

# Azure Developer Immersion

## VSTS Build

No matter what tools you use. In whatever language you prefer. Visual Studio Team Services Build builds your app, your way, for your platforms. Just open your web browser to tell it how you want it done. You can build for Windows, iOS, Android, Java (Ant, Maven, or Gradle), or Linux using the same domain-specific languages (DSL) you use every day on your dev machine. The build services even build Xamarin apps for both iOS and Android and run tests on Xamarin's Test Cloud as part of the build.

In this walkthrough, you will create a “team build” in Visual Studio Team Services. You will start with a manual build, then configure one for continuous integration.

There are 2 exercises in this walkthrough:

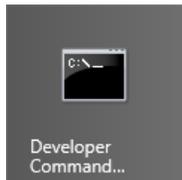
1. Create a Manual Build
2. Create a CI Build

**Expected duration: 30 minutes**

## Exercise 1: Create a Manual Build

In this exercise, you will learn how to create a manual Team Build that runs in Visual Studio Team Services. This makes it quick and easy to start building your code, either manually or in an automated fashion, without having to worry about any build server configuration.

1. Open the Developer Command Prompt for VS 2015.



2. Change to the directory **Rg.ClientApp** project's directory typing the following and pressing **ENTER**:

```
cd C:\VSTS\Repos\rGroup\Before\Rg.ClientApp
```

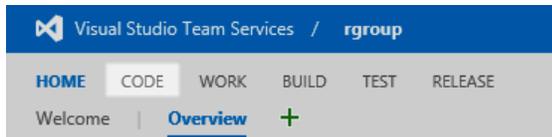
3. You're now going to “add” the **Rg.ClientApp\_TemporaryKey.pfx** file so you can commit and push it to VSTS.

```
git add Rg.ClientApp_TemporaryKey.pfx -f
```

Typically you don't commit PFX files to your repo which is why by default it is ignored by Visual Studio and Git (this is controlled via the .gitignore file). However, you need the PFX file to complete the build process for the Windows Store app. As this is a temporary key that can't be used for publishing to the store, it's not critical.

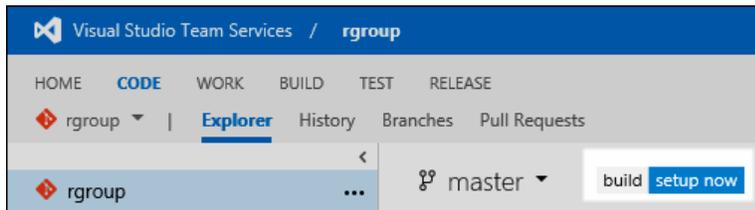
4. Type **exit** and press **ENTER** to close the command prompt.

5. Return to Visual Studio.
6. In Team Explorer, click the **Changes** button.
7. For a commit message, enter **Add Windows Store app certificate to repo**.
8. Click the **Commit and Push** button.
9. On the **Home** screen in **Team Explorer**, click the **Web Portal** link to open your VSTS Team Project's Home page.
10. Click the **CODE** hub link.



In the CODE hub, you can easily examine your repo's history, manage branches, and pull requests. In addition, it provides an easy way to get started creating a team build.

