Advanced Scripting   
Remote Sessions

Last Updated: 6/29/2021 1:43 PM Version 1  
Document Prepared for: CIT361 Student

# Name Zach Lucas ID 895649438

# Instructions

Save a copy of this document. Answer all questions directly in this document. You will save and upload this completed document as your homework submission.

# Overview

When you use Invoke-Command with a ComputerName the command first creates a PowerShell instance on the remote machine then passes the script block to the remote machine for processing. The results are passed back to the calling machine. When the script is done the PowerShell instance on the remote machine is terminated. If you use Invoke-Command frequently or say maybe in a loop, many instances of PowerShell are created and destroyed, this causes a lot of overhead. You can avoid this behavior by creating a PowerShell session. When the session is first started an instance of PowerShell is started on the remote machine. That instance gets reused as long as the session exists. Sessions exist until the user closes them, or the calling shell is terminated (disconnected sessions are a special case).

# Requirements

* SSH Client
* Internet access.

# Setup

SSH to the lab:  
Computer: **cit361-lab.citwdd.net**Port: **443**  
User: Your mailbox portion of your BYU-I email address. If your BYU-I email is [lin87690@byui.edu](mailto:lin87690@byui.edu) you would use **lin87690** for your username  
Password: Your I-Number (If you changed your password earlier in the semester you will need to use that one)

# Task 1—Interactive PowerShell Session

If you want to work interactively against a remote machine use an Interactive Session.

## Steps

1. Connect to the lab computer and Open PowerShell. Connect to DC.  
   Enter-PSSession DC
2. You should see a prompt that starts with a [dc] this indicates you are on a remote session. Ensure you are actually running on DC, enter the command  
   $env:ComputerName
   1. You should see DC
3. Anything you do in this session will be executed on DC. Get a directory of the root  
   dir \
   1. What directories are listed? PerfLogs, Program Files, Program Files (x86), Shares, Users, Windows
4. To exit the session just type exit  
   exit

# Task 2—Persistent (kind of) Sessions

You can create a session that can be reused as well.

## Steps

1. Create a remote session to DC  
   $dc=New-PSSession -ComputerName dc
2. View the sessions  
   Get-PSSession
   1. How many sessions are shown? 1
   2. What is the name of the session you just created? dc
3. Start another session  
   $sg=New-PSSession -ComputerName Slaygore
4. View the sessions  
   Get-PSSession
   1. How many sessions are shown? 2
   2. What is the name of the session you just created? Slaygore
5. Send a command to a session  
   invoke-command -Session $dc -ScriptBlock {get-date}
6. Send a command to multiple sessions (all one line)  
   invoke-command -Session $dc,$sg -ScriptBlock {"Then time on $($env:computername) is $(get-date)"}
   1. Is the time the same on both machines? No
7. Close PowerShell
8. Open PowerShell. View the sessions  
   Get-PSSession
   1. Are the sessions still there? No

# Task 3—Disconnected Sessions

## Steps

1. Create a variable on DC named $Answer and set it to 42  
   Invoke-Command -ComputerName dc -ScriptBlock {$Answer=42}
2. Read the value in Answer  
   Invoke-Command -ComputerName dc -ScriptBlock {"Answer is $Answer"}
   1. What is in answer. is
   2. Explain: No session was started beforehand
3. Get a list of sessions, Are there any sessions? No
4. Create a new session to DC  
   $dc=New-PSSession -ComputerName dc
5. Set the $Answer variable using the session  
   Invoke-Command -Session $dc -ScriptBlock {$Answer=42}
6. Read the value using the session  
   Invoke-Command -session $dc -ScriptBlock {"Answer is $Answer"}
   1. What is in answer. 42 is 42
   2. Explain: There was now a session for the variable to be stored at
7. Disconnect the session  
   Disconnect-PSSession $dc
8. Close PowerShell
9. Open PowerShell
10. Get a list of Sessions, Are there any sessions? No
11. Reconnect to the session  
    $dc=Connect-PSSession -ComputerName dc
12. Get the value of Answer  
    Invoke-Command -session $dc -ScriptBlock {"Answer is $Answer"}
    1. What is in answer. 42 is 42  
       is
    2. Explain: The original dc computer is still running with that variable stored, and this is a new one storing the vaule of “is”
13. Terminate the session  
    Remove-PSSession $dc

# Task 4—Importing Commands from a Remote Session

Now you will make a command on a remote system appear as if it is local to your system. However it will run on the remote system.

## Steps

1. Create a session on a remote machine  
   $s=Connect-PSSession -ComputerName dc
2. Use the session to define a function on that remote machine (all one line of code)  
   Invoke-Command $s -ScriptBlock {function who{"Hi, I'm $($env:computername)"}}
3. Prove the command does not exist on your machine  
   who
4. Prove the command exists on the remote machine  
   Invoke-Command $s {who}
5. Import that function to your machine  
   Import-PSSession $s -CommandName who
6. Try it  
   who
7. Clean up  
   Remove-PSSession $s
8. Prove it is gone  
   who

# Deliverable

Upload this document with completed answers to i-learn.