WorkshopPLUS

Microsoft Azure Infrastructure as a Service

Calling a Microsoft Azure Automation Runbook with a Webhook

Student Lab Manual

V1.3, October 12, 2016

Information in this document is subject to change without notice. The example companies, organizations, products, people, and events depicted herein are fictitious. No association with any real company, organization, product, person or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarked, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2014 Microsoft Corporation. All rights reserved.

Microsoft, MS-DOS, MS, Windows, Windows NT, MSDN, Active Directory, BizTalk, SQL Server, SharePoint, Outlook, PowerPoint, FrontPage, Visual Basic, Visual C++, Visual J++, Visual InterDev, Visual SourceSafe, Visual C#, Visual J#,  and Visual Studio are either registered trademarks or trademarks of Microsoft Corporation in the U.S.A. and/or other countries.

Other product and company names herein may be the trademarks of their respective owners.

Contents

[Calling a Microsoft Azure Automation Runbook with a Webhook 4](#_Toc464020534)

[Prerequisites 5](#_Toc464020535)

[Task 1 – Create an Azure Automation Account 5](#_Toc464020536)

[Task 2 – Import an Automation Runbook 7](#_Toc464020537)

[Task 3 – Add the Webhook 10](#_Toc464020538)

[Task 4 – Call the Webhook from PowerShell 12](#_Toc464020539)

# Calling a Microsoft Azure Automation Runbook with a Webhook

In this lab, you will import a pre-existing Azure Automation runbook that will be used to start and stop Azure VMs. In order to kick off the job of running the automation runbook, you will use a Webhook. Once the webhook is setup, you will use PowerShell as a client application to call the webhook to execute the runbook.

You'll learn:

* How to create an Automation Account
* How to import, edit and publish an Automation Runbook
* How to create a Webhook
* How to call the Webhook through PowerShell

## Prerequisites

The following is required to complete this hands-on lab:

* A Microsoft Azure subscription
* One or more virtual machines running in the subscription that can be shutdown. You can create the VMs using the Azure portal or PowerShell. Please start this process **before** continuing on to Task 1. Put the VM(s) in to any resource group name of your choosing.

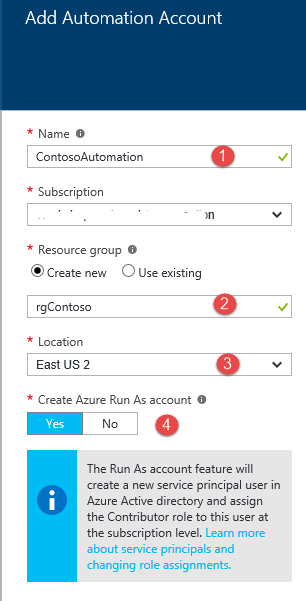
## Task 1 – Create an Azure Automation Account

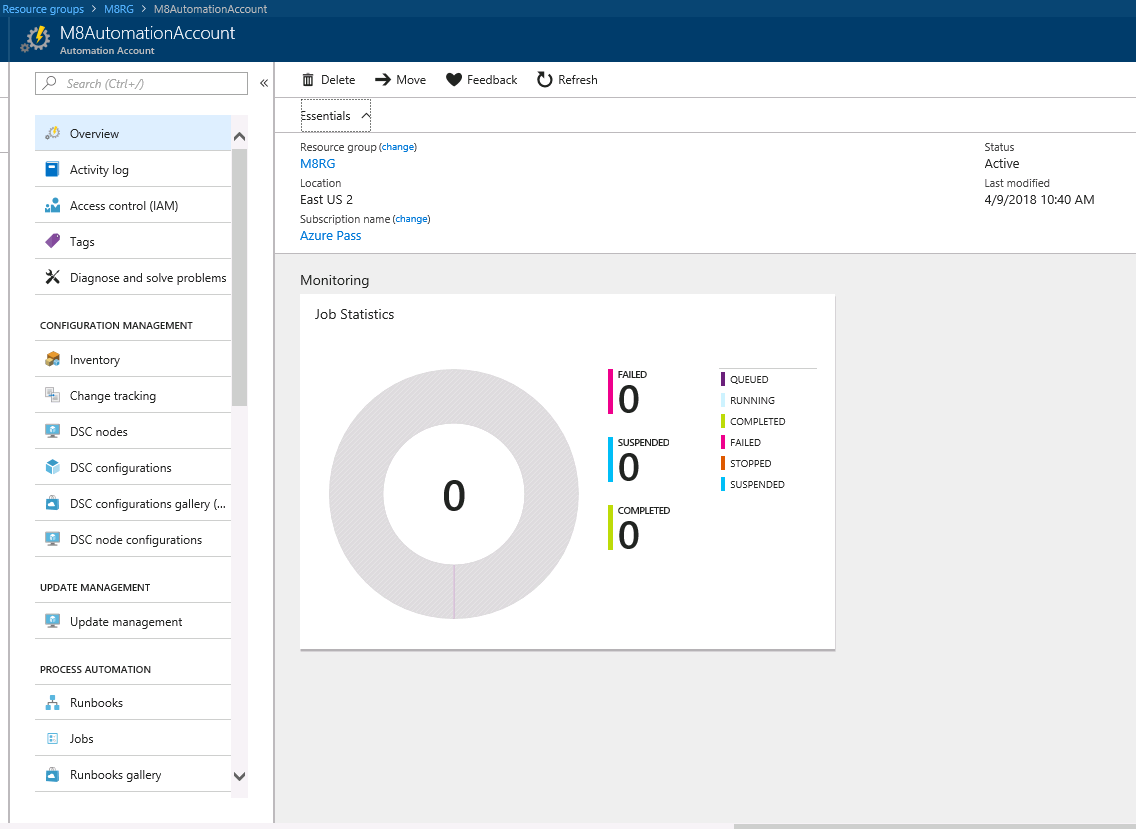
Please see the prerequisites listed above and create the required VM before continuing.

