Windows Server Administration with PowerShell

Active Directory,

Share and NTFS Permissions,

Files and Folders Access Auditing

By Ershad RAMEZANI

Table of Contents

Preface

In this article, we will explore how PowerShell can be used for managing and automating tasks related to Active Directory, share and NTFS permissions, and files and folders access auditing. Through the course of a sample project, we will walk through the various steps involved in setting up and configuring these systems, and demonstrate how PowerShell can be used to automate and streamline the process. Whether a beginner or an experienced administrator, this article will provide valuable insights and practical examples for using PowerShell to manage and maintain Windows systems and servers. We will cover topics such as creating and managing user and group accounts, configuring access controls, and monitoring and auditing file and folder activity. By the end of this article, we will have a better understanding of how PowerShell can be used to improve efficiency and productivity in our daily work.

Introduction to the project

Adrarform is an IT training center that has been in operation for five years. Recently, the administrators of the center have encountered several issues related to the management of the document sharing system. These issues include the need to create and delete a large number of user accounts each month, the presence of old records that have not been deleted, and the accessibility of sensitive files to unauthorized individuals.

To address these issues, Adrarform is looking to restructure and secure their document sharing system. Additionally, they want to improve the administration of user accounts and set up tools for auditing access to sensitive files. The training center includes several classes, such as BTS SIO, TSSR, AIS, DEV1, and DEV2, as well as teachers and an administrative team.

Administrative team share permissions

The members of the administrative team have a space which is only accessible by them in both writing and reading. Each member has their own personal space, and there is a common space for the team that everyone can read and write to, but they cannot delete files that they do not own.

Students share permissions

Each class at the training center also has its own spaces for students and teachers. The "students" space is where students can store documents that they want to share (common space) or return (private space), while the common space for teachers is where they can make course materials and other documents available to students and colleagues.

- Students are only able to view documents in the teacher area of their class, but they are not able to delete or save any documents there.
- Teachers, on the other hand, are able to save files in the spaces reserved for students, but they are not able to modify or delete the original files in any way.
- Additionally, teachers have access to the Student common space of the class, where they can save new documents or create copies of student documents (save as). However, they cannot delete any student documents.

Teachers resources share permissions

Teachers from the same discipline (AIS, DEV, TSSR, etc.) share a common folder that is dedicated to sharing resources. This folder is named after the discipline. Each teacher has the right to modify or delete their own documents, but they do not have the ability to modify or delete documents that belong to other teachers. This system is designed to ensure that each teacher has access to the resources they need, while also protecting the integrity of the shared folder.

Tasks to be Accomplished

As an IT professional, your job is to provide solutions to the issues facing Adrarform. This includes proposing a solution to the director of the center, setting up a model to address the problems, and validating each aspect of the solution with relevant examples. Additionally, you will be responsible for writing technical documentation for each solution implemented. This documentation should include all necessary elements, such as screenshots and scripts, to facilitate maintainability and upgrades.

Ensuring the security of shared folders

- Offer a folder hierarchy that addresses the requirements of the organization.
- Provide user groups for students, teachers and administrative team.
- Implement security at the file system level (Security tab) and the Sharing tab.
- Establish a method for automatically mounting network drives.
- Demonstrate through significant testing that the security requirements in the specifications are met.

Automating administration of accounts and folders

In order to make it easier to manage accounts and folders, the director gave you an excel file and asked you to come up with a system for automating certain tasks. This will help streamline the administration of these accounts and folders:

Last name	First name	Cohort	Start Date	End Date	Discipline	User Type
LEE	Axel	AIS-1910	10/01/2019	09/30/2020	-	Student
SANTOS	Romain	AIS-1910	10/01/2019	09/30/2020	-	Student
JACQY	Charly	AIS-1910	10/01/2019	09/30/2020	-	Student
DOFUS	David	AIS-1910	10/01/2019	09/30/2020	-	Student
TRANKIL	Lien	AIS-1910	10/01/2019	09/30/2020	-	Student
GRAND DUC	Sebastien	AIS-1910	10/01/2019	09/30/2020	-	Student
BERGAMOTH	Joel	AIS-1910	10/01/2019	09/30/2020	-	Student
NOUMA	Quentin	AIS-1910	10/01/2019	09/30/2020	-	Student
THIONG	Yan	AIS-1910	10/01/2019	09/30/2020	-	Student
BIENVENU	Pierre	AIS-1910	06/01/2020	05/30/2021	-	Student
DURAND	Sandra	TSSR-2001	01/05/2020	12/31/2020	-	Student
DOE	John	-	-	-	AIS	Teacher
SMITH	Jane	-	-	-	AIS	Teacher
JOHNSON	Bob	-	-	-	DEV	Teacher
BROWN	Alice	-	-	-	DEV	Teacher
JONES	Charlie	-	-	-	TSSR	Teacher
KNOX	Zackary	-	-	-	-	administrative
MARKS	Diego	-	-	-	-	administrative
SUAREZ	Milton	-	-	-	-	administrative
WU	Polly	-	-	-	-	administrative
OCONNELL	Betty	-	-	-	-	administrative

Automating user account creation

From an Excel file provided to you in the form above, automate the creation of folders and user accounts. The name of the cohort is provided to you, it is composed of the year followed by the month. You can recreate a file more suited to the work requested, from the original file. The required automation consists of:

- Organizing folders and subfolders for each cohort.
- Setting up accounts for each user using the format **FirstLetterOfFirstNameLastName**, with no spaces and only the first letter in capital letters (e.g. Lien TRANKIL's account would be **Ltrankil** and Sebastien GRAND DUC's would be **Sgrandduc**). We will keep the user's last name and first name in the account settings.
- Providing each student with a personal folder in their class folder with their login name.
- Creating groups in the format **gg-cohortName** (e.g. gg-ais1910) and **gg-disciplineName** for the teachers.
- Creating groups as gg-students, gg-teachers and gg-administrative.
- Creating OUs in the format **ou-cohortName** (e.g. ou-ais1910).

System Maintenance

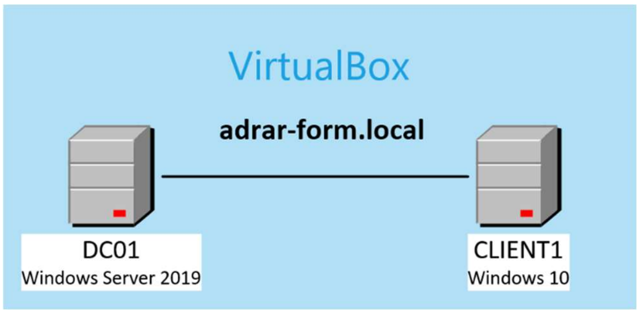
Propose a solution for automatically deleting accounts and files 6 months after the end of the training.

System auditing

Implement an audit system for the teachers folder of a class to track and monitor access.

