DevOps Kata

**Integrated testing / technical debt management**

Last updated: 11/06/2016



1. **TABLE OF CONTENT**

[Overview 3](#_Toc466841267)

[Prerequisites 3](#_Toc466841268)

[Exercises 3](#_Toc466841269)

[Exercise 1: Create a web application 4](#_Toc466841270)

[Task 1: Create a web application 4](#_Toc466841271)

[Exercise 2: Add a unit test project 5](#_Toc466841272)

[Task 1: Create a unit test project 5](#_Toc466841273)

[Task 2: Run your unit test 10](#_Toc466841274)

[Task 3: Check Code Coverage 11](#_Toc466841275)

[Exercise 3: Create a new build 12](#_Toc466841276)

[Task 1: Create a new build template 12](#_Toc466841277)

[Task 2: Run your build 18](#_Toc466841278)

## Overview

Whether your app is on-premises or in the cloud, you can automate build-deploy-test workflows and choose the technologies and frameworks, then test your changes continuously in a fast, scalable, and efficient manner. Continuous testing with Visual Studio Team Services or Team Foundation Server ensures your app still works after every check-in and build, enabling you to find problems earlier by running tests automatically with each build. Choose the test technologies and frameworks you prefer to use. When your build is done, review your test results to start resolving the problems you find. Rich and actionable build-on-build reports let you instantly see if your builds are getting healthier. But it's not just about speed - detailed and customizable test results measure the quality of your app.

To determine what proportion of your project’s code is actually being tested by coded tests such as unit tests, you can use the code coverage feature of Visual Studio. To guard effectively against bugs, your tests should exercise or ‘cover’ a large proportion of your code. Code coverage analysis can be applied to both managed (CLI) and unmanaged (native) code. Code coverage is an option when you run test methods using Test Explorer. The results table shows the percentage of the code that was run in each assembly, class, and method. In addition, the source editor shows you which code has been tested.

### Prerequisites

1. In order to complete the lab
2. 1. Log on to your Visual Studio / MSDN subscription and create or use your own VSTS instance. Alternatively, you can use a team sandbox VSTS instance if you have one.
3. 2. Have an azure subscription available that you can deploy resource to.
4. 3. Visual Studio Enterprise

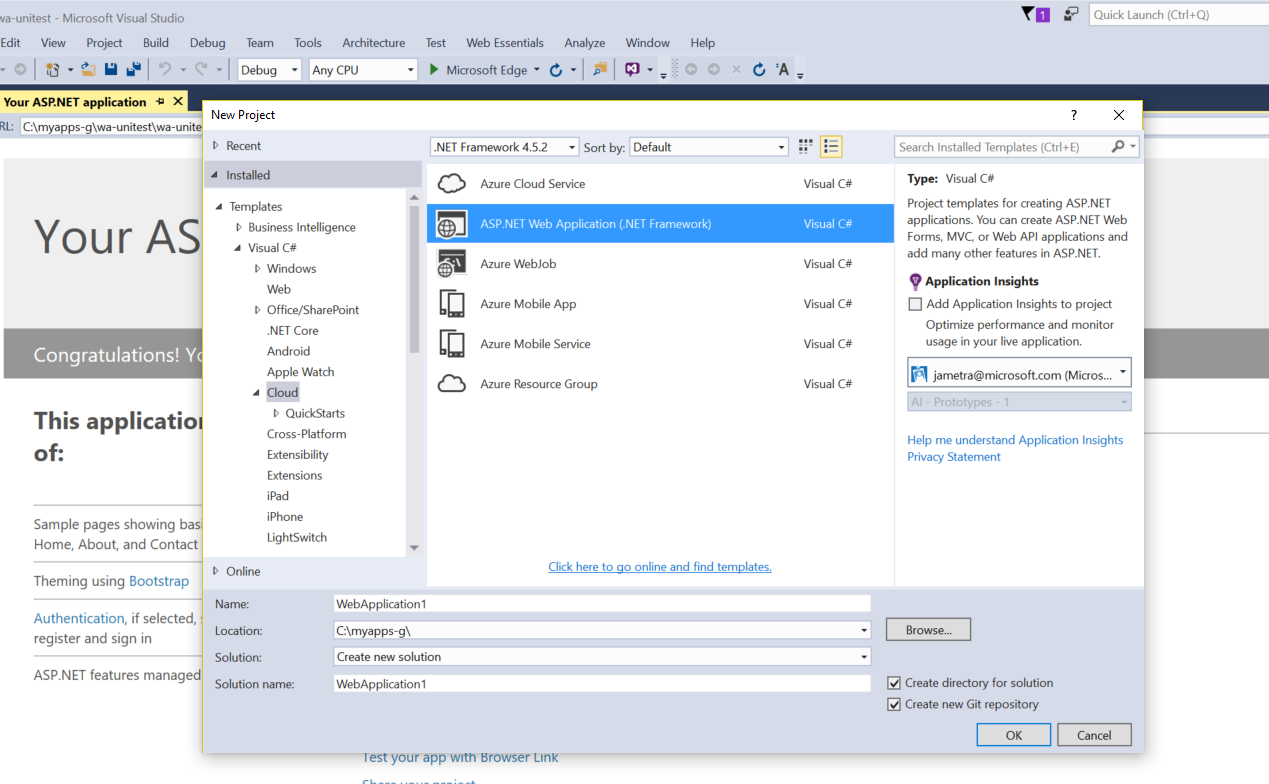
### Exercises

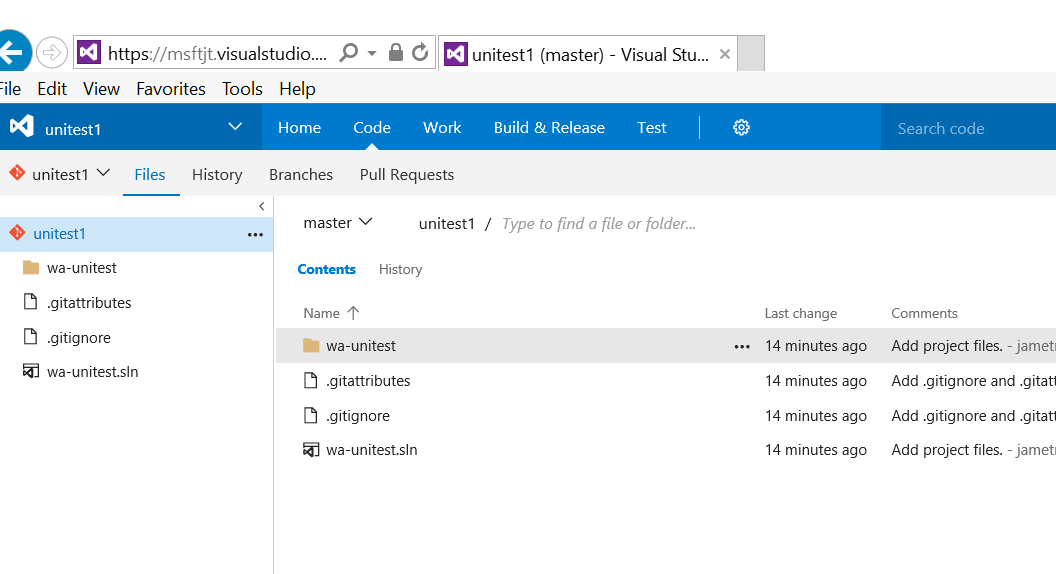
* 1. This hands-on lab includes the following exercises:
  2. Create a new web application
  3. Add unit tests and code coverage
  4. Add to VSTS build service and monitor results
  5. Estimated time to complete this exersize: **15 minutes**.

