Configuring Builds as Code with YAML in Visual Studio Team Services

Lab version: 15.7.3

Last updated: 6/17/2018

1. **TABLE OF CONTENTS**

[Overview 1](#_Toc517009916)

[Prerequisites 1](#_Toc517009917)

[Exercise 1: Configuring Builds as Code with YAML in Visual Studio Team Services 1](#_Toc517009918)

[Task 1: Configuring the Parts Unlimited project 1](#_Toc517009919)

[Task 2: Adding a YAML build definition 3](#_Toc517009920)

[Task 3: Crafting sophisticated YAML definitions 5](#_Toc517009921)

# Overview

Many teams prefer to define their builds using YAML (Yet Another Markup Language). This allows them to access the same build pipeline features as those using the visual designer, but with a markup file that can be managed like any other source file. YAML build definitions can be added to a project by simply adding their source file to the root of the repository.

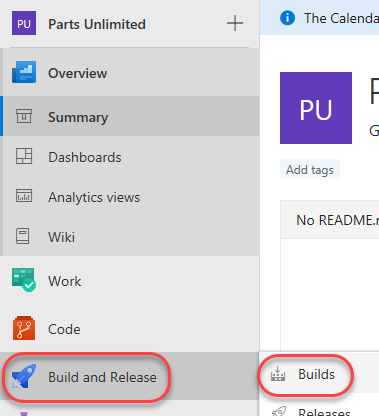
## Prerequisites

* This lab requires you to complete task 1 from the prerequisite instructions.
* YAML builds were a preview feature at the time this lab was drafted. Use the instructions at <https://docs.microsoft.com/en-us/vsts/collaborate/preview-features> to see if you need to enable this preview feature in your VSTS account.

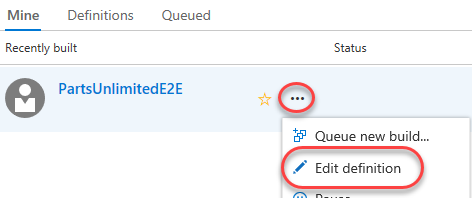
# Exercise 1: Configuring Builds as Code with YAML in Visual Studio Team Services

## Task 1: Configuring the Parts Unlimited project

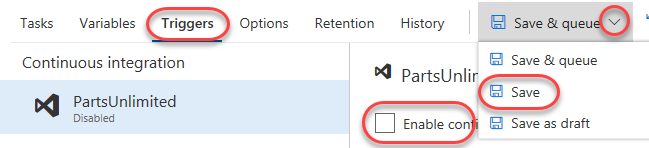
1. Navigate to your team project on VSTS. Before digging into the YAML builds, you will want to disable the continuous integration trigger for the default build definition.
2. Navigate to **Build and Release | Builds**.



1. This project includes a default build definition, **PartsUnlimitedE2E**. From its context dropdown, select **Edit definition** to edit it.

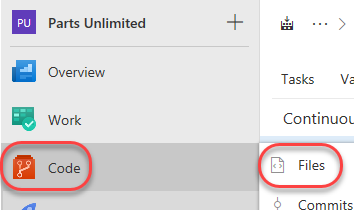


1. This build is scheduled to run whenever a change is committed. We don’t want to wait for this build every time there’s a change, so select the **Triggers** tab and uncheck **Enable continuous integration**. From the **Save & queue** dropdown, select **Save**.

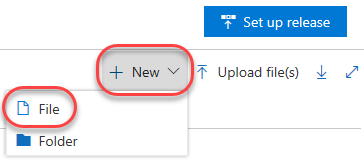


## Task 2: Adding a YAML build definition

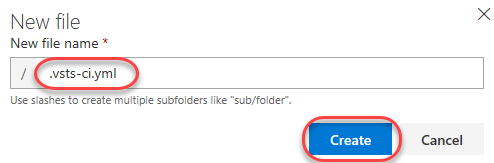
1. Navigate to the project files using **Code | Files**.



1. Add a new file to the root of the repository using **New | File**. YAML build definition are simply code files you add to your project. They offer all the benefits of other source files.



