PRÁCTICA 5 POWERSHELL ASO UD.2

Practica final PowerShell. Mantenimiento de usuarios y grupos en diferentes servidores

1. Exportar usuarios y grupos locales de un servidor

2. Conectarse desde PowerShell a otro servidor EC2 por SSHH 3. replicar usuarios y grupos en EC2

4. Crear un script que sincronice el servidor EC2 con el servidor local

4.1 crear usuarios que falten en el EC2

4.2 crear grupos que falten en el EC2

4.3 Añadir usuarios faltantes a grupos

5 Mantenimiento automático mediante tarea programada.

6 Opcional: Ampliación soporte para múltiples servidores EC2

7 Crear un menú para facilitar la interacción

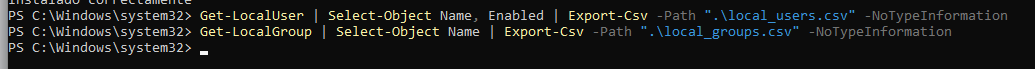
1. Exportar usuarios y grupos locales de un servidor

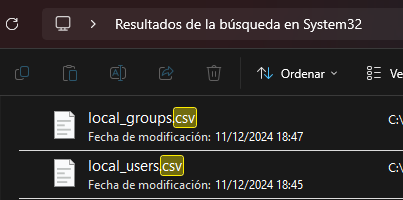
Este lo haré en mi ordenador

Utilizar PowerShell para obtener los usuarios y grupos locales de un servidor y exportar los datos a CSV.

Get-LocalUser | Select-Object Name, Enabled | Export-Csv -Path "usuarios.csv" – NoTypeInformation

Get-LocalGroup | Select-Object Name | Export-Csv -Path "grupos.csv" -NoTypeInformation



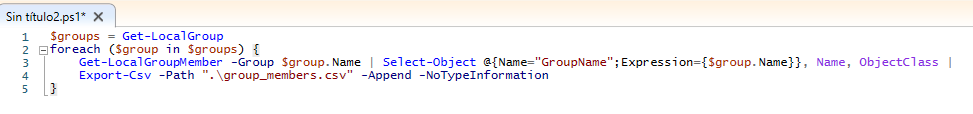


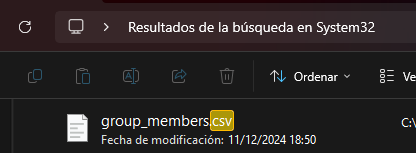
foreach ($grupo in Get-LocalGroup) {

Get-LocalGroupMember -Group $grupo.Name | Select-Object

@{Name='Grupo';Expression={$grupo.Name}}, Name, ObjectClass | Export-Csv -Path "$($grupo.Name)\_miembros.csv" -Append -NoTypeInformation

