
FIT5032 - Internet Applications Development

WEEK 12 - CONFIGURING FOR CONTINUOUS DEPLOYMENT (AZURE & GITHUB)

Last updated: 14th Oct 2018

Author: Jian Liew

Introduction

There is no need to complete this tutorial. It functions as a supplementary material to showcase how to use a version control system like Git using Github to achieve continuous delivery in Azure.

Even though, it is not compulsory to complete this it is highly suggested to at least understand what a version control tool is and continuous deployment.

The Monash lab computers does not come with Git and thus it is not suitable to complete this lab. It is highly recommended to use your own computer to complete these supplementary materials. Thus, it is optional to complete this lab.

It is also highly recommended to obtain <https://education.github.com/pack> if you are planning to have a career as a developer in the future.

Most modern applications are developed using the process known as Continuous Integration and Continuous Delivery. This supplementary material aims to showcase how it is done at a very basic level. If you wish to further learn about this, the unit FIT5171 aims to teach you more.

Software Requirements (**Do not proceed if you do not have the required software & accounts**)

- Github Account
- Azure account
- Understanding of basic continuous deployment process using Azure + Github

Objectives

Estimated Time To Complete ~ 1 hour (Depending on various factors)

Upon the completion of this tutorial, you will gain a basic understanding of

- Learning how to use git at a basic level
- Understand a basic continuous deployment process

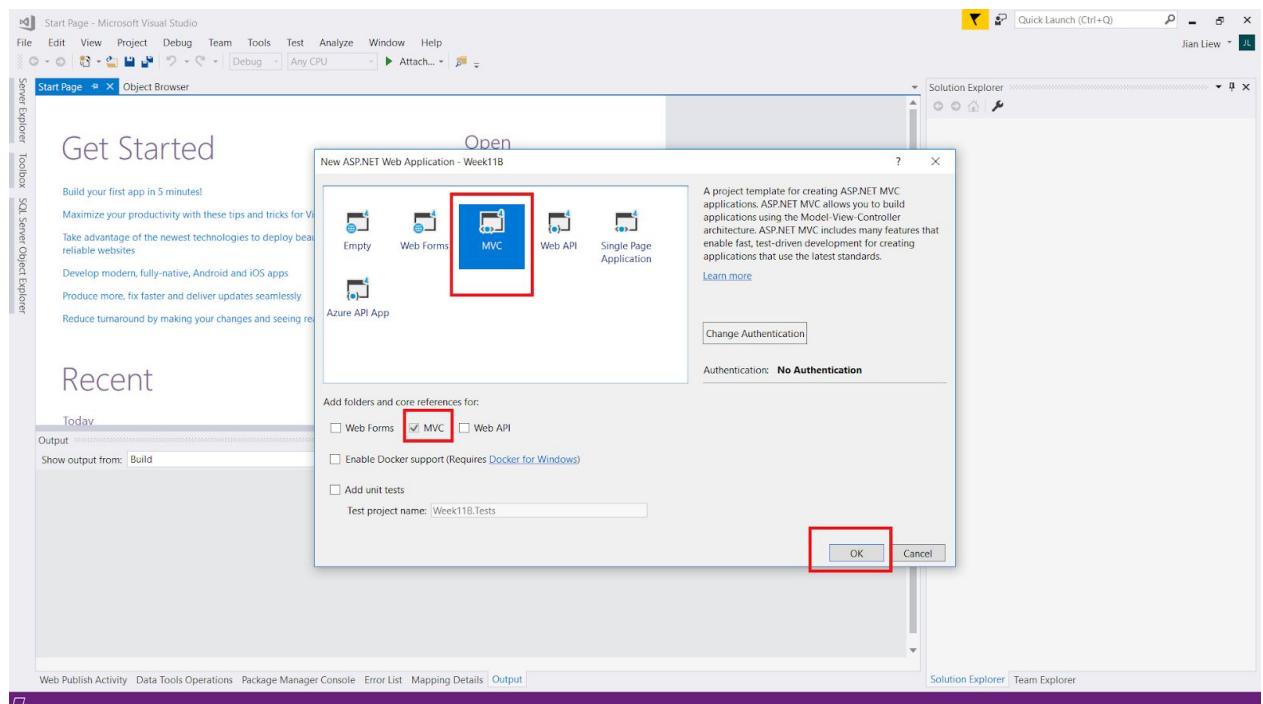
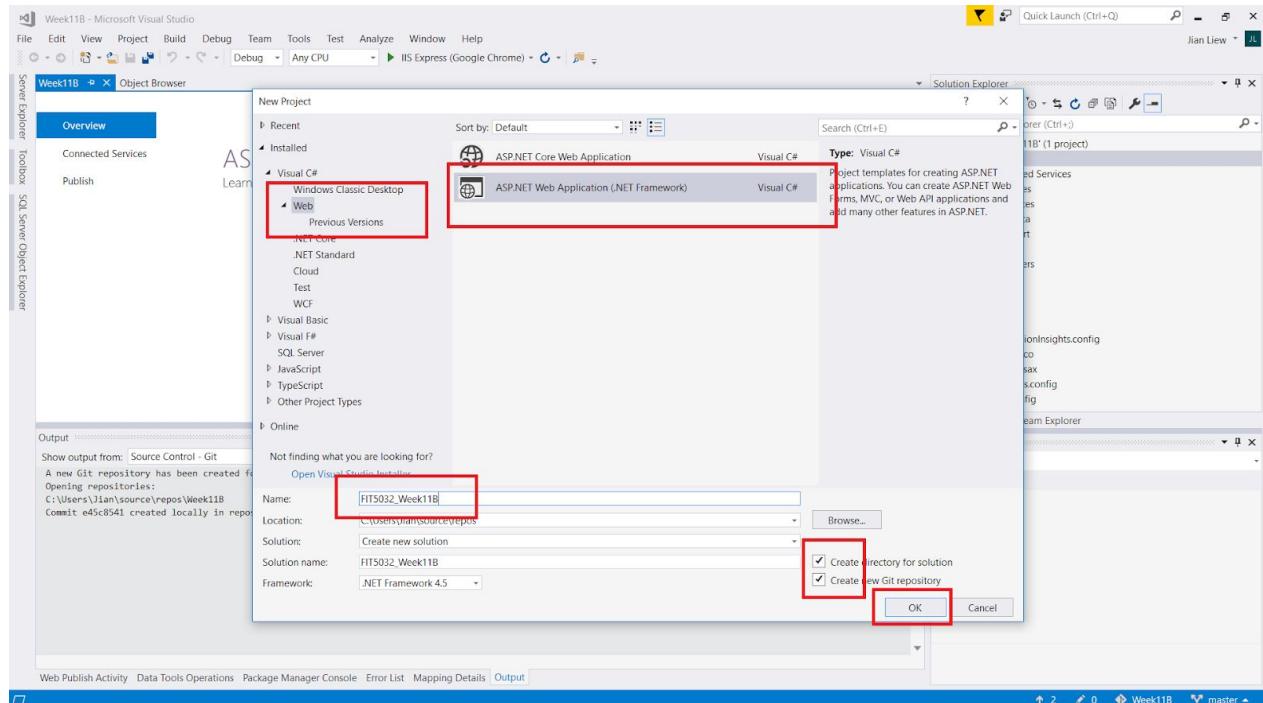
DoubtFire Submission

- **None**

Step 1

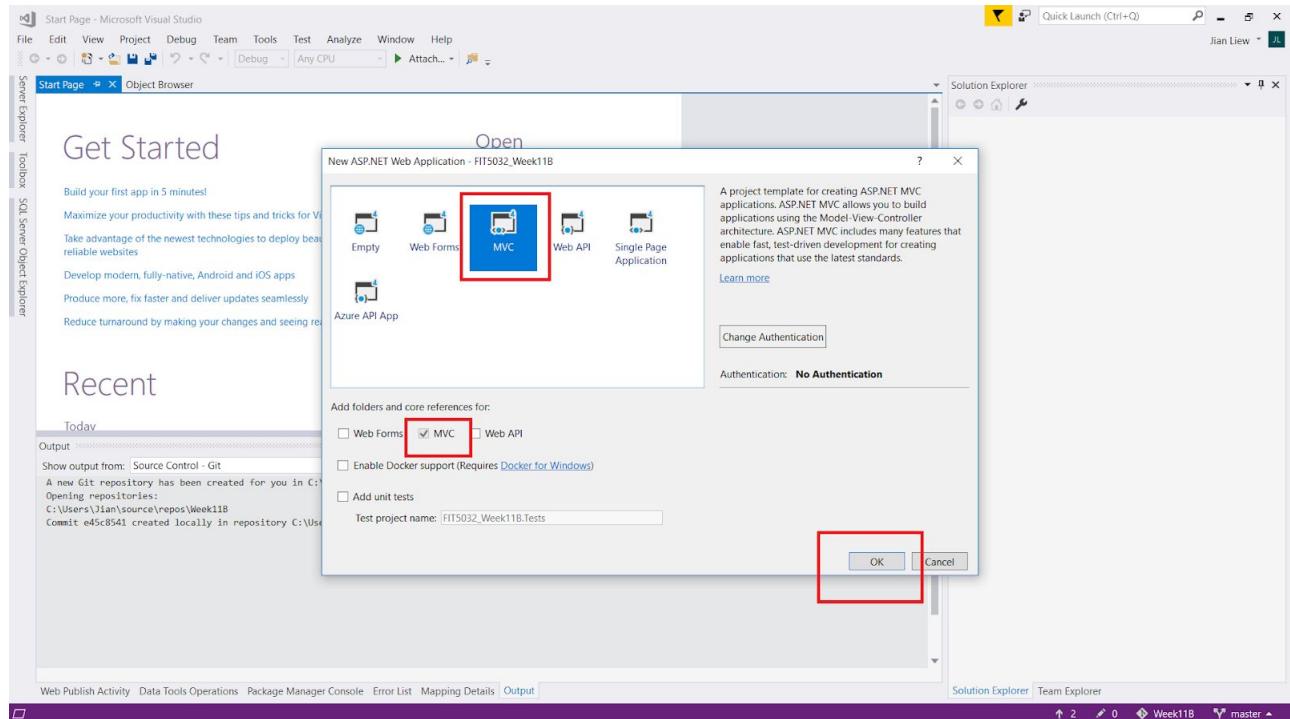
Before we start I will go ahead and make a new project.

This time, please have "Create a new Git repository selected"



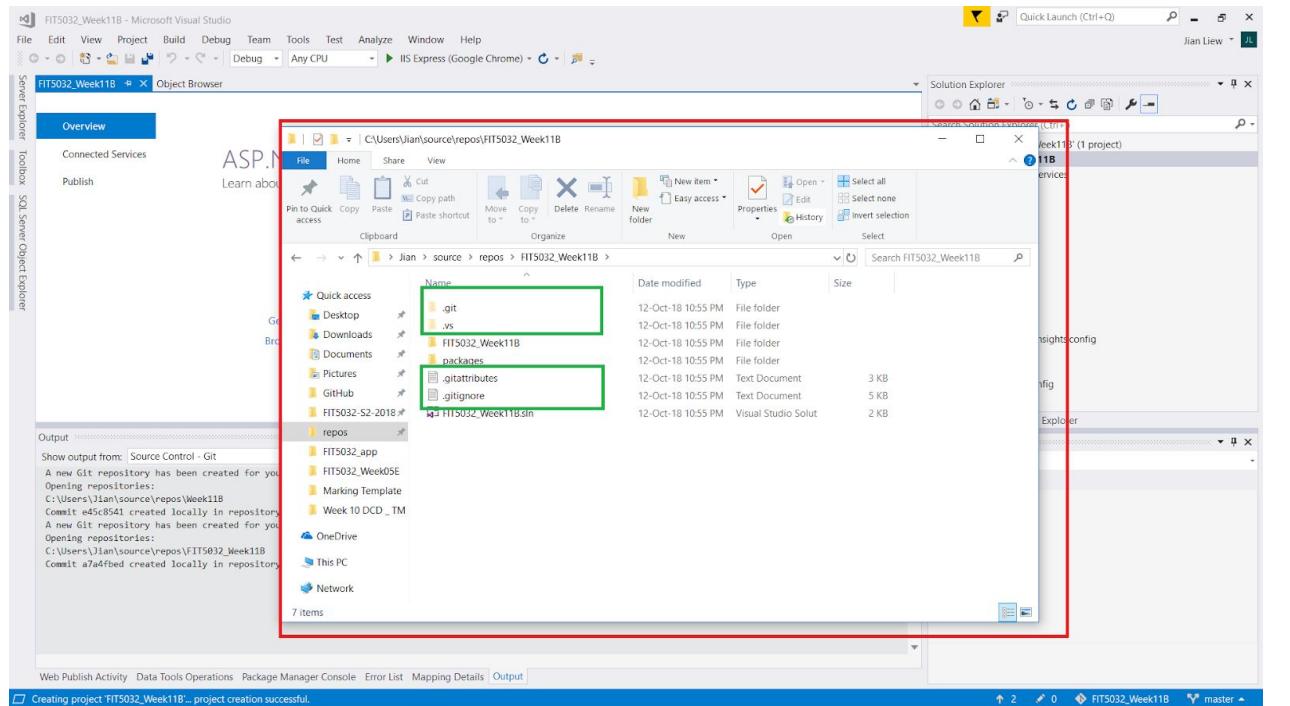
Most modern applications will often use Docker + Kubernetes. However, for our application we do not need to use it. (It is almost needed to know what Docker is these days)

Step 2



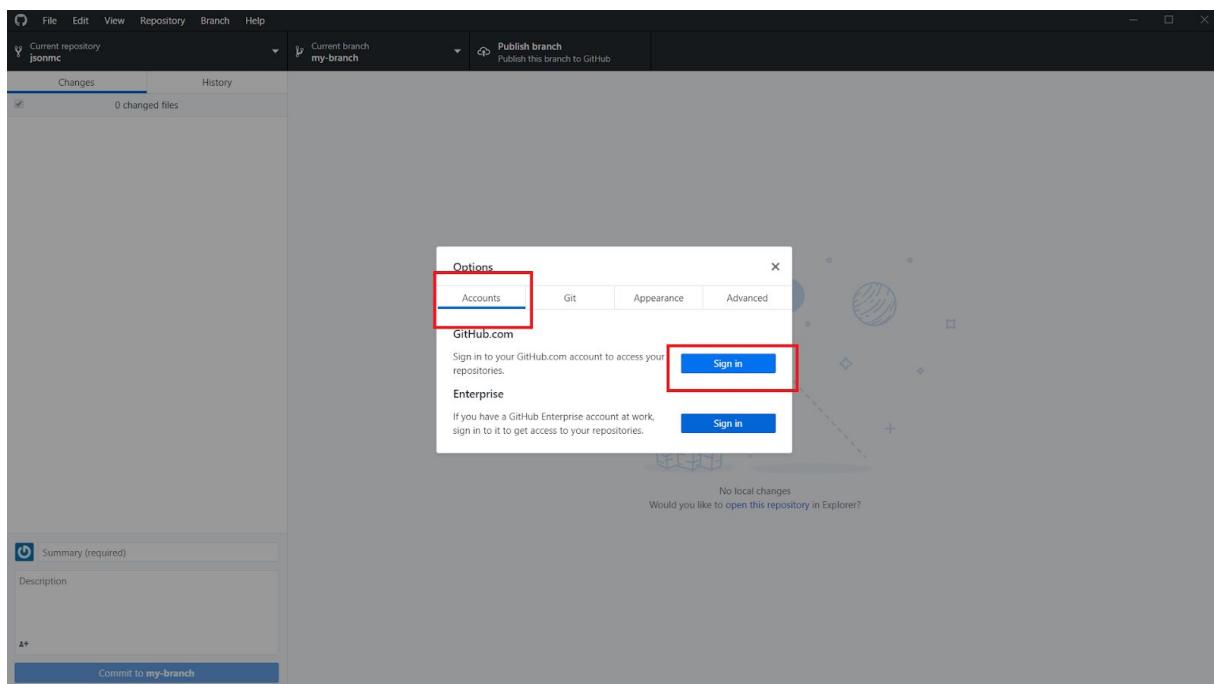
Step 3

If you open up, the folder which contains your solution, you will notice that if you created the project this way, a few files also created. (**.gitignore** as well as **.gitattribute**). **Both of these files are very important.** Newer developers will often not know what these are.

