## Writing PowerShell Scripts for Power BI

In this lab, you will gain experience writing PowerShell scripts to automate common tasks in a Power BI environment. You will begin by ensuring your Windows PC is configured for PowerShell script development and by installing the PowerShell library for Power BI named **MicrosoftPowerBIMgmt**. After that, you will write a few simple PowerShell scripts that connect to your Power BI test environment and execute commands to create workspaces, manage workspace users and import PBIX files. In later exercises, you will be required to write more advanced PowerShell code which uses the **Invoke-PowerBIRestMethod** to perform operations such as patching datasource credentials and updating dataset parameters.

You can complete these lab exercises using either Windows PowerShell 5 or PowerShell 7 (aka PowerShell Core). The instructions and screenshot in this lab will be based on writing and testing PowerShell scripts using Windows PowerShell 5 and the Windows PowerShell Integrated Scripting Environment (ISE).

### Exercise 1: Configure PowerShell to Run Scripts on Your Computer

In this exercise, xxx.

1. Download the student lab files to a local folder on your developer workstation.
   1. Create a new top-level folder on your workstation named **DevCamp** at a location such as **c:\DevCamp**.
   2. Download the ZIP archive with the student lab files from GitHub by clicking the following link.

<https://github.com/PowerBiDevCamp/PowerBI-PowerShell-Tutorial/archive/master.zip>

* 1. Extract the **StudentLabFiles** folder from **StudentLabFiles.zip** into a to a local folder such as **c:\DevCamp\StudentLabFiles**.
  2. The **StudentLabFiles** folder should contain the set of files shown in the following screenshot.

1. Ensure your computer has been configured to write and test PowerSHell scripts.
   1. ssss
2. Create a new PowerShell script named **Exercise01.ps1**.
   1. Launch the Windows PowerShell ISE and create a new PowerShell script.
   2. Save the new PowerShell script as **Exercise01.ps1** using the following path.

C:\DevCamp\Scripts\Exercise01.ps1

* 1. Add the following PowerShell code to Exercise01.ps1 to create an array of strings and enumerate through it.

Clear-Host

$hobbies = @("Pilates", "Kick boxing", "Power BI Embedding")

Write-Host

Write-Host "My Hobbies"

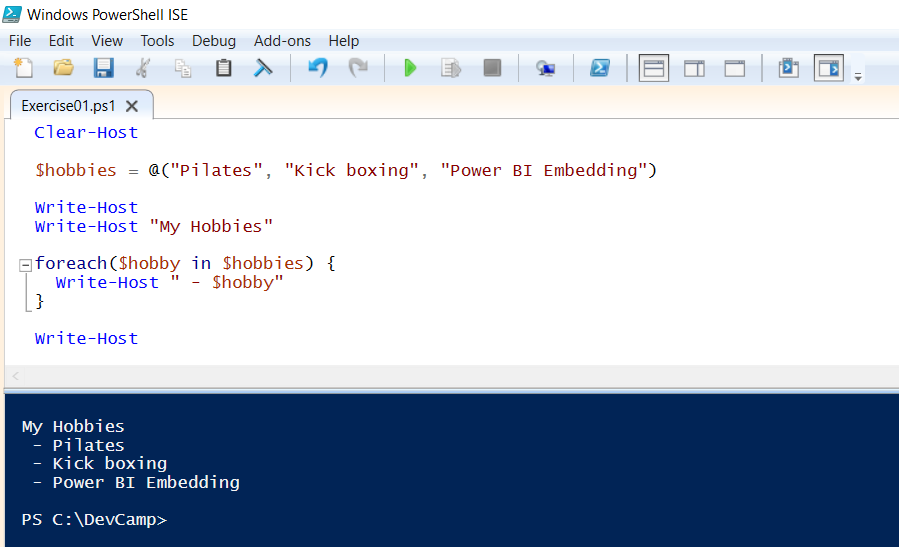
foreach($hobby in $hobbies) {

Write-Host " - $hobby"

}

Write-Host

* 1. Sss



* 1. Delete everything after the first line which calls Clear-Host
  2. Ss

$pets = @(

@{ Name="Bob"; Type="Cat" }

@{ Name="Diggity"; Type="Dog" }

@{ Name="Larry"; Type="Lizard" }

@{ Name="Penny"; Type="Porcupine" }

)

* 1. S

Write-Host

Write-Host "My Pets"

* 1. X

foreach($pet in $pets) {

$name = $pet.Name

$type = $pet.Type

Write-Host " - $name the $type"

}

* 1. Add Write-Host at the bottom.
  2. Your script should now match the following code listing.

