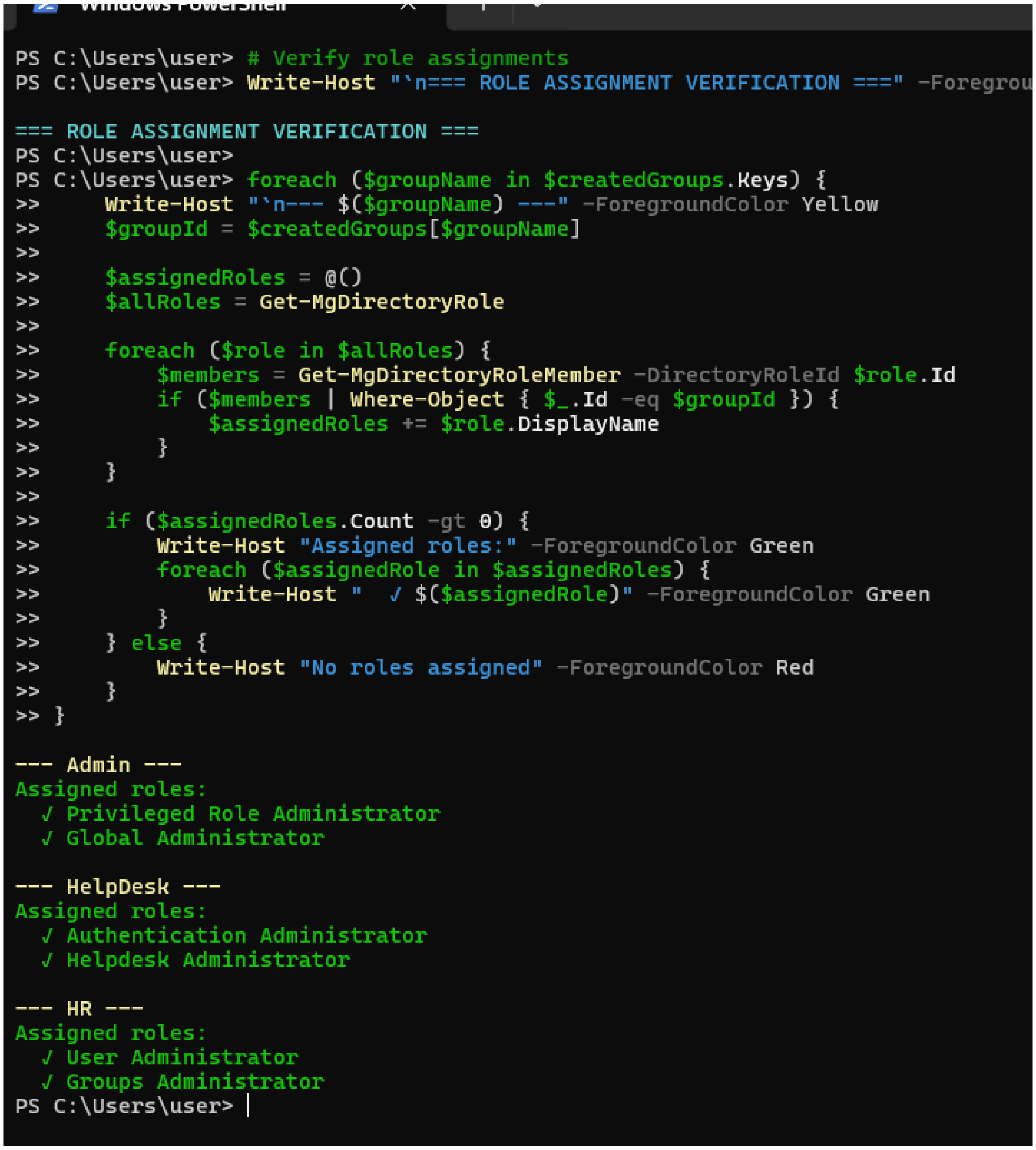
```
PS C:\Users\user> $roleAssignments = @{
>>     "HelpDesk" = @("Helpdesk Administrator", "Authentication Administrator")
>>     "HR" = @("User Administrator", "Groups Administrator")
>>     "Admin" = @("Global Administrator", "Privileged Role Administrator")
>>     "IT Security" = @("Security Administrator", "Compliance Administrator")
>> }
PS C:\Users\user> |
```