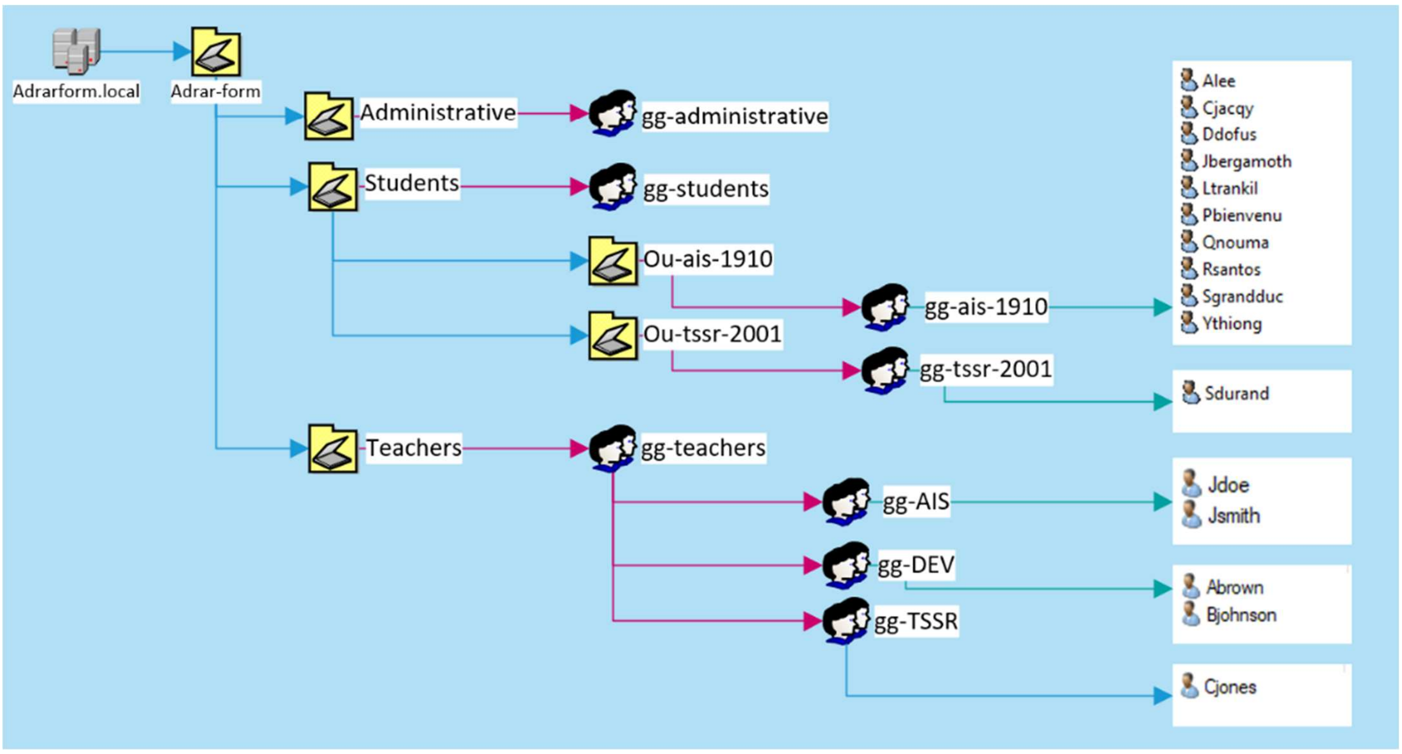
Plan of the project

To showcase our project, we will create a small model that includes an Active Directory server VM called DC01 and a windows client VM called CLIENT1 on VirtualBox. DC01 will be the domain controller for the domain "adrarform.local" and CLIENT1 will be connected to this domain.



