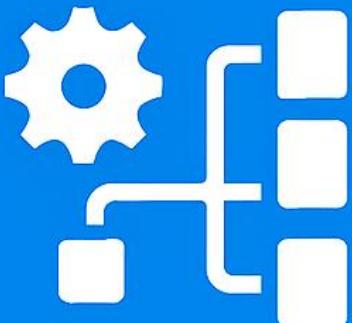
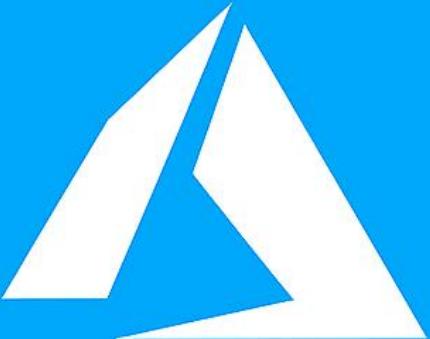


# Bulk Assign Licenses with Azure Automation



## Manual Lab:

# Mass Assignment of Licenses and Roles with Azure Automation

---

Author: Lucas Vercellini

Date: July 22, 2025

Version: 1.0

---

Description: This manual describes step-by-step how to configure and execute mass assignment of licenses and roles in Azure Active Directory using Azure Automation.

### Prerequisites:

- You must have Global Administrator or User Administrator permissions in Azure AD.
    - Automation Account created and managed identity configured.
    - PowerShell Az and AzureAD modules installed.
- 

### Glossary of Terms

Term	Definition
Azure Automation	Azure service that allows you to create, manage, and orchestrate automated processes (Runbooks) in the cloud.
Azure Active Directory	Cloud-based identity and access management service from Microsoft.
Automation Account	Container within Azure Automation where Runbooks, assets, and configurations are stored.
Managed Identity	Azure-managed identity that provides credentials to services for authentication without needing secrets.
Runbook	Script or workflow executed by Azure Automation to perform repetitive or scheduled tasks.
SKU (Stock Keeping Unit)	Unique identifier for a license plan in Azure (e.g., Office 365 E3 or Azure AD Premium P1).
CSV	Plain text file format (Comma Separated Values) used to import and export tabular data.
PowerShell Module	Package that contains cmdlets for managing Azure services via PowerShell (Az, AzureAD, etc.).
RBAC	Role-Based Access Control (RBAC) system in Azure used to manage permissions for resources.
Role Definition	Set of permissions grouped under a name (Reader, Contributor, Owner, etc.) in Azure RBAC.
Run As Connection	Connection method in Azure Automation that allows using credentials or identities for automatic authentication.

## 1. Prepare the environment

### 1. Ensure you have sufficient permissions in Azure AD

You must have either Global Administrator or User Administrator privileges in Azure AD, and Owner or User Access Administrator on the subscription.