Once we have well defined each aspect of the roles, continue assign each role



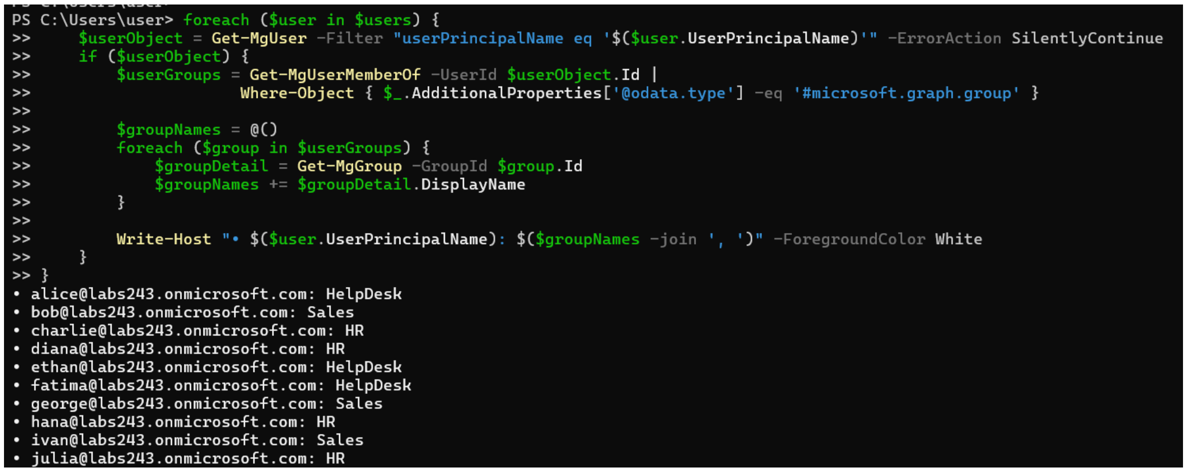
```
PS C:\Users\user>
PS C:\Users\user>
PS C:\Users\user> # Define role assignments
PS C:\Users\user> $roleAssignments = @{
>>     "HelpDesk" = @("Helpdesk Administrator", "Authentication Administrator")
>>     "HR" = @("User Administrator", "Groups Administrator")
>>     "Admin" = @("Global Administrator", "Privileged Role Administrator")
>> }
PS C:\Users\user>
PS C:\Users\user> Write-Host "`n=== ASSIGNING ROLES TO ROLE-ASSIGNABLE GROUPS ===" -ForegroundColor Cyan

=== ASSIGNING ROLES TO ROLE-ASSIGNABLE GROUPS ===
PS C:\Users\user>
PS C:\Users\user> foreach ($groupName in $roleAssignments.Keys) {
>>     if ($createdGroups.ContainsKey($groupName)) {
>>         $groupId = $createdGroups[$groupName]
>>
>>         foreach ($roleName in $roleAssignments[$groupName]) {
>>             try {
>>                 $role = Get-MgDirectoryRole -Filter "DisplayName eq '$roleName'"
>>                 if ($role) {
>>                     # Check if role assignment already exists
>>                     $existingAssignments = Get-MgDirectoryRoleMember -DirectoryRoleId $role.Id
>>                     $isAlreadyAssigned = $existingAssignments | Where-Object { $_.Id -eq $groupId }
>>
>>                     if ($isAlreadyAssigned) {
>>                         Write-Host "√ $($groupName) already has $($roleName) role" -ForegroundColor Yellow
>>                     } else {
>>                         # Assign role to group
>>                         $body = @{
>>                             "@odata.id" = "https://graph.microsoft.com/v1.0/directoryObjects/$groupId"
>>                         }
>>                         New-MgDirectoryRoleMemberByRef -DirectoryRoleId $role.Id -BodyParameter $body
>>                         Write-Host "√ Assigned $($roleName) to $($groupName) group" -ForegroundColor Green
>>                     }
>>                 }
>>             }
>>             catch {
>>                 if ($_.Exception.Message -like "*already exists*") {
>>                     Write-Host "√ $($groupName) already has $($roleName) role" -ForegroundColor Yellow
>>                 } else {
>>                     Write-Host "X Error assigning $($roleName) to $($groupName): $($_.Exception.Message)" -ForegroundColor Red
>>                 }
>>             }
>>         }
>>     }
>> }
√ Assigned Global Administrator to Admin group
√ Assigned Privileged Role Administrator to Admin group
√ Assigned Helpdesk Administrator to HelpDesk group
√ Assigned Authentication Administrator to HelpDesk group
√ Assigned User Administrator to HR group
√ Assigned Groups Administrator to HR group
```