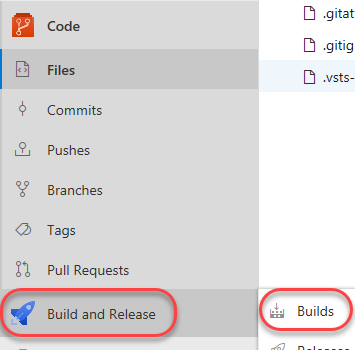
1. Name the new file **“.vsts-ci.yml”** and click **Create**. To make it more convenient to create YAML build definitions, VSTS automatically creates a definition when you add this file to the root of your repository. The build definition is created in a folder that has the same name as the repository.



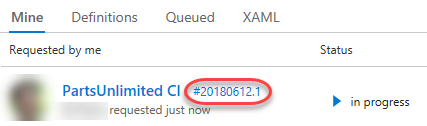
1. Add the YAML markup below to the file and click **Commit**. This simply instructs the build to execute the echo at the command line so we can confirm it’s all working. Once the commit has completed, it will queue a new build for this definition.
2. steps:
3. - script: echo hello world



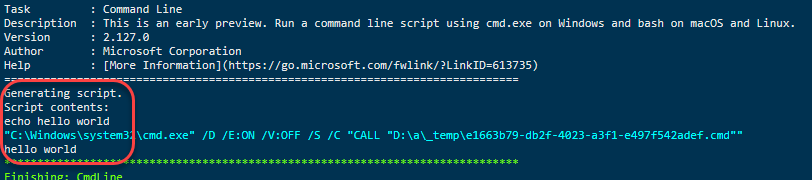
1. Select **Build and Release | Builds** to navigate to the build hub.



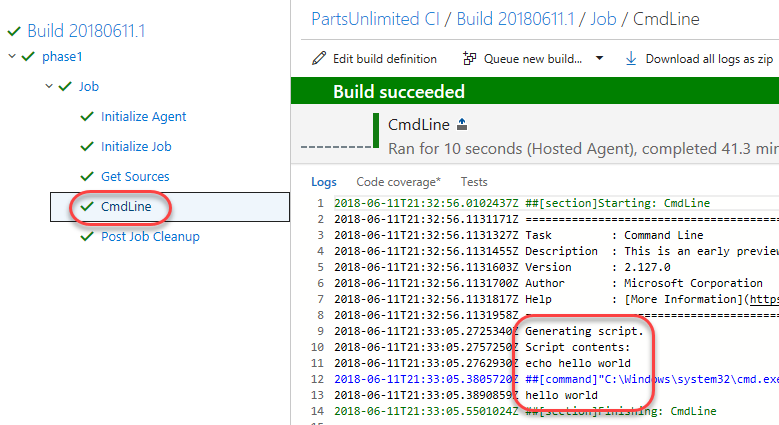
1. The newly created **PartsUnlimited CI** will run using the YAML settings. Click the build to follow its progress.



1. Track the second build until it completes. Near the end of the log you’ll see the part that generates and executes the script defined in the YAML file.

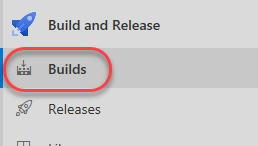


1. If the build has already succeeded by the time you get here, you can zero in on the script by selecting the **CmdLine** task.

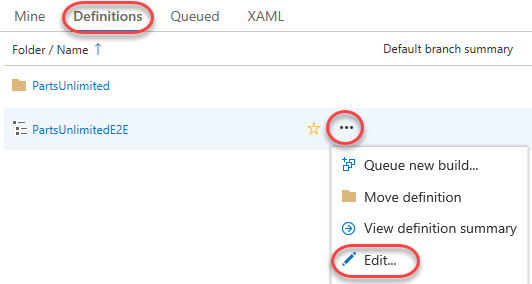


## Task 3: Crafting sophisticated YAML definitions

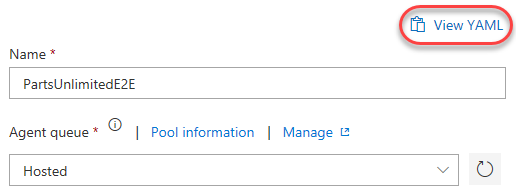
1. You can do virtually anything with YAML that you would do with the web experience. In fact, you can even export existing build definitions as YAML if you would prefer to manage them in code. Select **Builds** to return to the hub.