Exercise 1: Create a web application

#### Task 1: Create a web application

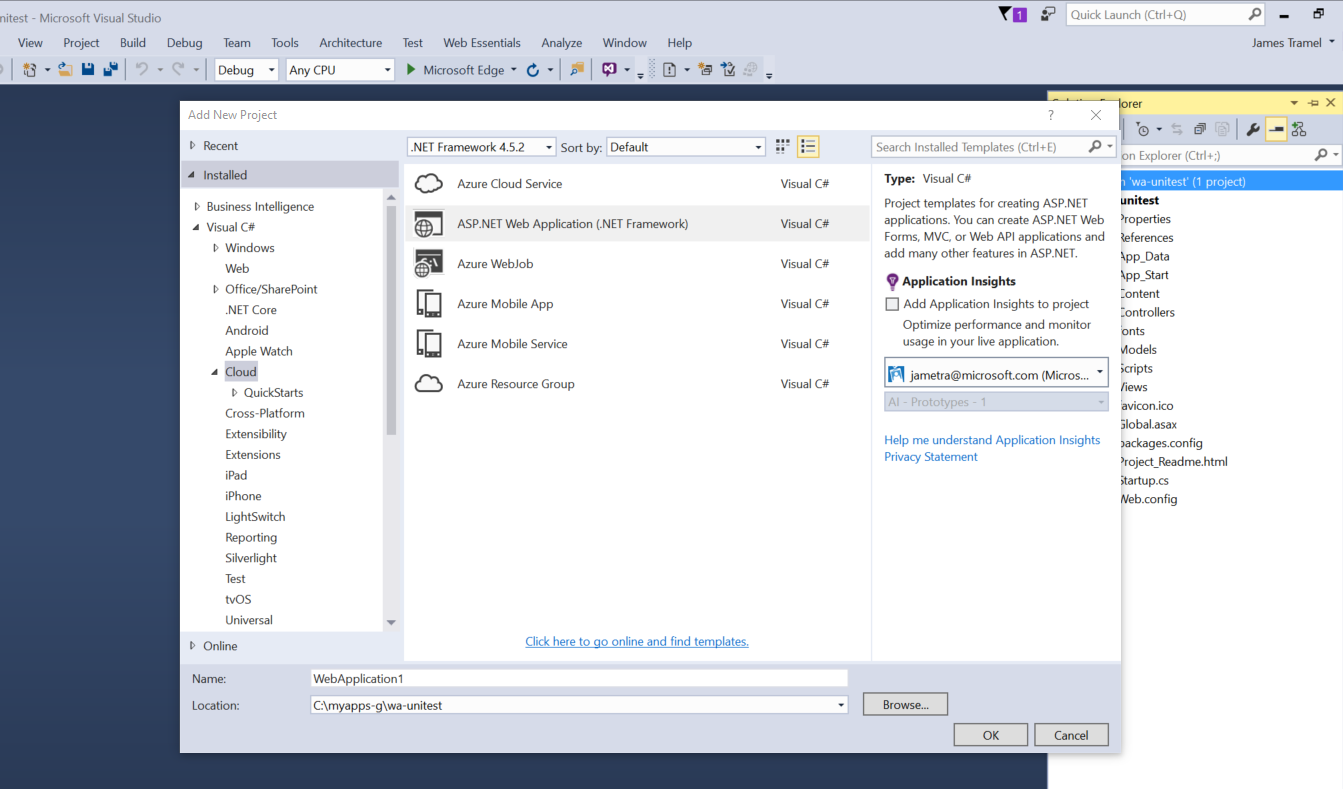
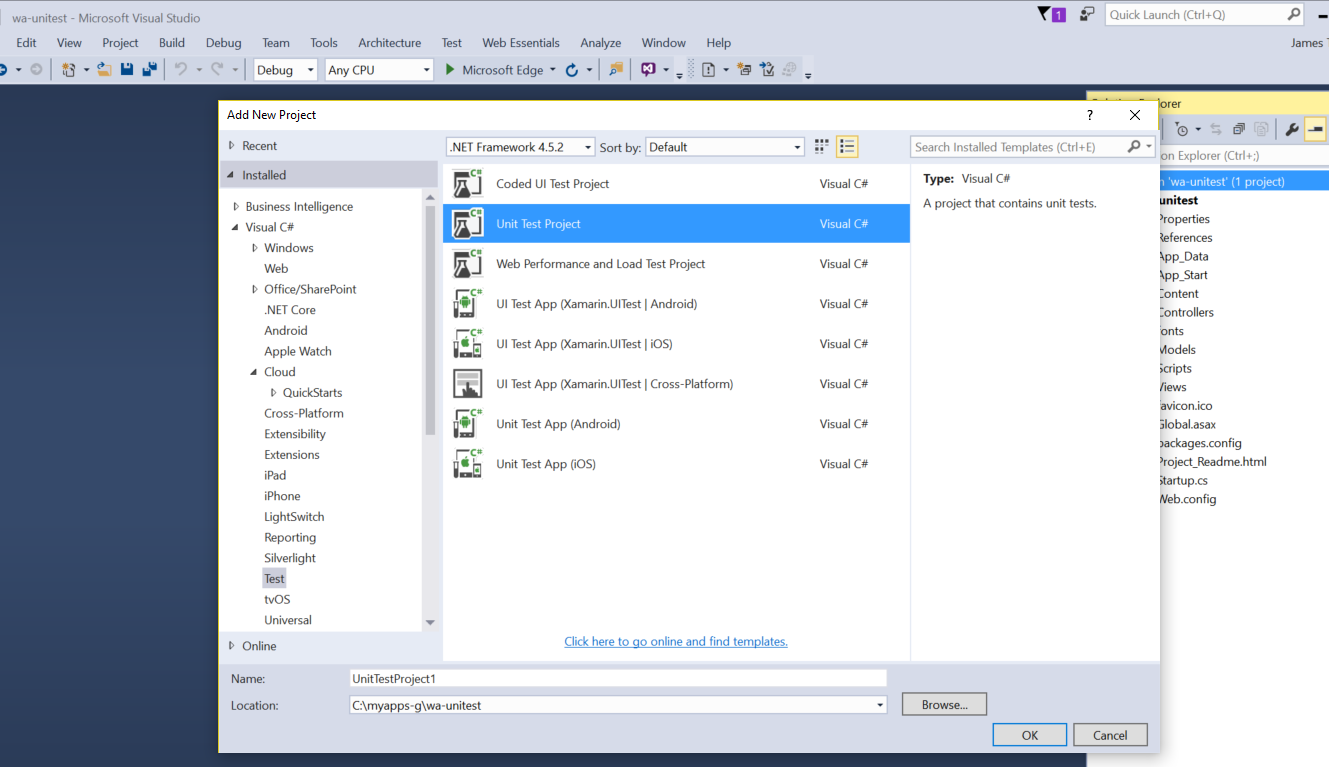
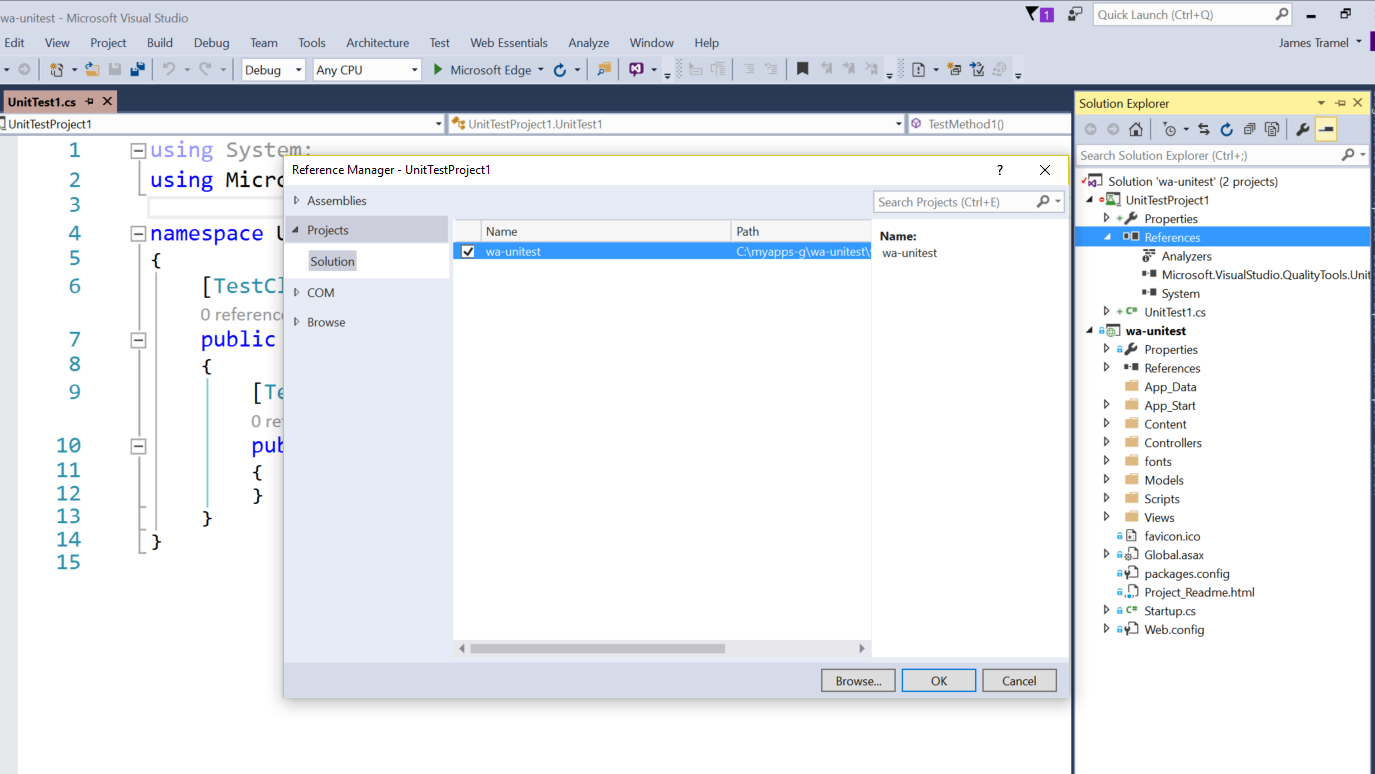
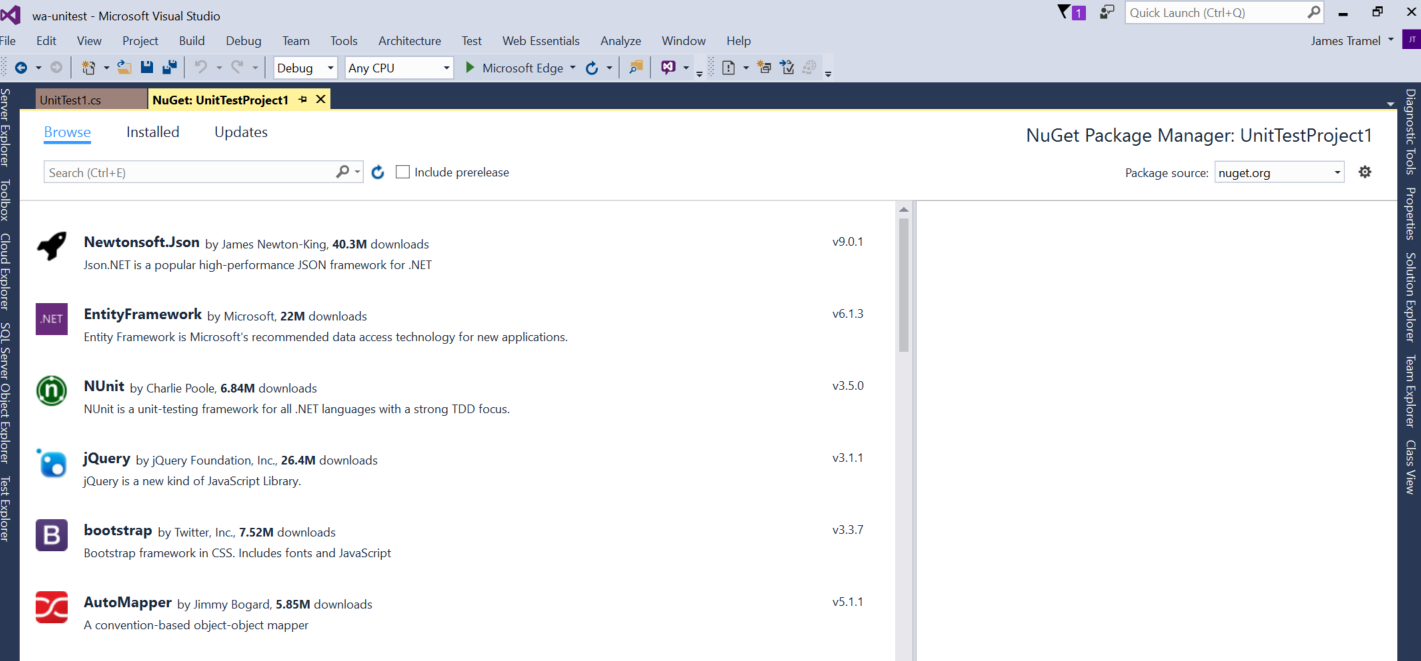
