Azure DevOps Pipelines CI CD Tutorial

Continuous integration and deployment of a Xamarin.Forms App

Introduction

In this tutorial, I will be using a fully featured continuous integration (CI) and continuous delivery (CD) service called Azure Pipelines to automate the build, testing, and deployment of a Xamarin.Forms mobile app to the Visual Studio App Center.

What’s covered in this tutorial

* Setting up an Azure Repos Project
* Creating a Xamarin.Forms App
* Setting up a build pipeline for CI
* Setting up a release pipeline for CD

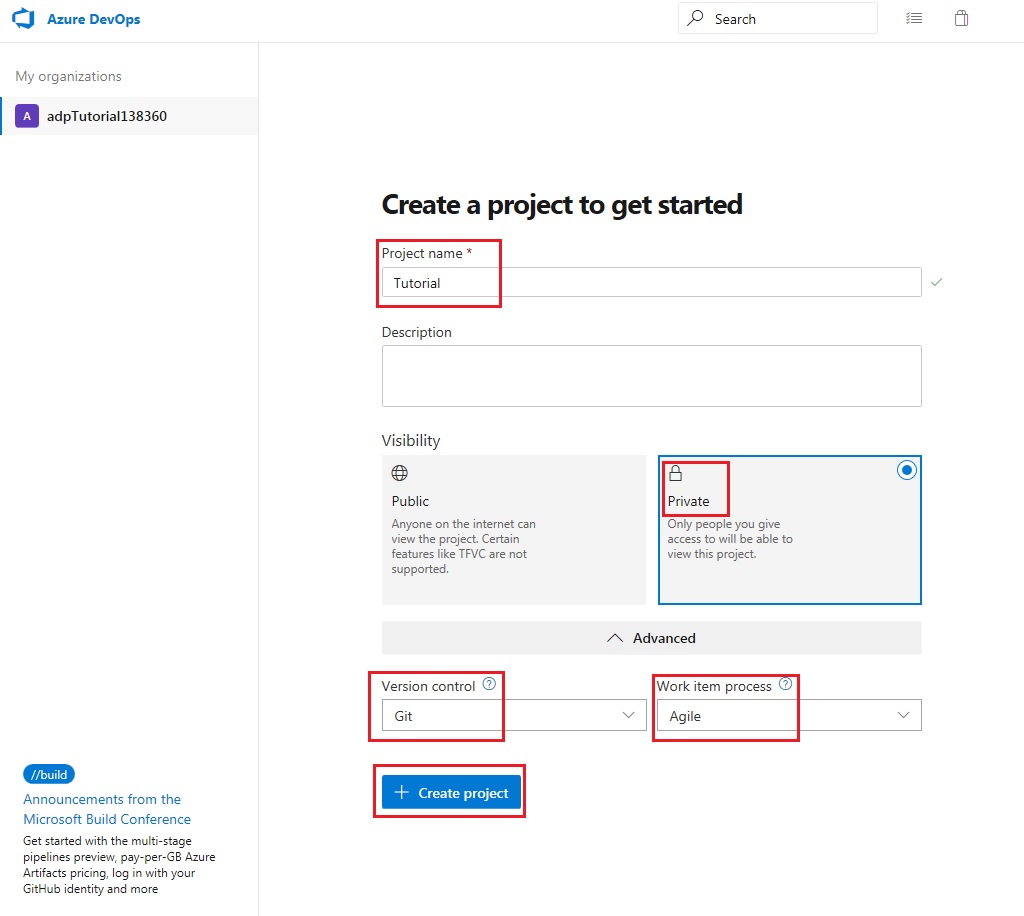
Prerequisites

* A free Azure DevOps account (https://dev.azure.com)
* Visual Studio 2017 IDE for Windows (<https://visualstudio.microsoft.com/vs/older-downloads/>)
  + Make sure Xamarin is selected during IDE installation
* A free App Center account (https://appcenter.ms/)

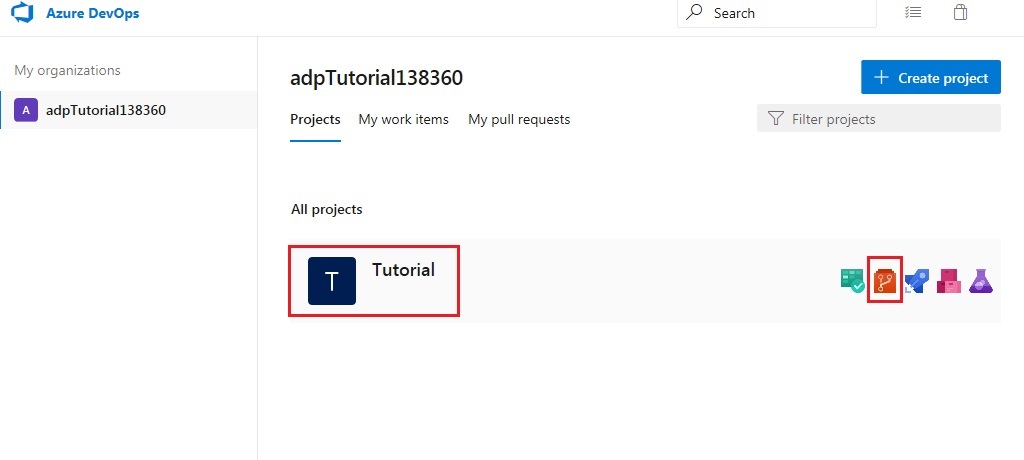
Creating a blank project in Azure repos

Navigate to your Azure account with the following url: [https://dev.azure.com/{username}](https://dev.azure.com/%7busername%7d) ; {username} should be replaced with an appropriate value associated with the account.

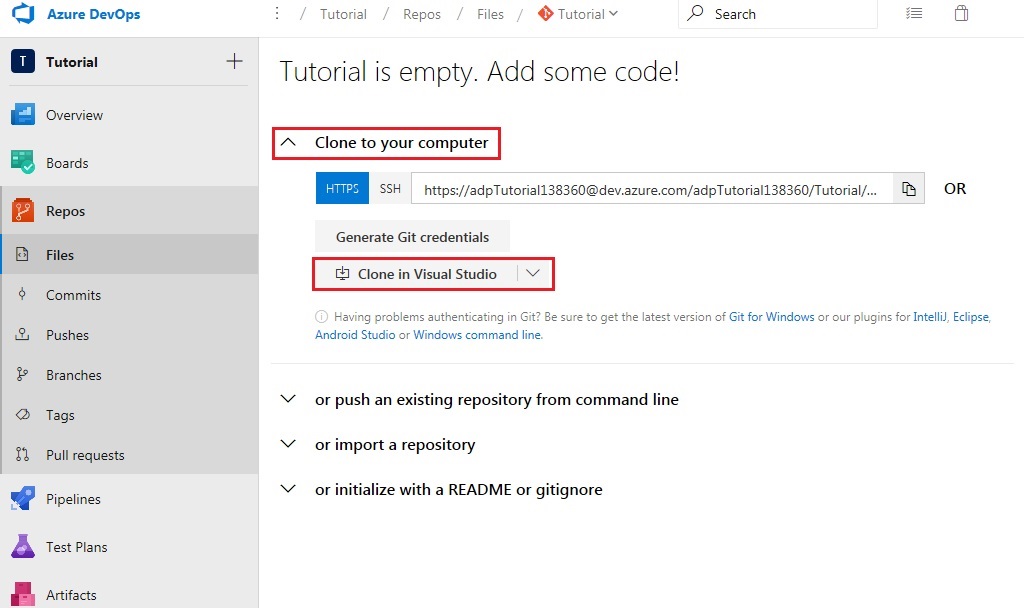
Create a new project as shown below



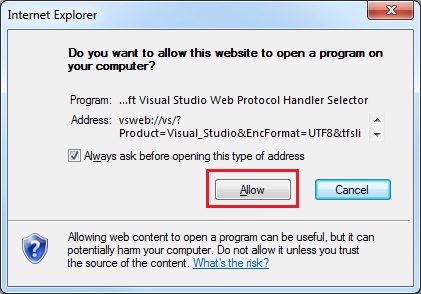
Click on the code icon of the newly created project