NOTE: If you have already created an Azure Automation account from previous exercises, you can use that same automation account for this lab. Skip to **Task 2** if you have already setup an automation account and want to use that account.

To begin using Azure Automation, the first thing you need is an Automation Account. Automation accounts securely contain the resources that configure and run your runbook jobs. An Automation Account is a logical container and security boundary for all automation assets (runbooks, connections, etc.). Assets within an Automation account are visible only to other assets within the account. An Automation Account also provides affinity to a region, which helps with data sovereignty concerns (if your runbook IP or data can't leave a region).

1. Log in to the Azure Management Portal at https://portal.azure.com.
2. Select **+ New | Monitoring + Management |** **Automation**.
3. Configure the Automation Account as follows:
   1. Enter a name for your automation account (unique to your subscription).
   2. Create a new resource group or use an existing one.
   3. Select a region.
   4. Leave the *Create Azure Run As account* setting as is.

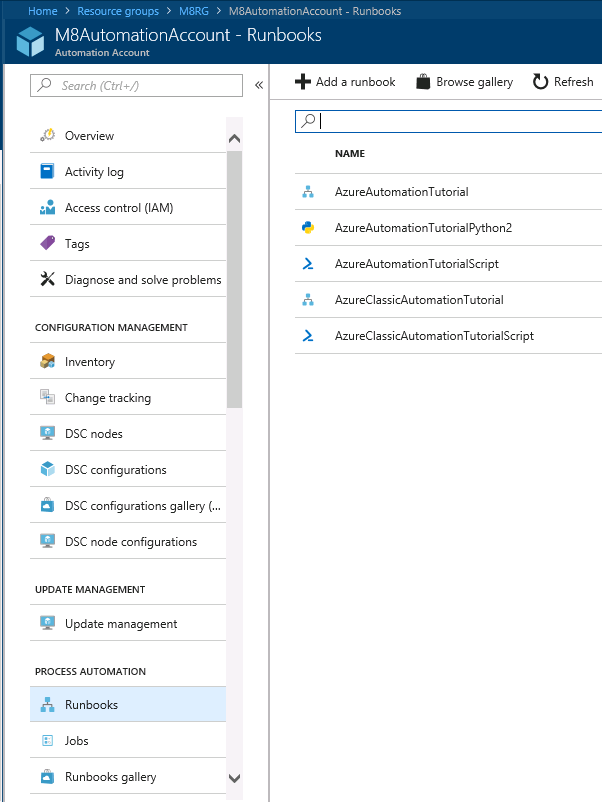


1. Click the **Create** button.
2. Once the automation account has been created, you can use the *Resource groups* menu item to find your new resource group and then select your automation account.
3. The view of your automation account should look something similar to this:  
     
   

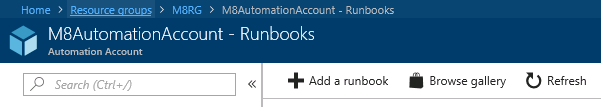
## Task 2 – Import an Automation Runbook

A Runbook is the central concept in Azure Automation. Runbooks are PowerShell Workflow scripts that contain the steps you want to automate. When you execute a Runbook, it is queued for execution as a job. The Automation service picks up the job, runs it, and records status. You do not have control over the Automation service. It is part of the Automation infrastructure and managed by Azure.