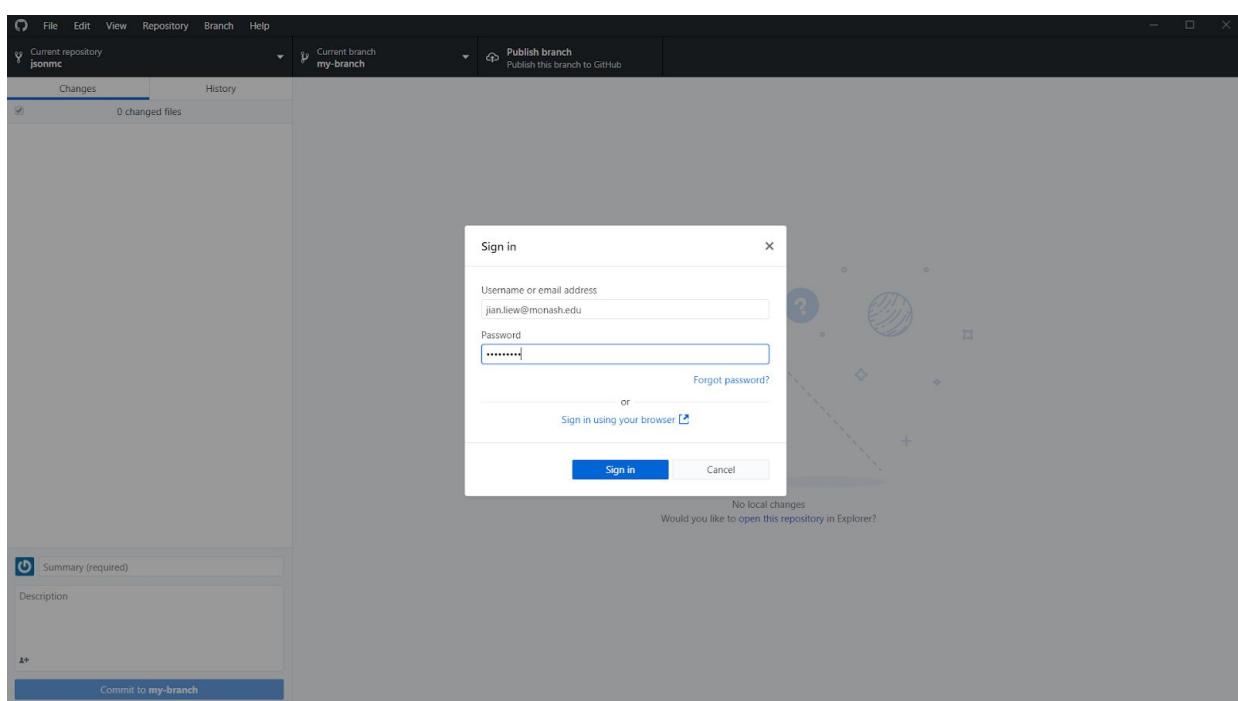


Step 4

I will now go ahead to my Github to create a new repository. You can do this via terminal, command line, git bash, or etc. However, I will use Github Desktop program instead. If you know what you are doing, you free to do it in another way.

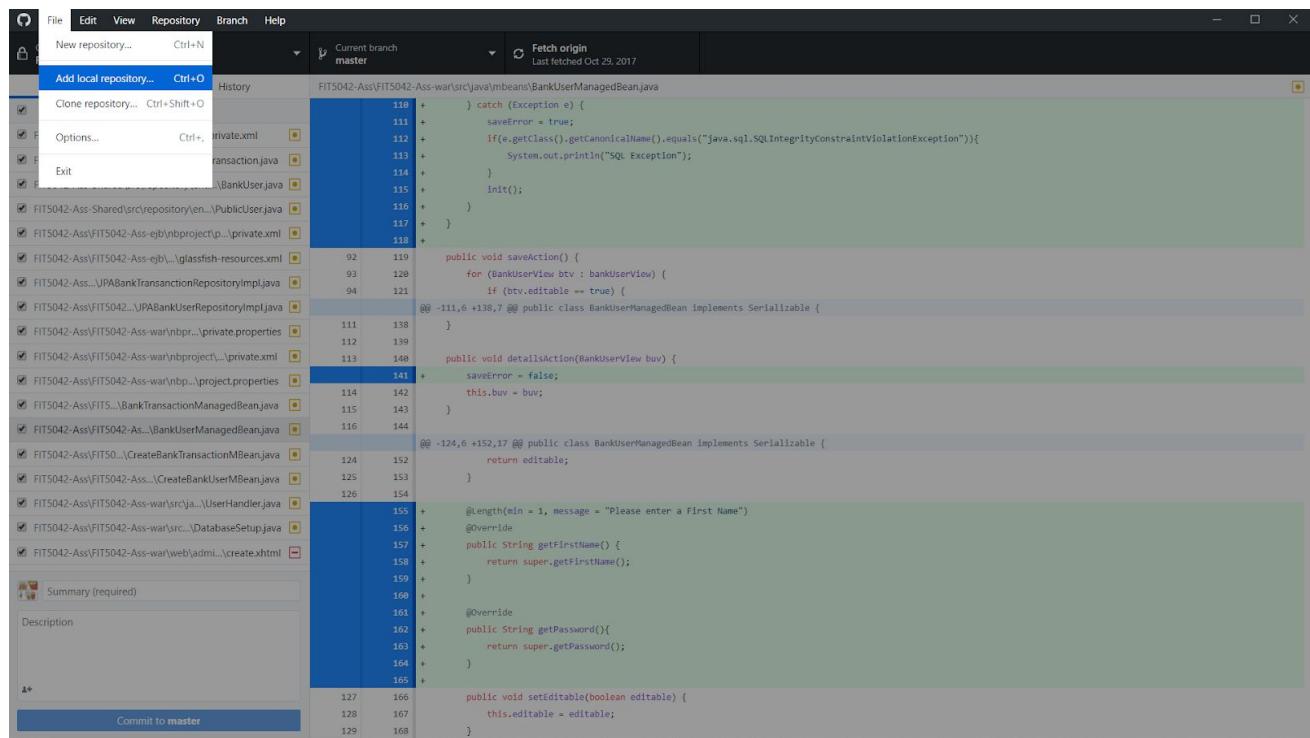


Please note that this is NOT your Monash Account, it is your Github username and password.



Step 5

Please add the Local Repository created in Step 3.



```

File Edit View Repository Branch Help
New repository... Ctrl+N Current branch master Fetch origin
Add local repository... Ctrl+O History FITS042-Ass-war(src\java)\mbeans\BankUserManagedBean.java
Clone repository... Ctrl+Shift+O
Options... Ctrl+, private.xml
Exit
FITS042-Ass-war\src\repository\en...\PublicUser.java
FITS042-Ass\Shared\src\repository\en...\PublicUser.java
FITS042-Ass\FIT5042-Ass-ejb\nbproject\ut...\Private.xml
FITS042-Ass\FIT5042-Ass-ejb\...\glassfish-resources.xml
FITS042-Ass...\JPABankTransactionRepositoryImpl.java
FITS042-Ass\FIT5042...\JPABankUserRepositoryImpl.java
FITS042-Ass\FIT5042-Ass-war\src\...\private.properties
FITS042-Ass\FIT5042-Ass-war\src\...\private.xml
FITS042-Ass\FIT5042-Ass-war\src\...\nbproject\...\project.properties
FITS042-Ass\FIT5042-Ass-war\src\...\BankTransactionManagedBean.java
FITS042-Ass\FIT5042-Ass-war\src\...\BankUserManagedBean.java
FITS042-Ass\FIT5042...\CreateBankTransactionMBean.java
FITS042-Ass\FIT5042-Ass...\CreateBankUserMBean.java
FITS042-Ass\FIT5042-Ass-war\src\...\UserHandler.java
FITS042-Ass\FIT5042-Ass-war\src\...\DatabaseSetup.java
FITS042-Ass\FIT5042-Ass-war\web\admin...\create.xhtml
Summary (required)
Description
Commit to master

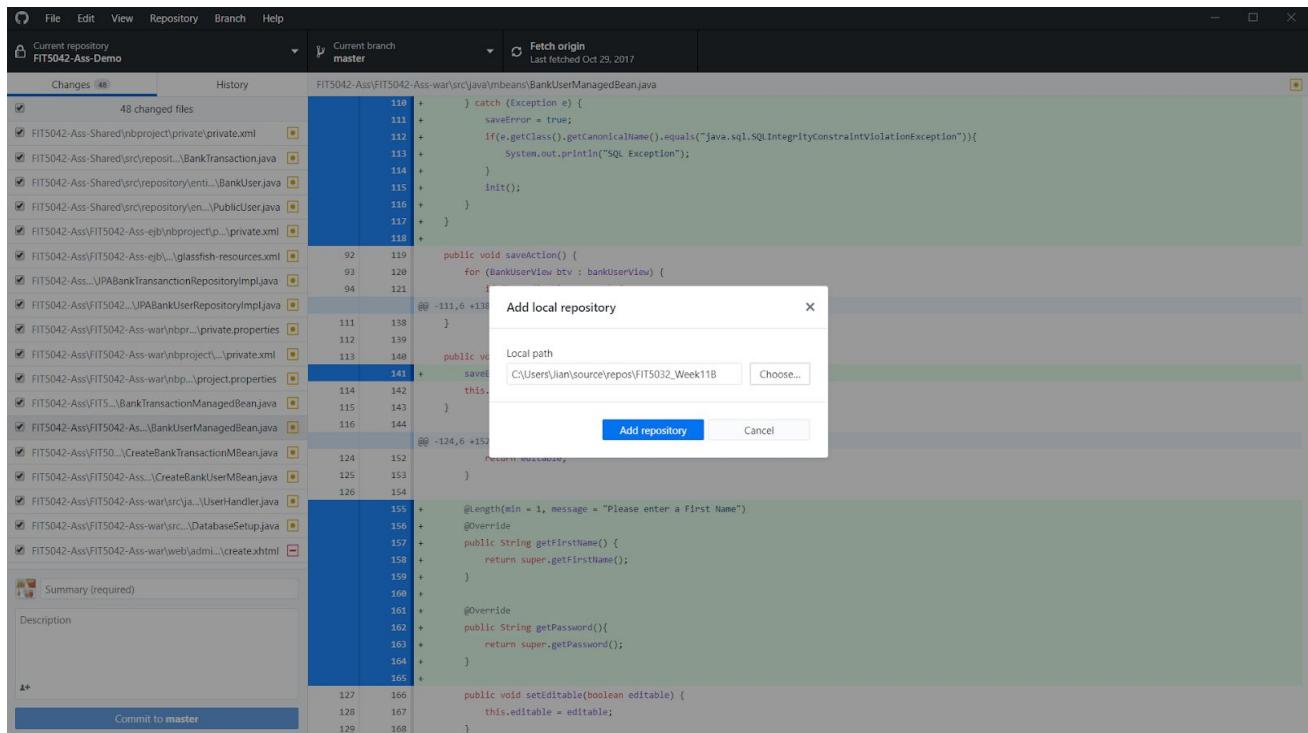
```

```

110 +     } catch (Exception e) {
111 +         saveError = true;
112 +         if (e.getClass().getCanonicalName().equals("java.sql.SQLIntegrityConstraintViolationException")){
113 +             System.out.println("SQL Exception");
114 +         }
115 +         init();
116 +     }
117 + }
118 +
119 +     public void saveAction() {
120 +         for (BankUserView btv : bankUserView) {
121 +             if (btv.setEditable == true) {
122 +                 @@ -111,6 +138,7 @@ public class BankUserManagedBean implements Serializable {
123 +                     ...
124 +                     return editable;
125 +                 }
126 +             }
127 +             @Length(min = 1, message = "Please enter a First Name")
128 +             @Override
129 +             public String getFirstName() {
130 +                 return super.getFirstName();
131 +             }
132 +             ...
133 +             @Override
134 +             public String getPassword(){
135 +                 return super.getPassword();
136 +             }
137 +             ...
138 +             public void setEditable(boolean editable) {
139 +                 this.setEditable = editable;
140 +             }
141 +         }
142 +     }
143 + }
144 +
145 +     @@ -124,6 +152,17 @@ public class BankUserManagedBean implements Serializable {
146 +         ...
147 +         ...
148 +         ...
149 +         ...
150 +         ...
151 +         ...
152 +         ...
153 +         ...
154 +         ...
155 +         ...
156 +         ...
157 +         ...
158 +         ...
159 +         ...
160 +         ...
161 +         ...
162 +         ...
163 +         ...
164 +         ...
165 +         ...
166 +         ...
167 +         ...
168 +         ...
169 +     }
170 + }

