Automatización segura con PowerShell en entornos empresariales

1. **Configuración segura de políticas de ejecución.**
    - Configura las políticas de ejecución en PowerShell para permitir únicamente la ejecución de scripts firmados por la organización y bloquear cualquier otro tipo de script.

    - Primero verificamos la política actual, lo que mostrará las políticas en los distintos niveles.

```
PS C:\Windows\system32> Get-ExecutionPolicy -List

        Scope ExecutionPolicy
        ----- ---------------
MachinePolicy       Undefined
   UserPolicy       Undefined
      Process       Undefined
  CurrentUser       Undefined
 LocalMachine    RemoteSigned
```

    - Luego configuramos la política segura, en el cual definimos ALLSigned en nuestra LocalMachine

```
PS C:\Windows\system32> Set-ExecutionPolicy -ExecutionPolicy AllSigned -Scope LocalMachine

Cambio de directiva de ejecución
La directiva de ejecución te ayuda a protegerte de scripts en los que no confías. Si cambias dicha
directiva, podrías exponerte a los riesgos de seguridad descritos en el tema de la Ayuda
about_Execution_Policies en https:/go.microsoft.com/fwlink/?LinkID=135170. ¿Quieres cambiar la
directiva de ejecución?
[S] Sí  [O] Sí a todo  [N] No  [T] No a todo  [U] Suspender  [?] Ayuda
(el valor predeterminado es "N"):s
```

    - Por último, volvemos a verificar la configuración y ahora debería mostrar “LocalMachine: ALLSigned”.

```
PS C:\Windows\system32> Get-ExecutionPolicy -List

        Scope ExecutionPolicy
        ----- ---------------
MachinePolicy       Undefined
   UserPolicy       Undefined
      Process       Undefined
  CurrentUser       Undefined
 LocalMachine       AllSigned
```

    - Creamos el certificado de firma de código.

```
PS Z:\Backups> $Cert = New-SelfSignedCertificate -CertStoreLocation Cert:\LocalMachine\my -Subject "
CN=FirmaScript"
```

    - confirmamos que se creó el certificado de script.

```
PS Z:\Backups> Get-ChildItem Cert:\LocalMachine\my

   PSParentPath: Microsoft.PowerShell.Security\Certificate::LocalMachine\my

Thumbprint                                Subject
----------                                -------
E391CB11FFCBF68F2E077D0C070AAD53831D5061  CN=FirmaScript
```

- firmamos el script que creamos

```
PS Z:\Backups> Set-AuthenticodeSignature -FilePath "Z:\Backups\Backup-Seguro.ps1" -Certificate $Cert


    Directorio: Z:\Backups

SignerCertificate                         Status                 Path
-----------------                         ------                 ----
047BFF9BD2D91FC3B7FC42EB7CE9E0075028C169  Valid                  Backup-Seguro.ps1
```

- por último, confirmamos que está validado y firmado

```
PS Z:\Backups> Get-AuthenticodeSignature "Z:\Backups\Backup-Seguro.ps1"

    Directorio: Z:\Backups

SignerCertificate                         Status                 Path
-----------------                         ------                 ----
047BFF9BD2D91FC3B7FC42EB7CE9E0075028C169  Valid                  Backup-Seguro.ps1
```

2. Desarrollo de un Script de automatización de respaldo.

Crea un script en PowerShell que realice las siguientes acciones:

- Realice un respaldo incremental de una carpeta con información crítica.

- Configuramos nuestros parámetros y creamos un respaldo.

```
# Parámetros de configuración
$Origen = "Z:\respaldo"  # Carpeta origen (compartida)
$Destino = "Z:\NuevoRespaldo"  # Cambia por ruta segura real
$FechaHora = Get-Date -Format "yyyyMMdd_HHmmss"
$BackupName = "Respaldo_$FechaHora.zip"
$BackupTemporal = "Z:\Temp\$BackupName"

# Crear carpetas si no existen
New-Item -Path "Z:\Temp" -ItemType Directory -Force
New-Item -Path $Destino -ItemType Directory -Force

# Crear respaldo incremental
function Hacer-Respaldo {
    try {
        Write-Host "Iniciando respaldo..."
```

- Comprima el respaldo en un archivo .zip con un nombre que incluya la fecha y hora del respaldo.

- Comprimimos nuestro archivo a .zip

```
        # Comprimir la carpeta
        Compress-Archive -Path $Origen -DestinationPath $BackupTemporal -
Update
        Write-Host "Archivo comprimido en $BackupTemporal"
    }
    catch {
        throw "Error al comprimir: $_"
    }
}
```

- Mueva el archivo .zip a una ubicación segura en la red corporativa.

- Movemos nuestro archivo .zip

```
# Mover respaldo a red
function Mover-Respaldo {
    try {
        Write-Host "Moviendo respaldo a red..."
        Move-Item -Path $BackupTemporal -Destination $Destino
        Write-Host "Respaldo movido a $Destino\$BackupName"
    }
    catch {
        throw "Error al mover respaldo: $_"
    }
}
```

- Envíe una notificación por correo electrónico al equipo de TI informando el estado del respaldo (éxito o error).

- enviamos la notificación por correo con el siguiente script

```powershell
# Enviar notificación por correo
function Enviar-Notificacion {
    param($Estado)

    # Datos del remitente y destinatario de prueba
    $CorreoOrigen = "matiassaavedragaj@gmail.com"
    $CorreoDestino = "matiassaavedragaj@gmail.com" # Puede ser el mismo para pruebas
    $SMTP = "smtp.gmail.com"
    $Puerto = 587
    $Credenciales = Get-Credential  # Se recomienda usar credencial segura

    $Asunto = "Estado del respaldo: $Estado"
    $Cuerpo = "El respaldo se realizo con estado: $Estado. Fecha: $FechaHora"

    Send-MailMessage `
        -From $CorreoOrigen `
        -To $CorreoDestino `
        -Subject $Asunto `
        -Body $Cuerpo `
        -SmtpServer $SMTP `
        -Port $Puerto `
        -UseSsl `
        -Credential $Credenciales
}

# MAIN
try {
    Hacer-Respaldo
    Mover-Respaldo
    Enviar-Notificacion -Estado "EXITO"
}
catch {
    Write-Error $_
    Enviar-Notificacion -Estado "ERROR"
}
```

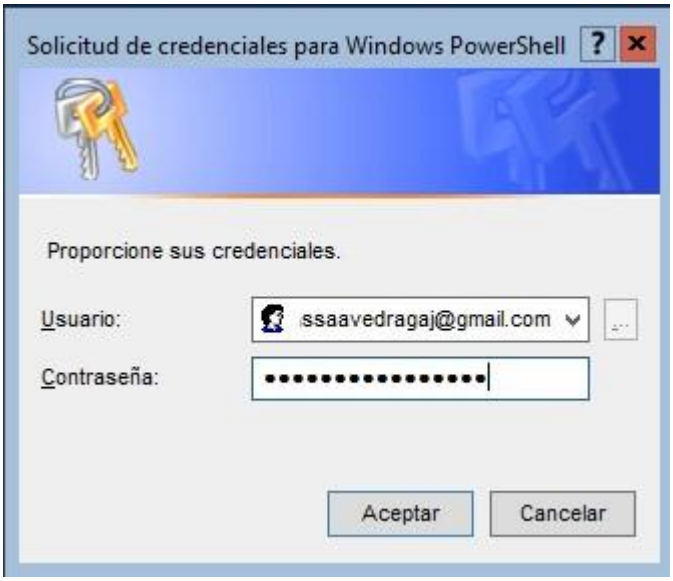
- recordar que todos los scripts anteriores están en un solo archivo .ps1

- ejecución del script

```
PS Z:\> & "Z:\Backup-Seguro.ps1"


    Directorio: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        15-07-2025     21:59                Temp


    Directorio: \\VBOXSVR\Backups


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        15-07-2025      0:32                NuevoRespaldo
Iniciando respaldo...
Archivo comprimido en C:\Temp\Respaldo_20250715_215948.zip
Moviendo respaldo a red...
```

```
Respaldo movido a Z:\NuevoRespaldo\Respaldo_20250715_215948.zip

cmdlet Get-Credential en la posición 1 de la canalización de comandos
Proporcione valores para los parámetros siguientes:
Credential
```

- validación del correo



- confirmación de llegada de correo

3. **Uso de Cmdlets avanzados y buenas prácticas de Scripting.**

   Asegúrate de que el script:

- Utilice cmdlets avanzados como Send-MailMessage, Compress-Archive, GetDate, entre otros.
- Incluya comentarios detallados explicando cada sección del código.
- Implemente manejo de errores robusto utilizando Try...Catch.
- Estructure el código de manera modular mediante funciones reutilizables.

```
# Parámetros de configuración
$Origen = "Z:\respaldo"  # Carpeta origen (compartida)
$Destino = "Z:\NuevoRespaldo"  # Cambia por ruta segura real
$FechaHora = Get-Date -Format "yyyyMMdd_HHmmss"
$BackupName = "Respaldo_$FechaHora.zip"
$BackupTemporal = "Z:\Temp\$BackupName"
```

```
# Crear carpetas si no existen
New-Item -Path "Z:\Temp" -ItemType Directory -Force
New-Item -Path $Destino -ItemType Directory -Force
```

```
# Crear respaldo incremental
function Hacer-Respaldo {
    try {
        Write-Host "Iniciando respaldo..."
```

- En la primera imagen se muestran los parametros de la configuracion como tambien se verifica si existe la carpeta Temp y al no haber la crea y se procede a realizar el respaldo.

```
        # Comprimir la carpeta
        Compress-Archive -Path $Origen -DestinationPath $BackupTemporal -
Update
        Write-Host "Archivo comprimido en $BackupTemporal"
    }
    catch {
        throw "Error al comprimir: $_"
    }
}
```

- En la Segunda Imagen se puede preciar como se compime la carpeta y la lleva en la ruta z:\temp.

```
# Mover respaldo a red
function Mover-Respaldo {
    try {
        Write-Host "Moviendo respaldo a red..."
        Move-Item -Path $BackupTemporal -Destination $Destino
        Write-Host "Respaldo movido a $Destino\$BackupName"
    }
    catch {
        throw "Error al mover respaldo: $_"
    }
}
```

- Luego de que la carpeta se haya comprimido se procede a mover el respaldo a la red y en caso de que no ser asi se imprime el error.

```
# Enviar notificación por correo
function Enviar-Notificacion {
    param($Estado)

    # Datos del remitente y destinatario de prueba
    $CorreoOrigen = "matiassaavedragaj@gmail.com"
    $CorreoDestino = "matiassaavedragaj@gmail.com" # Puede ser el mismo para pruebas
    $SMTP = "smtp.gmail.com"
    $Puerto = 587
    $Credenciales = Get-Credential  # Se recomienda usar credencial segura

    $Asunto = "Estado del respaldo: $Estado"
    $Cuerpo = "El respaldo se realizo con estado: $Estado. Fecha: $FechaHora"

    Send-MailMessage `
        -From $CorreoOrigen `
        -To $CorreoDestino `
        -Subject $Asunto `
        -Body $Cuerpo `
        -SmtpServer $SMTP `
        -Port $Puerto `
        -UseSsl `
        -Credential $Credenciales
}

# MAIN
try {
    Hacer-Respaldo
    Mover-Respaldo
    Enviar-Notificacion -Estado "EXITO"
}
catch {
    Write-Error $_
    Enviar-Notificacion -Estado "ERROR"
}
```

- En el primer recuadro podemos visualizar la funcion Enviarnotificacion Asignandole un parametro como a su vez el segundo recuadro muestra como se le asignan datos a las variables como tal es el caso de correo , de origen y de destino y a su vez se le adjunta que se ha entregado con éxito o error dependiendo del caso, el main es donde se le pasa la funcion para que sea realizada y luego la retorna ya concluida.