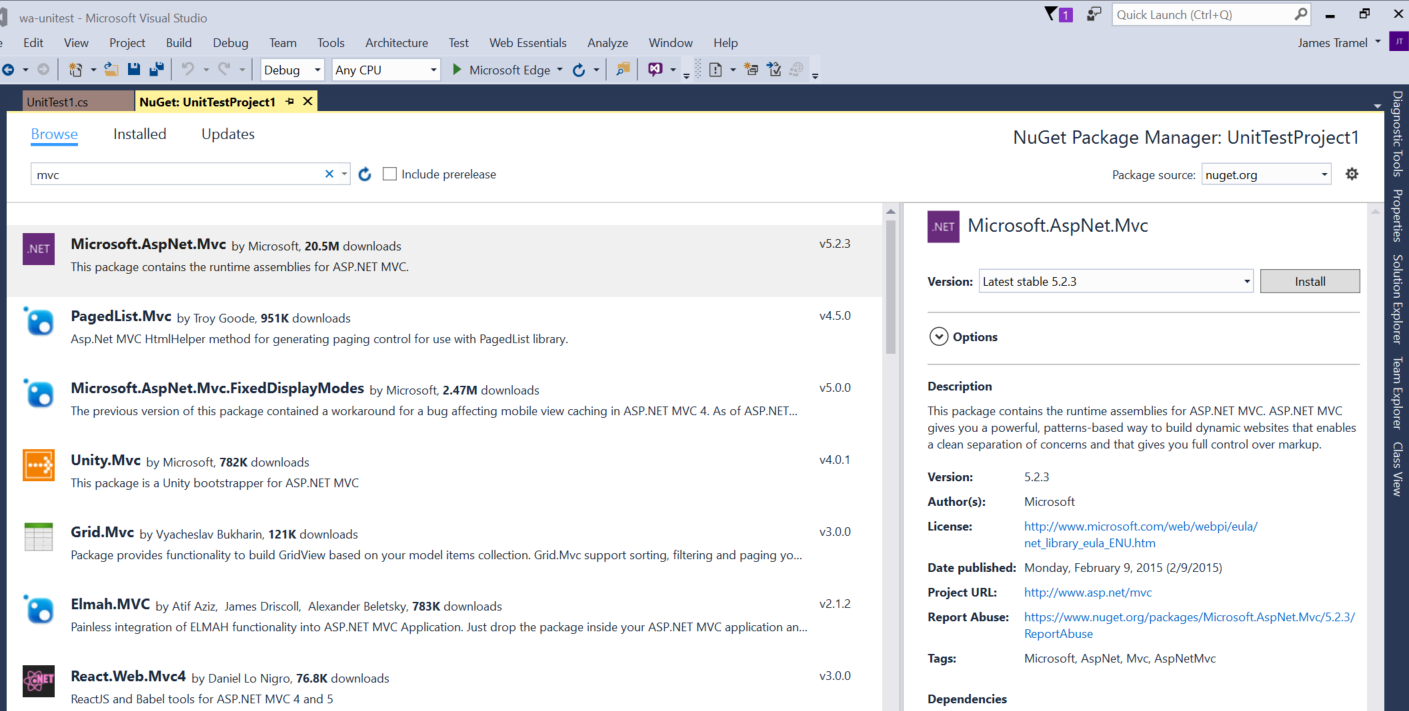
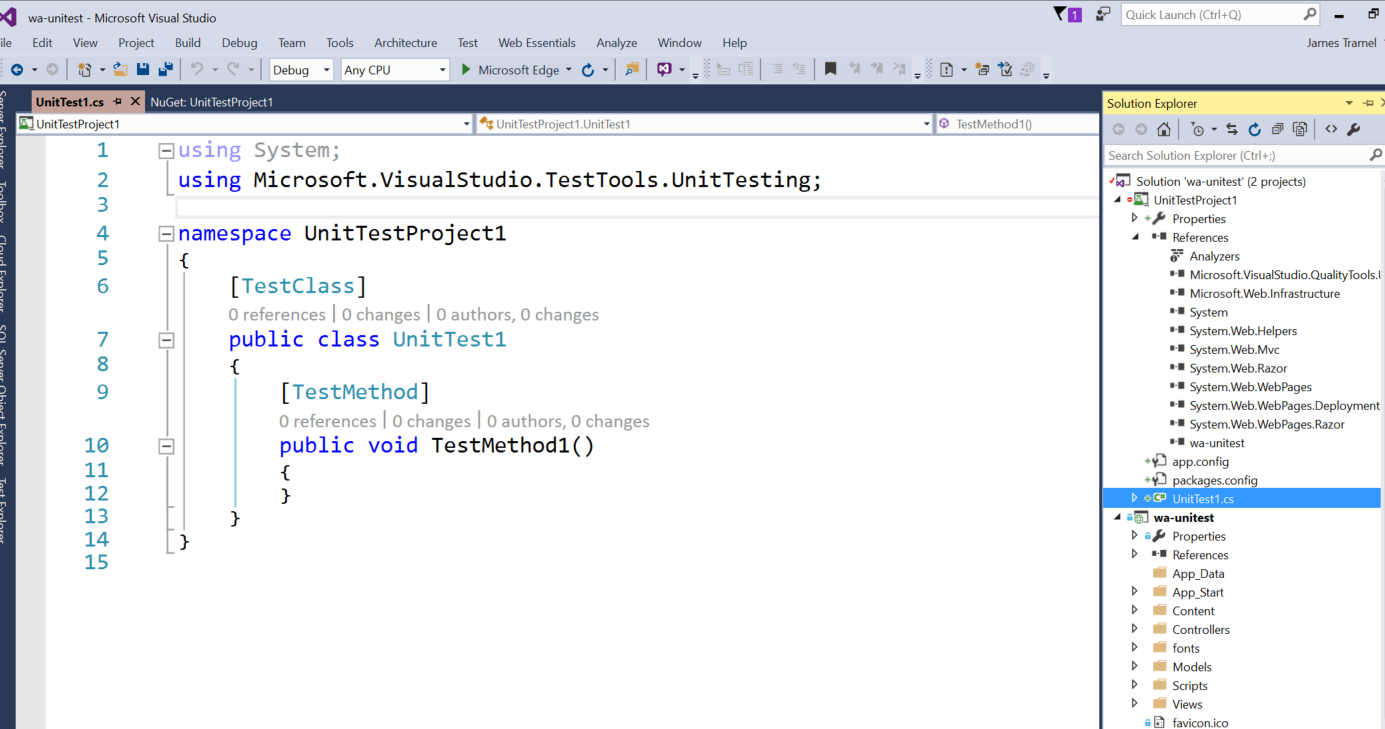
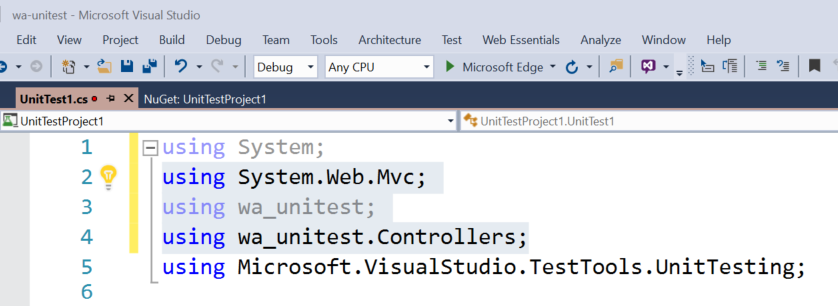
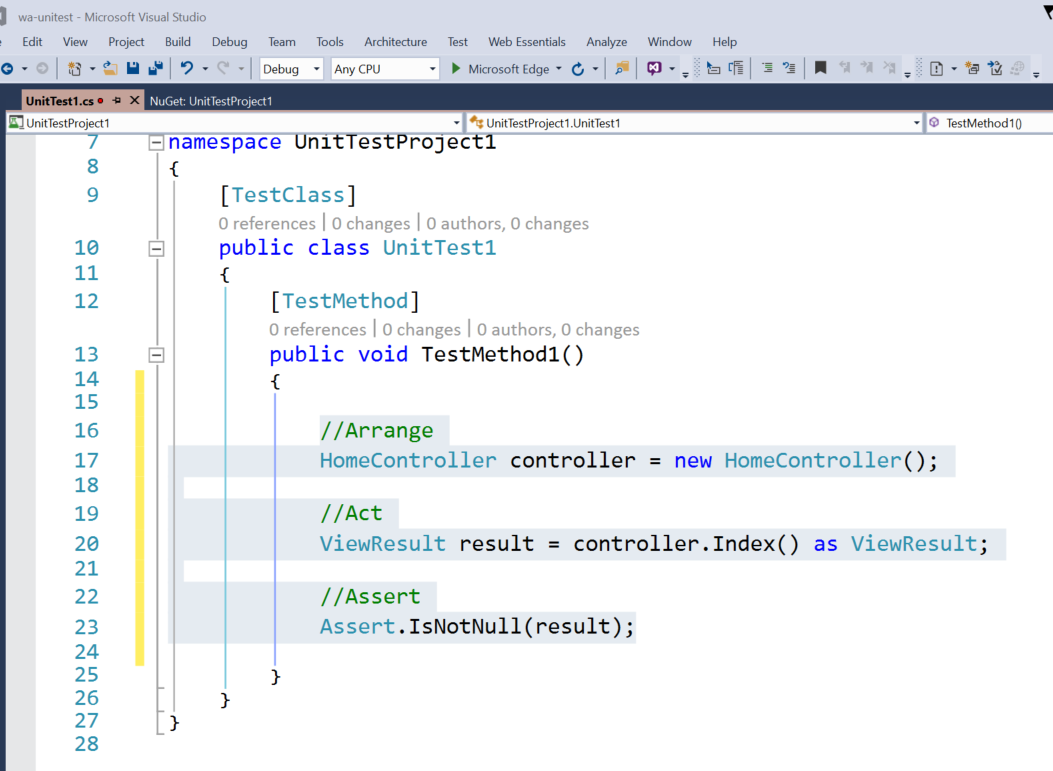
1. Create a new web application (using asp.net web application and choose MVC)