```

Step 6

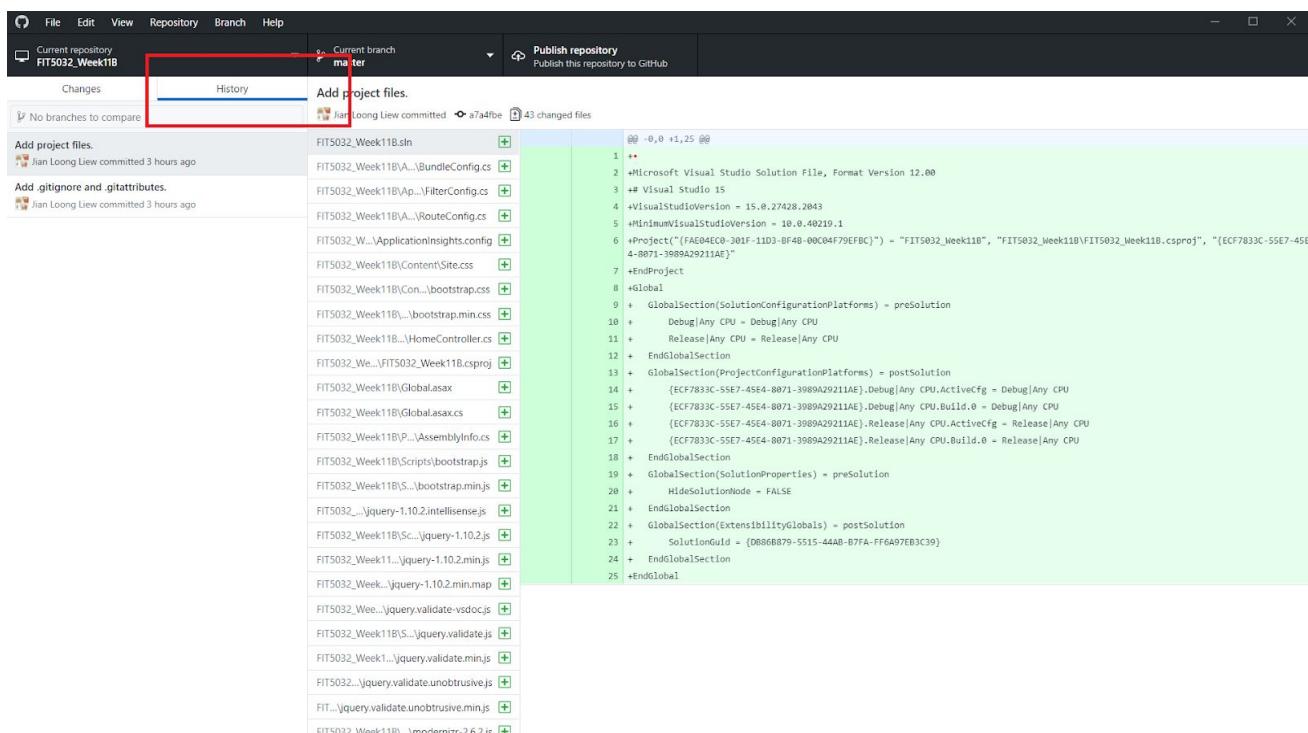


The screenshot shows the GitHub Desktop application interface. A pull request titled "FIT5042-Ass-Demo" is open, showing changes between the "Changes" tab and the "History" tab. The "Changes" tab lists 48 changed files, while the "History" tab shows a commit history with one entry from "Jian Loong Liew committed a7a4fbe". A modal dialog titled "Add local repository" is displayed over the code editor, prompting for a local path: "C:\Users\Jian\source\repos\FIT5032_Week11B". The code editor displays Java code for a BankUserManagedBean.java file, specifically focusing on the saveAction() method.

```

118 +         } catch (Exception e) {
119 +             saveError = true;
120 +             if(e.getClass().getCanonicalName().equals("java.sql.SQLIntegrityConstraintViolationException")){
121 +                 System.out.println("SQL Exception");
122 +             }
123 +             init();
124 +         }
125 +     }
126 +
127 +     public void saveAction() {
128 +         for (BankUserView btv : bankUserView) {
129 +             ...
130 +         }
131 +     }
132 + }
133 +
134 + public void save() {
135 +     ...
136 +     this...
137 + }
138 +
139 + public void setEditable(boolean editable) {
140 +     this.setEditable = editable;
141 + }
142 + }
143 +
144 + public String getFirstName() {
145 +     ...
146 +     return super.getFirstName();
147 + }
148 +
149 + @Override
150 + public String getPassword() {
151 +     ...
152 +     return super.getPassword();
153 + }
154 +
155 + @Length(min = 1, message = "Please enter a First Name")
156 + @Override
157 + public String getFirstName() {
158 +     ...
159 +     return super.getFirstName();
160 + }
161 +
162 + @Override
163 + public String getPassword() {
164 +     ...
165 +     return super.getPassword();
166 + }
167 +
168 + public void setEditable(boolean editable) {
169 +     this.setEditable = editable;
170 + }

```



The screenshot shows the GitHub Desktop application interface. A pull request titled "FIT5032_Week11B" is open, with the "History" tab selected (indicated by a red box). The "Changes" tab shows a commit from "Jian Loong Liew committed a7a4fbe" with 43 changed files. The code editor displays a Visual Studio Solution File (FIT5032_Week11B.sln) with numerous changes, primarily in the Global.asax.cs file, related to project configuration and solution properties.

```

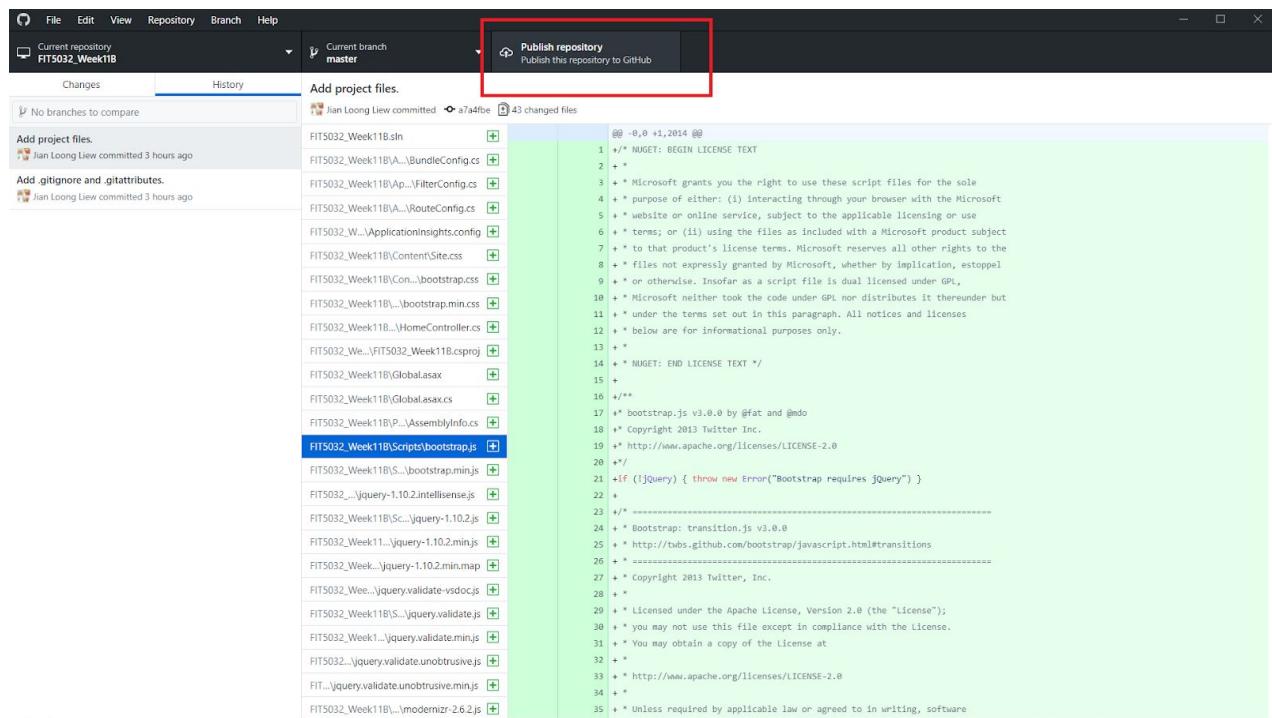
1 ++
2 +Microsoft Visual Studio Solution File, Format Version 12.00
3 +# Visual Studio 15
4 +VisualStudioVersion = 15.0.27428.2043
5 +MinimunVisualStudioVersion = 10.0.40219.1
6 +Project["{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}"] = "FIT5032_Week11B", "{ECF783C-SSE7-45E4-B071-3980A29211AE}"
7 +EndProject
8 +Global
9 + GlobalSection(SolutionConfigurationPlatforms) = preSolution
10 + Debug|Any CPU = Debug|Any CPU
11 + Release|Any CPU = Release|Any CPU
12 + EndGlobalSection
13 + GlobalSection(ProjectConfigurationPlatforms) = postSolution
14 + {ECF783C-SSE7-45E4-B071-3980A29211AE}.Debug|Any CPU.ActiveCfg = Debug|Any CPU
15 + {ECF783C-SSE7-45E4-B071-3980A29211AE}.Debug|Any CPU.Build.0 = Debug|Any CPU
16 + {ECF783C-SSE7-45E4-B071-3980A29211AE}.Release|Any CPU.ActiveCfg = Release|Any CPU
17 + {ECF783C-SSE7-45E4-B071-3980A29211AE}.Release|Any CPU.Build.0 = Release|Any CPU
18 + EndGlobalSection
19 + GlobalSection(SolutionProperties) = preSolution
20 + HideSolutionNode = FALSE
21 + EndGlobalSection
22 + GlobalSection(ExtensibilityGlobals) = postSolution
23 + SolutionGuid = {D886B879-5515-44AB-B7FA-FF6A97EB3C39}
24 + EndGlobalSection
25 +EndGlobal

```

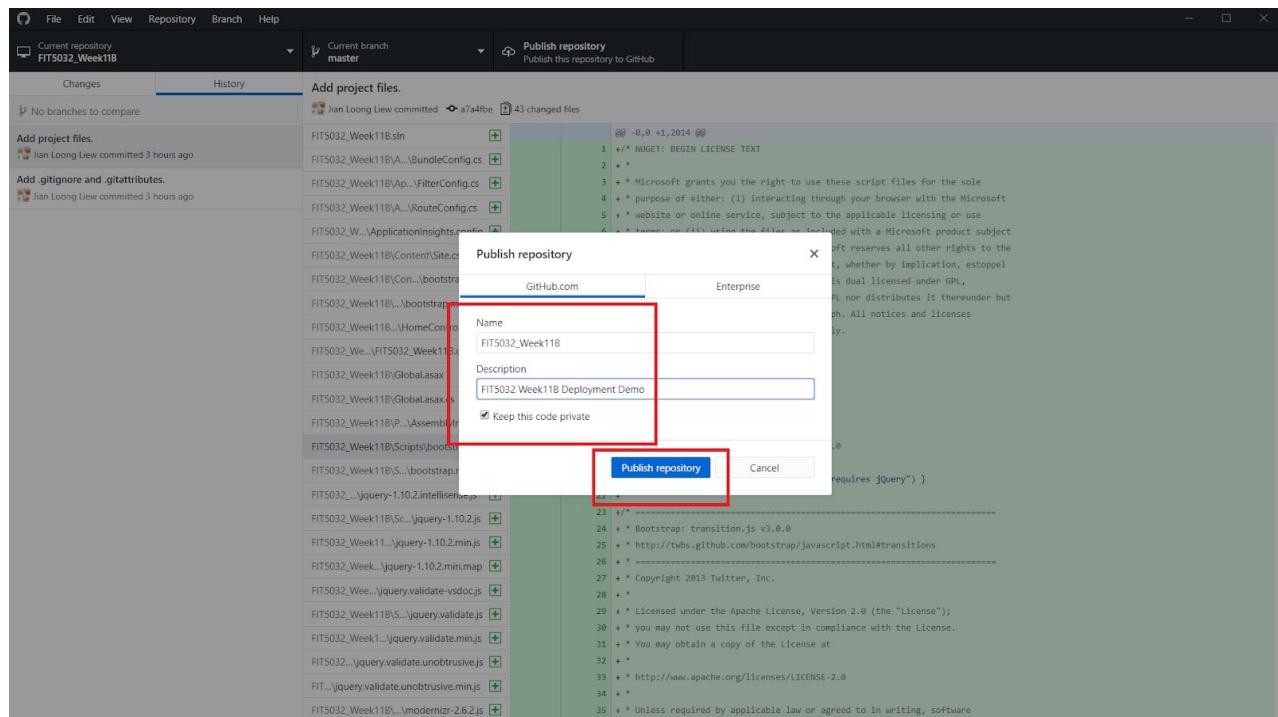
There should be some history if you check it now.

Step 7

I will go ahead and Publish my repository.

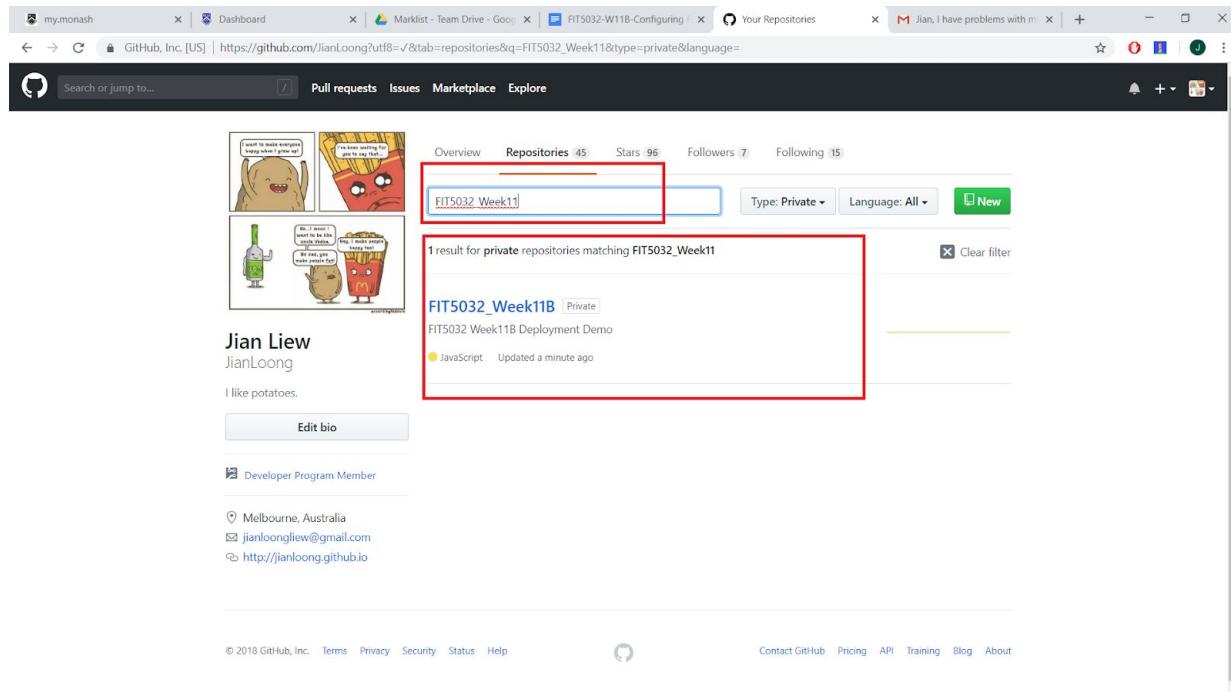


Publishing it means pushing it to the remote repository.



