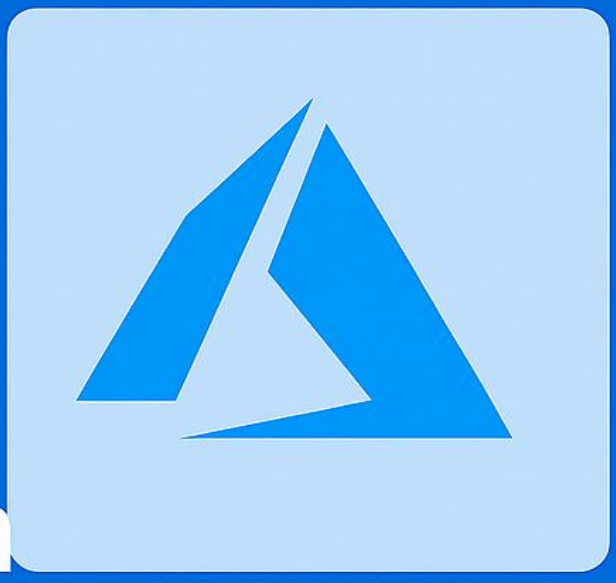
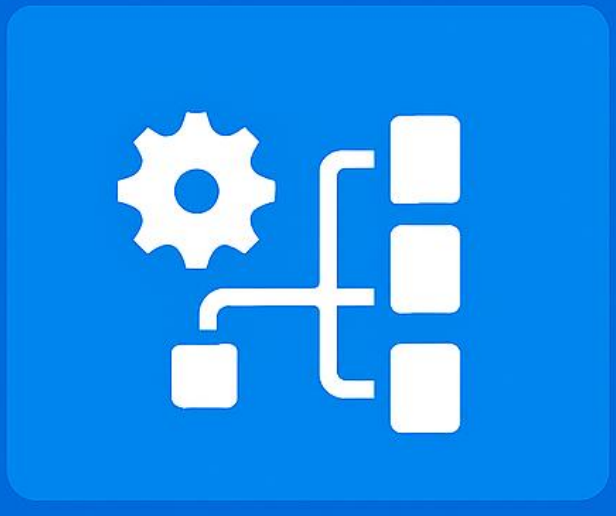
Bulk Assign Licenses with Azure Automation





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.