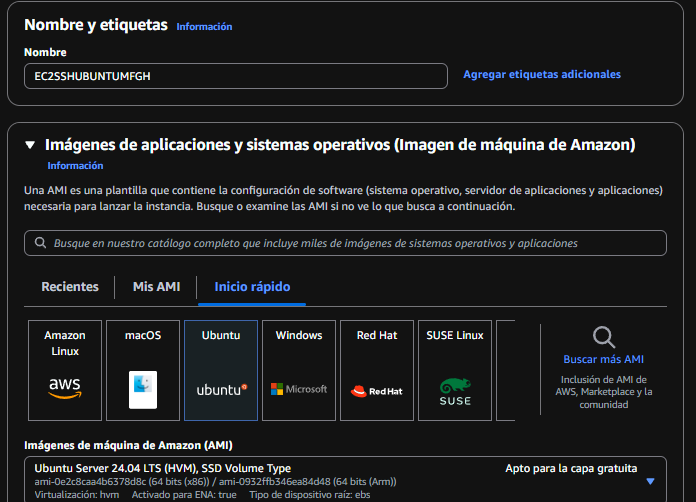
}





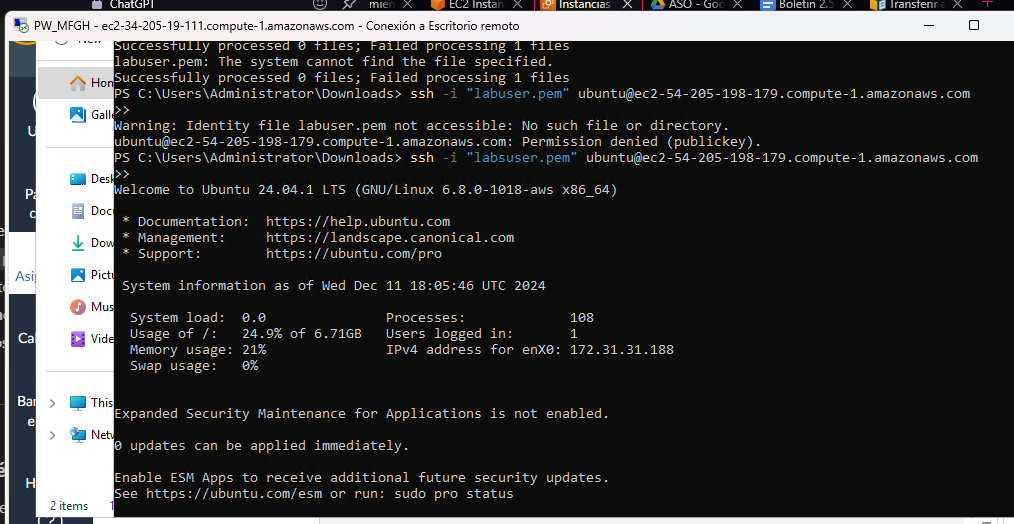
2. Conectarse desde PowerShell a otro servidor EC2 por SSHH

o Levantar en AWS un servidor Ubuntu sobre EC2



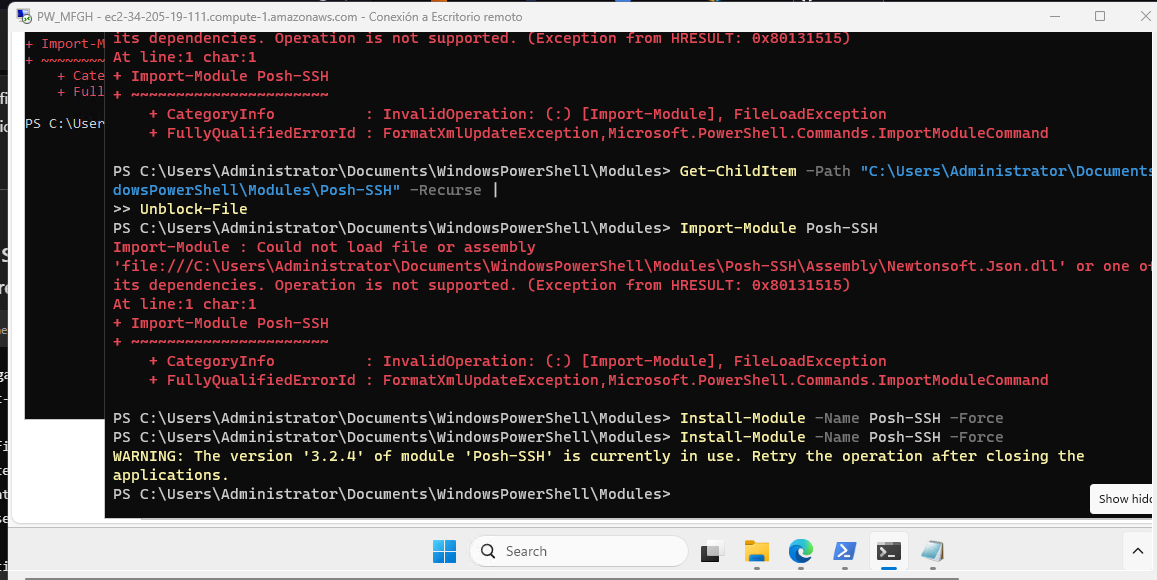
o Instalar el módulo de PowerShell para SSH si es necesario y por SSH nos conectaremos al servidor EC2.

Descargamos las claves en la máquina virtual y con eso ya las usamos.



3. replicar usuarios y grupos en EC2

Crearnos un script que a partir de los archivos CSV que tenemos en local, replique en el servidor remoto EC2 los usuarios y grupos.



Import-Module Posh-SSH

# Configuración del servidor y clave privada

$remoteServer = "ec2-54-205-198-179.compute-1.amazonaws.com"

$privateKeyPath = "C:\Users\Administrator\Downloads\labsuser.pem" # Ruta correcta de la clave PEM

$sshUser = "ubuntu"

$sshSession = New-SSHSession -ComputerName $remoteServer -KeyFile $privateKeyPath

if ($sshSession -eq $null) {

Write-Host "No se pudo establecer la conexión SSH." -ForegroundColor Red

exit

}

# Rutas de los archivos CSV en la carpeta Downloads

$usuariosCsv = "C:\Users\Administrator\Downloads\usuarios.csv"

$gruposCsv = "C:\Users\Administrator\Downloads\grupos.csv"

$miembrosCsv = "C:\Users\Administrator\Downloads\grupo\_miembros.csv"

# Cargar usuarios desde CSV y replicar

$usuarios = Import-Csv -Path $usuariosCsv

foreach ($usuario in $usuarios) {

$command = "sudo useradd -m -s /bin/bash $($usuario.Name)"

Invoke-SSHCommand -SessionId $sshSession.SessionId -Command $command

Write-Host "Usuario creado: $($usuario.Name)" -ForegroundColor Green

}

# Cargar grupos desde CSV y replicar

$grupos = Import-Csv -Path $gruposCsv

foreach ($grupo in $grupos) {

$command = "sudo groupadd $($grupo.Name)"

Invoke-SSHCommand -SessionId $sshSession.SessionId -Command $command

Write-Host "Grupo creado: $($grupo.Name)" -ForegroundColor Green

}

# Cargar membresías desde CSV y replicar

$miembros = Import-Csv -Path $miembrosCsv

foreach ($miembro in $miembros) {

$command = "sudo usermod -a -G $($miembro.Grupo) $($miembro.Name)"

Invoke-SSHCommand -SessionId $sshSession.SessionId -Command $command

Write-Host "Usuario $($miembro.Name) añadido al grupo $($miembro.Grupo)" -ForegroundColor Green

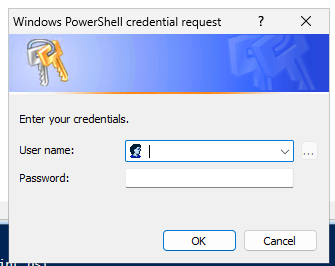
}

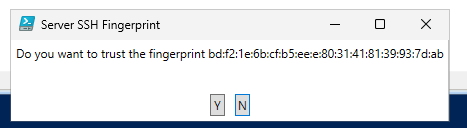
# Cerrar sesión SSH

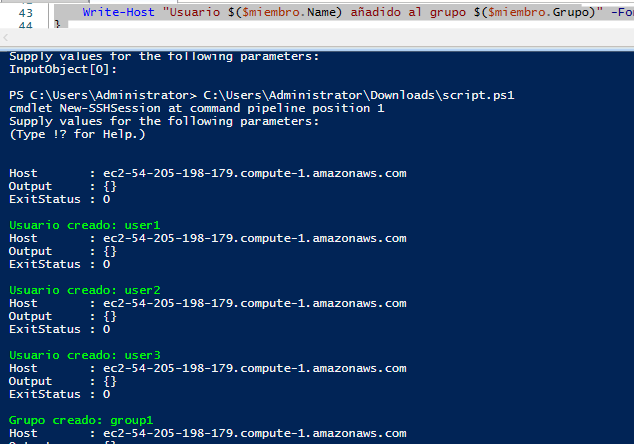
Remove-SSHSession -SessionId $sshSession.SessionId

Write-Host "Sincronización completada y sesión SSH cerrada." -ForegroundColor Yellow

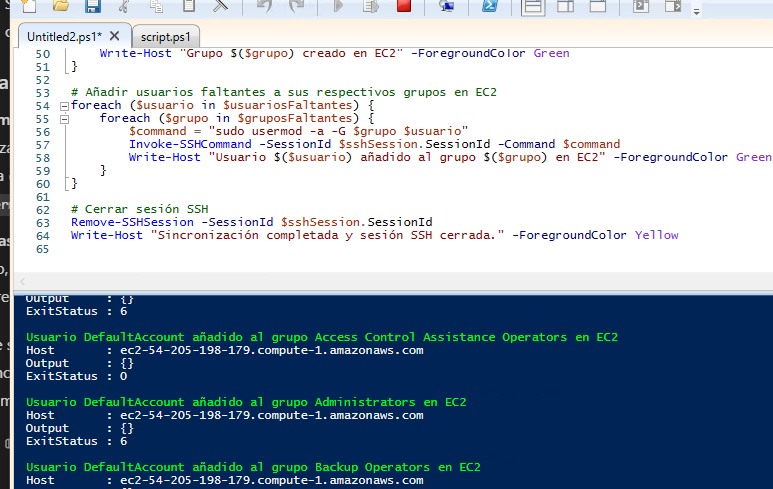
Introducimos ahi “ubuntu” de username y la password vacía.







4. Crear un script que sincronice el servidor EC2 con el servidor local



Import-Module Posh-SSH

$remoteServer = "ec2-54-205-198-179.compute-1.amazonaws.com"

$privateKeyPath = "C:\Users\Administrator\Downloads\labsuser.pem"

$sshUser = "ubuntu"

# Iniciar sesión SSH usando la clave privada y el usuario

$sshSession = New-SSHSession -ComputerName $remoteServer -KeyFile $privateKeyPath

if ($sshSession -eq $null) {

Write-Host "No se pudo establecer la conexión SSH." -ForegroundColor Red

exit

}

# Obtener usuarios locales

$localUsuarios = Get-LocalUser | Select-Object -ExpandProperty Name

# Obtener usuarios en el servidor EC2

$commandGetUsuariosEC2 = "cut -d: -f1 /etc/passwd"

$usuariosEC2 = Invoke-SSHCommand -SessionId $sshSession.SessionId -Command $commandGetUsuariosEC2

$usuariosEC2 = $usuariosEC2.Output -split "`n"

# Comparar usuarios y encontrar los que faltan en EC2

$usuariosFaltantes = $localUsuarios | Where-Object { $\_ -notin $usuariosEC2 }

# Crear usuarios faltantes

foreach ($usuario in $usuariosFaltantes) {

$command = "sudo useradd -m -s /bin/bash $($usuario)"

Invoke-SSHCommand -SessionId $sshSession.SessionId -Command $command

Write-Host "Usuario $($usuario) creado en EC2" -ForegroundColor Green

}

# Obtener grupos locales

$localGrupos = Get-LocalGroup | Select-Object -ExpandProperty Name

# Obtener grupos en el servidor EC2

$commandGetGruposEC2 = "cut -d: -f1 /etc/group"

$gruposEC2 = Invoke-SSHCommand -SessionId $sshSession.SessionId -Command $commandGetGruposEC2

$gruposEC2 = $gruposEC2.Output -split "`n"

# Comparar grupos y encontrar los que faltan en EC2

$gruposFaltantes = $localGrupos | Where-Object { $\_ -notin $gruposEC2 }

# Crear grupos faltantes en EC2

foreach ($grupo in $gruposFaltantes) {

$command = "sudo groupadd $($grupo)"

Invoke-SSHCommand -SessionId $sshSession.SessionId -Command $command

Write-Host "Grupo $($grupo) creado en EC2" -ForegroundColor Green

}

# Añadir usuarios faltantes

foreach ($usuario in $usuariosFaltantes) {

foreach ($grupo in $gruposFaltantes) {

$command = "sudo usermod -a -G $grupo $usuario"

Invoke-SSHCommand -SessionId $sshSession.SessionId -Command $command

Write-Host "Usuario $($usuario) añadido al grupo $($grupo) en EC2" -ForegroundColor Green