11. To the right of the master branch, click the **setup now** button.

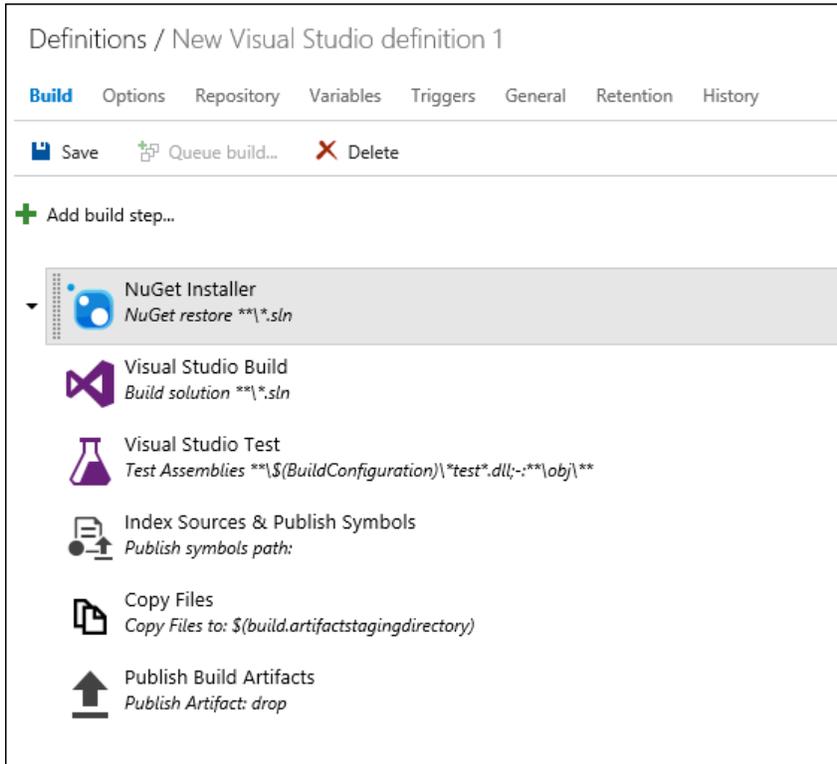


12. In the **Create new build definition** dialog, select the Visual Studio template and click **Next**.



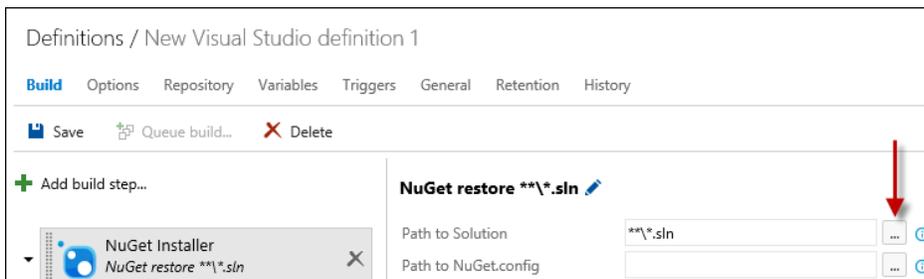
13. On the next page, you'll see you can configure which repo and branch to use if you have more than one of either. Accept the defaults and click **Create**.

Visual Studio Team Services opens a new browser tab in the BUILD hub where you can continue editing the Build Definition.

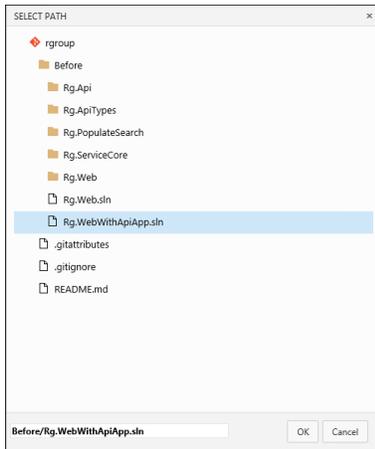


A Team Build is composed of multiple steps or tasks that are executed in sequence. The default build definition template for Visual Studio solutions does a number of steps including running tests and saving your build output to VSTS where you can download the compiled version. For your purposes, the default template with “just work”, but you’re going to tweak it a bit to be more efficient.

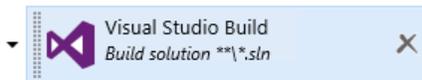
14. The first step should already be selected—NuGet Installer. To the right, click the ... button at far right of the **Path to Solution** option.