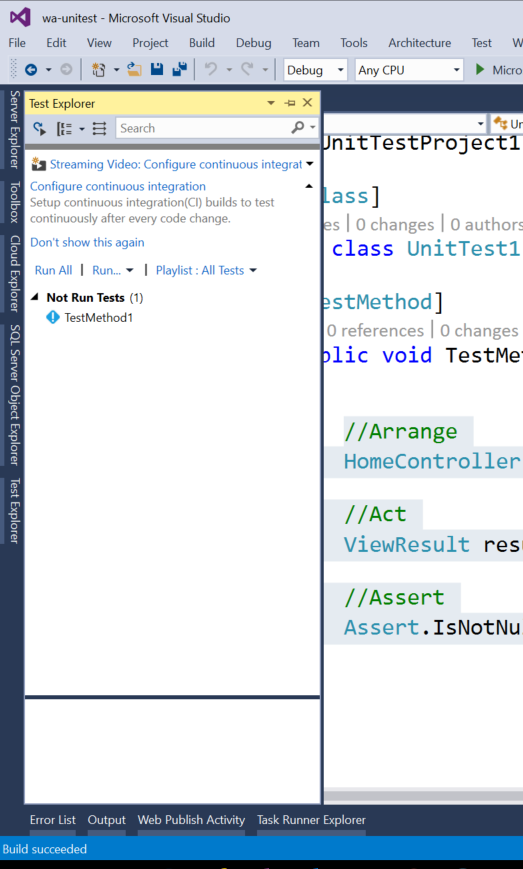
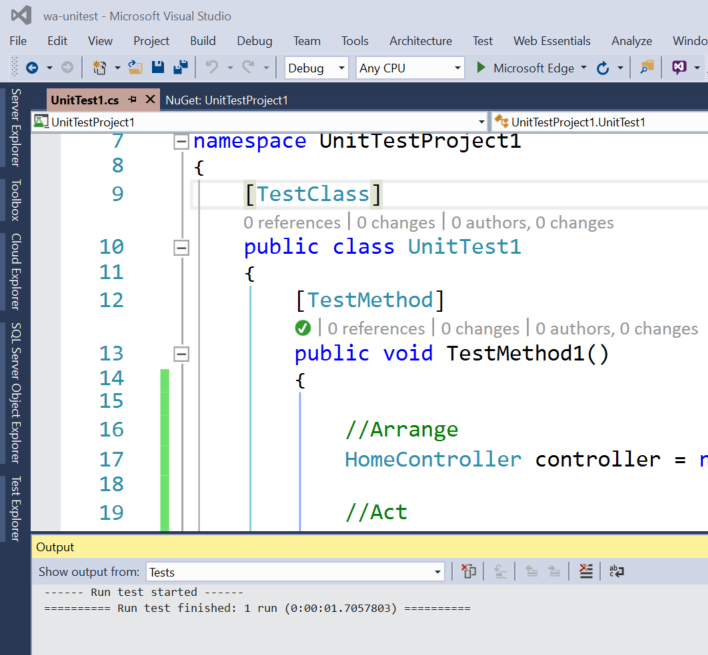
1. Add to source control 
2. Build and run locally
3. Check into VSTS

Exercise 2: Add a unit test project

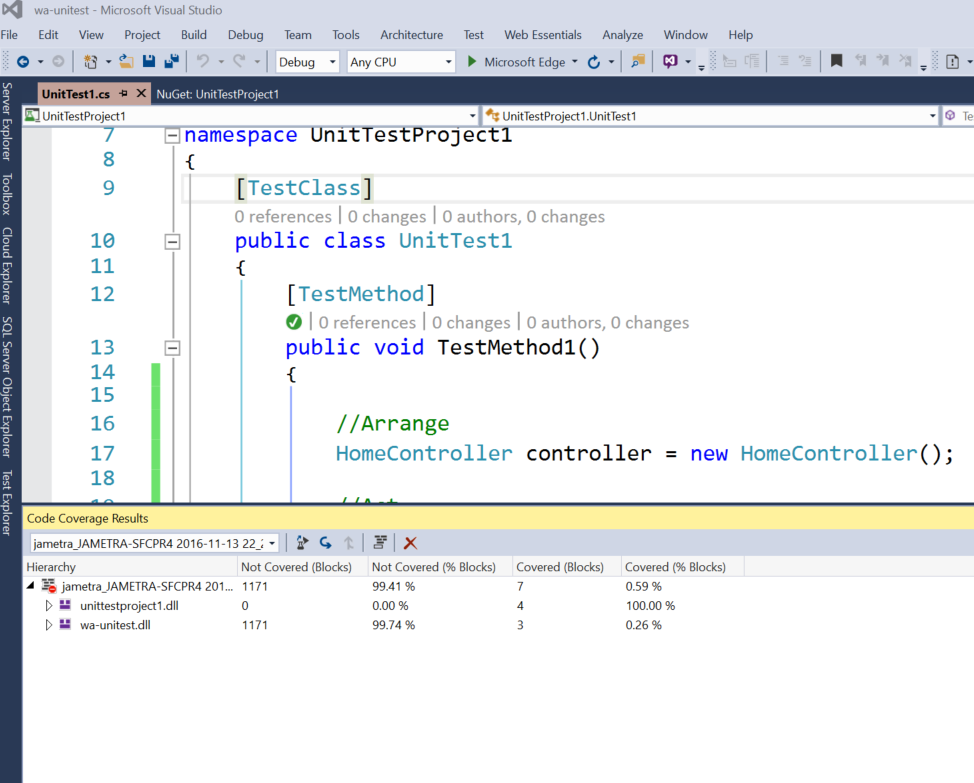
#### Task 1: Create a unit test project

* 1. Right click on solution, Add, New Project 
  2. Click on Test, Unit Test project 
  3. In unit test project, right click on add reference and add solution you’re working on to the unit test project 
  4. Click refernce again, and select manage nuget packages 
  5. Search for MVC and add instal Microsoft.ASPNet.MVC (click ok and accept license)
  6. Open Unit Test project file 
  7. Add using statements:  
     using System.Web.Mvc;  
     using [your MVC project name];  
     using [your MVC project name.Controllers]; 
  8. Add the following statements to your test method  
     //Arrange  
      HomeController controller = new HomeController();  
     //Act  
      ViewResult result = controller.Index() as ViewResult;  
      //Assert  
      Assert.IsNotNull(result); 
  9. Build your project

#### Task 2: Run your unit test

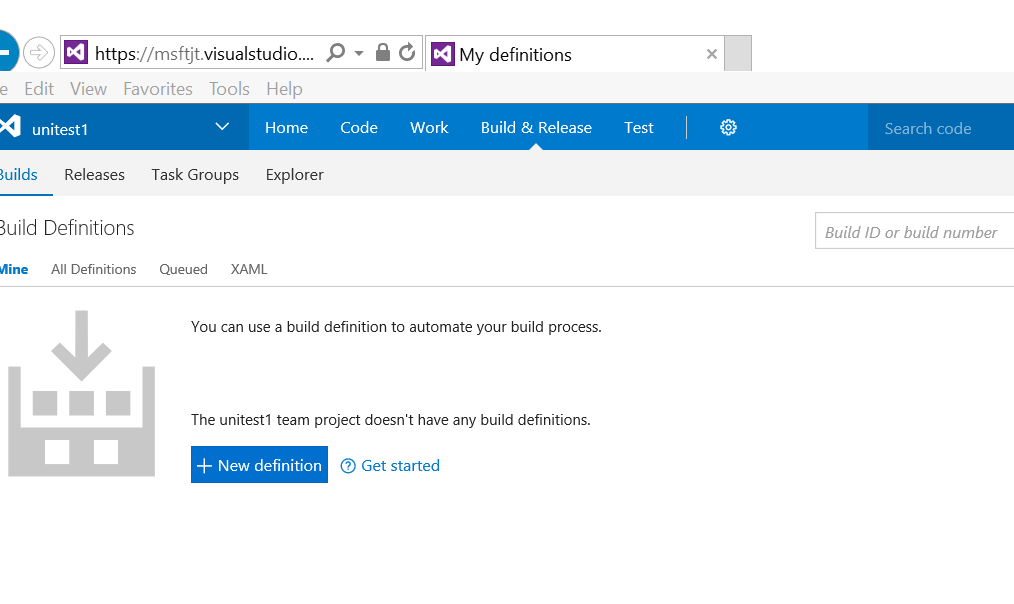
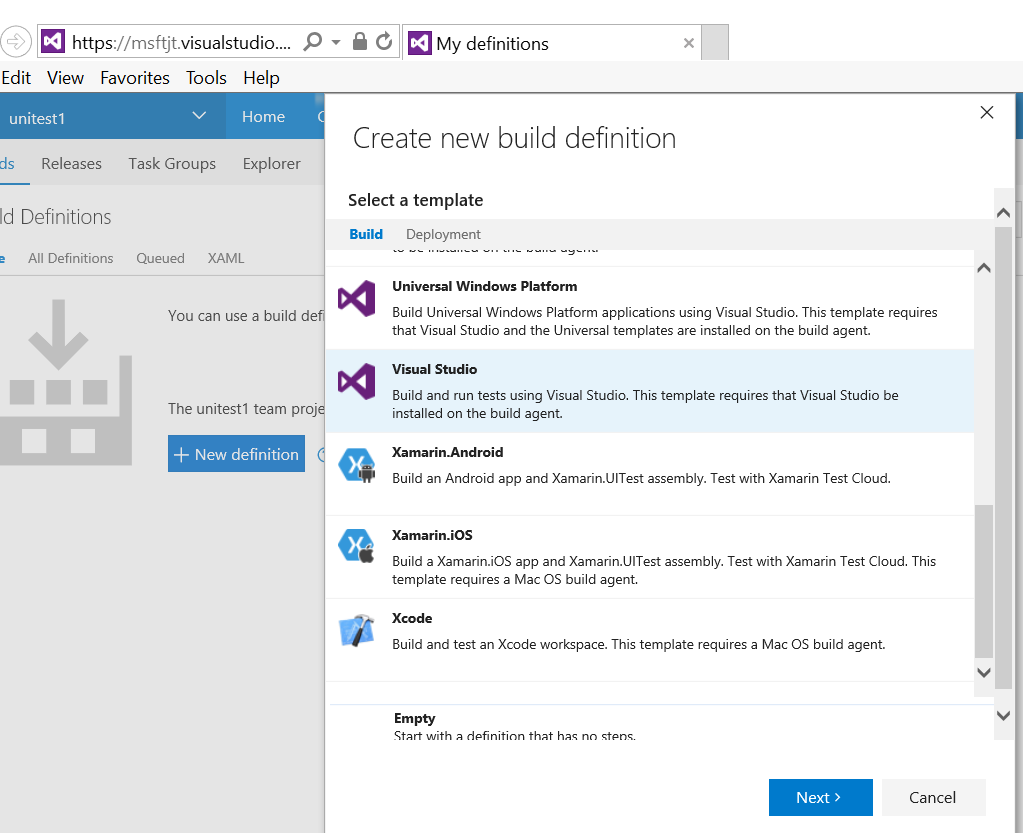
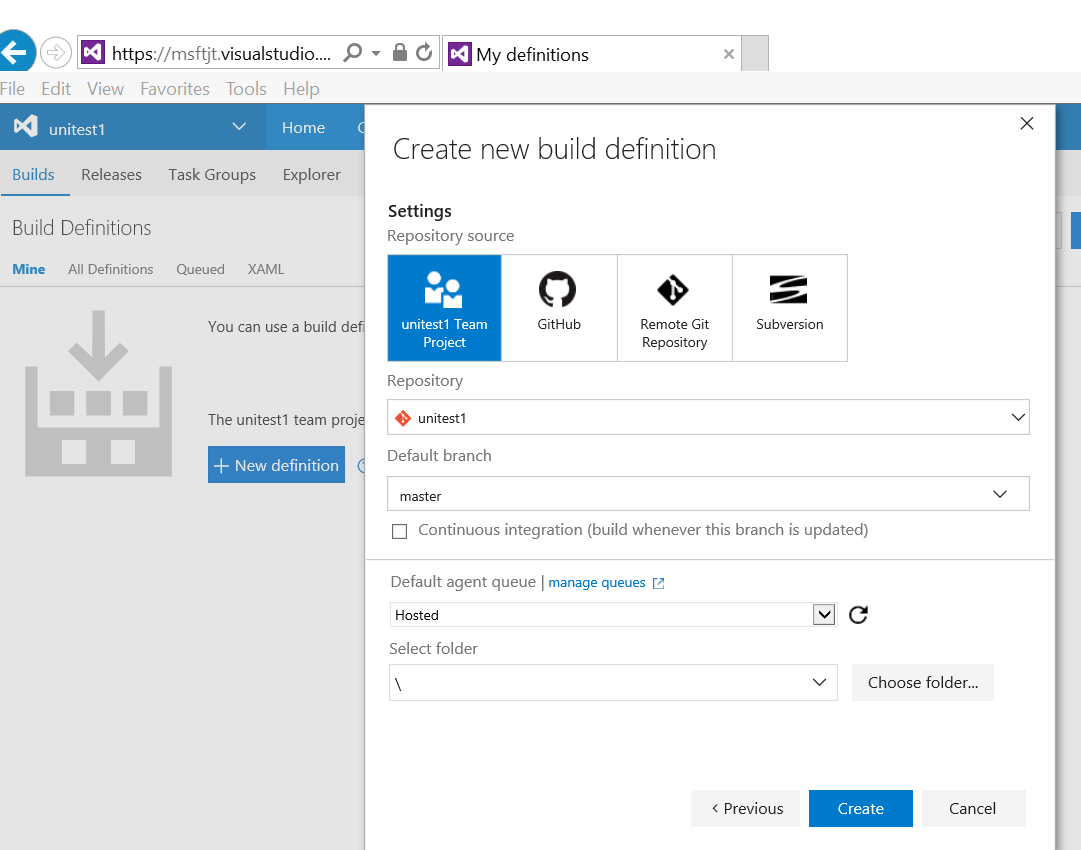
