Advanced Scripting   
Remoting

Last Updated: 11/15/2021 9:06 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

In this exercise you will explore the remoting features of PowerShell using our class network.

# 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—Cmdlet Based Remoting

Any cmdlet that has a -ComputerName parameter is capable of remoting. In this task you will use cmdlet based remoting to get data from a remote machine.

## Steps

1. Open PowerShell (not PowerShell core) and get a count of the # of processes.  
   gps|Measure
   1. How many processes are running on the local machine? 275
2. Now get a list of the processes on a remote machine  
   gps -ComputerName dc|Measure
   1. How many processes are running on DC? 83
3. You can do the same for services
   1. How many services are running on the local machine? 280
   2. How many services are running on DC? 250

# Task 2—PowerShell Remoting

When using cmdlet-based remoting, the cmdlet can use whatever mechanism the author desires to perform the remoting. Some of the advantages to cmdlet-based remoting include, flexibility and speed. A drawback is that you need to configure the underlying remoting for each technique used. This may include firewall rules, permissions etc. Also you can only remote the cmdlets that are specifically designed for remoting. PoweShell provides a PowerShell based remoting infrastructure that allows you to launch a remote PowerShell session and run any script on it you wish. The advantage is since the script is actually running locally on the remote machine you are not limited to what PowerShell cmdlets you use, and you only need to configure an secure one method of remoting. A drawback is that an instance of PowerShell must be started on the remote machine which can be slower.

## Steps

1. Run as script on a remote machine. The $env:computername variable contains the name of the computer PowerShell is running on. Enter:  
   $env:COMPUTERNAME
   1. What is the name of the computer you are connected to? Horace
2. Now run the same script on a remote computer. Using the Invoke-Command cmdlet.   
   Invoke-Command -ComputerName DC -ScriptBlock {$env:COMPUTERNAME}
   1. What was returned? DC
3. View All of the environment variables on DC  
   Invoke-Command -ComputerName DC -ScriptBlock {dir env:}
   1. What is the Processor\_Architecture? AMD64
   2. What is then Number\_of\_processors? 8

# Task 3—Configuring Remoting

You can use the Enable-PSRemoting and Disable-PSRemoting to manage remoting. You will not change the remoting on this machine since it is a shared resource. However, we can look at some of the remote configurations.

## Steps

1. View the firewall rules for remoting  
   Get-NetFirewallRule winrm\*
2. Look at the rule **WINRM-HTTP-In-TCP-NoScope**, from the description what port is being used?

5985

# Wrap-up

That’s all there is to it 😊

# Deliverable

Upload this document with completed answers to i-learn.