# DevOps with Visual Studio Team Services hands on lab

## Overview

In this lab, you will create a Visual Studio Team Services online account, check in your code, create a Continuous Integration pipeline, and test your cloud-based application.

## Objectives

In this hands-on lab, you will learn how to:

* Create a Visual Studio Team Services online account
* Create a VSTS Git repository
* Add your code to the VSTS Git repository
* Create a Continuous Integration pipeline

## Prerequisites

The source for the starter app is located in the HOL\dotnet\modern-cloud-apps\src folder.

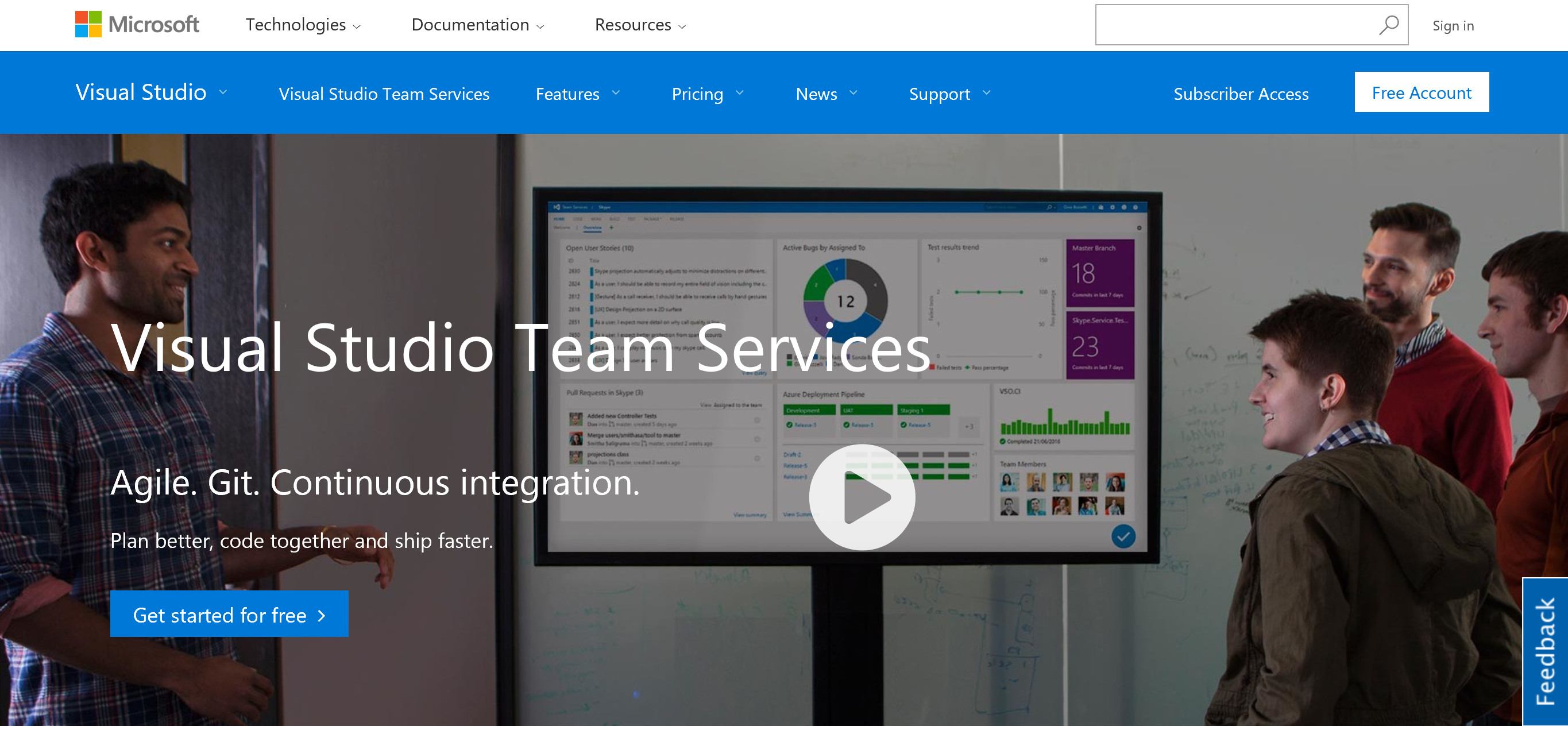
# Exercises

This hands-on-lab has the following exercises:

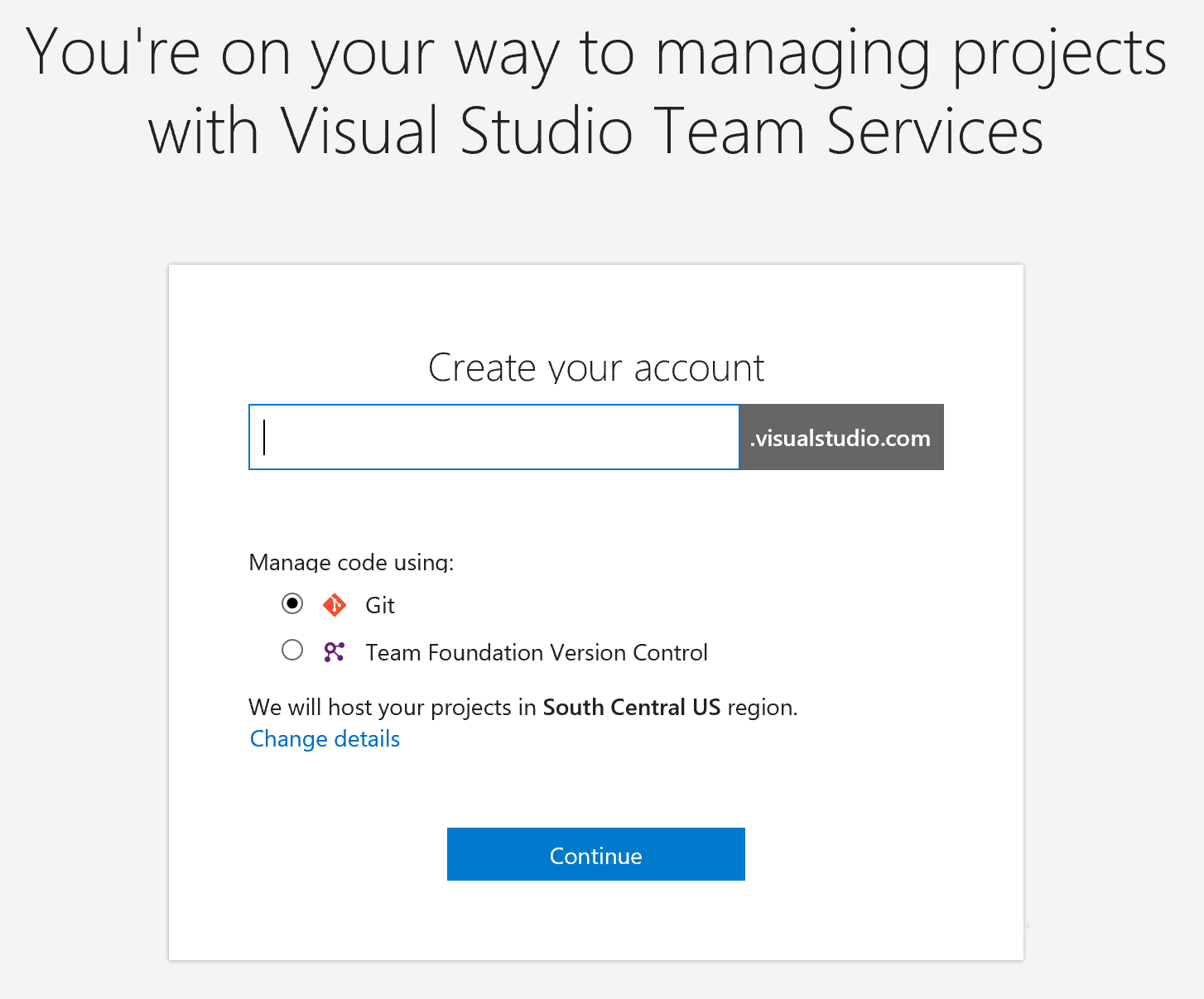
* Exercise 1: Create VSTS online account
* Exercise 2: Create VSTS Git repository
* Exercise 3: Add application to VSTS Git
* Exercise 4: Create a Continuous Integration pipeline

## Exercise 1: Create VSTS online account

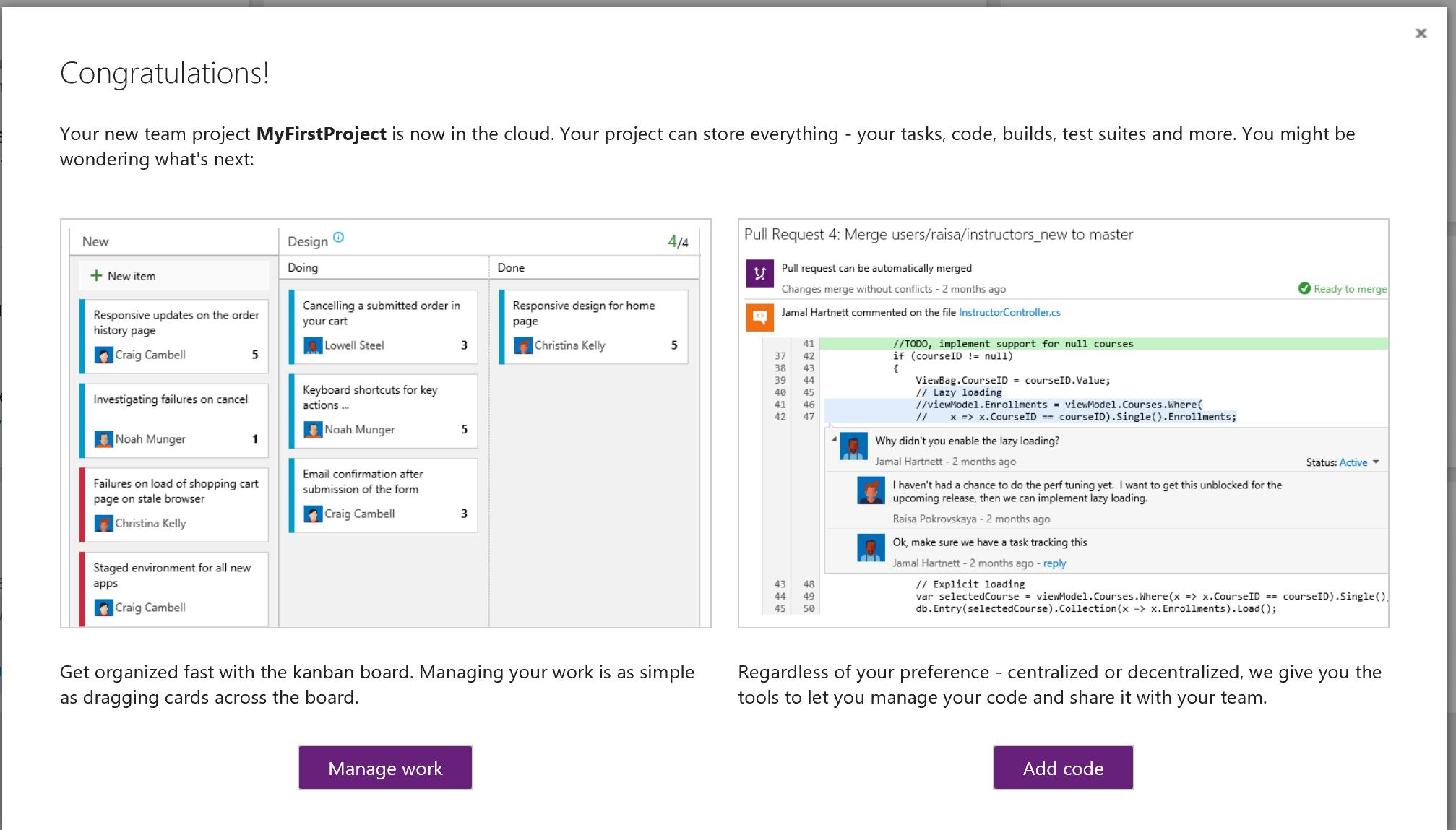
In your browser, go to <https://www.visualstudio.com/team-services/>. You should see this:



Log on using your Microsoft account

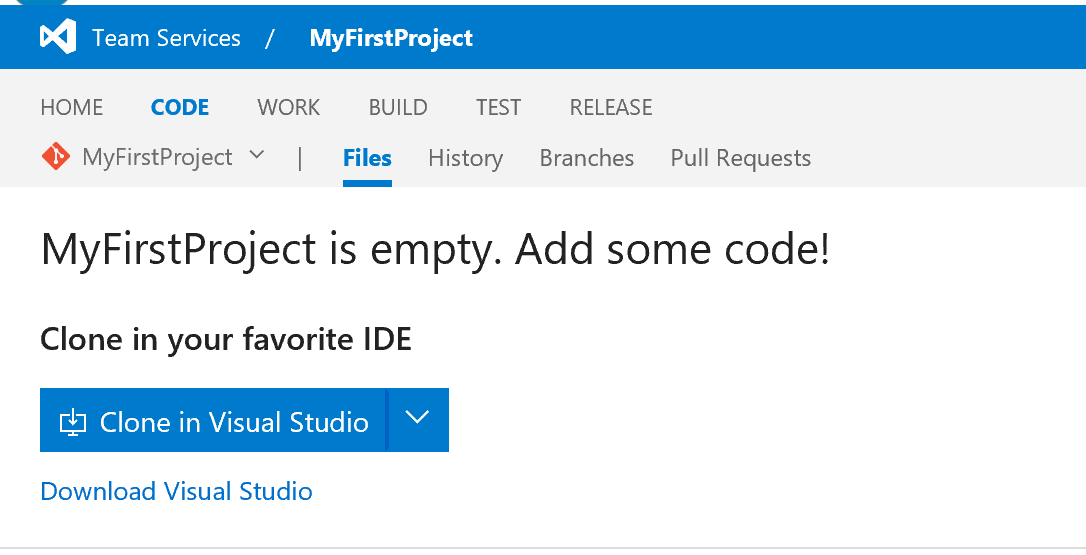


Choose an appropriate account name, and choose “Manage code using” git.

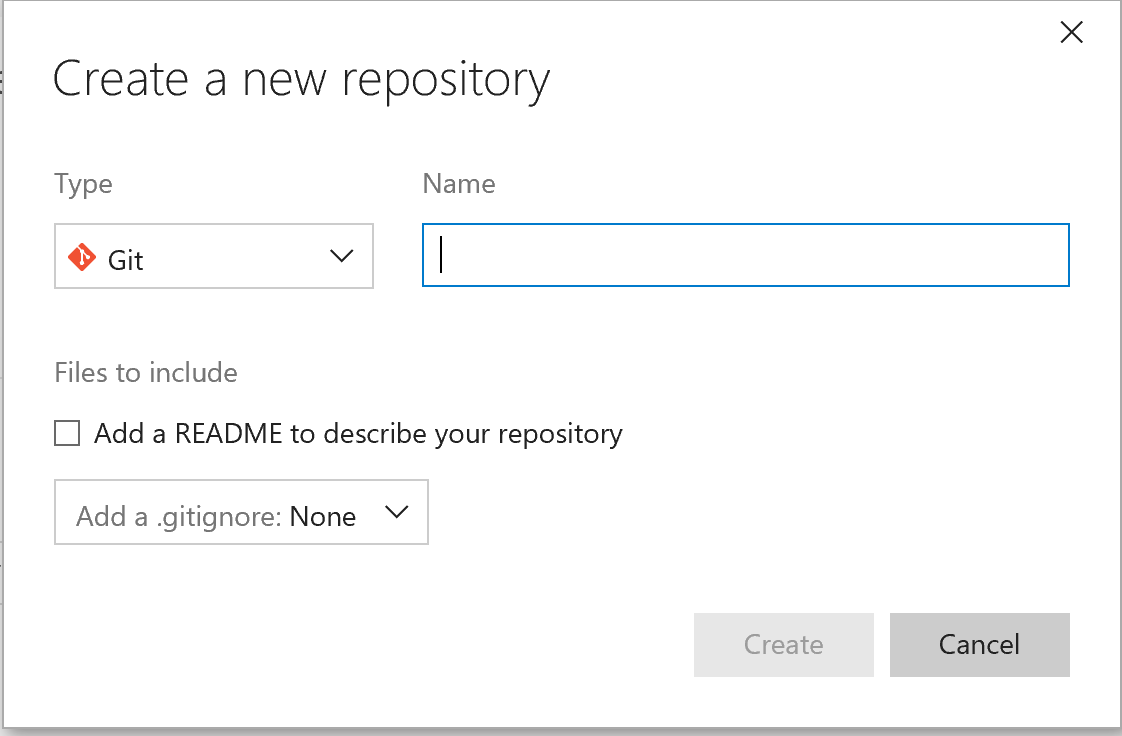


Choose “Add code”

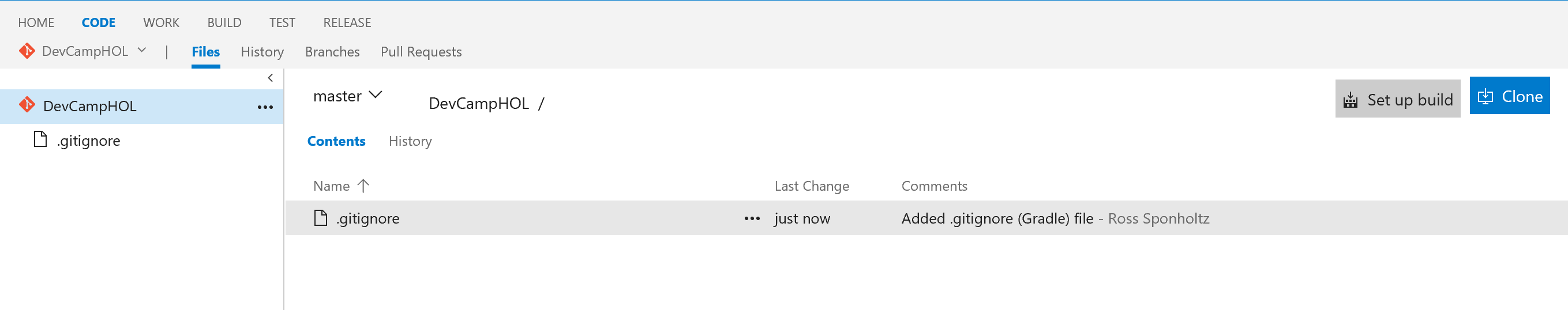
## Exercise 2: Create VSTS Git repository



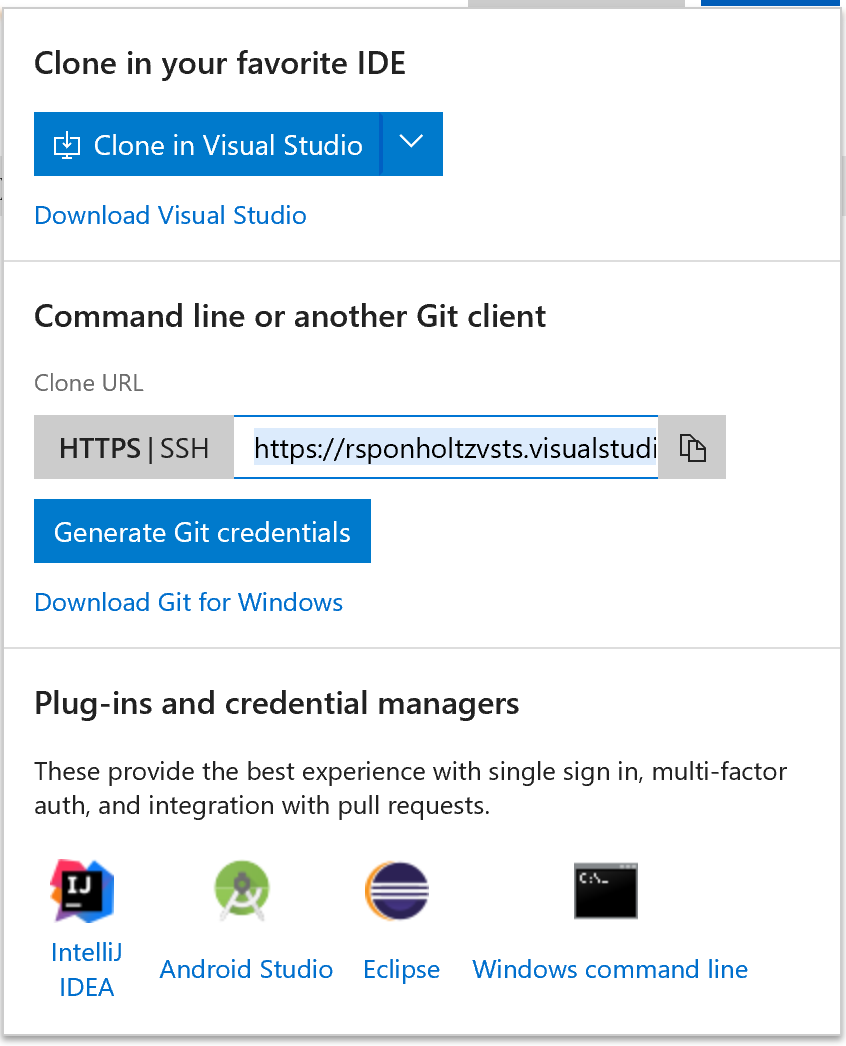
Click the V next to “my first project”, and choose “new repository”.



Use the name “DevCampHOL”, under “add a .gitignore choose “gradle”, and choose “Create”. This will take you to the main code repository page:



Choose “Clone”,



Click “generate Git Credentials, enter your password and click “Save Git Credentials”

## Exercise 3: Add application to VSTS Git

The next set of steps will be described using the Git command line, but the same process can be done via your IDE such as Visual Studio or Eclipse.

Open the GIT bash command line, and navigate to a working directory.

git clone <https://contosovsts.visualstudio.com/MyFirstProject/_git/DevCampHOL>

this will copy an empty project to your local machine. Next, go to your source directory for the project (add directory here) and copy all artifacts in the project, EXCEPT FOR THE .git DIRECTORY AND IT’S CONTENTS. Paste them into the working directory.

In the GIT bash window, CD to DevCampHOL:

Cd DevCampHOL

Git add .