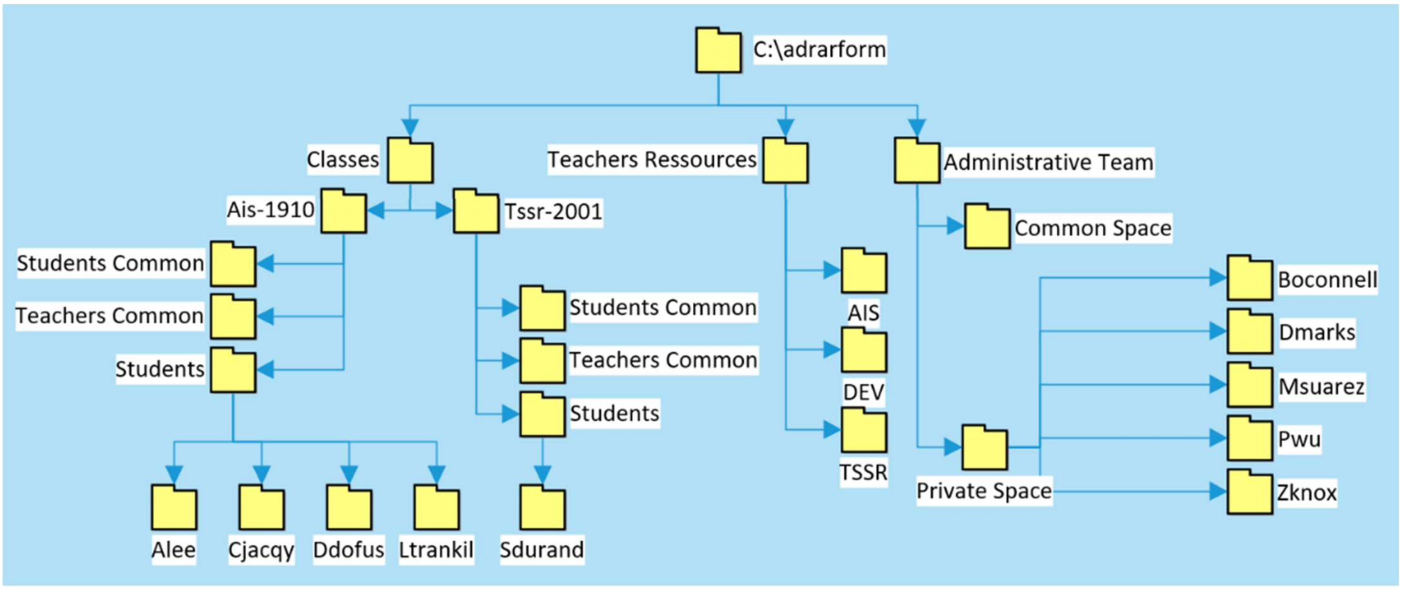
AD users, computers, groups and OUs

To effectively manage users and groups within the organization, it is advisable to create a main organizational unit (OU) within the adrarform.local domain. This main OU can be named adrarform and serve as the parent for additional OUs. This allows for the application of group policies to all users by simply applying them to the adrarform OU. It is also recommended to create a separate OU for each cohort of students. The illustration shows the use of multiple OUs and the creation of various groups to organize users:



Shared folders tree structure

DC01 will also serve as a file server, with a shared folder located on the C drive. The shared folder will have the following structure. Once it has been created and NTFS permissions have been properly applied according to the specifications, it will be shared on the network and made accessible to domain users. Additionally, a network drive will be mapped through the use of a home drive in each user's profile.



NTFS permissions

NTFS (New Technology File System) permissions are a system used by Microsoft Windows to control access to files and folders stored on NTFS-formatted hard drives. NTFS permissions allow you to specify which users or groups have access to a particular file or folder, and what actions they are allowed to perform on it.

There are two types of NTFS permissions: basic permissions and advanced permissions. Basic permissions include basic read, write, and execute permissions, while advanced permissions allow you to fine-tune access control by specifying more detailed actions, such as the ability to delete a file or change its attributes.

In addition to specifying permissions for individual users and groups, you can also set inherited permissions, which are passed down to child objects within a folder hierarchy. This allows you to set permissions at the top level of a folder structure and have them automatically apply to all subfolders and files contained within it.