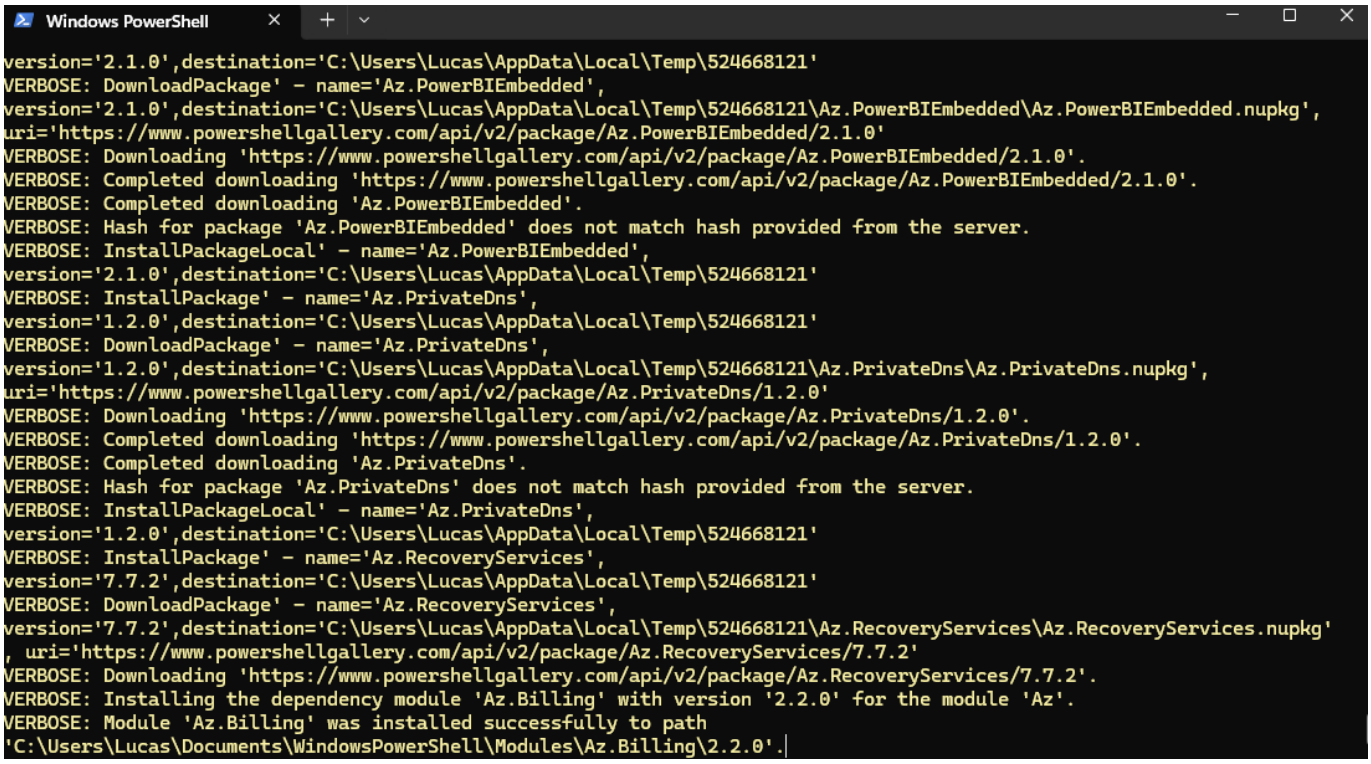
Glossary of Terms

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.

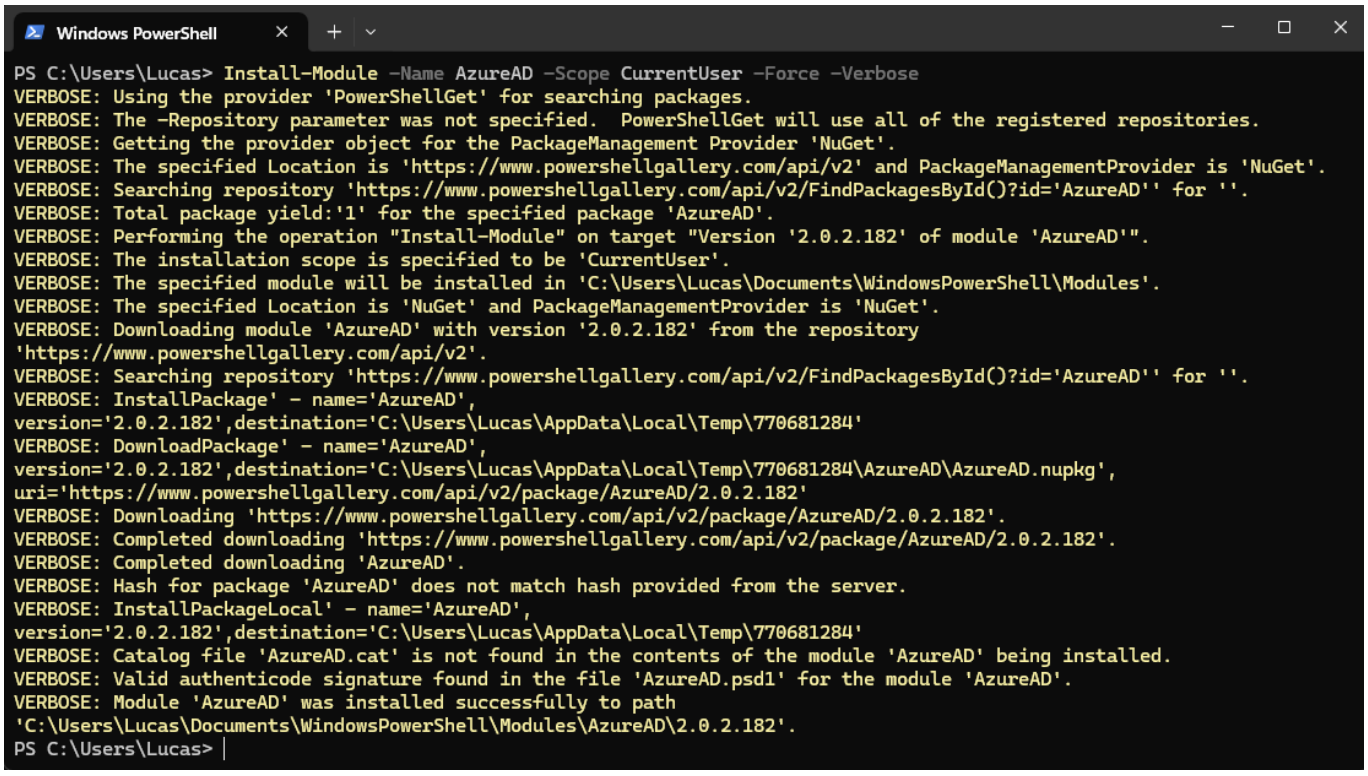
1. Prepare the environment

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

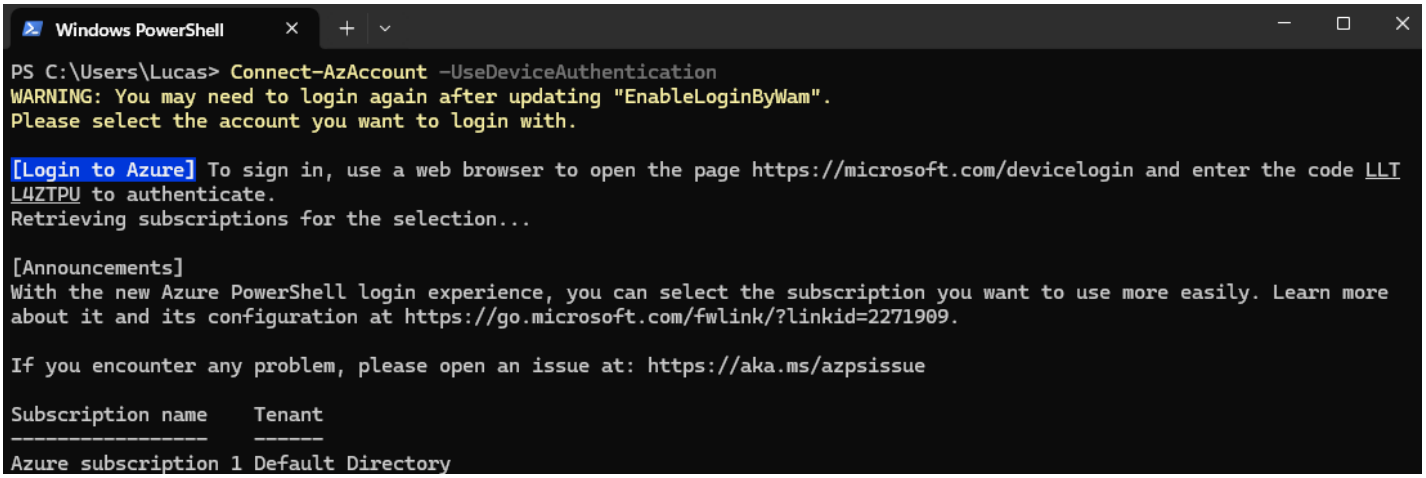


Install-Module -Name AzureAD -Scope CurrentUser -Force -Verbose



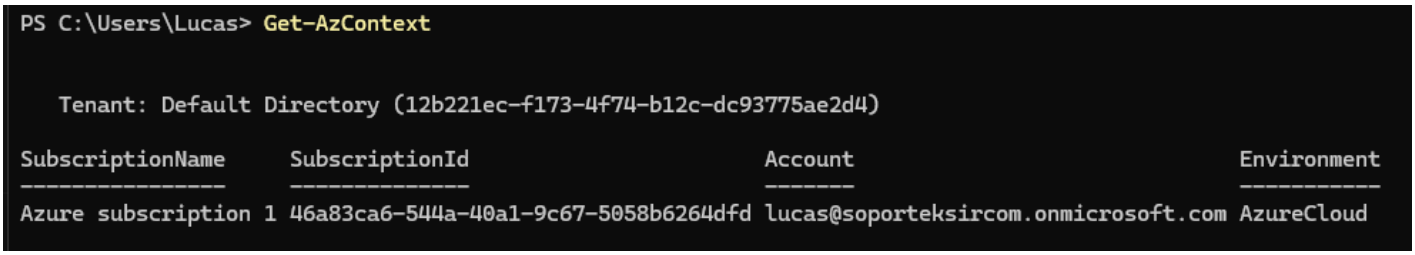
3. Verify and/or connect to Azure::
   3.1. Connect-AzAccount

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



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

Get-AzContext



⚠️ 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       Provisioning
                                                                                                          State
------------------ -----------------  --------------- -------      ----      ---------- ------------       ------------
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 SnapshotTime
----                 --------  ------          -----------                ------------         ---------- ------------
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"

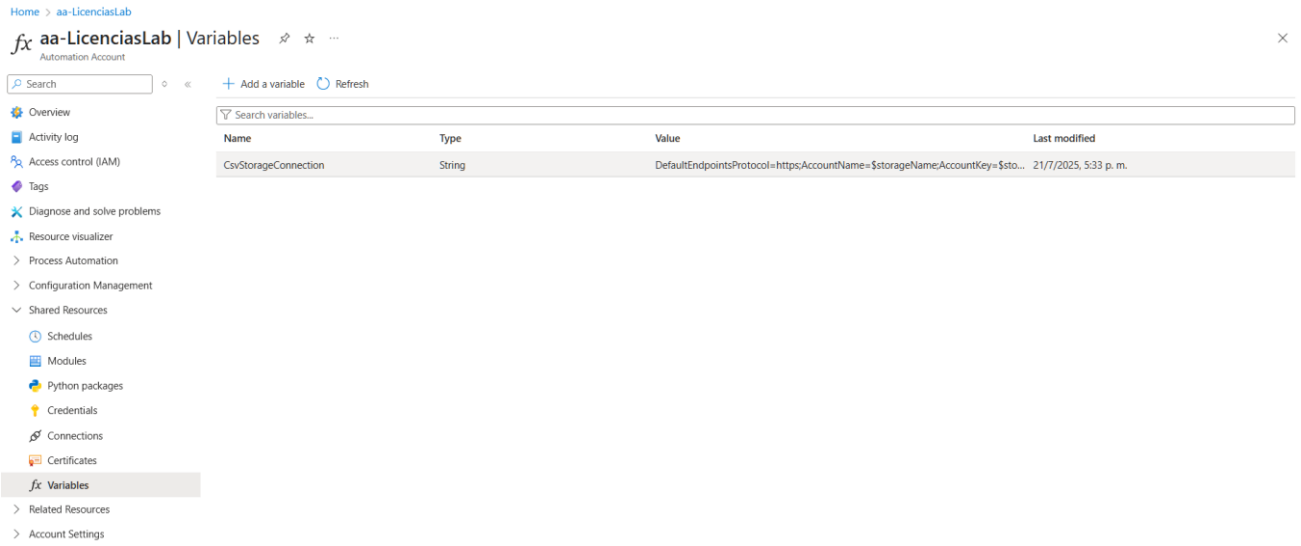
5.3. Save the connection string as a variable in Azure Automation

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)