### 2. Install the PowerShell modules locally (or in Cloud Shell):

Install-Module -Name Az -Scope CurrentUser -Force -Verbose

```
Windows PowerShell

version='2.1.0',destination='C:\Users\Lucas\AppData\Local\Temp\524668121'
VERBOSE: DownloadPackage' - name='Az.PowerBIEmbedded',
version='2.1.0',destination='C:\Users\Lucas\AppData\Local\Temp\524668121\Az.PowerBIEmbedded\Az.PowerBIEmbedded.nupkg',
uri='https://www.powershellgallery.com/api/v2/package/Az.PowerBIEmbedded/2.1.0'
VERBOSE: Downloading 'https://www.powershellgallery.com/api/v2/package/Az.PowerBIEmbedded/2.1.0'.
VERBOSE: Completed downloading 'https://www.powershellgallery.com/api/v2/package/Az.PowerBIEmbedded/2.1.0'.
VERBOSE: Completed downloading 'Az.PowerBIEmbedded'.
VERBOSE: Hash for package 'Az.PowerBIEmbedded' does not match hash provided from the server.
VERBOSE: InstallPackageLocal' - name='Az.PowerBIEmbedded',
version='2.1.0',destination='C:\Users\Lucas\AppData\Local\Temp\524668121'
VERBOSE: InstallPackage' - name='Az.PrivateDns',
version='1.2.0',destination='C:\Users\Lucas\AppData\Local\Temp\524668121'
VERBOSE: DownloadPackage' - name='Az.PrivateDns',
version='1.2.0',destination='C:\Users\Lucas\AppData\Local\Temp\524668121\Az.PrivateDns\Az.PrivateDns.nupkg',
uri='https://www.powershellgallery.com/api/v2/package/Az.PrivateDns/1.2.0'
VERBOSE: Downloading 'https://www.powershellgallery.com/api/v2/package/Az.PrivateDns/1.2.0'.
VERBOSE: Completed downloading 'https://www.powershellgallery.com/api/v2/package/Az.PrivateDns/1.2.0'.
VERBOSE: Completed downloading 'Az.PrivateDns'.
VERBOSE: Hash for package 'Az.PrivateDns' does not match hash provided from the server.
VERBOSE: InstallPackageLocal' - name='Az.PrivateDns',
version='1.2.0',destination='C:\Users\Lucas\AppData\Local\Temp\524668121'
VERBOSE: InstallPackage' - name='Az.RecoveryServices',
version='7.7.2',destination='C:\Users\Lucas\AppData\Local\Temp\524668121'
VERBOSE: DownloadPackage' - name='Az.RecoveryServices',
version='7.7.2',destination='C:\Users\Lucas\AppData\Local\Temp\524668121\Az.RecoveryServices\Az.RecoveryServices.nupkg',
uri='https://www.powershellgallery.com/api/v2/package/Az.RecoveryServices/7.7.2'
VERBOSE: Downloading 'https://www.powershellgallery.com/api/v2/package/Az.RecoveryServices/7.7.2'.
VERBOSE: Installing the dependency module 'Az.Billing' with version '2.2.0' for the module 'Az'.
VERBOSE: Module 'Az.Billing' was installed successfully to path
'C:\Users\Lucas\Documents\WindowsPowerShell\Modules\Az.Billing\2.2.0'.|
```

