INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA RIO GRANDE DO NORTE

**Professor: Macêdo Firmino**
**Disciplina: Administração de Sistemas Proprietários**
**Aula 18: Configurações de Rede com Script em PowerShell.**

---

Olá, meu queridos!! Tudo bem??? Na aula de hoje iremos realizar uma experiência prática na automação de tarefas de configuração de rede, utilizando os recursos do PowerShell para simplificar e efetuar mudanças na infraestrutura de rede. Vamos lá!!! Preparados???

## Configurações TCP/IP

O TCP/IP é um conjunto de protocolos padrão criado para permitir as comunicações na Internet. Os elementos básicos de configuração do TCP/IP são:

- Endereço IP: é uma *string* de identificação única de 32 *bits* (no IPv4);
- Máscara de Subrede (ou simplesmente netmask): é uma máscara de 32 *bits*. Este endereço separam porções de endereços IPs relacionados à uma rede de uma subrede. Através da máscara é possível determinar o endereço da rede e o endereço de *broadcast*.
- *Gateway*: é o endereço IP de *host* ou roteador na rede. Este equipamento vai habilitar o tráfico de informações de uma rede interna para outras redes externas, tais como Internet.
- Endereço de Servidores de Nomes (DNS): representam os endereços IP do servidor que convertem os nomes dos *hosts* da rede para seus respectivos endereços IP.

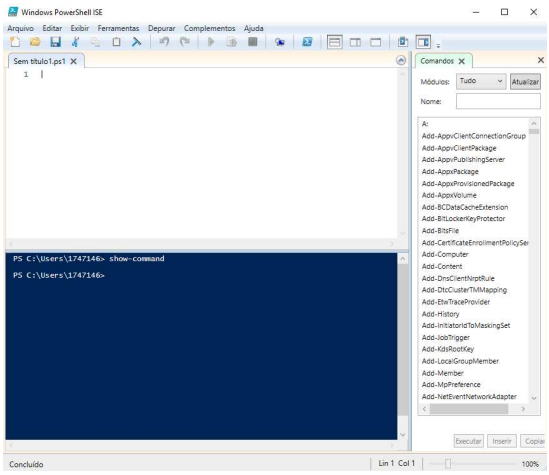
## PowerShell

O PowerShell é um shell desenvolvida pela Microsoft. Ele foi projetado para ser uma ferramenta poderosa de automação de tarefas e gerenciamento de sistemas, especialmente em ambientes Windows.

Para trabalharmos com o Powershell podemos utilizar o Shell diretamente (PS) ou o Powershell ISE. O PowerShell ISE (*Integrated Scripting Environment*), um ambiente de programação do PowerShell que facilita o desenvolvimento de *scripts*. Nele você pode executar comandos, gravar, testar e depurar *scripts* em uma interface de usuário gráfica baseada no Windows.

Para iniciar PS ISE:

- Clique em “Iniciar”, digite “ISE” e clique em “ISE do Windows PowerShell”. Ou no menu “Iniciar”, clique em “Todos os Programas”, “Acessórios”, clique na pasta “Windows PowerShell” e clique em “ISE do Windows PowerShell”.




**PowerShell Script**

É possível criar scripts PowerShell para posteriormente executa-lo. A vantagem de fazer uso de scripts é criarmos ferramentas de administração ou de automação de tarefas de gerenciamento.

Para criar um script inicial, o famoso “Hello Word” você deve utilizar um editor de texto ou o próprio PowerShell ISE que é o mais recomendado. A extensão para execução de scripts no PowerShell é “.PS1”.

```
Write-Output “Olá’ Turma de Redes!”
```

Salve o arquivo e execute o mesmo no PowerShell ISE.

No PowerShell podemos armazenar em variáveis: comandos, resultados de comandos e expressões, como nomes, caminhos, configurações e valores. O PowerShell detecta automaticamente o tipo da variável e a converte no tipo apropriado. As variáveis do PowerShell, ao contrário de outras linguagens, não diferenciam maiúsculas de minúsculas e podem incluir espaços e caracteres especiais.

Quando usada no PowerShell, uma variável é sempre especificada usando o caractere $ seguido pelo nome da variável.

```
$saida = Get-Location
Write-Output $saida
```

Podemos utilizar o cmdlet Read.Host para receber a entrada de dados e armazenar em forma de variável para ser utilizada na estrutura do seu script. Por exemplo:

```
$nome = Read-Host “Qual o seu nome ?”
Write-Output “Oi!, $nome tudo bem?”
```

## Operadores Condicionais e Lógicos

Assim como em outras linguagens de programação o PowerShell pode fazer o uso de operadores Condicionais e Lógicos. Na sequência iremos apresentar alguns operadores condicionais e lógicos.

| Operador | Descrição | Exemplo | Significado e Saída |
|---|---|---|---|
| -lt | Menor que | $a –lt $b | A é menor que B? Booleano |
| -le | Menor ou igual | $a –le $b | A for menor ou igual a B? Booleano |
| -gt | Maior que | $a –gt $b | A é maior que B? Booleano |
| -ge | Maior ou igual | $a –ge $b | A é maior ou igual a B? Booleano |
| -eq | Igual | $a –eq $b | A é igua a B? Booleano |
| -ne | Não igual | $a –ne $b | A não é igual a B? Booleano |
| -like | Como | $a –like $b | A inclui um valor como B? Booleano |
| -notlike | Não como | $a –notlike $b | A não inclui um valor como B? |
| -contains | Contém | $a –contains $b | A está contido em B? Booleano |
| -notcontains | Não Contém | $a –notcontains $b | A não está contido em B? Booleano |
| -match | Coincide | $a –match $b | A coincide com B? Booleano |
| -notmatch | Não coincide | $a –notmatch $b | A não coincide com B? Booleano |
| -replace | Substitui | $a –replace $b,c$ | Se A possui strings de B substitua por C |

| Operador | Descrição | Exemplo | Significado e Saída |
|---|---|---|---|
| and | Operador lógico AND | $a –and $b | Verdade (1) se ambas as variáveis de entrada forem verdade |
| or | Operador lógico OR | $a –or $b | Verdade (1) se e somente se pelo menos uma das variáveis de entrada for verdade |
| not | Operador lógico NOT | $a –not $b | Negação (inverso) da variável atua |
| xor | Operador lógico XOR | $a –xor $b | Verdade (1) quando as variáveis assumirem valores diferentes entre si. |

| Operador | Descrição | Exemplo | Significado e Saída |
|---|---|---|---|
| + | Adição | 2 + 2 | Retorna a soma |
| / | Divisão | 4 / 2 | Retorna a quociente |
| % | Modulo | 5 % 2 | Retorna o resto da divisão |
| * | Multiplicação | 7 * 8 | Retorna o produto |
| - | Subtração | 7 -5 | Retorna a subtração |
| - | Negação | -7 | Transforma o valor em negativo. |

## IF

Para controlar os fluxos das operações baseados em condições, a forma mais simples é utilizando o comando IF. Um exemplo de utilização do comando é mostrado a seguir.

```
$num = Read-Host ”Digite um número: ”
if ($num -gt 1) {
  Write-Host ”O número é maior do que 1”
}
```

## FOR

Num script as vezes se faz necessário fazer o uso de looping para executar repetidamente um determinado comando. Os comandos (cmdlet) FOR, ForEach, While, Do While e Do Until tem esta finalidade. Iremos apresentar o comando FOR.

```
for ($i = 1; $i -le 5; $i++){
  Write-Host “192.168.1.$i”
}
```

## Funções

Funções são comandos em um script, que dura apenas durante a sessão em que estiver sendo executado, ou seja, quando terminar de usar os comandos em uma função ela não existirá mais na memória. A sintaxe da função do PowerShell é:

```
function Somar-Numeros {
  param (
    [int]$Numero1,
    [int]$Numero2
  )
  $soma = $Numero1 + $Numero2
  Write-Host ”A soma de é: $soma” }
# Exemplo de uso
Somar-Numeros -Numero1 5 -Numero2 7
```

# Comandos Relacionados a Redes

Na sequência será apresentado alguns comandos do PowerShell relacionados à configuração de redes de computadores. Lembre-se de executar esses comandos com usuário administrador.

## Informações sobre as Interfaces de Redes

O comando Get-NetAdapter é utilizado para obter informações sobre as placas de rede (adaptadores de rede) em um sistema Windows. Por exemplo, para listar todas as placas de rede:

```
Get-NetAdapter
```

Cada placa de rede recebe um identificador, chamado de InterfaceIndex. Com ele poderemos mostrar as configurações e configura-la. Por exemplo, para mostrar as informações da placa 16, utilizamos:

```
Get-NetAdapter -InterfaceIndex 13
```

Cada placa de rede recebe também um nome, chamado de InterfaceAlias. Através dele podemos também especificar uma interface de rede, por exemplo, a interface Ethernet:

```
Get-NetAdapter -InterfaceAlias Ethernet
```

Para obtermos configurações IP das interfaces de rede podemos utilizar o comando Get-NetIPAddress. Por exemplo, para obtermos as configurações da interface Ethernet utilizamos o comando:

```
Get-NetIPAddress -InterfaceIndex 13
```

O comando Get-NetRoute fornece informações sobre a tabela de roteamento do computador. Digite e observe os resultados.

```
Get-NetRoute
```

## Atividade

**Em dupla** realize as atividades propostas e anexe posteriormente os scripts no Google Sala de Aula.

**1.** Crie um script PowerShell que lista todas as interfaces de rede no sistema, exibindo informações como o nome da interface, endereço IP atual, máscara de sub-rede e status. Faça uso de comentários explicativos sobre o que cada parte do código está fazendo.