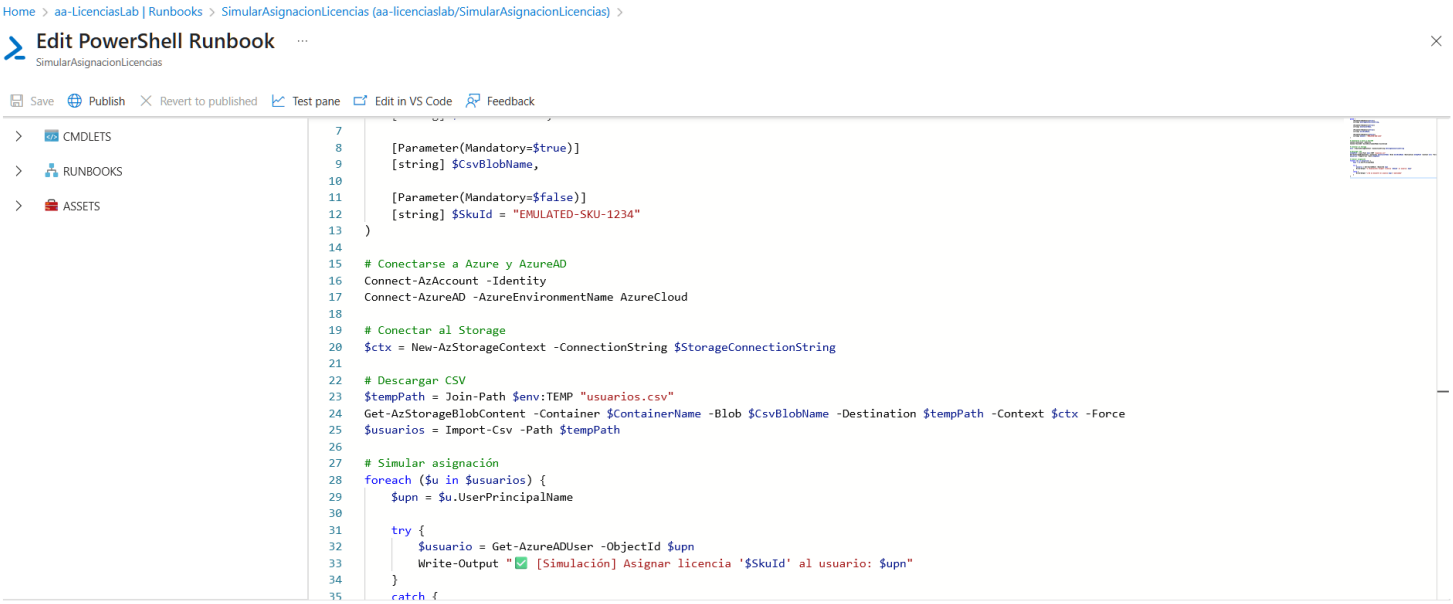
6. Create the Simulation Runbook (Optional – AsignarLicencias.ps1)

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

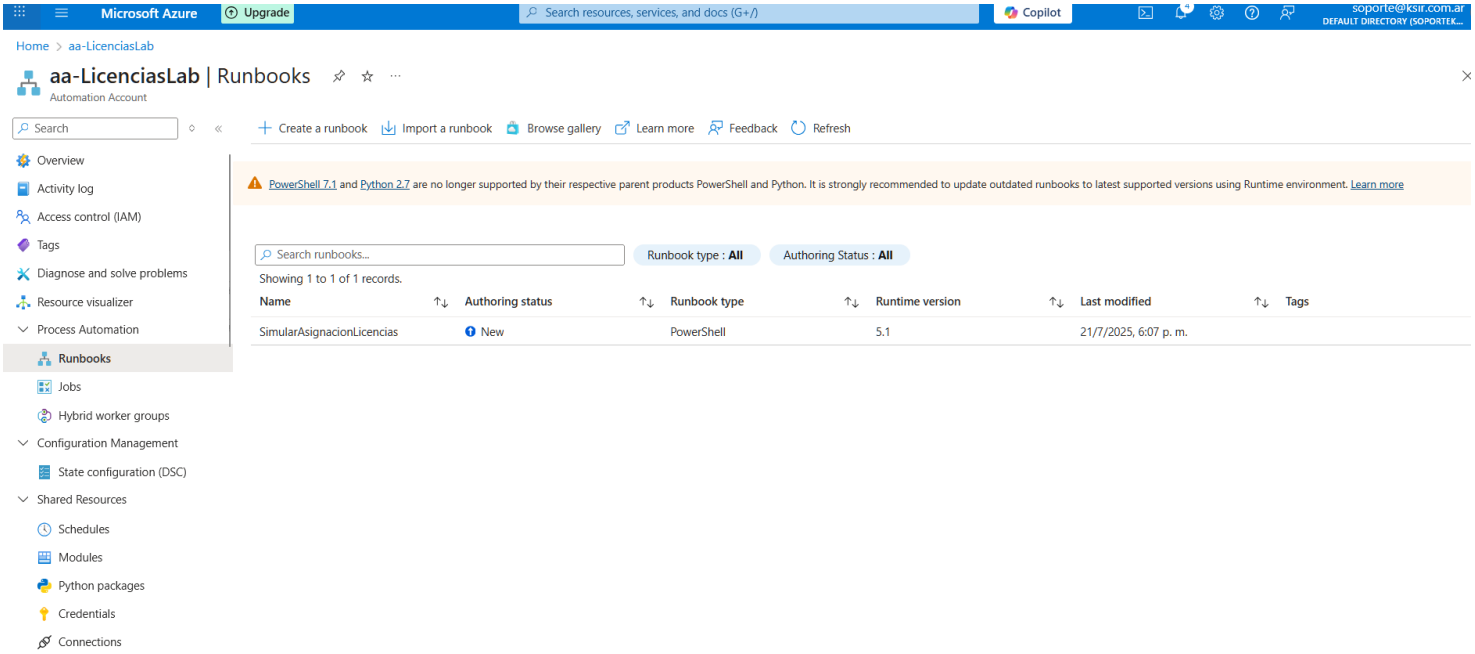
b. Fill in the required fields:
Name: SimularAsignacionLicencias
Runbook type: PowerShell
Runtime version: 5.1
c. Click on Review + Create, then click Create
d. Enter your simulation code