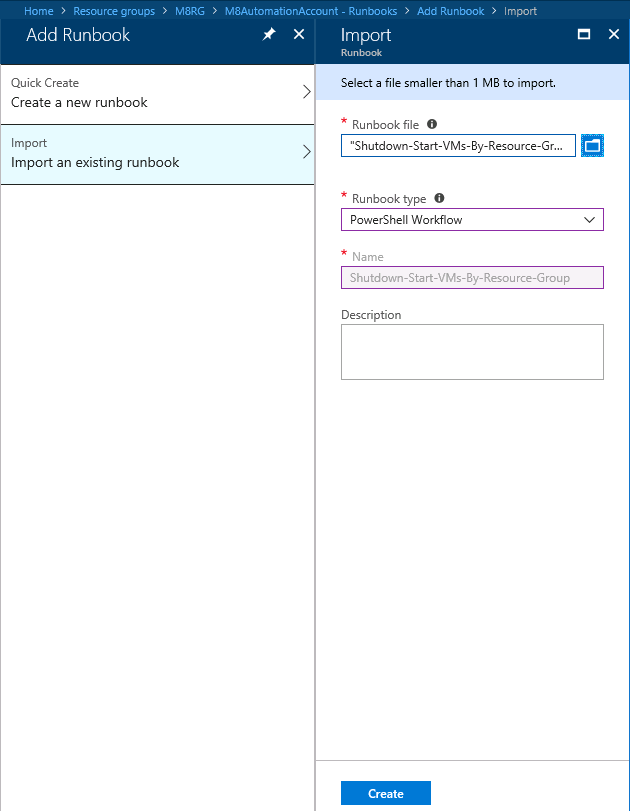
1. If necessary, navigate to your *Automation Account* blade.
2. Under PROCESS AUTOMATION, click on **Runbooks**.



1. Click on the **+Add a runbook** button.

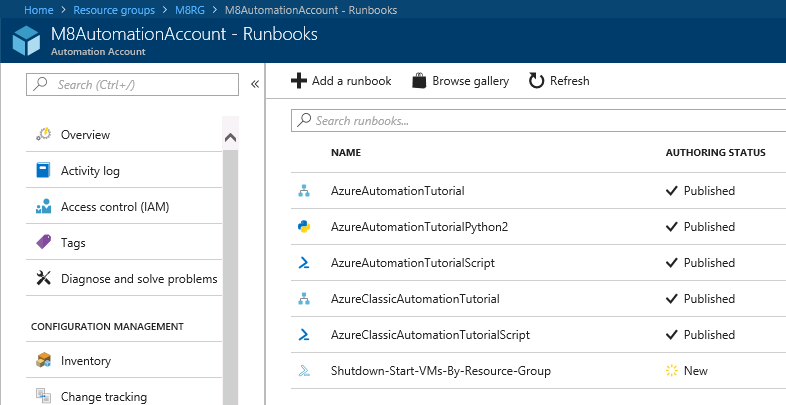


1. In the *Add Runbook* blade:
   1. Select the **Import** **an existing runbook** menu item
   2. Browse to the **C:\AzureIaaSWS\M8 - Azure Automation\Labs\AutomationUsingWebhook\Assets** folder and find the **Shutdown-Start-VMs-By-Resource-Group.ps1** file.
   3. Leave the other settings within the import blade as they are
   4. Click the **Create** button.

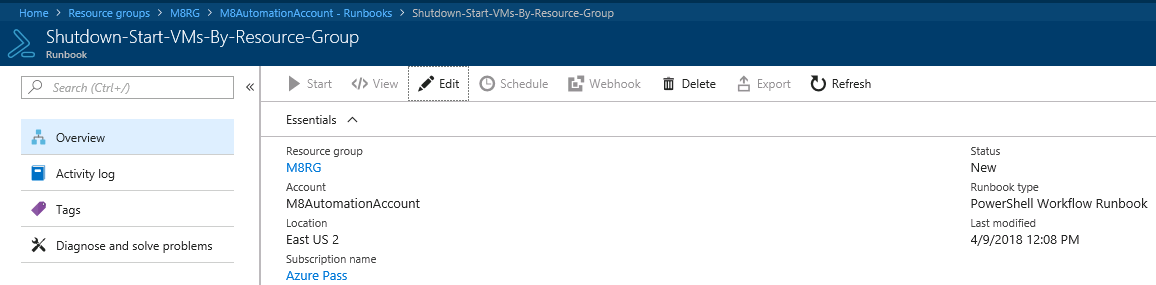


1. Once the runbook has been imported, click on the name of the runbook within the *Runbooks* blade to open the specific runbook blade.

