PowerShell-PowerShell Remoting

PowerShell Remoting is a feature in Windows that allows you to run commands on remote computers using the Invoke-Command cmdlet. Here's an overview of PowerShell Remoting and some examples:

What is PowerShell Remoting?

PowerShell Remoting is a feature that enables you to run PowerShell commands on one computer from another computer over a network. This allows you to manage and administer multiple computers remotely, without having to physically log in to each computer.

How does it work?

When you enable PowerShell Remoting on a computer, it creates a listening socket on port 5985 (or 5986 for SSL encryption). When you use the Invoke-Command cmdlet on another computer, it connects to this socket and sends the command to the remote computer. The remote computer then executes the command and returns the results back to the original computer.

Advantages of PowerShell Remoting

1. **Remote management**: Run commands on multiple computers from a single console.
2. **Centralized administration**: Manage multiple computers from one location.
3. **Efficient use of resources**: Avoid the need to log in to each computer individually.

Example: Running a command on a remote computer

Let's say you have two computers, `LocalMachine` and `RemoteComputer`. You want to run the following command on `RemoteComputer`:

```
Get-Process | Where-Object {$_.CPU > 10}
```

To do this using PowerShell Remoting:

1. Enable PowerShell Remoting on both computers (if it's not already enabled).
2. On `LocalMachine`, use the Invoke-Command cmdlet to run the command on `RemoteComputer`:

```
Invoke-Command -ComputerName RemoteComputer -ScriptBlock { Get-Process | Where-Objec
```

This will execute the command on `RemoteComputer` and return the results back to `LocalMachine`.

Example: Running a script on multiple remote computers

Let's say you have three computers, `RemoteComputer1`, `RemoteComputer2`, and `RemoteComputer3`. You want to run the following PowerShell script on each of these computers:

```
 # Get-Process | Where-Object {$_.CPU > 10}
Get-WmiObject -Class Win32_Bios
```

To do this using PowerShell Remoting:

1. Save the script as a file (e.g., `GetBiosInfo.ps1`).
2. Use the Invoke-Command cmdlet to run the script on each remote computer:

```
 Invoke-Command -ComputerName RemoteComputer1,RemoteComputer2,RemoteComputer3 -FilePa
```

This will execute the script on each remote computer and return the results back to `LocalMachine`.

Security considerations

PowerShell Remoting uses WS-Man (Windows Server Management) under the hood. By default, it uses Kerberos authentication, but you can also use other authentication methods such as NTLM or Basic authentication.

To secure your PowerShell Remoting environment:

1. Use strong passwords and enable account lockout policies.
2. Limit access to only authorized users and groups.
3. Use SSL encryption (port 5986) instead of plaintext (port 5985).

I hope this summary helps!

Curated by Brajesh Kumar