Ejercicios de PowerShell: realizar operaciones en un equipo remoto

```
cls
write-host "Ip del ordenador remoto"
$ipR=read-host

##Client
$port=2020
$endpoint    =    new-object    System.Net.IPEndPoint
([IPAddress]$ipR,$port)
$udpclient=new-Object System.Net.Sockets.UdpClient
$b=Invoke-Expression                              -Command
"[Text.Encoding]::ASCII.GetBytes('cls')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
$udpclient.Close()

while (1)
{

cls

Write-Host "Seleccione una opcion"

echo "1 - Sistema de archivos"
echo "2 - Agregar/eliminar software"
echo "3 - Actualizacion"
echo "4 - Gestion de procesos"
echo "5 - Programacion de tareas"
echo "6 - Gestion de usuarios"
echo "7 - Gestion de almacenamiento"
echo "8 - Gestion de red"
echo "9 - Copias de seguridad"
echo "10 - Reparacion del sistema"
echo "11 - Rendimiento del sistema"
echo "12 - Salir"
```

```
$prue1 = Read-Host
switch ( $prue1 )
    {
        1{
            cls
            Write-Host "Seleccione una opcion"
            echo "1 - Listar"
            echo "2 - Crear"
            echo "3 - Abrir"
            echo "4 - Eliminar"
            $selcec=Read-Host
            switch ( $selcec )
            {
                1{
                    ##Client
                    $port=2020
                                        $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                          $udpclient=new-Object
System.Net.Sockets.UdpClient
                     $b=[Text.Encoding]::ASCII.GetBytes('Get-
ChildItem')
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                    $udpclient.Close()
                    Pause
                }
                2{
                     $type=Read-Host "Escriba que tipo desea
crear (File o Directory)"
                      $name=Read-Host "Escriba el nombre del
$type"
                    ##Client
                    $port=2020
                                        $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                          $udpclient=new-Object
System.Net.Sockets.UdpClient
                                $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('New-Item -ItemType $type -
Name $name')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
```

```
                    $udpclient.Close()
                    Pause
                }
                3{
                     $name1=Read-Host "Nombre del archivo que
desea abrir"
                    ##Client
                    $port=2020
                                    $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                      $udpclient=new-Object
System.Net.Sockets.UdpClient
                              $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('start $name1')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                    $udpclient.Close()
                    Pause
                }
                4{
                    ls
                    $name2=Read-Host "¿Qué desea eliminar?"
                    ##Client
                    $port=2020
                                    $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                      $udpclient=new-Object
System.Net.Sockets.UdpClient
                              $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('Remove-Item $name2')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                    $udpclient.Close()
                    Pause
                }
            }
        }
        2{
            cls
             $nombre=Read-Host "Introduzca nombre del .msi que
desee instalar"
            ##Client
            $port=2020
```

```
              $endpoint = new-object System.Net.IPEndPoint
([IPAddress]$ipR,$port)
            $udpclient=new-Object System.Net.Sockets.UdpClient
                        $b=Invoke-Expression  -Command
"[Text.Encoding]::ASCII.GetBytes('msiexec.exe    /package
C:\windows\Installer\($nombre)')"
            $bytesSent=$udpclient.Send($b,$b.length,$endpoint)
            $udpclient.Close()
        Pause
        }
        3{
            cls
            echo "1 - Instalar"
            echo "2 - Desinstalar"
            Write-Host "Seleccione una opción"
            $actuali=Read-Host
            switch ($actuali)
            {
                1{
                    Write-Host "Como desea que se realize"
                    echo "1 - Normal"
                    echo "2 - En silencio"
                    $modo=Read-Host
                    switch ($modo)
                    {
                        1{
                             Write-Host "Escriba el nombre de
la actualizacion"
                            $nombre=Read-Host
                            ##Client
                            $port=2020
                                   $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                    $udpclient=new-Object
System.Net.Sockets.UdpClient
                                  $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('wusa $nombre')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                            $udpclient.Close()
                        }
                        2{
```

```
                            Write-Host "Eliga la opción desea"
                              echo "1 - No reiniciarse al
terminar"
                            echo "2 - Pedir confirmación antes
de reiniciar"
                            echo "3 - Obligar a que se cierren
todas las aplicaciones y se reinicie"
                            $Smodo=Read-Host
                            $Nombre=Read-Host
                            switch($Smodo)
                            {
                                1{
                                    ##Client
                                    $port=2020
                                      $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                       $udpclient=new-Object
System.Net.Sockets.UdpClient
                                       $b=Invoke-Expression -
Command "[Text.Encoding]::ASCII.GetBytes('wusa $nombre /quiet
/norestart')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                                    $udpclient.Close()
                                }
                                2{
                                    ##Client
                                    $port=2020
                                      $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                       $udpclient=new-Object
System.Net.Sockets.UdpClient
                                       $b=Invoke-Expression -
Command "[Text.Encoding]::ASCII.GetBytes('wusa $nombre /quiet
/warnrestart')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                                    $udpclient.Close()
                                }
                                3{
                                    ##Client
                                    $port=2020
                                      $endpoint = new-object
```

```
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                    $udpclient=new-Object
System.Net.Sockets.UdpClient
                                   $b=Invoke-Expression -
Command "[Text.Encoding]::ASCII.GetBytes('wusa $nombre /quiet
/forcerestart')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                                       $udpclient.Close()
                                  }
                              }
                          }
                      }
                  }
                  2{
                      cls
                      echo "1 - Normal"
                       echo "2 - Paquete asociado con el número
de actualización"
                      Write-Host "Seleccione una opción"
                      $modo=Read-Host
                      switch ($modo)
                      {
                          1{
                              ##Client
                              $port=2020
                                     $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                      $udpclient=new-Object
System.Net.Sockets.UdpClient
                                $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('wusa $nombre /unistall')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                              $udpclient.Close()
                          }
                          2{
                              ##Client
                              $port=2020
                                     $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                      $udpclient=new-Object
System.Net.Sockets.UdpClient
```

```
                                $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('wusa  $nombre  /unistall
/kb')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                                 $udpclient.Close()
                            }
                        }
                }
                }
                puase
        }
        4{
            cls
            echo "1 - Iniciar un proceso"
            echo "2 - Mostrar solo Nombres"
            echo "3 - Mostrar proceso de cierto programa"

            $sele=Read-Host
            switch ( $sele )
            {
                1{
                    ##Client
                    $port=2020
                                        $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                          $udpclient=new-Object
System.Net.Sockets.UdpClient
                                $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('wusa  $nombre  /quiet
/norestart')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                    $udpclient.Close()
                }
                2 {
                    ##Client
                    $port=2020
                                        $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                          $udpclient=new-Object
System.Net.Sockets.UdpClient
                                $b=Invoke-Expression -Command
```

```
"[Text.Encoding]::ASCII.GetBytes('Get-Process | select Name')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                    $udpclient.Close()
                    Pause
                }
                3 {
                     $selprog=Read-Host "Escriba el nombre del
programa"
                    ##Client
                    $port=2020
                                    $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                      $udpclient=new-Object
System.Net.Sockets.UdpClient
                             $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('Get-Process | Select-String
$selprog')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                    $udpclient.Close()
                    Pause
                }
            }
        }
        5{
            cls
            $program=Read-Host "que programa desea abrir"
            $time=Read-Host "a que hora"
            $nomre=Read-Host "escriba el nombre de la tarea"
            $action=New-ScheduledTaskAction -Execute $program
            $trigger=New-ScheduledTaskTrigger -Daily -At $time
            ##Client
            $port=2020
              $endpoint = new-object System.Net.IPEndPoint
([IPAddress]$ipR,$port)
            $udpclient=new-Object System.Net.Sockets.UdpClient
                          $b=Invoke-Expression  -Command
"[Text.Encoding]::ASCII.GetBytes('Register-ScheduledTask  -
Action $action -Trigger $trigger -TaskName $nomre')"
            $bytesSent=$udpclient.Send($b,$b.length,$endpoint)
            $udpclient.Close()
            Pause
```

```
        }
        6{
            cls
            echo "1 - Crear usuario"
            echo "2 - Eliminar usuario"
            echo "3 - Crear grupo"
            echo "4 - Ver miembros de un grupo"
            echo "5 - Eliminar grupo"
            $opcion=Read-Host "Eliga una opcion"
            switch ($opcion)
            {
            1{
                $nameuser=Read-Host "Nomre del nuevo usuario"
                $pass=Read-Host "Introduzca contraseña"
                   $pass1=ConvertTo-SecureString $pass -
asplaintext -force
                ##Client
                $port=2020
                 $endpoint = new-object System.Net.IPEndPoint
([IPAddress]$ipR,$port)
                                    $udpclient=new-Object
System.Net.Sockets.UdpClient
                           $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('New-LocalUser $nameuser -
Password $pass1')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                $udpclient.Close()
            }
            2{
                $usuer=Read-Host "Nombre del usuario"
                ##Client
                $port=2020
                 $endpoint = new-object System.Net.IPEndPoint
([IPAddress]$ipR,$port)
                                    $udpclient=new-Object
System.Net.Sockets.UdpClient
                           $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('Remove-LocalUser   -Name
$usuer')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                $udpclient.Close()
```

```
            }
            3{
                $group=Read-Host "Nombre para el grupo"
                ##Client
                $port=2020
                 $endpoint = new-object System.Net.IPEndPoint
([IPAddress]$ipR,$port)
                                    $udpclient=new-Object
System.Net.Sockets.UdpClient
                            $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('New-LocalGroup    -Name
$group')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                $udpclient.Close()
            }
            4{
                $Ngroup=Read-Host "Nombre del grupo"
                ##Client
                $port=2020
                 $endpoint = new-object System.Net.IPEndPoint
([IPAddress]$ipR,$port)
                                    $udpclient=new-Object
System.Net.Sockets.UdpClient
                            $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('Get-LocalGroupMember  -Name
$Ngroup')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                $udpclient.Close()
            }
            5{
                $Rgroup=Read-Host "Nombre del grupo que vaya a
eliminar"
                ##Client
                $port=2020
                 $endpoint = new-object System.Net.IPEndPoint
([IPAddress]$ipR,$port)
                                    $udpclient=new-Object
System.Net.Sockets.UdpClient
                            $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('Get-LocalGroupMember  -Name
$Rgroup')"
```

```
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                $udpclient.Close()
            }
            }
        }
        7{
            cls
            echo "1 - Particionar"
            echo "2 - Desfragmentar"
            echo "3 - Convertir sistemas de archivo"
            echo "4 - Cifrar"
            echo "5 - Descifrar"
            Write-Host  "Seleccione una opción"
            $almacenamiento=Read-Host
            switch ($almacenamiento)
            {
                1{
                     Write-Host "ADVERTENCIA: Nunca poner el
numero '0'"
                    $NumeroD=Read-Host "Introduzca número del
disco"
                   ##Client
                   $port=2020
                                    $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                      $udpclient=new-Object
System.Net.Sockets.UdpClient
                              $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('New-Partition  -DiskNumber
$NumeroD -UseMaximumSize -AssignDriveLetter')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                   $udpclient.Close()
                   Pause
                }
                2{
                    $letra=Read-Host "Letra del disco"
                    ##Client
                    $port=2020
                                    $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                      $udpclient=new-Object
```

```
System.Net.Sockets.UdpClient
                        $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('Optimize-Volume  $letra  -
Defrag')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                    $udpclient.Close()
                    Disable-BitLocker
                    Pause
                }
                3{
                    $letra=Read-Host "Letra del disco"
                    $tipo=Read-Host "Que formato desea"
                    ##Client
                    $port=2020
                                $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                  $udpclient=new-Object
System.Net.Sockets.UdpClient
                        $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('Format-Volume  -DriveLetter
$letra -FileSystem $tipo')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                    $udpclient.Close()
                    Pause
                }
                4{
                      $letra=Read-Host "Inroduzca letra del
disco"
                    ##Client
                    $port=2020
                                $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                  $udpclient=new-Object
System.Net.Sockets.UdpClient
                        $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('Enable-BitLocker -MountPoint
$letra -RecoveryPasswordProtector -UsedSpaceOnly -Verbose')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                    $udpclient.Close()
                    Pause
                }
```

```
                5{
                      $letra=Read-Host "Introduzca letra del
disco"
                    ##Client
                    $port=2020
                                        $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                          $udpclient=new-Object
System.Net.Sockets.UdpClient
                              $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('Disable-BitLocker    -
MountPoint $letra')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                    $udpclient.Close()
            }
        }
        }
        8{
            cls
            echo "1 - Saber ip"
            echo "2 - Hacer ping"
            $op=Read-Host "Seleccione una opcion"
            Switch ( $op )
            {
            1{
                ##Client
                $port=2020
                 $endpoint = new-object System.Net.IPEndPoint
([IPAddress]$ipR,$port)
                                          $udpclient=new-Object
System.Net.Sockets.UdpClient
                              $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('ipconfig')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                $udpclient.Close()
                Pause
            }
            2{
                $ip=Read-Host "Introduzca IPv4"
                ##Client
                $port=2020
```

```
                $endpoint = new-object System.Net.IPEndPoint
([IPAddress]$ipR,$port)
                                    $udpclient=new-Object
System.Net.Sockets.UdpClient
                        $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('ping $ip')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                $udpclient.Close()
                Pause
            }
            }
        }
        9{
            cls
            Write-Host "Se recomienda que ponga la ruta y para
quedarsela usted la meta en una carpeta compartida"
             $namea=Read-Host "Nombre del archivo que desea
hacer copia de seguridad"
             $ruta=Read-Host "Ruta donde desea que este la
copia de seguridad"
            ##Client
            $port=2020
              $endpoint = new-object System.Net.IPEndPoint
([IPAddress]$ipR,$port)
            $udpclient=new-Object System.Net.Sockets.UdpClient
                        $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('Copy-Item $namea $ruta')"
            $bytesSent=$udpclient.Send($b,$b.length,$endpoint)
            $udpclient.Close()
            Pause
        }
        10{
            cls
            Write-Host "Seleccione una opción"
            echo "1 - Ver puntos de restauracion"
            echo "2 - Restaurar un punto"
            $opcion=Read-Host
            switch ($opcion)
            {
                1{
                    ##Client
```

```
                $port=2020
                                $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                  $udpclient=new-Object
System.Net.Sockets.UdpClient
                            $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('Get-ComputerRestorePoint')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                $udpclient.Close()
            }
            2{
                Write-Host "Seleccione una opción"
                echo "1 - Ultimo punto"
                echo "2 - Restaura cierto punto"
                $opcion=Read-Host
                switch ($opcion)
                {
                    1{
                        ##Client
                        $port=2020
                                $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                  $udpclient=new-Object
System.Net.Sockets.UdpClient
                              $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('Get-ComputerRestorePoint  -
LastStatus')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                        $udpclient.Close()
                    }

                    2{
                          Write-Host "Ponga el numero de
restauración"
                        $punto=Read-Host
                        ##Client
                        $port=2020
                                $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                  $udpclient=new-Object
System.Net.Sockets.UdpClient
```

```
                                $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('Restore-Computer    -
RestorePoint $punto')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                                $udpclient.Close()
                            }
                        }
                    }
                }
                Pause
            }
            11{
                cls
                echo "1 - Monitorizar procesos"
                echo "2 - Monitorizar procesos Avanzado"
                 echo "3 - Ver todos los nombres de los contadores
de rendimiento"
                $seleccionar=Read-Host "Seleccione una opción"
                switch ( $seleccionar )
                {
                    1{
                        ##Client
                        $port=2020
                                        $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port
                        )
                                          $udpclient=new-Object
System.Net.Sockets.UdpClient
                                    $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('Get-Process')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                        $udpclient.Close()
                        Pause
                    }
                    2{
                        ##Client
                        $port=2020
                                        $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                          $udpclient=new-Object
System.Net.Sockets.UdpClient
```

```
                        $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('Get-WmiObject    -Class
win32_process')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                    $udpclient.Close()
                    Pause
                }
                3{
                    ##Client
                    $port=2020
                                $endpoint = new-object
System.Net.IPEndPoint ([IPAddress]$ipR,$port)
                                  $udpclient=new-Object
System.Net.Sockets.UdpClient
                        $b=Invoke-Expression -Command
"[Text.Encoding]::ASCII.GetBytes('Get-Counter -ListSet * |
Sort-Object CounterSetName | Format-Table CounterSetName')"
$bytesSent=$udpclient.Send($b,$b.length,$endpoint)
                    $udpclient.Close()
                    Pause
                }
            }
        }
}
if( $prue1 -eq "12" )
        {
            cls
            ##Client
            $port=2020
              $endpoint = new-object System.Net.IPEndPoint
([IPAddress]$ipR,$port)
            $udpclient=new-Object System.Net.Sockets.UdpClient
            $b=[Text.Encoding]::ASCII.GetBytes('break')
            $bytesSent=$udpclient.Send($b,$b.length,$endpoint)
            $udpclient.Close()
            break
        }
}
```