Git commit -a

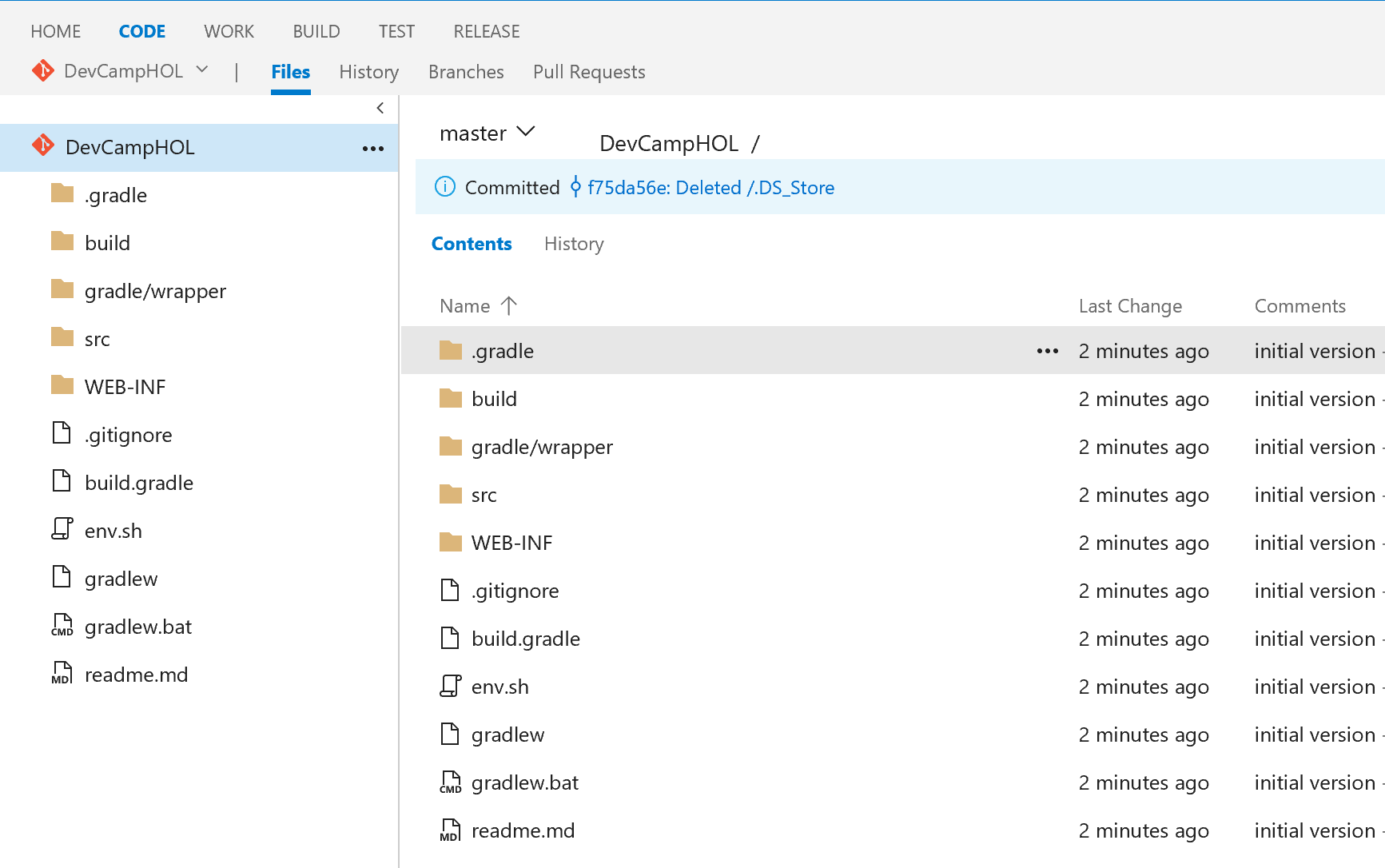
This will start a vi editor window for you to put in your comment. To insert, type I, then type your comment. Press <esc>wq to write and exit.

Then, type

Git push

This will synchronize the VSTS repository with your local one.

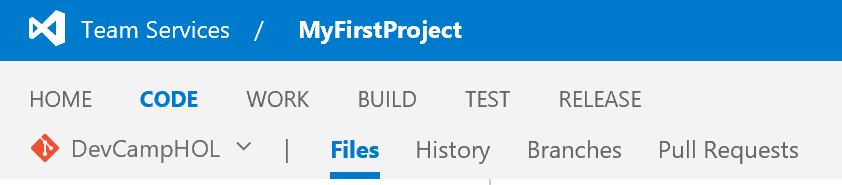
Go to your VSTS browser window and refresh – you should see the files in your project, stored in the VSTS git repository:



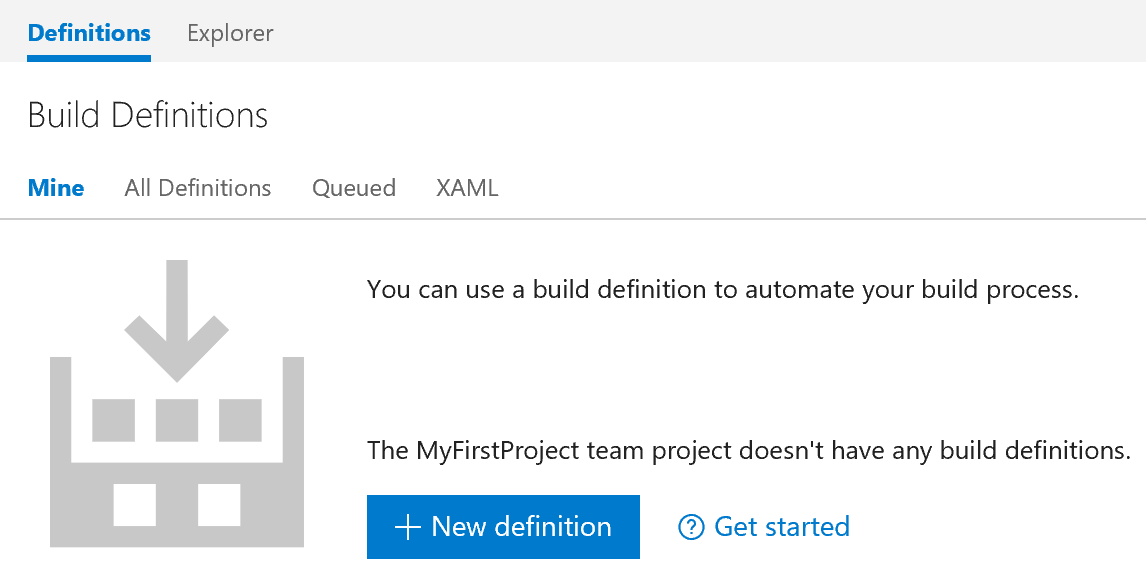
## Exercise 4: Create Continuous Integration pipeline

In this exercise, you will create an automated build definition that will pull your code from the repository that was created in Exercise 3, and then build it automatically. You will then create a release definition that will take the artifacts from the build and deploy them directly to your Azure Web App.

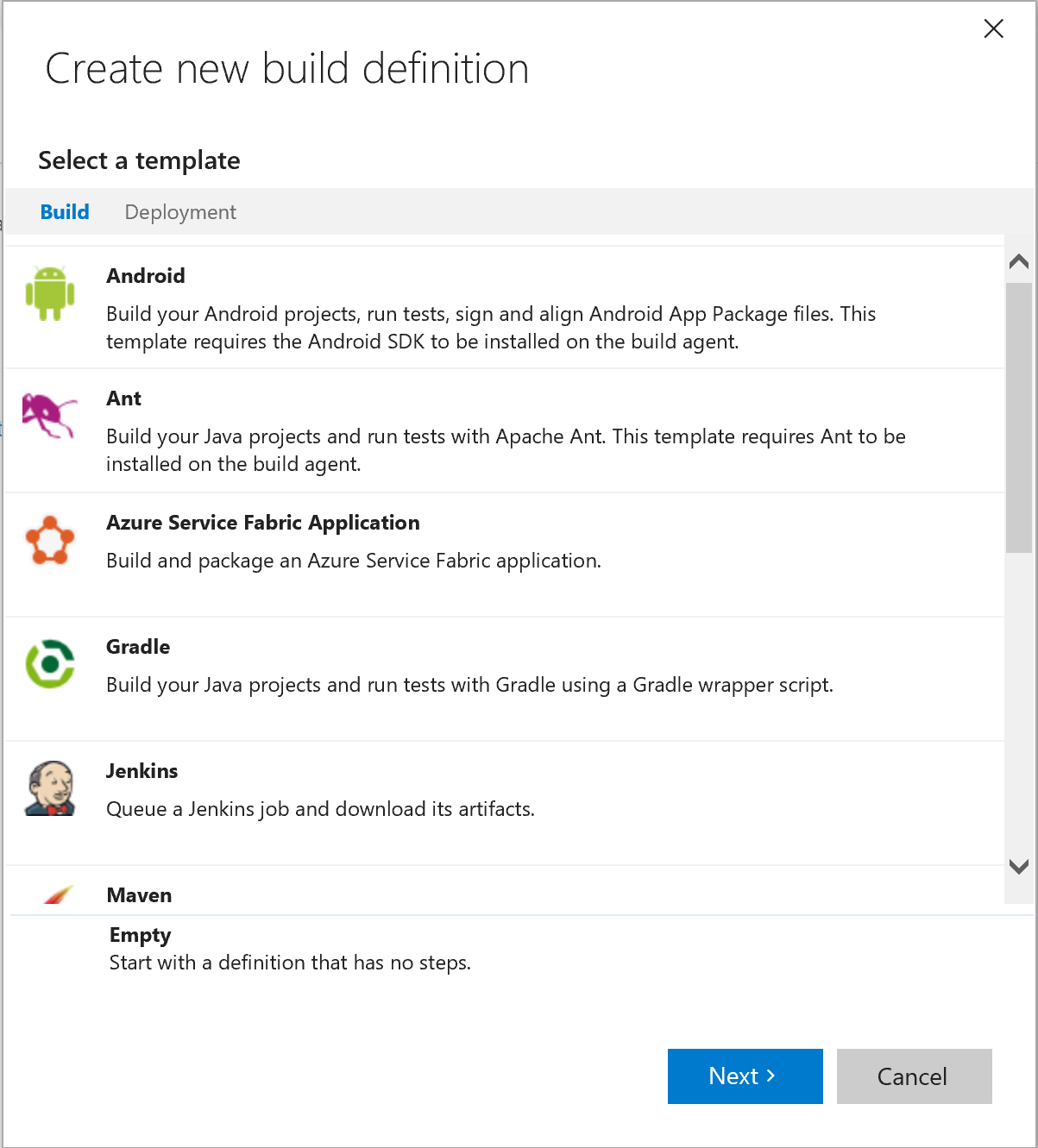
Click “BUILD” at the top of the VSTS window:



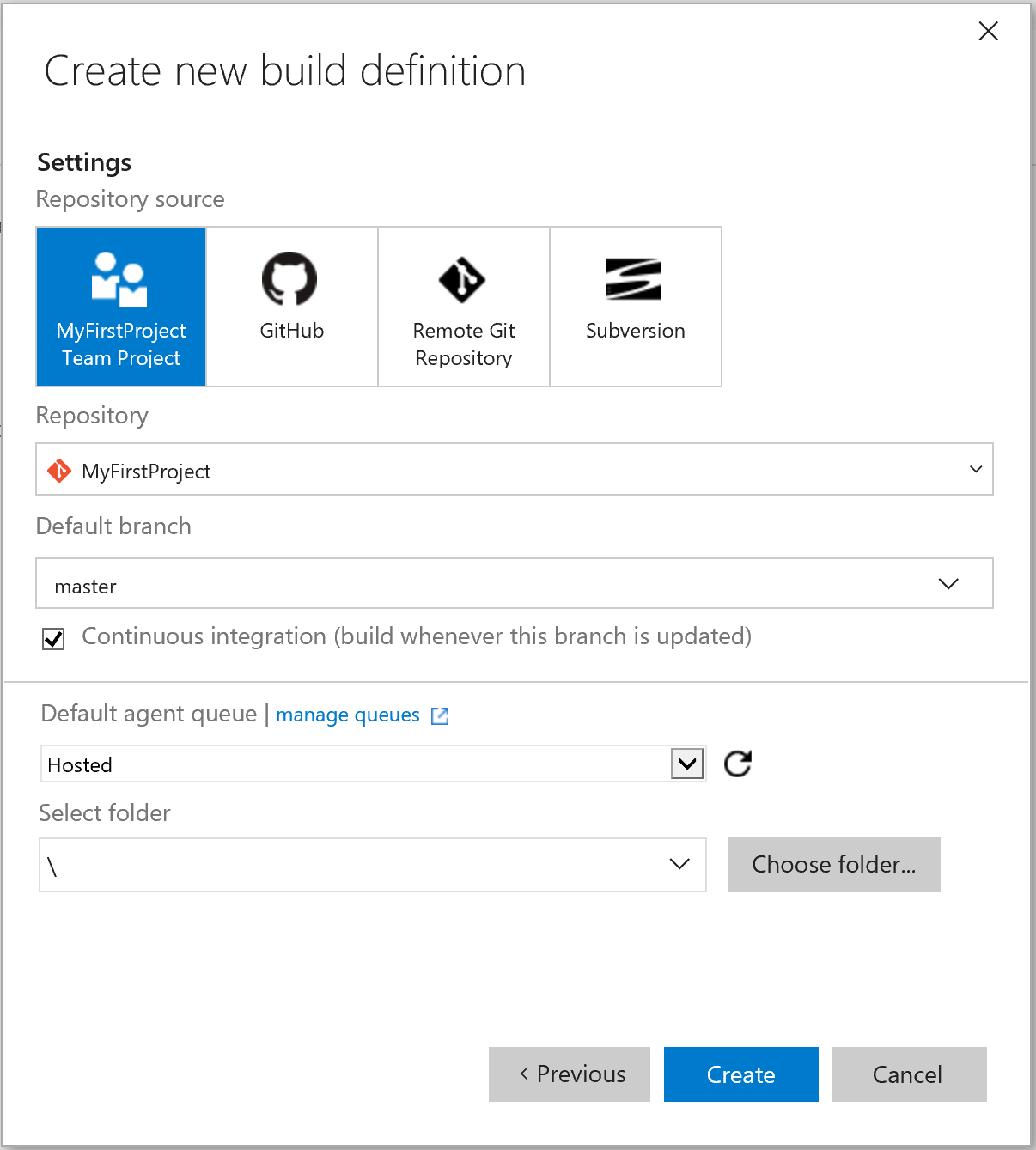
On the build page, click +New definition