Step 8

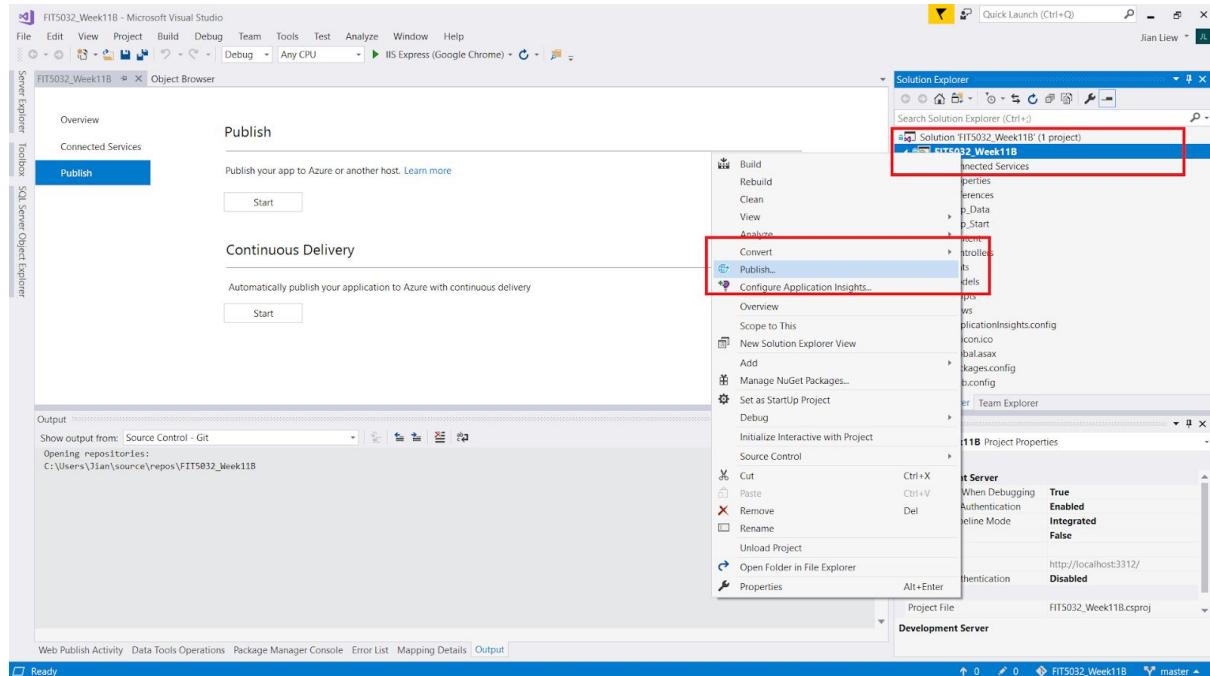
You should be able to see it as your private repository. On the Github website.



The screenshot shows a GitHub profile page for 'Jian Liew' (JianLoong). The user has 45 repositories, 96 stars, 7 followers, and 15 following. A search bar at the top is set to 'Type: Private'. A red box highlights the search results for 'FIT5032_Week11'. One result is shown: 'FIT5032_Week11B' (Private), which is a FIT5032 Week11B Deployment Demo in JavaScript, updated a minute ago. The GitHub interface includes standard navigation like Pull requests, Issues, Marketplace, and Explore.

Step 9

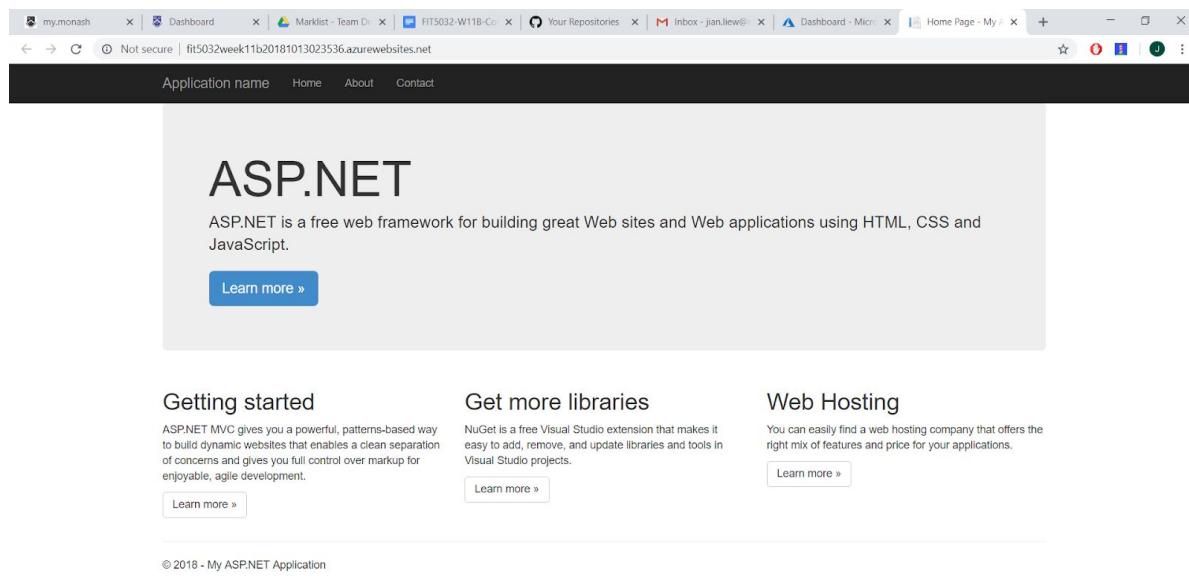
I will not go back into my Visual Studio and Publish/Deploy my application. The steps to do this has been covered in Week11A. (Quick Deployment to Azure via Service App)



The screenshot shows Microsoft Visual Studio with the project 'FIT5032_Week11B' selected in the Solution Explorer. The 'Publish' tab is active in the left-hand navigation. A context menu is open over the project node in the Solution Explorer, with 'Configure Application Insights...' highlighted. Other options in the menu include Convert, Publish..., and Set as StartUp Project. The Visual Studio interface includes the Object Browser, Task List, and Output windows.

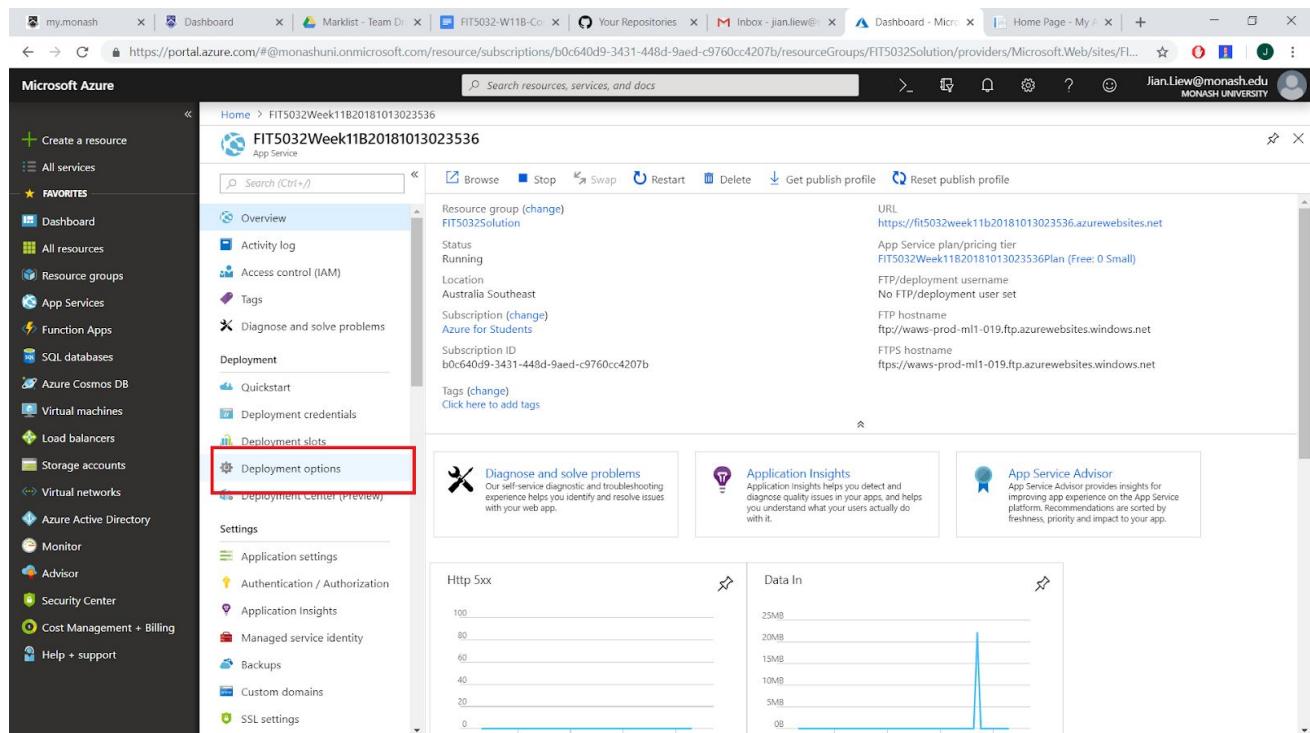
Step 10

You can just automatically deploy it by creating new resources.



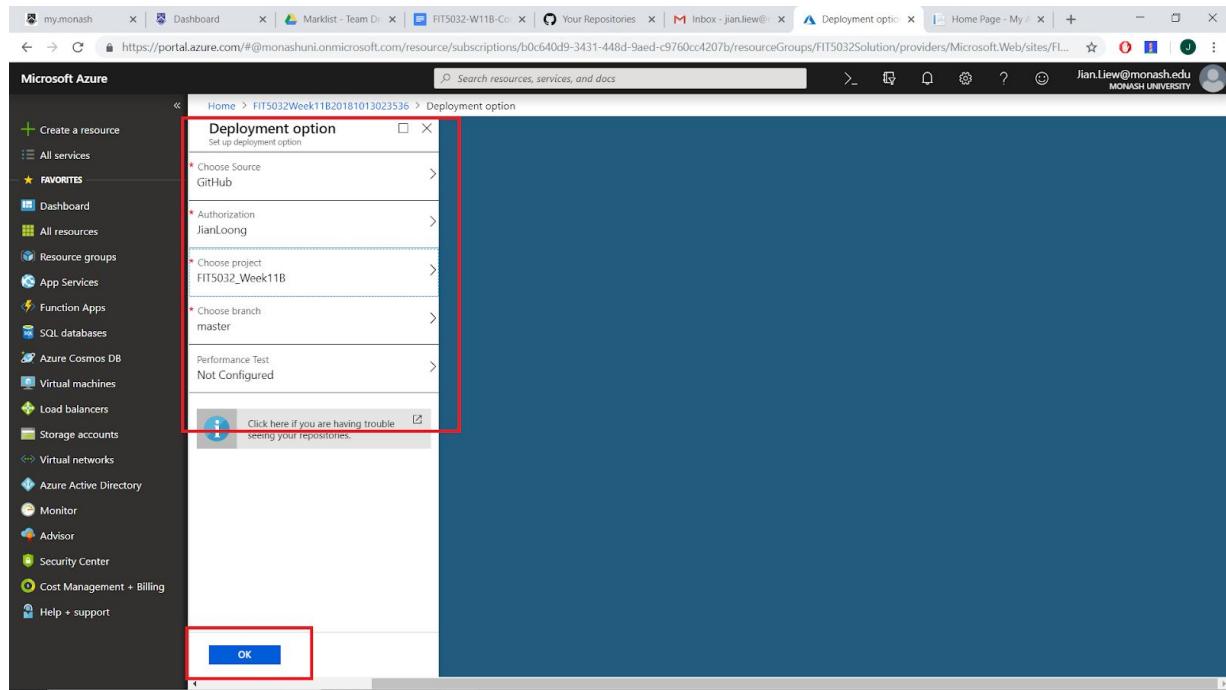
Step 11

We will now jump back to Azure and set up our CD. Find your App Service and go to Deployment options.