15. In the SELECT PATH dialog, select the **Before/Rg.WebWithApiApp.sln** file and click **OK**.

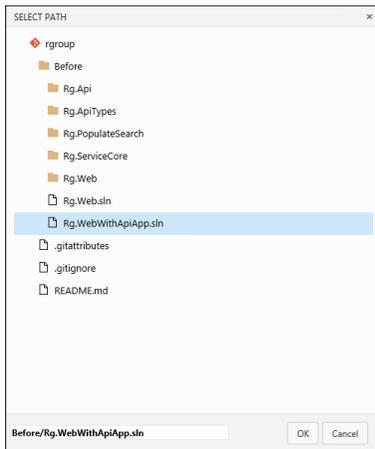


16. Select the Visual Studio Build step.

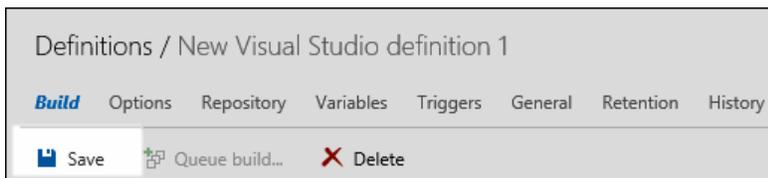


17. To the right, click the ... button at far right of the **Solution** option.

18. In the SELECT PATH dialog, select the **Before/Rg.WebWithApiApp.sln** file and click **OK**.

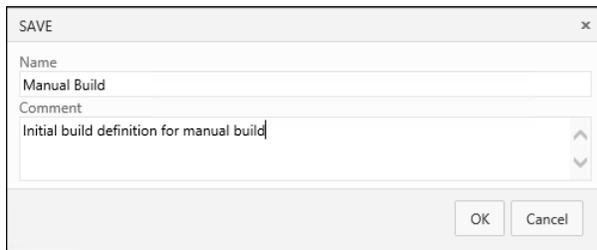


19. Click the **Save** button to save your build definition.



20. In the SAVE dialog type **Manual Build** for the **Name**.

21. In the **Comment** field, type **Initial build definition for manual build**. A great feature of Team Build is that VSTS versions your build definitions so you can see when and why a change was made and once you have two or more, you can diff the JSON-based definition.



SAVE

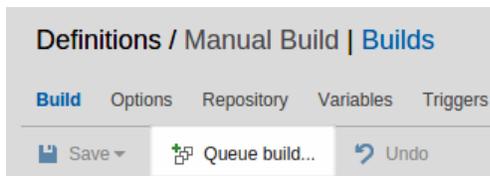
Name  
Manual Build

Comment  
Initial build definition for manual build

OK Cancel

22. Click **OK**.

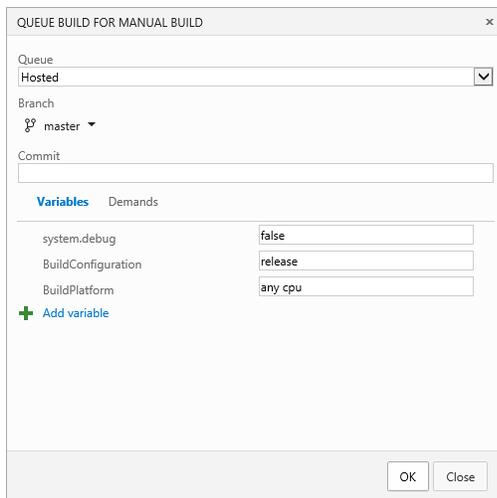
23. Click the **Queue build** button to open the **QUEUE BUILD FOR MANUAL BUILD** dialog.