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



e. Save, publish, and test the Runbook

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



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.

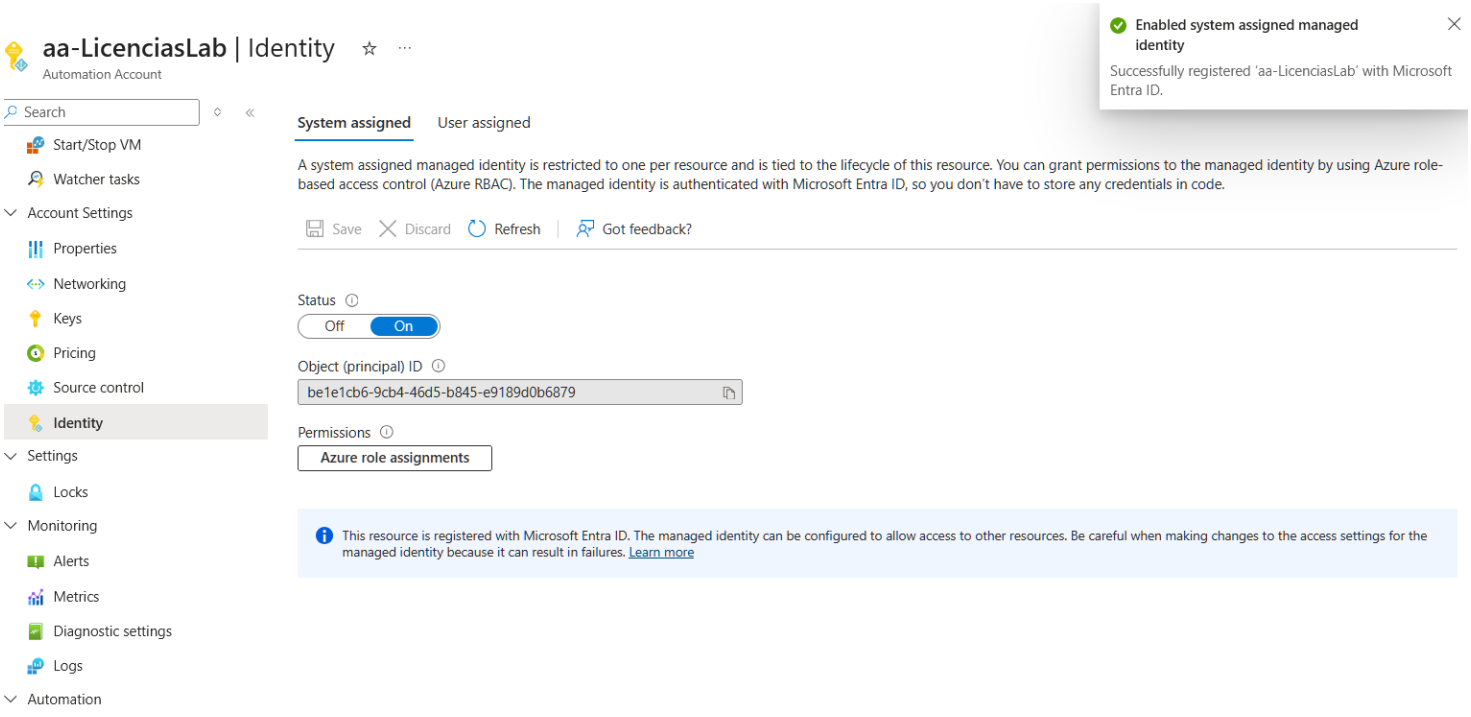
8. Simulated License Assignment (or real logic if valid licenses are available)