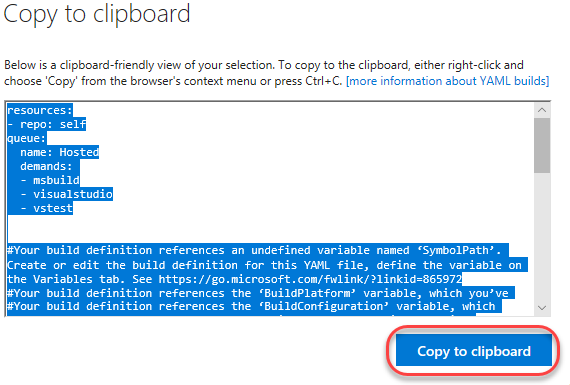
1. From the **Definitions** tab, expand the **PartsUnlimitedE2E** context menu and select **Edit**.



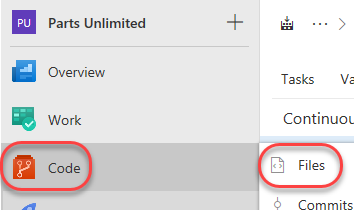
1. Click **View YAML** to see the YAML export of this definition.



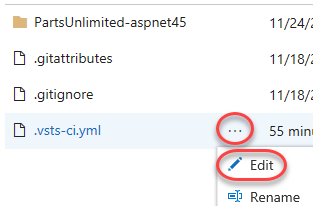
1. Click **Copy to clipboard**. We’ll take this as-is and use it for a YAML definition.



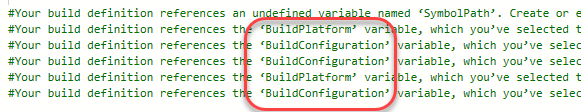
1. Navigate to the project files using **Code | Files**.



1. Open **.vsts-ci.yaml** in the **Editor**.



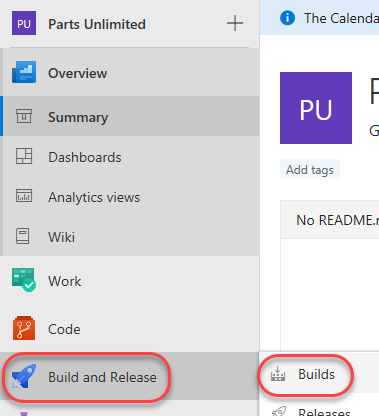
1. Replace the contents of the file with those on your clipboard.
2. Note that there is a comment block early in the file that reminds us that this definition is expecting some variables. The **SymbolPath** reference is coincidental (it’s a literal needed for the build), but the **BuildPlatform** and **BuildConfguration** variables will play a role later.



1. **Commit** the change back to the repo. This will queue a build.



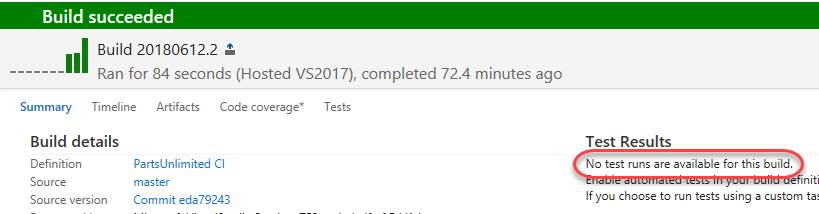
1. Select **Build and Release | Builds** to return to the build hub.