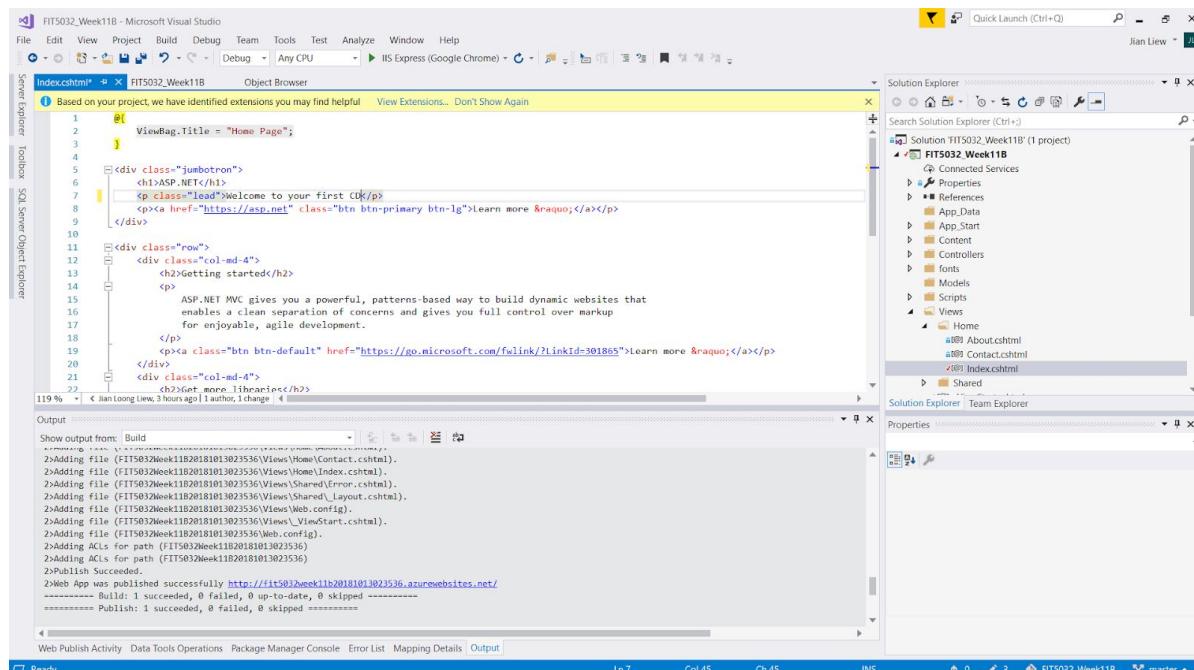
Step 12

Configure it to use your Github. (Make sure you select the correct Project)



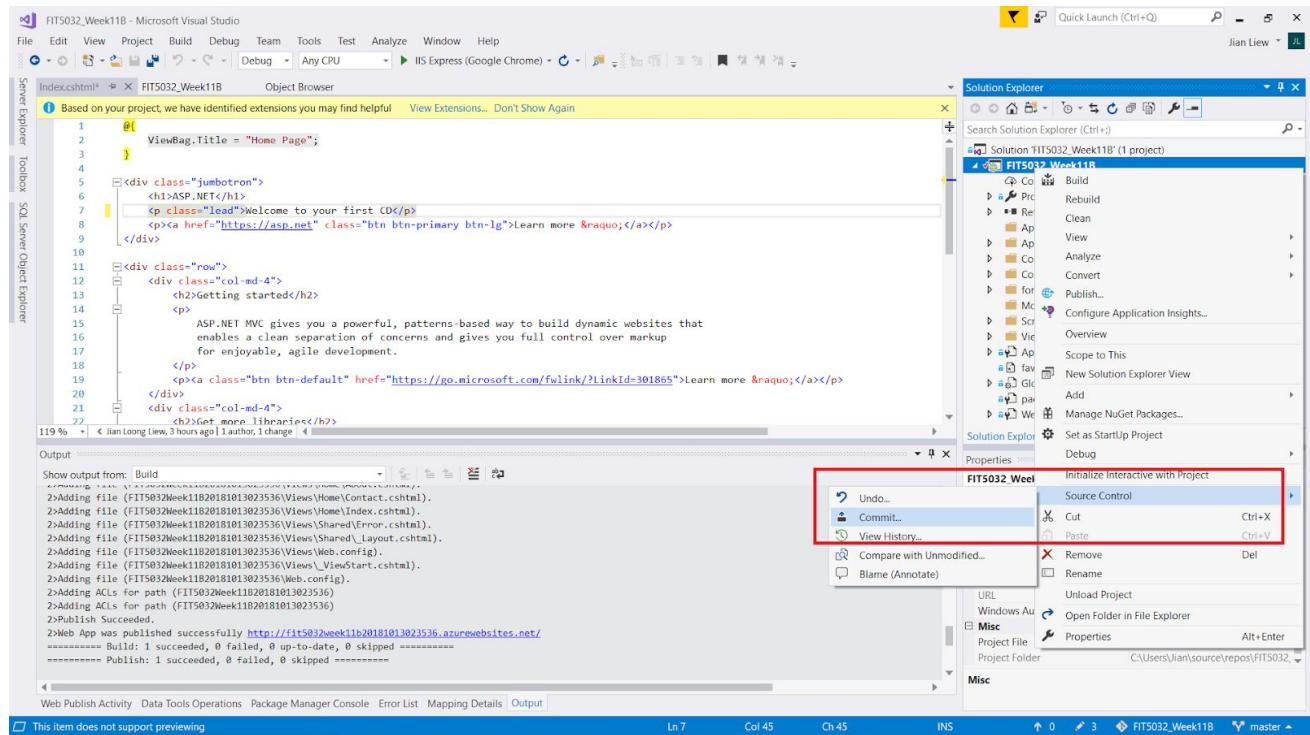
Step 13

We will now just change the information in the jumbotron.

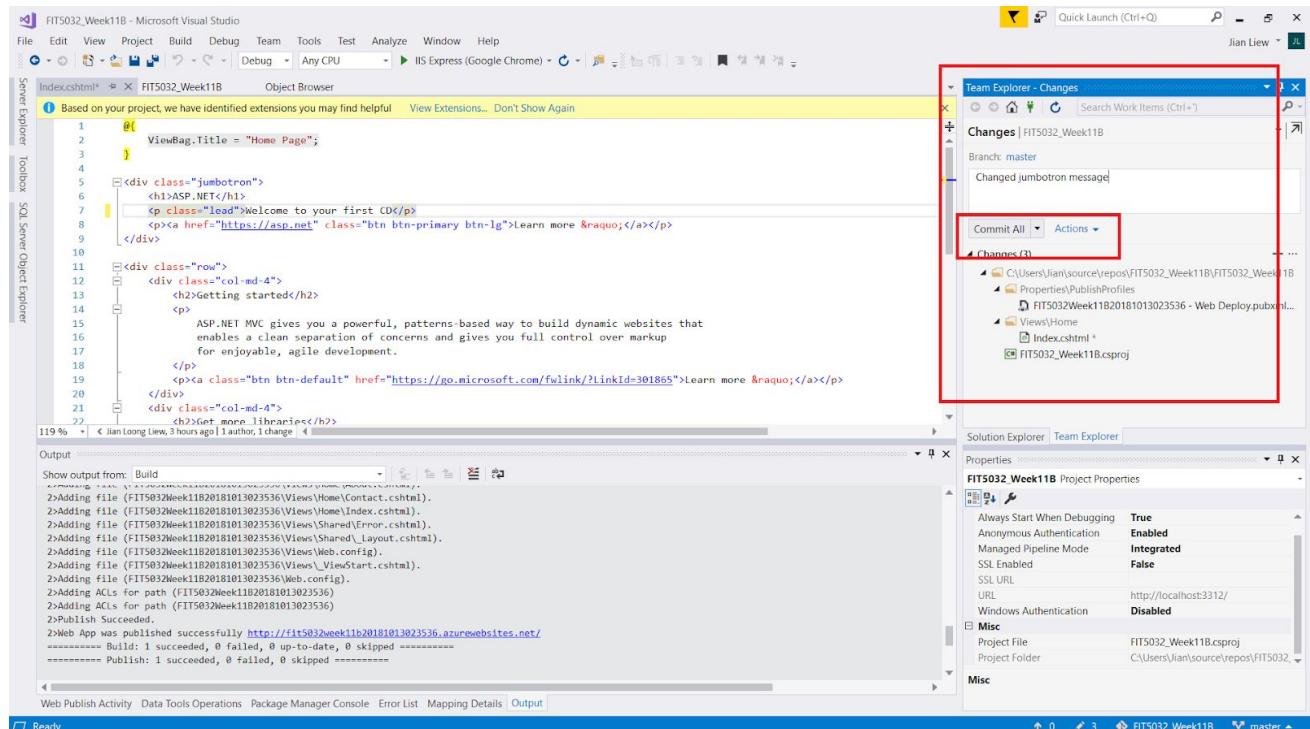


Step 14

Now, use the IDE to commit your changes.

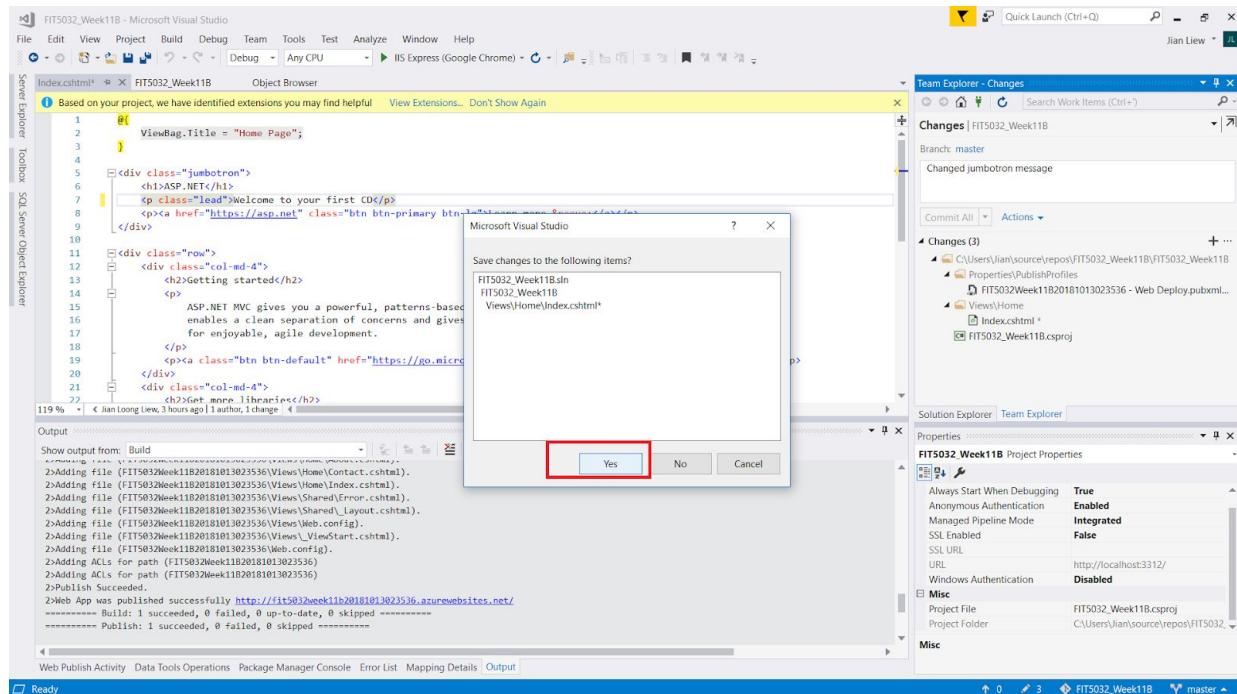


Step 15



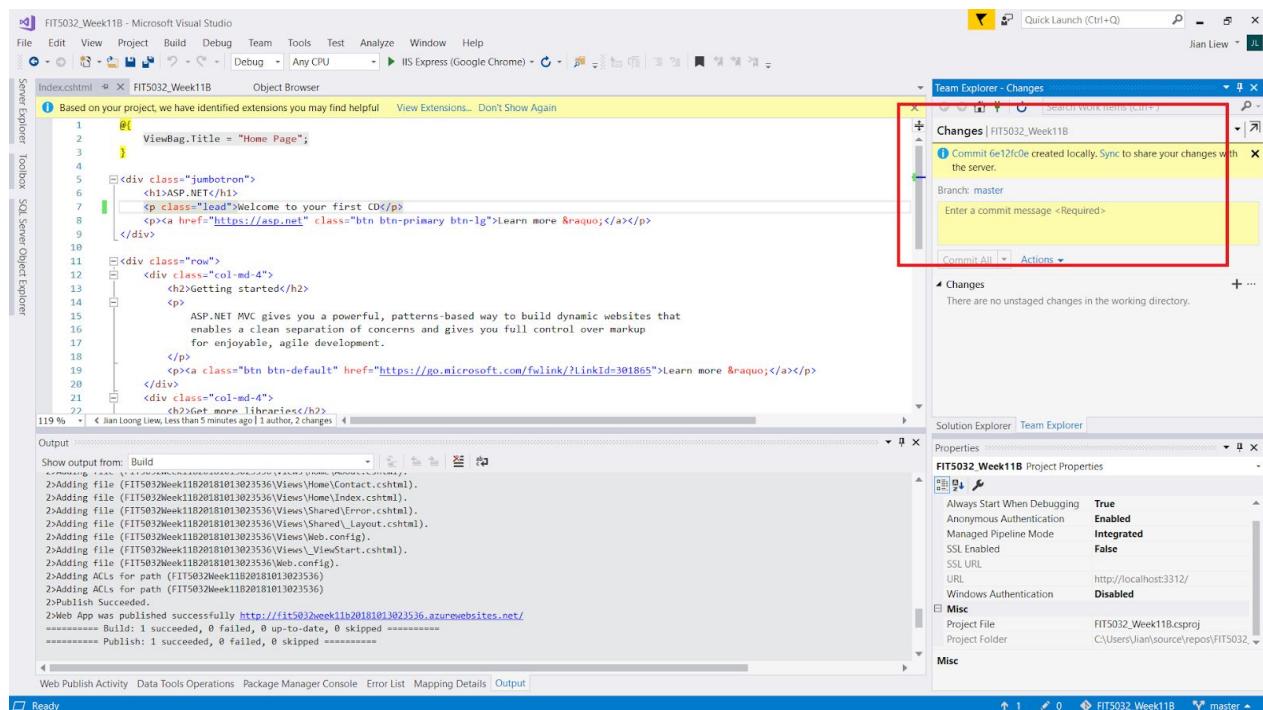
Step 16

If you see this, just click "Yes"

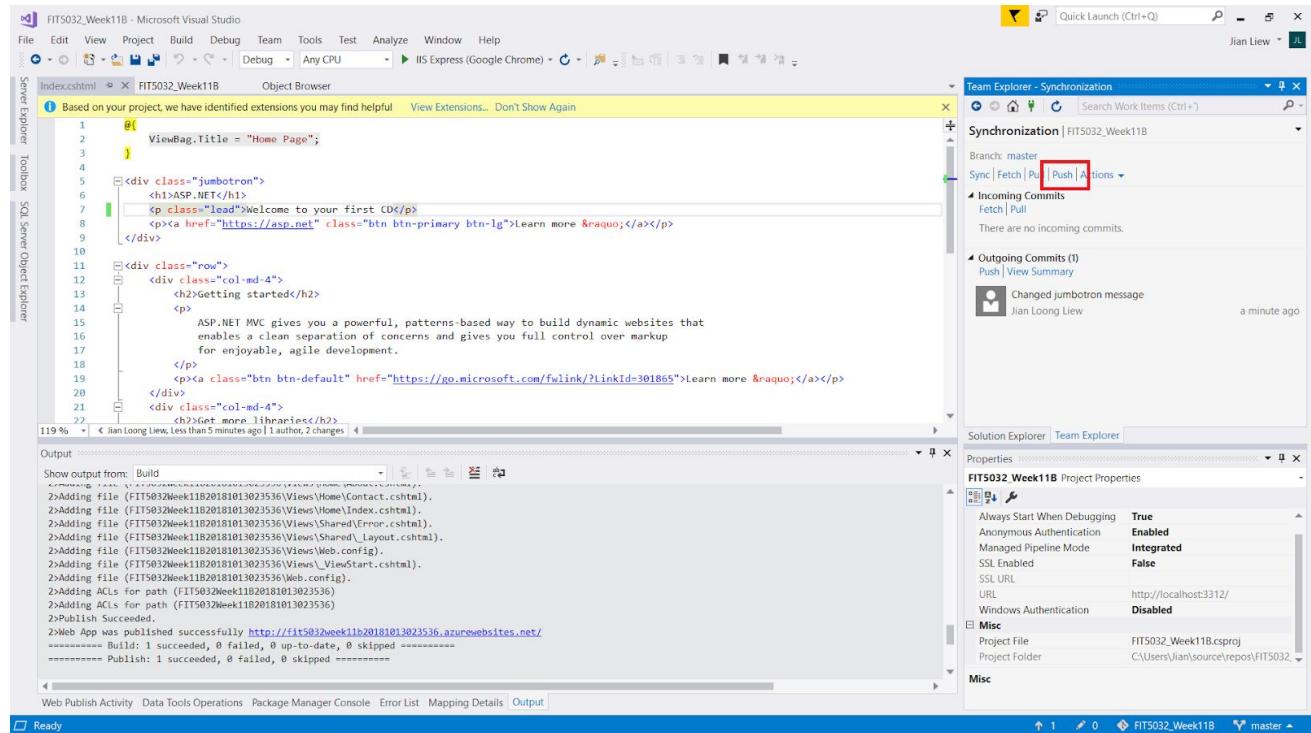


Step 17

Proceed to sync to the remote. By doing a "Push"