Now let's check if the roles were assigned correctly.



```
PS C:\Users\user> # Verify role assignments
PS C:\Users\user> Write-Host "`n=== ROLE ASSIGNMENT VERIFICATION ===" -Foregrou

=== ROLE ASSIGNMENT VERIFICATION ===
PS C:\Users\user>
PS C:\Users\user> foreach ($groupName in $createdGroups.Keys) {
>>     Write-Host "`n--- $($groupName) ---" -ForegroundColor Yellow
>>     $groupId = $createdGroups[$groupName]
>>
>>     $assignedRoles = @()
>>     $allRoles = Get-MgDirectoryRole
>>
>>     foreach ($role in $allRoles) {
>>         $members = Get-MgDirectoryRoleMember -DirectoryRoleId $role.Id
>>         if ($members | Where-Object { $_.Id -eq $groupId }) {
>>             $assignedRoles += $role.DisplayName
>>         }
>>     }
>>
>>     if ($assignedRoles.Count -gt 0) {
>>         Write-Host "Assigned roles:" -ForegroundColor Green
>>         foreach ($assignedRole in $assignedRoles) {
>>             Write-Host "  ✓ $($assignedRole)" -ForegroundColor Green
>>         }
>>     } else {
>>         Write-Host "No roles assigned" -ForegroundColor Red
>>     }
>> }

--- Admin ---
Assigned roles:
  ✓ Privileged Role Administrator
  ✓ Global Administrator

--- HelpDesk ---
Assigned roles:
  ✓ Authentication Administrator
  ✓ Helpdesk Administrator

--- HR ---
Assigned roles:
  ✓ User Administrator
  ✓ Groups Administrator
PS C:\Users\user> |
```

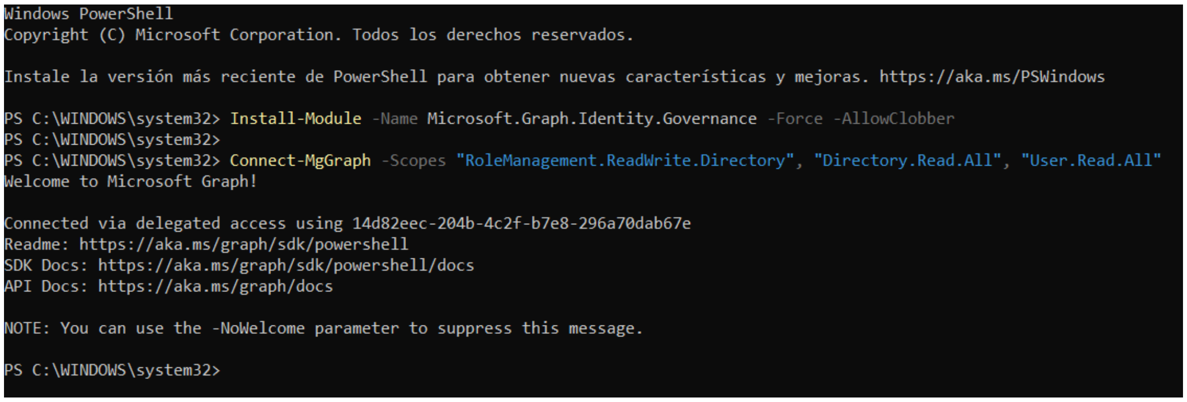
Now we can verify If all users were adding correctly.



```
PS C:\Users\user> foreach ($user in $users) {
>>     $userObject = Get-MgUser -Filter "userPrincipalName eq '$($user.UserPrincipalName)'" -ErrorAction SilentlyContinue
>>     if ($userObject) {
>>         $userGroups = Get-MgUserMemberOf -UserId $userObject.Id |
>>                       Where-Object { $_.AdditionalProperties['@odata.type'] -eq '#microsoft.graph.group' }
>>
>>         $groupNames = @()
>>         foreach ($group in $userGroups) {
>>             $groupDetail = Get-MgGroup -GroupId $group.Id
>>             $groupNames += $groupDetail.DisplayName
>>         }
>>
>>         Write-Host "• $($user.UserPrincipalName): $($groupNames -join ', ')" -ForegroundColor White
>>     }
>> }
• alice@labs243.onmicrosoft.com: HelpDesk
• bob@labs243.onmicrosoft.com: Sales
• charlie@labs243.onmicrosoft.com: HR
• diana@labs243.onmicrosoft.com: HR
• ethan@labs243.onmicrosoft.com: HelpDesk
• fatima@labs243.onmicrosoft.com: HelpDesk
• george@labs243.onmicrosoft.com: Sales
• hana@labs243.onmicrosoft.com: HR
• ivan@labs243.onmicrosoft.com: Sales
• julia@labs243.onmicrosoft.com: HR
```

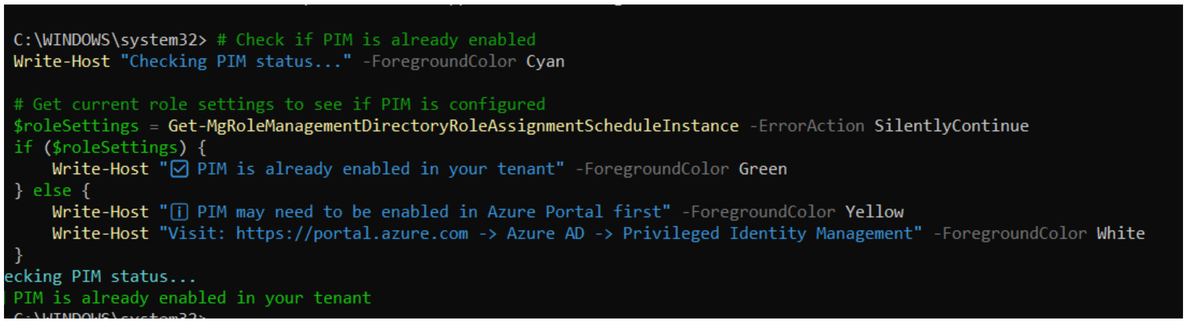
Part II. Configure privilege identity management (PIM)

We start with installing the necessary module for the implementation of PIM.