Notice that the *Authoring Status* is set to **New**, meaning that the runbook has not been published yet and therefore it cannot be executed unless through testing.

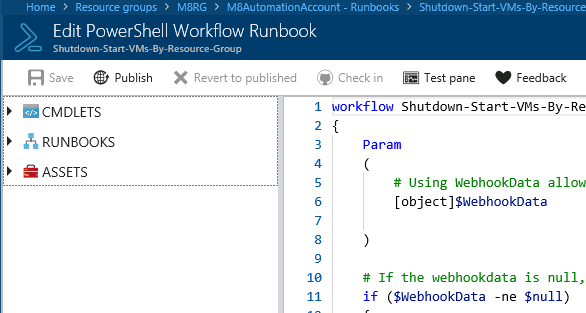


1. From within the *Runbook* blade, click the **Edit** button. This will take you to the view where you can publish your workbook.



1. Review the commented code in the runbook. Once you are finished reviewing, select the **Publish** button on the toolbar.

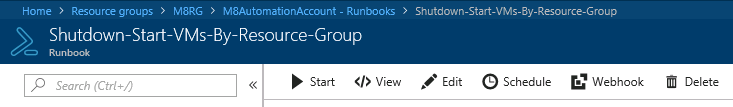
NOTE: If we were not using a Webhook, you could click on **Test Pane** to test the runbook, but in our case, we specifically have coded the runbook to only allow a Webhook to be used.



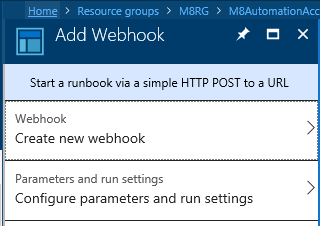
## Task 3 – Add the Webhook

A webhook allows you to start a runbook in Azure Automation through a single HTTP POST request. This allows external services such as Visual Studio Team Services, GitHub, or customer applications to start runbooks without implementing a full solution using the Azure Automation API.

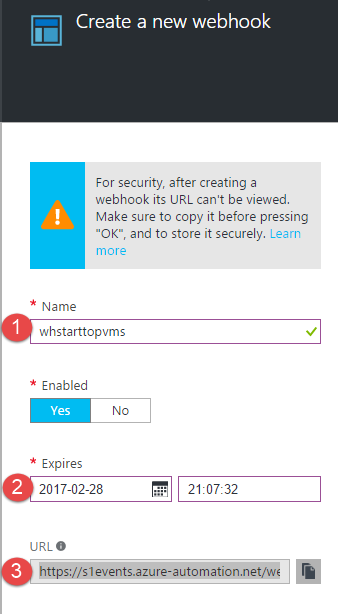
1. Within the *Runbook* blade, select the **Webhooks** icon.



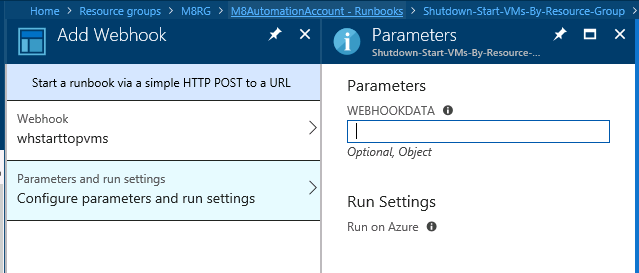
1. On the **Add Webhook** blade, select **Create new webhook**.



1. Configure the webhook as follows and then click the **OK** button:
   1. Enter in a name for your webhook
   2. Enter the expiration date
   3. MOST IMPORTANTLY, copy the URL to the clipboard or paste in to Notepad. If you do not do this and close this blade, you will not be able to retrieve the URL and you will have to create a new webhook!



1. Click on the **Configure parameters and run settings** link. For our purpose, we do not need to put any data in to the *WebhookData* field; this will be passed in via *PowerShell*.



1. Click the **OK** button.
2. Click the **Create** button in the *Add Webhook* blade.

## Task 4 – Call the Webhook from PowerShell

There are multiple ways to actually call a webhook. Any language or API that can implement an HTTP POST call, can call the webhook. For the purposes of this lab, the client application will be a PowerShell script.

1. Open **PowerShell ISE** as an Administrator.
2. Open the script **C:\AzureIaaSWS\M8 - Azure Automation\Labs\AutomationUsingWebhooks\Assets\Client.ps1**.
3. Replace the existing URL with the URL that you copied from when you created the webhook.
4. Replace the existing Resource Group name with the name of the resource group that contains your VM(s).