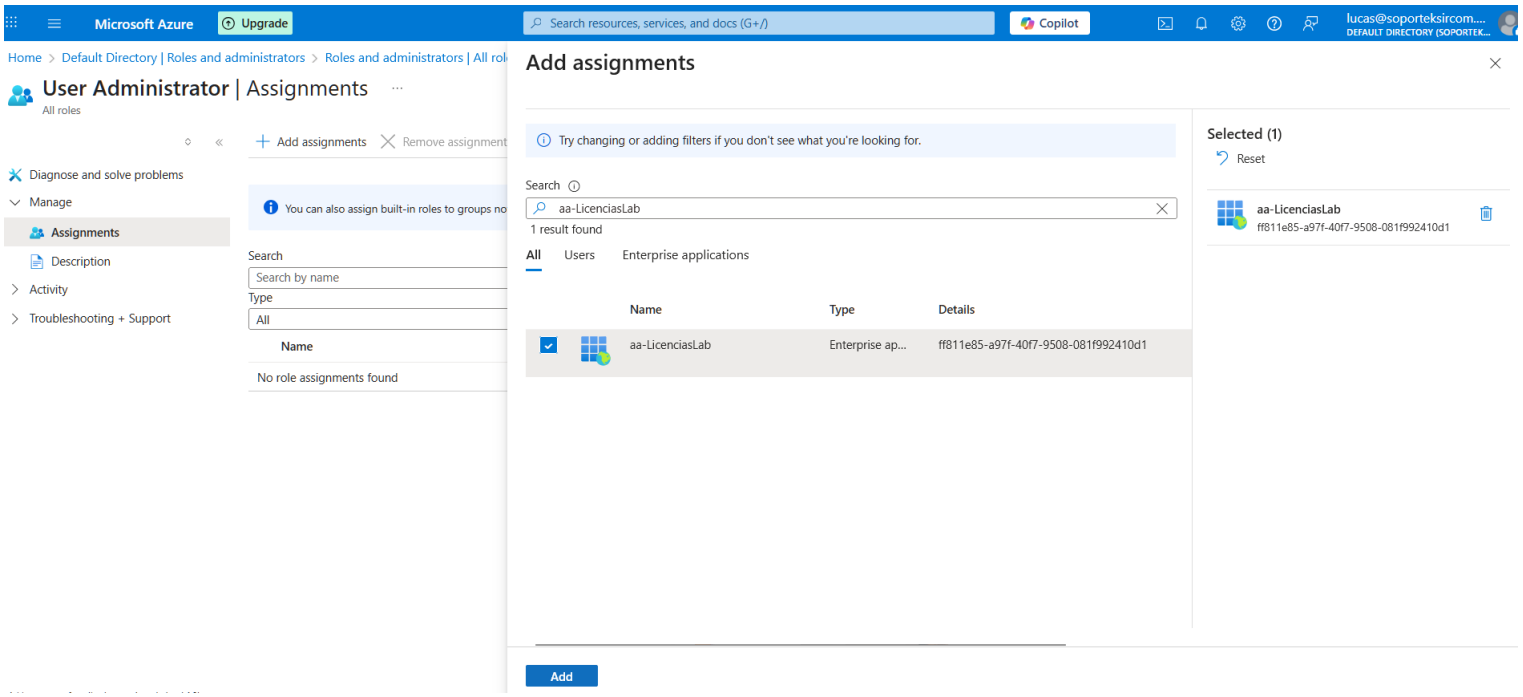
a. If you want to avoid errors due to missing SKUs or invalid licenses you can simulate the process by reading the CSV file, identifying each user, and displaying a message showing which license would be assigned.