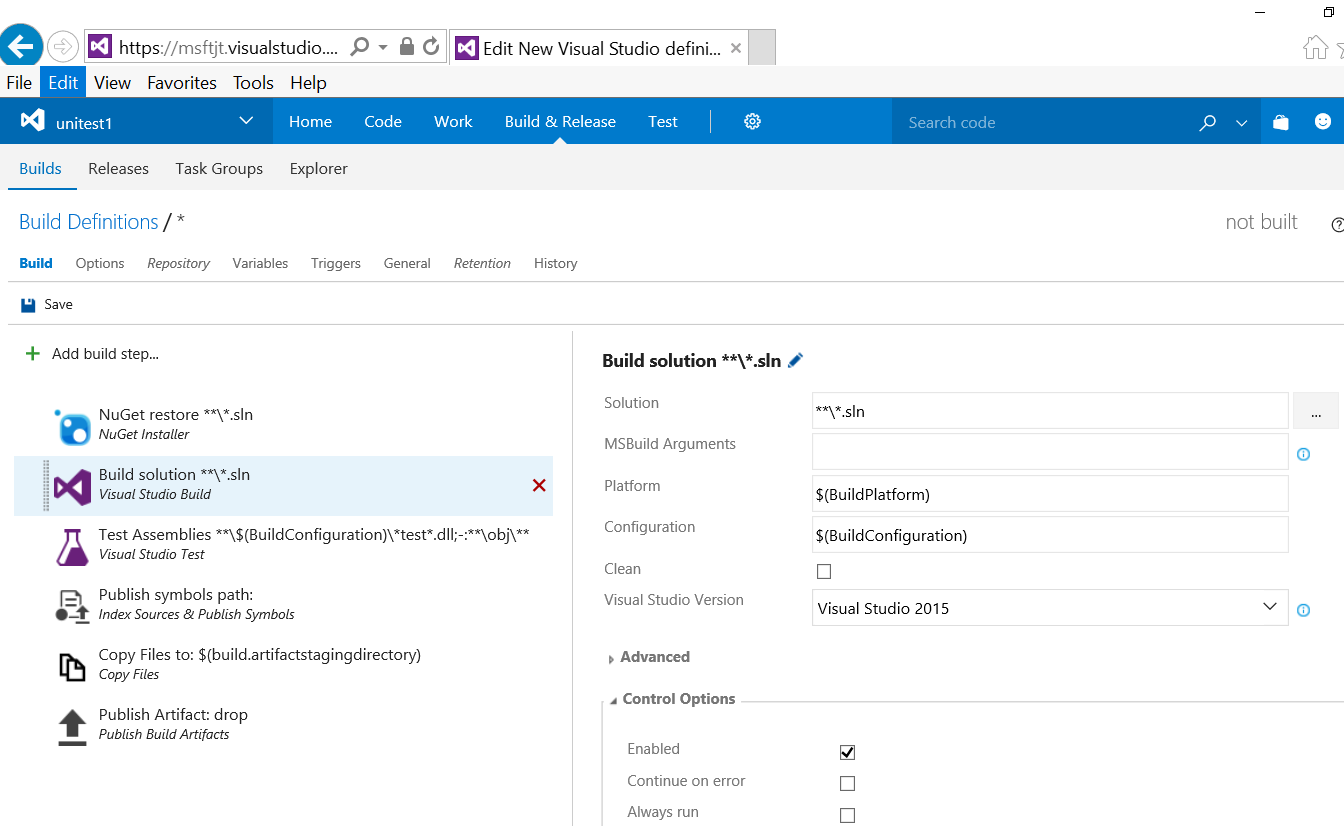
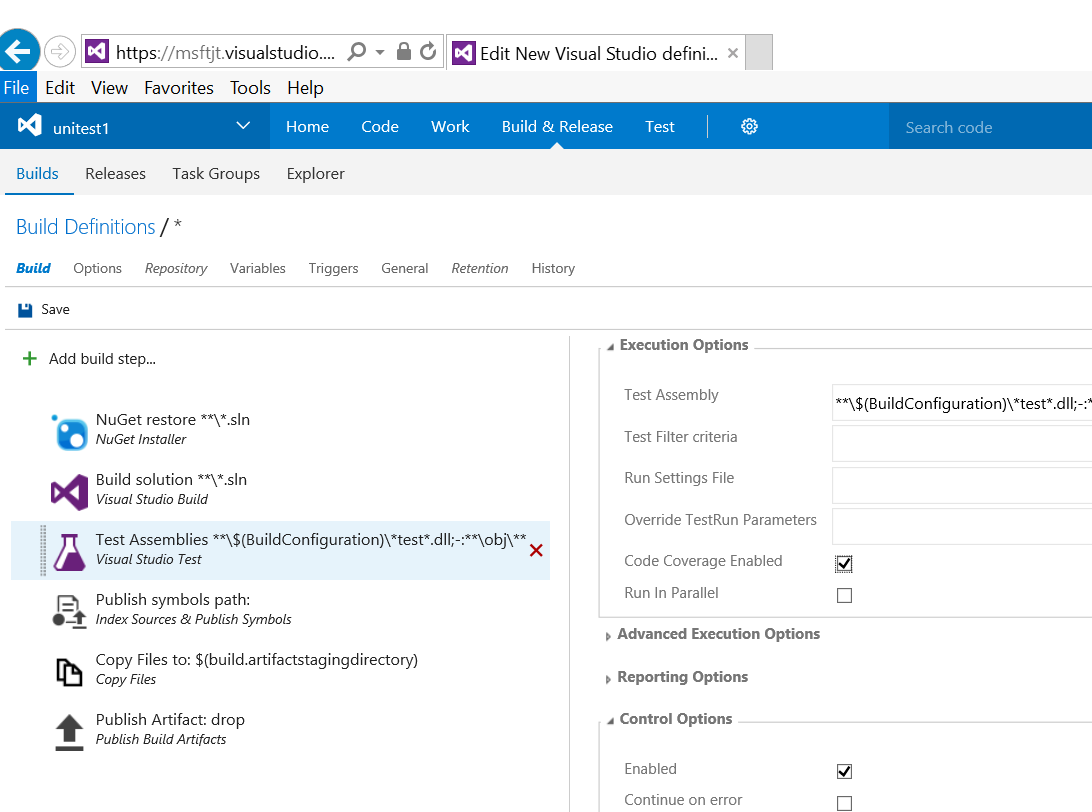
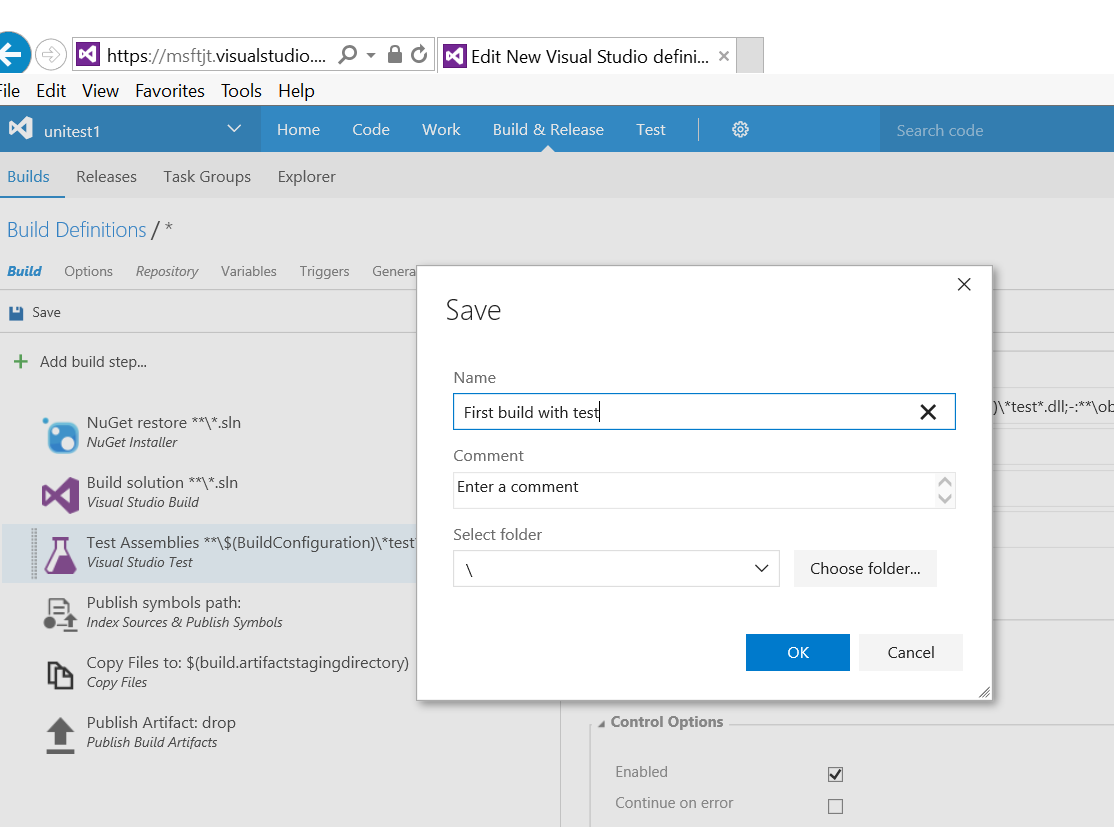
* 1. Open Test explorer   
      