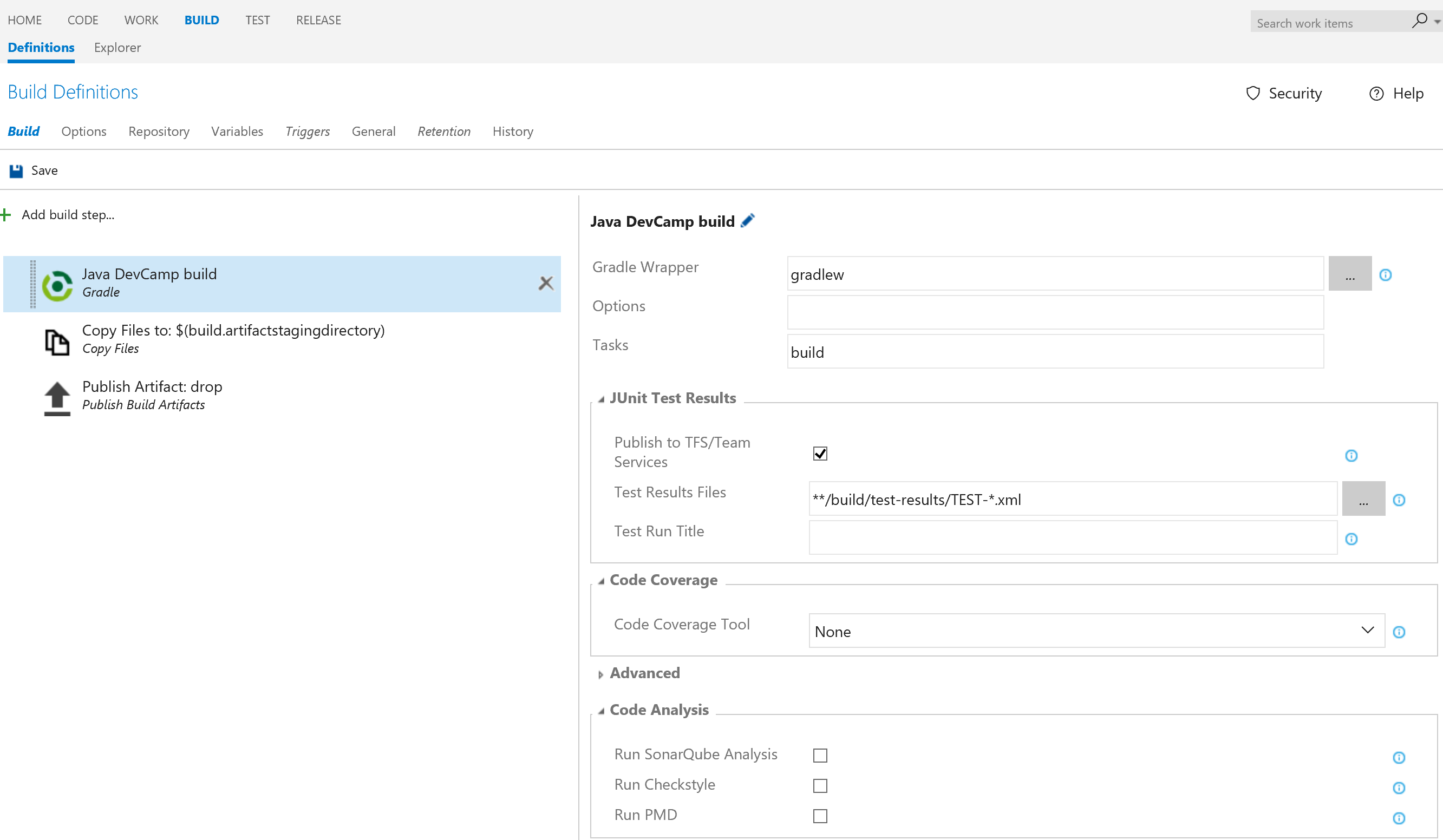
In the window that pops up, choose “gradle” and click next:



Next, choose the following options, and click “Create”:



You should see this page next:



Click the pencil next to “gradle build” and rename the build definition to “Java DevCamp build”

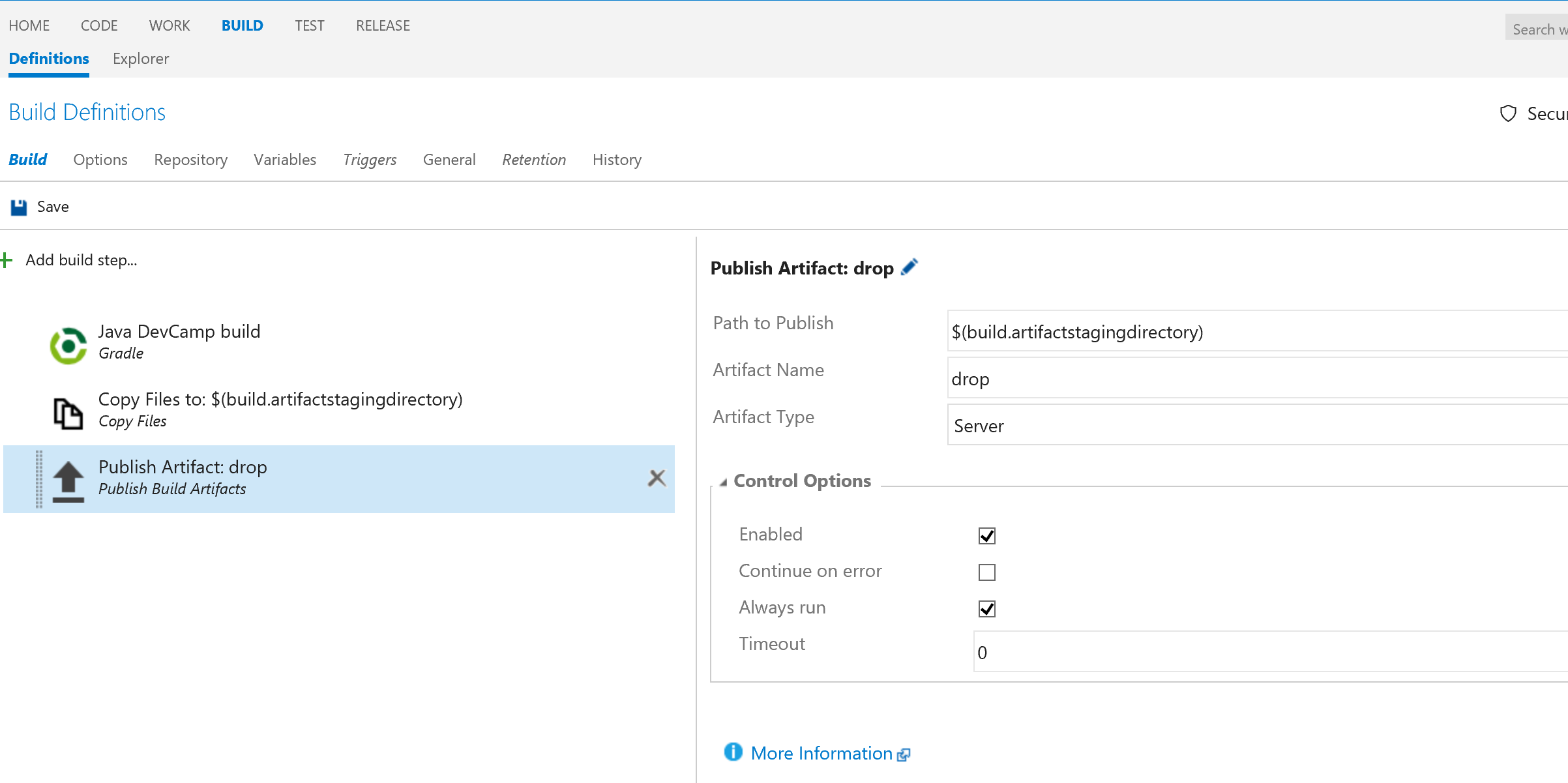
Change “Tasks” to “war”

Click off the checkbox for “Publish to TFS/Team Services” in the Junit Test Results box.

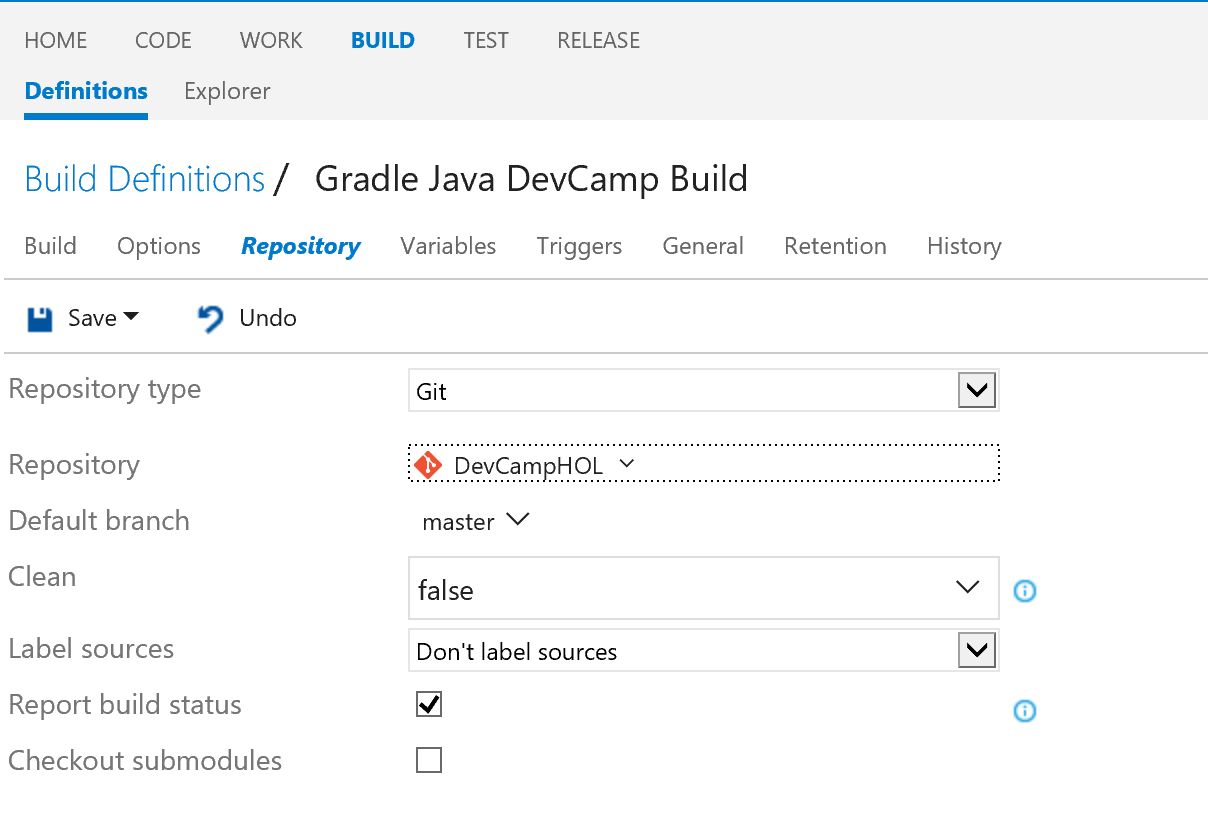
Click the “Copy Files to : $(build.artifactstagingdirectory) build step in the left pane. You should see this:



Change the contents to “\*\*/\*.war”

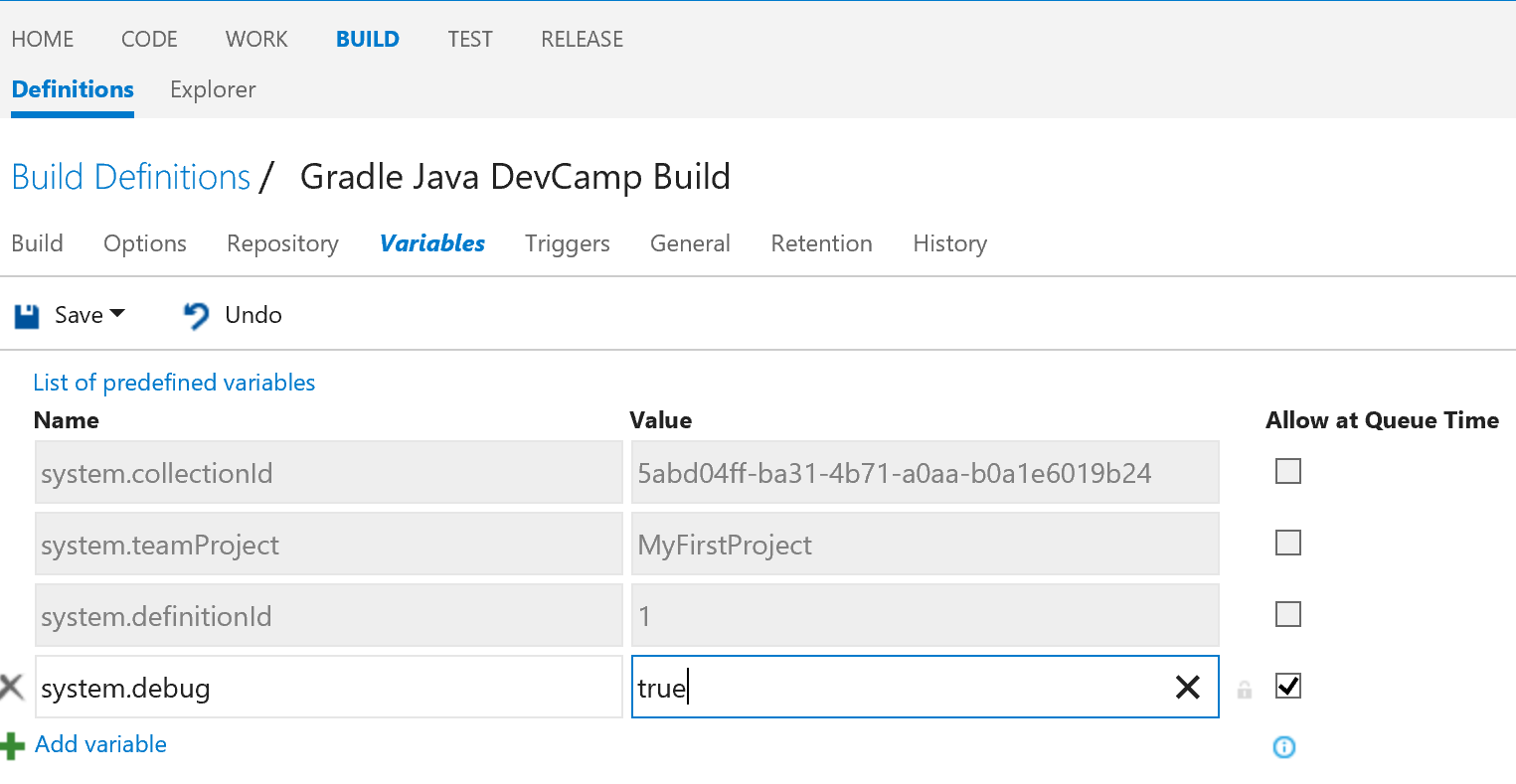


Click “Repository”:

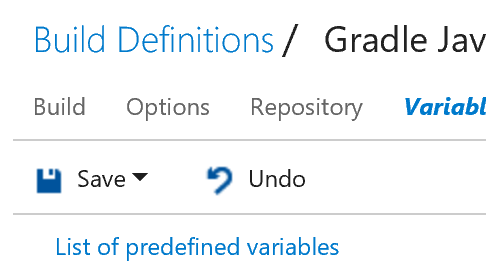


Change the Repository to “DevCampHOL”.

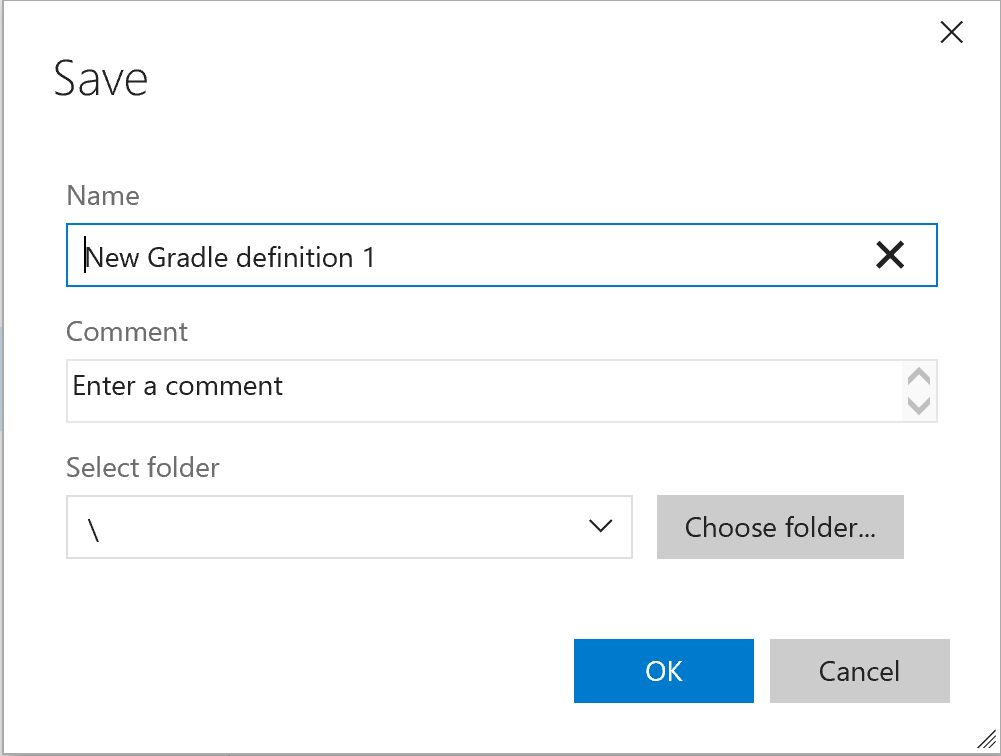
Click “Variables” and change the value for system.debug to true:



Finally click save.

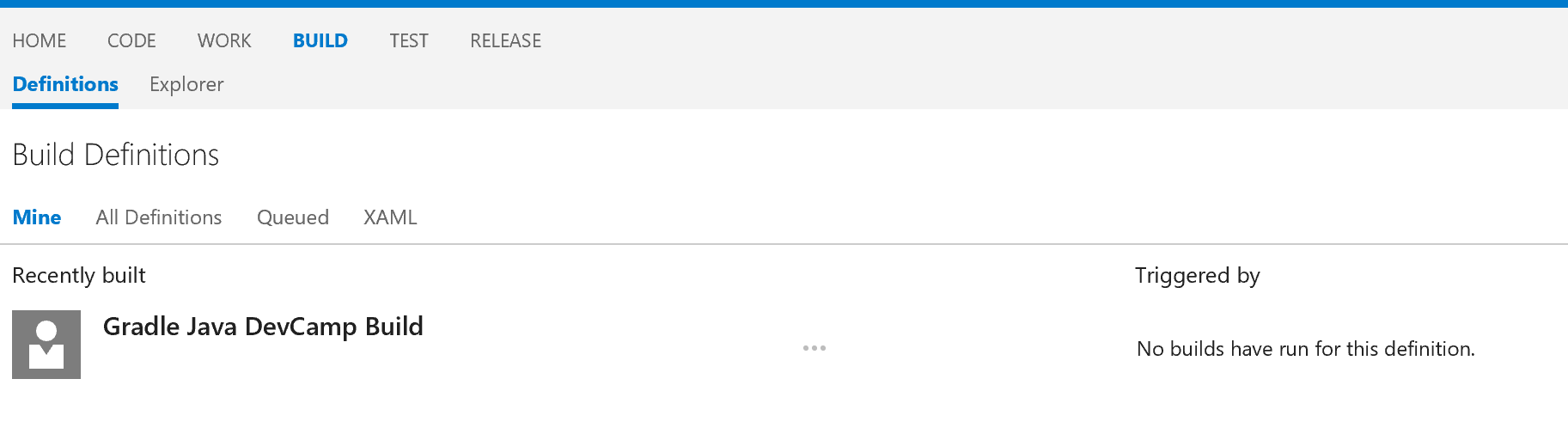


Which will pop up a dialog box:

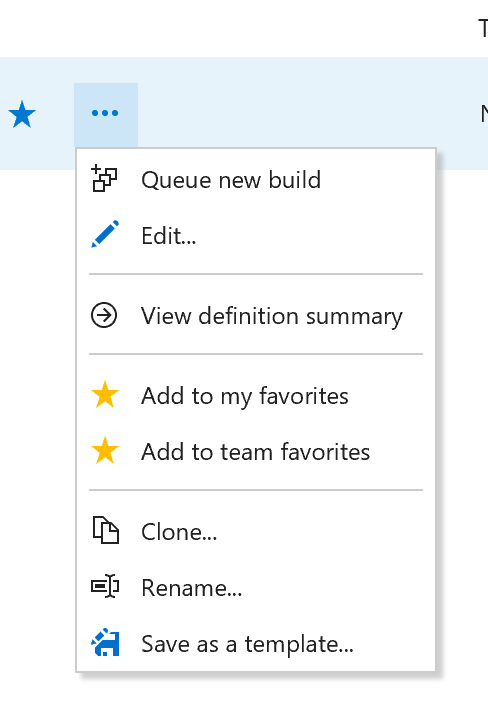


Change the Name to “Gradle Java DevCamp Build” and click ok.