}

}

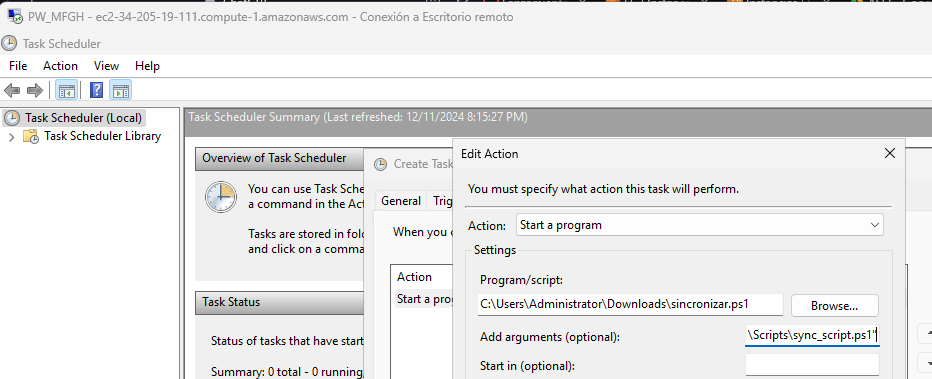
# Cerrar sesión SSH

Remove-SSHSession -SessionId $sshSession.SessionId

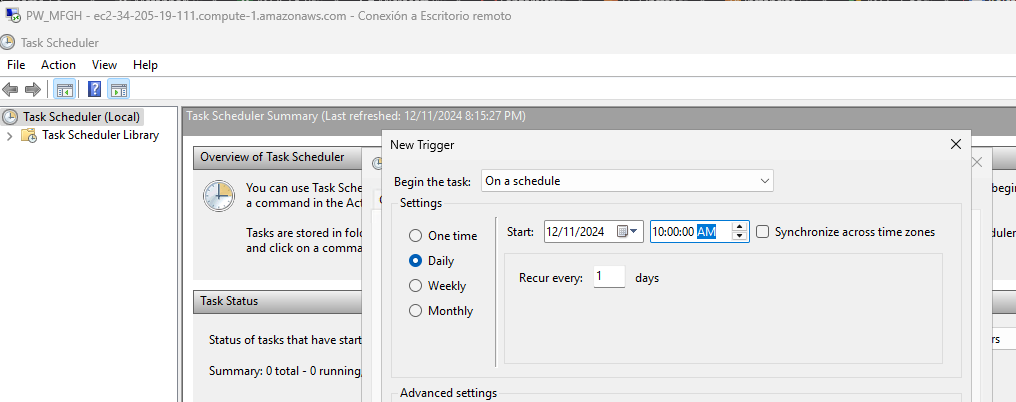
Write-Host "Sincronización completada y sesión SSH cerrada." -ForegroundColor Yellow

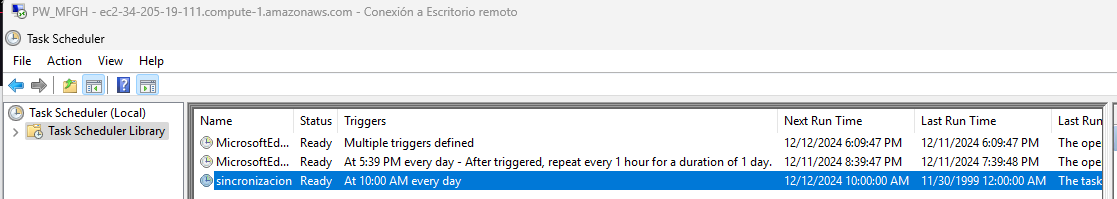
5 Mantenimiento automático mediante tarea programada.

Configurar una tarea programada en el servidor local para exportar usuarios y sincronizarlos con EC2 todos los días a las 10:00 am. Debe dejar registro en un log. Tenemos que crear un script donde exportamos y sincronizamos, o sea, une el punto 1 y punto 4.



Añadimos el siguiente argumento -ExecutionPolicy Bypass -File "C:\Scripts\sync\_script.ps1"





6 Opcional: Ampliación soporte para múltiples servidores EC2

La idea es sincronizar usuarios y grupos no solo en una instancia EC2, sino en varias. Esto se puede lograr creando un script que lea una lista de servidores y ejecute la sincronización en cada uno. Para ello lanzaremos dos nuevos servidores EC2 (tendremos tres en total)

Implementación:

1. Lista de servidores: Crear un archivo CSV con los detalles de las instancias EC2:

ServerName,IPAddress,KeyPath

EC2-Instance1,192.168.1.1,C:\Keys\ec2-key1.pem

EC2-Instance2,192.168.1.2,C:\Keys\ec2-key2.pem

2. Script para iterar sobre los servidores: Modificar el script de sincronización para que recorra la lista de servidores:

powershell

Copiar código

$servers = Import-Csv -Path "servers.csv"

foreach ($server in $servers) {

Write-Output "Conectando a $($server.ServerName)..."

New-SSHSession -ComputerName $server.IPAddress -KeyFile $server.KeyPath

# Sincronizar usuarios y grupos en este servidor

$usuarios = Import-Csv -Path "usuarios.csv"

foreach ($usuario in $usuarios) {

Invoke-SSHCommand -SessionId 0 -Command "sudo useradd $($usuario.Name)"

}

# Cerrar la sesión SSH

Remove-SSHSession -SessionId 0

}

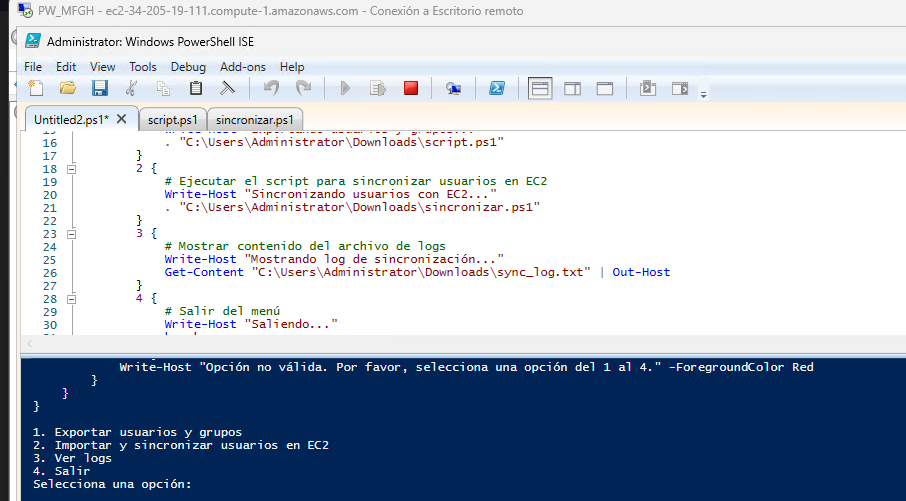
7 Crear un menú para facilitar la interacción

5

PRÁCTICA 5 POWERSHELL ASO UD.2

3. Crear un menú interactivo en la consola de PS para facilitar la interacción

La idea es facilitar la interacción con el script para usuarios que no están familiarizados con PowerShell.



Function Mostrar-Menu {

Write-Host "1. Exportar usuarios y grupos"

Write-Host "2. Importar y sincronizar usuarios en EC2"

Write-Host "3. Ver logs"

Write-Host "4. Salir"

}

while ($true) {

Mostrar-Menu

$opcion = Read-Host "Selecciona una opción"

switch ($opcion) {

1 {

# Ejecutar el script para exportar usuarios y grupos

Write-Host "Exportando usuarios y grupos..."

. "C:\Users\Administrator\Downloads\script.ps1"

}

2 {

# Ejecutar el script para sincronizar usuarios en EC2

Write-Host "Sincronizando usuarios con EC2..."

. "C:\Users\Administrator\Downloads\sincronizar.ps1"

}

3 {

# Mostrar contenido del archivo de logs

Write-Host "Mostrando log de sincronización..."

Get-Content "C:\Users\Administrator\Downloads\sync\_log.txt" | Out-Host

}

4 {

# Salir del menú

Write-Host "Saliendo..."

break

}

default {

Write-Host "Opción no válida. Por favor, selecciona una opción del 1 al 4." -ForegroundColor Red

}

}

}