  2. Run your test by right click your test method and clicking run selected test. Verify outcome. 

#### Task 3: Check Code Coverage

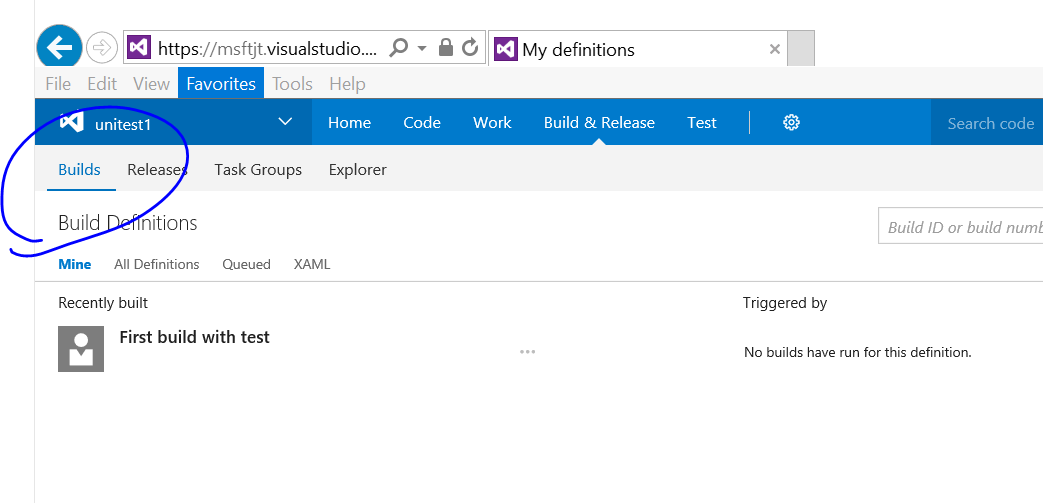
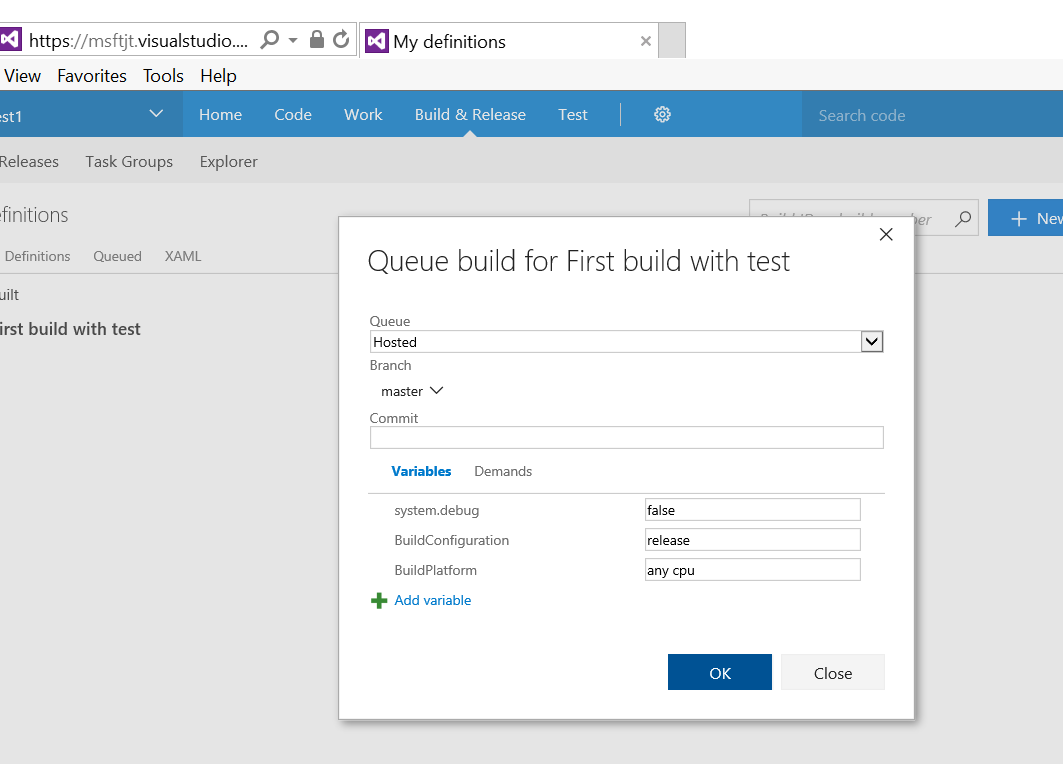
* 1. Test menu, Analyze Code Coverage, All tests
  2. Verify results  
     
  3. Don’t forget to check in all changes

Exercise 3: Create a new build

#### Task 1: Create a new build template

1. Navigate in VSTS to Build&Release  
   
2. Click new definition and choose Visual Studio 
3. Keep the defaults or make changes that make sense to your project, and click create 
4. View the build steps 
5. Click on test assemblies and select code coverage enabled, then click save near the top under build definitions 
6. Name your build something meaningful 

#### Task 2: Run your build

1. Click on builds  .
2. Click on the … (ellipses) and choose queue new build
3. Choose OK 
4. Verify your results 