|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | | |
|  |  |  |  | |
| PowerShell CI/CD Tools |  | PowerShell |  |  |
|  |  |  |  |
|  |  |  | **Hands-on lab** |
|  |  |  |  | |
| PowerShell Desired State Configuration can be a core component of the Continuous Integration / Continuous Deployment pipeline. Come build and run an end-to-end pipeline using PowerShell features and tooling. You will leave this ILL with hands on experience in building and running a PowerShell based pipeline. |  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  | | |

This document supports a preliminary release of a software product that may be changed substantially prior to final commercial release. This document is provided for informational purposes only and Microsoft makes no warranties, either express or implied, in this document. Information in this document, including URL and other Internet Web site references, is subject to change without notice. The entire risk of the use or the results from the use of this document remains with the user. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in examples herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, 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, trademarks, 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.

Copyright 2016 © Microsoft Corporation. All rights reserved.

Microsoft ***<list other MS trademarks used alphabetically>*** are trademarks of the Microsoft group of companies.

***<This is where mention of specific, contractually obligated to, third party trademarks should be listed.>***

All other trademarks are property of their respective owners.

# Introduction

### Estimated time to complete this lab

90 minutes

### Objectives

After completing this lab, you will be able to:

* Learn how to implement a Continuous Integration pipeline with PowerShell
* Utilize the latest PowerShell tooling

### Prerequisites

Before working on this lab, you must have:

* an understanding of Continuous Integration and Continuous Deployment concepts
* an understanding of Git source control concepts
* experience with PowerShell scripting

### Overview of the lab

PowerShell can be a core component of the Continuous Integration / Continuous Deployment pipeline. Come build and run an end-to-end pipeline using PowerShell features and tooling. You will leave this ILL with hands on experience in building and running a PowerShell based pipeline.

### Scenario

In this lab you will configure a Continuous Integration (CI) pipeline using Visual Studio Team Foundation Server and the MyGet online services. Once the pipeline is configured, you will use it to fully build, test, and deploy a PowerShell module to a private PowerShellGet feed. Although this Lab will not cover it, the same process can be extended to take the changes through a test and production for managing infrastructure environments as well.

### Reference material

* [The Release Pipeline Model](http://releasepipeline.blob.core.windows.net/pdf/The%20Release%20Pipeline%20Model.pdf): This document provides a great framework and defines all of the concepts use in this lab.
* [Team Foundation Server](https://www.visualstudio.com/products/visual-studio-team-services-vs?utm_source=Google&utm_medium=CPC&utm_term=Branded&utm_campaign=VSTS%20EvergreenWT.srch=1&WT.mc_id=SEM_xeWGA7kH): Used as source code repository as well as build system for release pipeline in this lab.

# Exercise 1: Setup Work Environment

In this first exercise, you will setup a VSTS account and configure the build definitions.

## Task 0 – Setup free accounts in VSTS and MyGet

In this task, you will setup two accounts in free online services

\*\*Note\*\* - This requires a (free) Microsoft account. (https://account.microsoft.com/)

* Task instructions

1. In a browser, open https://www.visualstudio.com
2. Click Free Visual Studio
3. In the Visual Studio Team Services box, Click Free Account
4. Sign in using your Microsoft Account
5. Enter a unique name for your project and select the radio button next to Git (this is the default), then click Continue.

* Task instructions (please use a new browser tab and do not close the VSTS tab)

1. In **a new browser tab**, open https://www.myget.org
2. Click Sign Up
3. Fill out the form with your unique name, email, and password
4. Check your Email and follow the link to your new account
5. Click on the link 
6. Provide a name for the feed such as psrepository and a description, or leave the default description
7. Click on the link 
8. On the resulting page, click Feed Details, you will need this information in the next step

* IMPORTANT: Do not close the browser. You will need it in the following exercises and tasks.

## Task 1 – Clone code from GitHub in to your new project

In this task, you will import an example PowerShell module stored in a public GitHub repository.