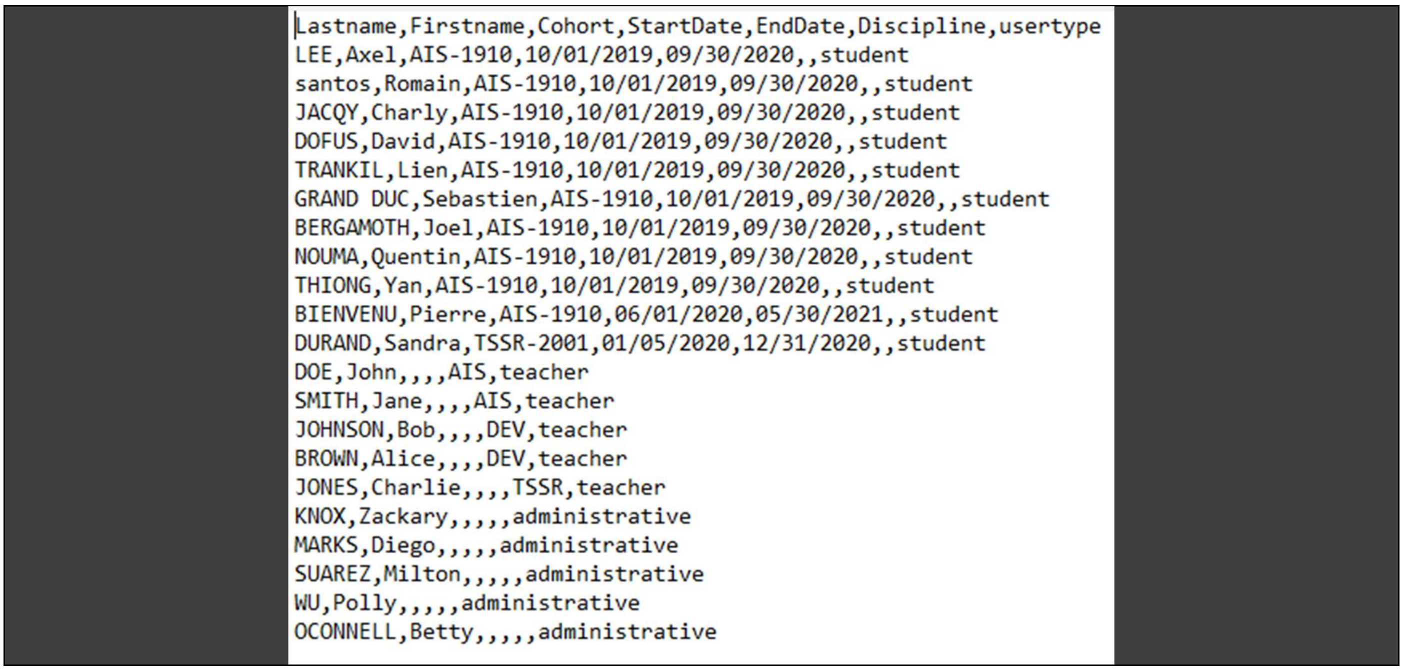
ACE and ACL

An Access Control Entry (ACE) is a specific set of permissions that is applied to a user or group in Microsoft Windows. ACEs are used to define the access rights and permissions for a particular user or group, and are stored within Access Control Lists (ACLs).

In Windows, an ACE consists of a security identifier (SID) that identifies the user or group to which the ACE applies, and a set of access rights that specify what actions the user or group is allowed to perform on a particular resource. For example, an ACE might grant a user the right to read and execute a file, or give a group permission to read and write to a folder.

When a user attempts to access a resource, the system checks the ACEs associated with the user and the resource to determine whether the access request should be granted or denied. If the user's SID is present in an ACE that grants the requested access rights, the request is granted; otherwise, it is denied.

to import Adrarform users into PowerShell we need to convert them into an appropriate CSV file:



```
Lastname,Firstname,Cohort,StartDate,EndDate,Discipline,usertype
LEE,Axel,AIS-1910,10/01/2019,09/30/2020,,student
santos,Romain,AIS-1910,10/01/2019,09/30/2020,,student
JACQY,Charly,AIS-1910,10/01/2019,09/30/2020,,student
DOFUS,David,AIS-1910,10/01/2019,09/30/2020,,student
TRANKIL,Lien,AIS-1910,10/01/2019,09/30/2020,,student
GRAND DUC,Sebastien,AIS-1910,10/01/2019,09/30/2020,,student
BERGAMOTH,Joel,AIS-1910,10/01/2019,09/30/2020,,student
NOUMA,Quentin,AIS-1910,10/01/2019,09/30/2020,,student
THIONG,Yan,AIS-1910,10/01/2019,09/30/2020,,student
BIENVENU,Pierre,AIS-1910,06/01/2020,05/30/2021,,student
DURAND,Sandra,TSSR-2001,01/05/2020,12/31/2020,,student
DOE,John,,,,AIS,teacher
SMITH,Jane,,,,AIS,teacher
JOHNSON,Bob,,,,DEV,teacher
BROWN,Alice,,,,DEV,teacher
JONES,Charlie,,,,TSSR,teacher
KNOX,Zackary,,,,,administrative
MARKS,Diego,,,,,administrative
SUAREZ,Milton,,,,,administrative
WU,Polly,,,,,administrative
OCONNELL,Betty,,,,,administrative
```