Install-Module -Name AzureAD -Scope CurrentUser -Force -Verbose

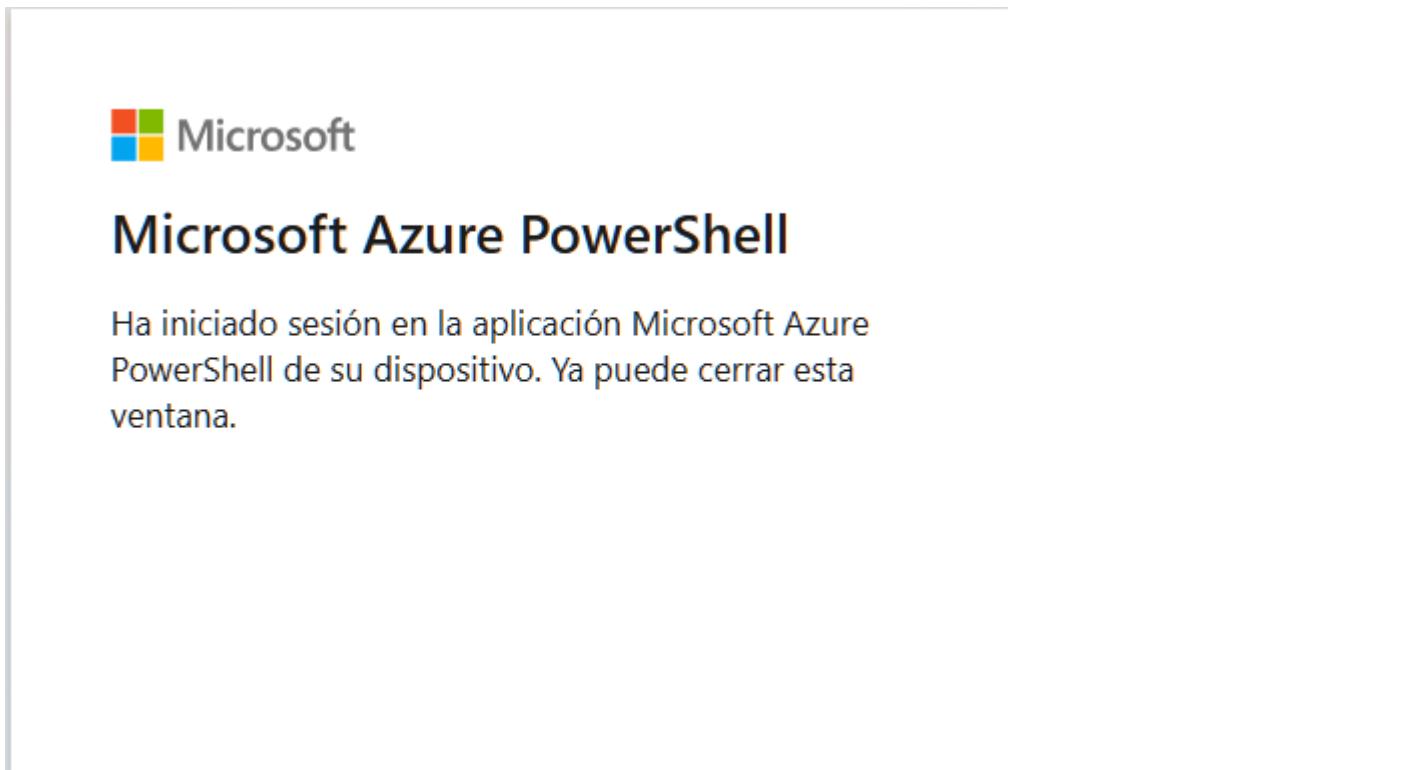
```
Windows PowerShell

PS C:\Users\Lucas> Install-Module -Name AzureAD -Scope CurrentUser -Force -Verbose
VERBOSE: Using the provider 'PowerShellGet' for searching packages.
VERBOSE: The -Repository parameter was not specified. PowerShellGet will use all of the registered repositories.
VERBOSE: Getting the provider object for the PackageManagement Provider 'NuGet'.
VERBOSE: The specified Location is 'https://www.powershellgallery.com/api/v2' and PackageManagementProvider is 'NuGet'.
VERBOSE: Searching repository 'https://www.powershellgallery.com/api/v2/FindPackagesById()?id='AzureAD'' for ''.
VERBOSE: Total package yield:'1' for the specified package 'AzureAD'.
VERBOSE: Performing the operation "Install-Module" on target "Version '2.0.2.182' of module 'AzureAD'".
VERBOSE: The installation scope is specified to be 'CurrentUser'.
VERBOSE: The specified module will be installed in 'C:\Users\Lucas\Documents\WindowsPowerShell\Modules'.
VERBOSE: The specified Location is 'NuGet' and PackageManagementProvider is 'NuGet'.
VERBOSE: Downloading module 'AzureAD' with version '2.0.2.182' from the repository
'https://www.powershellgallery.com/api/v2'.
VERBOSE: Searching repository 'https://www.powershellgallery.com/api/v2/FindPackagesById()?id='AzureAD'' for ''.
VERBOSE: InstallPackage' - name='AzureAD',
version='2.0.2.182',destination='C:\Users\Lucas\AppData\Local\Temp\770681284'
VERBOSE: DownloadPackage' - name='AzureAD',
version='2.0.2.182',destination='C:\Users\Lucas\AppData\Local\Temp\770681284\AzureAD\AzureAD.nupkg',
uri='https://www.powershellgallery.com/api/v2/package/AzureAD/2.0.2.182'
VERBOSE: Downloading 'https://www.powershellgallery.com/api/v2/package/AzureAD/2.0.2.182'.
VERBOSE: Completed downloading 'https://www.powershellgallery.com/api/v2/package/AzureAD/2.0.2.182'.
VERBOSE: Completed downloading 'AzureAD'.
VERBOSE: Hash for package 'AzureAD' does not match hash provided from the server.
VERBOSE: InstallPackageLocal' - name='AzureAD',
version='2.0.2.182',destination='C:\Users\Lucas\AppData\Local\Temp\770681284'
VERBOSE: Catalog file 'AzureAD.cat' is not found in the contents of the module 'AzureAD' being installed.
VERBOSE: Valid authenticode signature found in the file 'AzureAD.psdl' for the module 'AzureAD'.
VERBOSE: Module 'AzureAD' was installed successfully to path
'C:\Users\Lucas\Documents\WindowsPowerShell\Modules\AzureAD\2.0.2.182'.
PS C:\Users\Lucas> |
```

## 3. Verify and/or connect to Azure::

### 3.1. Connect-AzAccount

3.2. This will open a browser window asking you to authenticate your device using a code displayed in the PowerShell console. After that, you will proceed with logging in using your Azure account:  
[youremail@yourdomain.onmicrosoft.com](mailto:youremail@yourdomain.onmicrosoft.com)



```
PS C:\Users\Lucas> Connect-AzAccount -UseDeviceAuthentication
WARNING: You may need to login again after updating "EnableLoginByWam".
Please select the account you want to login with.

[Login to Azure] To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code LLT_L4ZTPU to authenticate.
Retrieving subscriptions for the selection...

[Announcements]
With the new Azure PowerShell login experience, you can select the subscription you want to use more easily. Learn more about it and its configuration at https://go.microsoft.com/fwlink/?linkid=2271909.

If you encounter any problem, please open an issue at: https://aka.ms/azpsissue

Subscription name      Tenant
-----      -----
Azure subscription 1 Default Directory
```

3.3. To confirm that your account is properly connected, you can run:

```
Get-AzContext
```

```
PS C:\Users\Lucas> Get-AzContext

Tenant: Default Directory (12b221ec-f173-4f74-b12c-dc93775ae2d4)

SubscriptionName      SubscriptionId          Account                                Environment
-----      -----          -----
Azure subscription 1 46a83ca6-544a-40a1-9c67-5058b6264dfd lucas@soporeksircom.onmicrosoft.com AzureCloud
```

⚠ Note: Even if the account used is a Global Administrator, it may not have explicit permissions on the subscription.

Make sure the user you're connecting with has proper access (in this lab, I assigned "Owner" directly for simplicity, but in production environments, you should always restrict permissions based on the principle of least privilege).

3.4. If you don't already have an Automation Account, create one in your desired resource group:

```
# Variables
$rgName  = "rg-LicenciasLab"
$location = "eastus"
$aaName   = "aa-LicenciasLab"
```

```
# Create RG y AA
```

```
New-AzResourceGroup -Name $rgName -Location $location
```

```
ResourceGroupName : rg-LicenciasLab
Location         : eastus
ProvisioningState : Succeeded
Tags             :
ResourceId       : /subscriptions/46a83ca6-544a-40a1-9c67-5058b6264dfd/resourceGroups/rg-LicenciasLab
```

```
New-AzAutomationAccount -Name $aaName -ResourceGroupName $rgName -Location $location
```

```
SubscriptionId      : 46a83ca6-544a-40a1-9c67-5058b6264dfd
ResourceGroupName   : rg-LicenciasLab
AutomationAccountName : aa-LicenciasLab
Location           : eastus
State              : Ok
Plan               : Basic
CreationTime       : 21/7/2025 16:32:00 +02:00
LastModifiedTime   : 21/7/2025 16:32:00 +02:00
LastModifiedBy     :
Tags              : {}
Identity          :
Encryption        : Microsoft.Azure.Management.Automation.Models.EncryptionProperties
PublicNetworkAccess :
```

#### 4. Upload the CSV with users and configure modules: Upload the CSV file to an Azure Storage account

##### 4.1. Create the Storage Account (one-time setup)

**⚠ I had issues executing this step because PowerShell didn't recognize my Azure free subscription as valid. To solve it, I forced the registration of the Storage and Azure Automation providers using PowerShell:**

```
Register-AzResourceProvider -ProviderNamespace "Microsoft.Storage"
ProviderNamespace : Microsoft.Storage
RegistrationState : Registering
ResourceTypes    :{locations/ActionsRPOperationStatuses, storageAccounts/reports,
                  storageAccounts/storageTaskAssignments, storageAccounts/storageTaskAssignments/reports...}
Locations       :{East US 2, Japan East, Japan West, South India...}
```

**✓ To verify that they are properly registered: Get-AzResourceProvider -ProviderNamespace "Microsoft.Storage"**

```
# Create the resource group in a region allowed by Free Trial:
```

```
New-AzResourceGroup -Name "rg-LicenciasEastUS" -Location "eastus"
```

```
# Generate a unique name for the Storage Account:
```

```
$storageName = "storlicencias$(Get-Random -Maximum 9999)"
```

```
# Create the Storage Account in a valid region:
```

```
New-AzStorageAccount -ResourceGroupName "rg-LicenciasEastUS" ` 
-Name $storageName ` 
-Location "eastus" ` 
-SkuName Standard_LRS ` 
-Kind StorageV2
```

```
PS C:\Users\Lucas> New-AzStorageAccount -ResourceGroupName "rg-LicenciasEastUS" ` 
>> -Name $storageName ` 
>> -Location "eastus" ` 
>> -SkuName Standard_LRS ` 
>> -Kind StorageV2
```

StorageAccountName	ResourceGroupName	PrimaryLocation	SkuName	Kind	AccessTier	CreationTime	ProvisioningState
storlicencias9832	rg-LicenciasEastUS	eastus	Standard_LRS	StorageV2	Hot	21/7/2025 14:59:06	Succeeded

5. Create the container and upload the CSV file: Let's assume you used "rg-LicenciasEastUS" and the Storage Account name is stored in the variable \$storageName

5.1. To create a container in your Storage Account, run:

```
$ctx = (Get-AzStorageAccount -ResourceGroupName "rg-LicenciasEastUS" -Name $storageName).Context  
New-AzStorageContainer -Name "csv" -Context $ctx -Permission Off  
  
PS C:\Users\Lucas> $ctx = (Get-AzStorageAccount -ResourceGroupName "rg-LicenciasEastUS" -Name $storageName).Context  
PS C:\Users\Lucas> New-AzStorageContainer -Name "csv" -Context $ctx -Permission Off
```

Storage Account Name: storlicencias9832					
Name	PublicAccess	LastModified	IsDeleted	VersionId	
csv	Off	21/7/2025 15:25:22 +00:00			

Upload your local CSV file (⚠ Make sure you have the file available locally, for example:

```
C:\temp\usuarios.csv)  
-File "C:\temp\usuarios.csv" `  
-Blob "usuarios.csv" `  
-Context $ctx
```

This uploads the file usuarios.csv into the "csv" container.

```
PS C:\Users\Lucas> Set-AzStorageBlobContent -Container "csv" `  
-> -File "C:\temp\usuarios.csv" `  
-> -Blob "usuarios.csv" `  
-> -Context $ctx  
  
AccountName: storlicencias9832, ContainerName: csv  
  
Name BlobType Length ContentType LastModified AccessTier SnapshotT  
ime  
--- ---- -- ----- ----- -----  
usuarios.csv BlockBlob 9012 application/octet-stream 2025-07-21 15:30:20Z Hot
```

5.2. Get the connection string (to be used later in the Runbook)

```
$storageKey = (Get-AzStorageAccountKey -ResourceGroupName "rg-LicenciasEastUS" -Name $storageName)[0].Value  
$connectionString =  
"DefaultEndpointsProtocol=https;AccountName=$storageName;AccountKey=$storageKey;EndpointSuffix=core.windows.net"
```

5.3. Save the connection string as a variable in Azure Automation

Go to the Azure portal: <https://portal.azure.com>

Navigate to your Automation Account > Shared Resources > Variables

Click on + Add a variable and fill in the following:

- **Name:** CsvStorageConnection
- **Type:** String
- **Value:** (paste the \$connectionString)
- **Encrypted:** No (optional)

Home > aa-LicenciasLab

## Variables

Add a variable Refresh

Search variables...

Name	Type	Value	Last modified
CsvStorageConnection	String	DefaultEndpointsProtocol=https;AccountName=\$storageName;AccountKey=\$sto...	21/7/2025, 5:33 p. m.

Schedules

Modules

Python packages

Credentials

Connections

Certificates

Variables

Related Resources

Account Settings

## 6. Create the Simulation Runbook (Optional – AsignarLicencias.ps1)

This step allows you to simulate the license assignment to ensure that the CSV reading and authentication flow works correctly before applying any real changes to user accounts.

- a. Go to the Azure portal > Azure Automation > Tu cuenta (por ejemplo, aa-LicenciasLab) > Runbooks > + Create a runbook

Navigate to: Azure Portal > Azure Automation > Your account (e.g., [AUTOMATION\_ACCOUNT\_NAME]) > Runbooks > Click + Create a runbook

- b. Fill in the required fields:

Name: SimularAsignacionLicencias

## Runbook type: PowerShell

Runtime version: 5.1

- c. Click on Review + Create, then click Create

- d. Enter your simulation code

This code can be a simple version that just reads the CSV file and logs its content to the output console.