Clear-Host

$pets = @(

@{ Name="Bob"; Type="Cat" }

@{ Name="Diggity"; Type="Dog" }

@{ Name="Larry"; Type="Lizard" }

@{ Name="Penny"; Type="Porcupine" }

)

Write-Host

Write-Host "My Pets"

foreach($pet in $pets) {

$name = $pet.Name

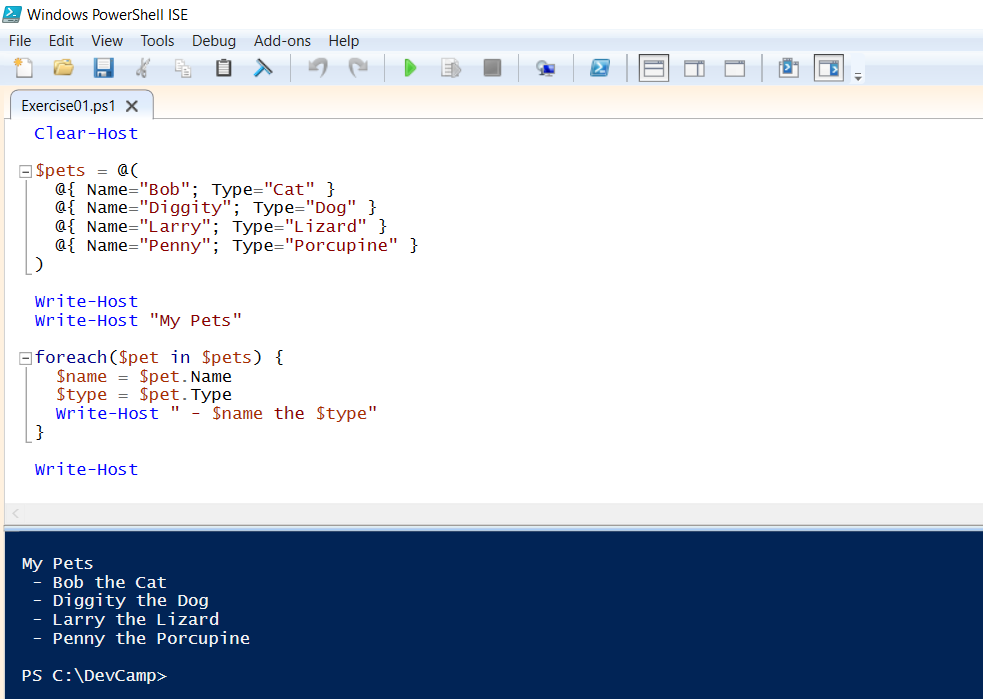
$type = $pet.Type

Write-Host " - $name the $type"

}

Write-Host

* 1. Press **{F5}** to execute the script. You should see output in the console that matches the following screenshot.



1. Write script output to a text file.
   1. Delete all the code in Exercise01.ps1.
   2. Add the following line to create a file path in the same folder as the script named **Pets.txt**.

$outputFilePath = "$PSScriptRoot/Pets.txt"

* 1. Add the following code to create an array of dictionary objects for a collection of pets.

$pets = @(

@{ Name="Bob"; Type="Cat" }

@{ Name="Diggity"; Type="Dog" }

@{ Name="Larry"; Type="Lizard" }

@{ Name="Penny"; Type="Porcupine" }

)

* 1. Add the following line of code to write a heading of **My Pets** into the output file.

"My Pets" | Out-File $outputFilePath

* 1. Create a foreach loop to enumerate the array of dictionary objects and to output a line of text for each pet which includes it name and type.

foreach($pet in $pets) {

$name = $pet.Name

$type = $pet.Type

" - $name the $type" | Out-File $outputFilePath -Append

}

* 1. Add one more line of PowerShell code to open up the text file in notepad.

notepad.exe $outputFilePath

* 1. Your script should now match the following code listing.

$outputFilePath = "$PSScriptRoot/Pets.txt"

$pets = @(

@{ Name="Bob"; Type="Cat" }

@{ Name="Diggity"; Type="Dog" }

@{ Name="Larry"; Type="Lizard" }

@{ Name="Penny"; Type="Porcupine" }

)

"My Pets" | Out-File $outputFilePath

foreach($pet in $pets) {

$name = $pet.Name

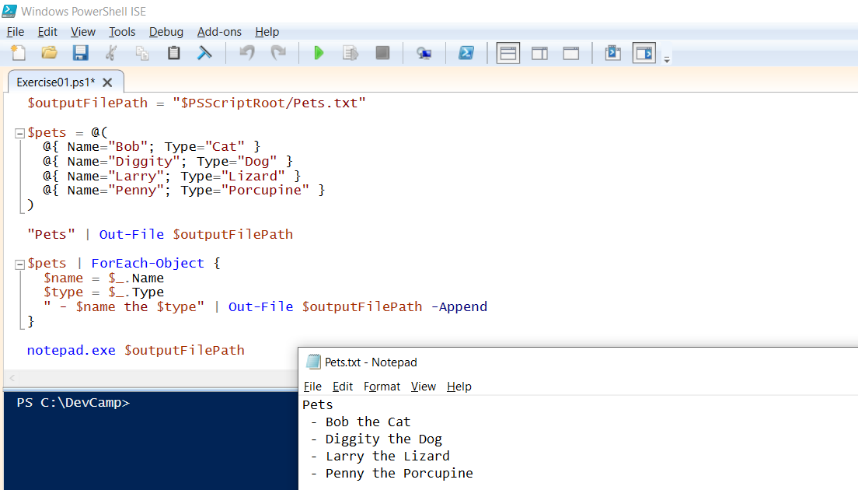
$type = $pet.Type

" - $name the $type" | Out-File $outputFilePath -Append

}

notepad.exe $outputFilePath

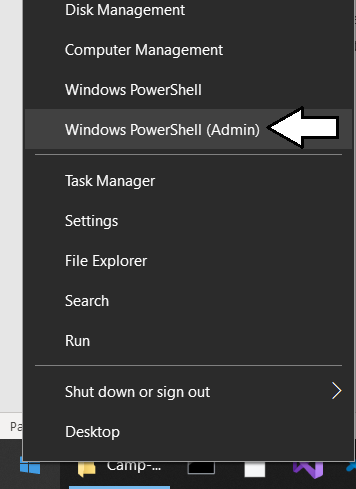
* 1. X



### Exercise 2: Install the Microsoft Power BI Cmdlets for Windows PowerShell

In this exercise, xxx.

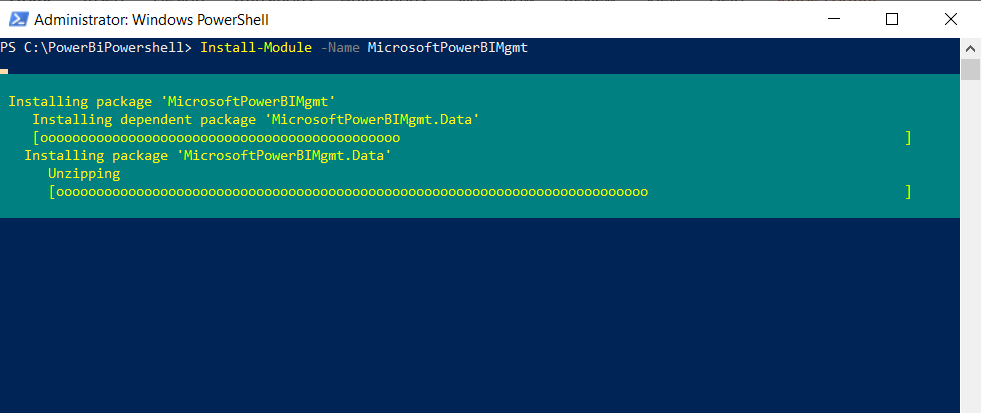
1. Install the PowerShell module named **MicrosoftPowerBIMgmt**. If you already installed **MicrosoftPowerBIMgmt** move to step 2.
   1. Right-click on the Windows Start menu and open



* 1. Type and execute the following

Install-Module -Name MicrosoftPowerBIMgmt

* 1. Xxx



Once you have installed the **MicrosoftPowerBIMgmt** module, there is no more need to use a Administrative PowerShell session. You can now return to the PowerShell ISE and use a standard PowerShell session.

1. Create a new PowerShell script named **Exercise02.ps1**.
   1. Return to the Windows PowerShell ISE and create a new PowerShell script,
   2. Save the new PowerShell script as **Exercise02.ps1** using the following path.

C:\DevCamp\Scripts\Exercise02.ps1

1. Use the **Connect-PowerBIServiceAccount** cmdlet to connect to the Power BI Service.
   1. Copy and paste the following PowerShell code into **Exercise02.ps1**.

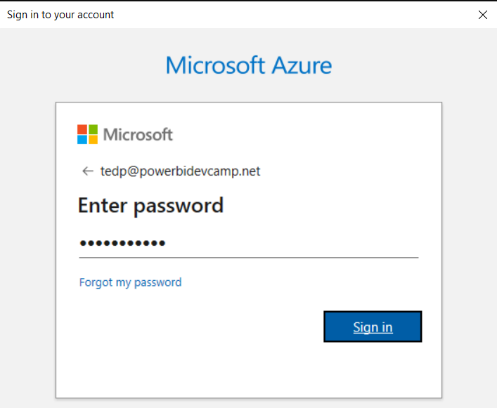
$user = Connect-PowerBIServiceAccount

$userName = $user.UserName

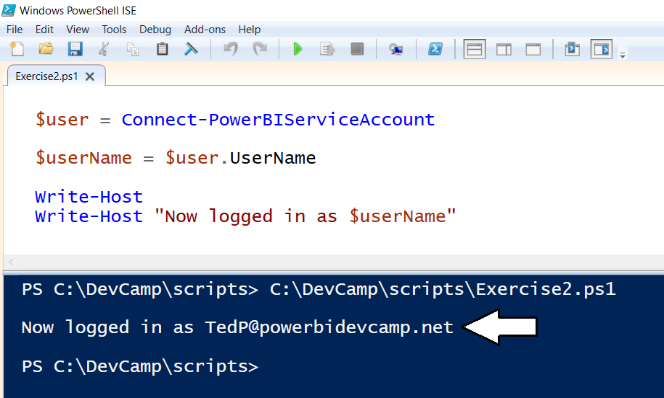
Write-Host

Write-Host "Now logged in as $userName"

* 1. Save your changes to **Exercise02.ps1**.
  2. Press the **{F5}** key to execute the PowerShell code in **Exercise02.ps1**.
  3. When the script executes, you should be prompted to sign in.
  4. Sign in with your a user name and password.



* 1. After **Connect-PowerBIServiceAccount** executes, you script should display your user account in the console window.



As you can see, you can write scripts that don't contain any user names or passwords. This type of script can be run by a user interactively where the user supplies a user name and password when the script begins to execute. In some scenarios such as PowerShell script development, it can helpful to hard-code the user name and password into the script so that it runs without any need for user interaction. In the next step you will modify the script with a hard-code user name and password.

1. Update **Exercise02.ps1** to log in without requiring interaction on the part of the user.
   1. Delete all the code in **Exercise02.ps1**.
   2. Copy the following PowerShell code and paste it into **Exercise02.ps1**.

# log into Azure AD user account with hard-code user name and password

$userName = "user1@tenant1.onMicrosoft.com"

$password = "myCat$rightLeg"

# convert password to secure string

$securePassword = ConvertTo-SecureString -String $password -AsPlainText -Force

# create PSCredential object to serve as login credentials

$credential = New-Object -TypeName System.Management.Automation.PSCredential `

-ArgumentList $userName, $securePassword

# log into Power BI unattended without any user interaction

$user = Connect-PowerBIServiceAccount -Credential $credential

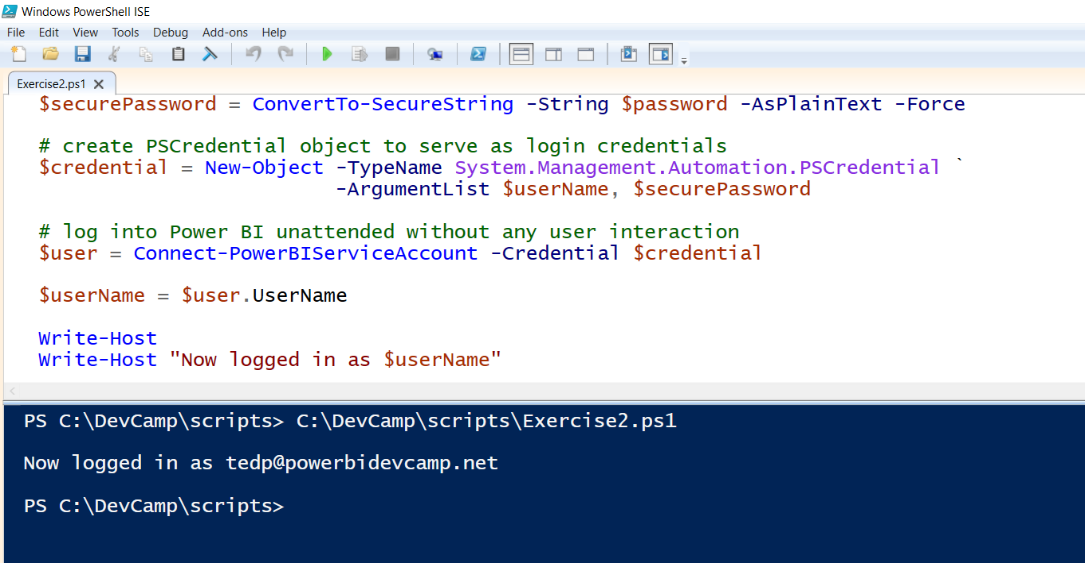
$userName = $user.UserName

Write-Host

Write-Host "Now logged in as $userName"

This script demonstrates a common technique of creating a PSCredential object using a secure string to include the password.

* 1. Press the **{F5}** key to execute the PowerShell code in **Exercise02.ps1**.
  2. The script should now execute successfully without requiring you to sign in interactively.