Start of our script

- ✓ Note: Entire code is available in my GitHub at: https://github.com/ershad-ra/Powershell-AD-NTFS-audit

To begin our script, we will import the required modules. Next, we will import users.csv file, declare some variables that will be utilized throughout the script. Afterwards, we will create organizational units (OUs), folders, users and groups, add users to the groups, set NTFS permissions, enable auditing, and share folders as needed.

Defining variables

```
Import-Module ActiveDirectory
Import-Module NTFSsecurity
Import-Module SmbShare
$users          = Import-CSV "C:\users.csv"
$NetBios        = "adrarform"
$DomainDN       = "DC=$NetBios,DC=local"
$adrarOUDN      = "OU=$NetBios,$DomainDN"
$mainPath       = "C:\Share"
$mainDirectories = @(
    $mainPath,
    "$mainPath\Administrative Team",
    "$mainPath\Administrative Team\Common Space",
    "$mainPath\Administrative Team\Private Space",
    "$mainPath\Classes",
    "$mainPath\Teachers Ressources"
)
$OUNames = @("Students", "Teachers", "Administrative")
$GHT = @{
    GroupCategory = "Security"
    GroupScope    = "Global"
}
```

This code snippet imports three PowerShell modules. The script then defines several variables:

$users: This variable is assigned the contents of the CSV file located at C:\users.csv.

$NetBios: This variable is assigned the string value "adrarform" which represents NetBios of the domain.

$DomainDN: This variable is assigned the string value "DC=$NetBios,DC=local", which represents the distinguished name (DN) of the domain. The $NetBios variable is used to substitute the value of adrarform in the string.

$adrarOUDN: This variable is assigned the string value "OU=$NetBios,$DomainDN", which represents the DN of the main organizational unit (OU).

$mainPath: This variable is assigned the string value "C:\Share", which represents the path of the main shared folder on the server.

$mainDirectories: This variable is assigned an array of strings containing the paths of several subdirectories within the main shared folder.

$OUNames: This variable is assigned an array of strings containing names of OUs we want to create.

$GHT: This variable is assigned a hash table containing two key-value pairs. This hash table will be used to create groups.