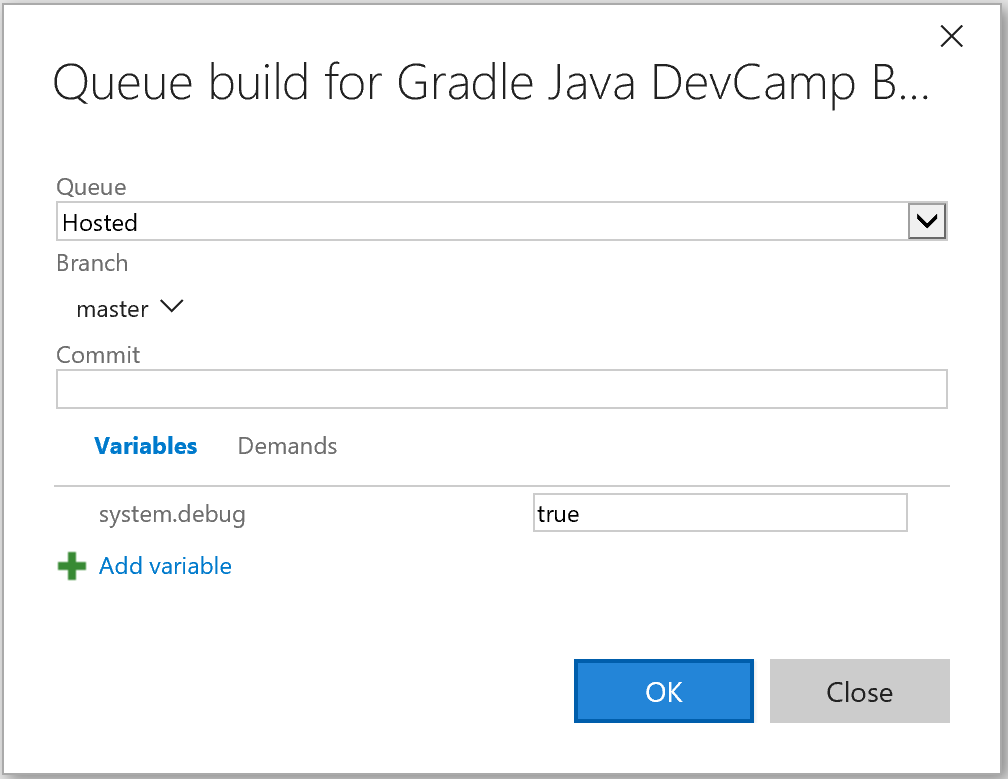
Click the build tab, and you will see:



Click the … by Gradle Java Devcamp Build:

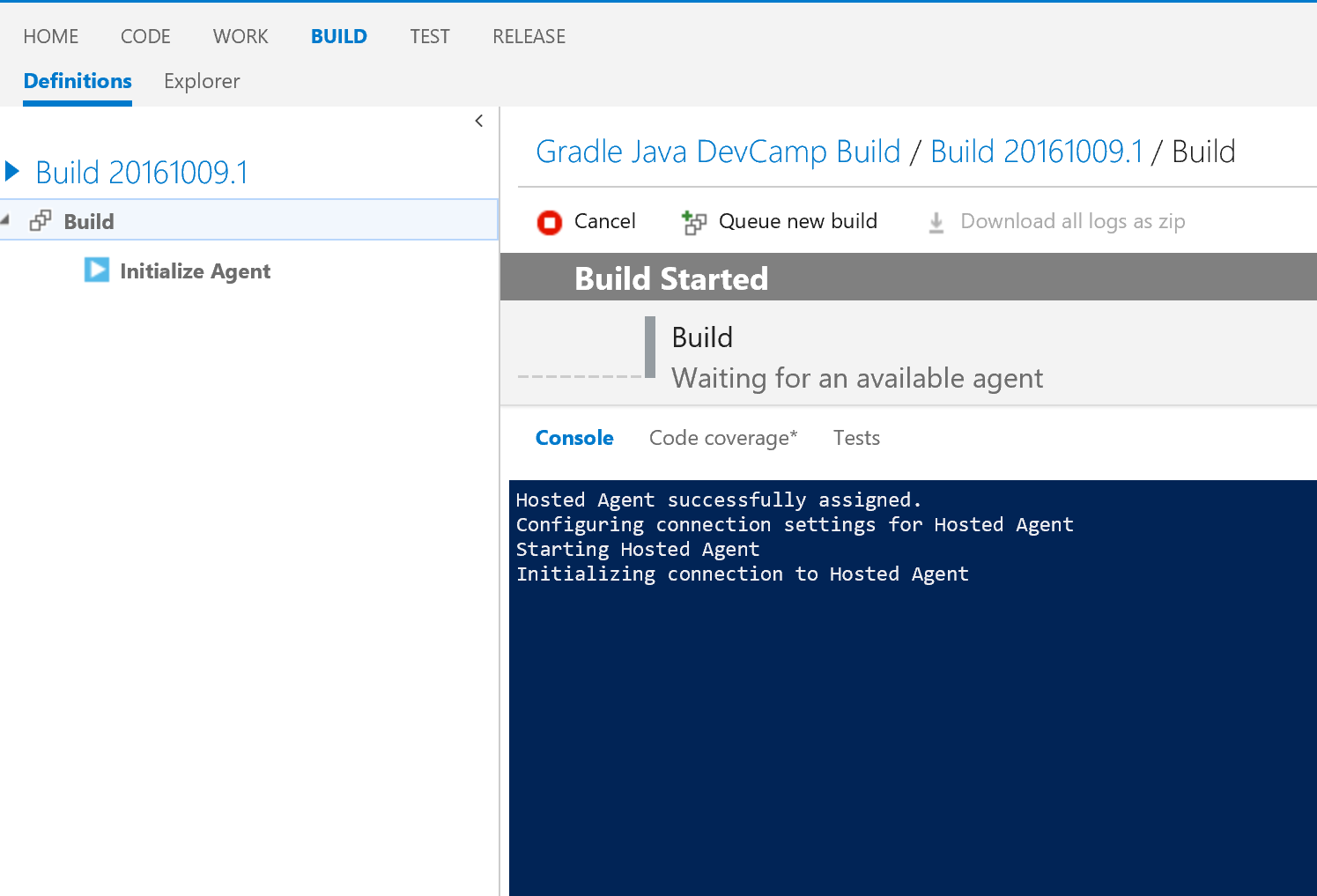


Choose “Queue new build”

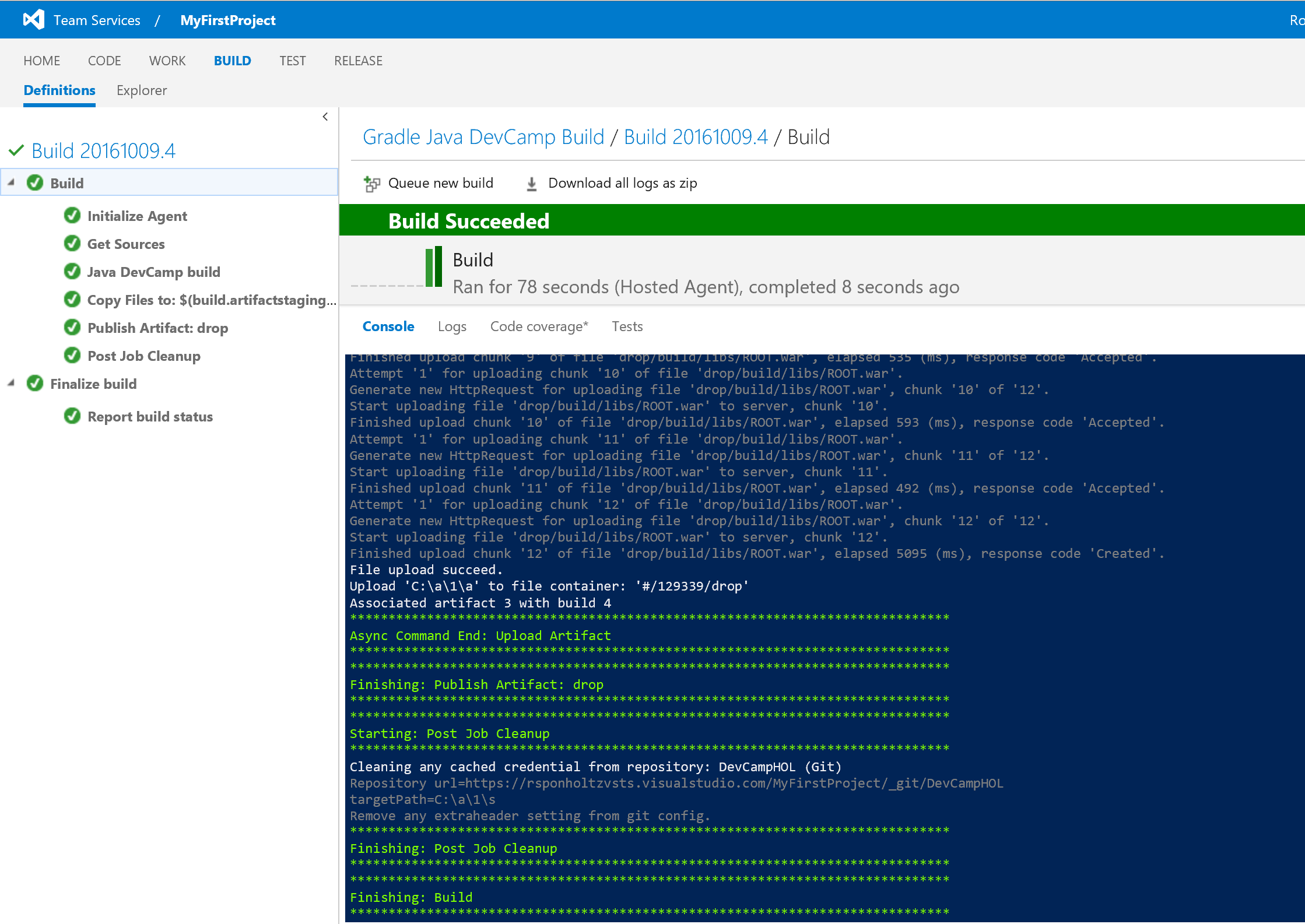


Leave all the defaults in place, and click OK.

You should see the build start, it will look something like this:



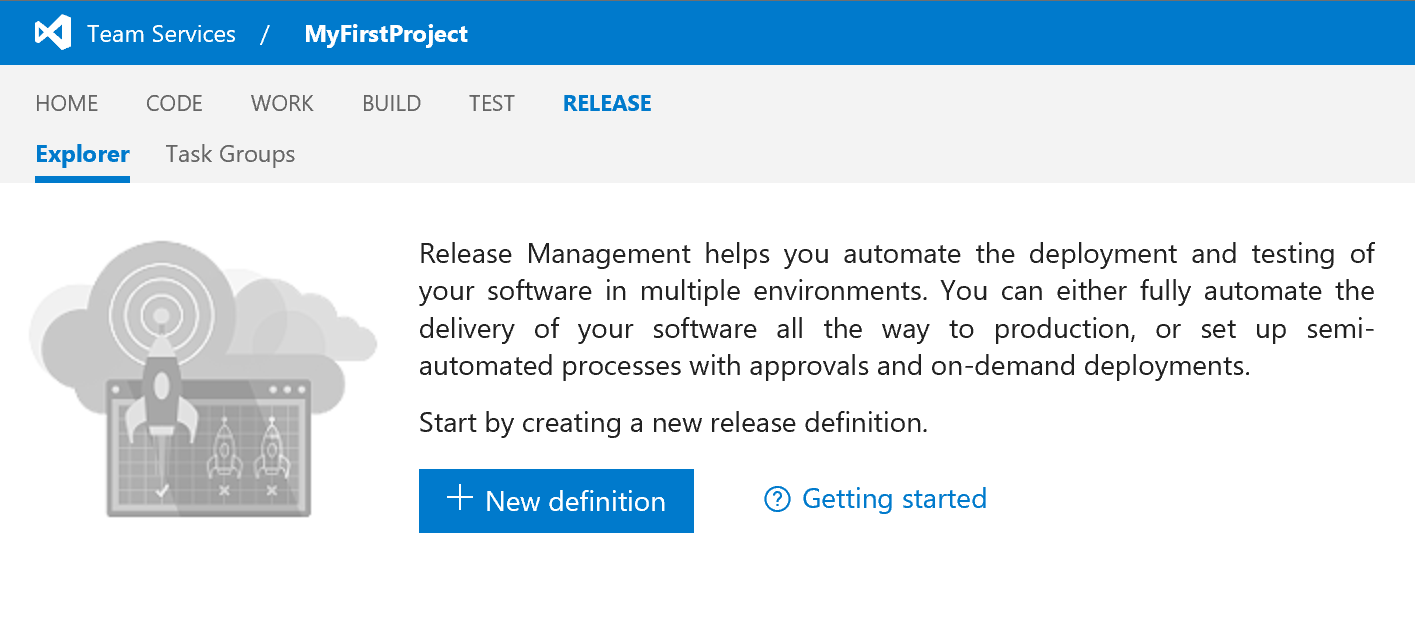
It will run for a while, and display this:



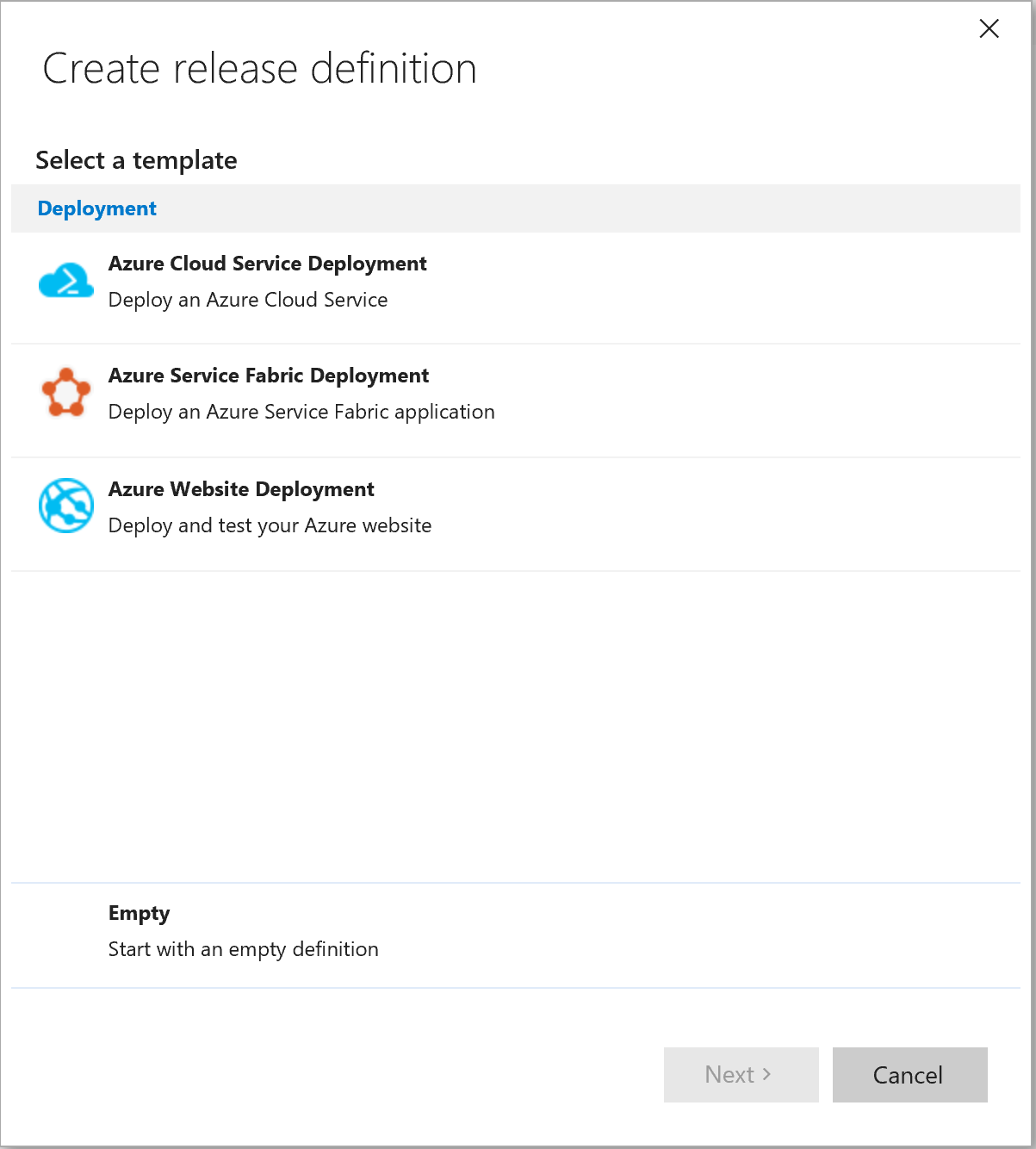
Set up release to Azure Web App

This step will create a release definition that will package and deploy the build package (.war file) to the running Azure Web App. We are going to use Microsoft WebDeploy to place the .war file on the Web App by placing the .war file in a webapps directory and creating a zip file from it. WebDeploy will unzip the zip file on the server, thus placing the .war file in the webapps directory automatically.

In the browser window showing Visual Studio Team Services and the “MyFirstProject” project, click on the release tab, and then click +New Definition

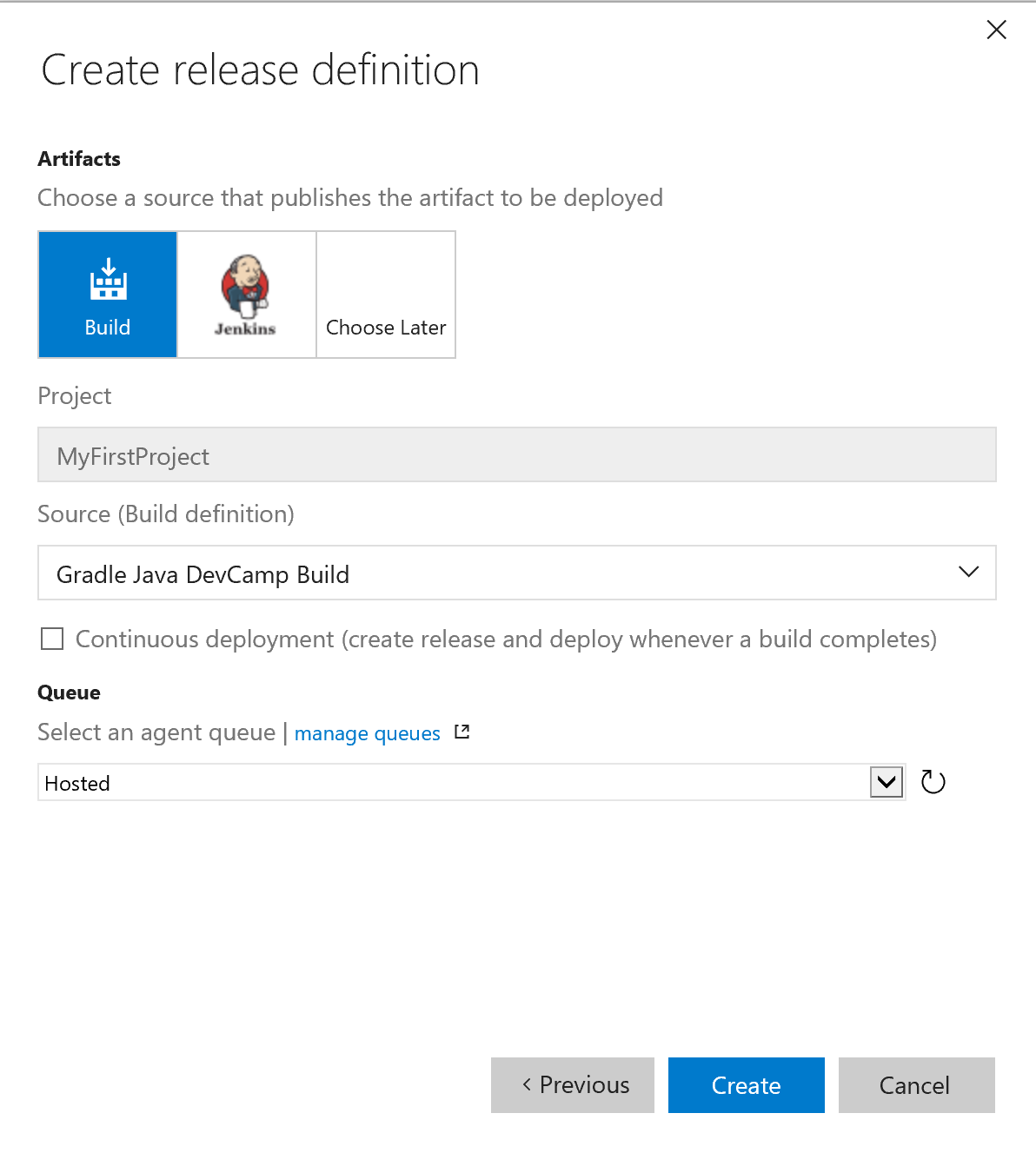


This will pop up a dialog:

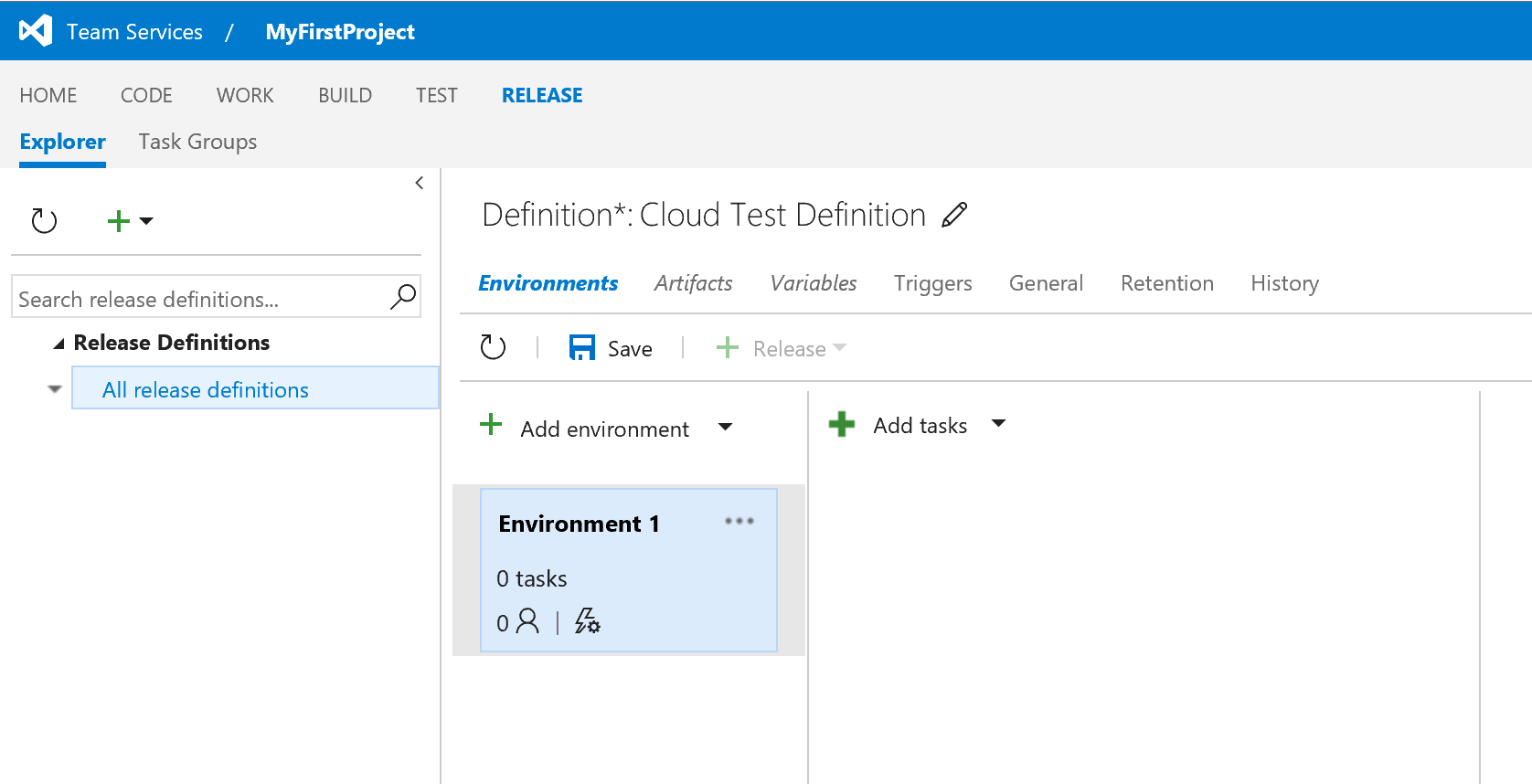


Choose “empty” at the bottom to start with an empty release definition.