Definitions / Manual Build | Builds

Build Options Repository Variables Triggers

Save Queue build... Undo



QUEUE BUILD FOR MANUAL BUILD

Queue  
Hosted

Branch  
master

Commit

Variables Demands

system.debug	false
BuildConfiguration	release
BuildPlatform	any cpu

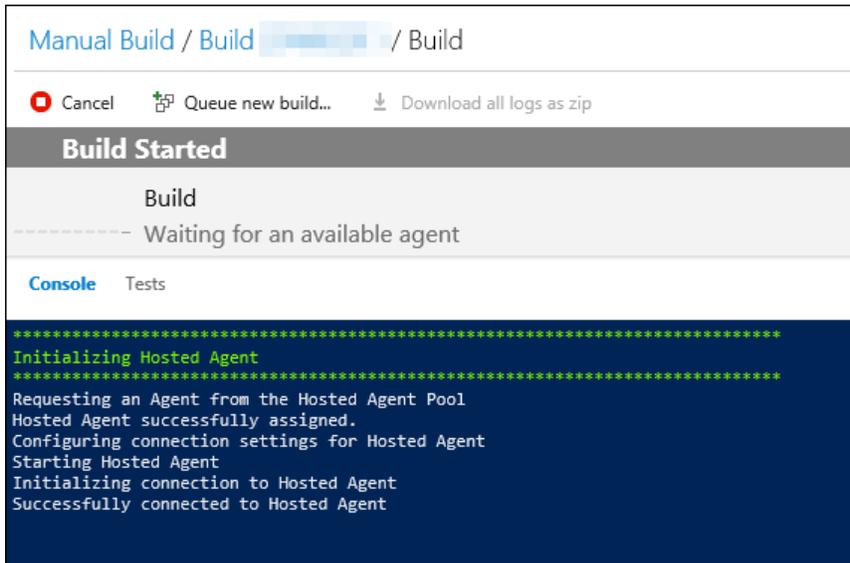
+ Add variable

OK Close

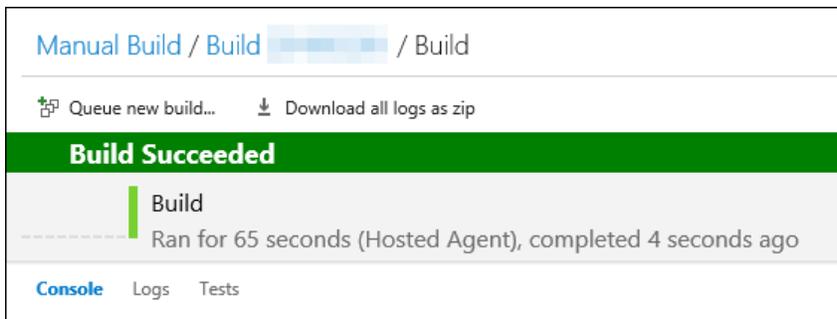
The **QUEUE BUILD FOR MANUAL BUILD** dialog provides a number of options to run your team build. These are all read from the build definition. You can change these on a build by build basis or just accept the default settings. For this build, you'll be using the VSTS Hosted Agent. Your VSTS account comes with 240 build minutes a month. If you need more than that you rent a hosted agent or set up your own agent in a VM in Azure or on premises.

24. Click **OK** to queue the build.

VSTS will queue the build and keep you posted on its progress.



25. Monitor the build until it completes. It can take anywhere from 45 seconds to a few minutes depending upon load on the hosted build service.



26. Once it's done, you examine the build details by click the "Build #" link where # is the build run number.

The build report will tell you lots of details including related commits, associated work items, and test results (if any).

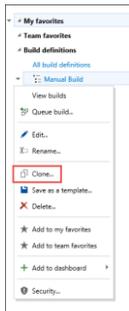
The screenshot displays a build report interface. At the top, it shows 'Manual Build / Build [redacted]' with options to 'Queue new build...' and 'Download all logs as zip'. A green banner indicates 'Build Succeeded'. Below this, a progress bar shows the build 'Ran for 77 seconds (Hosted), completed 109 seconds ago'. The 'Summary' tab is active, showing 'Build details' on the left and 'Test Results', 'Code Coverage', and 'Tags' on the right. The 'Build details' section lists: Definition (Manual Build), Source branch (refs/heads/master), Source version (Commit 03c47413), Requested by (Azurepass Admin), Queued (2016 2:35:24 PM), Started (2016 2:35:32 PM), and Finished (2016 2:36:48 PM). The 'Test Results' section states 'No test runs are available for this build.' The 'Code Coverage' section states 'No build code coverage data available.' The 'Tags' section has an 'Add...' button. The 'Issues' section shows a warning: 'No test assemblies found matching the pattern: \*\*\release\\*test\*.dll;-\*\obj\*\*'. The 'Associated work items' section lists two tasks: 'Task 3 Complete the 01-Lift and Shift Walkthrough' and 'Task 4 Complete the 02-API App Walkthrough', both with a 'Current state is Done' and assigned to 'Azurepass Admin'. The 'Associated changes' section lists two commits: 'Commit 03c4741' (Add Windows Store app certificate) and 'Commit 70a0afb' (Completed configuring API app and test client). At the bottom, it shows 'Related Work Items: #4'.

27. At this point a build will only run if you ask queue it manually. Continue to the next exercise where you will add support for continuous integration.

## Exercise 2: Create a CI Build

Now let's take the manual build we created and create a continuous integration or "CI" build.

28. Click the BUILD hub link.
29. Select the **Manual Build**.
30. Select the arrow next to **Manual Build** and select **Clone...** from the menu options.

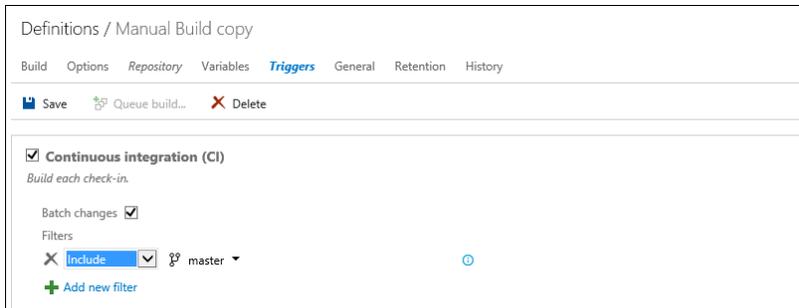


This creates a copy of the build definition called **Manual Build copy**. It also opens to the new build definition.

31. Select the **Triggers** tab.



32. On the **Triggers** tab, check the box next to **Continuous Integration (CI)**. Take the defaults.



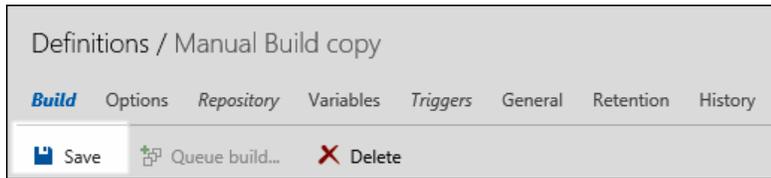
33. Click the **Build** tab. You're going to remove some unnecessary steps to make the CI build run faster.

34. Remove the following steps by clicking the **X** button:

- Index Sources & Publish Symbols
- Copy Files
- Publish Build Artifacts.

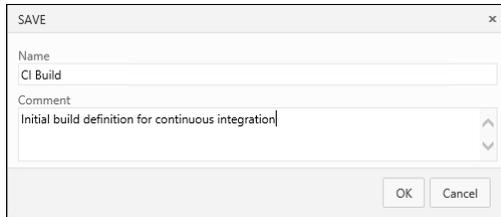
Generally, you want your CI builds to run as fast as possible to give you and your team members immediate feedback if you missed any files or broken something. Since the current sample doesn't have any tests, you could also remove the Visual Studio Test task. But it's better to leave it as a reminder that you should add some tests later.

35. Click the **Save** button to save your build definition.



36. In the SAVE dialog type **CI Build** for the **Name**

37. In the **Comment** field, enter **Initial build definition for continuous integration.**

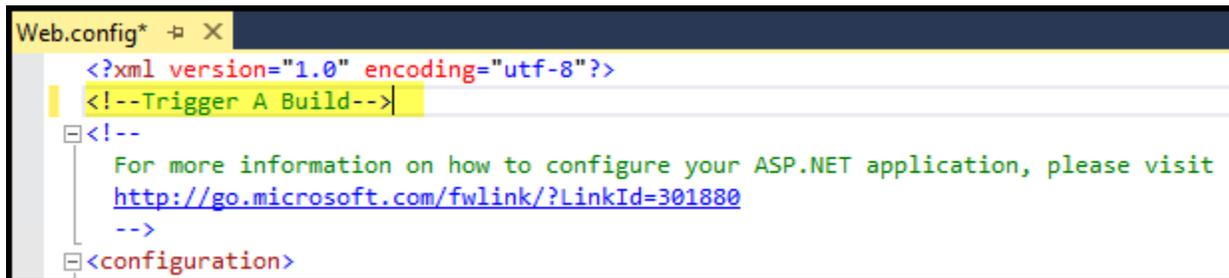


38. Click **OK**.

39. Now you need to make a change in your solution, commit & push it, and see if the build runs.  
Return to **Visual Studio**.

40. In the **Rg.Web** project, open the **Web.Config** file.

41. At the top of the file, add the following comment.



42. Save your changes.

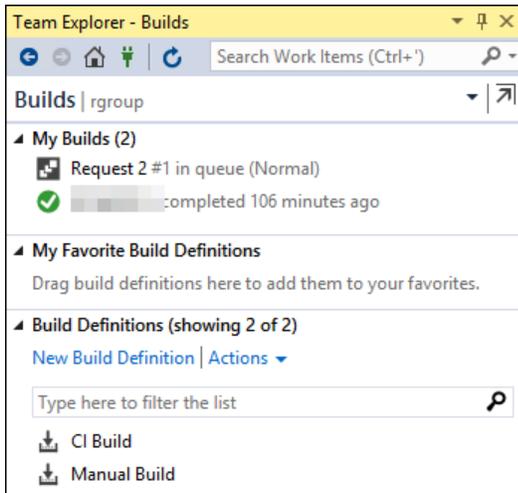
43. In **Team Explorer**, click the **Changes** button.

44. Enter the following commit message: **Make a quick change to trigger a CI Build.**

45. Click the **Commit and Push** button. If prompted to save changes click **Yes**.

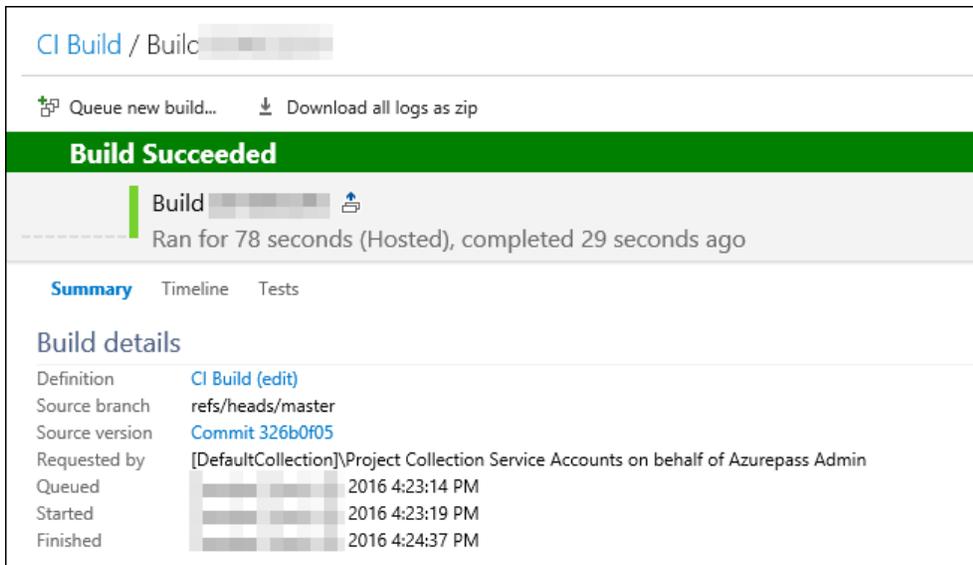
46. Click the **Home** button in Team Explorer.

47. Click the **Builds** button on the **Home** screen.

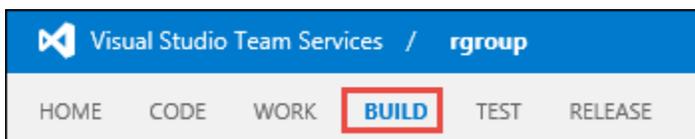


48. You'll see a build has been initiated. Double click the build to open it in the VSTS web portal and watch it run.

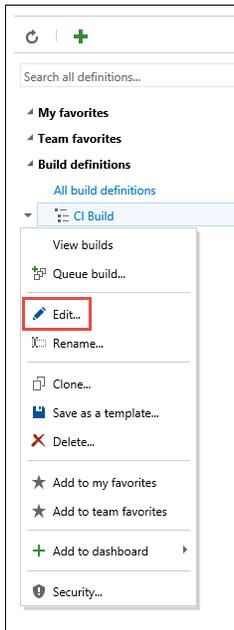
49. As before, once the build finishes running, click the build name to view the report.



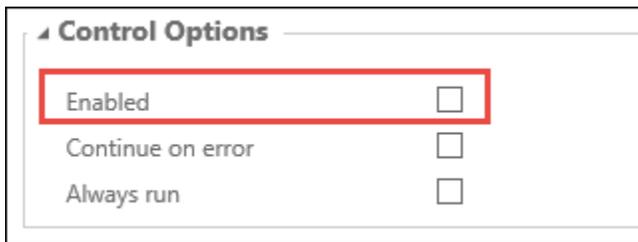
50. For now, you can disable the CI Build, so it doesn't use all you build minutes up during the lab. Click the **Build** tab at the top of Web Access.



51. Select the arrow next to **CI Build** and select **Edit** from the menu.



52. Uncheck the **Enabled** checkbox under **Control Options**.



53. Click the **Save** button.

54. In the **Comment** field, enter **Disabled CI**.

55. Click **OK**.

56. You're now ready to continue working on the next walkthrough.