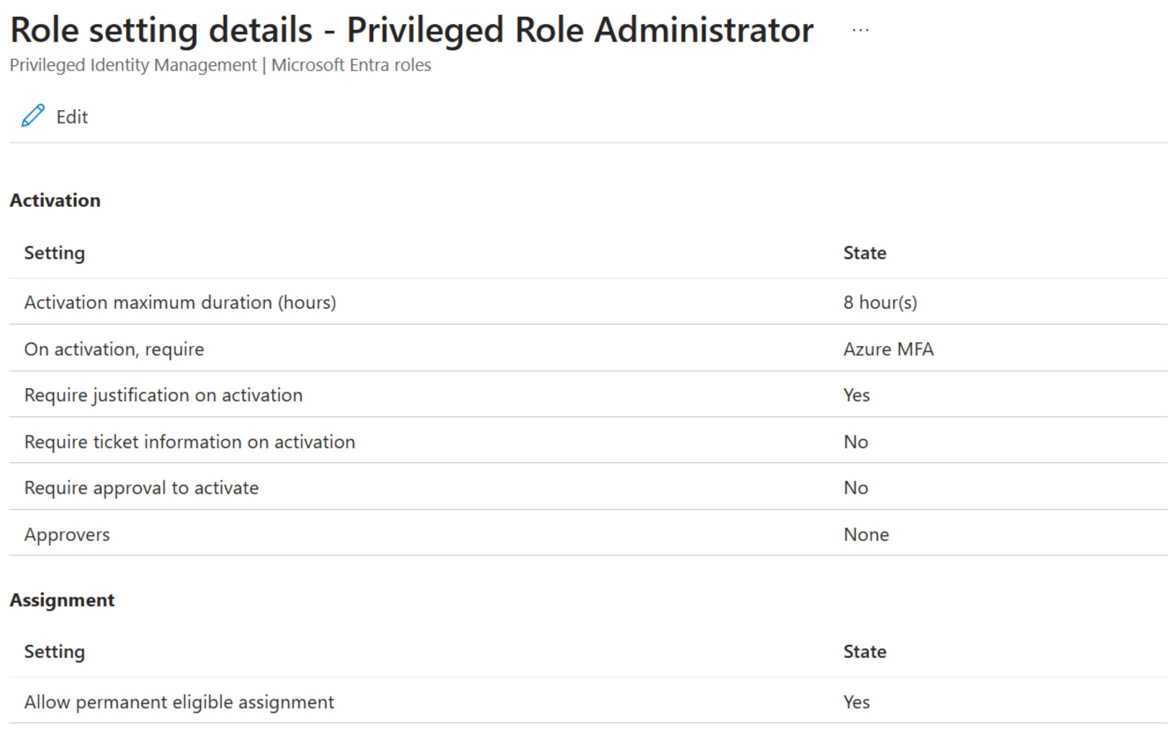
```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\WINDOWS\system32> Install-Module -Name Microsoft.Graph.Identity.Governance -Force -AllowClobber
PS C:\WINDOWS\system32>
PS C:\WINDOWS\system32> Connect-MgGraph -Scopes "RoleManagement.ReadWrite.Directory", "Directory.Read.All", "User.Read.All"
Welcome to Microsoft Graph!

Connected via delegated access using 14d82eec-204b-4c2f-b7e8-296a70dab67e
Readme: https://aka.ms/graph/sdk/powershell
SDK Docs: https://aka.ms/graph/sdk/powershell/docs
API Docs: https://aka.ms/graph/docs

NOTE: You can use the -NoWelcome parameter to suppress this message.

PS C:\WINDOWS\system32>
```

Then verify if PIM is enabled in the tenant.



```
 C:\WINDOWS\system32> # Check if PIM is already enabled
 Write-Host "Checking PIM status..." -ForegroundColor Cyan

 # Get current role settings to see if PIM is configured
 $roleSettings = Get-MgRoleManagementDirectoryRoleAssignmentScheduleInstance -ErrorAction SilentlyContinue
 if ($roleSettings) {
     Write-Host "☑ PIM is already enabled in your tenant" -ForegroundColor Green
 } else {
     Write-Host "ⓘ PIM may need to be enabled in Azure Portal first" -ForegroundColor Yellow
     Write-Host "Visit: https://portal.azure.com -> Azure AD -> Privileged Identity Management" -ForegroundColor White
 }
ecking PIM status...
 PIM is already enabled in your tenant
```

