Module 06 - Azure Automation - Source Control and Authentication

Exercise 1: Configure Azure Automation Connection

Introduction

In this lab, we will create a VM for testing and create the Service Principal Name (SPN) for your Azure Automation Runbooks to use, so they can access Azure Resources in your Subscription.

Summary

In the later labs in this module, we will be configuring Runbooks to start and Stop Virtual Machines. To do this, we first have to place the Azure Subscription ID into an Azure Automation Variable, so it may be read at Runtime.

Estimated Time to Complete This Lab

10 minutes

Task 1: Deploy a New Lab VM

1. [] In the Azure Portal open the Cloud Shell and start PowerShell by typing *pwsh*.

2. [] Enter the code below to create a new virtual machine. Be sure to specify a location and enter a user name and password when prompted.

```
$location = ""
New-AzVM -ResourceGroupName ContosoIntegrationRG -Name IntegrationVM -Location $location -
PublicIpAddressName IntegrationVM
```

3. [] Wait for the prompt to return - confirm the VM is running by executing the code below.

```
Get-AzVM -ResourceGroupName ContosoIntegrationRG -Status
```

Task 2: Enable a Managed Identity on the Automation Account

1. [] Select your automation account (Contoso Automation Account) in the Azure Portal

2. [] Click on **Identity** under the **Acccount Settings** section

3. [] Change the **Status** slider to on and click **Save**. Click **Yes** when prompted.

4. [] A new identity will be created in Azure AD which you can then assign a role to. Use the code below in the Cloud Shell to give the new identity the Contributor role on the subscription.

```
$scope = "/subscriptions/$((Get-AzContext).Subscription)"
$identity = (Get-AzResource -Name ContosoAutomationAccount).Identity.PrincipalId
New-AzRoleAssignment -Scope $scope -ObjectId $identity -RoleDefinitionName 'Contributor'
```

Exercise 2: Create a PowerShell Runbook to Start an Azure VM

Introduction

This scenario will walk through how to create a PowerShell based Runbook, which starts Azure Virtual Machines. This will be used later in this module, with a schedule, demonstrating how you may start or stop VMs, automatically, via scheduling.

Summary

In this lab, we will:

- Add the new Az modules to the Automation Account
- Create a PowerShell Runbook
- Use a PowerShell script to start an Azure VM
- Test the Runbook via the Azure Automation portal

Estimated Time to Complete This Lab

20 minutes

Task Description

1. [] Go to https://portal.azure.com and from the menu on the left, click **Virtual Machines**.

2. [] In your list of VMs, you should see **IntegrationVM**, which is the VM created in the previous module. Notice, this is in Resource Group **ContosoIntegrationRG**.

3. [] Click the IntegrationVM and then click **Stop**, then **Yes** to stop the VM.

4. [] Next, in the Azure Portal, go to your **ContosoAutomationAccount**. Automation account

5. [] Click **Runbooks** and click **+Create a runbook**.

6. [] Enter the following for your new Runbook:

    - **Name:** StartAzureVMs
    - **Runtime version:** 5.1
    - **Runbook Type:** Powershell

7. [] Click **Create**.

8. [] Type or copy/paste the following PowerShell into your new Runbook. **NOTE:** If you have a problem with copy/paste, a copy of the script is located in your lab files in **C:\Labs\Module6\StartAzureVMs.ps1**

```
Param(
    [Parameter(Mandatory = $false)]
    [String] $ResourceGroupName
)

Connect-AzAccount -Identity

if ($ResourceGroupName) {
    Write-Output "Resource Group specified: $($ResourceGroupName)"
    $VMs = Get-AzVM -ResourceGroupName $ResourceGroupName
}
else {
    Write-Output "No Resource Group specified"
    $VMs = Get-AzVM
}

foreach ($VM in $VMs) {
    try {
        Write-Output "Starting VM: $($VM.Name)"
        $VM | Start-AzVM -ErrorAction Stop
        Write-Output ($VM.Name + " has been started")
    }
    catch {
        Write-Output ($VM.Name + " failed to start")
    }
}
```

9. [] Click **Save** and the click **Test pane**.

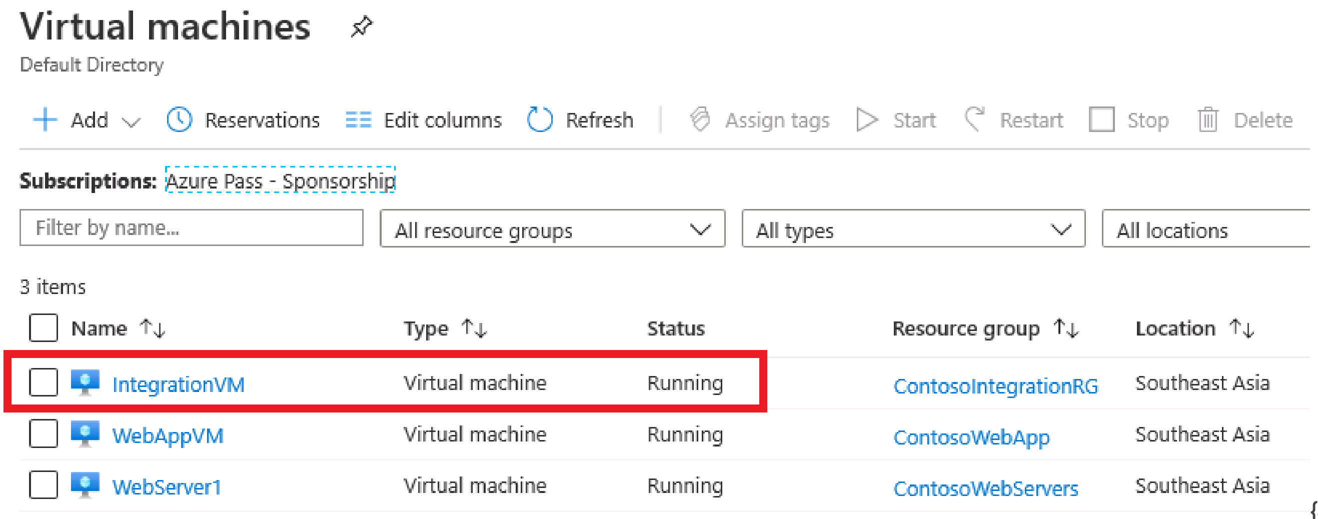
10. [] In the RESOURCEGROUPNAME parameter, enter **ContosoIntegrationRG** (this is the resource group where your IntegrationVM is located), then click **Start**.