* Task instructions   
  (if you closed the first browser tab, open a new tab and navigate to https://youruniquename.visualstudio.com)

1. Continuing in the first browser tab, expand the text “or import a repository” and click Import
2. In the Source Type window, keep the default choice, Git. In the Clone URL window, type https://github.com/PowerShell/Demo\_CI/

The browser window should automatically redirect to the Code tab

1. Click on the  button to select a different branch, select All branches and then module-example

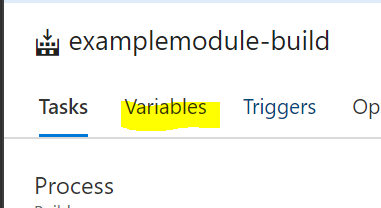
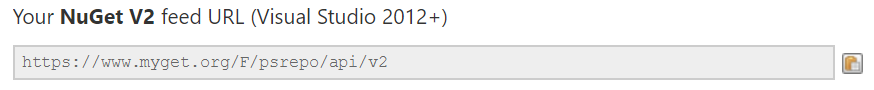
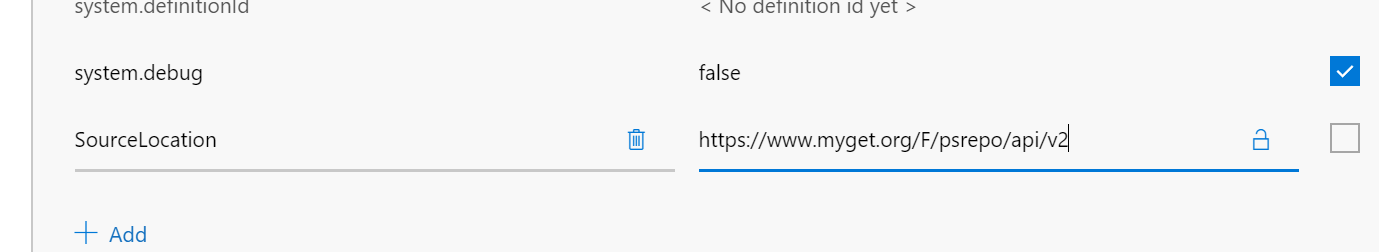
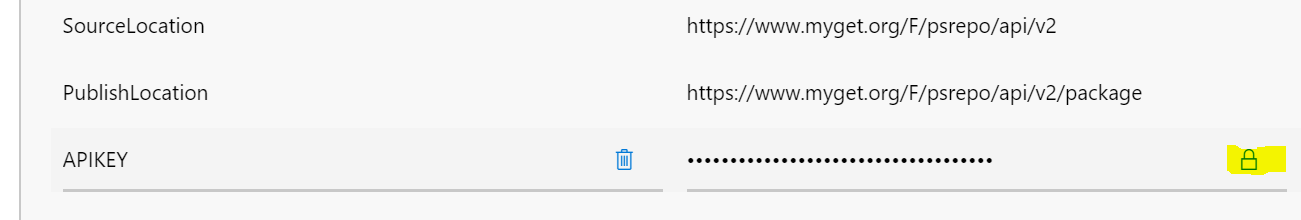
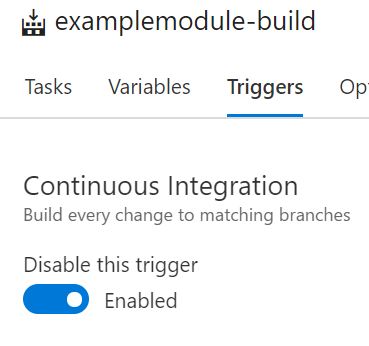
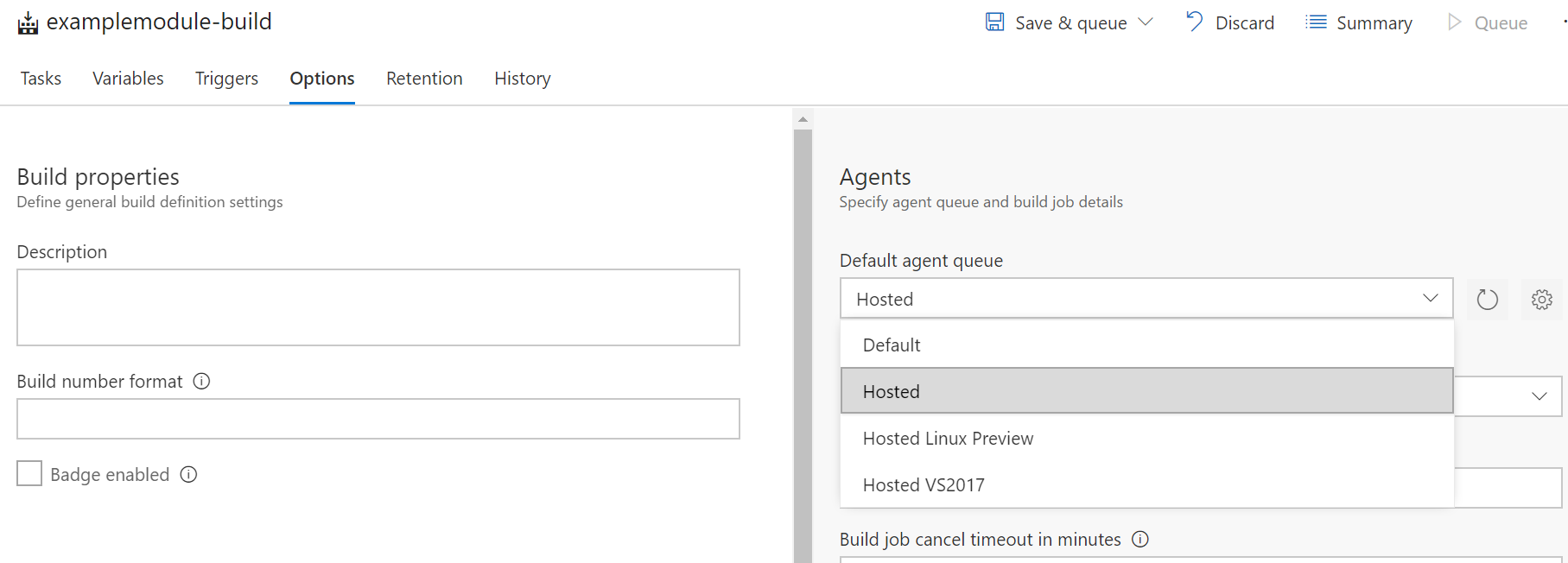
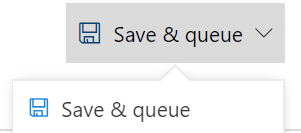
The code changes to the *ExampleModule* file content automatically