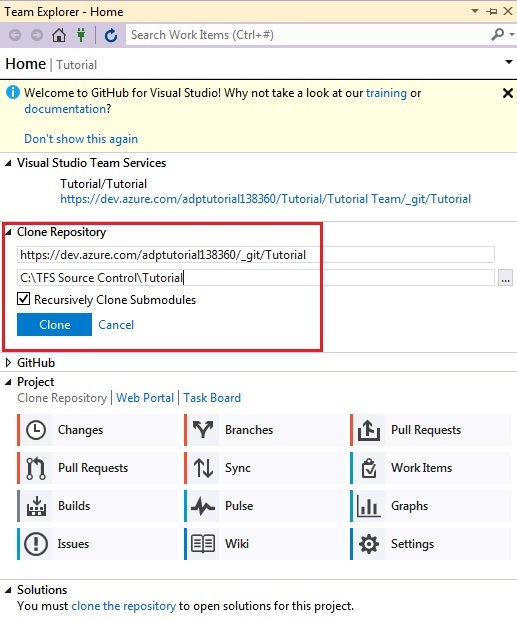
This takes you to the newly created project page with the following greeting message: “Tutorial is empty. Add some code!”. Before adding some code, the repository will need to be cloned in Visual Studio.



You will be redirected to a new tab with a popup message. Click ‘Allow’ when prompted by your web browser. This will cause Visual Studio to be launched.

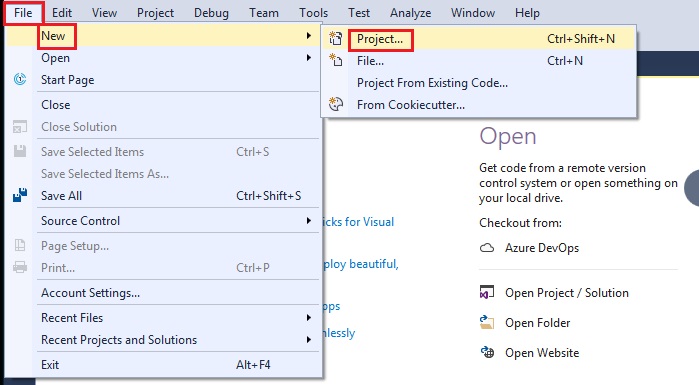


Select an appropriate location and clone the repository to your computer.

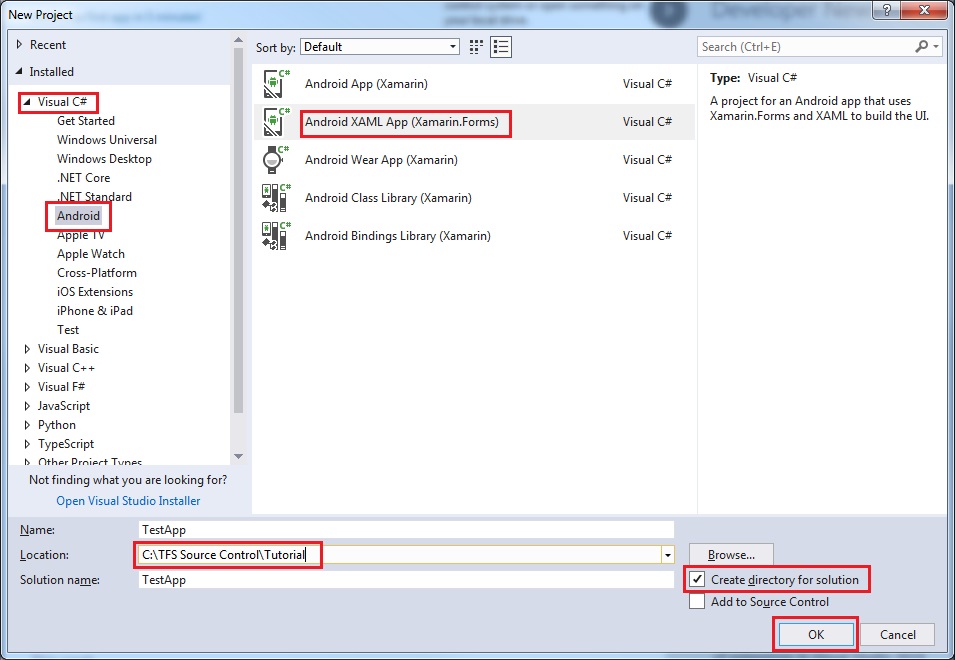


Creating a Xamarin.forms app in vs2017

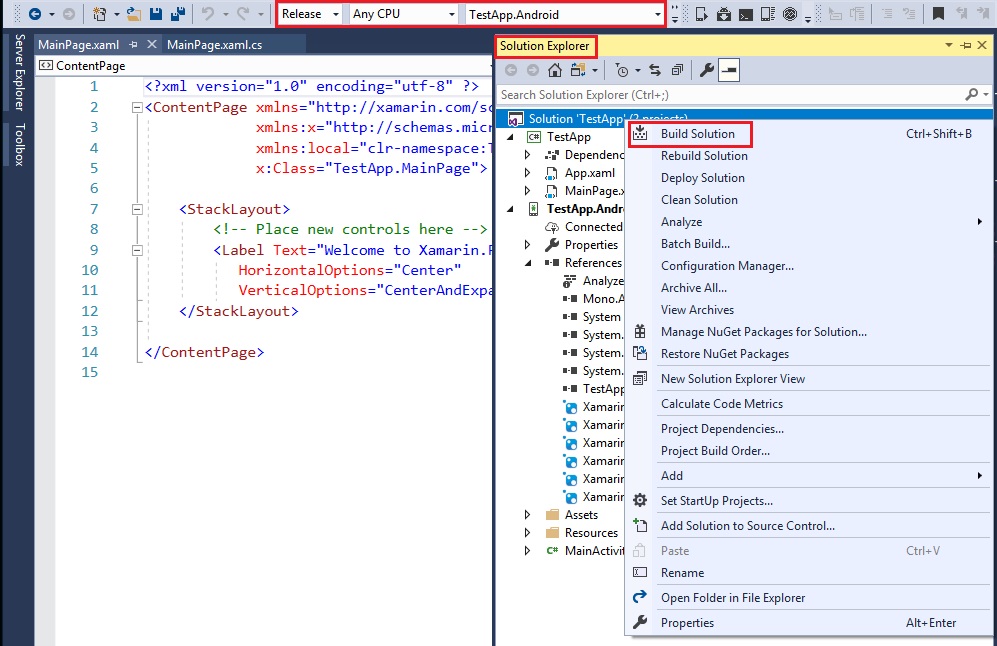
While in Visual Studio, create a new project as follows



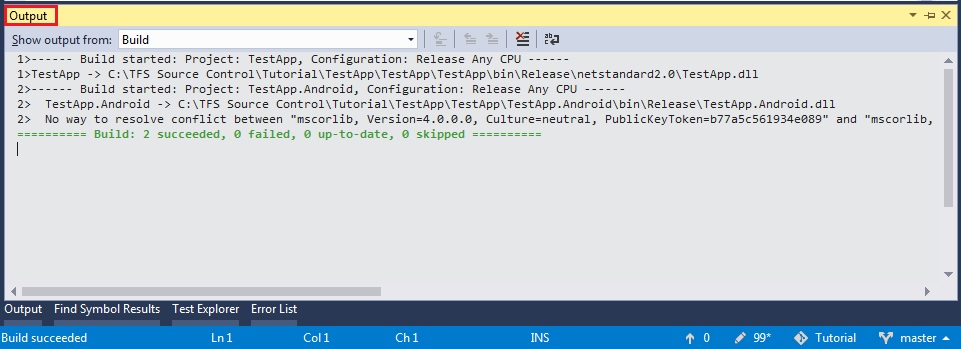
Make sure the location selected matches the folder where your repository was cloned



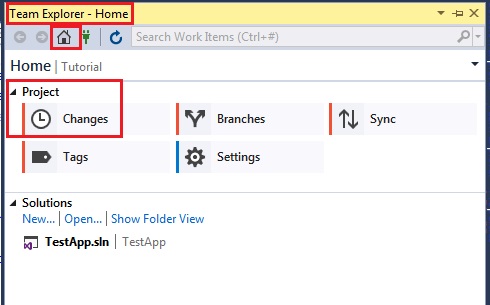
Open the Solution Explorer window in Visual Studio to build the solution. Set build platform, configuration and start-up project as shown below.



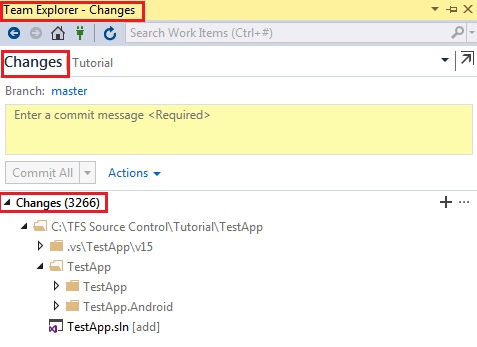
After a few minutes, the Output window in Visual Studio should indicate that the build succeeded.



Now, open the Team Explorer window in Visual Studio. Click on the ‘Changes’ button under the ‘Home’ tab.



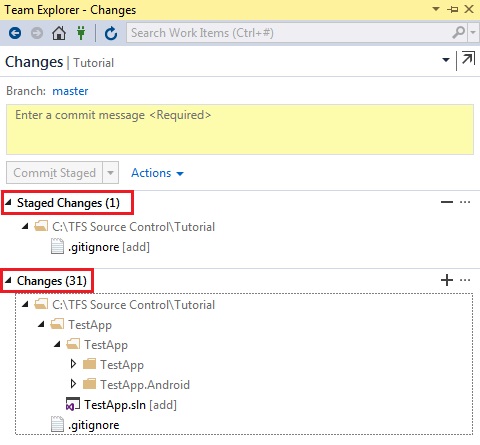
Do not be shocked at the number of changes. As Most of these can be ignored.



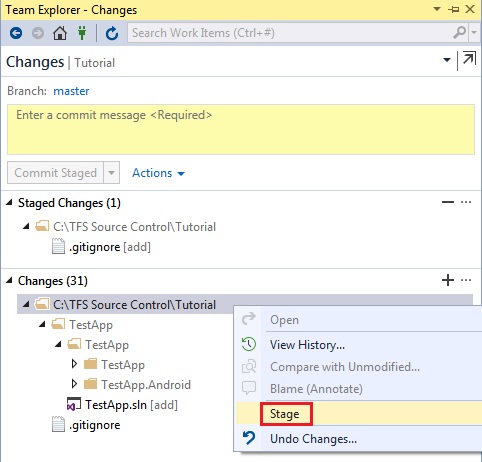
Ignore the following items:

* .vs\TestApp\v15
* TestApp\TestApp\bin\Release\netstandard2.0
* TestApp\TestApp\obj
* TestApp\TestApp.Android\bin\Release
* TestApp\TestApp.Android\obj

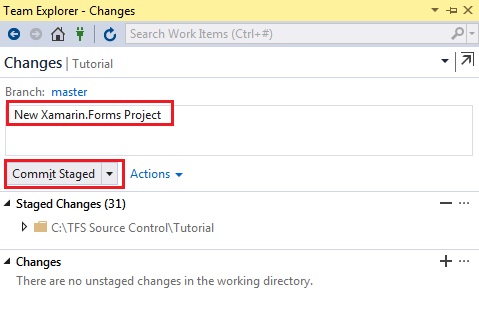
This should lead to a considerable decrease in the number of changes as well as an update of the ‘.gitignore’ file which will appear under ‘Staged Changes’.



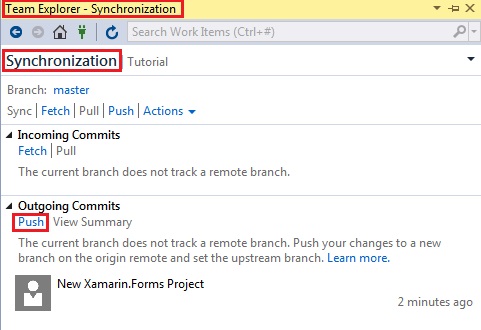
Stage the remaining changes



Enter an appropriate comment and commit the changes

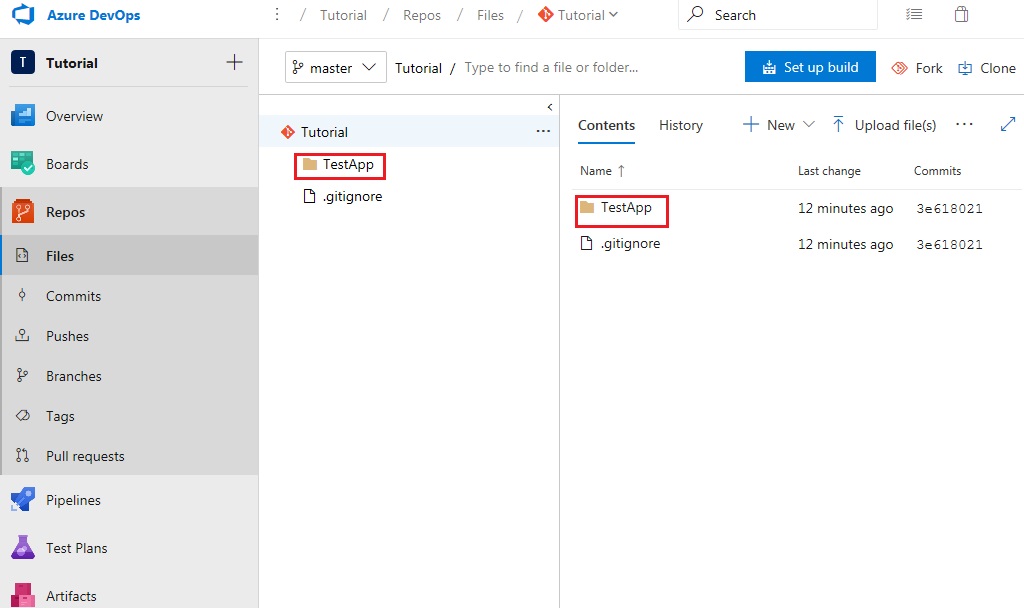


Click on the ‘Sync’ button under the ‘Home’ tab of the Team Explorer window in Visual Studio and ‘Push’ your commit to the server.



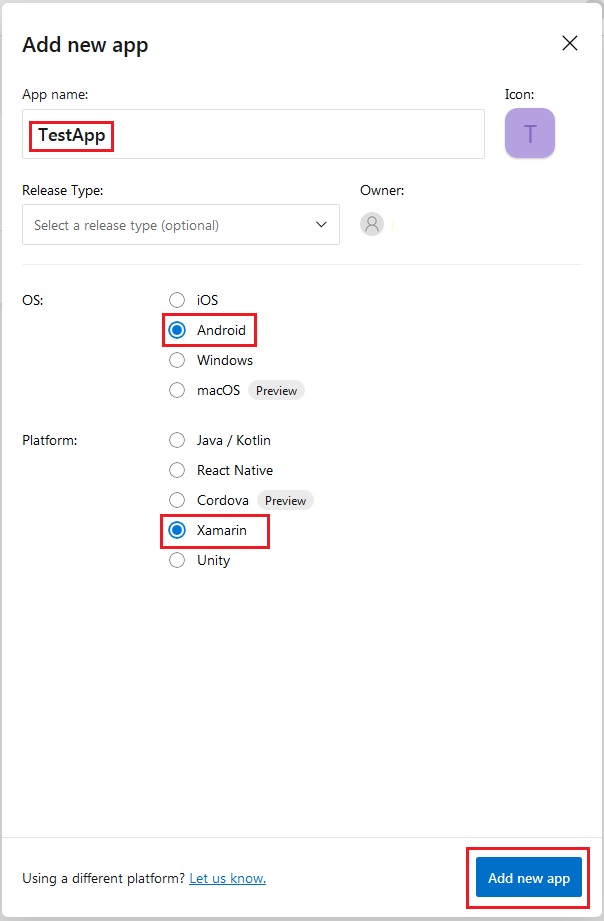
Return to your browser and refresh the tab or access your repository using the following link: [https://dev.azure.com/{username}/\_git/{projectname}](https://dev.azure.com/%7busername%7d/_git/%7bprojectname%7d); where {username} is your Azure account name and {projectname} is ‘Tutorial’ or whatever name you chose for the project.

Instead of the previous greeting message, you should now be presented with your TestApp project.



Now that the project has been successfully added to your Azure repository, open a new tab and navigate to your app Center dashboard using the following link: https://appcenter.ms/apps

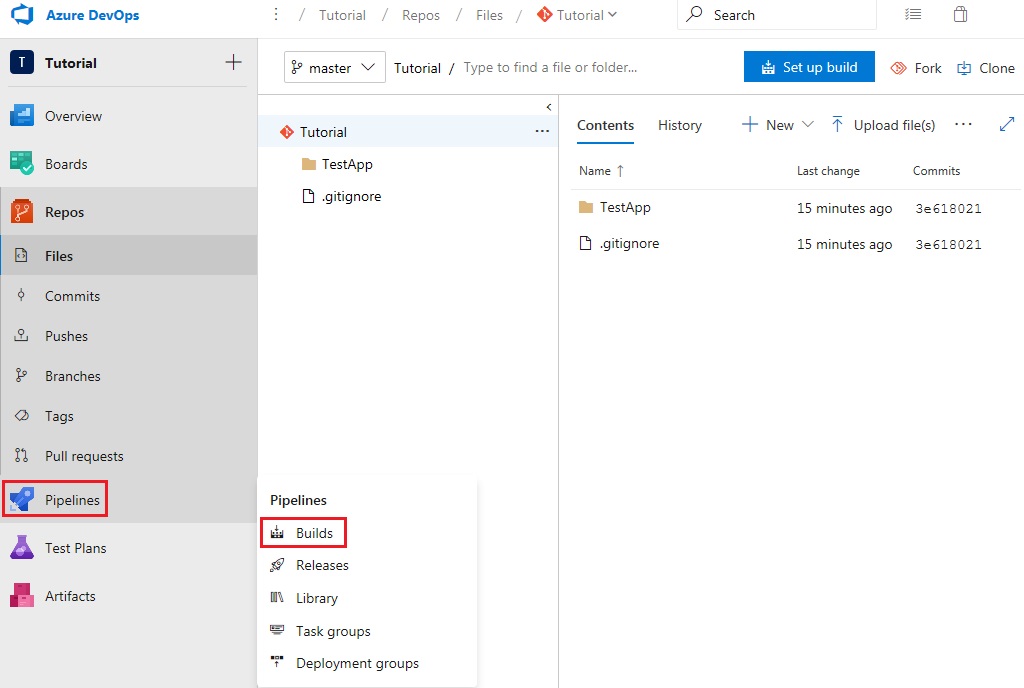
Once you’ve signed in, select the ‘Add new’ button to create a new app as shown below. Make sure that the app name matches the name of the Xamarin.Forms project created in Visual Studio.



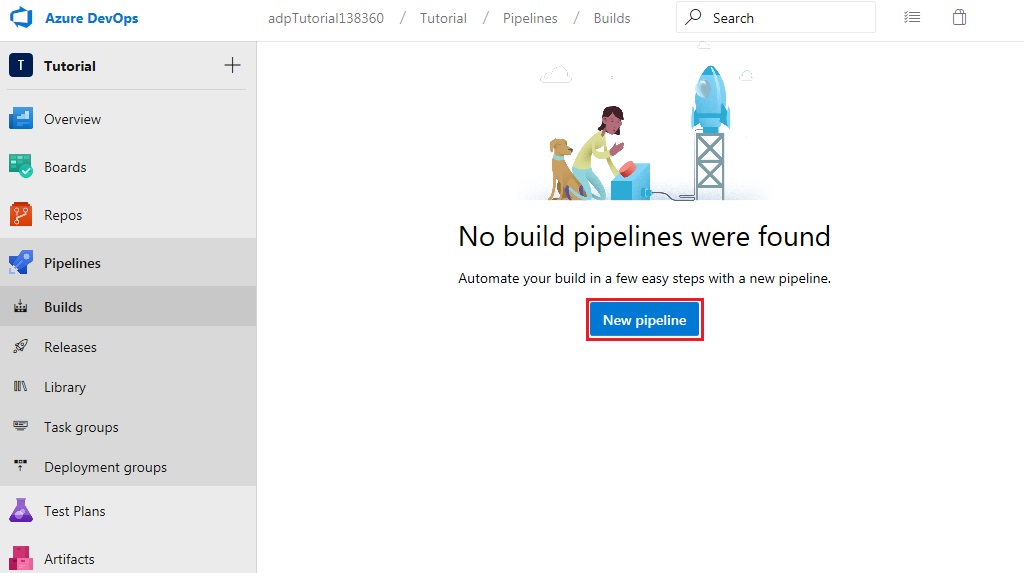
Now return to your Azure Repository to setup the build pipeline

Setting up the build pipeline

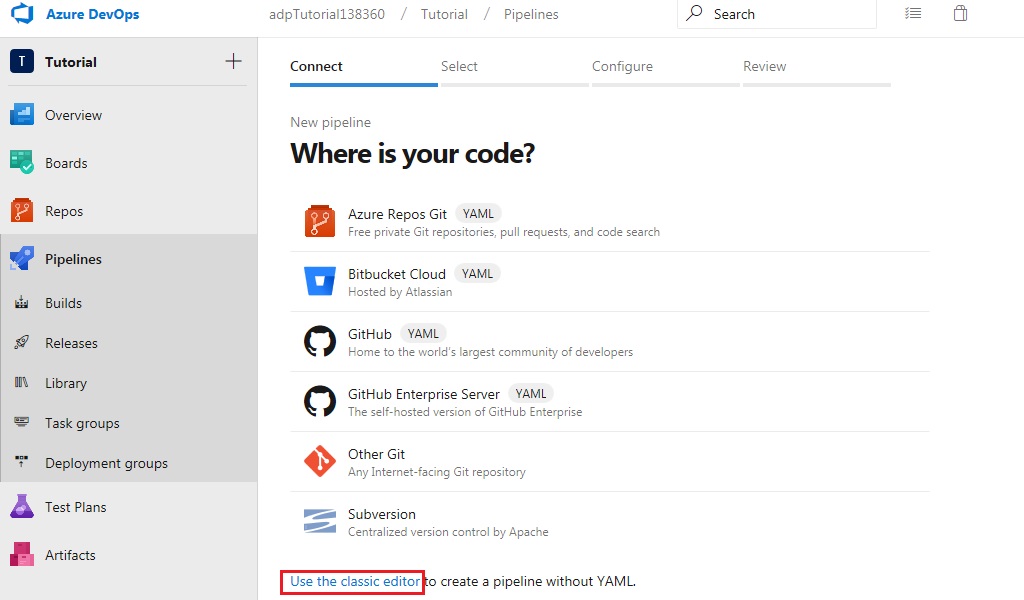
Navigate to the build pipeline



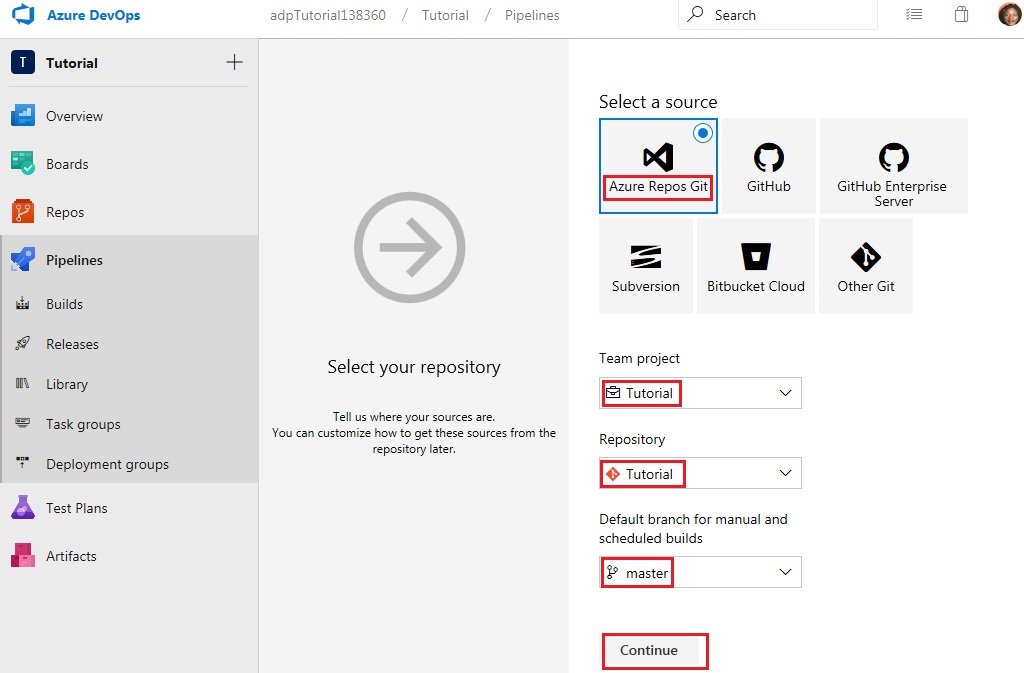
Click ‘New pipeline’



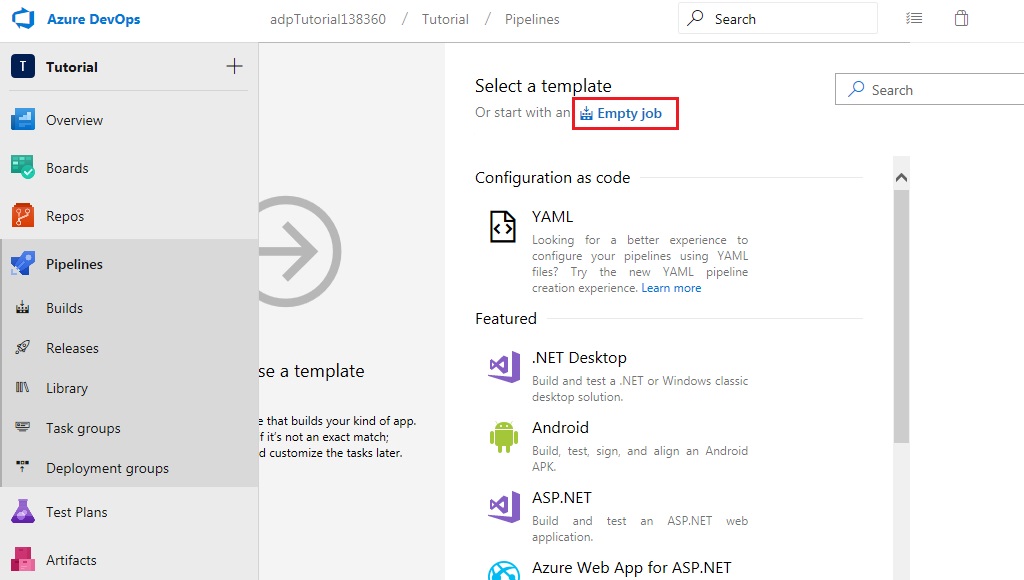
Use the classic editor



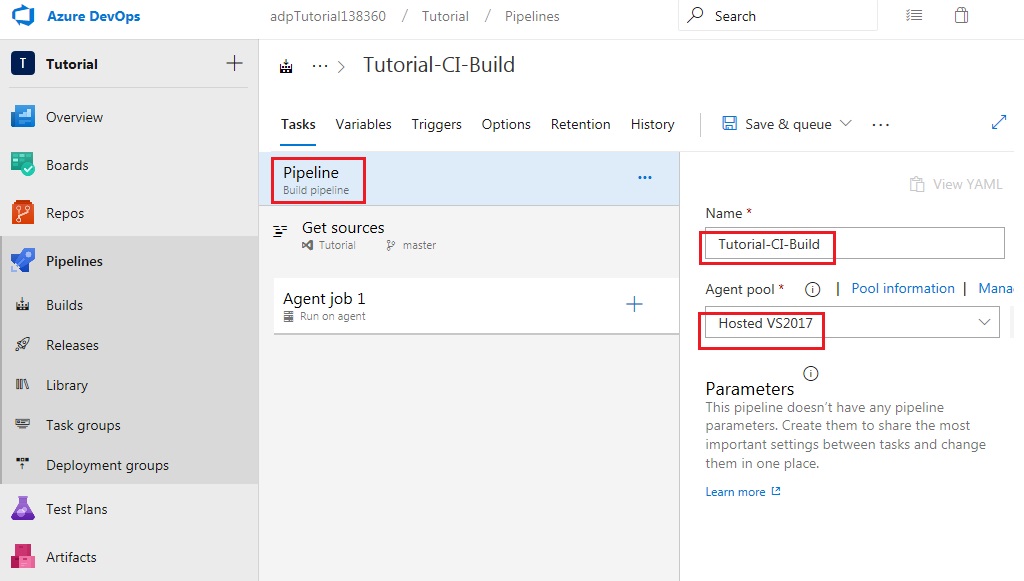
Continue with details set as shown below



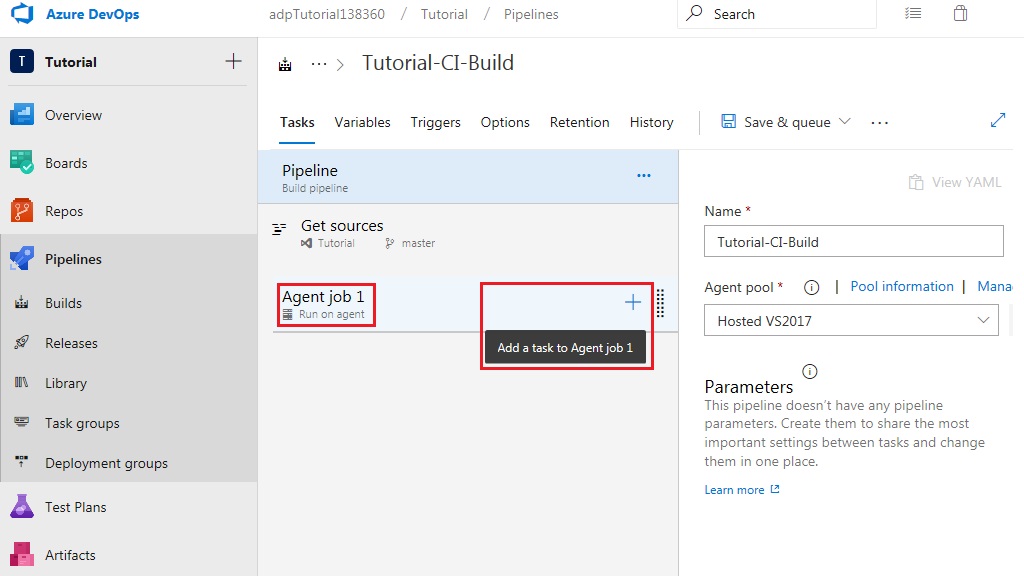
Start with an empty job



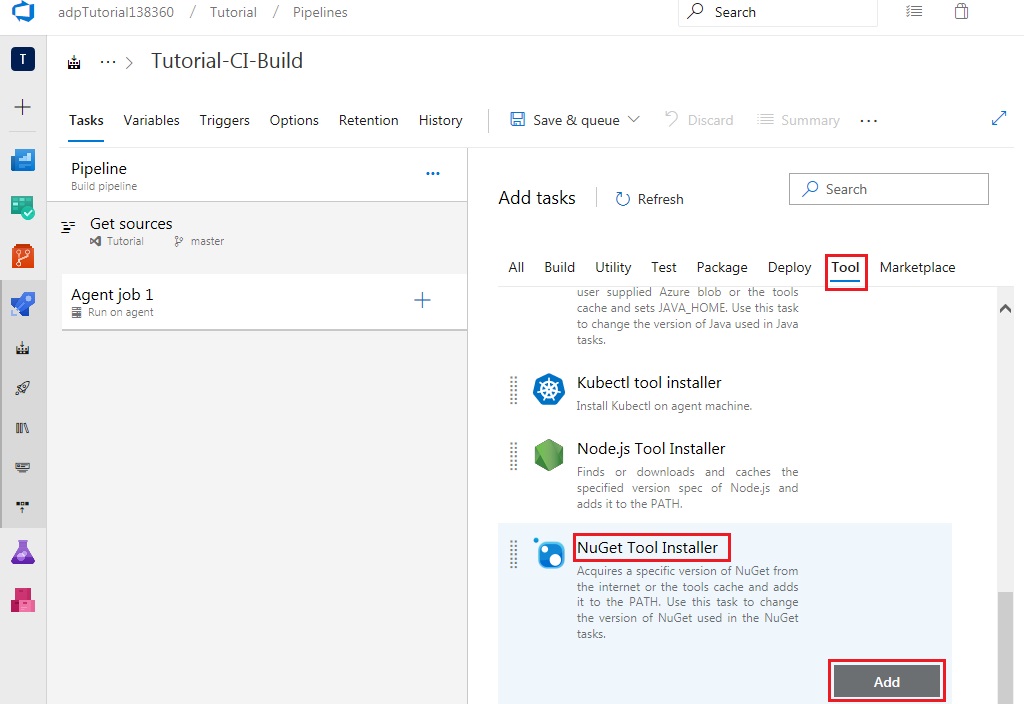
Give a Name to the pipeline and select ‘Hosted VS2017’ as the ‘Agent pool’



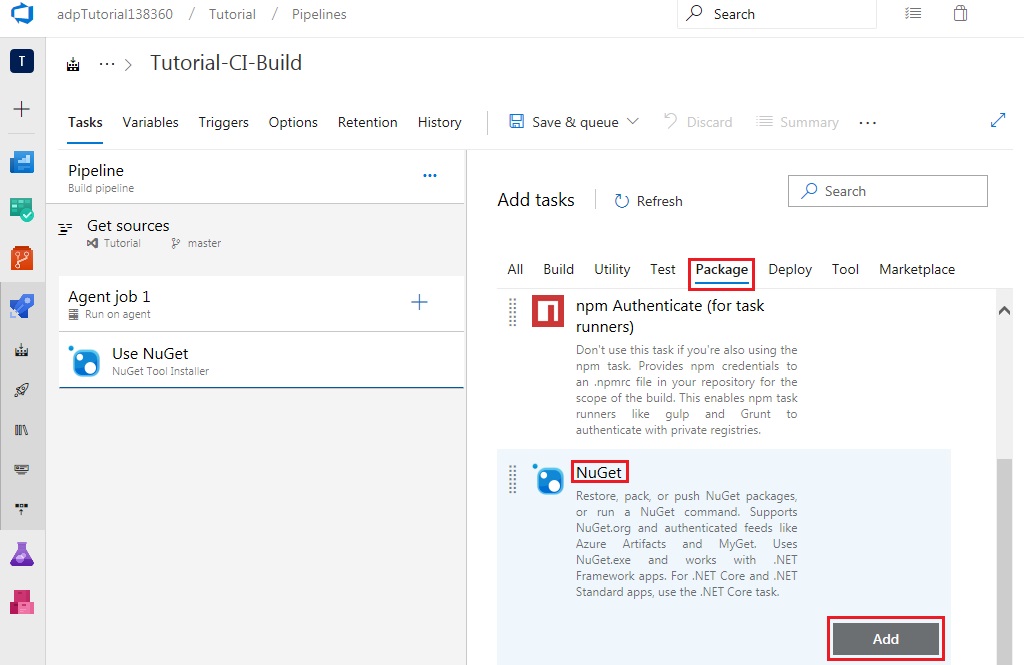
We shall proceed by adding five tasks to job 1



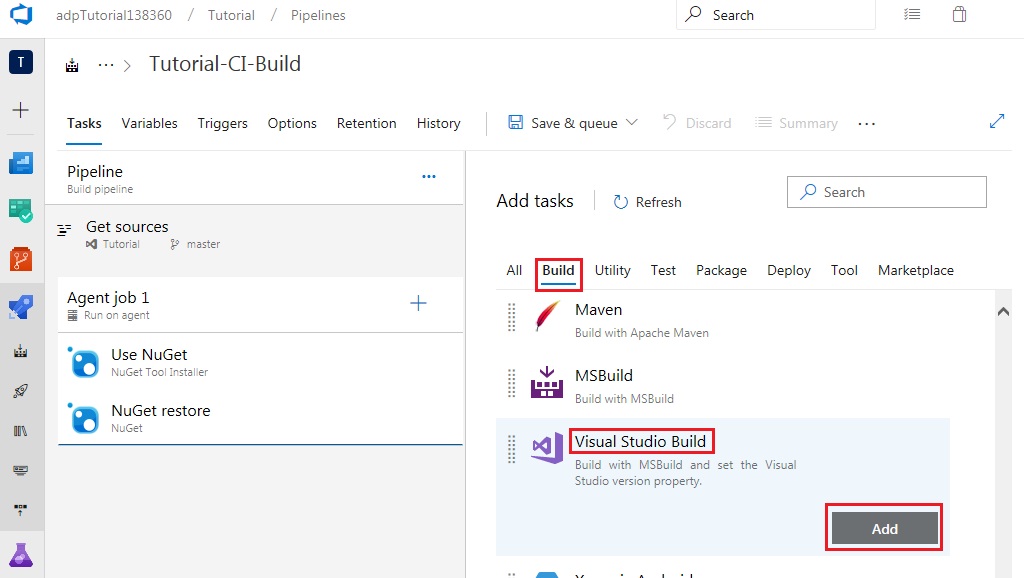
Add the ‘NuGet Tool Installer’ from the ‘Tool’ tab



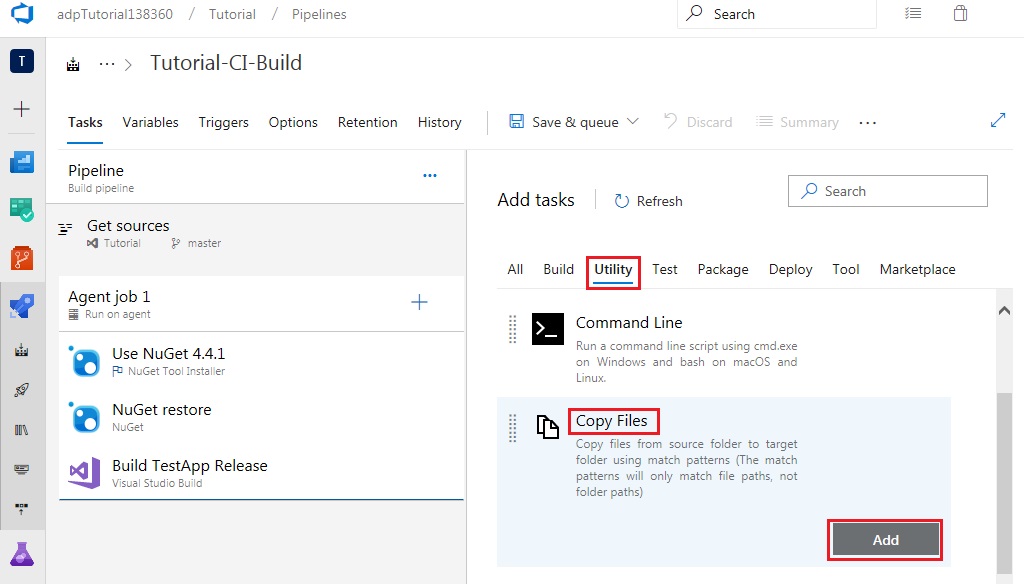
Add ‘NuGet’ from the ‘Package’ tab



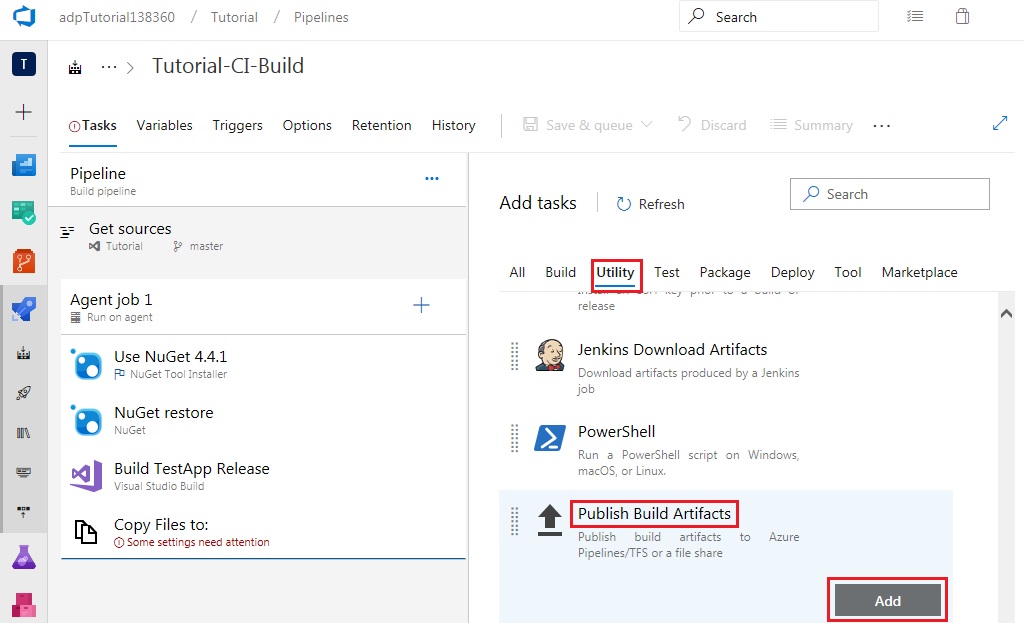
Add ‘Visual Studio Build’ from the ‘Build’ tab



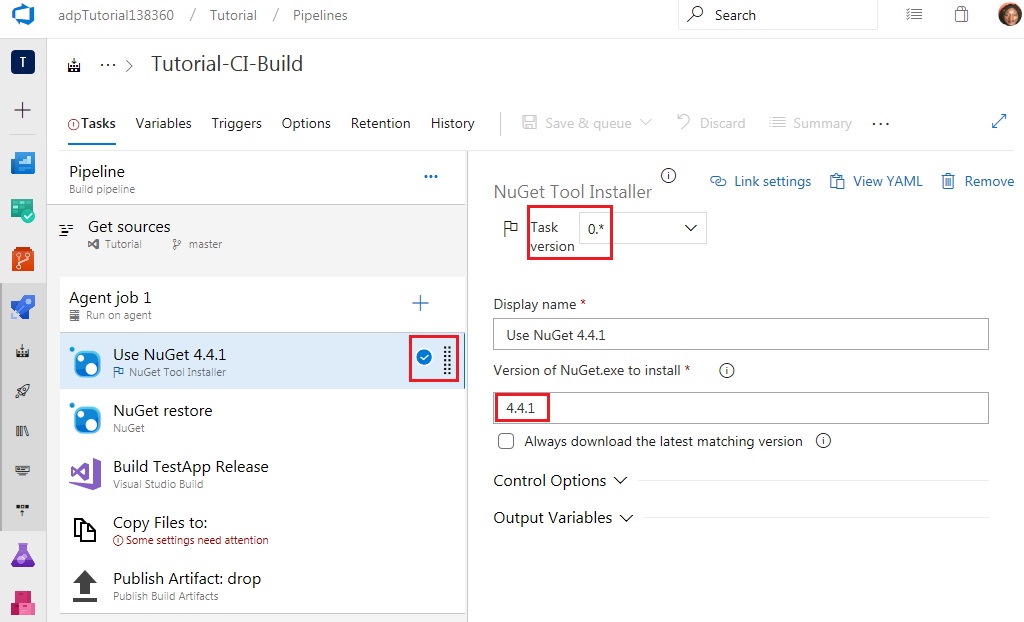
Add ‘Copy Files’ from the ‘Utility’ tab



Finally, add ‘Publish Build Artifacts’ from the ‘Utility’ tab



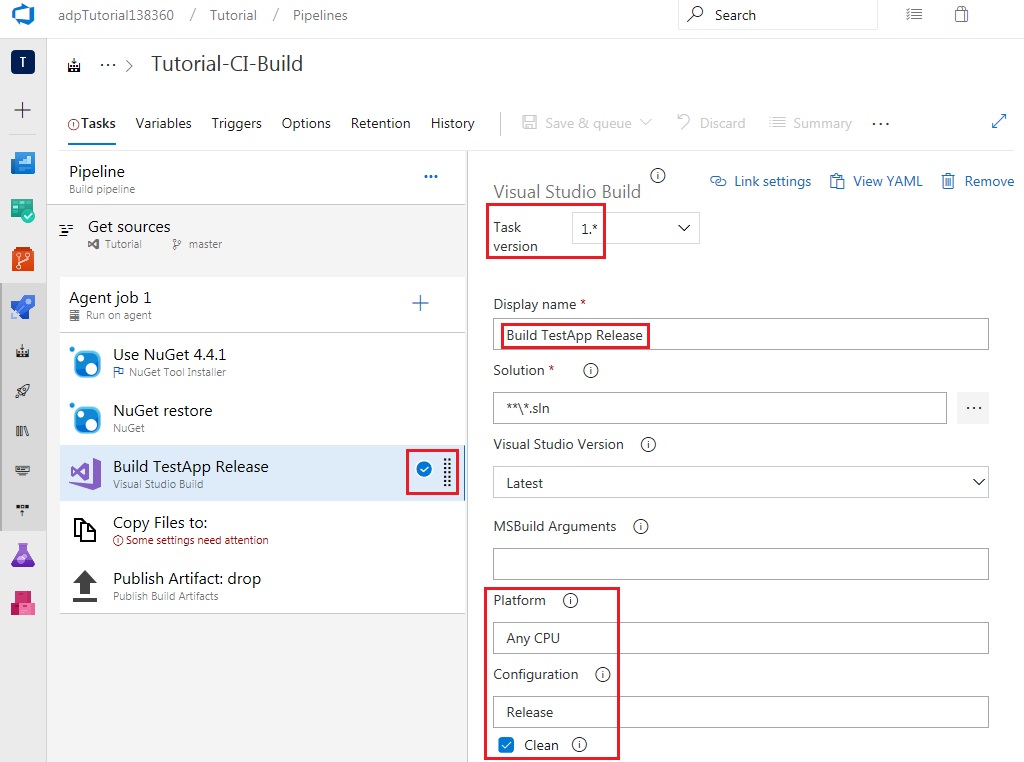
Select the ‘NuGet Tool Installer’ on the left-hand side and specify version 4.4.1



Select ‘NuGet’ on the left-hand side and browse the repository to find the .sln file. Alternatively, leave this as default.