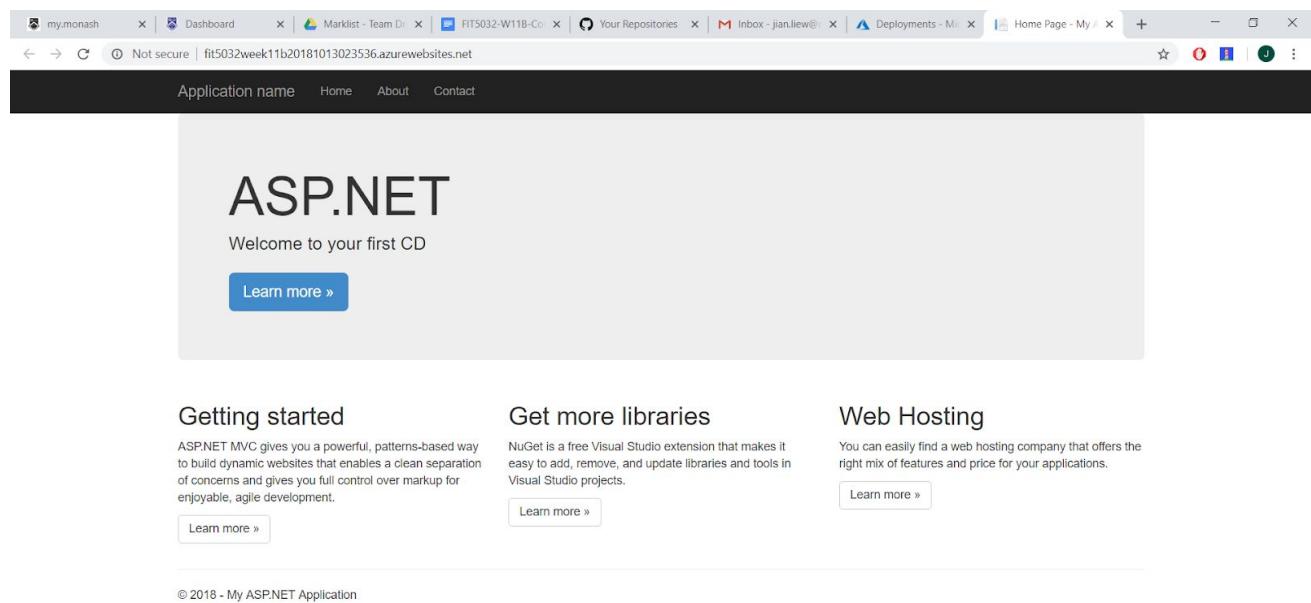
Step 18



The screenshot shows the Microsoft Visual Studio interface with the 'Team Explorer - Synchronization' pane open. The 'Push' button in the top right of the pane is highlighted with a red box. The 'Incoming Commits' and 'Outgoing Commits' sections are visible, showing a single commit from 'Jian Loong Liew' titled 'Changedjumbotron message' pushed a minute ago.

Step 19

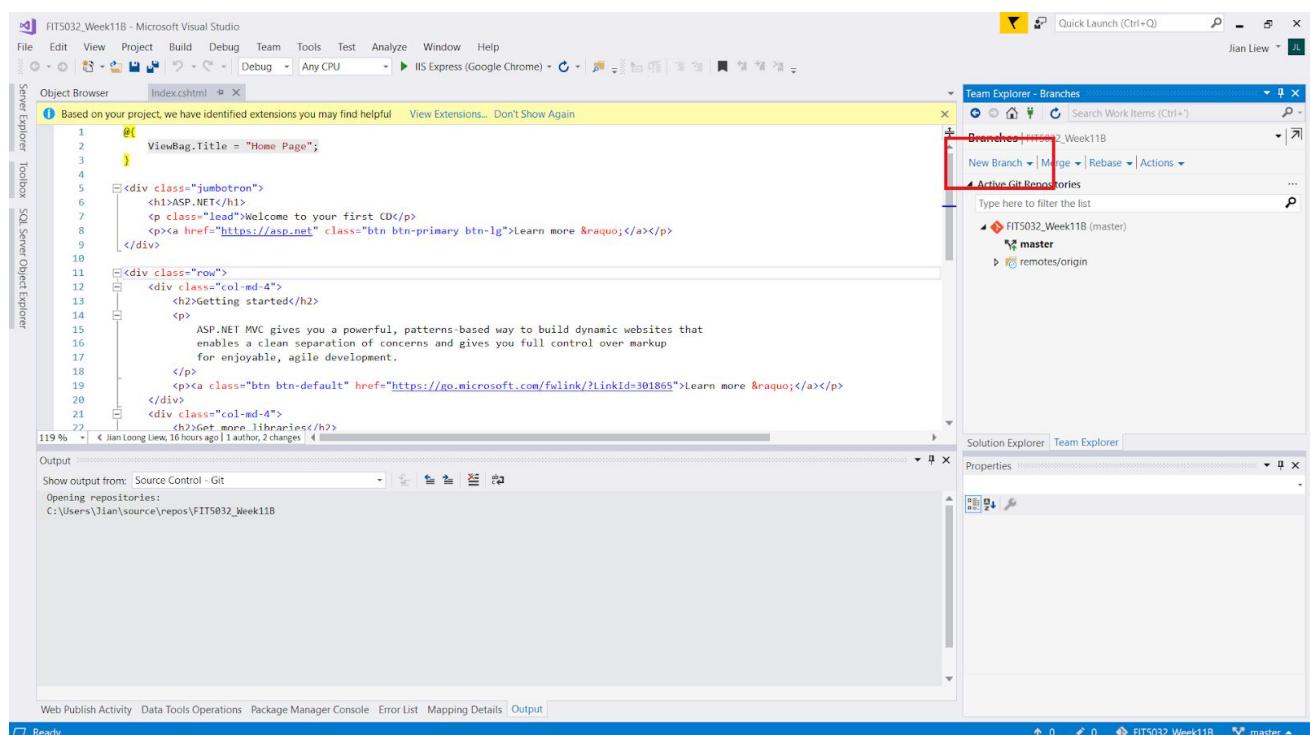
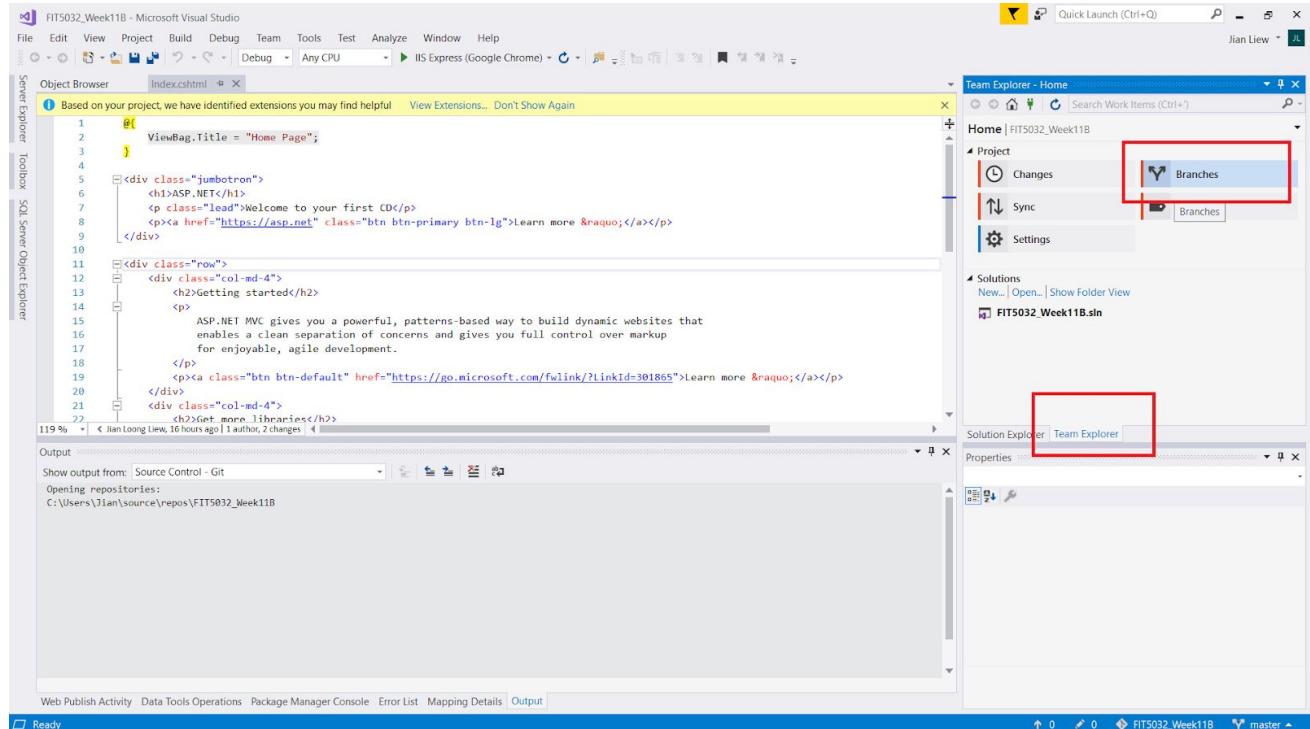
Once, you have completed that, upon a "Push", the continuous deployment will automatically be triggered. So, when you go back to the website, you will notice it has been changed.



The screenshot shows a web browser displaying the deployed ASP.NET application. The URL is my.monash. The page content includes the 'ASP.NET' logo, a 'Welcome to your first CD' message, and three main sections: 'Getting started', 'Get more libraries', and 'Web Hosting'. Each section has a 'Learn more >' button. At the bottom, there is a copyright notice: '© 2018 - My ASP.NET Application'.

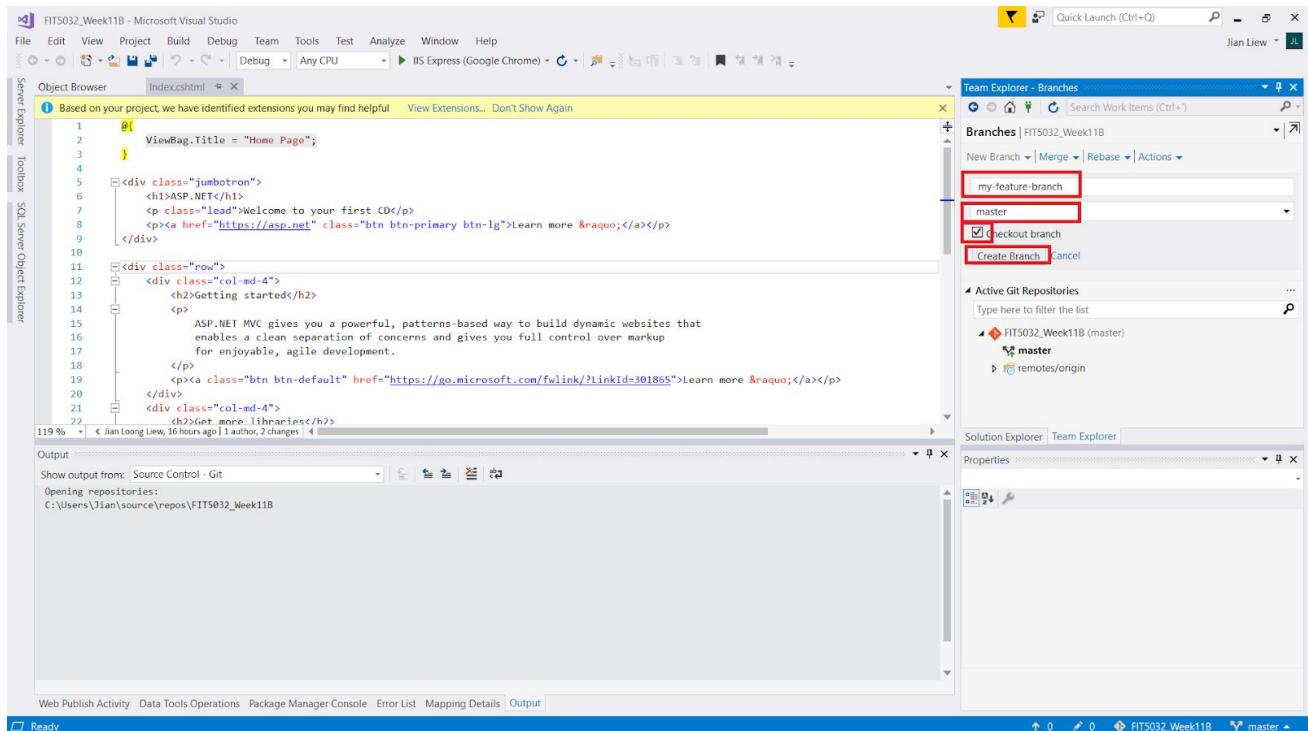
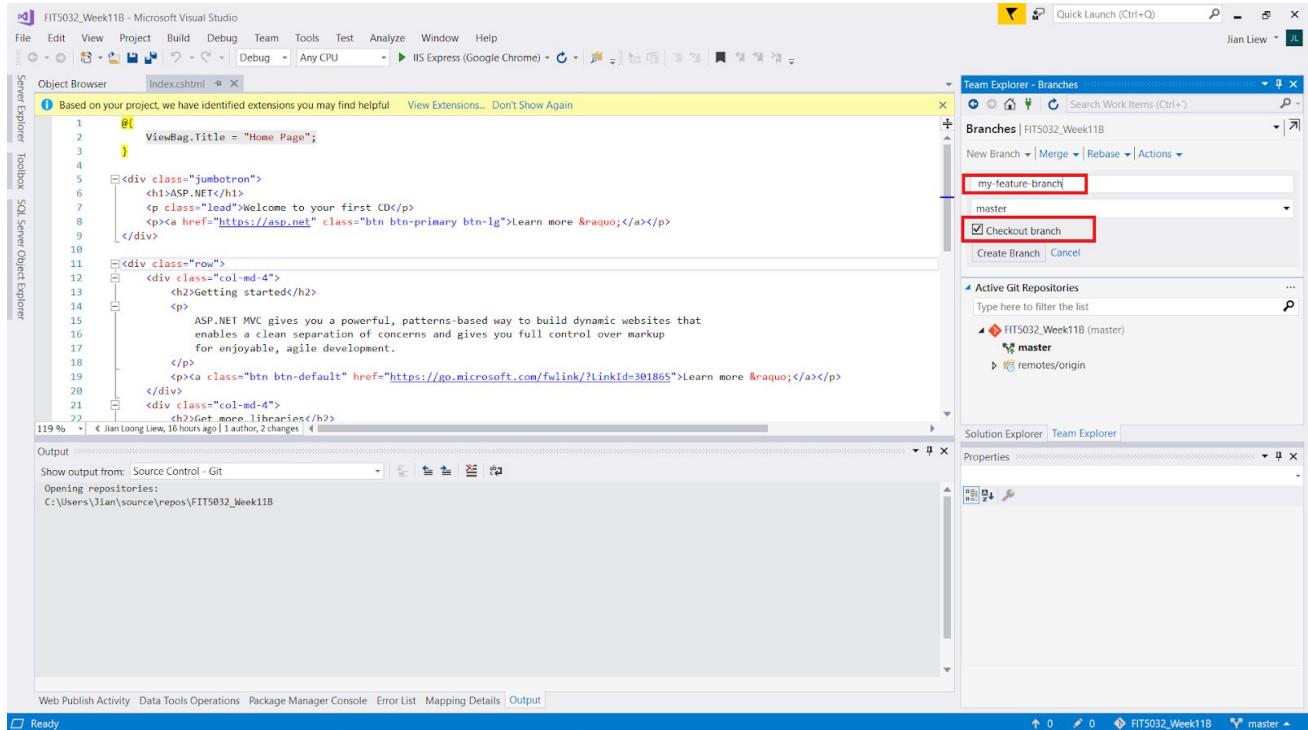
Step 20

