

SCRIPTING EN SISTEMAS OPERATIVOS.

Raúl Arribas Redondo

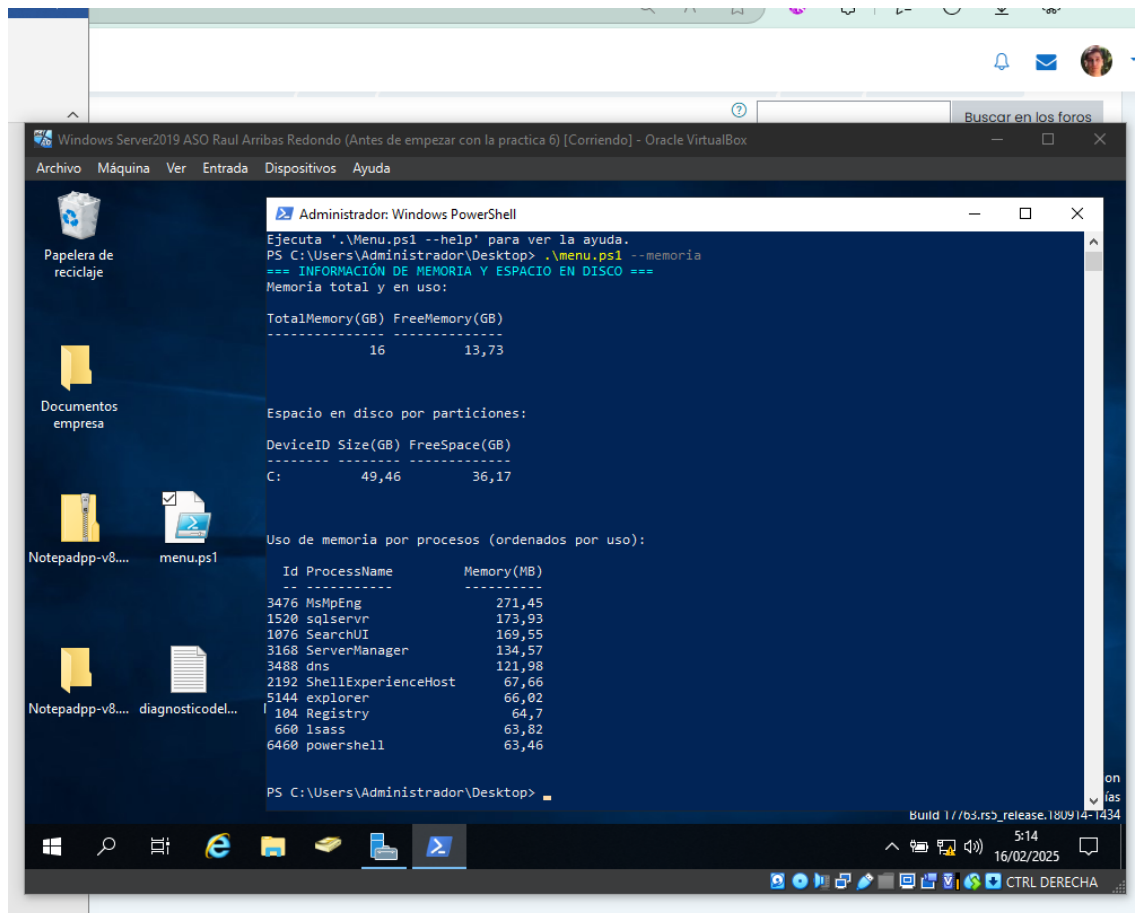
ASO

1 Contenido

2	Apartado 1 (script bash o PowerShell- GNU/Linux o Windows Server)	2
2.1	Codigo del script utilizado:.....	5
3	Apartado 2 (script PowerShell - Windows Server)	10
3.1	Script para crear usuarios OU:	12
4	Apartado 3 (script cmd o PowerShell- Windows Server)	17
5	Apartado 4a (script en bash - GNU/Linux).....	20
5.1	Script utilizado:	22

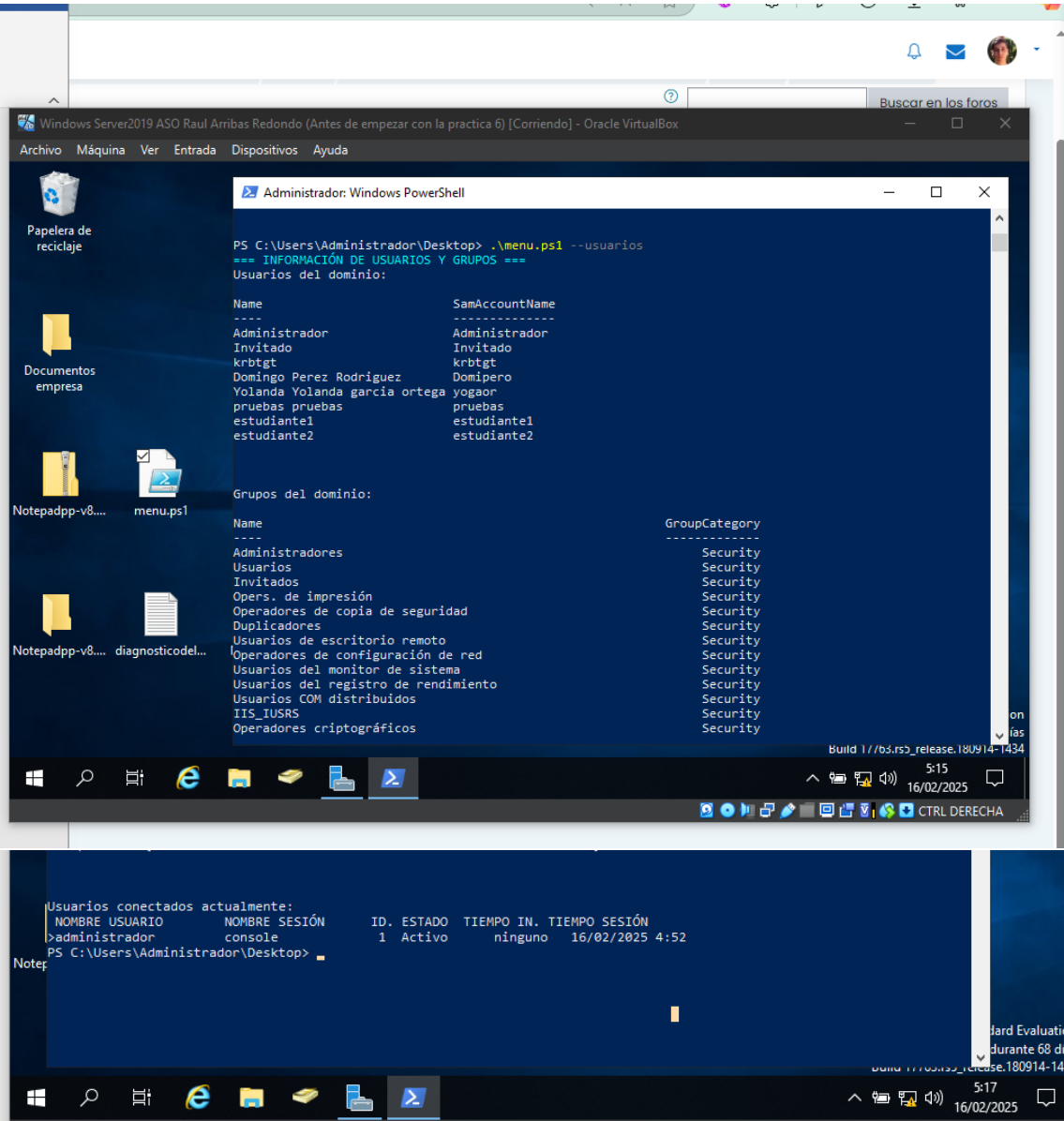
2 Apartado 1 (script bash o PowerShell- GNU/Linux o Windows Server)

- Crea un script en Powershell o en bash que muestre un menú por consola con tres apartados o que los muestre como parámetros para llevar a cabo distintas operaciones de unidades vistas previamente:
- Un apartado que muestre información relacionada con el estado y monitorización de la memoria y espacio en disco del sistema (cantidad de memoria real, porcentaje de uso de los procesos, prioridades, gráficas, particiones, etc.)



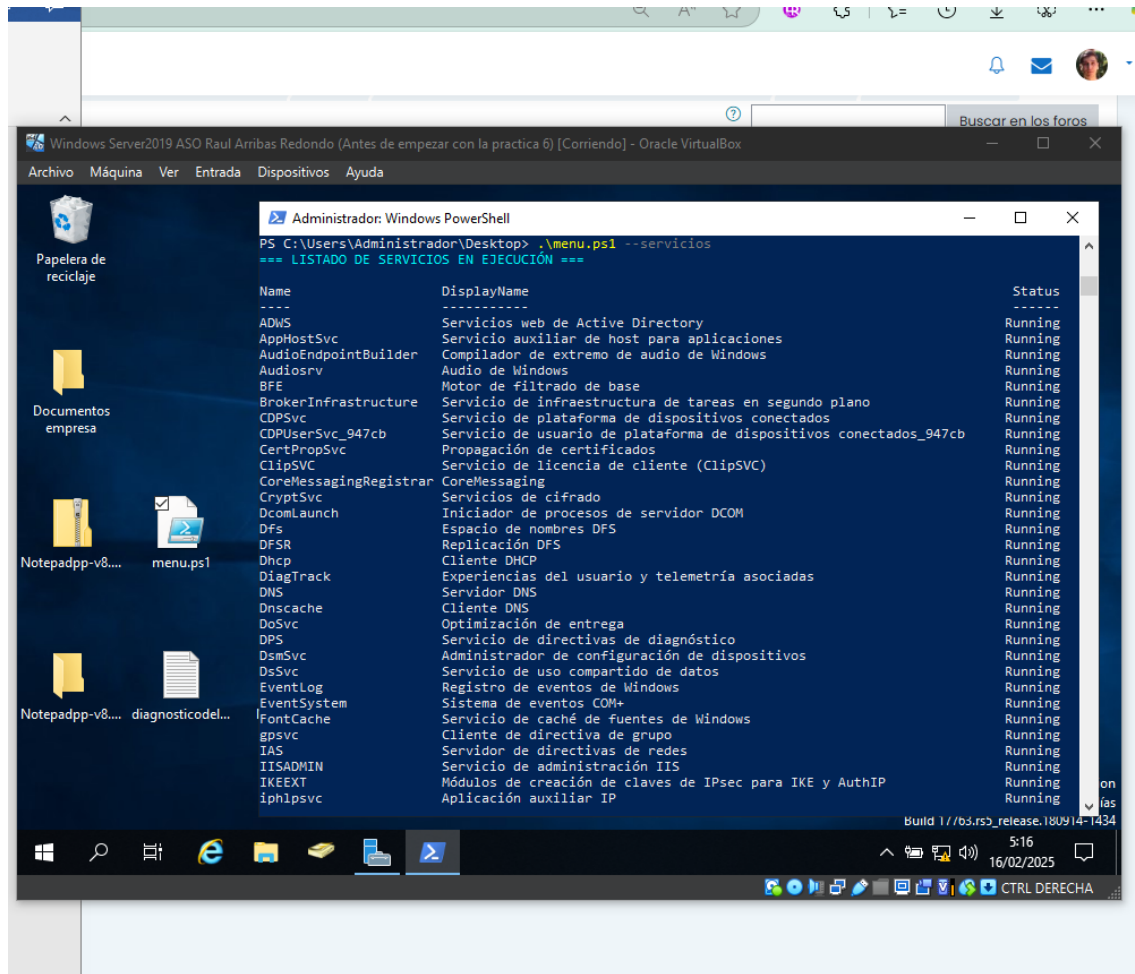
- Un apartado que muestre información relacionada con los usuarios y unidades organizativas o grupos del Dominio: administradores, nombres de las uo o grupos y usuarios del Dominio actuales

Scripting en sistemas operativos.

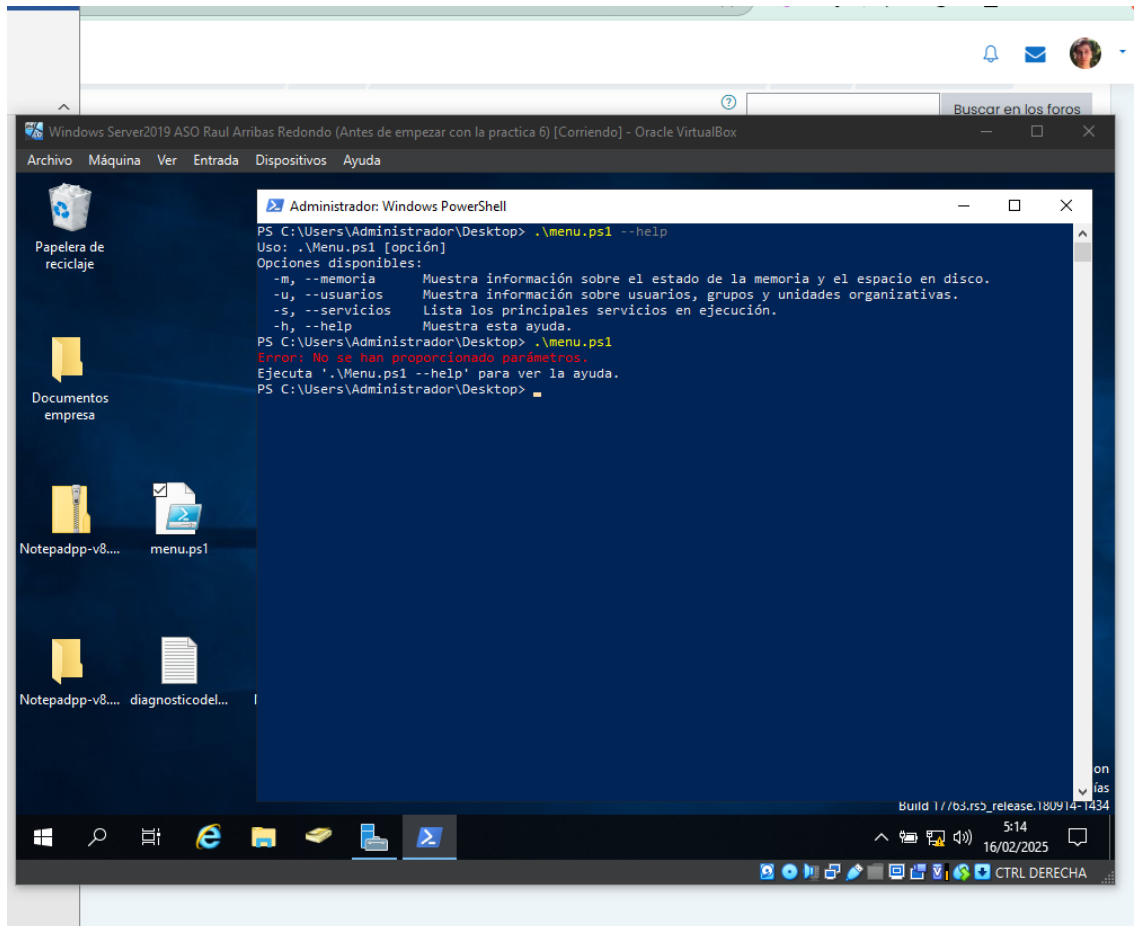


Scripting en sistemas operativos.

- Un último apartado que muestre información relacionada con los principales servicios del servidor: Listar los principales que estén actualmente en ejecución



- El script sólo podrá ser ejecutado por un usuario con permisos de administrador. Si no es así, deberá salir y mostrar un mensaje de error. Además si al script no se le pasan parámetros, deberá mostrar un mensaje de ayuda indicando una sintaxis de uso.



2.1 Código del script utilizado:

Script: Menu.ps1

Descripción: Muestra un menú con opciones para monitorizar el sistema, gestionar usuarios y listar servicios.

Requisitos: El usuario debe tener permisos de administrador para ejecutar el script.

Función para mostrar la ayuda

```
function Mostrar-Ayuda {
```

```
    Write-Host "Uso: .\Menu.ps1 [opción]"
```

```
    Write-Host "Opciones disponibles:"
```

```
    Write-Host " -m, --memoria    Muestra información sobre el estado de la memoria y el espacio en disco."
```

Scripting en sistemas operativos.

```
Write-Host " -u, --usuarios    Muestra información sobre usuarios, grupos y unidades organizativas."
```

```
Write-Host " -s, --servicios  Lista los principales servicios en ejecución."
```

```
Write-Host " -h, --help      Muestra esta ayuda."
```

```
exit 0
```

```
}
```

Función para verificar si el usuario es administrador

```
function Verificar-Permisos {
```

```
    if (-not ([Security.Principal.WindowsPrincipal]
[Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([Security.Principal.WindowsBuiltInRole] "Administrator")) {
```

```
        Write-Host "Error: Este script debe ser ejecutado como administrador." -ForegroundColor Red
```

```
        exit 1
```

```
    }
```

```
}
```

Función para mostrar información sobre memoria y espacio en disco

```
function Informacion-MemoriaDisco {
```

```
    Write-Host "=== INFORMACIÓN DE MEMORIA Y ESPACIO EN DISCO ===" -ForegroundColor Cyan
```

```
    Write-Host "Memoria total y en uso:"
```

```
    Get-WmiObject -Class Win32_OperatingSystem | Select-Object
@{Name="TotalMemory(GB)";Expression={[math]::Round($_.TotalVisibleMemorySize / 1MB, 2)}}, @
{Name="FreeMemory(GB)";Expression={[math]::Round($_.FreePhysicalMemory / 1MB, 2)}} | Format-Table -AutoSize
```

```
    Write-Host ""
```

```
    Write-Host "Espacio en disco por particiones:"
```

```
    Get-WmiObject -Class Win32_LogicalDisk | Where-Object { $_.DriveType -eq 3 } | Select-Object DeviceID, @
{Name="Size(GB)";Expression={[math]::Round($_.Size / 1GB, 2)}}, @
{Name="FreeSpace(GB)";Expression={[math]::Round($_.FreeSpace / 1GB, 2)}} | Format-Table -AutoSize
```

```
    Write-Host ""
```

```
    Write-Host "Uso de memoria por procesos (ordenados por uso):"
```

Scripting en sistemas operativos.

```
Get-Process | Sort-Object WS -Descending | Select-Object Id, ProcessName,  
@{Name="Memory(MB)";Expression={[math]::Round($_.WS / 1MB, 2)}} -First 10 | Format-  
Table -AutoSize  
}
```

Función para mostrar información sobre usuarios y grupos

```
function Informacion-UsuariosGrupos {  
    Write-Host "=== INFORMACIÓN DE USUARIOS Y GRUPOS ===" -ForegroundColor Cyan  
    Write-Host "Usuarios del dominio:"  
    Get-ADUser -Filter * | Select-Object Name, SamAccountName | Format-Table -AutoSize  
    Write-Host ""  
    Write-Host "Grupos del dominio:"  
    Get-ADGroup -Filter * | Select-Object Name, GroupCategory | Format-Table -AutoSize  
    Write-Host ""  
    Write-Host "Usuarios conectados actualmente:"  
    query user  
}
```

Función para listar los principales servicios en ejecución

```
function Listar-Servicios {  
    Write-Host "=== LISTADO DE SERVICIOS EN EJECUCIÓN ===" -ForegroundColor Cyan  
    Get-Service | Where-Object { $_.Status -eq "Running" } | Select-Object Name, DisplayName,  
    Status | Format-Table -AutoSize  
}
```

Verificar si se pasaron argumentos

```
if ($args.Count -eq 0) {  
    Write-Host "Error: No se han proporcionado parámetros." -ForegroundColor Red  
    Write-Host "Ejecuta '.\Menu.ps1 --help' para ver la ayuda."  
    exit 1  
}
```

Scripting en sistemas operativos.

Procesar los argumentos

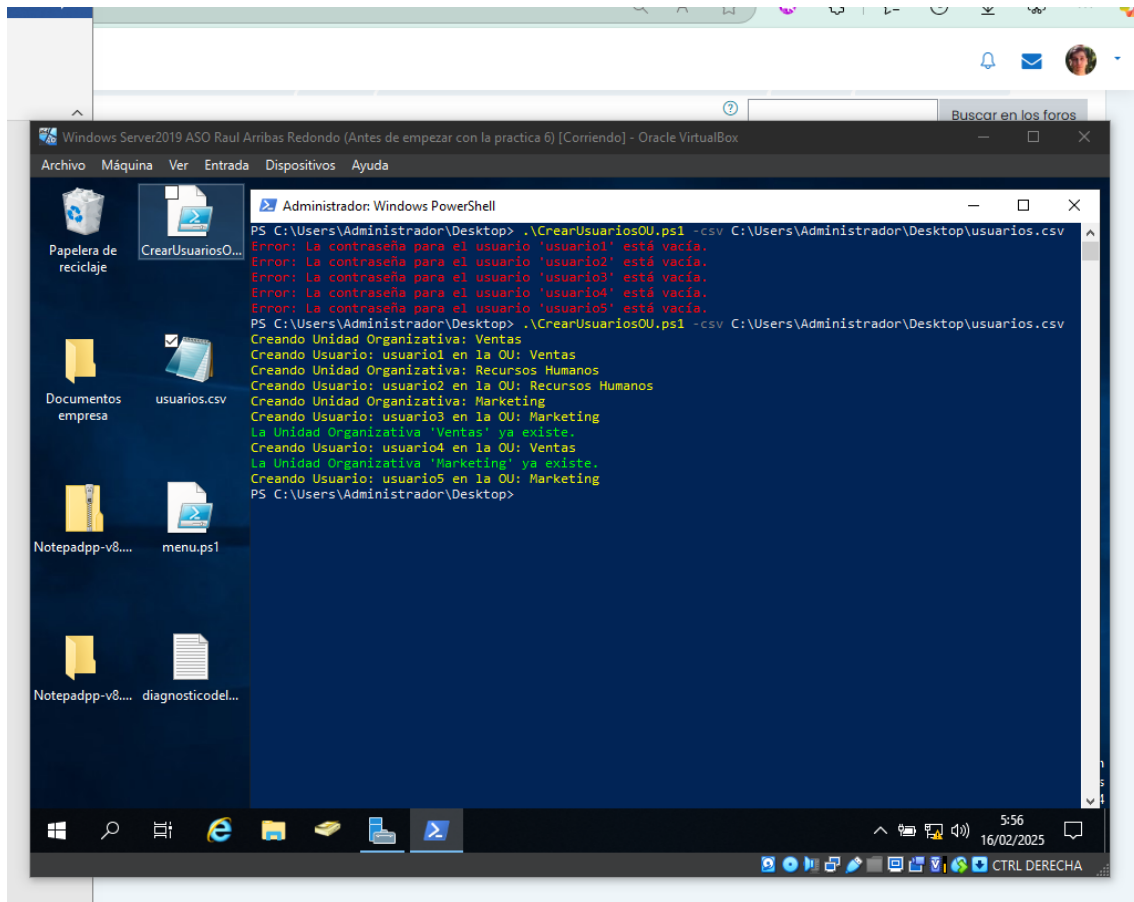
```
switch ($args[0]) {  
    "-m" {  
        Verificar-Permisos  
        Informacion-MemoriaDisco  
    }  
    "--memoria" {  
        Verificar-Permisos  
        Informacion-MemoriaDisco  
    }  
    "-u" {  
        Verificar-Permisos  
        Informacion-UsuariosGrupos  
    }  
    "--usuarios" {  
        Verificar-Permisos  
        Informacion-UsuariosGrupos  
    }  
    "-s" {  
        Verificar-Permisos  
        Listar-Servicios  
    }  
    "--servicios" {  
        Verificar-Permisos  
        Listar-Servicios  
    }  
    "-h" {  
        Mostrar-Ayuda  
    }  
    "--help" {  
        Mostrar-Ayuda  
    }  
}
```

Scripting en sistemas operativos.

```
}  
default {  
    Write-Host "Error: Opción no válida." -ForegroundColor Red  
    Write-Host "Ejecuta '.\Menu.ps1 --help' para ver la ayuda."  
    exit 1  
}  
}
```

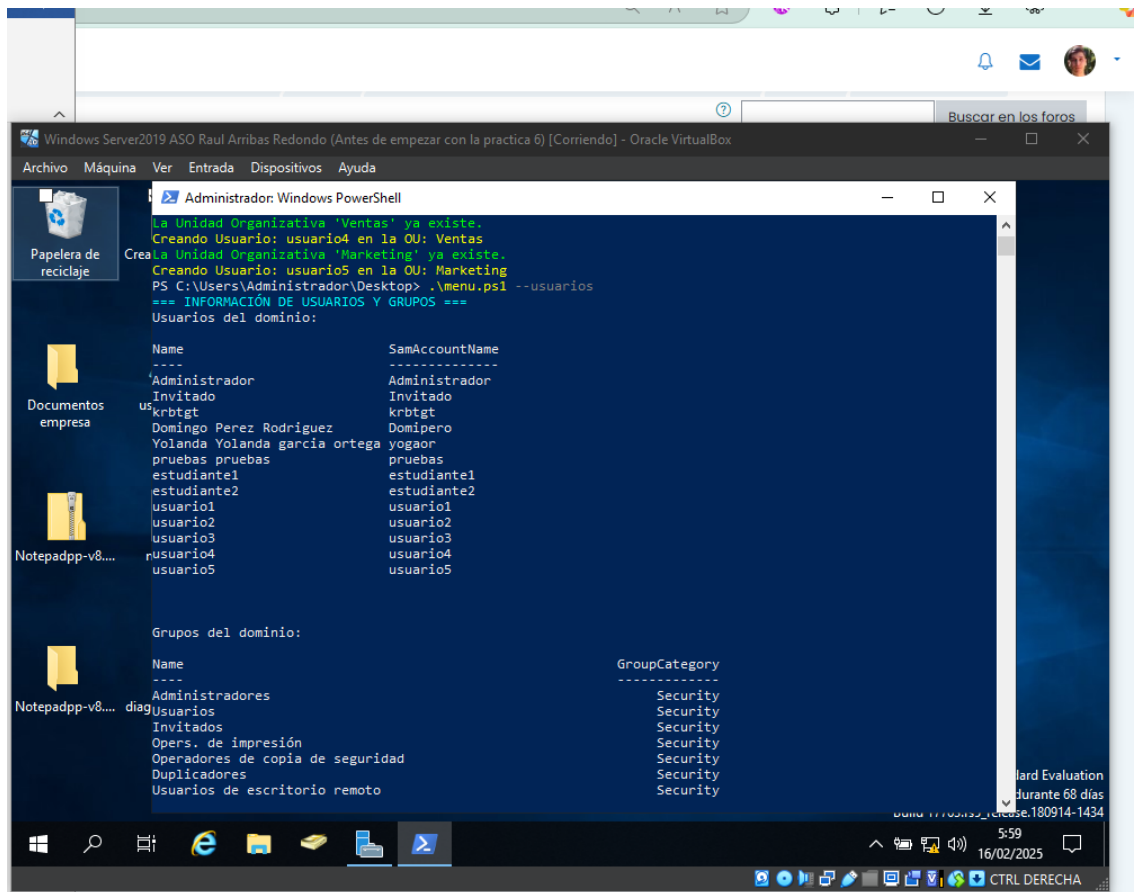
3 Apartado 2 (script PowerShell- Windows Server)

- Crea un archivo .csv con las siguientes columnas: nombre de unidad organizativa, nombre de usuario, contraseña.
- El script deberá de hacer lo siguiente:
- Creará la unidad organizativa de la columna, si no existe previamente.
- Creará en esa unidad organizativa el nombre de usuario descrito con la contraseña que se le haya especificado. Dicha contraseña deberá cambiarla en el primer inicio de sesión

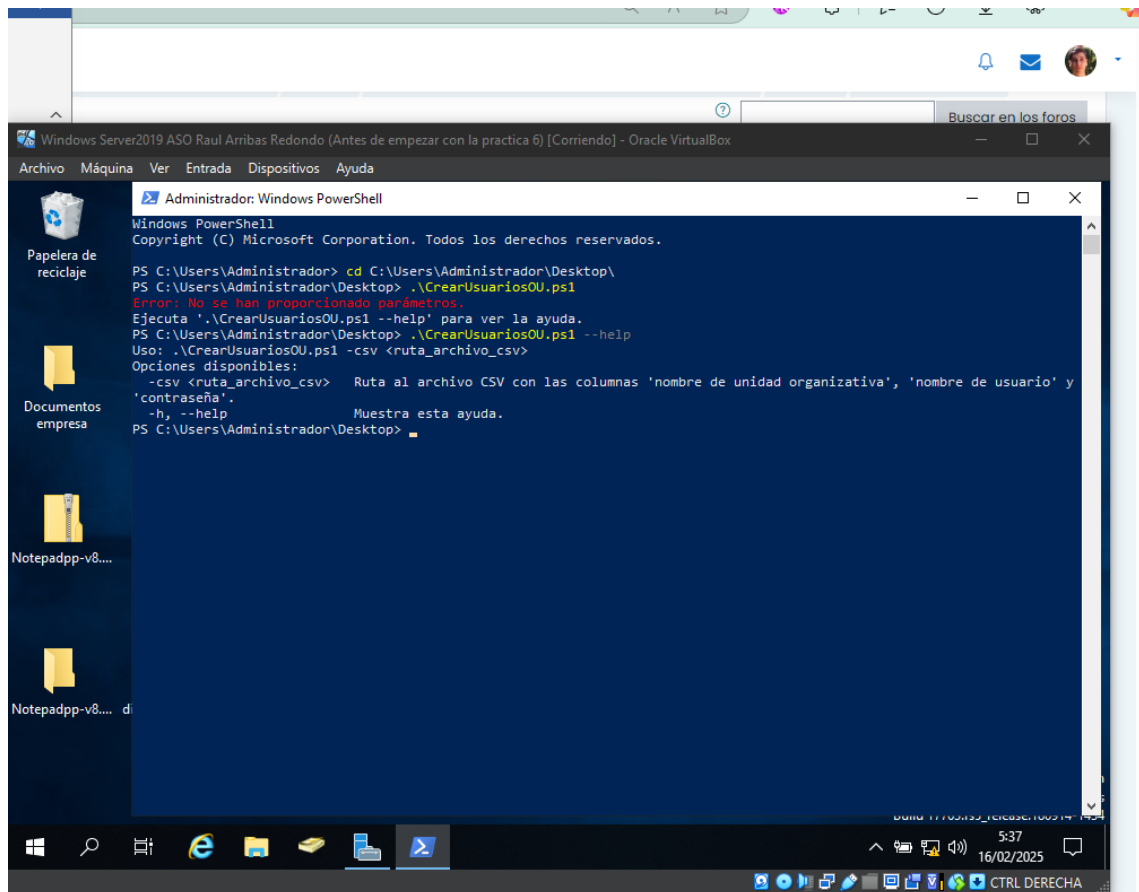


Comprobamos que los usuarios se han creado con la herramienta que hemos creado antes:

Scripting en sistemas operativos.



- El script sólo podrá ser ejecutado por un usuario con permisos de administrador. Si no es así, deberá salir y mostrar un mensaje de error. Además si al script no se le pasan parámetros, deberá mostrar un mensaje de ayuda indicando una sintaxis de uso.



3.1 Script para crear usuarios OU:

Script: CrearUsuariosOU.ps1

Descripción: Crea unidades organizativas (OU) y usuarios en Active Directory a partir de un archivo CSV.

Requisitos: El usuario debe tener permisos de administrador para ejecutar el script.

Función para mostrar la ayuda

```
function Mostrar-Ayuda {
```

```
    Write-Host "Uso: .\CrearUsuariosOU.ps1 -csv <ruta_archivo_csv>"
```

```
    Write-Host "Opciones disponibles:"
```

```
    Write-Host "    -csv <ruta_archivo_csv> Ruta al archivo CSV con las columnas 'nombre de unidad organizativa', 'nombre de usuario' y 'contraseña'."
```

```
    Write-Host "    -h, --help          Muestra esta ayuda."
```

```
    exit 0
```

```
}
```

```
# Función para verificar si el usuario es administrador

function Verificar-Permisos {

    if (-not ([Security.Principal.WindowsPrincipal]
[Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([Security.Principal.WindowsBui
ltInRole] "Administrator")) {

        Write-Host "Error: Este script debe ser ejecutado como administrador." -
ForegroundColor Red

        exit 1

    }

}

# Función para crear OU y usuarios a partir del archivo CSV

function Crear-UsuariosYUnidades {

    param (

        [string]$rutaCSV

    )

    # Verificar si el archivo CSV existe

    if (-not (Test-Path $rutaCSV)) {

        Write-Host "Error: El archivo CSV no existe en la ruta especificada." -ForegroundColor
Red

        exit 1

    }

    # Importar el archivo CSV

    $datos = Import-Csv -Path $rutaCSV

    foreach ($fila in $datos) {

        $ouNombre = $fila.'nombre de unidad organizativa'

        $usuarioNombre = $fila.'nombre de usuario'

        $contraseña = $fila.contraseña
```

```
# Verificar que la contraseña no esté vacía

if ([string]::IsNullOrEmpty($contraseña)) {

    Write-Host "Error: La contraseña para el usuario '$usuarioNombre' está vacía." -
ForegroundColor Red

    continue

}

# Convertir la contraseña a SecureString

$securePassword = ConvertTo-SecureString -String $contraseña -AsPlainText -Force

# Verificar si la OU existe, si no, crearla

if (-not (Get-ADOrganizationalUnit -Filter { Name -eq $ouNombre })) {

    Write-Host "Creando Unidad Organizativa: $ouNombre" -ForegroundColor Yellow

    New-ADOrganizationalUnit -Name $ouNombre -Path "DC=wbc,DC=local" # Dominio
ajustado a wbc.local

} else {

    Write-Host "La Unidad Organizativa '$ouNombre' ya existe." -ForegroundColor
Green

}

# Verificar si el usuario existe, si no, crearlo

if (-not (Get-ADUser -Filter { SamAccountName -eq $usuarioNombre })) {

    Write-Host "Creando Usuario: $usuarioNombre en la OU: $ouNombre" -
ForegroundColor Yellow

    New-ADUser -Name $usuarioNombre `

        -SamAccountName $usuarioNombre `

        -AccountPassword $securePassword `

        -Enabled $true `

        -ChangePasswordAtLogon $true `

        -Path "OU=$ouNombre,DC=wbc,DC=local" # Dominio ajustado a wbc.local

} else {
```

```
        Write-Host "El Usuario '$usuarioNombre' ya existe." -ForegroundColor Green
    }
}
}
```

Verificar si se pasaron argumentos

```
if ($args.Count -eq 0) {
    Write-Host "Error: No se han proporcionado parámetros." -ForegroundColor Red
    Write-Host "Ejecuta '.\CrearUsuariosOU.ps1 --help' para ver la ayuda."
    exit 1
}
```

Procesar los argumentos

```
switch ($args[0]) {
    "-csv" {
        if ($args.Count -lt 2) {
            Write-Host "Error: Falta la ruta del archivo CSV." -ForegroundColor Red
            Write-Host "Ejecuta '.\CrearUsuariosOU.ps1 --help' para ver la ayuda."
            exit 1
        }
        Verificar-Permisos
        Crear-UsuariosYUnidades -rutaCSV $args[1]
    }
    "-h" {
        Mostrar-Ayuda
    }
    "--help" {
        Mostrar-Ayuda
    }
    default {
        Write-Host "Error: Opción no válida." -ForegroundColor Red
    }
}
```

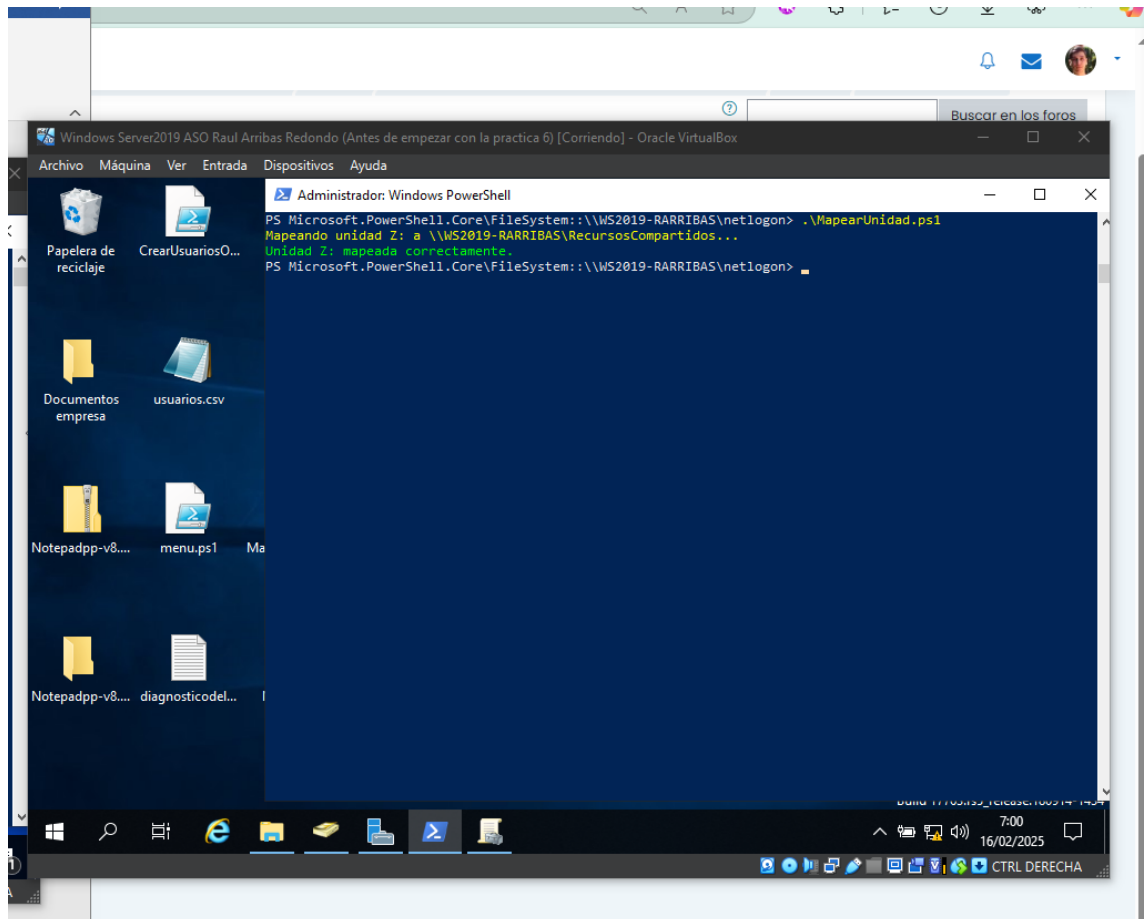
Scripting en sistemas operativos.

```
    Write-Host "Ejecuta '.\CrearUsuariosOU.ps1 --help' para ver la ayuda."  
    exit 1  
}  
}
```

4 Apartado 3 (script cmd o PowerShell- Windows Server)

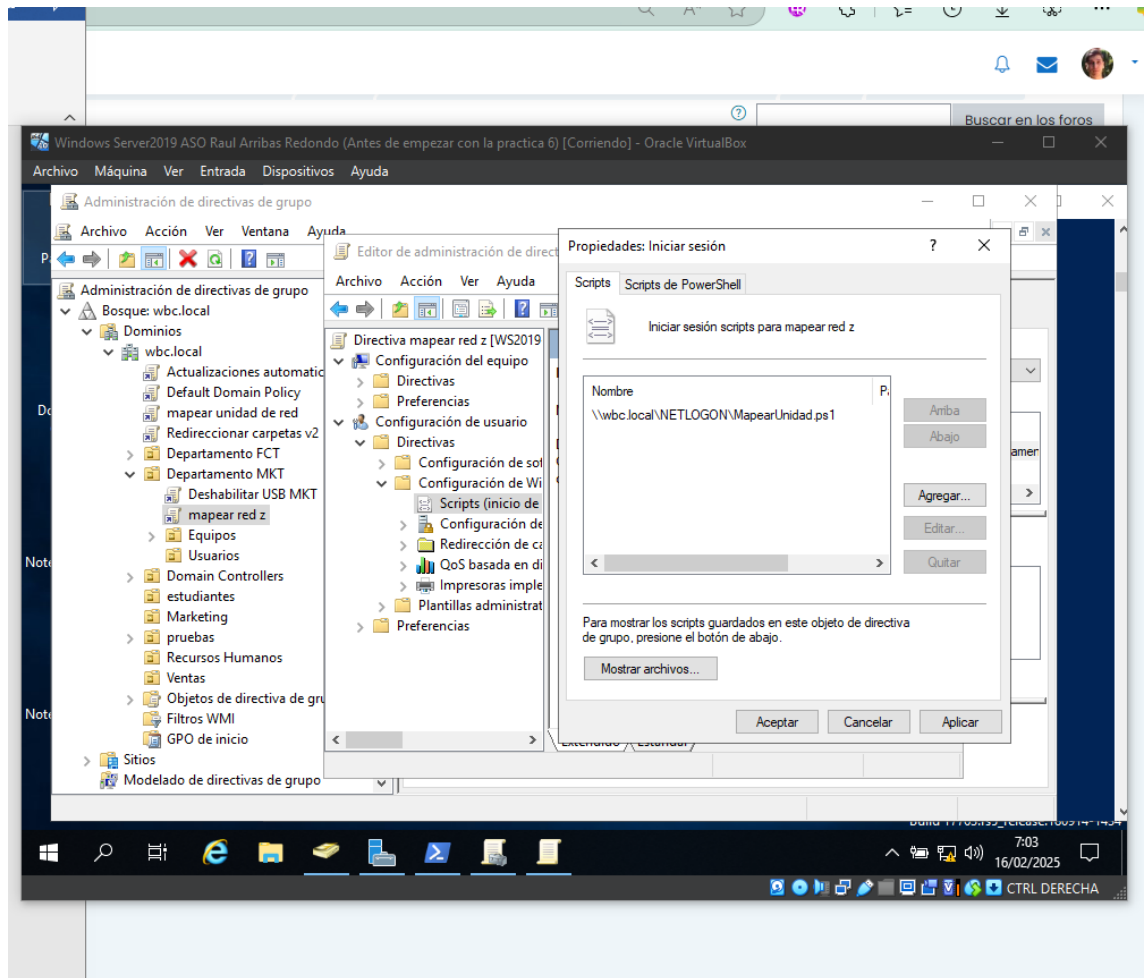
Creación de un script en Windows desde el apartado adecuado para asignar una unidad lógica (unidad de red mapeada) a cada usuario de una Unidad Organizativa (OU) en Windows Server cuando inicien sesión.

Primero probamos el script en el pc administrador:



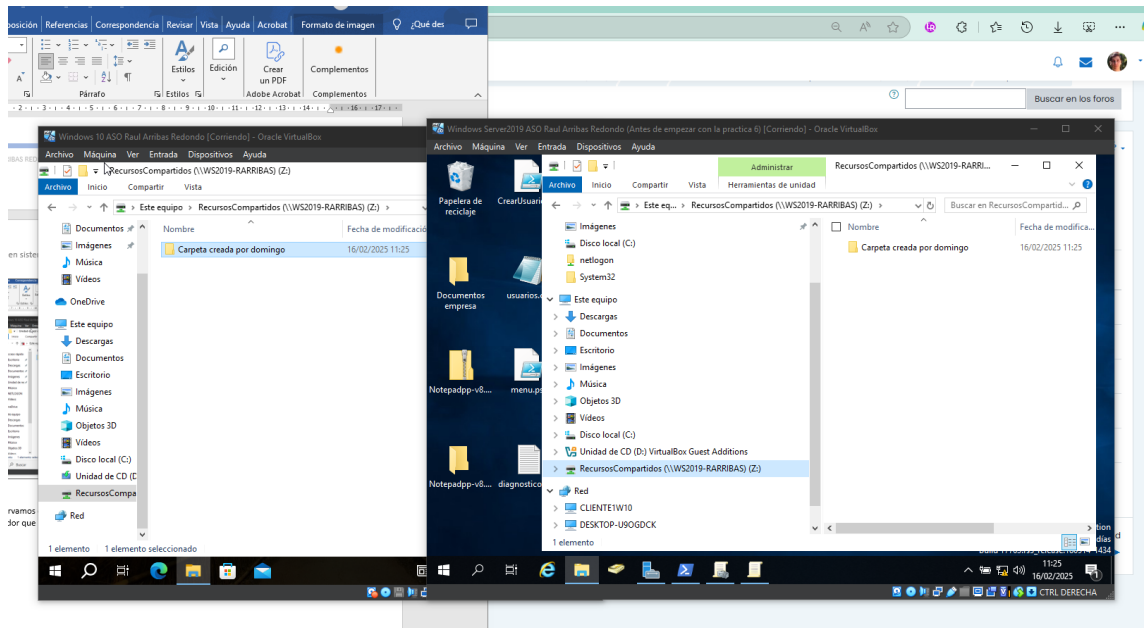
Scripting en sistemas operativos.

Lo siguiente sería crear una política para que cuando un usuario del departamento MKT inicie sesión, le mapee la unidad z, agregamos el script creado:



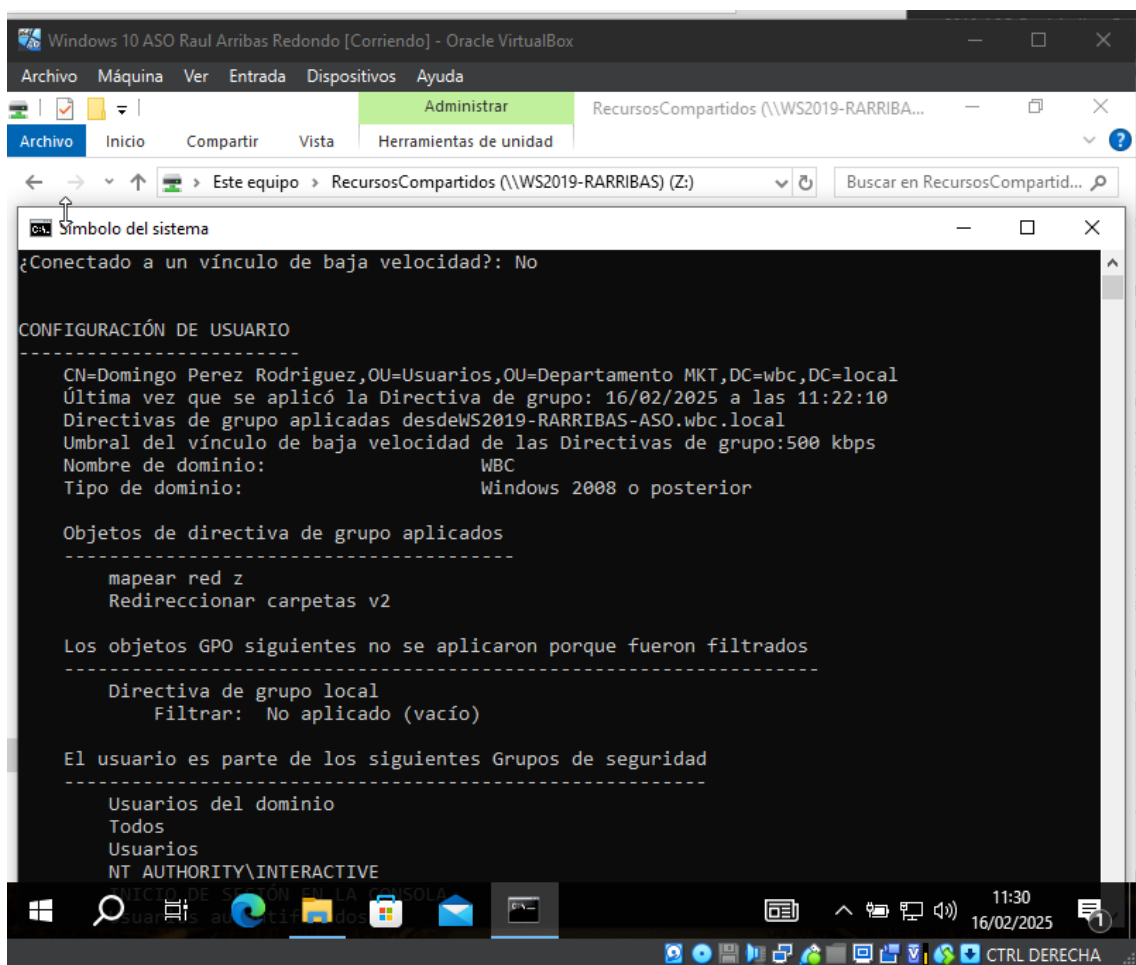
Ahora iniciamos sesión con el usuario del grupo MKT, y hacemos un `gpupdate /force` en caso de que no nos aparezca de inmediato, al cabo de unos minutos debería de aparecernos la carpeta compartida:

Scripting en sistemas operativos.



A la izquierda de la imagen el usuario DOMINGO conectado y se le genera automáticamente la ruta de red.

Si hacemos un `gpresult /r` vemos que se está aplicando la política de mapear la ruta de red Z:



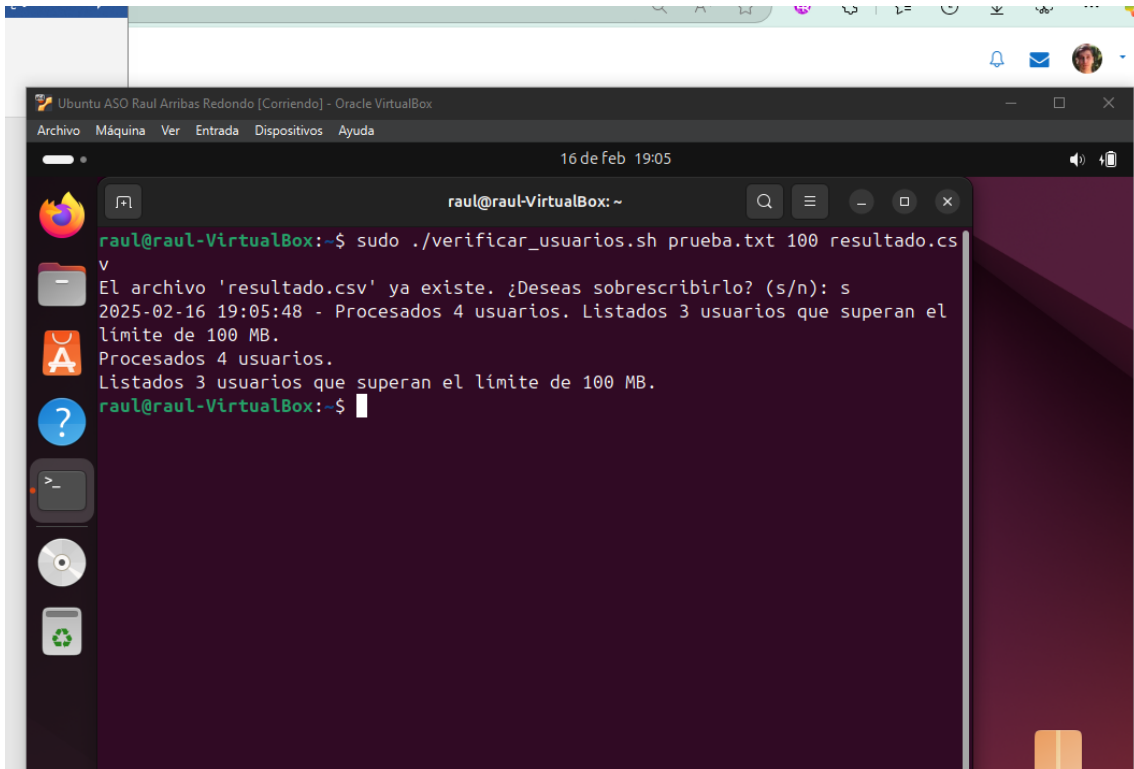
Elegir para desarrollar entre uno de los script siguientes del apartado 4a o del apartado 4b:

Nota: Para que dichos scripts funcionen es necesario que existan en el sistema usuarios con UID > 500; en caso contrario crearemos los suficientes que el script funcione.

5 Apartado 4a (script en bash- GNU/Linux)

- Crear un script que recorra todos los usuarios del sistema cuyo UID sea mayor de 500 (ayúdate del comando awk) y que reciba como primer parámetro el nombre de un fichero, como segundo parámetro un valor numérico en MB y como tercer parámetro el nombre de un fichero CSV a exportar. Si dicho fichero ya existe deberá preguntar si se desea sobrescribir.
- El script deberá comprobar para cada usuario si su directorio personal (/home/usuario) ha superado los Mb indicados en el segundo parámetro, en cuyo caso deberá generar en un fichero CSV que muestre las columnas Usuario,UID,Espacio(MB) de los usuarios que cumplan dicha condición.
- Al finalizar, el script deberá indicar el número de líneas procesadas y usuarios listados. Esta información deberá de guardarse también en un fichero log (historial.log) junto con los posibles errores en su ejecución. Los ficheros log indican fecha y hora en cada línea. (Nota: es posible usar también la utilidad logger de Linux)
- El script sólo podrá ser ejecutado por el usuario root. Si no es así, deberá salir y mostrar un mensaje de error. Además si al script no se le pasan parámetros, deberá mostrar un mensaje de ayuda indicando su sintaxis de uso.

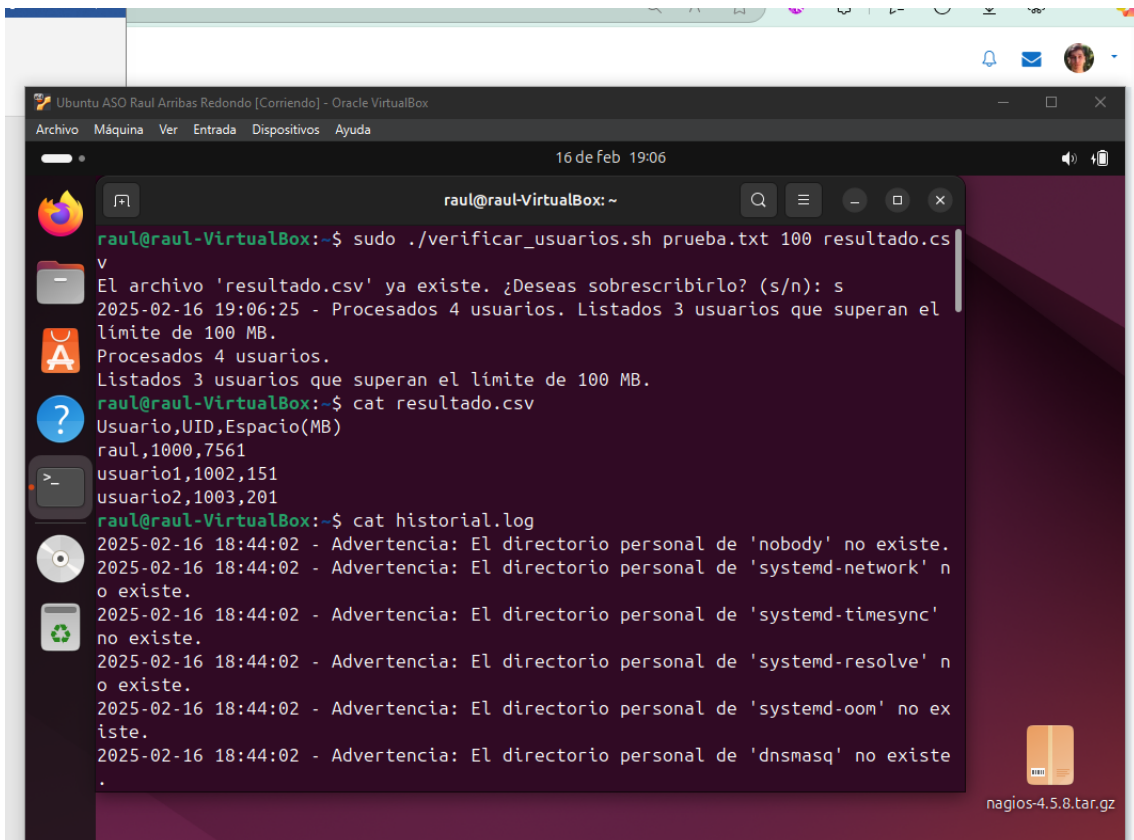
Ejecutamos el script y vemos los resultados:



The screenshot shows a terminal window titled 'raul@raul-VirtualBox: ~' with a search bar and window controls. The terminal output is as follows:

```
raul@raul-VirtualBox:~$ sudo ./verificar_usuarios.sh prueba.txt 100 resultado.csv
v
El archivo 'resultado.csv' ya existe. ¿Deseas sobrescribirlo? (s/n): s
2025-02-16 19:05:48 - Procesados 4 usuarios. Listados 3 usuarios que superan el
límite de 100 MB.
Procesados 4 usuarios.
Listados 3 usuarios que superan el límite de 100 MB.
raul@raul-VirtualBox:~$
```

Podemos ver que hay 3 usuarios que han superado el espacio:



The screenshot shows the same terminal window with additional commands and output:

```
raul@raul-VirtualBox:~$ cat resultado.csv
Usuario,UID,Espacio(MB)
raul,1000,7561
usuario1,1002,151
usuario2,1003,201
raul@raul-VirtualBox:~$ cat historial.log
2025-02-16 18:44:02 - Advertencia: El directorio personal de 'nobody' no existe.
2025-02-16 18:44:02 - Advertencia: El directorio personal de 'systemd-network' n
o existe.
2025-02-16 18:44:02 - Advertencia: El directorio personal de 'systemd-timesync'
no existe.
2025-02-16 18:44:02 - Advertencia: El directorio personal de 'systemd-resolve' n
o existe.
2025-02-16 18:44:02 - Advertencia: El directorio personal de 'systemd-oom' no ex
iste.
2025-02-16 18:44:02 - Advertencia: El directorio personal de 'dnsmasq' no existe
.
```

5.1 Script utilizado:

```
#!/bin/bash
```

```
# Función para mostrar ayuda
```

```
mostrar_ayuda() {  
    echo "Uso: $0 <nombre_fichero> <valor_MB> <nombre_csv>"  
    echo "Descripción: Este script verifica si los usuarios con UID > 500 han superado un límite de espacio en /home."  
    echo "Parámetros:"  
    echo " <nombre_fichero>: Nombre del fichero a verificar."  
    echo " <valor_MB>: Límite de espacio en MB."  
    echo " <nombre_csv>: Nombre del archivo CSV a exportar."  
    exit 1  
}
```

```
# Función para registrar logs
```

```
registrar_log() {  
    local mensaje="$1"  
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $mensaje" | tee -a historial.log  
}
```

```
# Verificar que el script se ejecute como root
```

```
if [[ $EUID -ne 0 ]]; then  
    registrar_log "Error: Este script debe ser ejecutado como root."  
    exit 1  
fi
```

```
# Verificar que se pasen los parámetros necesarios
```

```
if [[ $# -ne 3 ]]; then  
    mostrar_ayuda
```

Scripting en sistemas operativos.

fi

Asignar parámetros

fichero="\$1"

limite_mb="\$2"

csv="\$3"

Validar que el segundo parámetro sea un número

if ! ["\$limite_mb" =~ ^[0-9]+\$]; then

 registrar_log "Error: El segundo parámetro debe ser un número entero."

 exit 1

fi

Verificar si el archivo CSV ya existe

if [-f "\$csv"]; then

 read -p "El archivo '\$csv' ya existe. ¿Deseas sobrescribirlo? (s/n): " respuesta

 if ["\$respuesta" != "s" && "\$respuesta" != "S"]; then

 registrar_log "Operación cancelada por el usuario."

 exit 0

 fi

fi

Inicializar contadores

total_procesados=0

usuarios_listados=0

Crear encabezado del archivo CSV

echo "Usuario,UID,Espacio(MB)" > "\$csv"

Obtener usuarios con UID > 500 y procesar sus directorios

while read usuario; do

Scripting en sistemas operativos.

```
home_dir="/home/$usuario"

# Ignorar si el directorio personal no existe
if [[ ! -d "$home_dir" ]]; then
    continue # Saltar al siguiente usuario sin generar advertencias
fi

total_procesados=$((total_procesados + 1))

# Calcular el tamaño del directorio en MB
espacio=$(du -sm "$home_dir" 2>/dev/null | awk '{print $1}')
if [[ -z "$espacio" ]]; then
    continue # Saltar si no se puede calcular el espacio
fi

if [[ "$espacio" -gt "$limite_mb" ]]; then
    uid=$(id -u "$usuario")
    echo "$usuario,$uid,$espacio" >> "$csv"
    usuarios_listados=$((usuarios_listados + 1))
fi

done < <(awk -F: '$3 > 500 {print $1}' /etc/passwd)

# Registrar resumen en el log
registrar_log "Procesados $total_procesados usuarios. Listados $usuarios_listados usuarios que
superan el límite de $limite_mb MB."

# Mostrar resumen en la consola
echo "Procesados $total_procesados usuarios."
echo "Listados $usuarios_listados usuarios que superan el límite de $limite_mb MB."
```