* IMPORTANT: Do not close the browser. You will need it in the following exercises and tasks.

## Task 2 – Create and configure Build Definition

In this task, you will configure VSTS with a build definition.

* Task instructions

1. Continuing in the first browser tab, click the Build & Release tab in the blue ribbon at the top of the page
2. Click the button 
3. Click on the link in the left pane, empty process
4. Type in a build name, examplemodule-build
5. The Get Sources item can remain at default, but observe the source can be other source control platforms including GitHub
6. Click on Add Tasks, then on the right pane select Utility, scroll down to PowerShell. Select it and click Add
7. In the left pane, select the PowerShell Script task and then in the right pane, next to the Script Path field click 
8. Select VSTS.ps1 and click OK
9. In the Arguments field, type ExampleModule
10. Next, in the tabs available for the Build definition, click Variables (see image)  
    
11. In the right pane, click 
12. In the Name column, type SourceLocation, in the Value column, paste in the string from the MyGet browser tab under Feed Details and then NuGet v2 Feed URL (see image)  
    *From your MyGet feed, on the Feed Details tab:*  
     *In VSTS:*
13. Complete this step again, but for the Name column type in PublishLocation and for the Value column paste in the value from the MyGet page under Push NuGet Packages to
14. Finally, add a third variable with the name APIKEY and copy the value from the MyGet feed details in the API key section. Mouse over the lock icon and enable it, which makes this a secure value. (see image)  
    
15. Next, navigate to the Triggers tab of the build definition and enable Continuous Integration (see image)  
    
16. Then on the Options tab, in the right pane under Agents, select Hosted (see image)  
    
17. Finally, click Save & Queue, and then in the drop-down click again on Save & Queue (see image)  
    
18. On the last prompt, leave the default values and click 

Before moving on, wait for the Build to complete and observe the output in VSTS. Refresh the tab for MyGet and verify the module was successfully published.