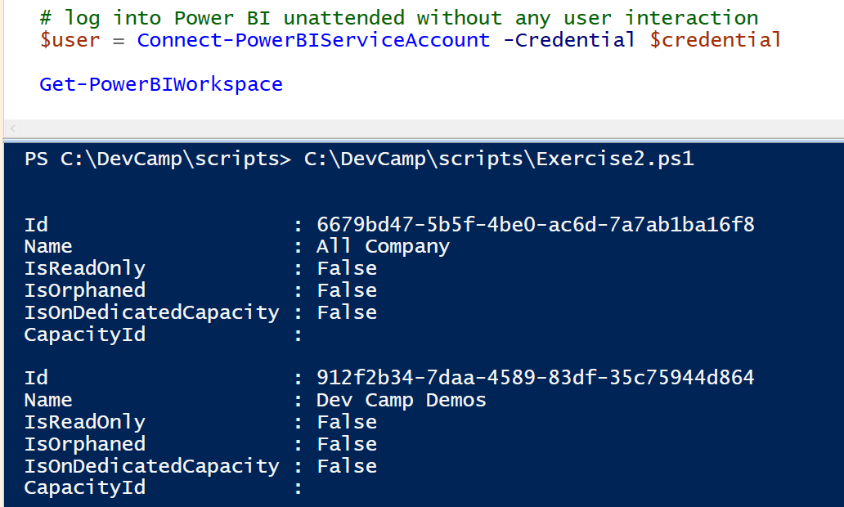
1. Add a call to **Get-PowerBIWorkspace**.
   1. Delete the lines of PowerShell code that appear after the call to Connect-PowerBIServiceAccount.
   2. Add a call to **Get-PowerBIWorkspace**.

# log into Power BI unattended without any user interaction

$user = Connect-PowerBIServiceAccount -Credential $credential

Get-PowerBIWorkspace

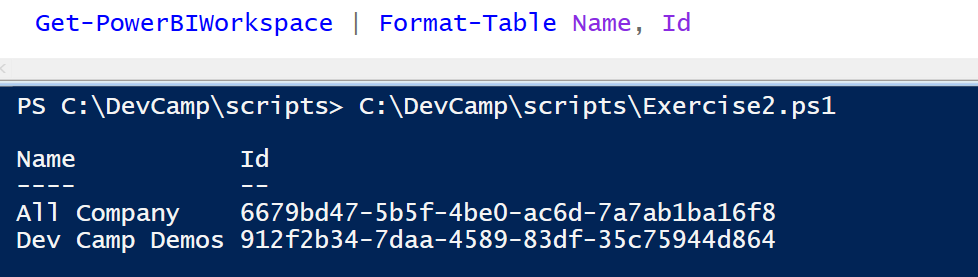
* 1. Press the **{F5}** key to execute the PowerShell code in **Exercise02.ps1**.
  2. The script display output for each Power BI workspace that your user account has permissions to view..



* 1. Reformat the output of **Get-PowerBIWorkspace** using the **Format-Table** cmdlet.

Get-PowerBIWorkspace | Format-Table Name, Id

* 1. Press the **{F5}** key to execute the PowerShell code in **Exercise02.ps1**.
  2. The script display output for each Power BI workspace that your user account has permissions to view..



### Exercise 3: Write a Script to Create Workspaces and Add Workspace Users

In this exercise, xxx.

1. Create a new PowerShell script named **Exercise03.ps1**.
   1. Return to the Windows PowerShell ISE and create a new PowerShell script,
   2. Save the new PowerShell script as **Exercise03.ps1** using the following path.

C:\DevCamp\Scripts\Exercise03.ps1

* 1. xxx

Write-Host

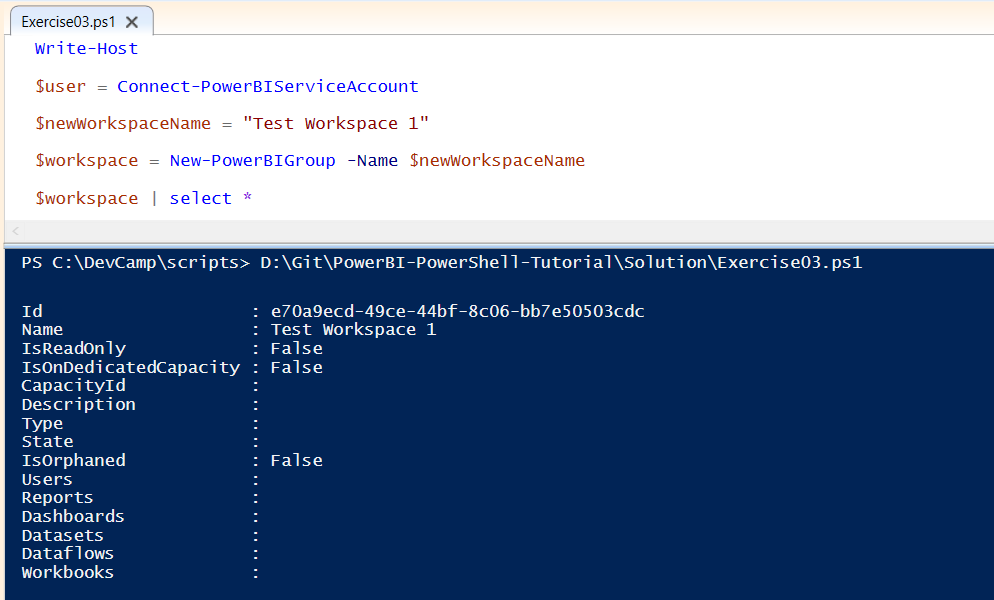
$user = Connect-PowerBIServiceAccount

$newWorkspaceName = "Test Workspace 1"

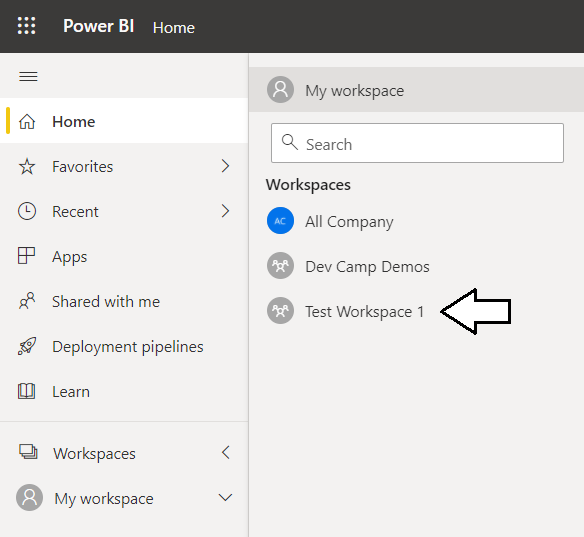
$workspace = New-PowerBIGroup -Name $newWorkspaceName

$workspace | select \*

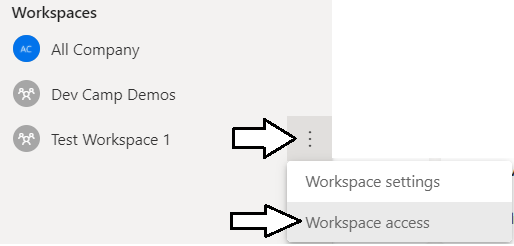
* 1. ssss



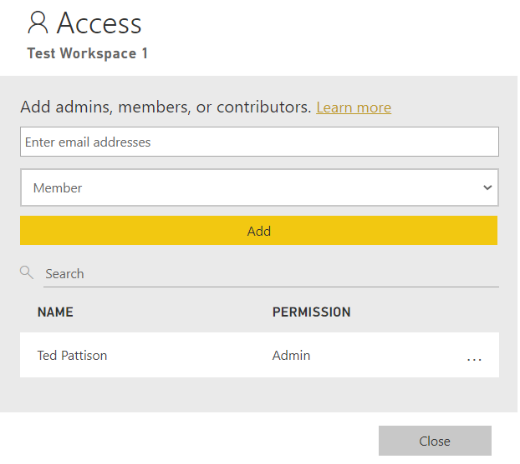
* 1. xxxxx



* 1. sssss

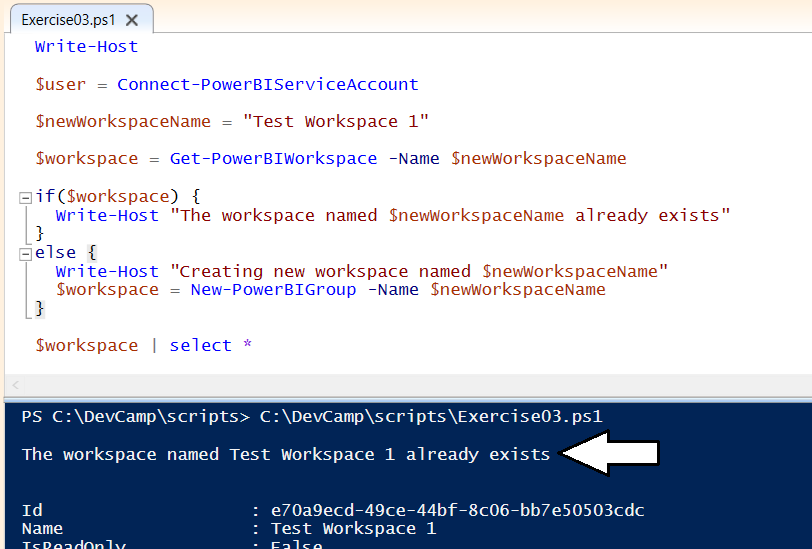


* 1. x



* 1. x

x



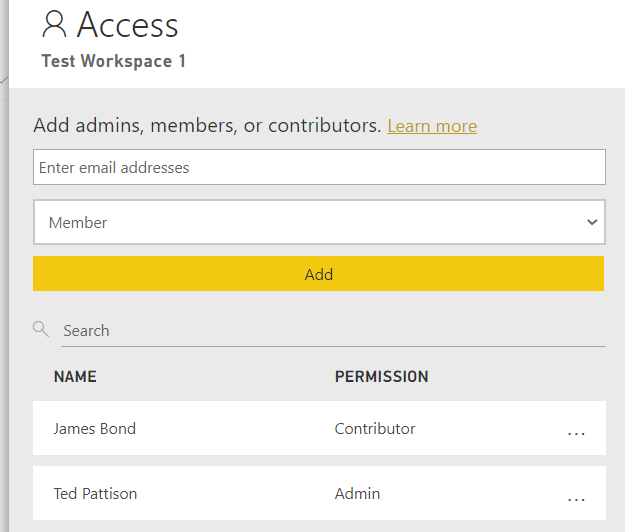
1. Add a new user
   1. Ssss

# add user as workspace member

$userEmail = "JamesB@powerbidevcamp.net"

Add-PowerBIWorkspaceUser -Id $workspace.Id -UserEmailAddress $userEmail -AccessRight Contributor

* 1. Sss



* 1. s

### Exercise 4: Write a Script to Upload and Publish Content

In this exercise, xxx.

1. Create a new PowerShell script named **Exercise04.ps1**.
   1. Return to the Windows PowerShell ISE and create a new PowerShell script,
   2. Save the new PowerShell script as **Exercise04.ps1** using the following path.

C:\DevCamp\Scripts\Exercise04.ps1

* 1. S

Write-Host

$user = Connect-PowerBIServiceAccount

$newWorkspaceName = "Test Workspace 1"

$workspace = Get-PowerBIWorkspace -Name $newWorkspaceName

if($workspace) {

Write-Host "The workspace named $newWorkspaceName already exists"

}

else {

Write-Host "Creating new workspace named $newWorkspaceName"

$workspace = New-PowerBIGroup -Name $newWorkspaceName

}

1. Add PowerShell code to publish a PBIX file.
   1. Aaaa

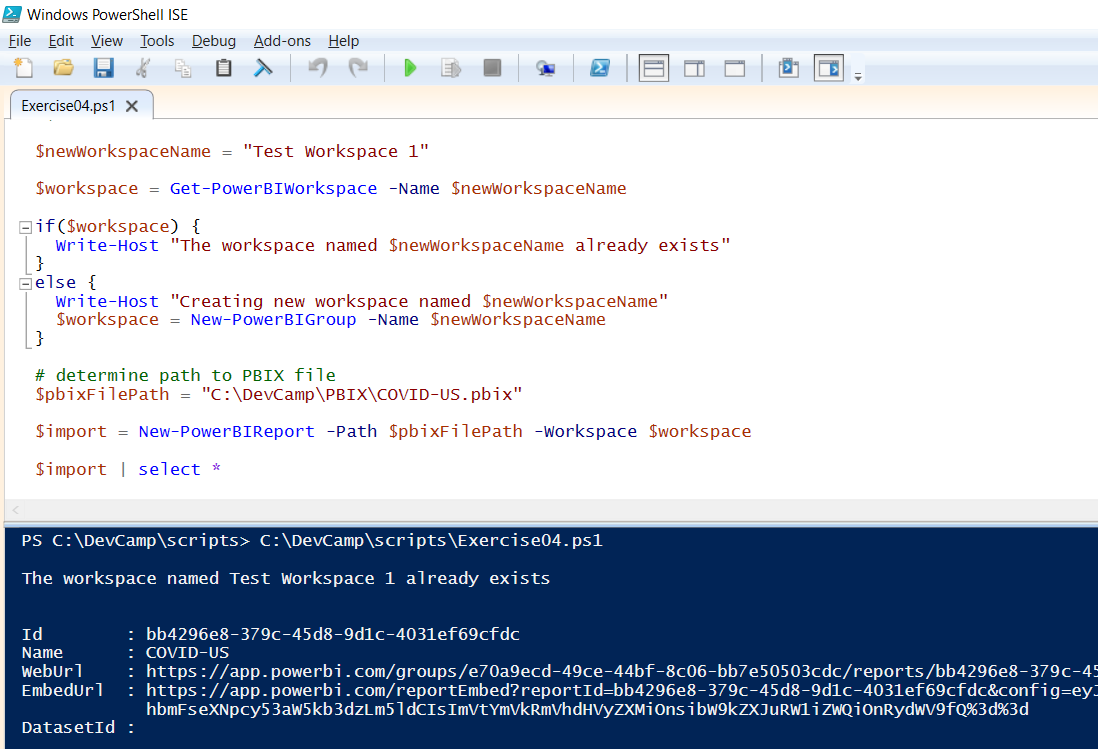
# determine path to PBIX file

$pbixFilePath = "C:\DevCamp\PBIX\COVID-US.pbix"

$import = New-PowerBIReport -Path $pbixFilePath -Workspace $workspace

$import | select \*

* 1. Add PowerShell code to create a variable which tracks a path to a PBIX file..