Creating OUs and main groups and directories

```
if (!(Get-ADOrganizationalUnit -Filter {Name -eq $NetBios})) {
    Write-Host "`n`nCreating the $NetBios OU!" -ForegroundColor Green
    New-ADOrganizationalUnit -Name $NetBios -Path $DomainDN
}
foreach ($OUName in $OUNames) {
    $OUDN = "OU=$OUName,$adrarOUDN"
    if (!(Get-ADOrganizationalUnit -Filter {Name -eq $OUName})) {
        Write-Host "Creating the $OUName OU!" -ForegroundColor Green
        New-ADOrganizationalUnit -Name $OUName -Path $adrarOUDN
        New-ADGroup -Name "gg-$OUName" @GHT -Path $OUDN
    }
}
foreach ($path in $mainDirectories) {
    if (!(Test-Path -Path $path -PathType Container )) {
        New-Item -ItemType Directory $path > $null
        Write-Host "$path has beed created." -ForegroundColor Green
        Clear-NTFSAccess -Path $path -DisableInheritance
        Add-NTFSAccess -Path $path -Account "$NetBios\Domain Admins" -AccessRights "FullControl"
        switch ($path) {
            $mainPath {
                Add-NTFSAccess -Path $path -Account "$NetBios\Domain Users"`
                 -AccessRights ReadAndExecute -AppliesTo ThisFolderSubfoldersAndFiles
            }
            $($mainDirectories[1]) {
                Add-NTFSAccess -Path $path -Account "$NetBios\gg-$($OUNames[2])"`
                 -AccessRights ReadAndExecute -AppliesTo ThisFolderSubfoldersAndFiles
            }
            $($mainDirectories[2]) {
                Add-NTFSAccess -Path $path -Account "$NetBios\gg-$($OUNames[2])"`
                 -AccessRights ReadAndExecute,Write -AppliesTo ThisFolderOnly
                Add-NTFSAccess -Path $path -Account "$NetBios\gg-$($OUNames[2])"`
                 -AccessRights ReadAndExecute -AppliesTo ThisFolderSubfoldersAndFiles
                Add-NTFSAccess -Path $path -Account "CREATOR OWNER"`
                 -AccessRights Modify -AppliesTo SubfoldersAndFilesOnly
            }
            $($mainDirectories[3]) {
                Add-NTFSAccess -Path $path -Account "$NetBios\gg-$($OUNames[2])"`
                 -AccessRights ReadAndExecute -AppliesTo ThisFolderSubfoldersAndFiles
            }
            $($mainDirectories[4]) {
                Add-NTFSAccess -Path $path -Account "$NetBios\gg-$($OUNames[0])"`
                 -AccessRights ReadAndExecute -AppliesTo ThisFolderSubfoldersAndFiles
                Add-NTFSAccess -Path $path -Account "$NetBios\gg-$($OUNames[1])"`
                 -AccessRights ReadAndExecute -AppliesTo ThisFolderSubfoldersAndFiles
            }
            default {
                Add-NTFSAccess -Path $path -Account "$NetBios\gg-$($OUNames[1])"`
                 -AccessRights ReadAndExecute -AppliesTo ThisFolderSubfoldersAndFiles
            }
        }
    }
}
```

This part of the script checks if an OU with the name specified in the $NetBios variable exists. If it does not exist, the script creates an OU with the name adrarform in the domain specified by $DomainDN.

Next, the script loops through the names in the $OUNames array, and for each name it checks if an OU with that name exists. If it does not exist, the script creates an OU and a main security group with the name gg-OUName inside each OU.

Until here the code created OUs (Adrarform, students, teachers, administrative) and the groups (gg-students, gg-teachers, gg-administrative).

Finally, the script loops through the paths in the $mainDirectories array and checks if each path exists. If it does not exist, the script creates the directory with New-Item cmdlets and sets the permissions on it using the Clear-NTFSAccess, Add-NTFSAccess.

If, elseif and else

if statements are used to execute a block of code only if a condition is met. You can also use an *else* clause to specify code to execute if the condition is not met. You can also use an *elseif* clause to specify additional conditions to check.

Foreach loop

The *foreach* loop is a looping construct that allows you to iterate over a collection of items, such as an array or a list. It allows you to execute a block of code for each item in the collection.

Switch statement

The *switch* statement is used to execute a block of code based on a matching value. It is similar to the if statement, but is easier to read when you have multiple conditions to test. The *default* keyword is used in the switch statement to specify a block of code that should be executed if none of the other cases match.

Comparison operators

-*eq* is a comparison operator that is used to test whether two values are equal. It is short for "equal to". You can also use the -*ne* operator to test for inequality, the -*gt* operator to test for greater than, the -*lt* operator to test for less than, and many other comparison operators.

Write-Host cmdlet

This cmdlet is used to write output to the console. It is commonly used to display text messages or other information to the user. You can use the -*ForegroundColor* parameter to change the text color of the output.

Add-NTFSAccess cmdlet

It allows you to specify the path of the file or folder, the security principal (such as a user or group) that you want to grant permissions to, and the permissions that you want to grant.

Here are some of the key parameters that you can use with Add-NTFSAccess: