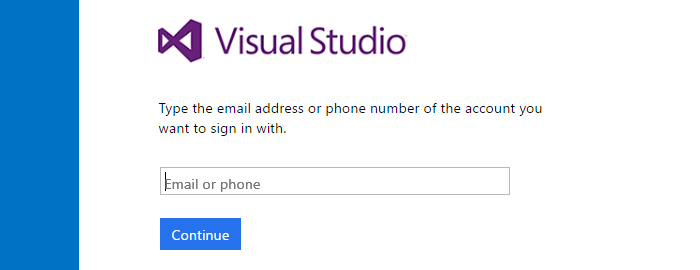
CI CD Pipeline using Visual Studio Team Services

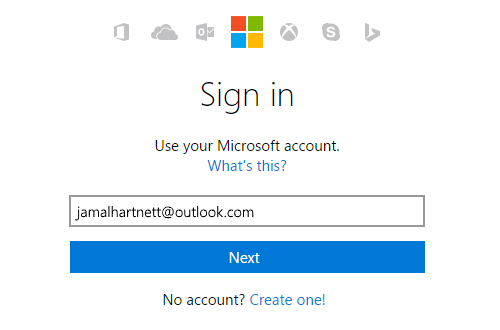
**Sign up for Visual Studio Team Services with a personal Microsoft account**

1. Go to Visual Studio Team Services. Enter your email address for your Microsoft account.

If you're a Visual Studio subscriber and get Visual Studio Team Services as a benefit, use the Microsoft account associated with your subscription.

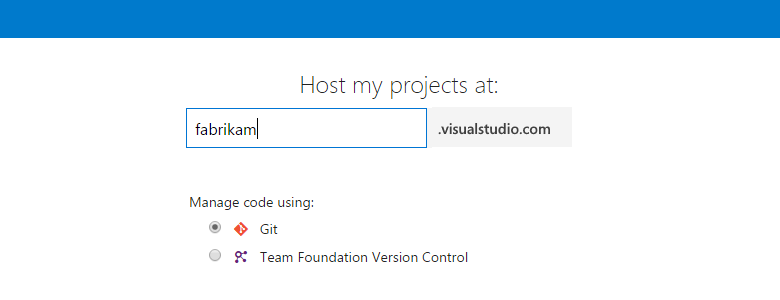


1. On the Microsoft account sign-in page, enter your email address for your Microsoft account. Enter your password, and finish signing in.

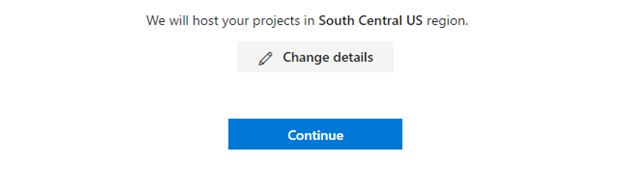


If you don't have a Microsoft account, you can create a Microsoft account at this time.

1. Name your Visual Studio Team Services account. To manage your code, choose Git or Team Foundation Version Control.



1. Confirm your account's location.



Team Services will create your first team project and will use Agile as your default work item process to organize your work. Choose **Change details** to rename your team project, change the account location, or select another process, like Scrum.

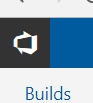
1. After Visual Studio Team Services creates your account and team project, add your code, work items, or more users.

Congratulations, you're now a Visual Studio Team Services account owner!

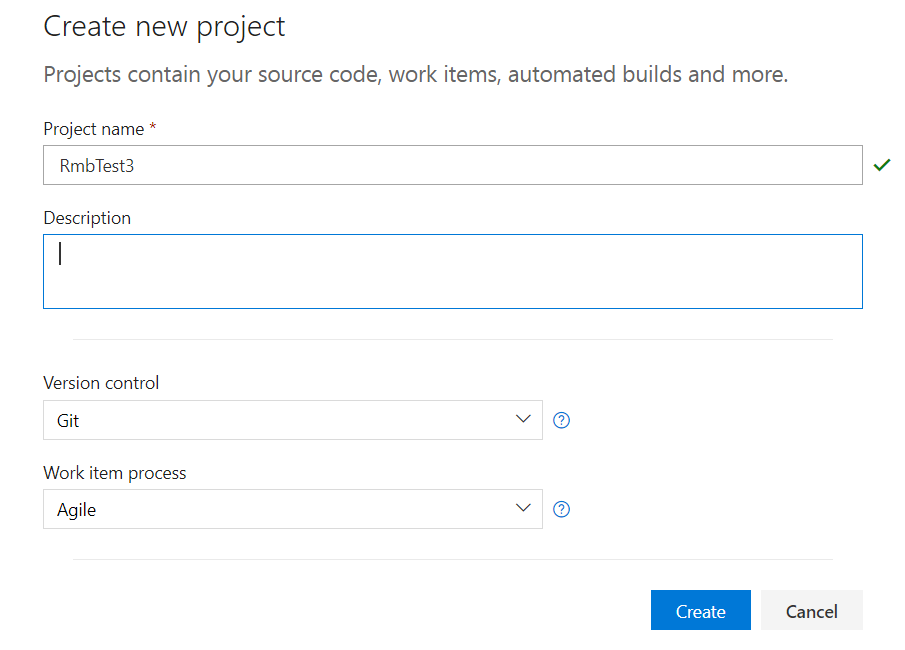
To sign in to your Team Services account at any time, go to https://{youraccount}.visualstudio.com.

**Create a new team project**

Navigate to the home page. Easiest way to do that is to click on your VSTS home icon



Once at the VSTS home page, click on the “New Project” button. A new page will appear, fill in the details. Choosing “Git” as your version control and “Agile” as your Work item process.



**Import repository from GitHub**

Next you will need to either create a new repository, or upload pre-existing code. You can also import a repository from another repository host such as GitHub. We will be importing a sample project from GitHub.

Expand “or import a repository” section and click on the import button.

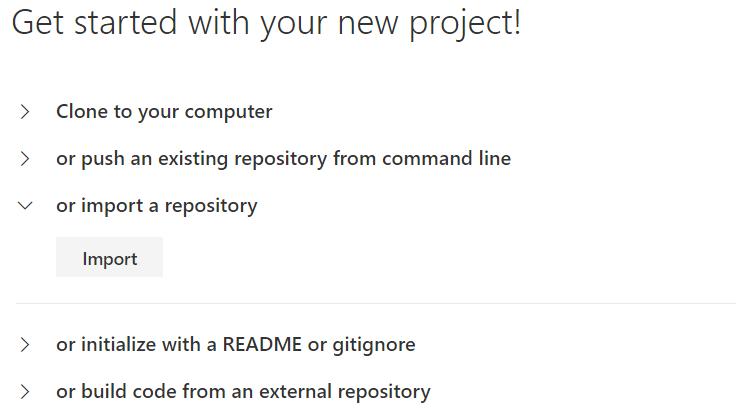
Select Git as your repository type.

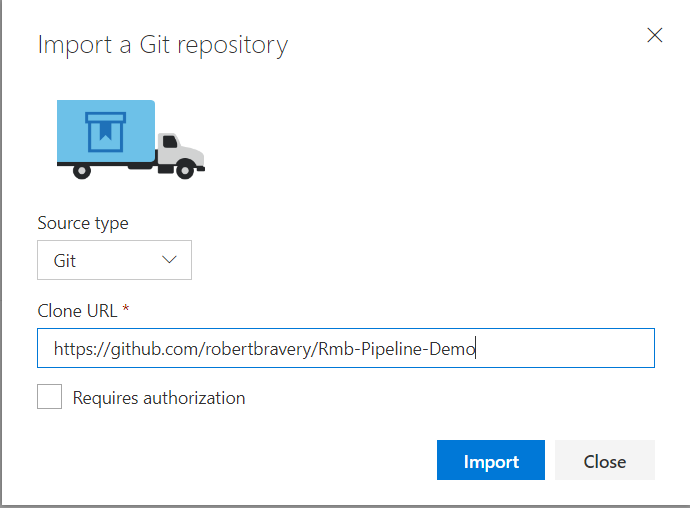
Enter the following into the repository URL

<https://github.com/robertbravery/Rmb-Pipeline-Demo>

Leave the “Requires authorization” unchecked

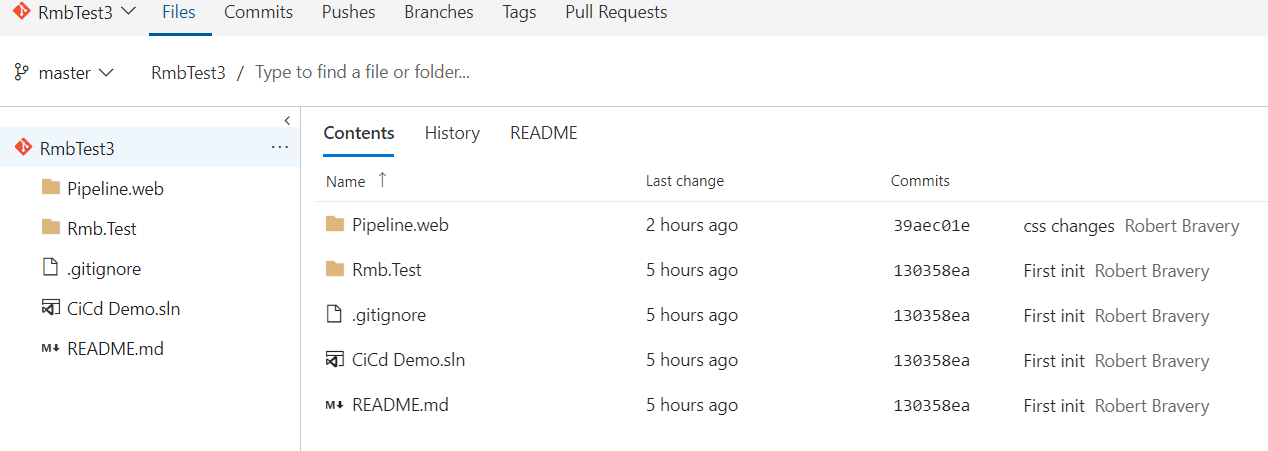
Click the import button.





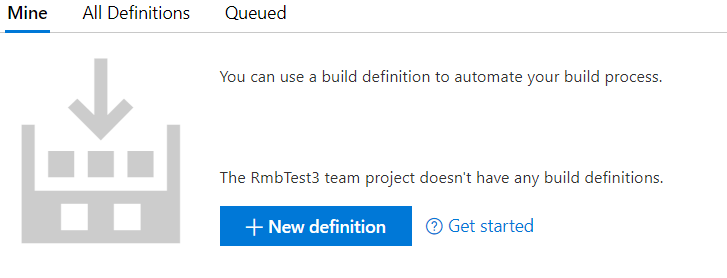
**View Code in repository**

You can now view the imported code in your code repository. IF you have not been taken to your code directly, then you can just select the “Code” menu item from the menu bar.



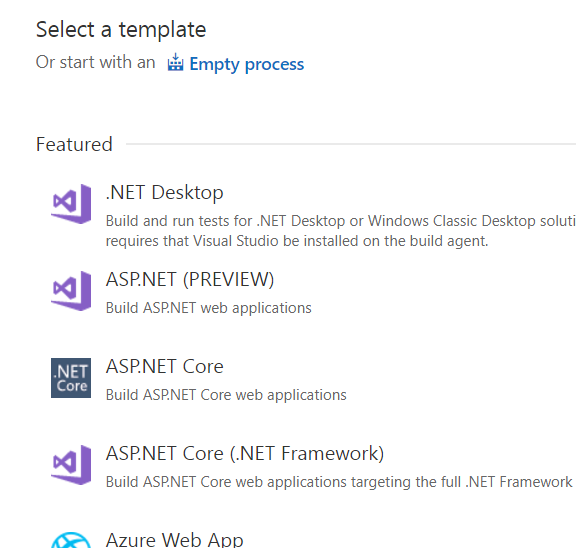
**New Build definition**

You now need to create a build pipeline. Select the “Build and Release” menu, then the “Builds” sub-menu. A list of build definition will be displayed. If no build definitions have been created you will see, as in the below image, a page that requires you to create a new definition. Click on the “New definition” button.

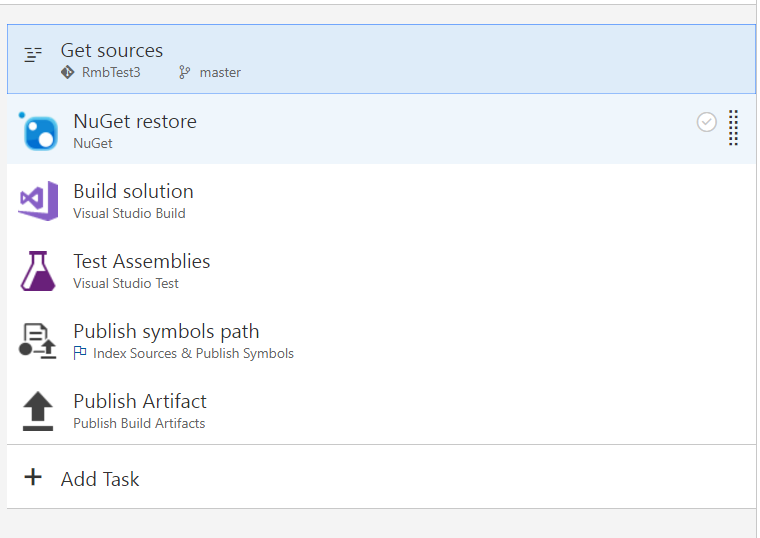


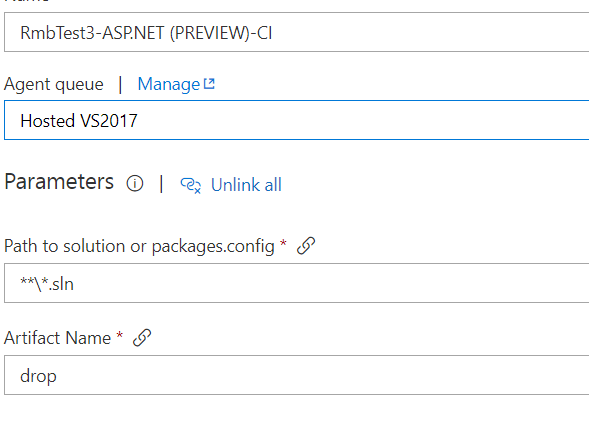
A new page will be presented with a list of available templates on the left hand side. There are many templates to choose from, for this exercise we will select the “Asp.net (Preview)” from the available list and click apply.

This will give us some default tasks to build an ASP.net website and code.



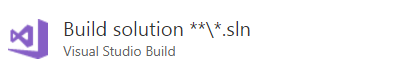
A list of default task will be displayed on the left hand side. Select the “Process” task, if not already selected, and then select “Hosted VS2017” from the Agent Queue dropdown list for the build process. Leave the other defaults as is.



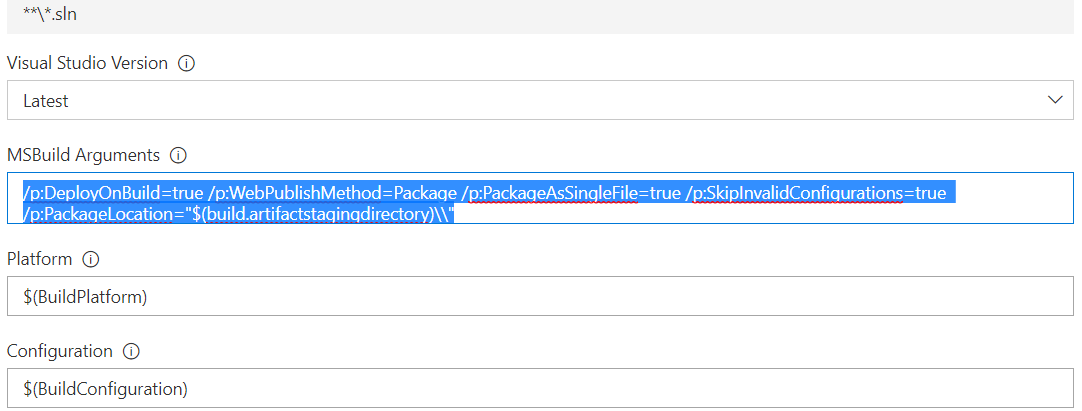


No need to change anything for the “Get resources” and “NuGet restore” tasks. Leave the selected defaults for the “Get resources” and “NuGet Restore” tasks

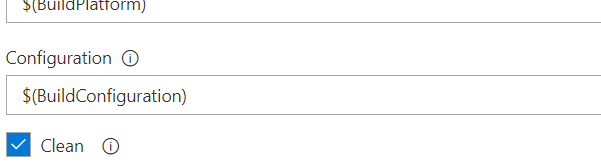
Select the “Build Solution – Visual Studio Build” task from the available tasks on the left.



Keep the suggested defaults. We will only change the values for the “MSBuildArguments”. Delete all MSBuild Arguments



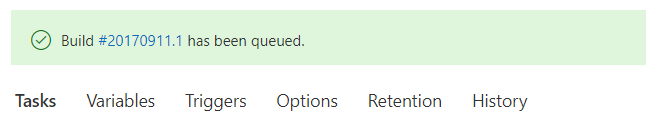
If you cannot see it, scroll down a little to make sure “Clean” is checked to clean out the build folder at each build.



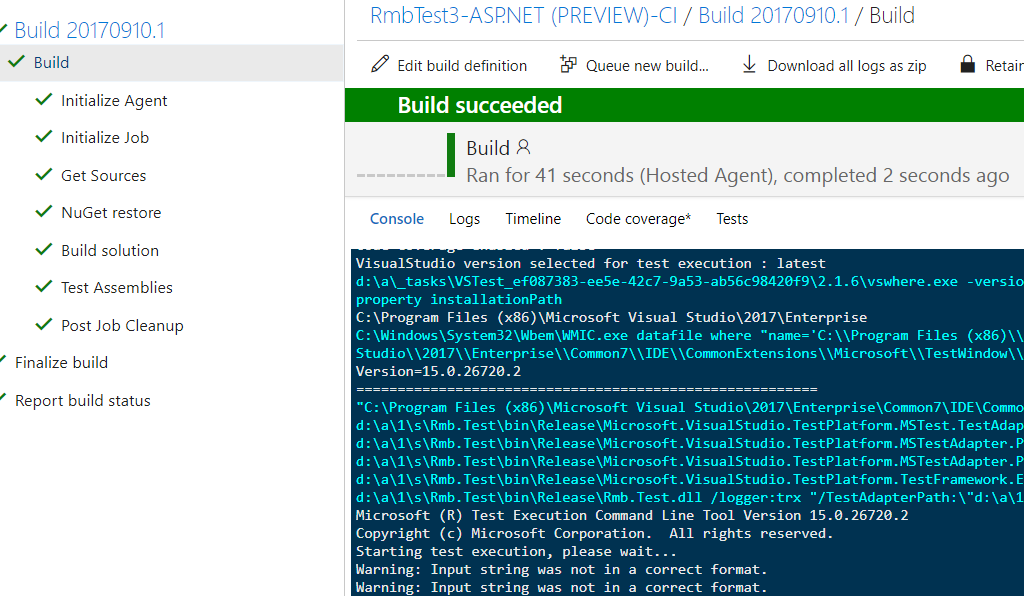
Next, Leave defaults for “Test Assemblies” Task.

Disable “Publish Symbols Path” and “Publish Artifacts” task for now. We will come back to this later. Right click the task and select Disable.

Now select “Save and Queue” from menu to save and Queue your first CI pipeline. Doing so will queue your build definition. Notice that the build has been queued with a generated build id. Primarily consisting of the day and a sequential build number for that day.

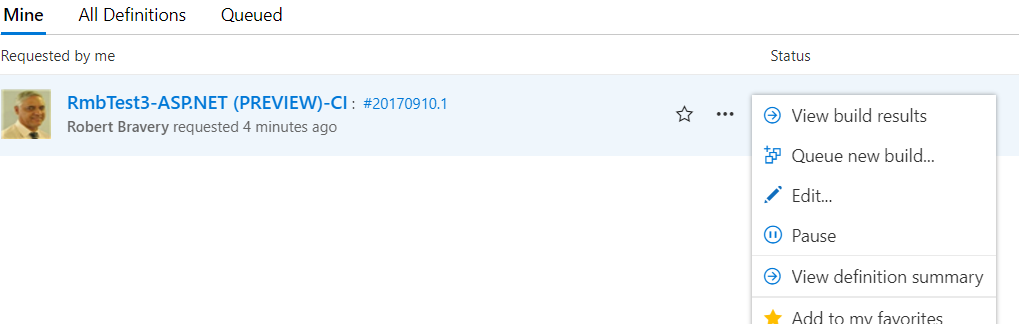


Selecting the build version will take you to the view of the build process.

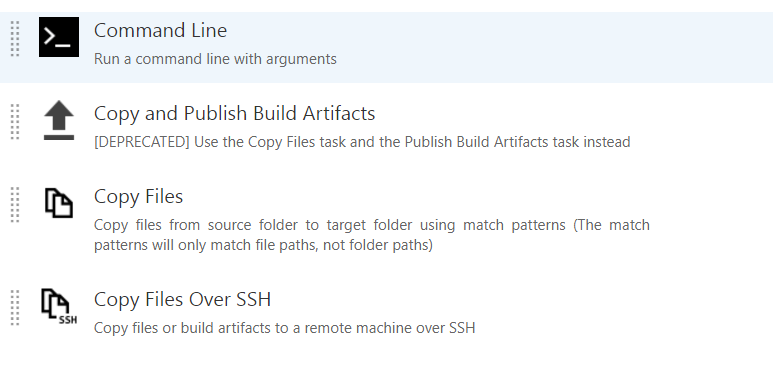


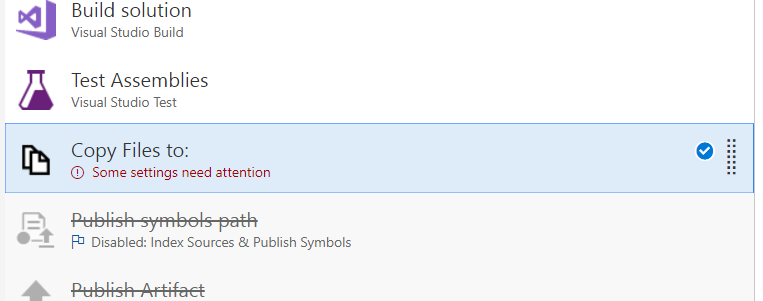
**Adding a new task to the pipeline**

Select the build menu to show your build definition. Edit the build definition by selecting the ellipse corresponding to the build definition, then select edit from the drop-down menu



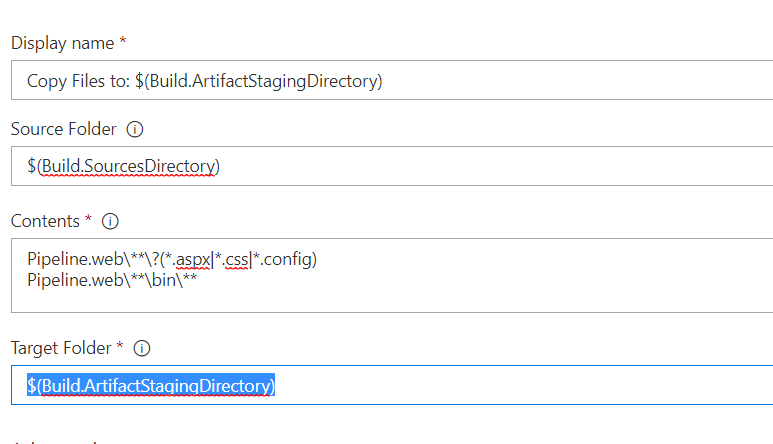
Click Add New Task and select Copy files from the available templates and apply. You can also drag and drop the selected template into position amongst your current templates. Place the Copy Files template below the Test Assemblies task to create a Copy Files Task





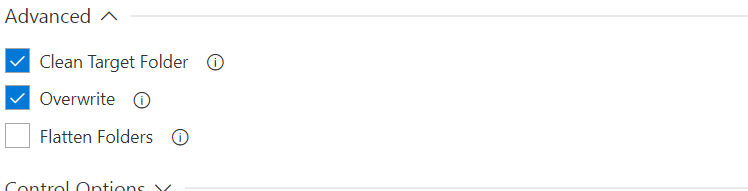
Fill in the source folder, content and Target Folder boxes with the below.

* Source Folder: $(Build.SourcesDirectory)
* Content: Pipeline.web\\*\*\?(\*.aspx|\*.css|\*.config)
  + Pipeline.web\\*\*\bin\\*\*
* Target Folder: $(Build.ArtifactStagingDirectory)



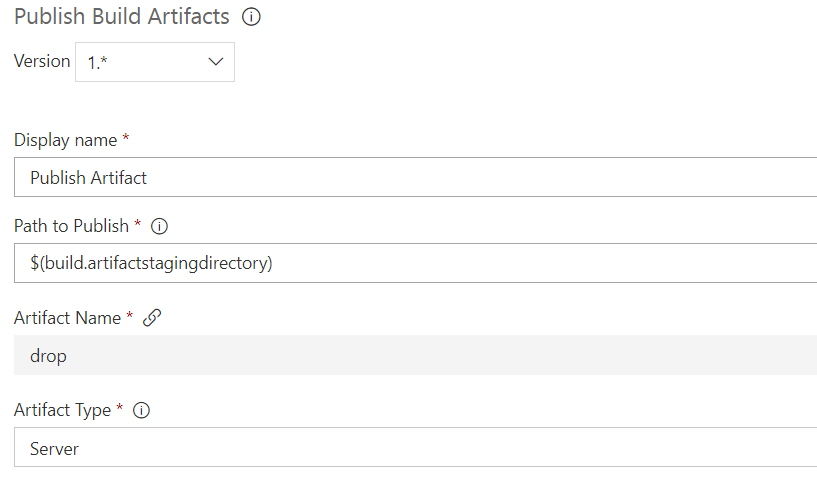
Expand the advance section and check “Clean Target Folder” and “Overwrite”. This does the obvious in the it will clean and overwrite the target folder. This is the Artifact staging area.

Save and Queue to test

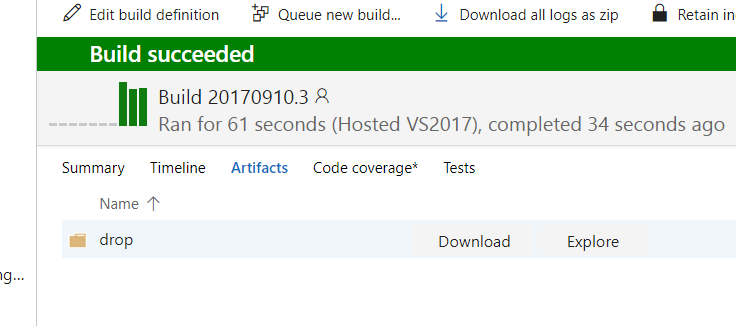


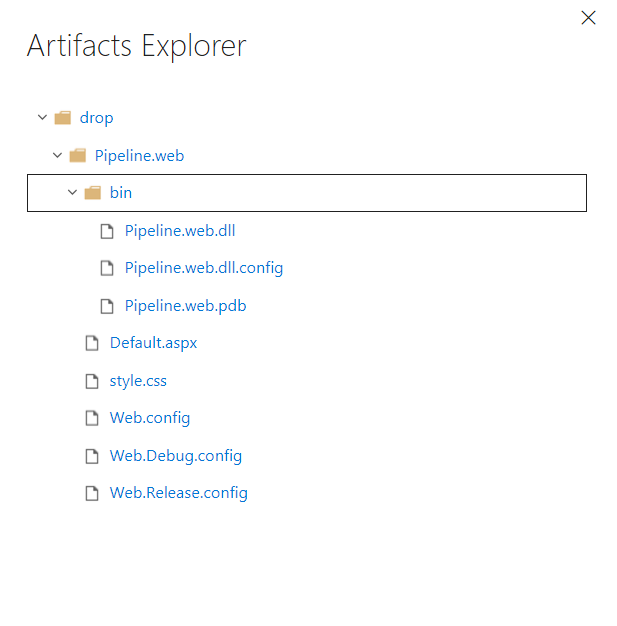
**Publish Artifacts**

Right click and enable the “Publish Artifacts” task. Leave the defaults as is. Save and Queue



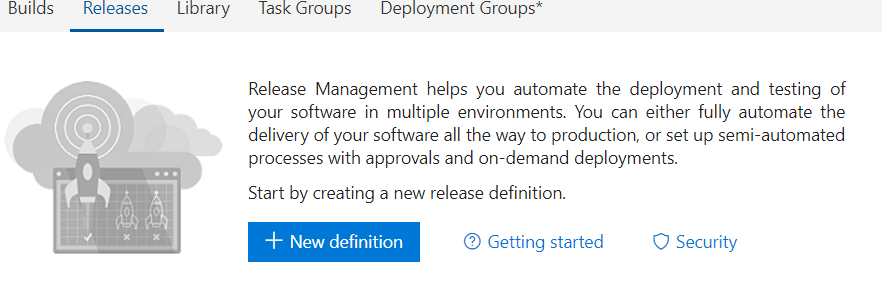
Select the completed build and notice the Artifact menu. Select the artifacts menu and explore the artifacts



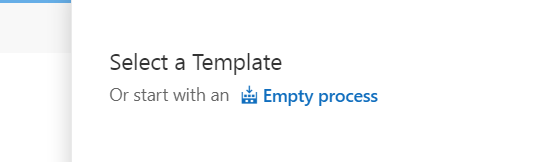


**Release and Deploy**

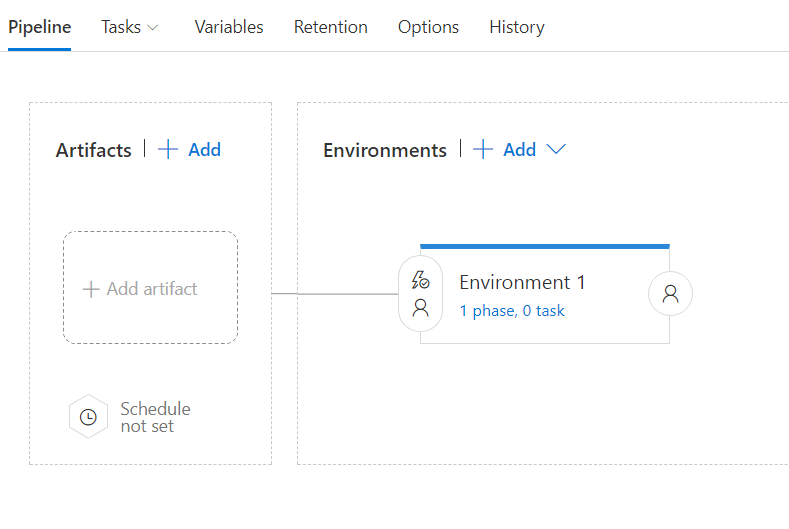
Select Releases from the menu and create a new release definition



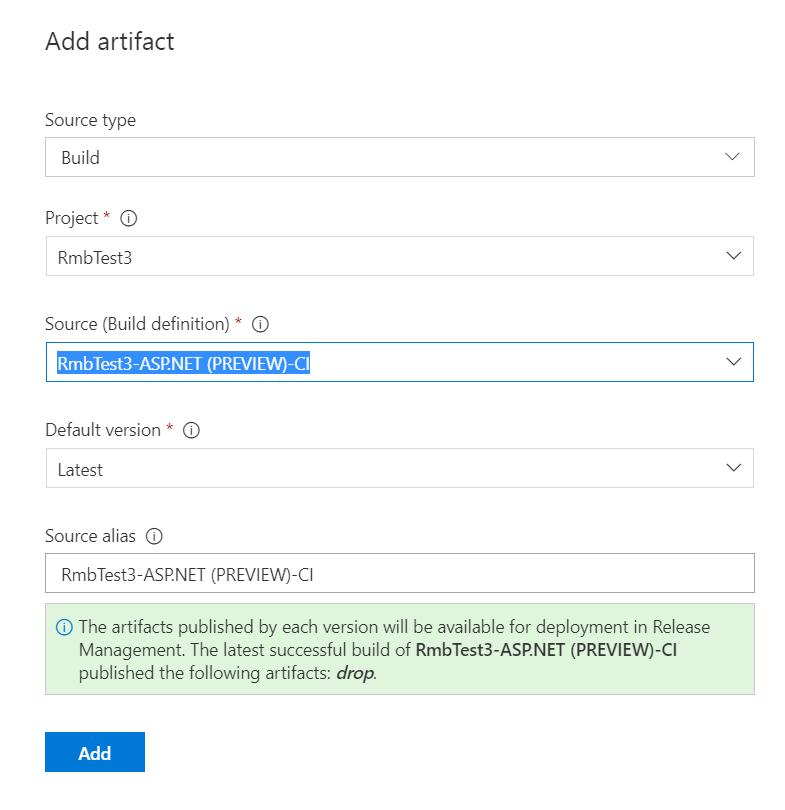
Select start with an empty process



Click “Add artefact” to add Artifacts produced during the build process. This enable VSTS to create a release to deploy. Release and deploy are two different processes. You create a release and then deploy that release to an environment.



Select your Build Source Definition from the available dropdown menu. VSTS should automatically discover any published artifacts. If it does not, check to see of you have created and saved a “Publish Artifacts” task. If you have created and a “Publish Artifact” task the you need to Queue the build so that VSTS can create and publish the artifacts.

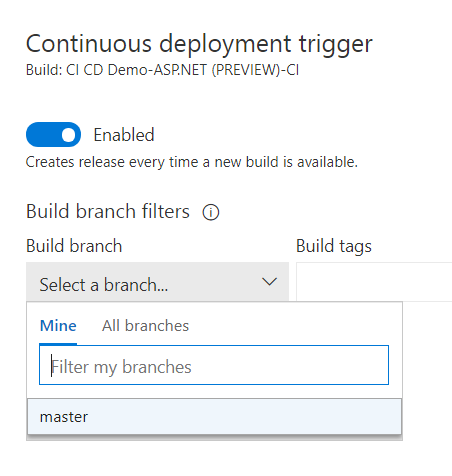


Click add to add the artifact to your release pipeline. Remember to save the release.

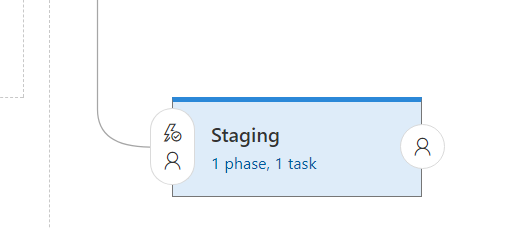
Click the trigger icon associated with the newly created release on the left.

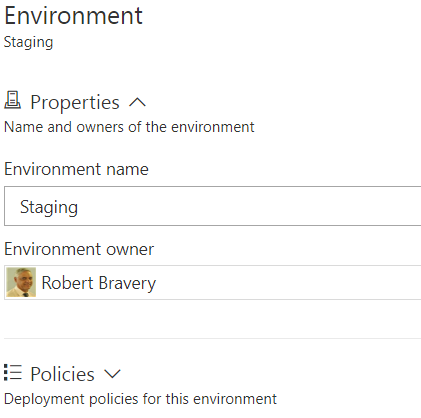


This will display a trigger section on the right. Make sure that the continues deployment trigger is enables. Then make sure that the correct branch is selected. For this exercise we will select master from the drop down box.

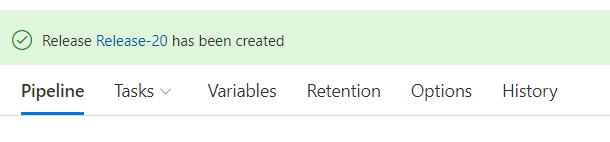


Select the “environment” and change the name to something like “staging” by changing the environment name on the properties section displayed on the right hand side. Remember to save.

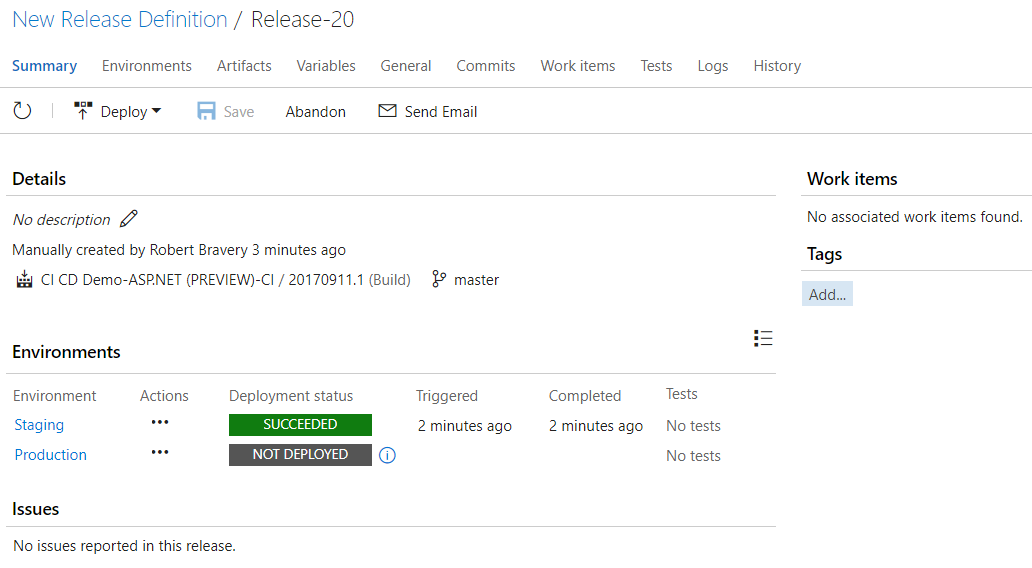




Select “Release” from the menu. Create the release using the latest build (or one of your choosing) and then queue the release. Just like the build, a release id will be generated as a hyperlink. Click the hyperlink to navigate to the release page.

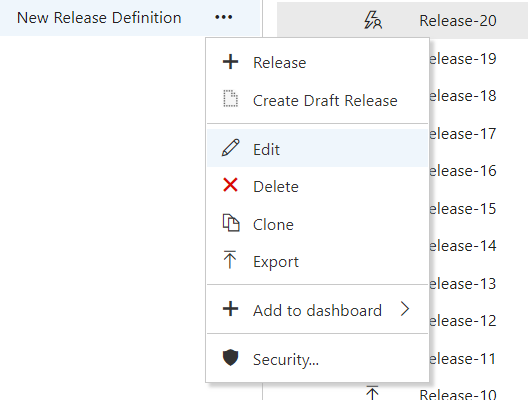


A new page will be displayed. Here you will be able to see any details of the release as well as which environments were or were not deployed to. Some environments will have an automatic trigger to deploy to (will go through this step later). From there you can selectively and manually deploy to environments.

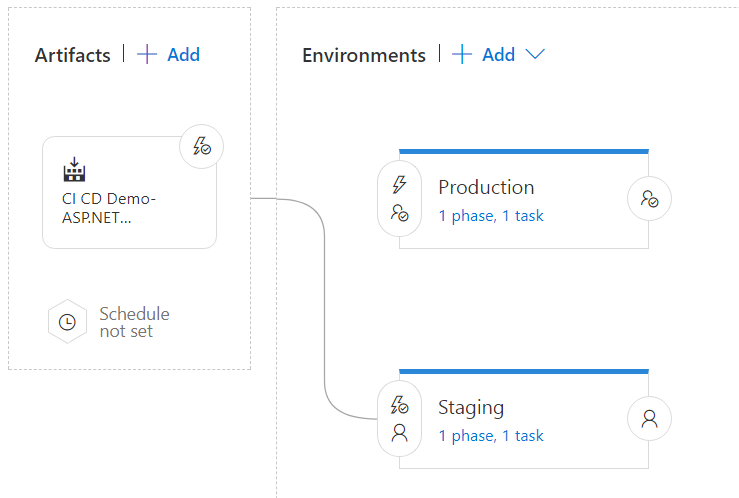


**Deploying to an environment**

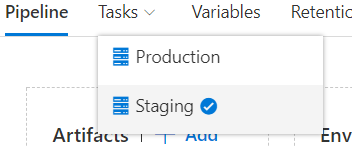
Select the releases menu under “Build and release”. This will take you to the release management page. On the left hand side is a section of filters. Select the ellipse for the “New release definition” filter. A menu opens, select edit to edit the release pipeline.



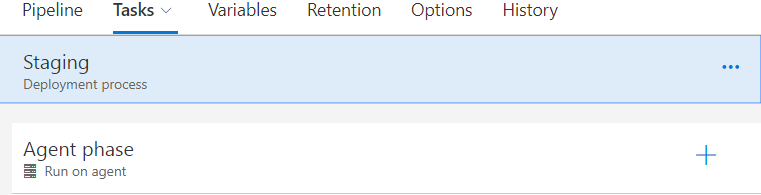
This will take you to your previously created release pipeline page.



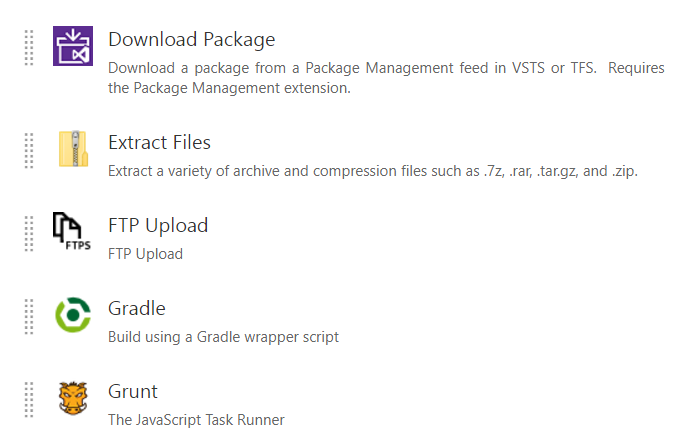
Select the task menu and then select the Staging environment, or whichever environment you would like to set up a deployment to.



You will see a Staging Deployment process with an Agent Phase. Click the “+” icon on the Agent phase to add a deployment task.



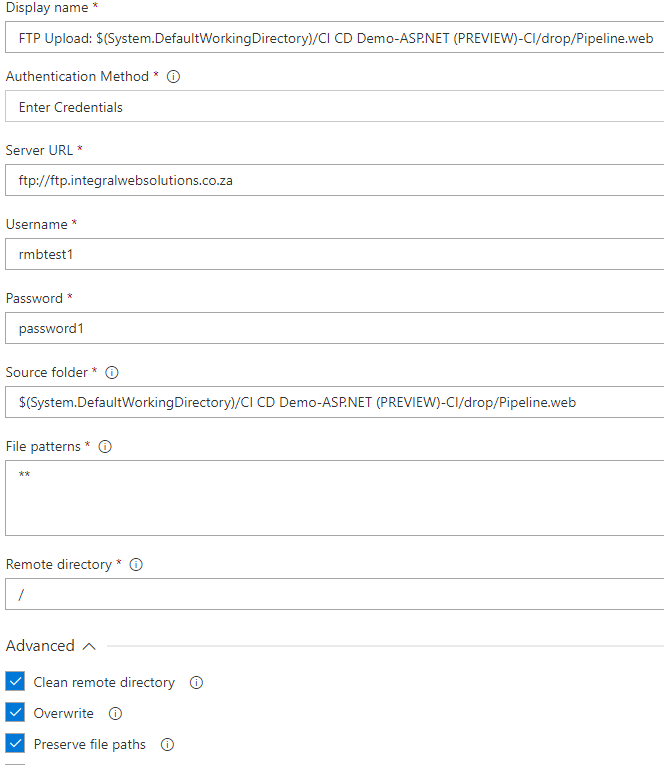
You will then see numerous available deployment tasks. For this demo we will be making use of the “FTP Upload” task. Scroll down till you see the “FTP Upload” task and add it to your task list.



Select the FTP Upload task and fill in the properties.

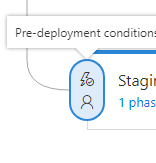
* Authentication method. Select “Enter Credentials” from the dropdown box.
* Server URL. Enter ftp://ftp.integralwebsolutions.co.za
* Username. Enter your team’s user name as in team3
* Password. Enter your team’s password as in password1
* Source folder. Use the folder navigation to navigate to and select “CI CD Demo-ASP.NET (PREVIEW)-CI/drop/Pipeline.web” folder.
* File patter. Leave file pattern as \*\*
* Remote directory. Enter in the route directory by enter a “/”
* Expand the advance section and check
  + Clean remote directory
  + Overwrite
  + Preserve file paths.

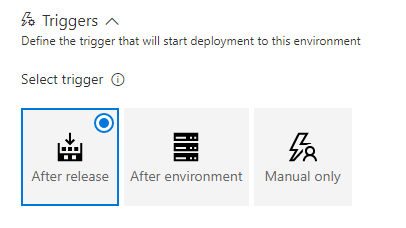
Save your FTP Upload task.



After your task is saved go back to the release pipeline by selecting the pipeline menu item

Select the “Deployment conditions icon” and change the trigger to “After Release” this ensure a continuous automated deployment process.





Save the release. Now create and queue a new release by selecting the Release menu and selecting Create release from the dropdown menu