* IMPORTANT: Do not close the browser. You will need it in Exercise 2.

## Task 3 – Use PowerShellGet with your own Module Feed

In this task, you will configure PowerShellGet to use your own feed of PowerShell modules

* Task instructions

1. Open a PowerShell console as administrator (*Windows PowerShell or PowerShell Core 6 beta*)
2. In the browser, on tab with the MyGet page
3. Click on the Packages link in the left navigation
4. Click on the ExampleModule package
5. Click on PowerShellGet
6. Select the text in the box and use Ctrl-C to copy
7. In the PowerShell console window, right-click to Paste the text, and press Enter to execute the last command
8. Type Get-Example1 and press Enter to verify the module was installed and commands are now available

* OPTION: Do not close the PowerShell console. You can use it again in Exercise 2.

# Exercise 2: Edit files locally and publish using Git

In this exercise, you will bring the project files to your local machine, make changes, and release an update to your feed.

## Task 1 – Clone the project to your workstation

In this Task, you will use Git to clone the module to your workstation.

* Task instructions

1. In the browser tab for VSTS, click on Code inside the blue ribbon at the top of the page
2. Click on  at the top-right corner
3. Click the Copy icon to copy the full HTTPS path to your clipboard (see image)  
   
4. Open PowerShell (*Windows PowerShell or PowerShell Core 6 Beta*)
5. Create a working folder such as C:\Git
6. Using the Git command, clone the project from VSTS to your workstation  
   *example, this command will create a new folder and download the project:*

git clone https://youruniquename.visualstudio.com/\_git/MyFirstProject

1. Open the project in Visual Studio Code  
   *example, this command will open the folder in VSCode*

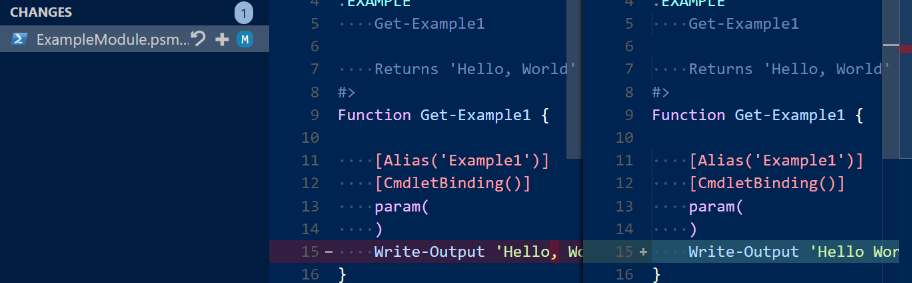
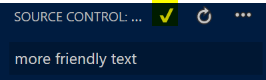
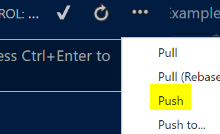
code-insiders .\MyFirstProject

1. Click on the  icon in VSCode
2. Expand the ExampleModule folder
3. Edit line 15, to change

* Write-Output ‘Hello, World’

to

* Write-Output ‘Hello World!’

1. Type Ctrl-S to save
2. Click on the  icon
3. Notice the change is recognized by VSCode/Git. Click on the change to see the changed line. (see image)  
   
4. In the Message field, type ‘more friendly text’
5. Click on the checkmark icon (see image)  
   
6. Click on the  icon, and in the drop down menu click Push (see image)  
   
7. In the browser, in the Build & Release view, verify a new build is automatically running
8. Click in the second tab of the browser to and navigate to the Packages section in MyGet to verify a new package is automatically published

* OPTION: Do not close the Visual Studio Code. You can use it again in further tasks.

## Task 2 – Install the new version of your module

In this task, you will install the new version of your module and verify the functions are udpated.

* Task instructions

1. Open a PowerShell console as administrator or use the existing window if it is still open (*Windows PowerShell or PowerShell Core 6 beta*)
2. Type

* Update-Module -Name ExampleModule

1. Type Get-Example1 and press Enter to verify the module now has the changed output.

**Congratulations**. You have successfully configured and run a Continuous Integration / Continuous Deployment pipeline for a development environment. Now take these concepts forward and build your own CI pipelines.