11. [] After a moment, you will see the Runbook start, and output that the Virtual Machine was Started Successfully:

```
Logging in to Azure...



Environments
Context

------------
-------

{[AzureCloud, AzureCloud], [AzureChinaCloud, AzureChinaCloud], [AzureUSGovernment,
AzureUSGovernment]} Microsoft.Azur...

Setting context to a specific subscription



Account                              Environment Subscription                         Tenant

-------                              ----------- ------------                         ------

d3f8e4c0-52dr-5521-aa9e-c77e5dbfb61f AzureCloud  d90fd978-er23-4ce6-bdbb-l38473a6cc1b7
2c65fbfe-0e3b-416b-a0c1-4def648...

Resource Group specified: ContosoIntegrationRG

Starting VM: IntegrationVM



RequestId IsSuccessStatusCode StatusCode ReasonPhrase

--------- ------------------- ---------- ------------

                         True         OK OK

IntegrationVM has been started
```

12. [] Close the Test pane. In your Runbook, click **Publish**, then **Yes**.

13. [] In the Azure Portal, go back to your Virtual Machines. Verify that the **IntegrationVM** has now started and is running:



Subscriptions: Azure Pass - Sponsorship

Filter by name... | All resource groups | All types | All locations

3 items

Name	Type	Status	Resource group	Location
IntegrationVM	Virtual machine	Running	ContosoIntegrationRG	Southeast Asia
WebAppVM	Virtual machine	Running	ContosoWebApp	Southeast Asia
WebServer1	Virtual machine	Running	ContosoWebServers	Southeast Asia

{400}

Exercise 3: Using a Webhook

Introduction

This scenario will walk through how to trigger Runbooks via a HTTPS Webhook.

Summary

In this lab, we will:

- Create a Webhook to Stop an Azure VM
- Trigger the Webhook

Estimated Time to Complete This Lab

20 minutes

Task Description

1. [] Go to https://portal.azure.com and navigate to your **ContosoAutomationAccount** Automation Account.

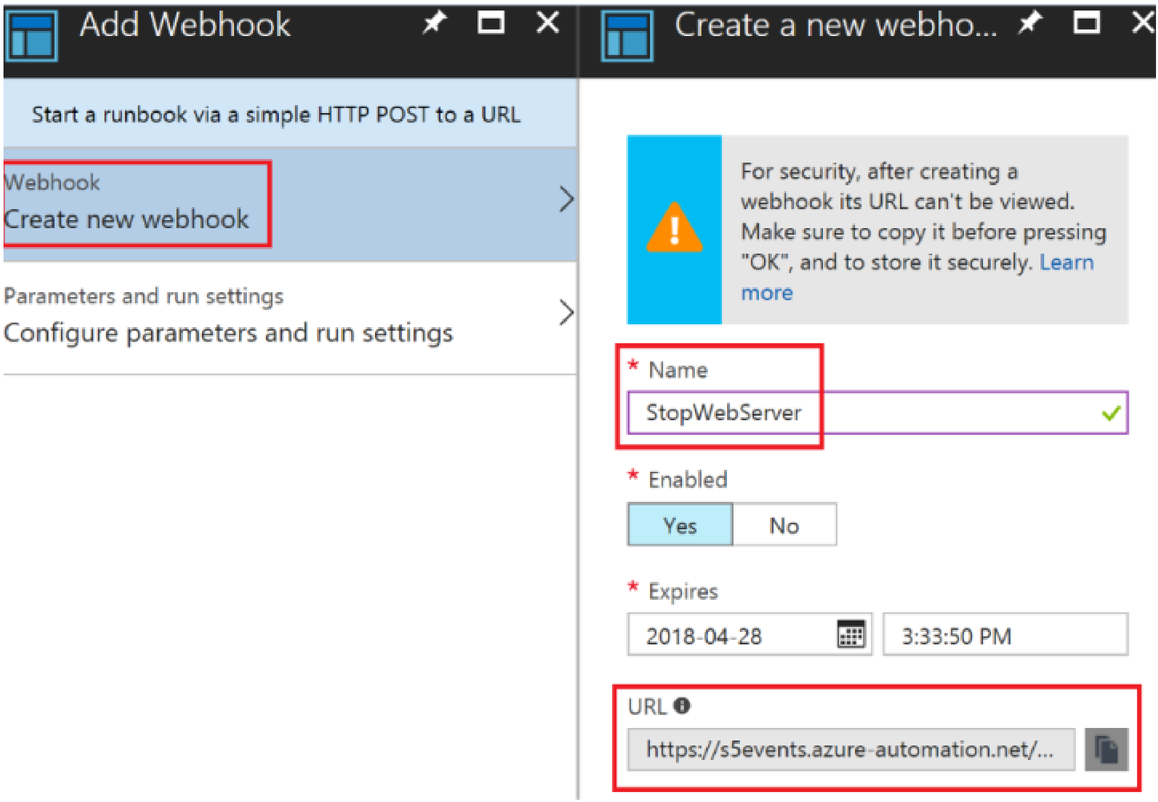
2. [] Click **Runbooks** and click the Runbook **StopAzureV2Vm**.

3. [] In the StopAzureV2Vm Runbook, click **Webhooks** under RESOURCES.

4. [] Click **+Add Webhook**, select **Create new webhook** and name the Webhook **StopWebServer**.

5. [] Copy the **URL** of the Webhook and paste it into Notepad.

> IMPORTANT! Copy the URL of the Webhook and make sure you make a note of it. Once you have created the Webhook, you cannot see the URL again!