You will now learn a git feature known as branching. Branching is quite a common technique. Often times when multiple developers work together, it is much easier to manage the "master" source codes via branches. The reason for this, is because merge conflicts are easier to be handled this way.



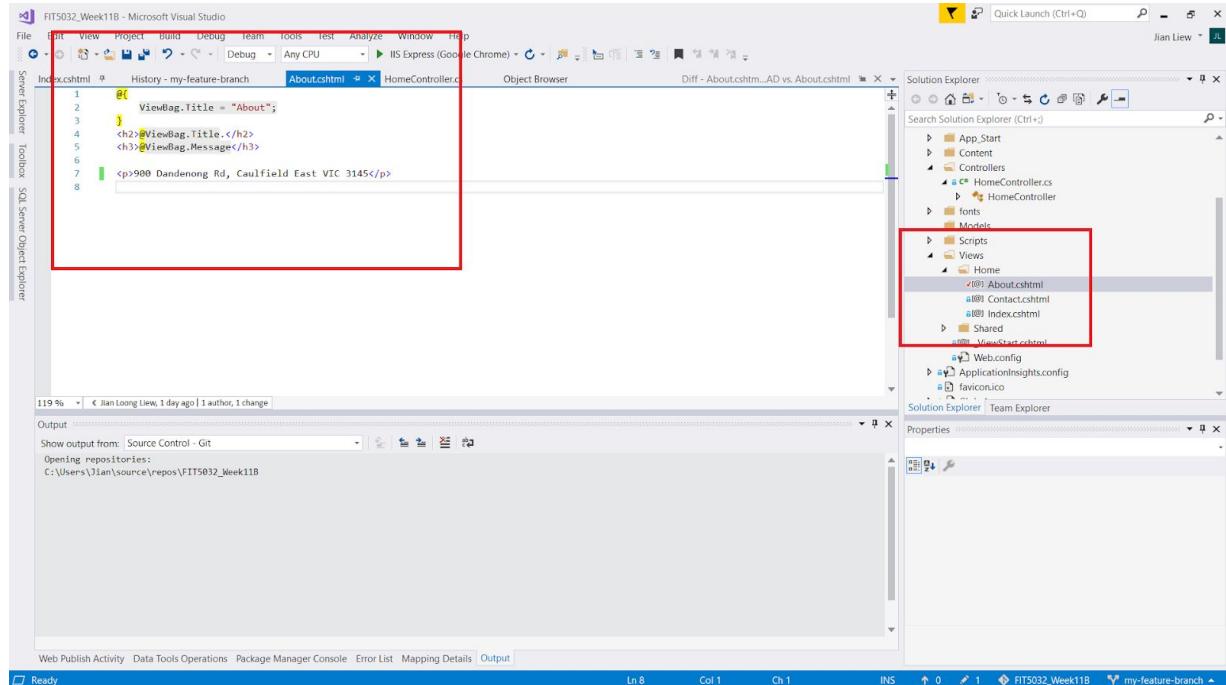
Step 21

The name of your branch does not matter, as it only exist in the local repository for now. Proceed to create a branch and check it out. **Checking it out means that your current workspace will be switched to the branch, and you are no longer working on the master.**



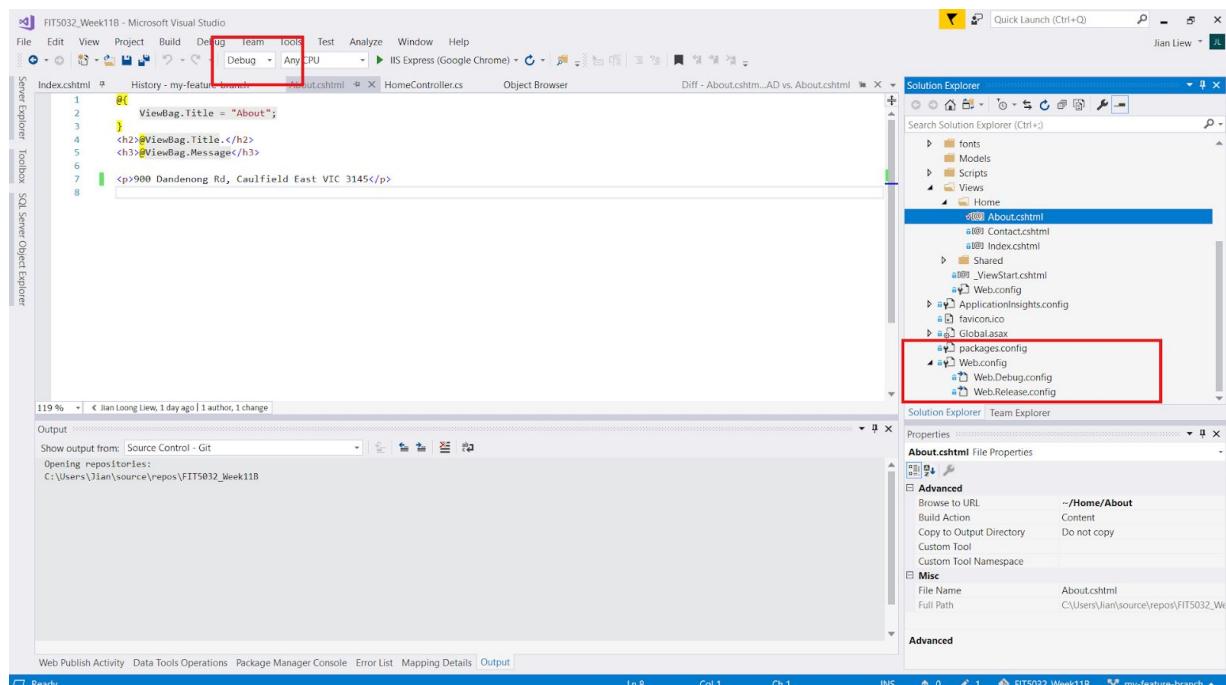
Step 21

After we have done so we can now introduce a feature that we want. **This can be a whole use case, a minor change or anything.** We will then introduce the changes. The objective of this is to introduce the git feature known as branching. We will introduce a simple feature like the one below.



Step 22

Keep in the various configurations for different environments. Debug / Release or etc.



Step 23

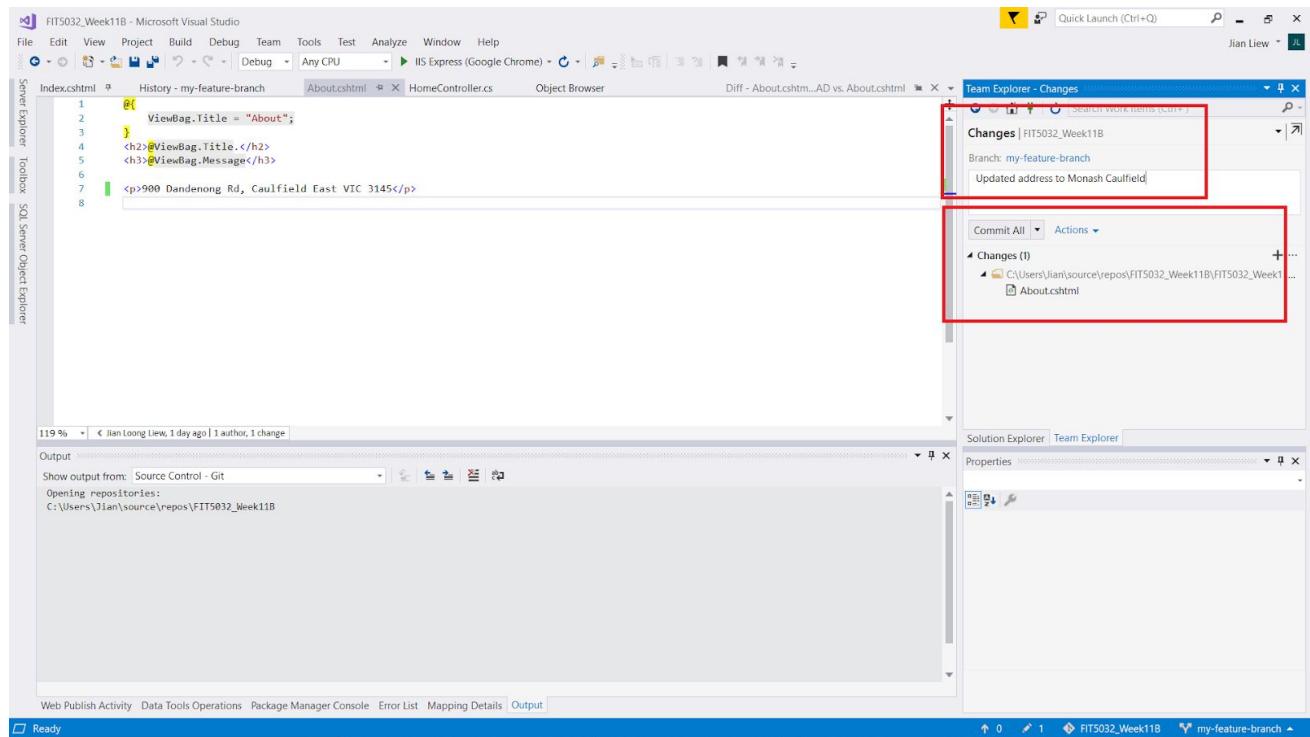
Since we have made changes and introduced a feature on our branch. We will now, merge it back into the main branch (master). There is a concept known as fast forward merging. (But we are not doing that)

Normally though, what we should do is to check if the remote master branch has changes, thus often times we will need to pull changes from the remote master. This however, would depend on how your team uses version handling. If there are changes, we will need to merge the changes from the new master to our feature or dev branch. We can also rebase as well.

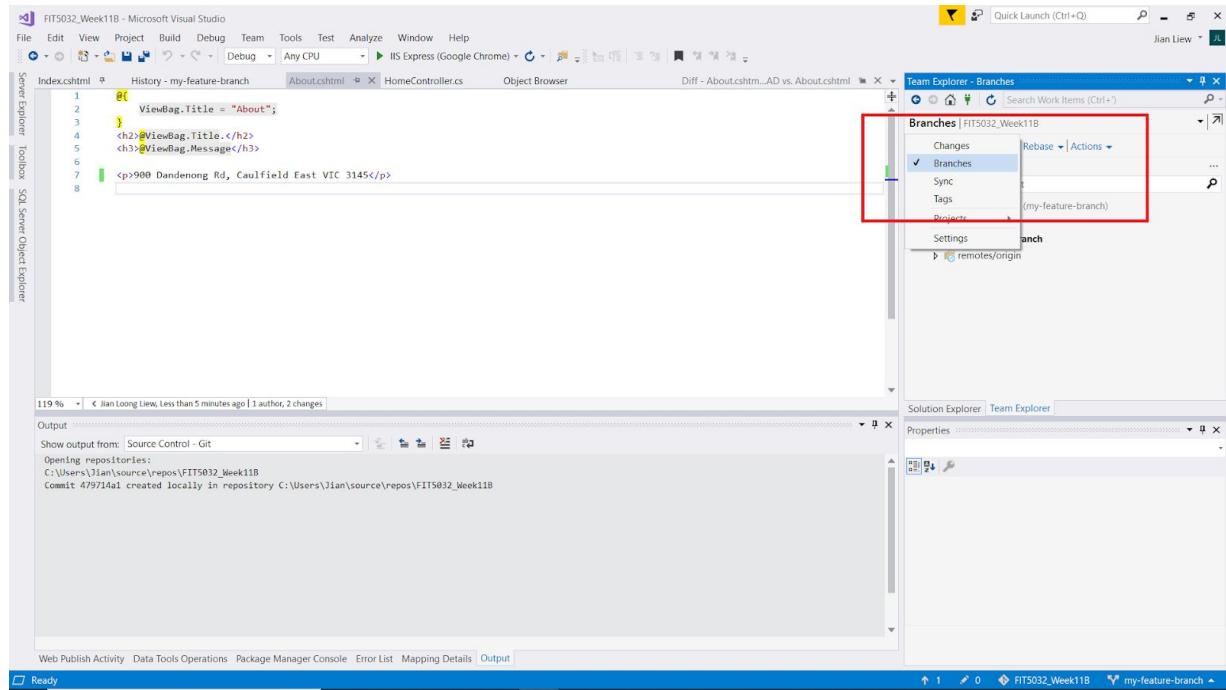
Before you do all the steps, we will need to add & commit the newly made changes to our feature branch. (git status) → a command line command that tells you the status of the git repository

Put a reasonable commit message and commit the changes. (Make sure you are on the right branch)

As of this moment, everything may seem very confusing however, over time you will learn more about it when you use it more and work with teams. Keep in mind that there is always a remote (stored online) and a local version (stored on your computer).