From the portal modified each aspect for each role



Role setting details - Privileged Role Administrator

Privileged Identity Management | Microsoft Entra roles

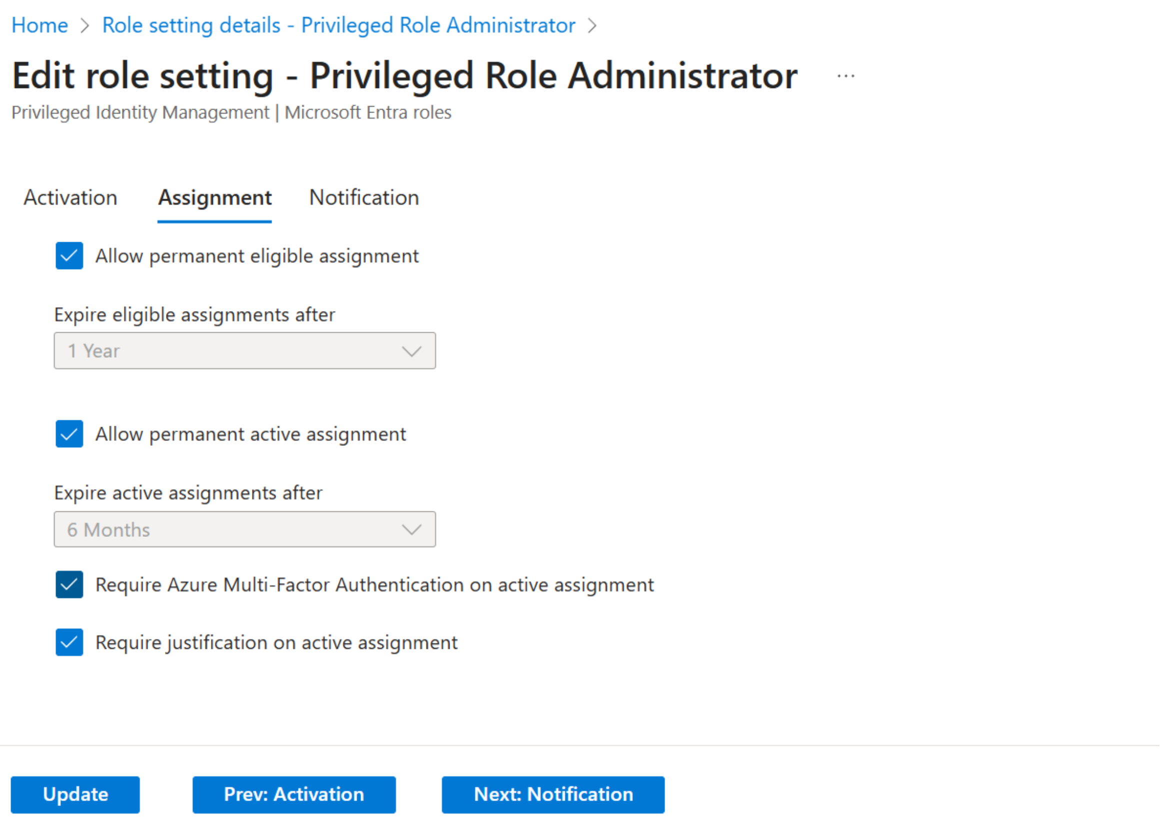
Edit

Activation

Setting	State
Activation maximum duration (hours)	8 hour(s)
On activation, require	Azure MFA
Require justification on activation	Yes
Require ticket information on activation	No
Require approval to activate	No
Approvers	None

Assignment

Setting	State
Allow permanent eligible assignment	Yes


Home > Role setting details - Privileged Role Administrator >

Edit role setting - Privileged Role Administrator ...

Privileged Identity Management | Microsoft Entra roles

✓ Allow permanent eligible assignment

Expire eligible assignments after: 1 Year

✓ Allow permanent active assignment

Expire active assignments after: 6 Months

✓ Require Azure Multi-Factor Authentication on active assignment

✓ Require justification on active assignment

Global Administrator (Most privileged role)

- **Activation**: Require MFA
- **Maximum activation duration**: 2 hours
- **Justification required**: Yes
- **Approval required**: Yes (recommended)
- **Eligible assignments**: Assign to your "Admin" group

Privileged Role Administrator (Manages PIM)

- **Activation**: Require MFA
- **Maximum activation duration**: 2 hours
- **Justification required**: Yes
- **Approval required**: Yes
- **Eligible assignments**: Assign to your "Admin" group

Helpdesk Administrator

- **Activation**: Require MFA
- **Maximum activation duration**: 8 hours
- **Justification required**: Yes
- **Approval required**: No
- **Eligible assignments**: Assign to your "HelpDesk" group

Authentication Administrator

- **Activation**: Require MFA
- **Maximum activation duration**: 4 hours
- **Justification required**: Yes
- **Approval required**: No
- **Eligible assignments**: Assign to your "HelpDesk" group

User Administrator

- **Activation**: Require MFA
- **Maximum activation duration**: 8 hours
- **Justification required**: Yes
- **Approval required**: No
- **Eligible assignments**: Assign to your "HR" group

Groups Administrator

- **Activation**: Require MFA
- **Maximum activation duration**: 8 hours
- **Justification required**: Yes
- **Approval required**: No
- **Eligible assignments**: Assign to your "HR" group