?

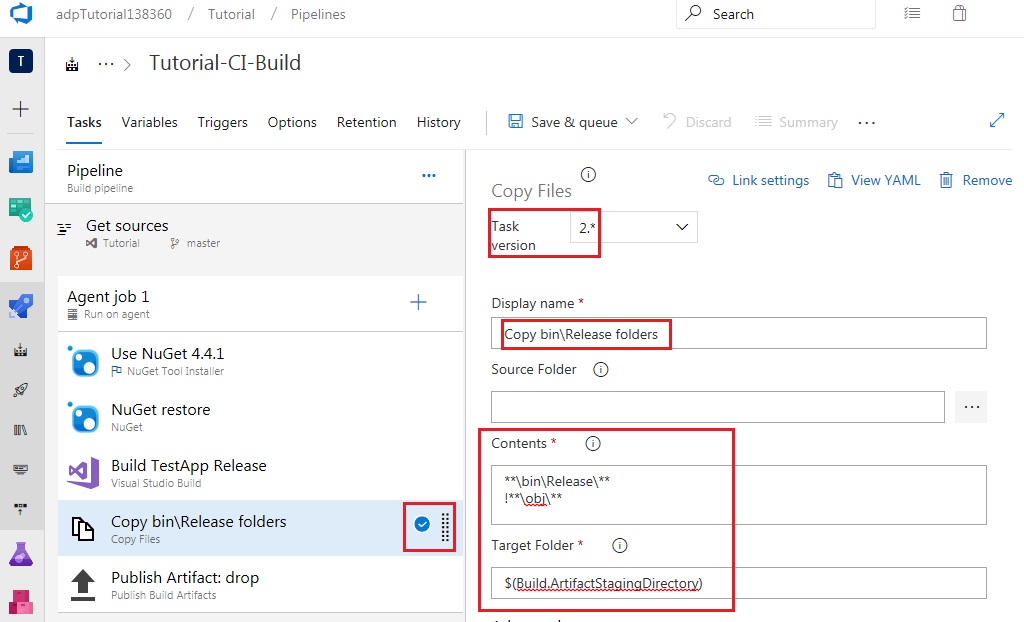
Select ‘Visual Studio Build’ on the left-hand side, specify a Display name, check the ‘Clean’ box and specify the Platform and Configuration



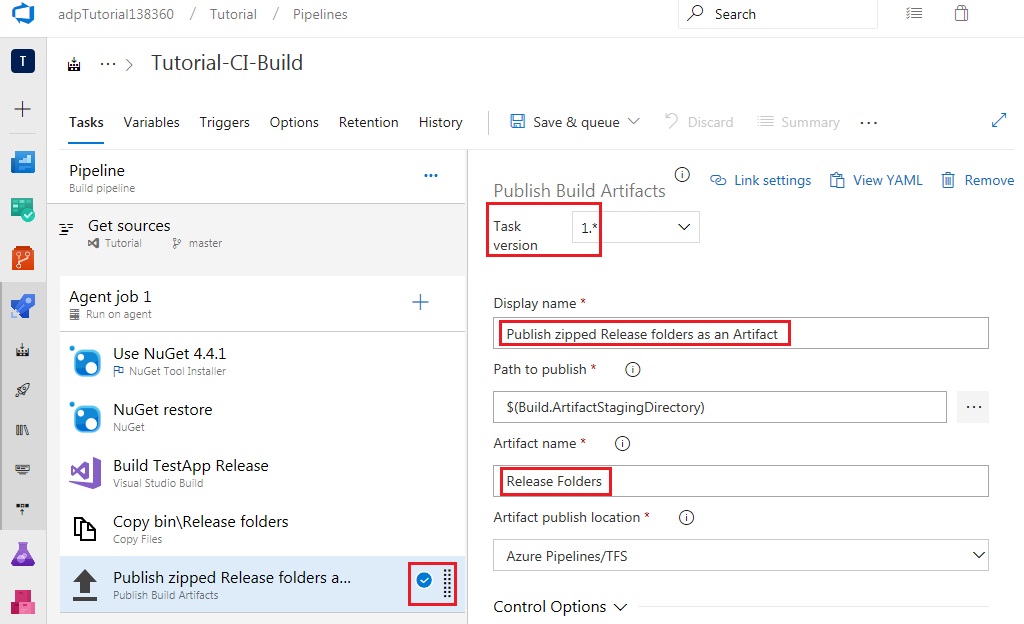
Select ‘Copy Files’ on the left-hand side, specify a Display name and set the Contents and Target Folder as shown.

Note that the Target Folder, $(Build.ArtifactStagingDirectory), will be the same one used in the Path to publish field of the final task, Publish Build Artifacts.

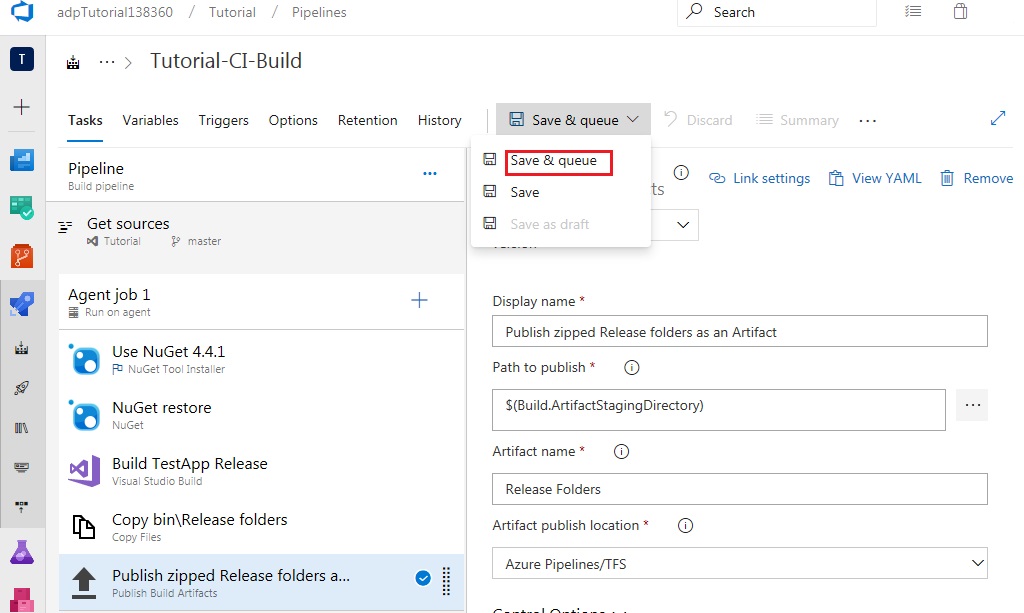
The first line of the contents field indicates that all items in any bin\Release folders present within the repository will be copied. The second line indicates that all items in any obj folders present within the repository will be excluded from being copied.



Select ‘Publish Build Artifacts’ on the left-hand side and specify both a Display name and an Artifact name.

select CI on the Triggers tab

Save and queue



Basic pipeline

Additional tasks

Setting up the release pipeline

Basic pipeline

Deployment to app center

Deployment gates

Approvals