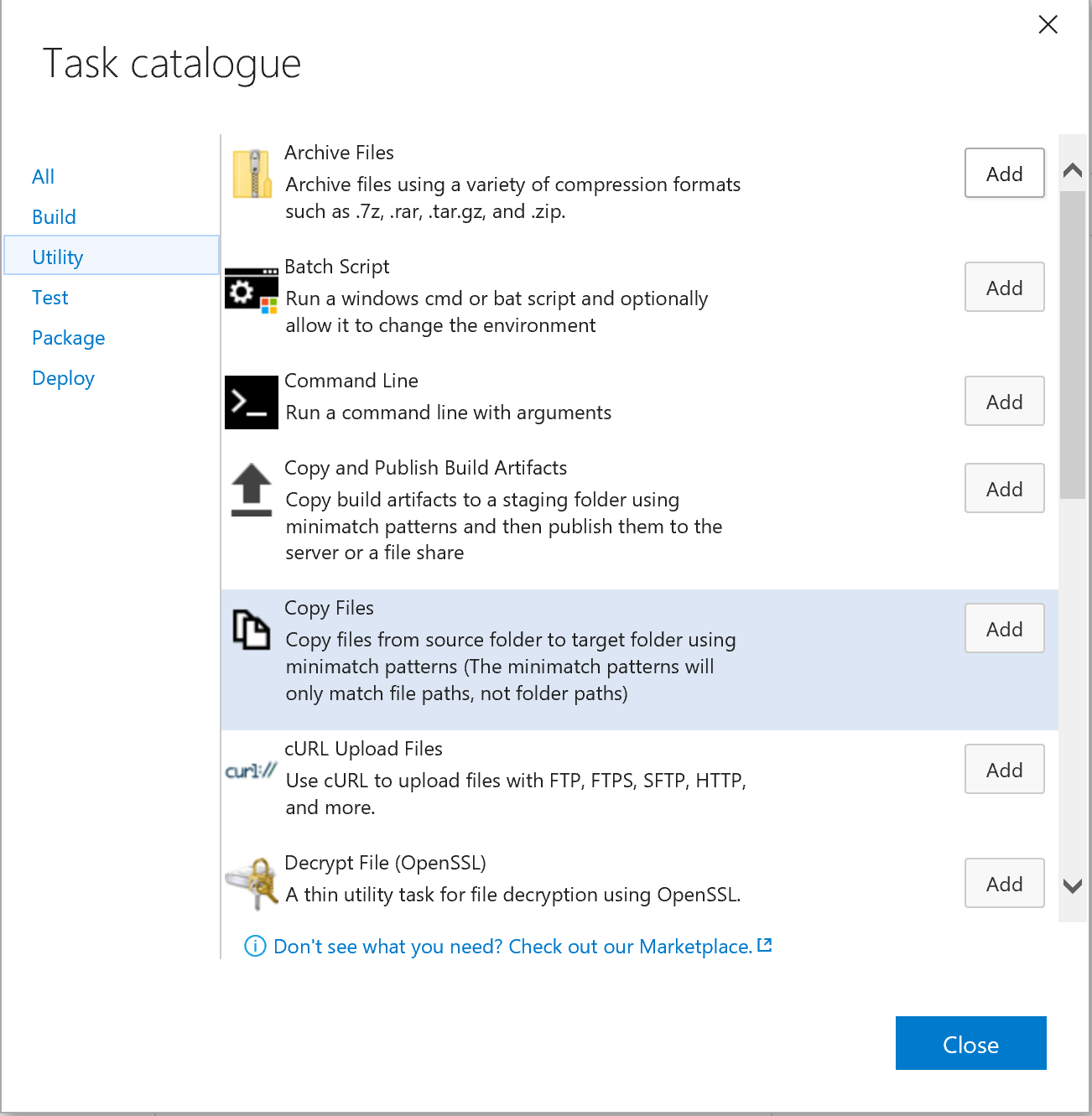
In the next dialog box, choose “Build” as the artifact source, choose “continuous deployment”, and make sure the source is “Gradle Java DevCamp Build” and the queue setting is “Hosted”. Finally click “Create”.



In the following page, use the pencil at the top to change the definition name to “Cloud Test Release”:



Then click “+ Add tasks”



In the task catalog, click “utility” and press the add button by “Copy Files”, then “Archive Files”, and then click on “deploy” and press the add button by “Azure Web App Deployment” and choose “close”. Your release definition window should look like this:



Click on “Copy Files to:” in the middle and make these changes:

Source Folder: $(System.DefaultWorkingDirectory)/Gradle Java DevCamp Build/drop/build/libs/

Contents: ROOT.war

Target Folder: $(System.DefaultWorkingDirectory)/Gradle Java DevCamp Build /drop/build/libs/webapps

Click on “Archive Files” in the middle, and make these changes:

Root Folder to archive: $(System.DefaultWorkingDirectory)/Gradle Java DevCamp Build /drop/build/libs/

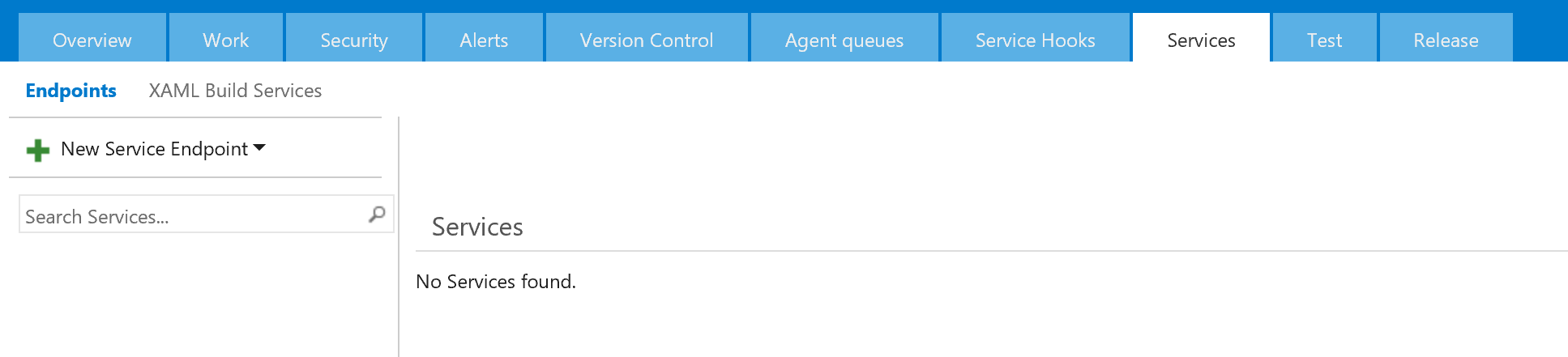
Uncheck the “Prefix root folder name to archive paths”

Archive file to create: $(System.DefaultWorkingDirectory)/Gradle Java DevCamp Build/drop/build/libs/deploy.zip

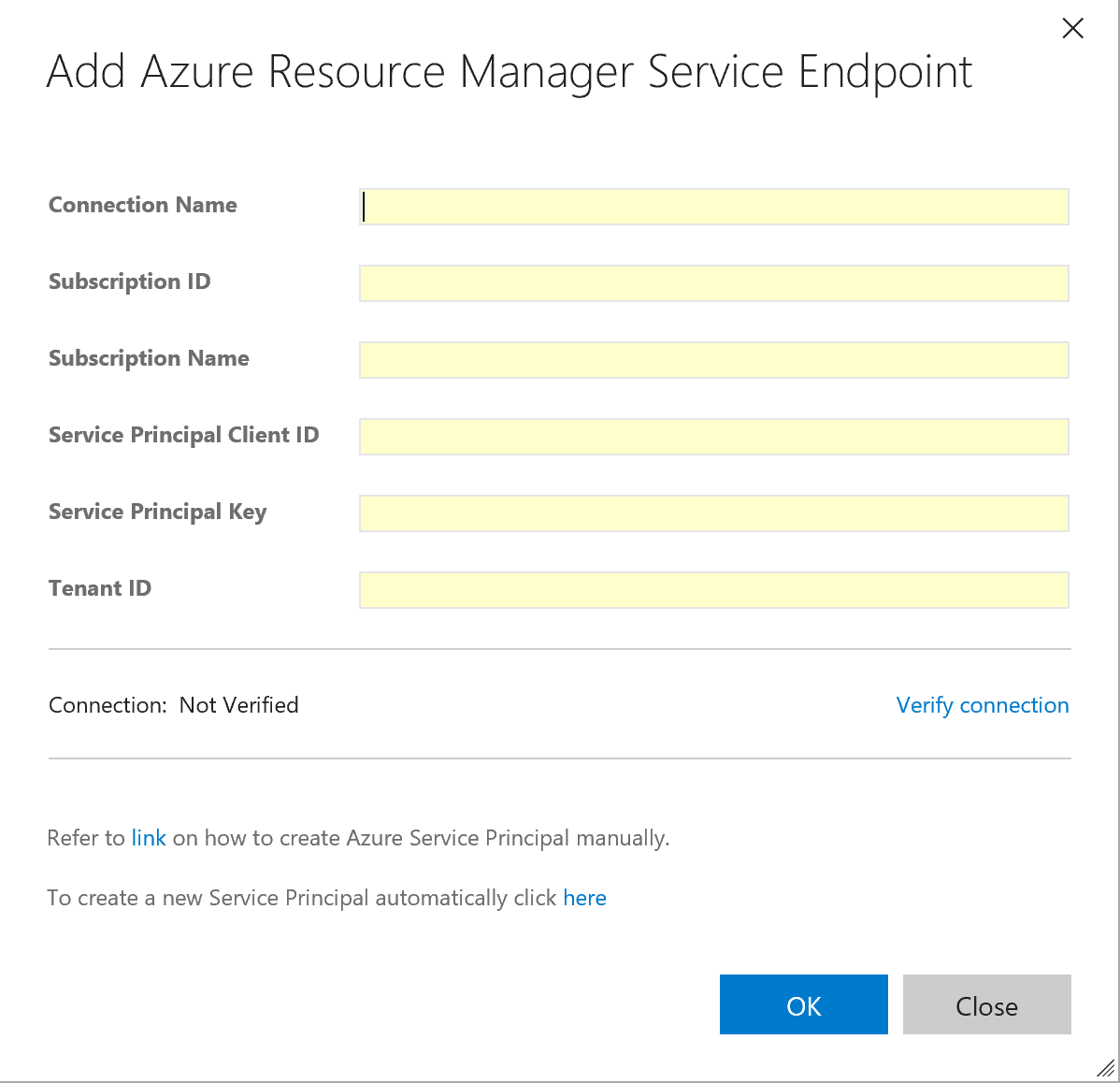
Click on “Deploy AzureRM App Service” in the middle.



Click Manage on the right side of the “AzureRM Subscription” box. A new window will pop up that looks like this:



Click +New Service Endpoint and choose Azure Resource Manager. This dialog will pop up:



In Connection Name, type “DevCamp Azure Connection”. If your VSTS subscription shares a common directory with your Azure subscription, you can simply choose your subscription to connect to using the dropdown listbox. If you are deploying to a subscription that does not share a common AD, you will have to create a service principal using this process:

1. Download and run [**this PowerShell script**](https://raw.githubusercontent.com/Microsoft/vso-agent-tasks/master/Tasks/DeployAzureResourceGroup/SPNCreation.ps1) in an Azure PowerShell window.
2. Enter a user-friendly name to use when referring to this service endpoint connection.
3. Copy these fields from the output of the PowerShell script into the Azure subscription dialog textboxes:
   * Subscription ID
   * Subscription Name
   * Service Principal ID
   * Service Principal Key
   * Tenant ID

Once the service manager endpoint has been created,

# Summary

In this hands-on lab, you learned how to:

* Create a Visual Studio Team Services online account
* Create a VSTS Git repository
* Add your code to the VSTS Git repository
* Create a Continuous Integration pipeline

Copyright 2016 Microsoft Corporation. All rights reserved. Except where otherwise noted, these materials are licensed under the terms of the MIT License. You may use them according to the license as is most appropriate for your project. The terms of this license can be found at https://opensource.org/licenses/MIT.