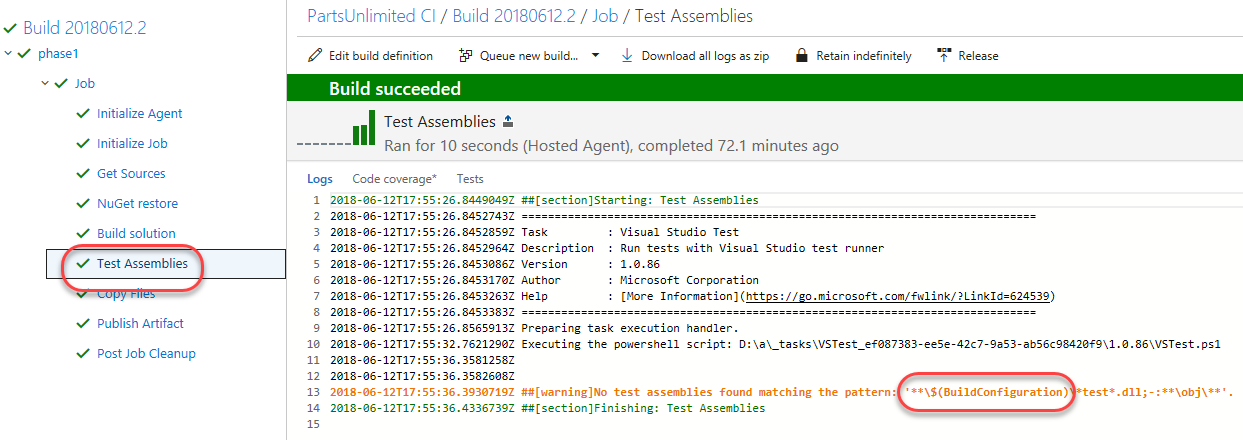
1. Click the new build to follow its progress.



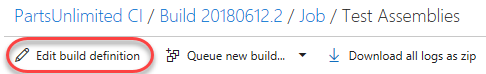
1. The build should succeed, but note that it reports that there were no tests. There should have been, so now we’ll need to figure out why they weren’t picked up.



1. Select the **Test Assemblies** task from the left panel. Note that there is a warning that the test assembly pattern didn’t find any matches. Upon closer inspection, it appears that it was expecting a build variable for **BuildConfiguration** to be available, but since it wasn’t, the platform simply used the text as-is. This was something we probably should have expected given the warnings in the YAML.



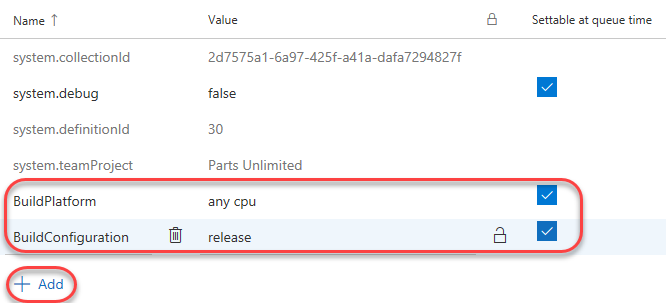
1. Click **Edit build definition**.



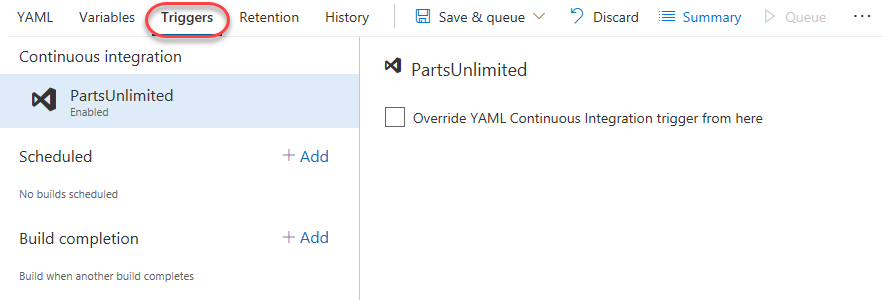
1. Select the **Variables** tab. Although the definition itself is in YAML, we can still manage things like variables and triggers externally.



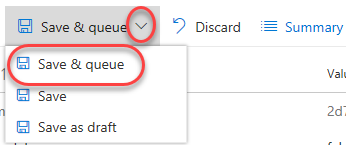
1. Use the **Add** button to add two new variables. **“BuildPlatform”** should be **“any cpu”** and **“BuildConfiguration”** should be **“release”**. Check the **Settable at queue time** options for each.



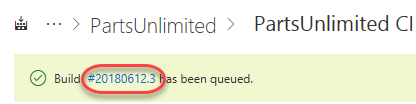
1. Select the **Triggers** tab. Note that you can override the YAML settings and configure continuous integration like web-based builds.



1. Select **Save & queue | Save & queue** to save the definition and queue a build.



1. Click the new build to follow its progress.



1. When the build completes, you should now see everything has succeeded as expected, including the tests.