Start a runbook via a simple HTTP POST to a URL

Webhook
Create new webhook

Parameters and run settings
Configure parameters and run settings

For security, after creating a webhook its URL can't be viewed. Make sure to copy it before pressing "OK", and to store it securely. Learn more

* Name: StopWebServer

* Enabled: [✓ Yes] No

* Expires: 2018-04-28 3:33:50 PM

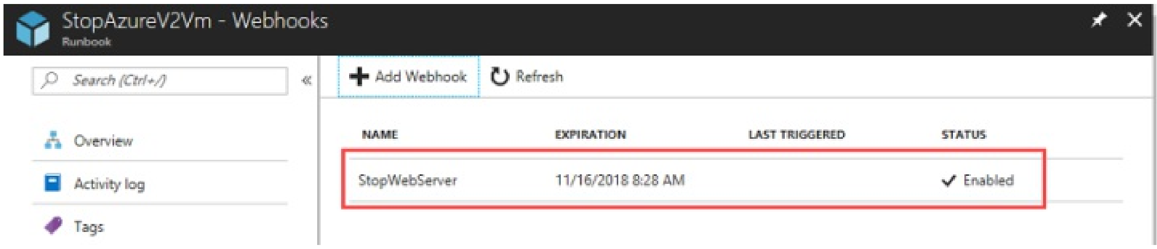
URL: https://s5events.azure-automation.net/...

{400}

6. [] You're not done yet. Click **OK** then click the **Configure parameters and run settings**.

7. [] In the Parameters, for **RESOURCEGROUPNAME** enter **ContosoIntegrationRG** Click **OK**.

8. [] Click **Create**.



NAME	EXPIRATION	LAST TRIGGERED	STATUS
StopWebServer	11/16/2018 8:28 AM		✓ Enabled

{400}

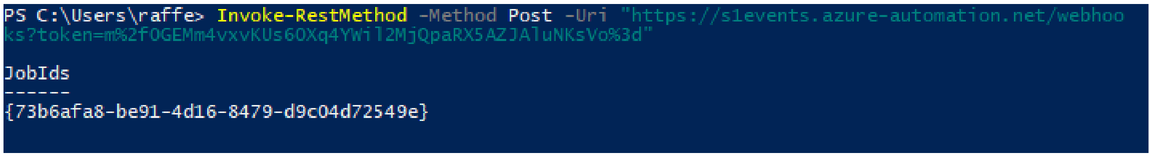
9. [] Let's make sure the VM is running. In the Azure Portal, go to **Virtual Machines**. Click the VM **IntegrationVM**.

10. [] If the VM is not Running, click **Start**.

11. [] Next, let's start the Webhook. On your lab machine, open **Windows PowerShell**.

12. [] Enter the following command, pasting in your **Webhook URI** as appropriate:

```
Invoke-RestMethod -Method Post -Uri "MYWEBHOOKURI"
```



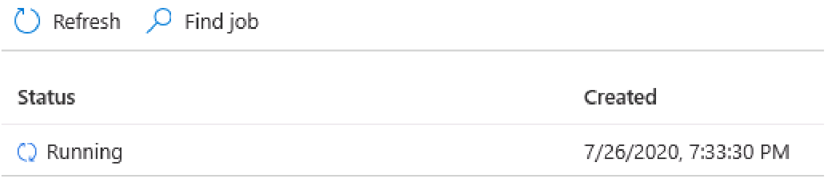
```
PS C:\Users\raffe> Invoke-RestMethod -Method Post -Uri "https://s1events.azure-automation.net/webhoo
ks?token=m%2fOGEMm4vxvKUs6OXq4YWil2MjQpaRX5AZJAluNKsVo%3d"

JobIds
------
{73b6afa8-be91-4d16-8479-d9c04d72549e}
```

{400}

> IMPORTANT! Webhooks are unauthenticated and rely on the privacy of the webhook URL. For this reason it is important to validate the use of a webhook, some ways we can do this are listed at https://docs.microsoft.com/en-us/azure/automation/automation-webhooks#webhook-security. If you want to use Azure AD authentication or some other auth provider consider using an Azure Function (which also supports PowerShell).

13. [] From your Automation account, click **Runbooks** and click the **StopAzureV2Vm** Runbook. Click **Jobs**.

14. [] You should see that the Runbook was successfully triggered by your Webhook:



Status	Created
Running	7/26/2020, 7:33:30 PM

{400}

15. [] In the Azure Portal, navigate to **Virtual machines** validate that the VM is stopped.

16. [] Start **IntegrationVM** so it will be ready for a later exercise.

Exercise 4: Starting a Runbook in Response to an Azure Alert

Introduction

This lab will demonstrate the native integration between Azure Monitor and Azure Automation. You can configure Runbooks to be automatically triggered when an alert is raised. You can either use your own Runbook, or take advantage of some of the pre-defined Runbooks in Azure.

Summary

In this lab, we will:

- Setup a metric alert on an Azure VM
- Manually cause the alert to be raised
- Verify the Runbook executed successfully

Estimated Time to Complete This Lab

20 minutes

Task Description

1. [] Go to https://portal.azure.com and navigate to **Virtual Machines**.

2. [] Click your **IntegrationVM** VM. If it is not started, click the **Start** button.

3. [] We need to register the resource providers for Azure Monitor - open the Azure Cloud shell and at the prompt enter the following commands.

```
az provider register -n Microsoft.Insights
az provider register -n Microsoft.AlertsManagement
```

4. [] From **IntegrationVM** under MONITORING click **Alerts** and click **New alert rule**.

5. [] In the **Condition** section click **Select Condition** and from the list of signals click on **Percentage CPU**. Configure your signal with the following settings in the alert logic section and click **Done**.

   - **Threshold:** Static
   - **Operator:** Greater than
   - **Aggregation type:** Average
   - **Threshold value:** 40
   - **Aggregation granularity (Period):** 5 minutes
   - **Frequency of evaluation:** Every 1 minute

6. [] In **Action** section click the **Add action groups** button.

7. [] Click **+ Create Action Group**

8. [] Specify an action group name and a short name. Set the resource group to **ContosoIntegrationRG**.

9. [] Moving on to the Actions section, specify an action name and select **Automation runbook** as the action type. In the tab that pops up, set the following values:

   - **Run runbook:** Enabled
   - **Runbook source:** Built-in
   - **Runbook:** Restart VM
   - **Automation account:** ContosoAutomationAccount
   - **Enable the common alert schema:** No

10. [] Click **OK** to save the Runbook Configuration and Click **OK** to add the action group. Once the action group is created, click on **Select action group** and pick your new action group.

11. [] Name your alert **RestartOnHighCPU** in the **Alert rule name** section, **Enable rule upon creation** should be set to **Yes**. Create your alert rule by clicking on **Create alert rule**.

12. [] From **IntegrationVM** , click **Overview**, then click **Connect** to open a Remote Desktop session to the VM:

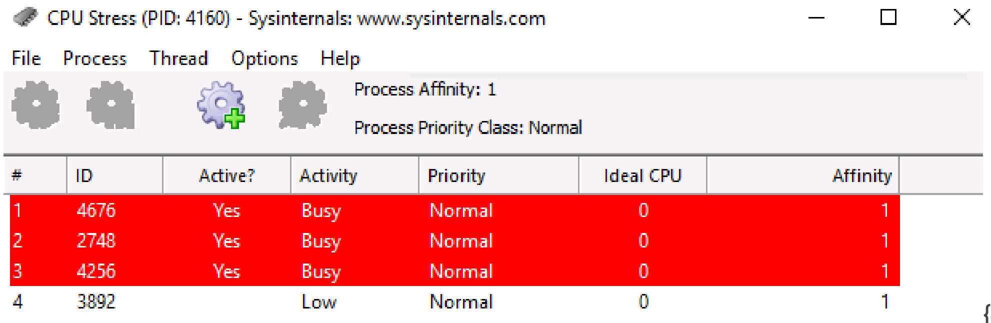
13. [] Click **RDP**, download the file and use the credential you used to create the virtual machine to log on.

14. [] Open a PowerShell window and run the code below to download the CPUSTRES application and start it.

```
Invoke-WebRequest -Uri http://download.sysinternals.com/files/CPUSTRES.zip -UseBasicParsing
-Outfile CPUSTRES.zip
Expand-Archive CPUSTRES.ZIP
.\CPUSTRES\CPUSTRES.exe
```

15. [] In CPU Stress, check the boxes to set Thread 1, Thread 2 and Thread 3 to **Active**.

16. [] For Thread 1, Thread 2 and Thread 3, Change the **Activity** drop-down to **Busy**.



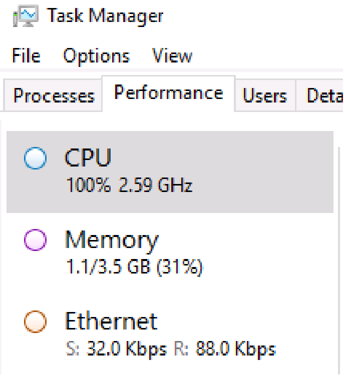
Process Affinity: 1

Process Priority Class: Normal

#	ID	Active?	Activity	Priority	Ideal CPU	Affinity
1	4676	Yes	Busy	Normal	0	1
2	2748	Yes	Busy	Normal	0	1
3	4256	Yes	Busy	Normal	0	1
4	3892		Low	Normal	0	1

{400}

17. [] If you look at Task Manager in the VM, you will notice high CPU Utilization.



CPU
100% 2.59 GHz

Memory
1.1/3.5 GB (31%)

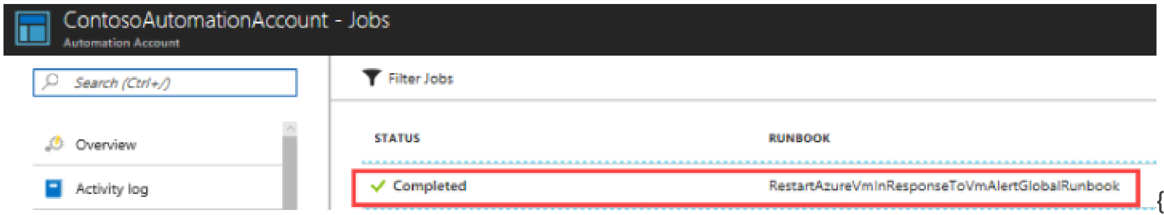
Ethernet
S: 32.0 Kbps R: 88.0 Kbps

{400}

18. [] After a few minutes, navigate to your **ContosoAutomationAccount** Automation Account in the Azure Portal.

19. [] From the Automation Account, click **Jobs**.

20. [] You should see a job **RestartAzureVMInResponseToVMAlertGlobalRunbook**. Click this job.



STATUS	RUNBOOK
✓ Completed	RestartAzureVmInResponseToVmAlertGlobalRunbook

{400}

21. [] In the Job, click **All Logs** in the Job. Scroll down in the logs, and you should see the Verbose message **Restarting the VM: IntegrationVM**.

Exercise 5: Configuring Source Control

Introduction

This lab will demonstrate how to configure Source Control, for your Azure Automation Runbooks.

Summary

In this lab, we will:

- Create a GitHub account

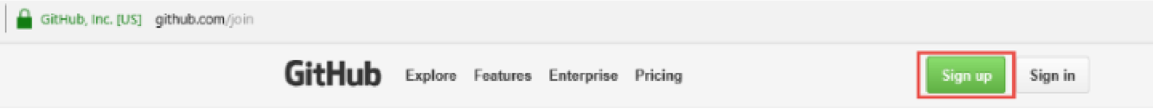
- Associate the GitHub account with Azure Automation