## Alterando as Interfaces de Rede

Para alterarmos as configurações de rede, precisamos inicialmnete remover a configuração de IP atual da interface antes de criar uma nova, para isso vamos usar o comando Remove-NetIPAddress, por exemplo, na interface 13:

```
Remove-NetIPAddress -InterfaceIndex 13 -AddressFamily IPv4
```

Na sequência precisamos remover a configuração do Gateway da interface também, para isso iremos utilizar o comando Remove-NetRoute, por exemplo da interface 13.

```
Remove-NetRoute -InterfaceIndex 13 -AddressFamily 'IPv4'
```

O comando New-NetIPAddress permite criarmos a nova configuração de IP na interface. Utilizaremos os parâmetros AddressFamily você deverá informar se é um IP tipo v4, IpAddress deverá informar o IP que deseja definir para a interface, PrefixLength seria o prefixo de rede em bits, e por fim, o IP de Gateway de rede padrão que será utilizado.

Para facilitarmos a apresentação utilizaremos uma variável temporária chamada de EnderecoIP. Esta variável corresponde a um array por isso, será utilizado a declaração com @ ()

```
New-NetIPAddress -InterfaceIndex 13 -AddressFamily IPv4 -IPAddress 192.168.0.200 -PrefixLength 24 -DefaultGateway 192.168.0.1
```

Para alterar o servidor DNS de uma interface de rede no PowerShell, podemos usar o cmdlet Set-DnsClientServerAddress. Por exemplo, para configurarmos os servidores DNS do Google (8.8.8.8 e 8.8.4.4) na interface Ethernet usamos o comando:

```
Set-DnsClientServerAddress -InterfaceIndex 13 -ServerAddresses 8.8.8.8, 8.8.4.4
```

## Atividade

**2.** Crie um script para permitir a configuração de um endereço IP estático em uma interface de rede específica. Os usuários devem fornecer como parâmetro o identificador da interface, o novo endereço IP, a máscara de sub-rede, DNS e o gateway. Utilize variáveis para armazenar informações como o nome da interface, endereços IP, máscaras de sub-rede, etc., para tornar o script mais flexível.

## Habilitar Cliente DHCP

Para habilitarmos a configuração automática (DHCP) em uma interface específica (por exemplo, 13) repita os passos anteriores para remover as configurações atuais e utilize os seguintes parametros para utilizar a configuração automatica:

```
Remove-NetIPAddress -InterfaceIndex 13 -AddressFamily IPv4

Remove-NetRoute -InterfaceIndex 13 -AddressFamily 'IPv4'

Set-NetIPInterface -InterfaceIndex 13 -DHCP Enabled

Set-DnsClientServerAddress -InterfaceIndex 13 -ResetServerAddresses
```

## Testar as Interfaces

Por último iremos conhecer alguns dos principais comandos do PowerShell para teste de rede, que são úteis para diagnóstico e verificação da conectividade.

**Conectividade**

O cmdlet Test-Connection é usado para testar a conectividade básica entre o computador local e um host remoto. Ele envia pacotes ICMP e fornece informações sobre a resposta. Por exemplo testando a conectividade para o endereço www.Google.com

```
Test-Connection www.Google.com
```

**Portas Abertas**

Para verificarmos porta aberta podemos utilizar o comando Testar-NetConnection. Ele permite testar a conectividade em um nível mais avançado, incluindo a verificação de portas específicas. Por exemplo, testando a porta 80

```
Test-NetConnection www.Google.com -Port 80
```

**Resolução de Nomes**

O cmdlet Resolve-DnsName é usado para testar a resolução de nomes de host para endereços IP e vice-versa. Por exemplo:

```
Resolve-DnsName www.Google.com
```

**3.** Crie um novo script de testes, que utiliza os comandos Test-Connection, Test-NetConnection, e Resolve-DnsName para realizar testes de conectividade, verificar uma porta e resolver um nome de host. O usuário deverá escolher a operação desejada e passar os parâmetros adequados: EnderecoParaTestar, Porta e Opção (conectividade, verificar porta e resolver nome).