* 1. Xx
  2. xx

1. Create a new top-level folder

### Exercise 5: Write a Script to Patch Datasource Credentials

In this exercise, xxx.

1. Create a new PowerShell script named **Exercise05.ps1**.
   1. Return to the Windows PowerShell ISE and create a new PowerShell script,
   2. Save the new PowerShell script as **Exercise05.ps1** using the following path.

C:\DevCamp\Scripts\Exercise05.ps1

1. Download the student lab files to a local folder on your developer workstation.
2. Create a new top-level folder

### Exercise 6: Write a Script to Update Dataset Parameters

In this exercise, xxx.

1. Create a new PowerShell script named **Exercise07.ps1**.
   1. Return to the Windows PowerShell ISE and create a new PowerShell script,
   2. Save the new PowerShell script as **Exercise07.ps1** using the following path.

C:\DevCamp\Scripts\Exercise07.ps1

1. cc

### Exercise 7: Run Get-PowerBIWorkspace at Organization Scope

In this exercise, xxx.

1. Create a new PowerShell script named **Exercise05.ps1**.
   1. Return to the Windows PowerShell ISE and create a new PowerShell script,
   2. Save the new PowerShell script as **Exercise05.ps1** using the following path.

C:\DevCamp\Scripts\Exercise05.ps1

1. Download the student lab files to a local folder on your developer workstation.
   1. Xx

Get-PowerBIWorkspace -Scope Organization

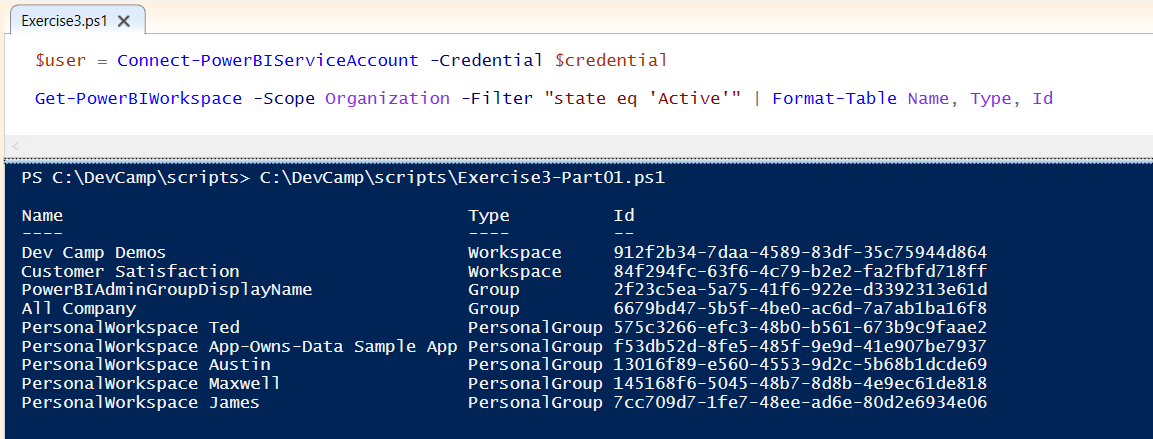
* 1. Sss

Get-PowerBIWorkspace -Scope Organization -Filter "state eq 'Active'"

* 1. Sss

Get-PowerBIWorkspace -Scope Organization -Filter "state eq 'Active'" | Format-Table Name, Type, Id

* 1. Sss



1. Zzz
   1. Zzzz

### Exercise 8: Write a Script that Exports Power BI Activity Events

In this exercise, xxx.

1. Create a new PowerShell script named **Exercise08.ps1**.
   1. Return to the Windows PowerShell ISE and create a new PowerShell script,
   2. Save the new PowerShell script as **Exercise08.ps1** using the following path.

C:\DevCamp\Scripts\Exercise08.ps1

1. Cc

### Exercise 9: Write a Script that Runs as a Service Principal

In this exercise, xxx.

1. Create a new PowerShell script named **Exercise09.ps1**.
   1. Return to the Windows PowerShell ISE and create a new PowerShell script,
   2. Save the new PowerShell script as **Exercise09.ps1** using the following path.

C:\DevCamp\Scripts\Exercise09.ps1

1. Takeover and Refresh a Dataset as Service Principal

### Exercise 10: Write a Script to Import a Dataflow

In this exercise, xxx.

1. Create a new PowerShell script named **Exercise10.ps1**.
   1. Return to the Windows PowerShell ISE and create a new PowerShell script,
   2. Save the new PowerShell script as **Exercise10.ps1** using the following path.

C:\DevCamp\Scripts\Exercise10.ps1

1. xx