Home > aa-LicenciasLab | Runbooks > SimularAsignacionLicencias (aa-licenciaslab/SimularAsignacionLicencias) >

## Edit PowerShell Runbook

SimularAsignacionLicencias

Save Publish Revert to published Test pane Edit in VS Code Feedback

> CMDLETS

> RUNBOOKS

> ASSETS

```
7     [Parameter(Mandatory=$true)]
8     [string] $CsvBlobName,
9
10    [Parameter(Mandatory=$false)]
11    [string] $Skuid = "EMULATED-SKU-1234"
12
13 )
14
15 # Conectarse a Azure y AzureAD
16 Connect-AzAccount -Identity
17 Connect-AzureAD -AzureEnvironmentName AzureCloud
18
19 # Conectar al Storage
20 $ctx = New-AzStorageContext -ConnectionString $StorageConnectionString
21
22 # Descargar CSV
23 $tempPath = Join-Path $env:TEMP "usuarios.csv"
24 Get-AzStorageBlobContent -Container $containerName -Blob $CsvBlobName -Destination $tempPath -Context $ctx -Force
25 $usuarios = Import-Csv -Path $tempPath
26
27 # Simular asignación
28 foreach ($u in $usuarios) {
29     $upn = $u.UserPrincipalName
30
31     try {
32         $usuario = Get-AzureADUser -ObjectId $upn
33         Write-Output " [Simulación] Asignar licencia '$Skuid' al usuario: $upn"
34     }
35 }
```

- #### e. Save, publish, and test the Runbook

**⚠️** Make sure everything runs correctly before proceeding to real license assignment.

Home &gt; aa-LicenciasLab

## aa-LicenciasLab | Runbooks

Automation Account

- Search
  - + Create a runbook
  - Import a runbook
  - Browse gallery
  - Learn more
  - Feedback
  - Refresh
- 
- Overview
  - Activity log
  - Access control (IAM)
  - Tags
  - Diagnose and solve problems
  - Resource visualizer
  - Process Automation
  - Runbooks**
  - Jobs
  - Hybrid worker groups
  - Configuration Management
    - State configuration (DSC)
  - Shared Resources
    - Schedules
    - Modules
    - Python packages
    - Credentials
    - Connections

**⚠️** PowerShell 7.1 and Python 2.7 are no longer supported by their respective parent products PowerShell and Python. It is strongly recommended to update outdated runbooks to latest supported versions using Runtime environment. [Learn more](#)

Search runbooks...

Runbook type : All

Authoring Status : All

Showing 1 to 1 of 1 records.

Name	Authoring status	Runbook type	Runtime version	Last modified	Tags
SimularAsignacionLicencias	New	PowerShell	5.1	21/7/2025, 6:07 p. m.	

## 7. Create Users from CSV (if they don't already exist)

a. Ensure that the target groups already exist: Before running the script, make sure the destination groups (e.g., "Tech Support", "Marketing") are already created in Azure Active Directory.

b. Run the Runbook (or local script) to create users from the CSV file (The script should read the CSV and create each user, assigning them to the corresponding groups.)

### **⚠️ Future improvement suggestion:**

If a user already exists, check if they belong to all the groups listed in the CSV. If not, add them automatically.

## 8. Simulated License Assignment (or real logic if valid licenses are available)

a. If you want to avoid errors due to missing SKUs or invalid licenses you can simulate the process by reading the CSV file, identifying each user, and displaying a message showing which license would be assigned.

b. If you do have valid licenses available in the tenant you can use the AzureAD module or Microsoft Graph to assign them. Keep in mind that real SKUs require specific internal names (e.g., ENTERPRISEPACK for Office 365 E3).

c. Run the Runbook containing the license assignment logic: Monitor the output to validate its execution.

d. Confirm that licenses were applied in Azure AD (if you used real logic): You can verify in the Azure portal or with PowerShell that each user has the intended license applied.

aa-LicenciasLab | Identity

Automation Account

System assigned User assigned

A system assigned managed identity is restricted to one per resource and is tied to the lifecycle of this resource. You can grant permissions to the managed identity by using Azure role-based access control (Azure RBAC). The managed identity is authenticated with Microsoft Entra ID, so you don't have to store any credentials in code.

Status: On

Object (principal) ID: be1e1cb6-9cb4-46d5-b845-e9189d0b6879

Permissions: Azure role assignments

This resource is registered with Microsoft Entra ID. The managed identity can be configured to allow access to other resources. Be careful when making changes to the access settings for the managed identity because it can result in failures. [Learn more](#)

Microsoft Azure

User Administrator | Assignments

Add assignments

Try changing or adding filters if you don't see what you're looking for.