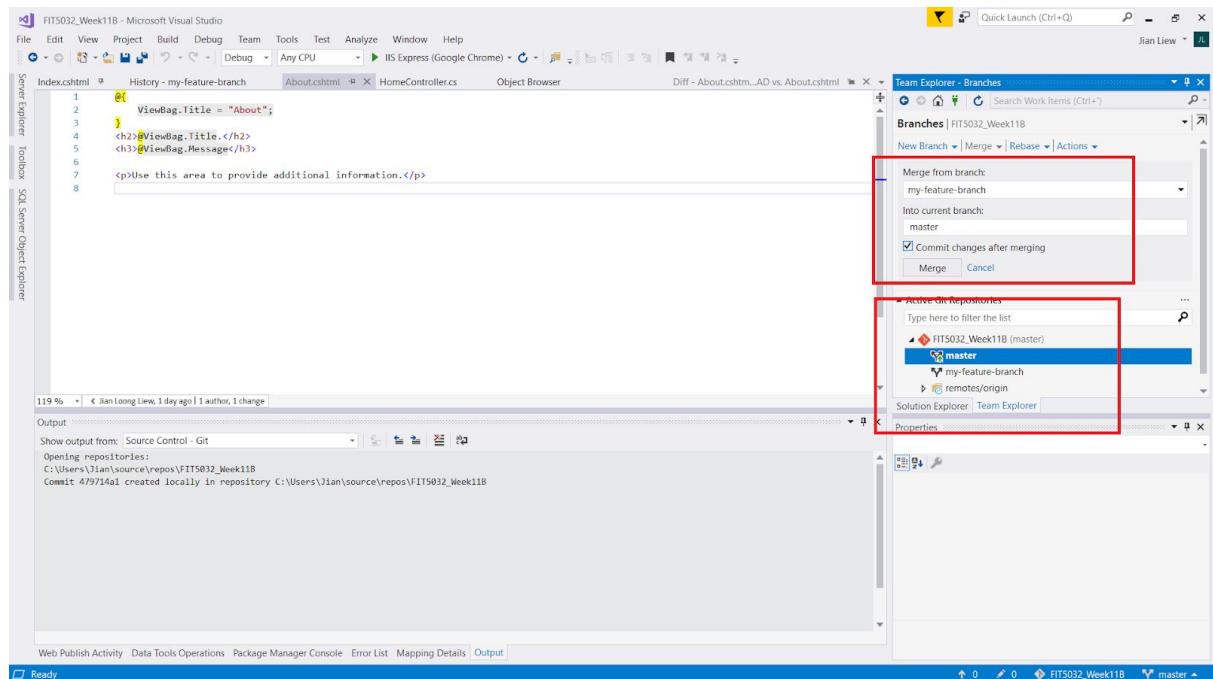


Step 24



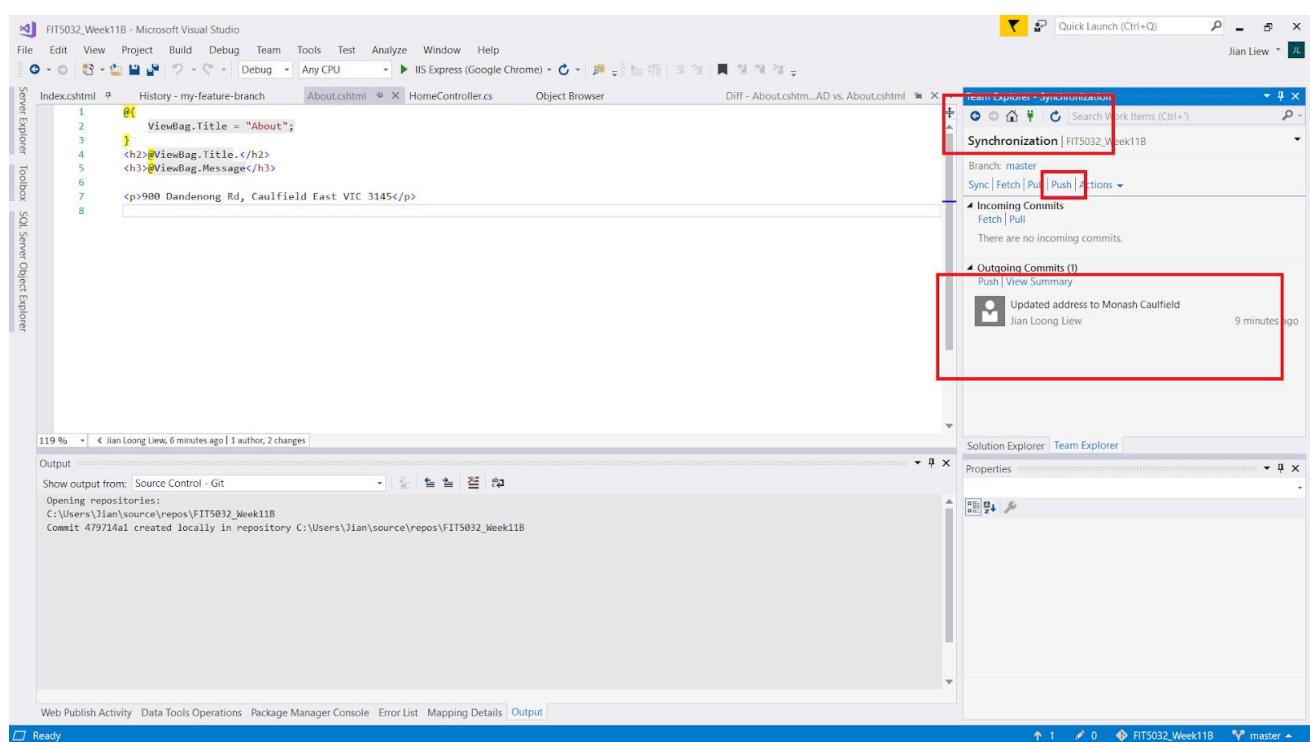
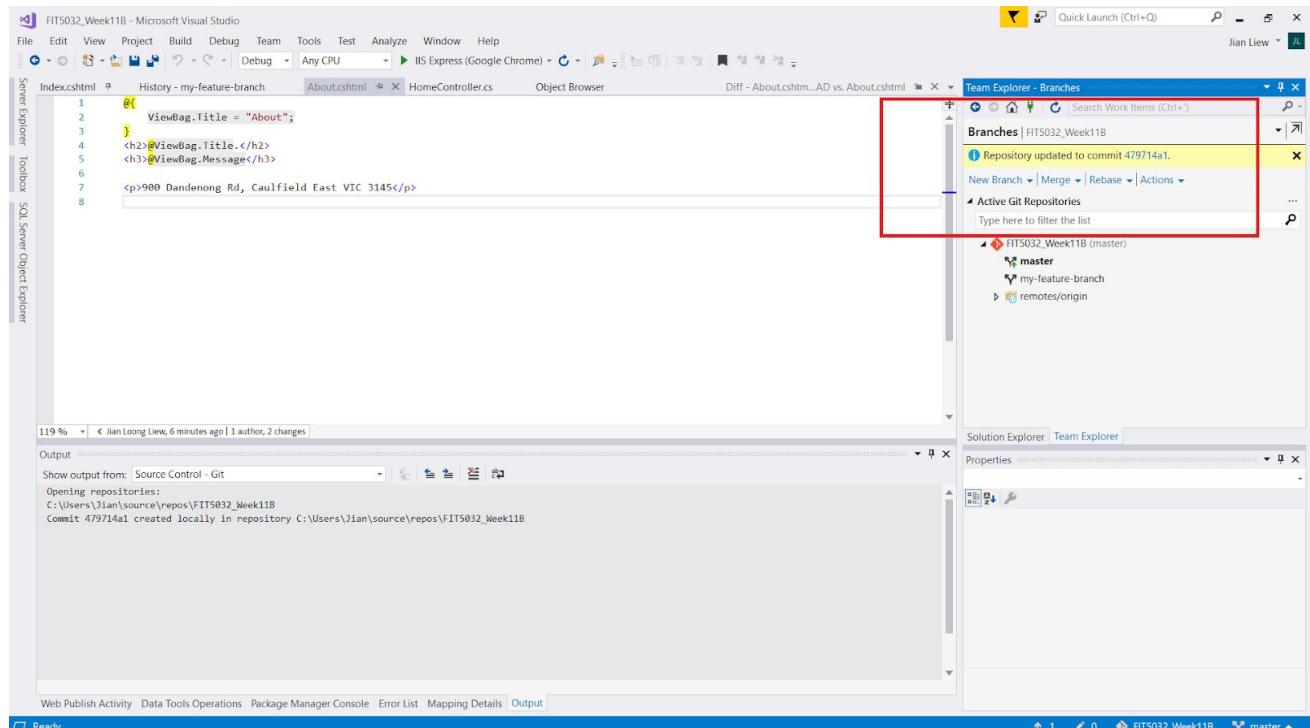
We will now go to the branches, and merge our feature branch into the master. **The reason we are doing it this way is to prevent any merge conflicts. (If our local master is up to date with the remote master from earlier and we merge, and later when we push into the remote, it will not have any merge conflicts)**

Remember that we are merging from the branch to the master. You might need to switch your active repository.



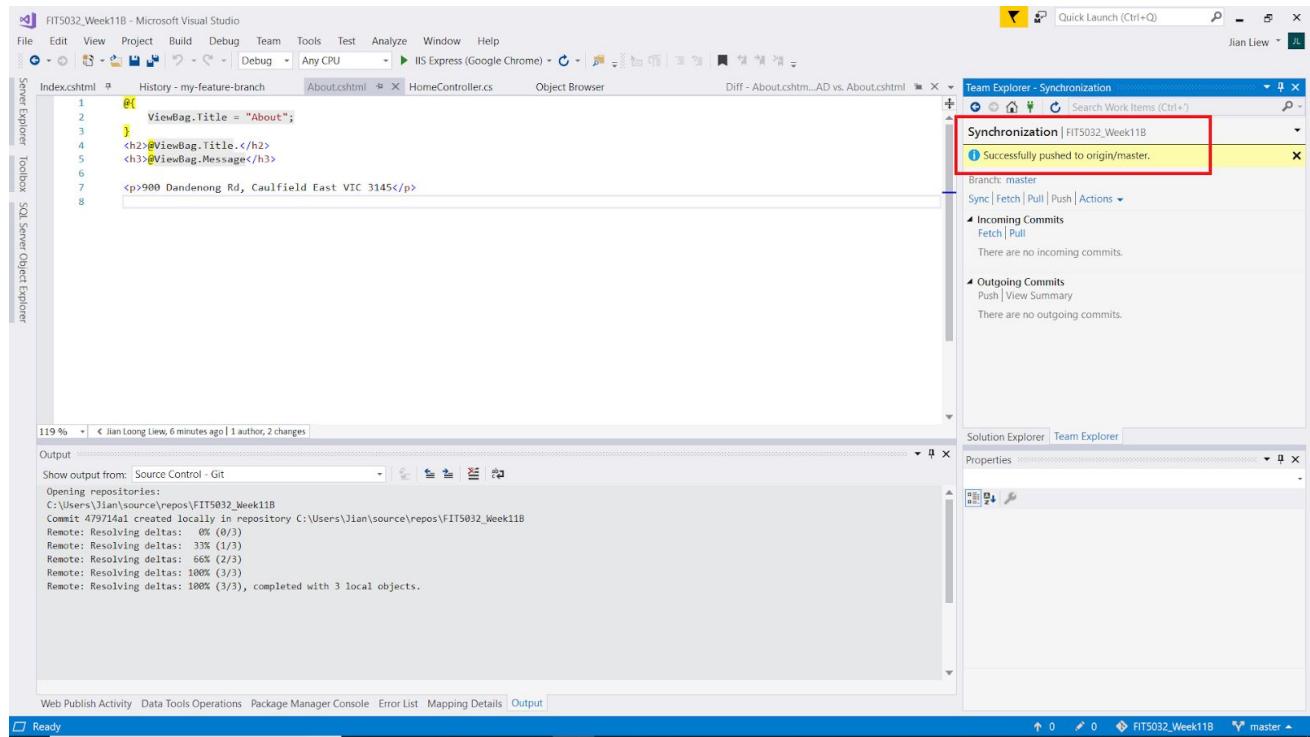
Step 25

After you have done this, you will need to push the master into the remote master. Shown below.
This will trigger a build on Azure as our service application was set up to do so.

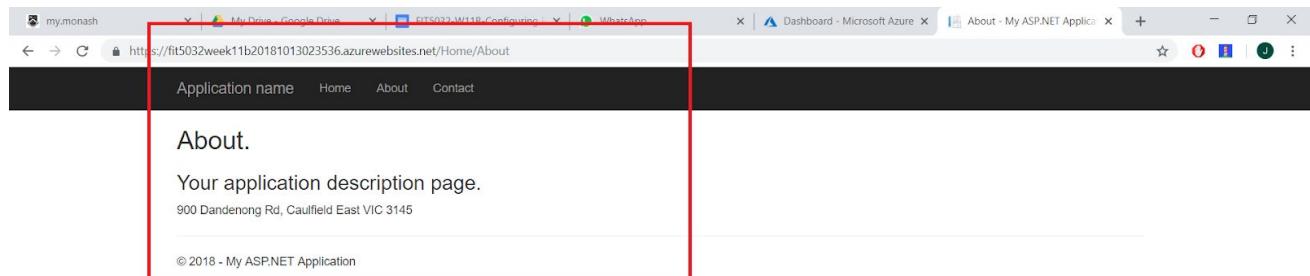


Step 26

Due to our workflow, it is almost for certain this will be successful.



If you now go to your **live website**, you will notice that the changes has been made. This is what we call Continuous Deployment.



Explanation

This supplementary material aims to showcase how you can use a simple deployment process. (When you do a push to your remote repository, it is automatically build & deployed).

Normally it is considered to be good practice to work on branches and do merges to the master branch. It is also possible to create various branches for testing etc. This becomes more evident when the teams becomes bigger and more people work on the project.

It is also important to understand how continuous deployment works. However, at the end of the day each company would have their own set of practices and they would differ from each other.

Conclusion

Upon the completion of this supplementary tutorial, you would have gained an understanding of

- Simple version handling using Visual Studio with Github
- Simple usage of continuous deployment.

It is almost a must to know how continuous deployment process work for any modern web application.

For MIT students, it is highly recommended to learn how to use the command line first so that you understand the process in more details.