Estimated Time to Complete This Lab

20 minutes

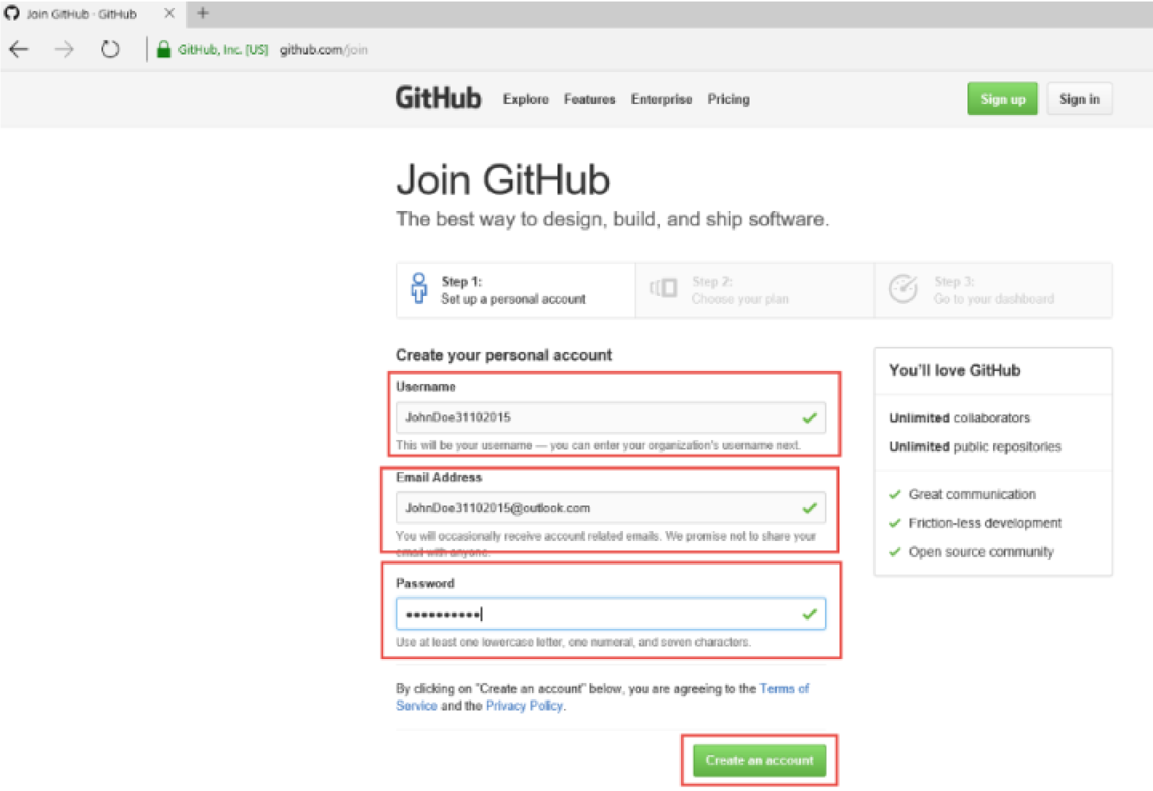
Task Description

1. [] First, we will need to create a GitHub account.
2. [] Browse to https://github.com/ and click **Sign up**. If you have an existing GitHub account and want to use it skip to step 8.



{400}

3. [] Type in the following:
   - **Username:** The first part of your Trial account name. For example, **johndoe31102015**
   - **Email address:** The trial accounts email address
   - **Password:** Password of your choice
4. [] Click Create an account.



Join GitHub

The best way to design, build, and ship software.

Step 1: Set up a personal account | Step 2: Choose your plan | Step 3: Go to your dashboard

Create your personal account

Username: JohnDoe31102015 ✓

This will be your username — you can enter your organization's username next.

Email Address: JohnDoe31102015@outlook.com ✓

You will occasionally receive account related emails. We promise not to share your email with anyone.

Password: ••••••••••• ✓

Use at least one lowercase letter, one numeral, and seven characters.

By clicking on "Create an account" below, you are agreeing to the Terms of Service and the Privacy Policy.

Create an account

You'll love GitHub

Unlimited collaborators

Unlimited public repositories

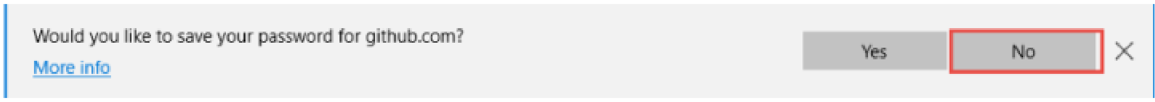
✓ Great communication

✓ Friction-less development

✓ Open source community

{400}

5. [] Select **No** if you get a request to save the password.



Would you like to save your password for github.com?

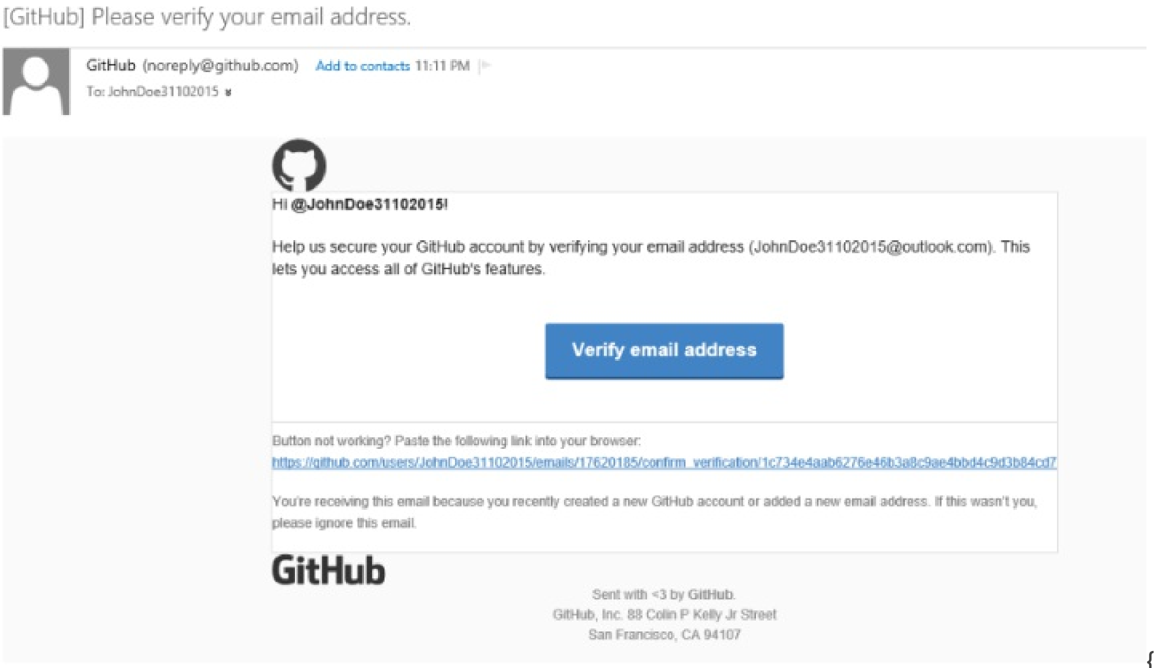
More info

Yes | No | ×

{400}

6. [] Select the **Free** tier as the chosen personal plan and click **Continue**, then **Submit**.

7. [] You *might* need to sign in to your email account and verify the account before proceeding to the next step (if required).



[GitHub] Please verify your email address.

GitHub (noreply@github.com) Add to contacts 11:11 PM
To: JohnDoe31102015

Hi **@JohnDoe31102015!**

Help us secure your GitHub account by verifying your email address (JohnDoe31102015@outlook.com). This lets you access all of GitHub's features.

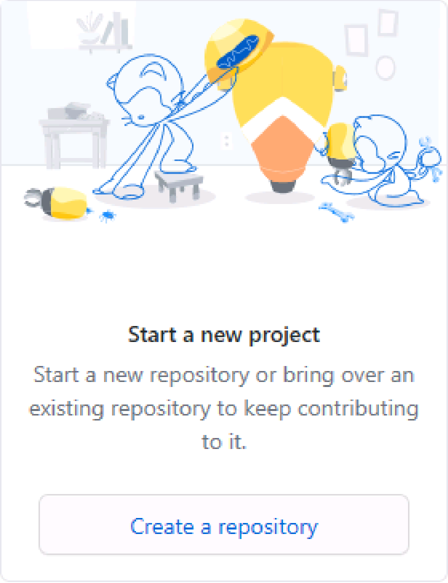
Button not working? Paste the following link into your browser:
https://github.com/users/JohnDoe31102015/emails/17620185/confirm_verification/1c734e4aab6276e46b3a8c9ae4bbd4c9d3b84cd7

You're receiving this email because you recently created a new GitHub account or added a new email address. If this wasn't you, please ignore this email.

Sent with <3 by GitHub.
GitHub, Inc. 88 Colin P Kelly Jr Street
San Francisco, CA 94107

{400}

8. [] Click **Create a repository**.



Start a new project

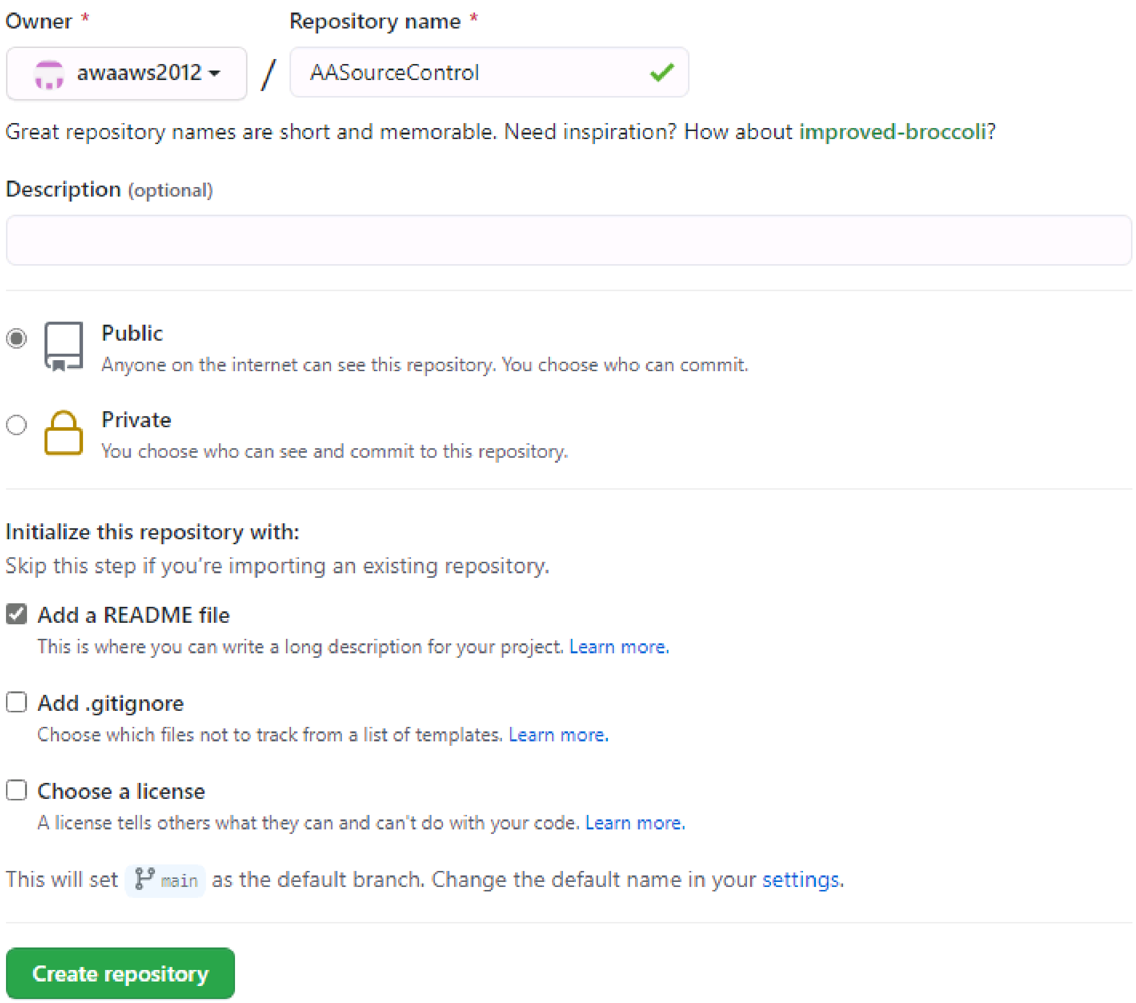
Start a new repository or bring over an existing repository to keep contributing to it.

{400}

9. [] In Repository name, type **AASourceControl**.

10. [] Select the **Initialize this repository with a README option**.

11. [] Click **Create** repository.



Owner *

awaaws2012 ▾ / Repository name * AASourceControl ✓

Great repository names are short and memorable. Need inspiration? How about improved-broccoli?

Description (optional)

● Public
Anyone on the internet can see this repository. You choose who can commit.

○ Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☑ Add a README file
This is where you can write a long description for your project. Learn more.

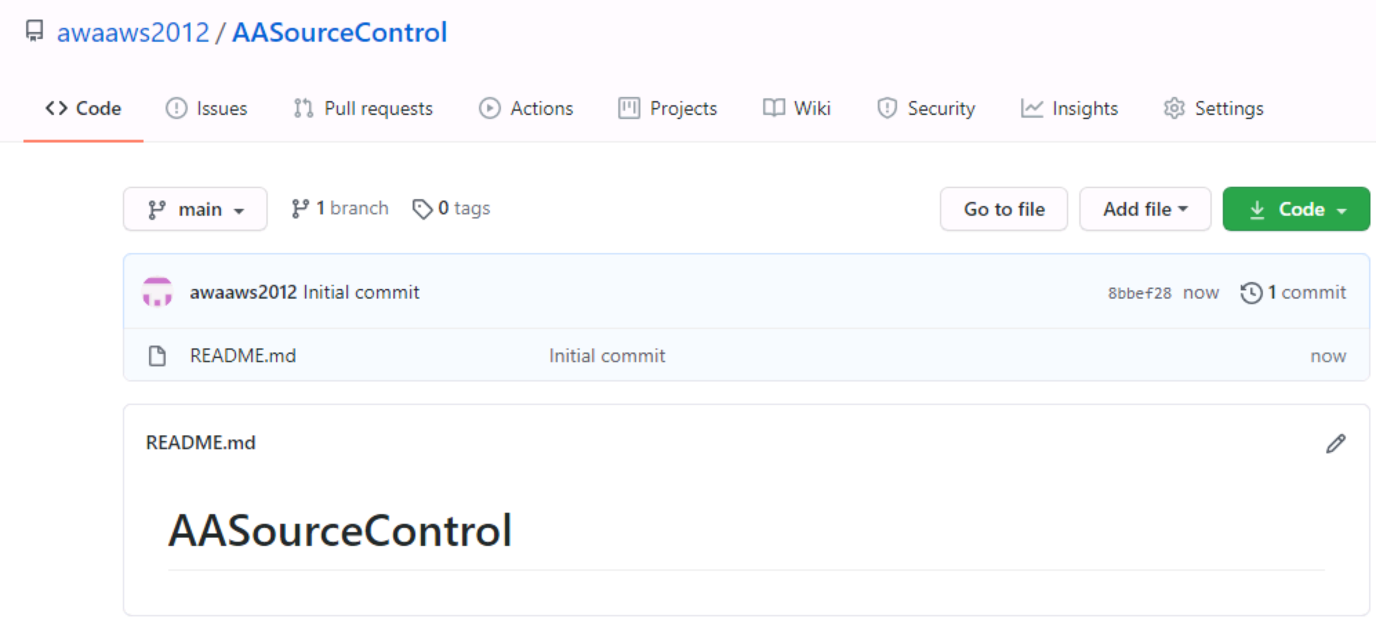
□ Add .gitignore
Choose which files not to track from a list of templates. Learn more.

□ Choose a license
A license tells others what they can and can't do with your code. Learn more.

This will set `main` as the default branch. Change the default name in your settings.

{400}

> You should now have a new Repository called **AASourceControl**



awaaws2012 / AASourceControl

awaaws2012 Initial commit 8bbef28 now 1 commit

README.md Initial commit now

README.md

AASourceControl

{400}

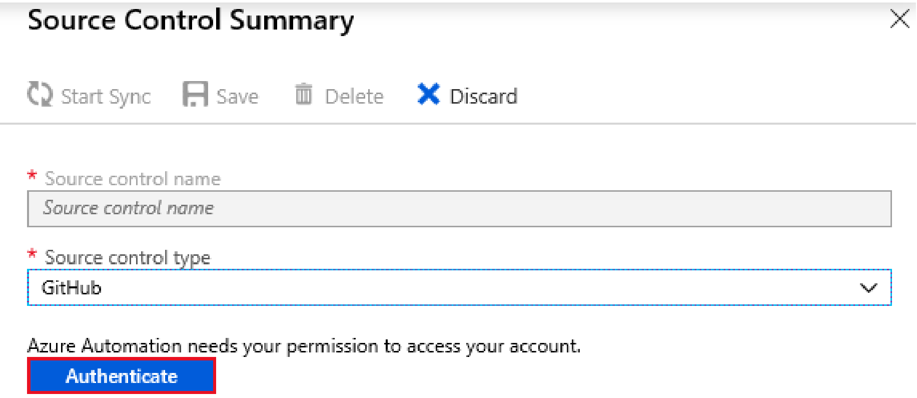
12. [] Sign back into https://portal.azure.com/ close any open blades and browse into the

ContosoAutomationAccount.

13. [] In the Automation account, click **Source control** under Account Settings.

14. [] Click **Add**

15. [] In **Source Control Type** choose **Github** and click on **Authenticate**



Source Control Summary

* Source control name: *Source control name*

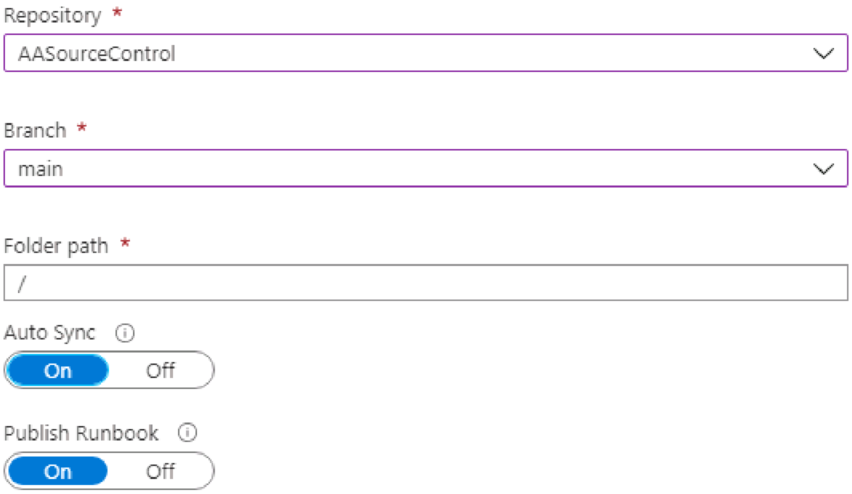
* Source control type: GitHub

Azure Automation needs your permission to access your account.

{400}

16. [] Following the prompts to continue the authentication process. You will need to sign in to Github and allow the Automation service access to your profile.

17. [] When successful you can select the **AASourceControl** repository. Select the *main* branch as below and click **Save**



Repository *: AASourceControl

Branch *: main

Folder path *: /

Auto Sync: On | Off

Publish Runbook: On | Off

{400}

Exercise 6: Source Control Check-in (Push)

Introduction

This lab will demonstrate how to synchonise runbooks from your Github account to Azure Automation

Summary

In this lab, we will:

- Create a Runbook in Github

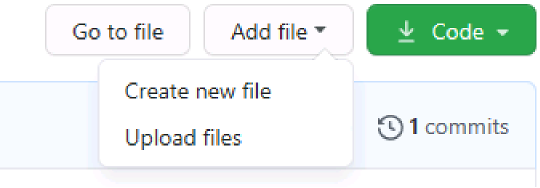
- Synchronise from Github to Azure Automation

Estimated Time to Complete This Lab

20 minutes

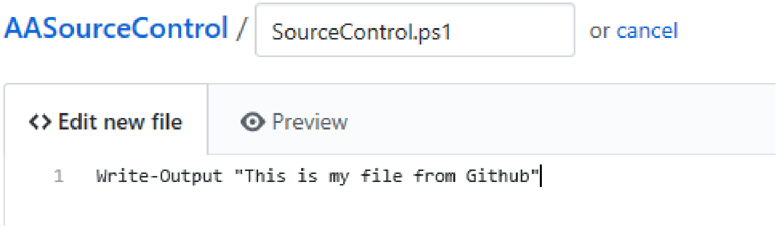
Task Description

1. [] Sign in to github.com using the github account created before and select your repository.
2. [] Click on **Add file** and from the dropdown select **Create new file**



{400}

3. [] Give the file a name and add some dummy code as below.

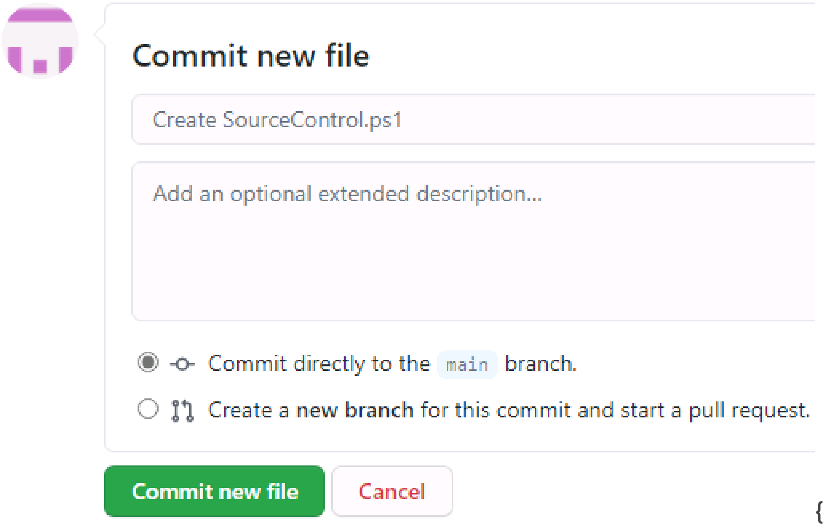


SourceControl.ps1

```
Write-Output "This is my file from Github"
```

{400}

4. [] Scroll down and click **Commit new file**


Commit new file

Create SourceControl.ps1

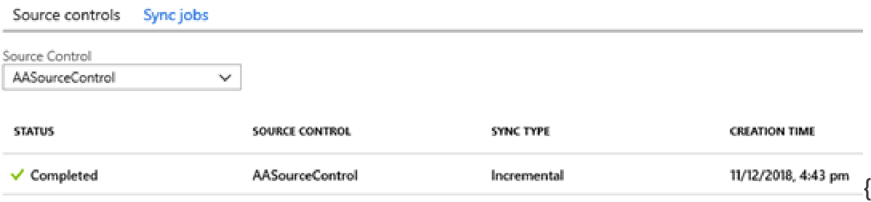
Add an optional extended description...

● Commit directly to the main branch.

○ Create a **new branch** for this commit and start a pull request.

{400}

5. [] When you do this - Github fires a webhook - which forces the automation service to perform a synchronisation.
6. [] In the Azure portal - select your Automation Account - select **Source Control** and click on **Sync Jobs**. You should see the job created which is synchronising content.

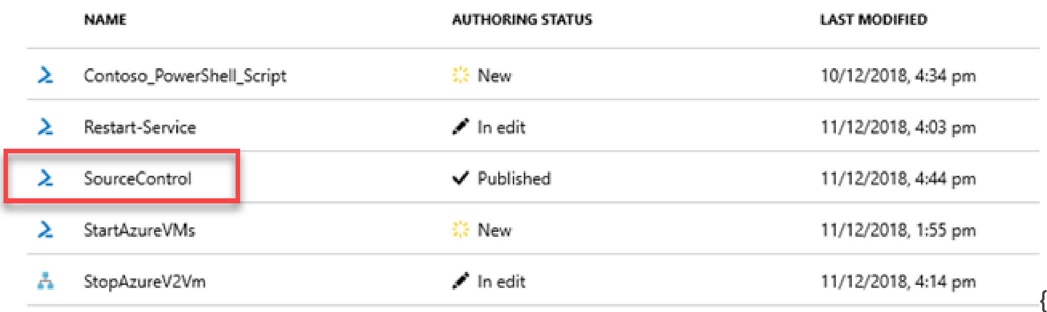


Source Control: AASourceControl

STATUS	SOURCE CONTROL	SYNC TYPE	CREATION TIME
✓ Completed	AASourceControl	Incremental	11/12/2018, 4:43 pm

{400}

7. [] When complete the Runbook will appear in your Automation Account under **Runbooks**



NAME	AUTHORING STATUS	LAST MODIFIED
Contoso_PowerShell_Script	New	10/12/2018, 4:34 pm
Restart-Service	In edit	11/12/2018, 4:03 pm
SourceControl	✓ Published	11/12/2018, 4:44 pm
StartAzureVMs	New	11/12/2018, 1:55 pm
StopAzureV2Vm	In edit	11/12/2018, 4:14 pm

{400}

> [!note]: The synchronisation is one way - so if you make changes to a runbook in Azure Automation it will not be reflected in Github. Also any changes that are made in the Automation Account editing pane will be overwritten during synchronisation.