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.



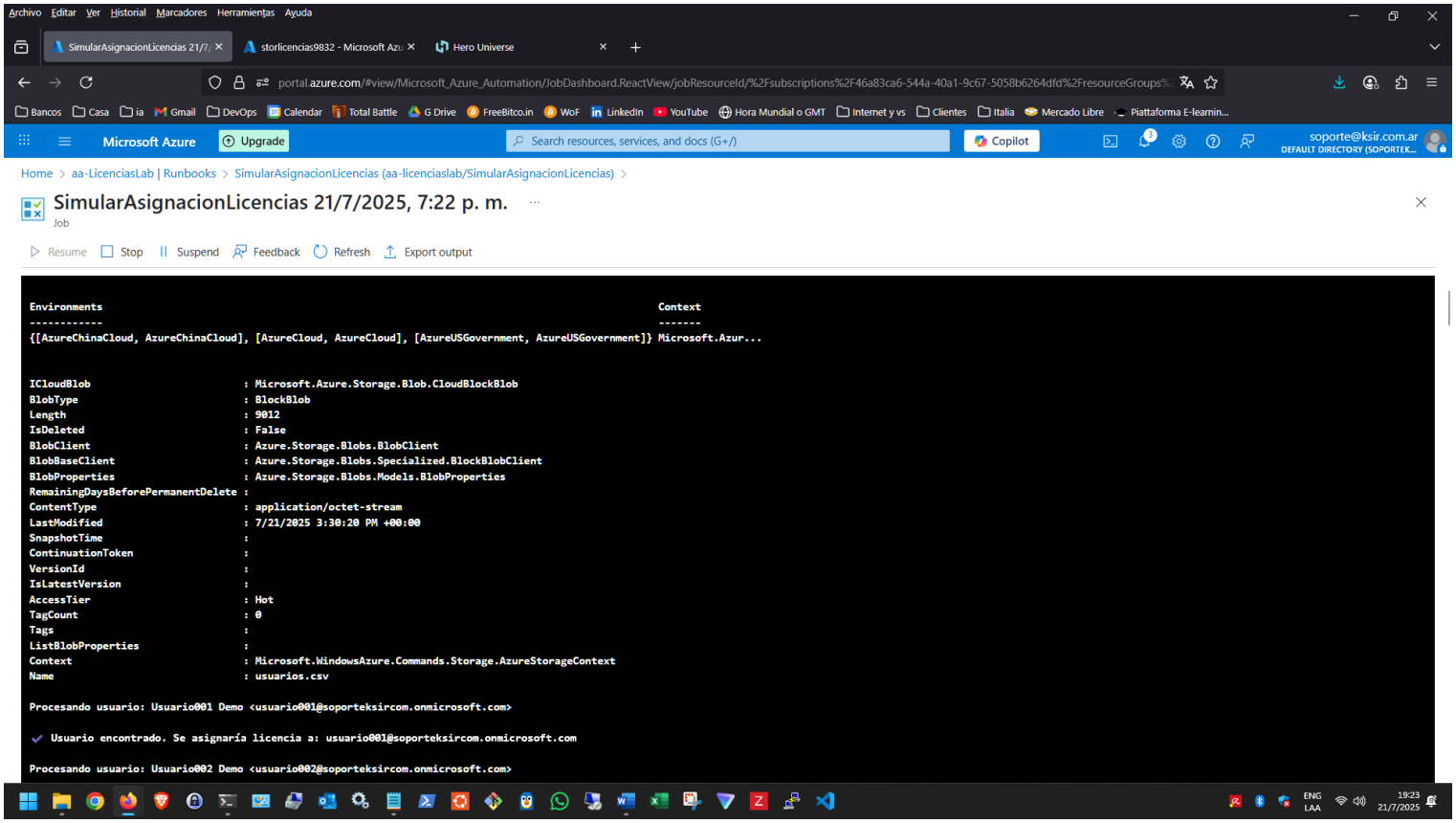


📌 9. Runbook Execution and Final Validations

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)"
  }
}
```



📸 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