Search: aa-LicenciasLab

1 result found

All Users Enterprise applications

Name	Type	Details
aa-LicenciasLab	Enterprise application	ff811e85-a97f-40f7-9508-081f992410d1

Add

## 9. Runbook Execution and Final Validations

Once the file `usuarios.csv` is successfully uploaded to the csv container, and the Automation variable named `CsvStorageConnection` contains the full connection string, you can run the Runbook.

- Example script used inside the Runbook:

```
# Authenticate using the Managed Identity
Connect-AzAccount -Identity

# Retrieve the connection string from the Automation variable
$connectionString = Get-AutomationVariable -Name "CsvStorageConnection"

# Create Azure Storage context
$ctx = New-AzStorageContext -ConnectionString $connectionString

# Define local temp path and blob name
$csvLocalPath = "$env:TEMP\usuarios.csv"
$blobName = "usuarios.csv"
$containerName = "csv"

# Download the CSV file from Blob Storage
Get-AzStorageBlobContent -Container $containerName ` 
    -Blob $blobName ` 
    -Destination $csvLocalPath ` 
    -Context $ctx ` 
    -Force

# Check if the file was downloaded correctly
if (!(Test-Path $csvLocalPath)) {
    throw "✖ Could not download usuarios.csv"
}

# Load CSV data
$usuarios = Import-Csv -Path $csvLocalPath

foreach ($usuario in $usuarios) {
    Write-Output "Processing user: $($usuario.DisplayName) <$($usuario.UserPrincipalName)>"

    # Search for the user in Azure AD
    $user = Get-AzADUser -UserPrincipalName $usuario.UserPrincipalName
    if ($user) {
        # Simulate license assignment
        Write-Output "✓ User found. Would assign license to: $($user.UserPrincipalName)"
    } else {
        Write-Output "✖ User not found: $($usuario.UserPrincipalName)"
    }
}
```

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

SimularAsignacionLicencias 21/7 | storlicencias9832 - Microsoft Azure | Hero Universe

Bancos Casa ia Gmail DevOps Calendar Total Battle G Drive FreeBitco.in Wof LinkedIn YouTube Hora Mundial o GMT Internet y vs Clientes Italia Mercado Libre Plataforma E-learnin...

Microsoft Azure Upgrade Search resources, services, and docs (G+) Copilot

soporte@ksir.com.ar DEFAULT DIRECTORY (SOPORTEK...)

Home > aa-LicenciasLab | Runbooks > SimularAsignacionLicencias (aa-licenciaslab/SimularAsignacionLicencias) >

SimularAsignacionLicencias 21/7/2025, 7:22 p. m. Job

Resume Stop Suspend Feedback Refresh Export output

```
Environments
-----
{[AzureChinaCloud, AzureCloud], [AzureCloud, AzureCloud], [AzureUSGovernment, AzureUSGovernment]} Microsoft.Azure...
```

```
ICloudBlob : Microsoft.Azure.Storage.Blob.CloudBlockBlob
BlobType : BlockBlob
Length : 9012
IsDeleted : False
BlobClient : Azure.Storage.Blobs.BlobClient
BlobBaseClient : Azure.Storage.Blobs.Specialized.BlockBlobClient
BlobProperties : Azure.Storage.Blobs.Models.BlobProperties
RemainingDaysBeforePermanentDelete :
ContentType : application/octet-stream
LastModified : 7/21/2025 3:30:20 PM +00:00
SnapshotTime :
ContinuationToken :
VersionId :
IsLatestVersion :
AccessTier : Hot
TagCount : 0
Tags :
ListBlobProperties :
Context : Microsoft.WindowsAzure.Commands.Storage.AzureStorageContext
Name : usuarios.csv

Procesando usuario: Usuario001 Demo <usuario001@soportecksirc.com.onmicrosoft.com>
    ✓ Usuario encontrado. Se asignaría licencia a: usuario001@soportecksirc.com.onmicrosoft.com

Procesando usuario: Usuario002 Demo <usuario002@soportecksirc.com.onmicrosoft.com>
```

The execution clearly shows that the file was downloaded, interpreted, and that users were found in Azure AD. The simulation simply prints a message for each user to indicate the intended action.

**⚠ Note:** This procedure is designed for lab environments. For production use, make sure to:

- Use real license assignment via Set-AzureADUserLicense or Microsoft Graph
- Confirm that sufficient licenses are available
- Check if a user already has a license before assigning a new one
- Avoid duplicates in the CSV file
- Use Managed Identity with the minimum required permissions