NOTE: The $headers variable contains a From and Date field, but you can put anything in here you wish to since we don’t use this in our script.

1. Your end script should look something like this (with the highlighted items edited to reflect your values):  
     
   $uri = "https://s1events.azure-automation.net/webhooks?token=plZ%2bQdMRRxVJxUysxxxxxxxxxxxxxxxx3d"

$headers = @{"From"="user@contoso.com";"Date"="02/23/2016 15:47:00"}

$myvars = @(

@{AzureResourceGroup="rgMyVM";Shutdown="true" }

)

$body = ConvertTo-Json -InputObject $myvars

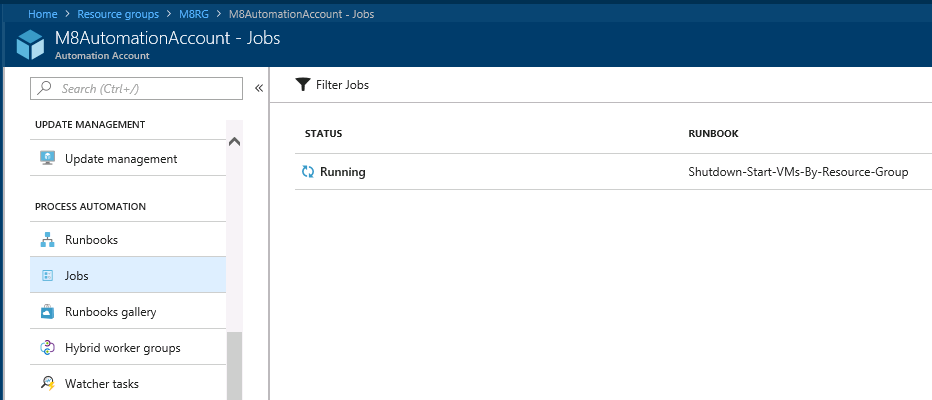
$response = Invoke-RestMethod -Method Post -Uri $uri -Headers $headers -Body $body

Write-Output $response.JobIds

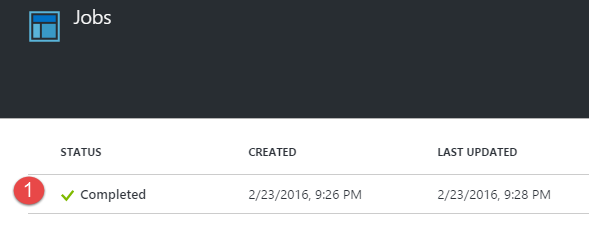
1. Put a breakpoint on the first line of code and save the file. Select **F5** to begin stepping through the code.
2. When you move past the last line of code, you will see the job number output to the PowerShell command window.



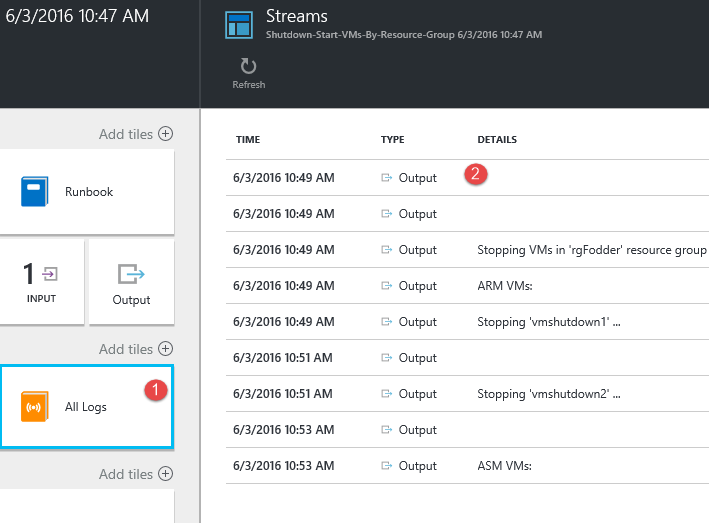
1. Navigate to the *Automation Account* blade, and select on the **Jobs** node.



1. The job status will either be *Queued, Starting, Suspended, Running* or *Completed*.



1. Click on the job status after it has completed. If it has been suspended or stopped, you can also click on it to see what has happened.
2. In the *Jobs* status blade:
   1. Click on the **All Logs** icon.
   2. This will open up a **Streams** blade where you can click on each individual output that occurred which the script was executing.



For example, if you click on the top item, you should see something like:

