MONASH University

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 high 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

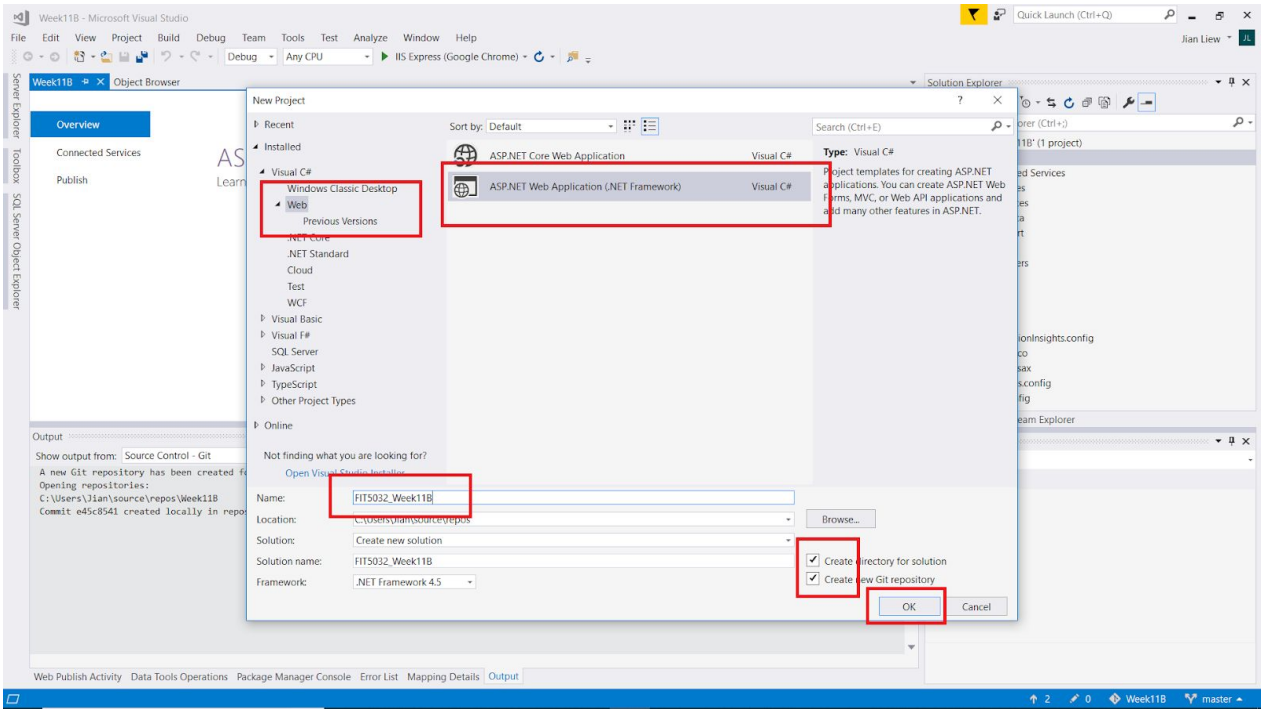
DoubtFire Submission

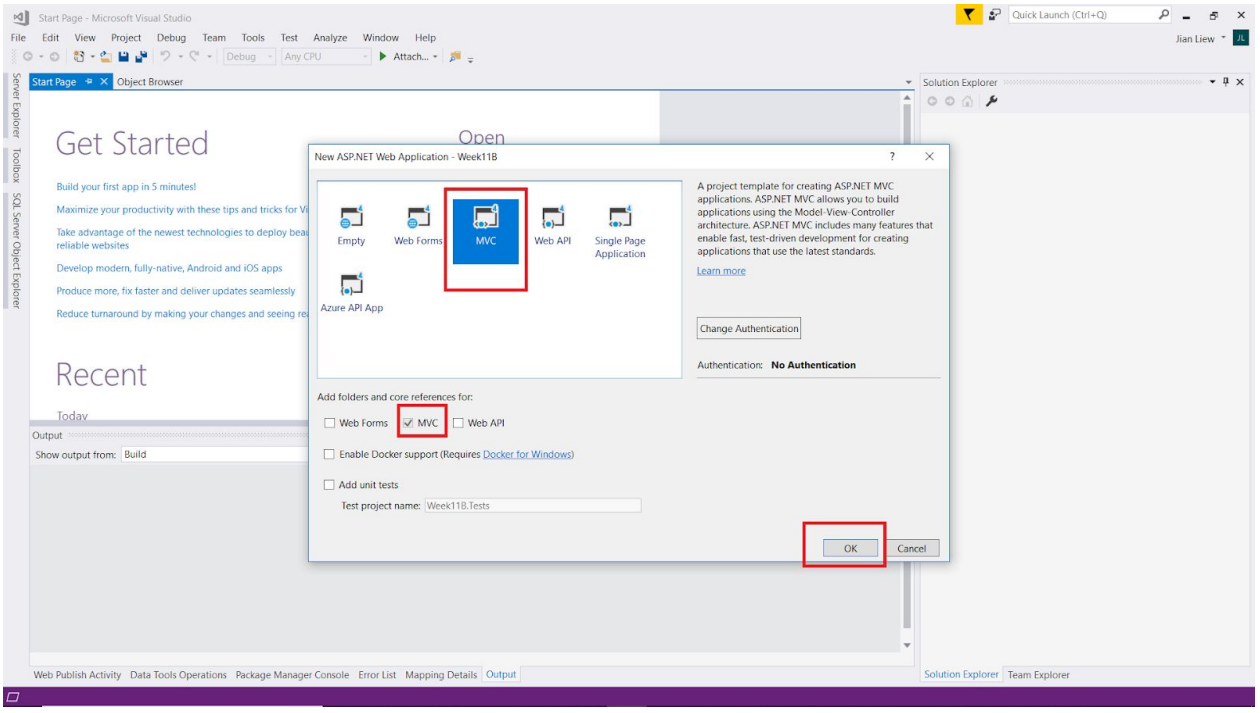
- **None**

Step 1

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

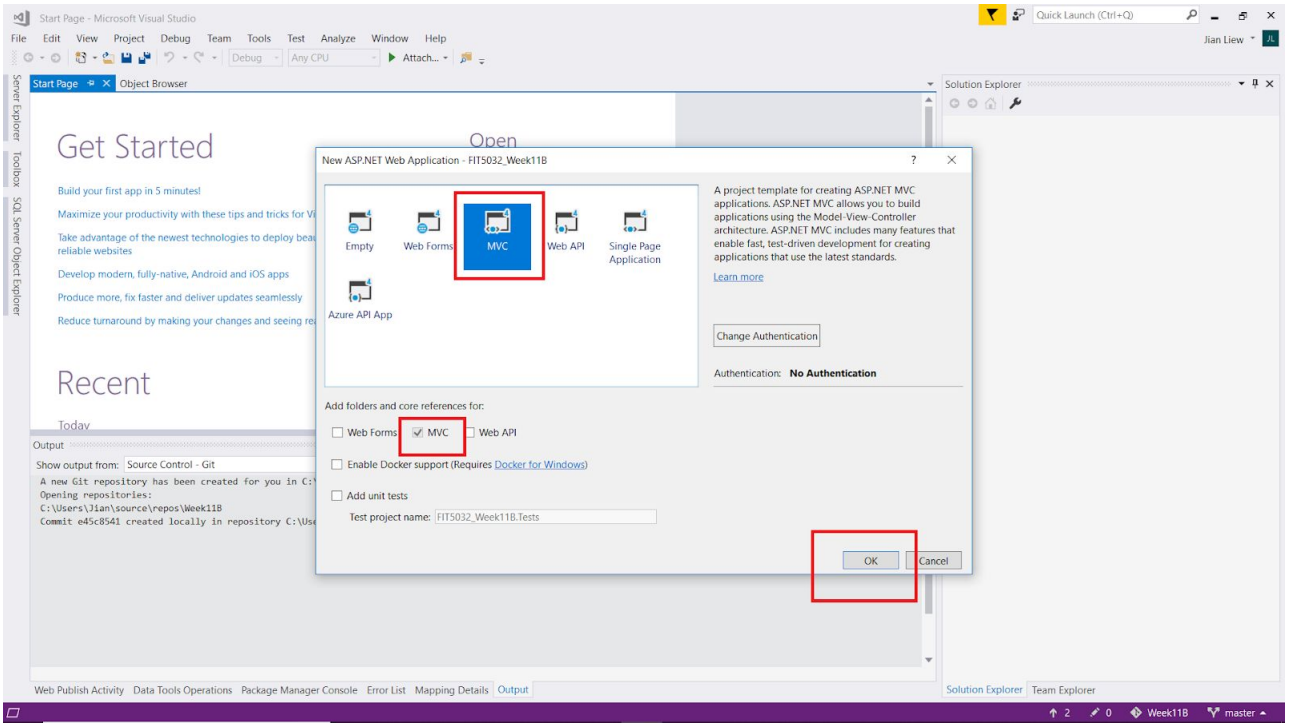
This time, please have "Create a new Git repository selected"





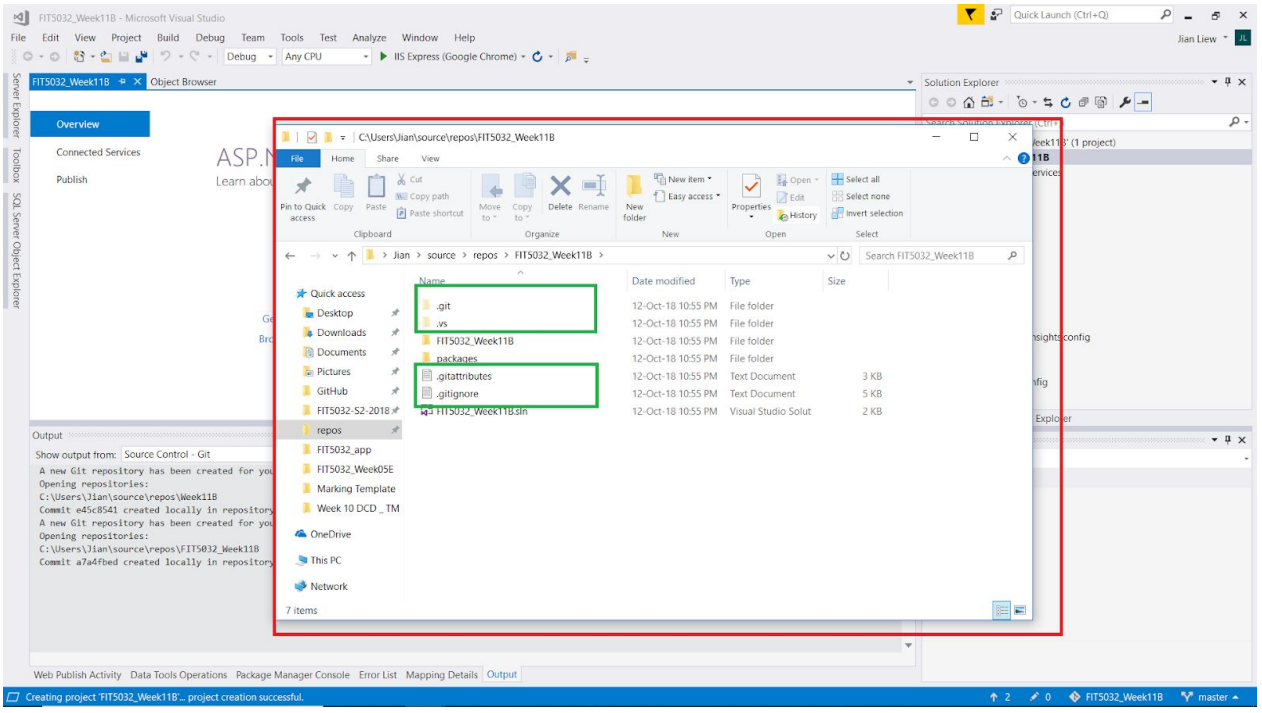
Most modern applications will often use Docker + Kubernetes. However, for our application we do not need to use it. (It is almost needed to know what Docker is these days)

Step 2



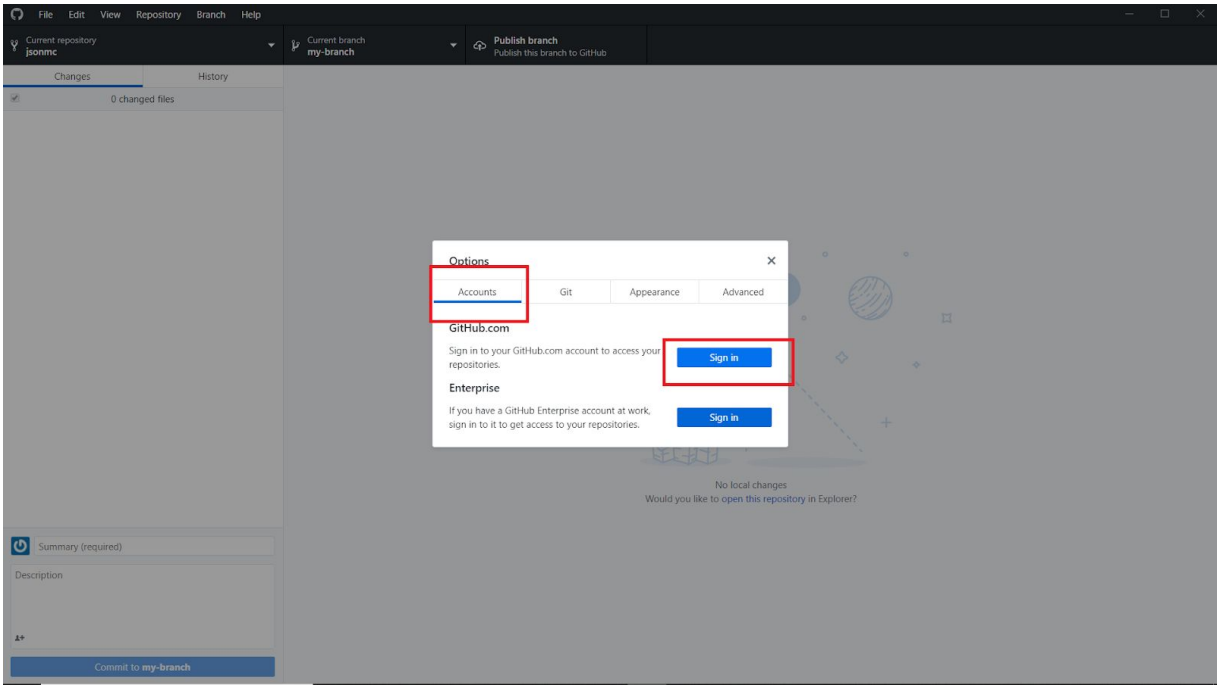
Step 3

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.

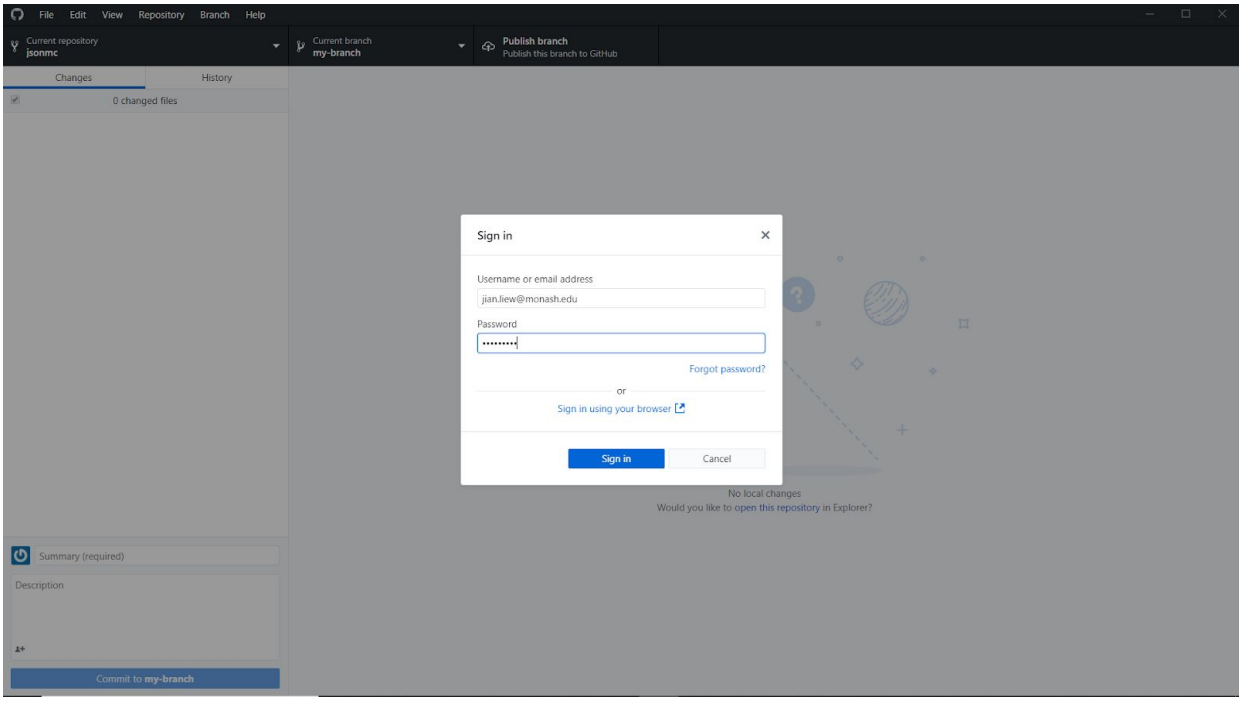


Step 4

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.

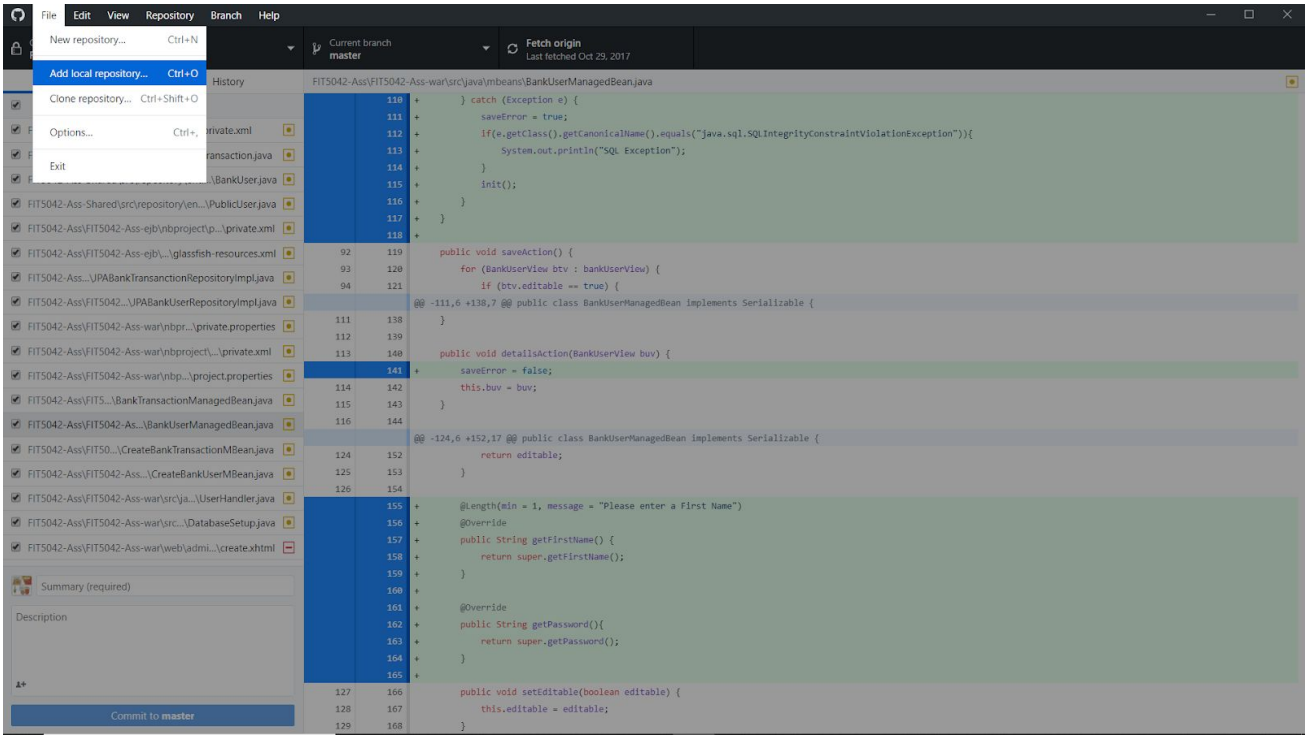


Please note that this is NOT your Monash Account, it is your Github username and password.

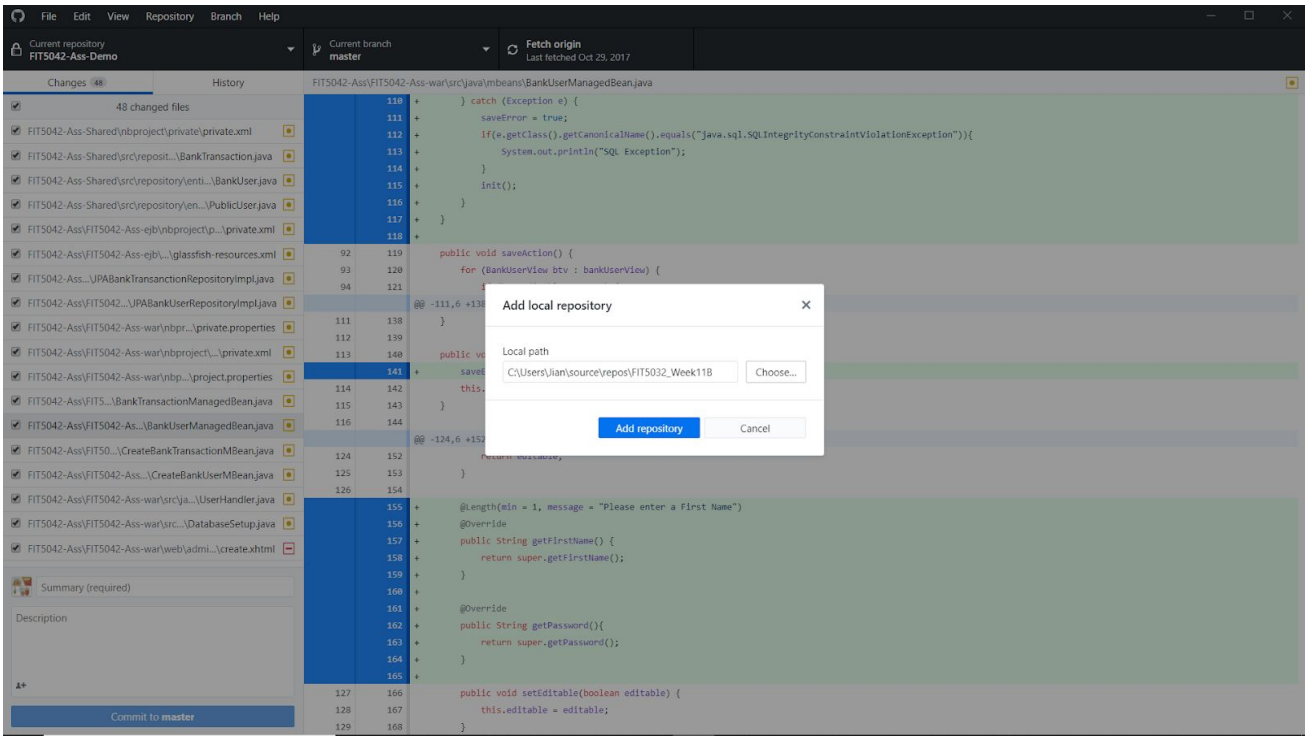


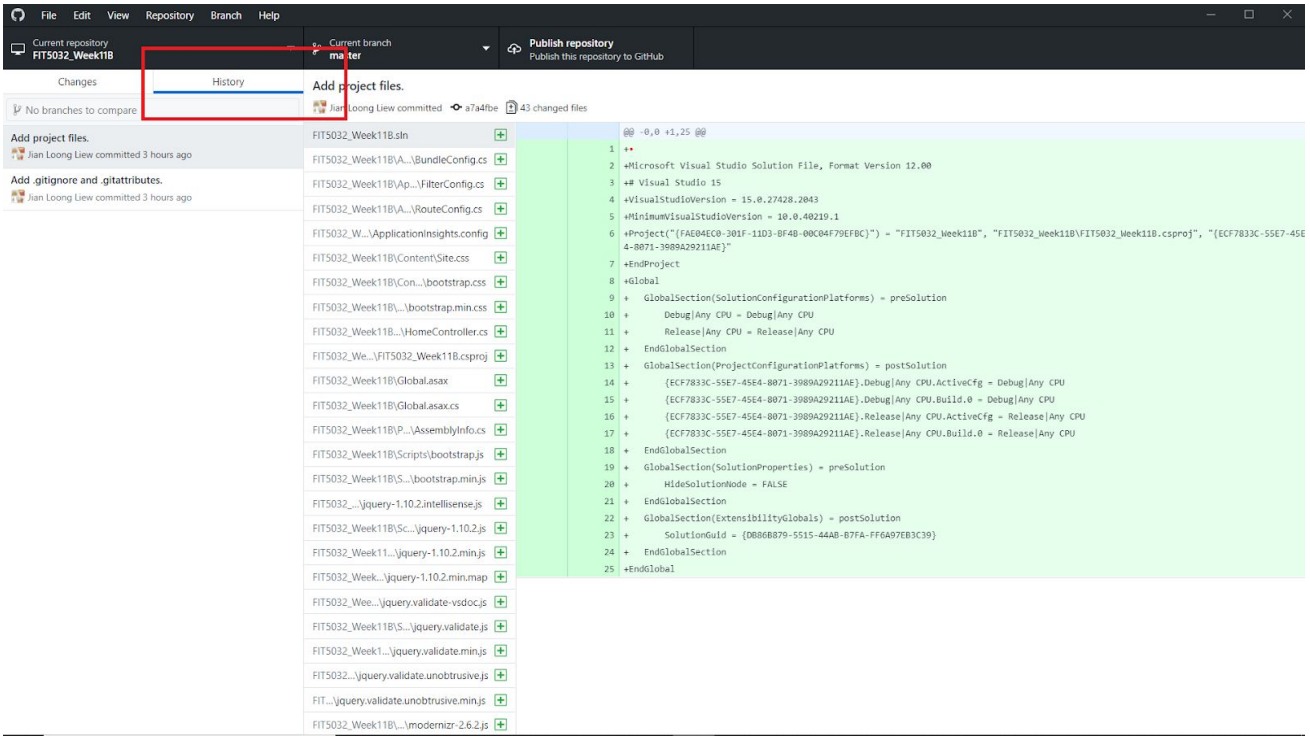
Step 5

Please add the Local Repository created in Step 3.



Step 6

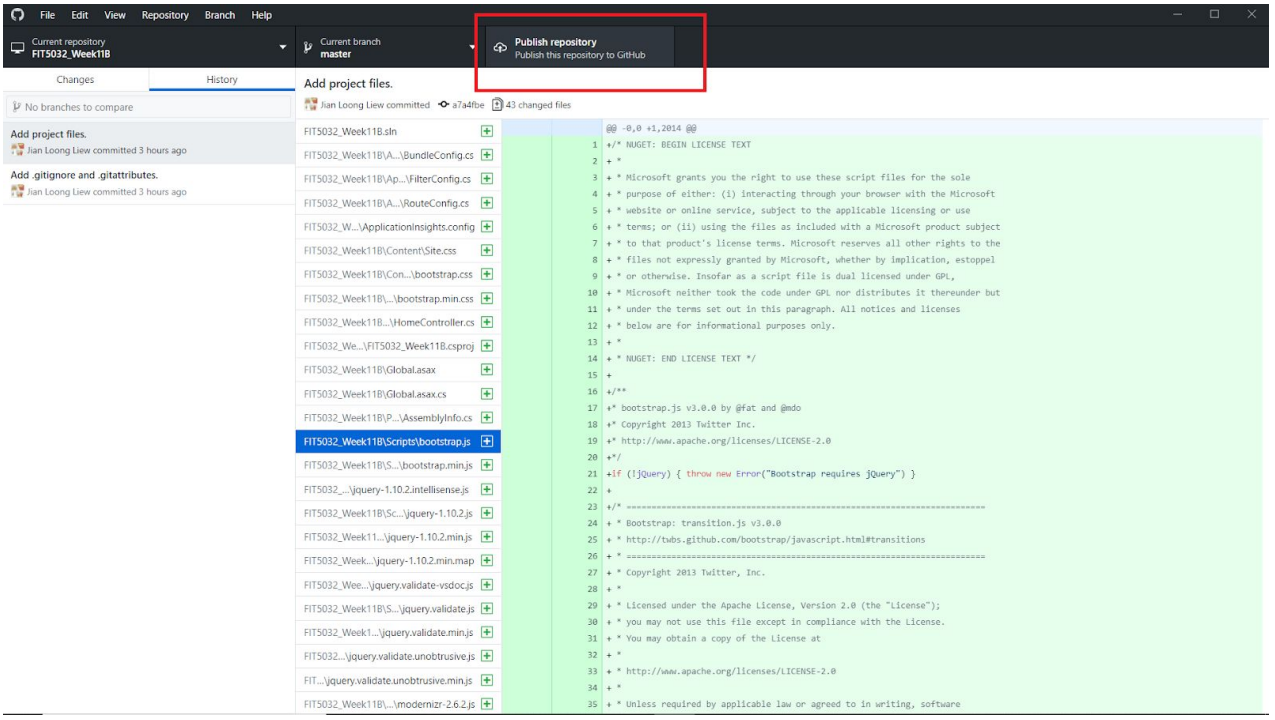




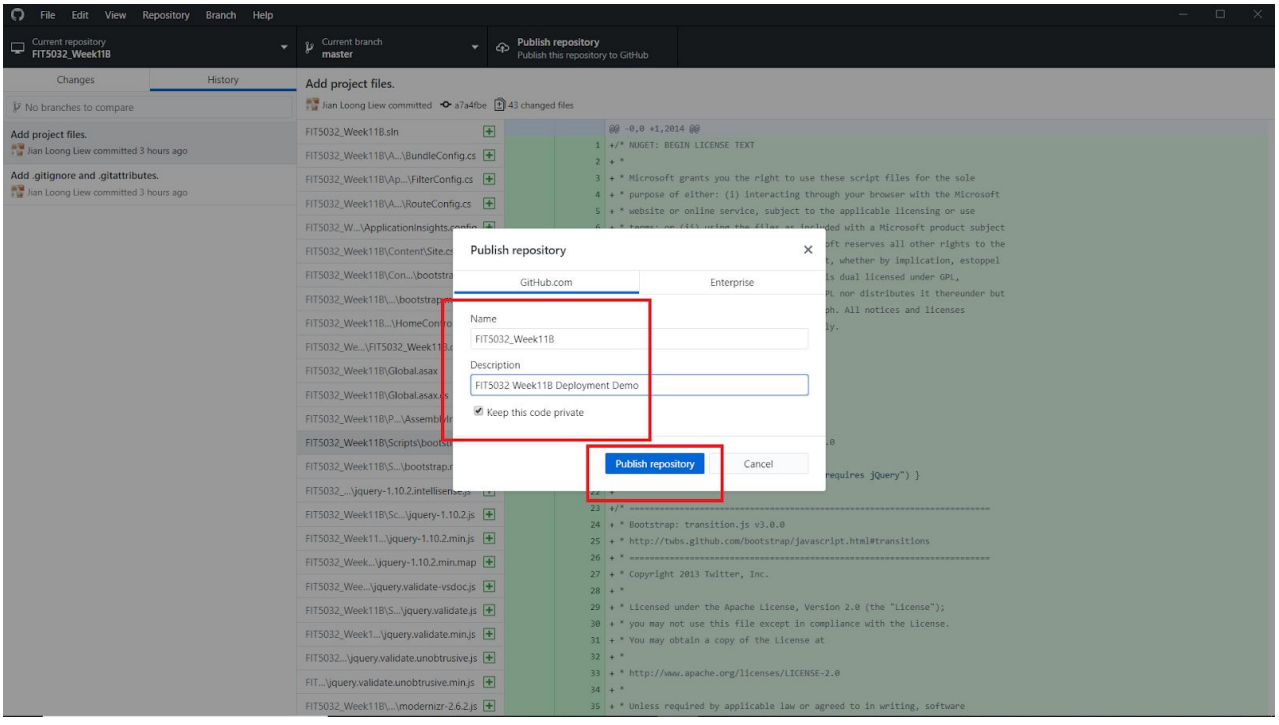
There should be some history if you check it now.

Step 7

I will go ahead and Publish my repository.

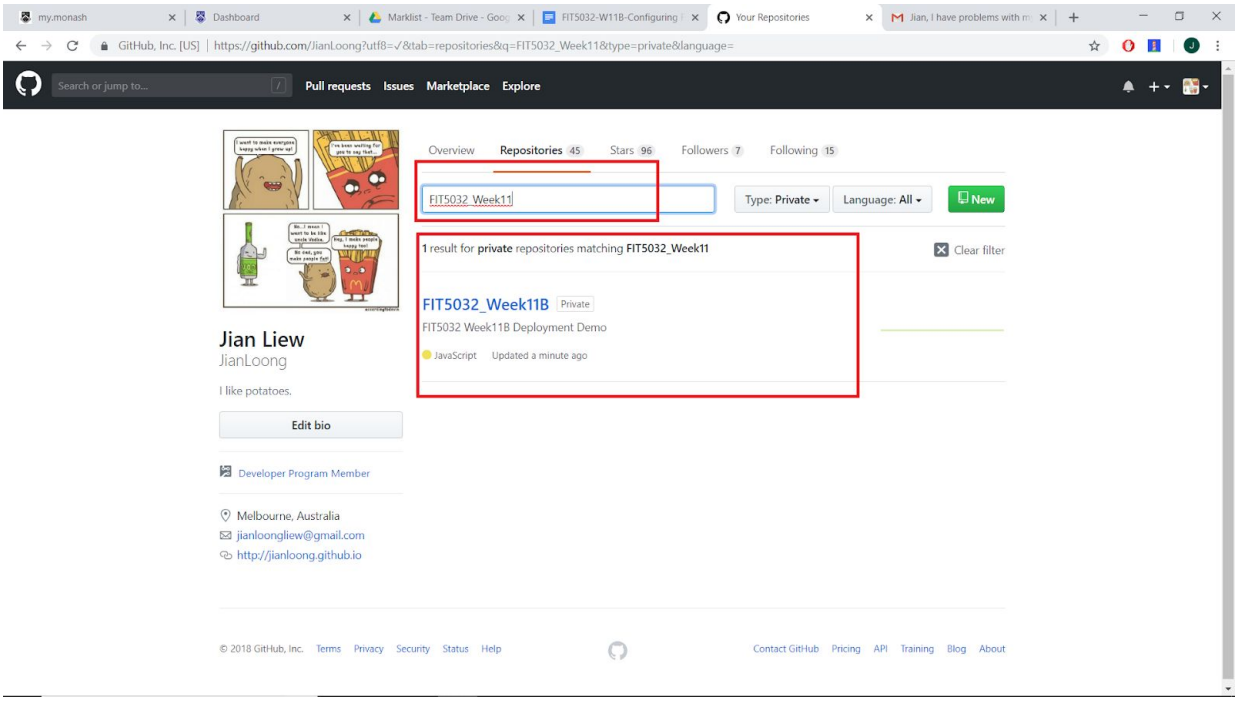


Publishing it means pushing it to the remote repository.



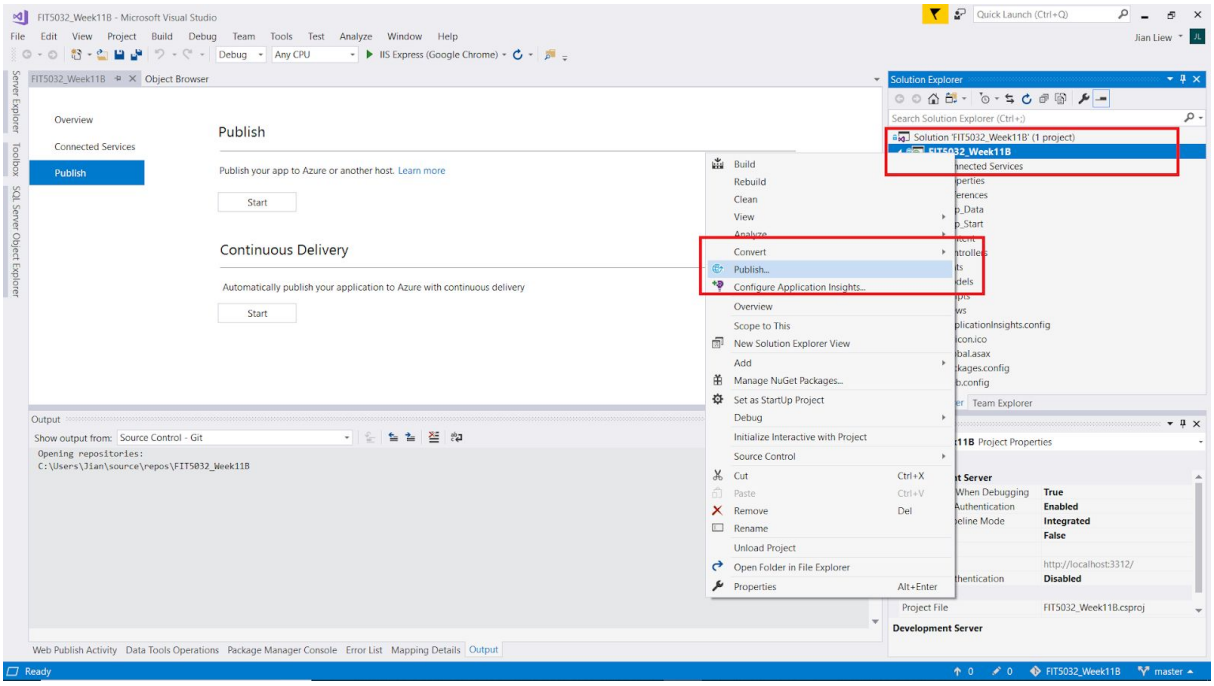
Step 8

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



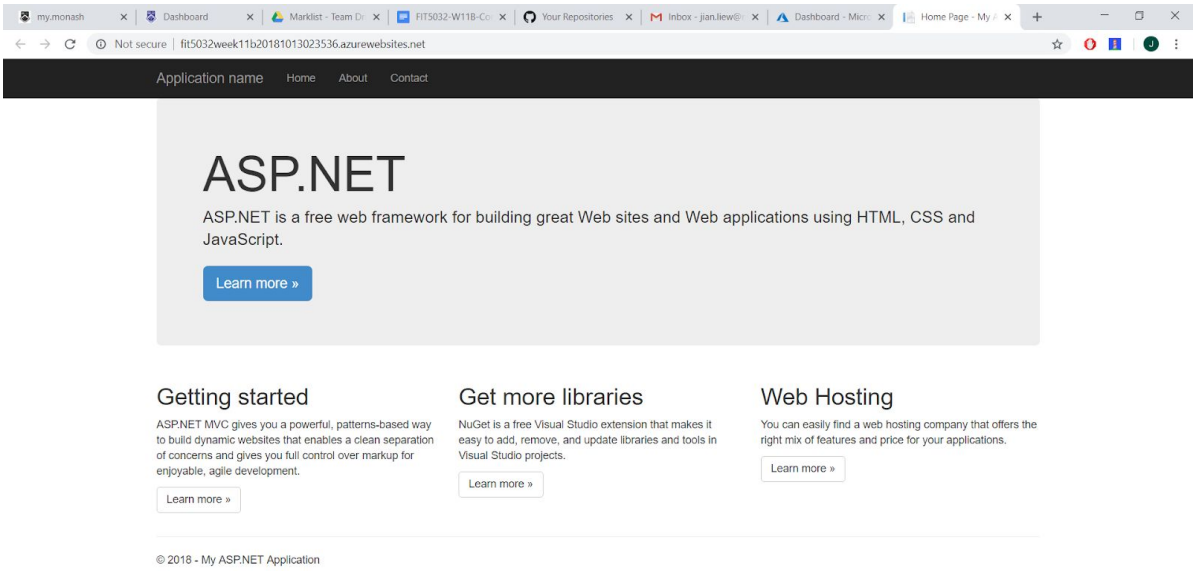
Step 9

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)



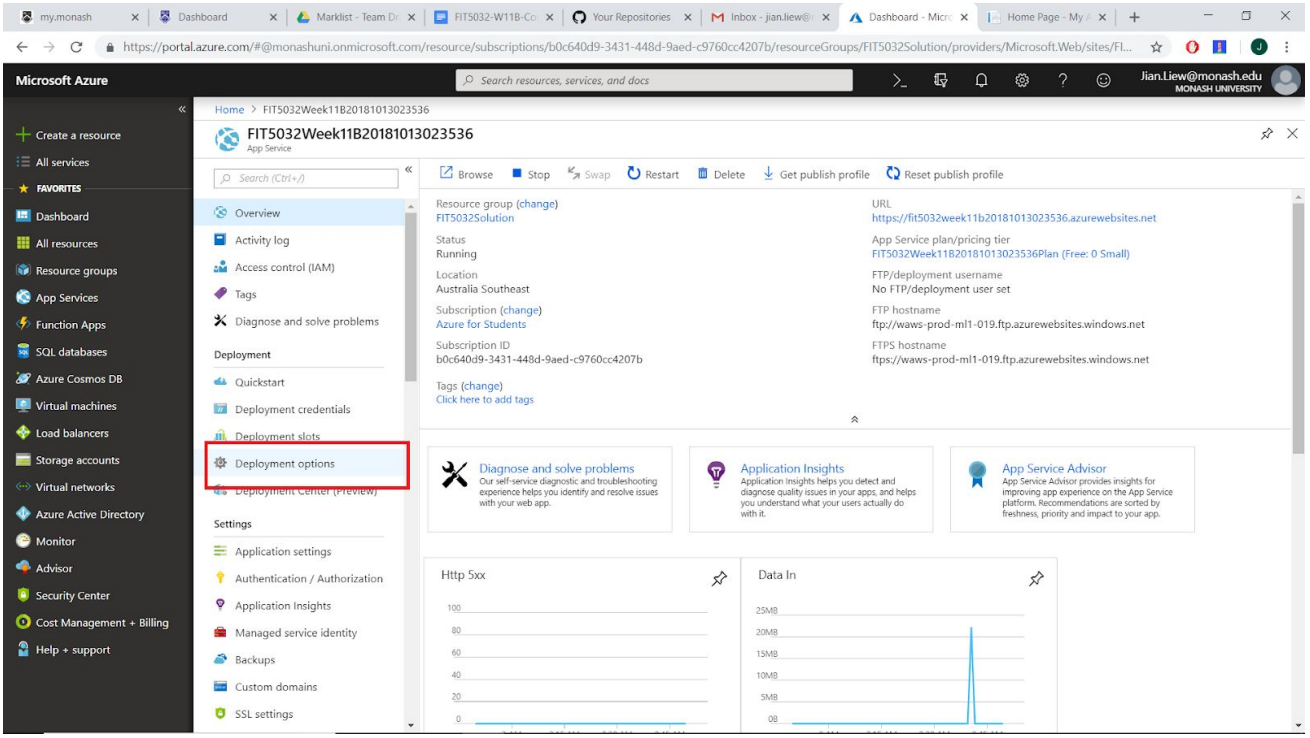
Step 10

You can just automatically deploy it by creating new resources.



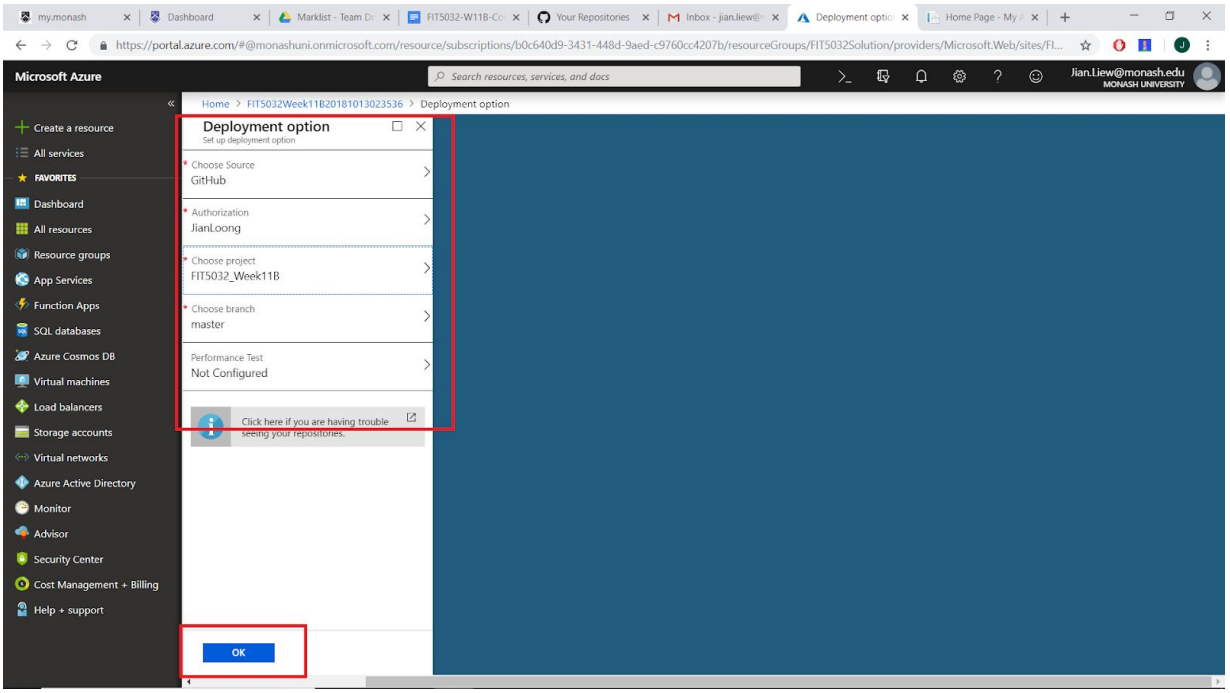
Step 11

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



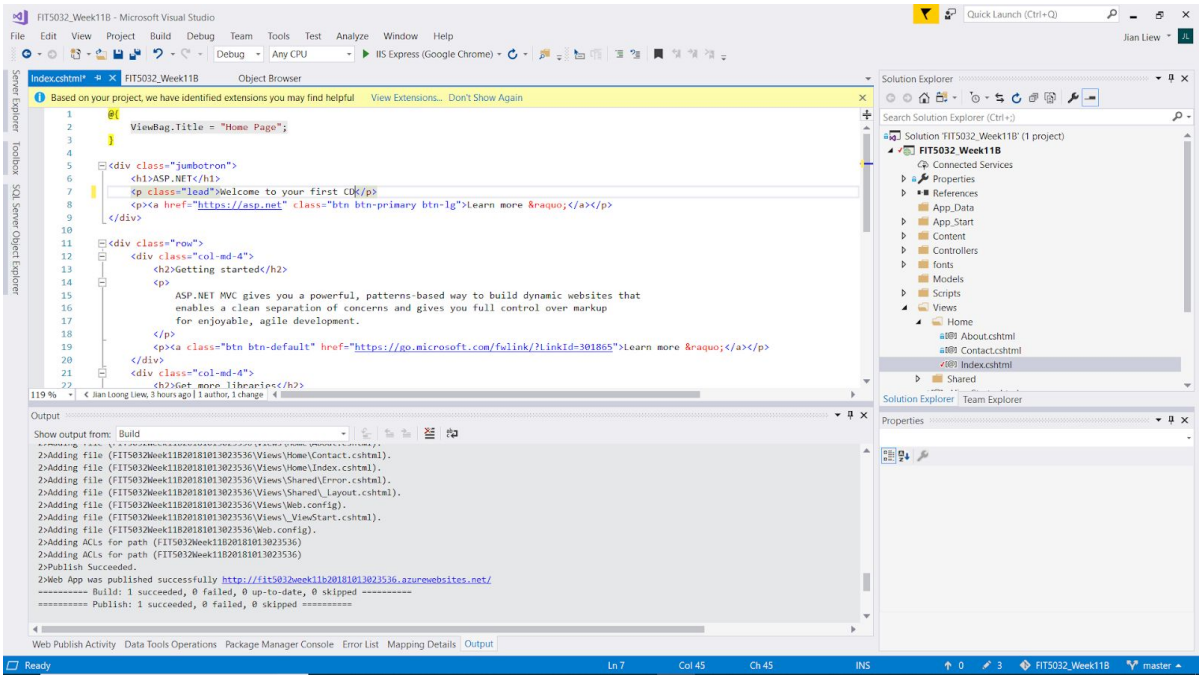
Step 12

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



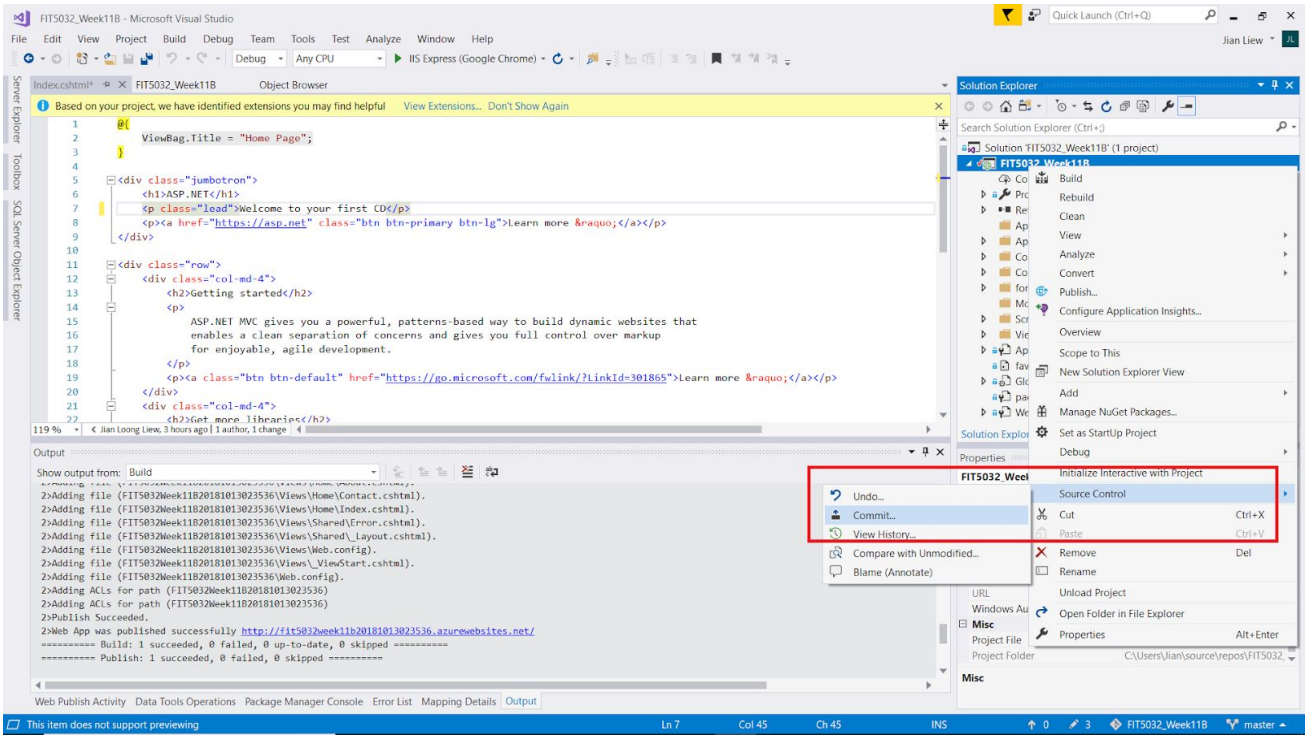
Step 13

We will now just change the information in the jumbotron.

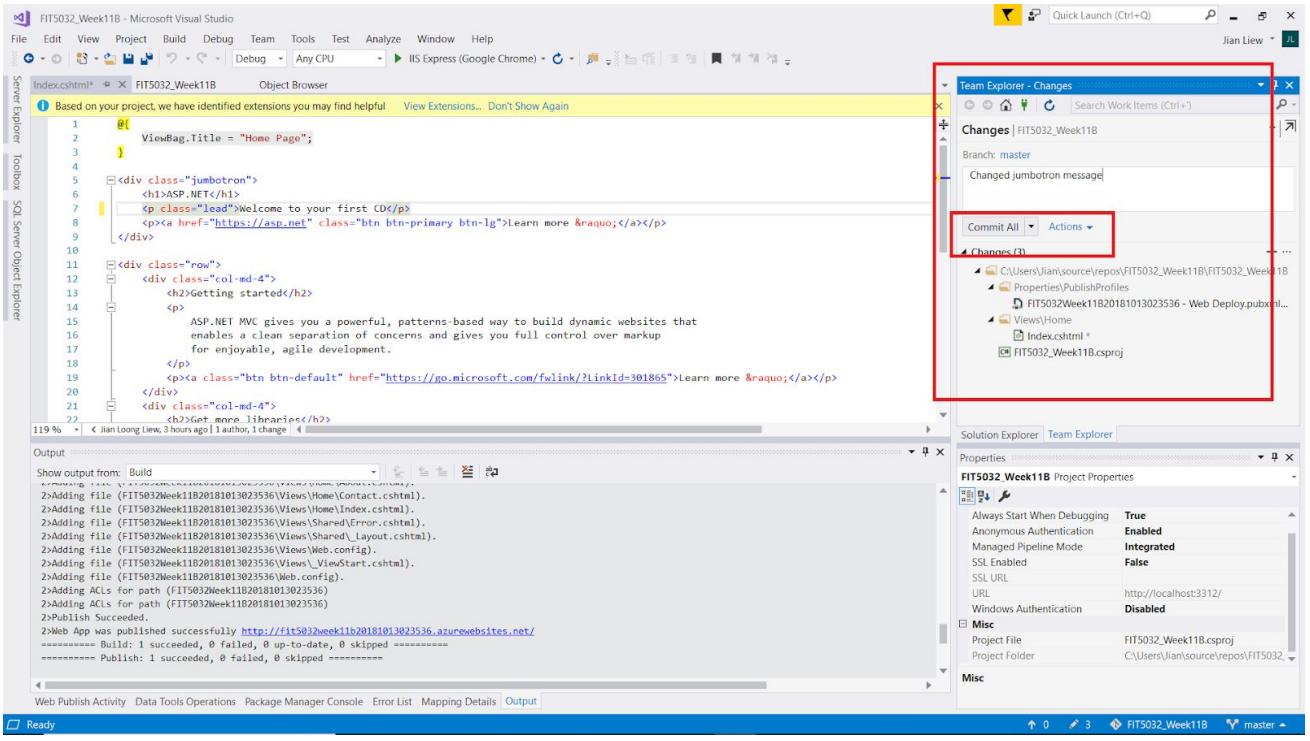


Step 14

Now, use the IDE to commit your changes.

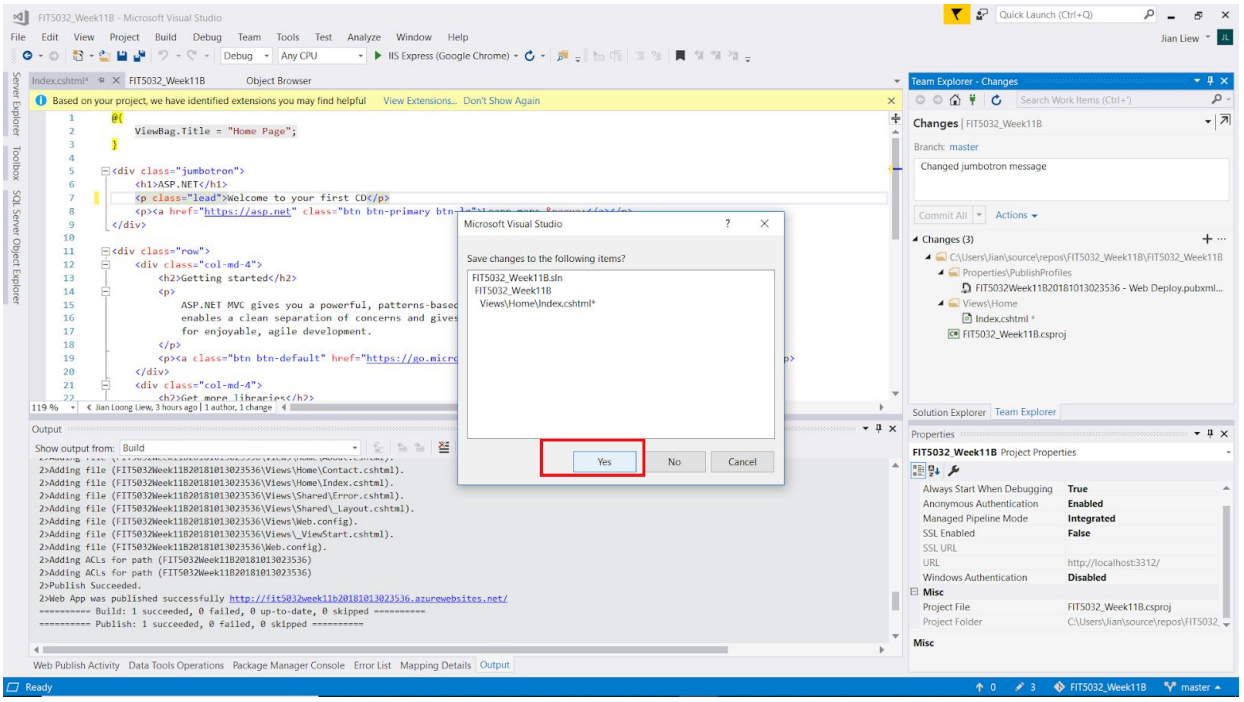


Step 15



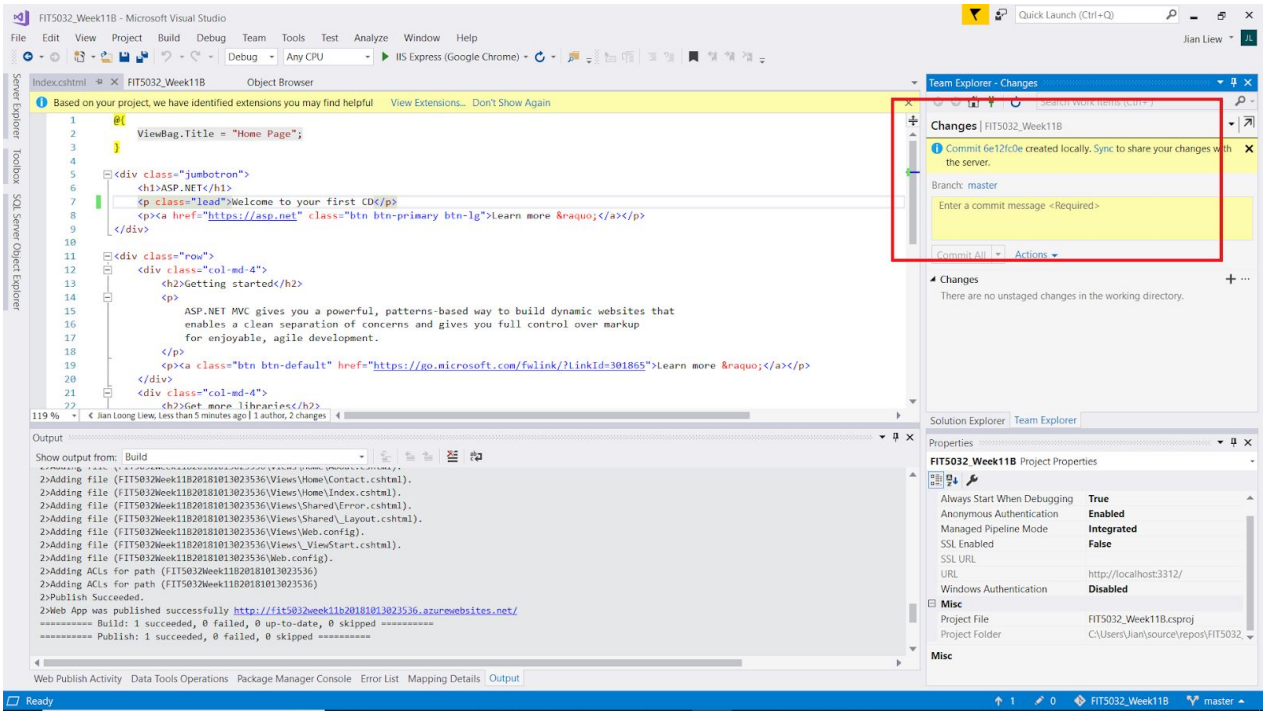
Step 16

If you see this, just click "Yes"

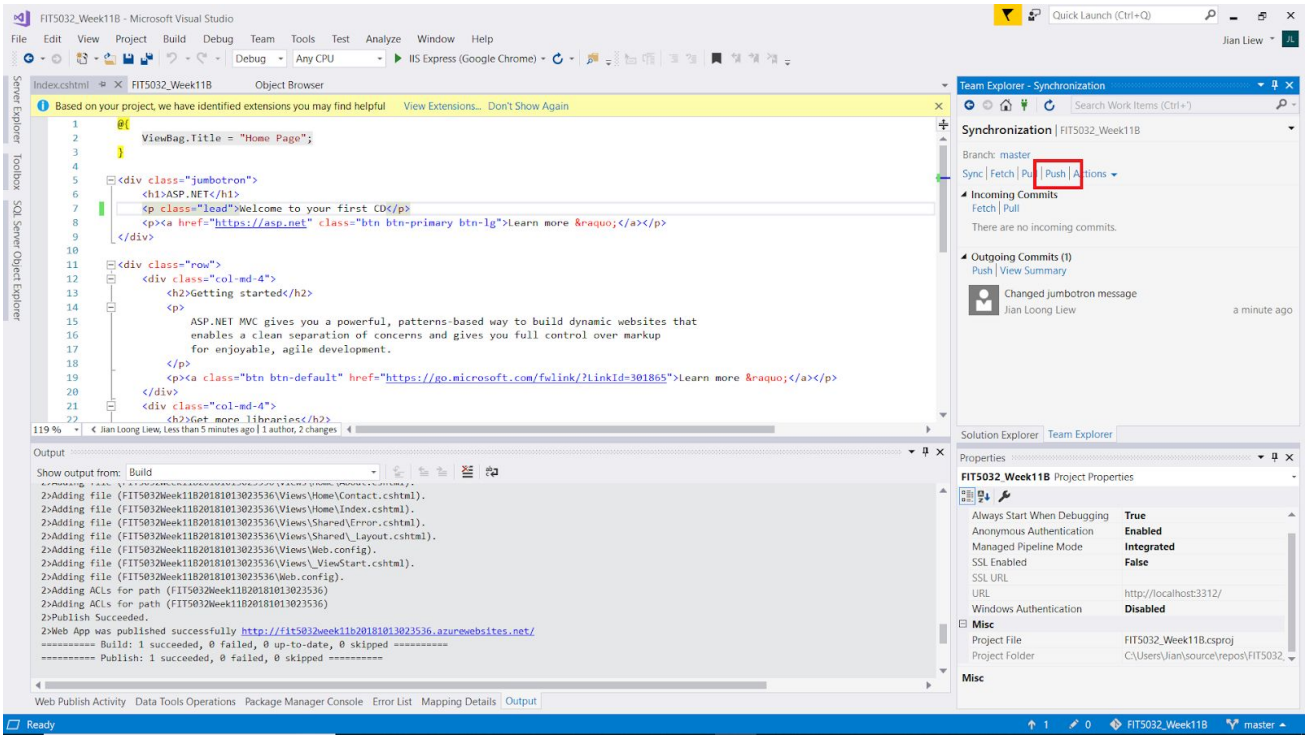


Step 17

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

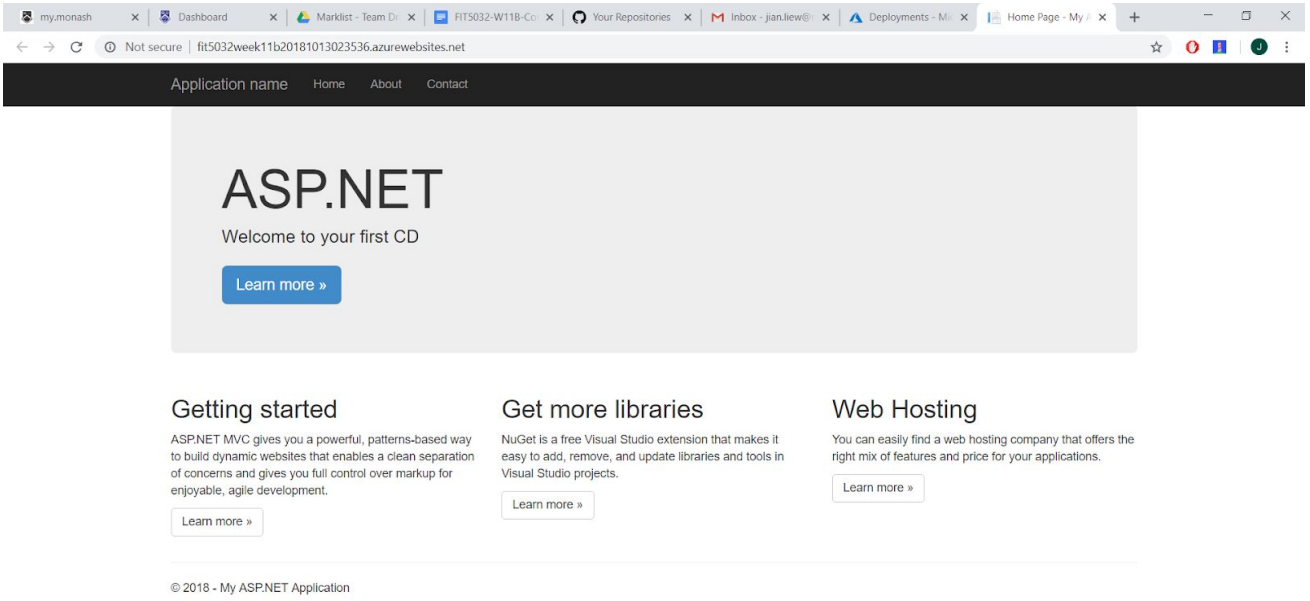


Step 18



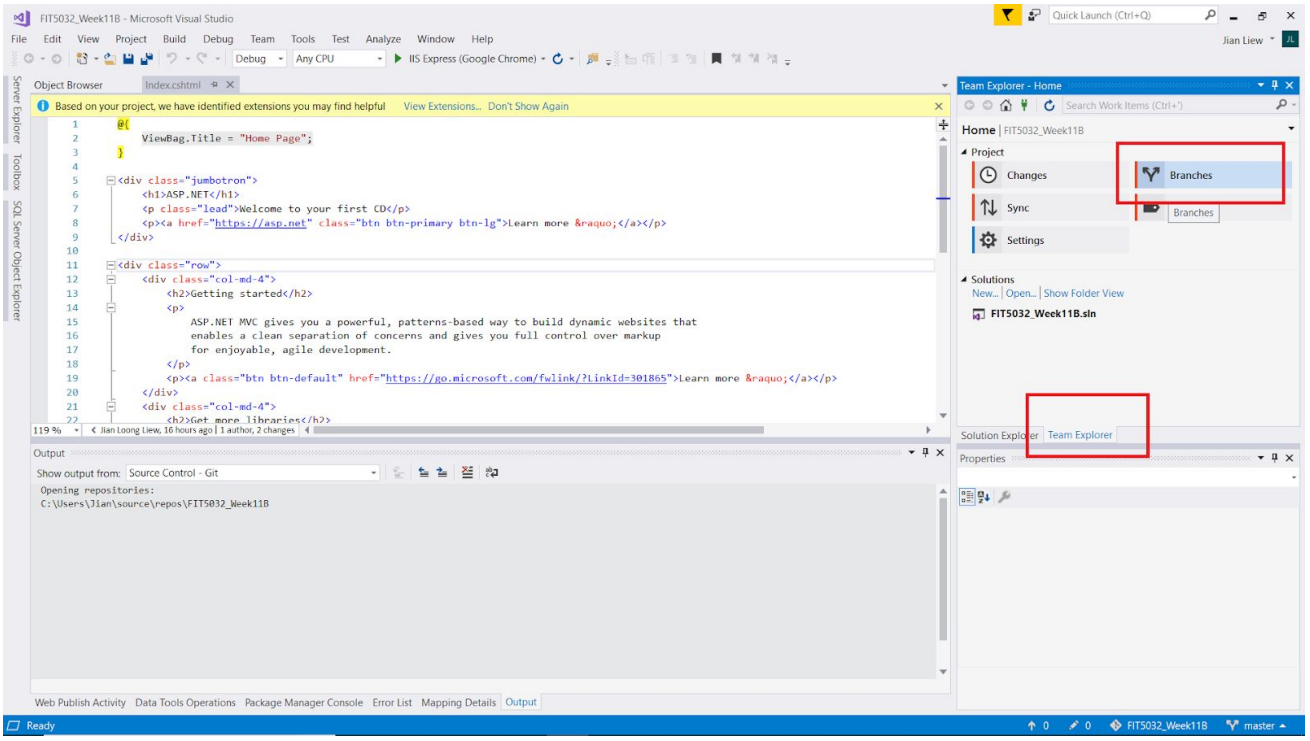
Step 19

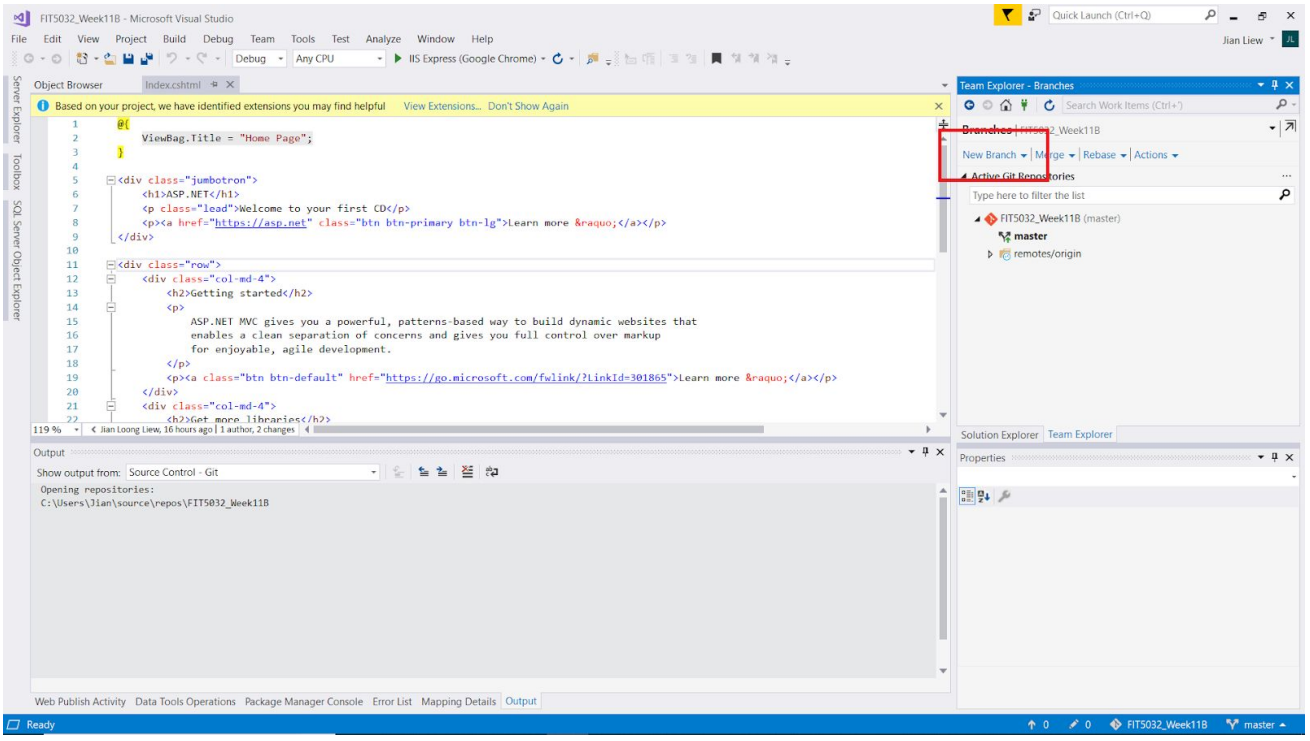
Once, you have completed that, upon a "Push", the continuous deployment will automatically be triggered. So, when you go back to the website, you will notice it has been changed.



Step 20

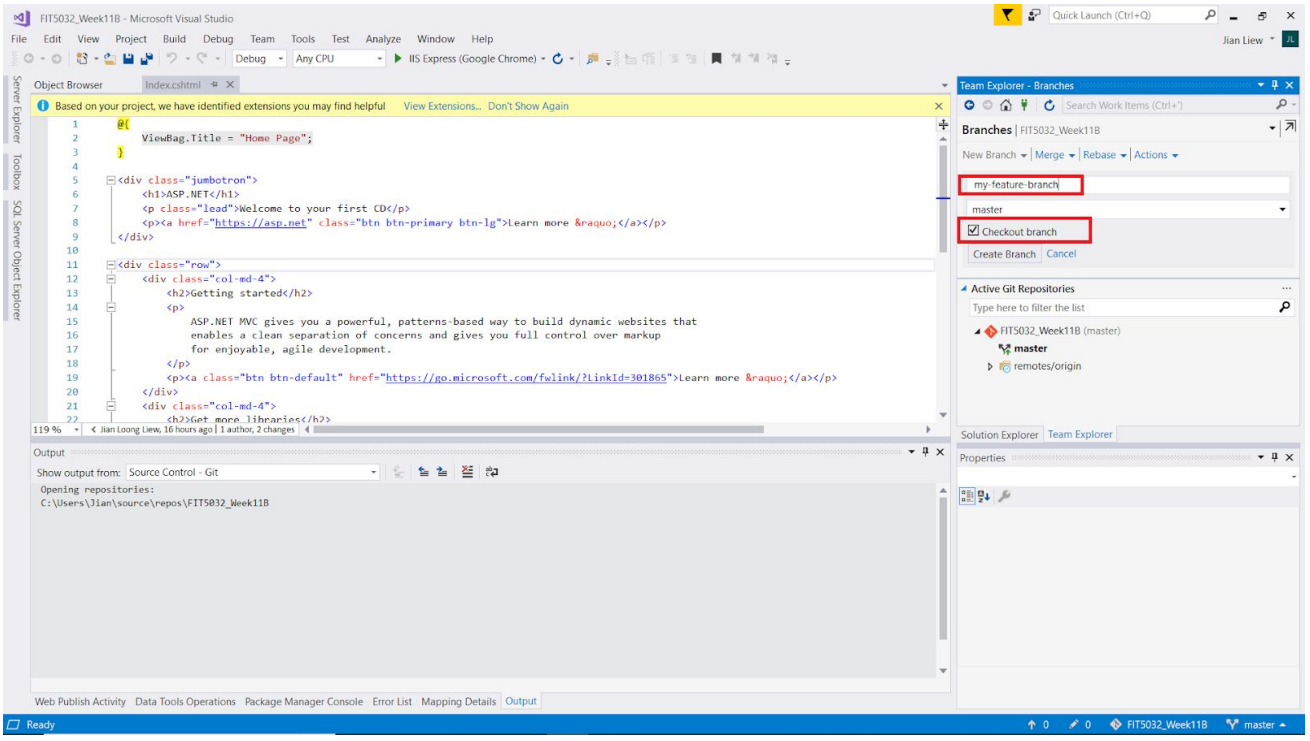
You will now learn a git feature known as branching. Branching is quite a common technique. Often times when multiple developers work together, it is much easier to manage the "master" source codes via branches. The reason for this, is because merge conflicts are easier to be handled this way.

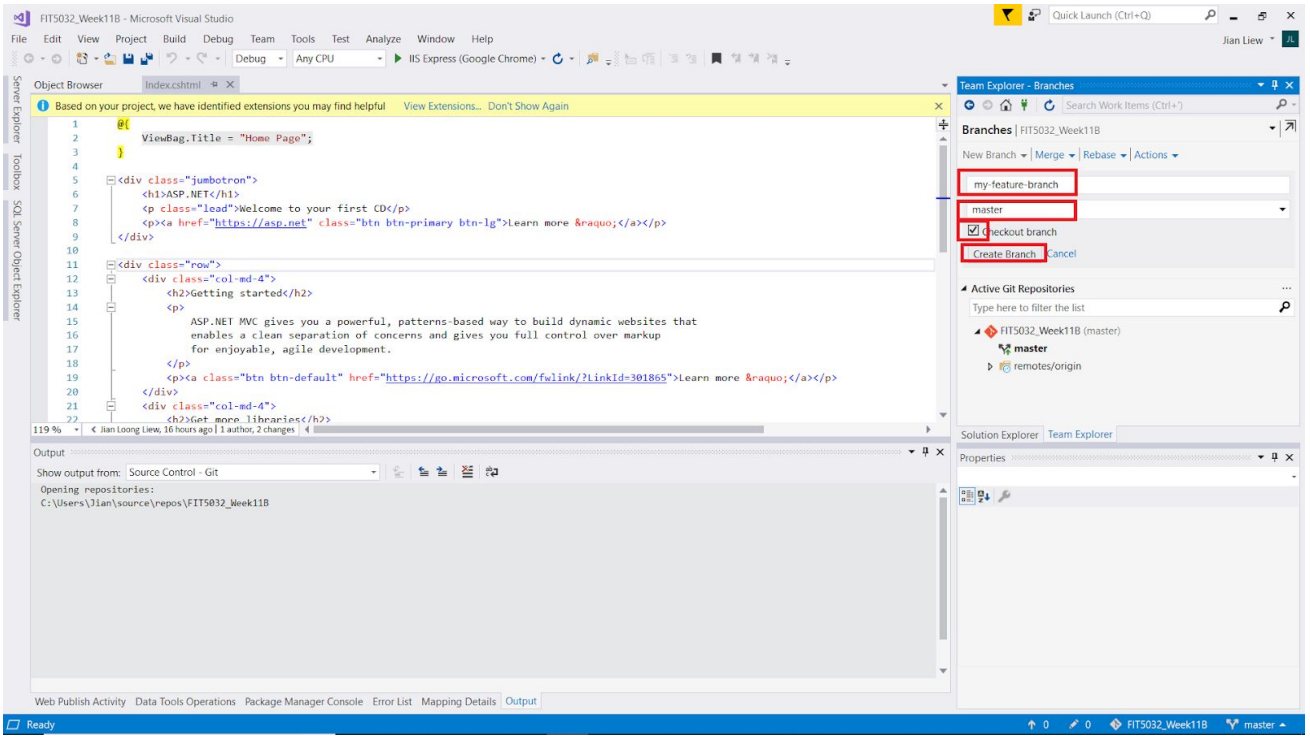




Step 21

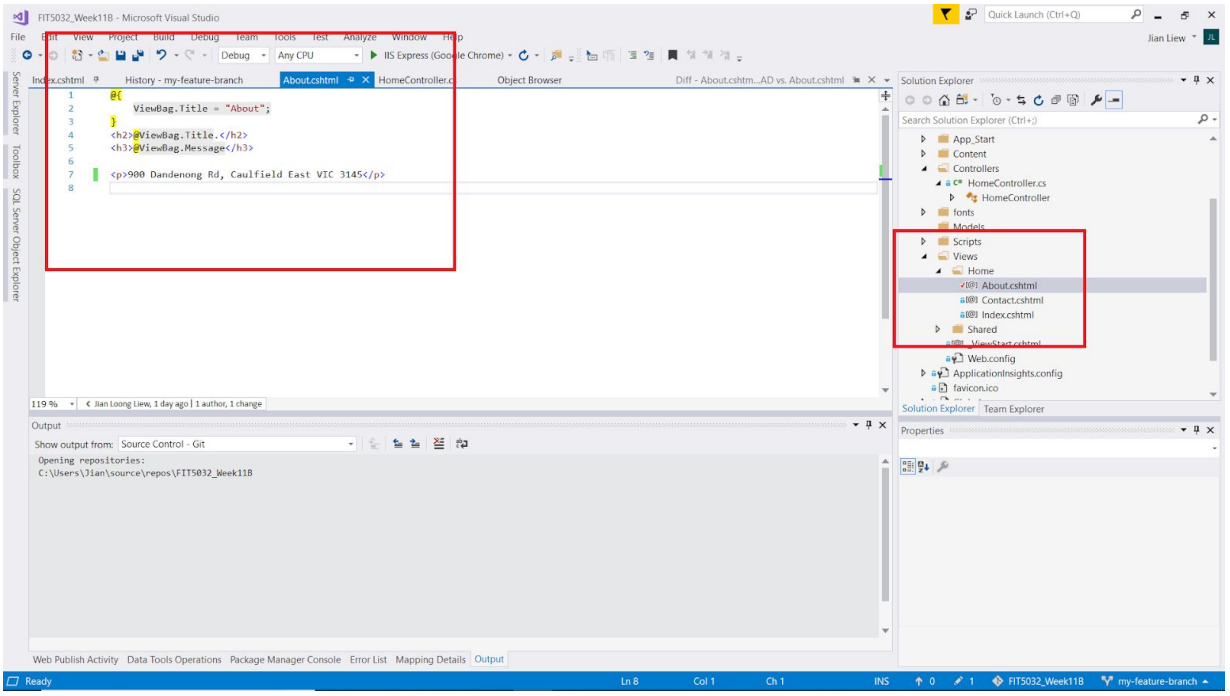
The name of your branch does not matter, as it only exist in the local repository for now. Proceed to create a branch and check it out. **Checking it out means that your current workspace will be switched to the branch, and you are no longer working on the master.**





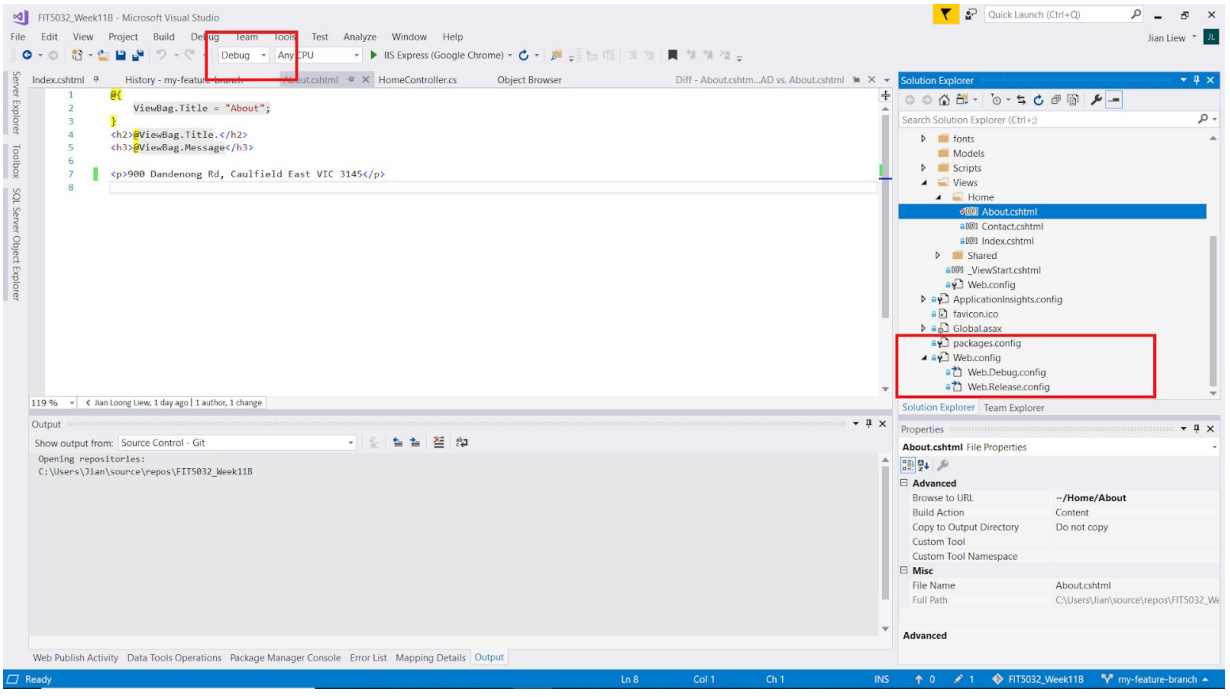
Step 21

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

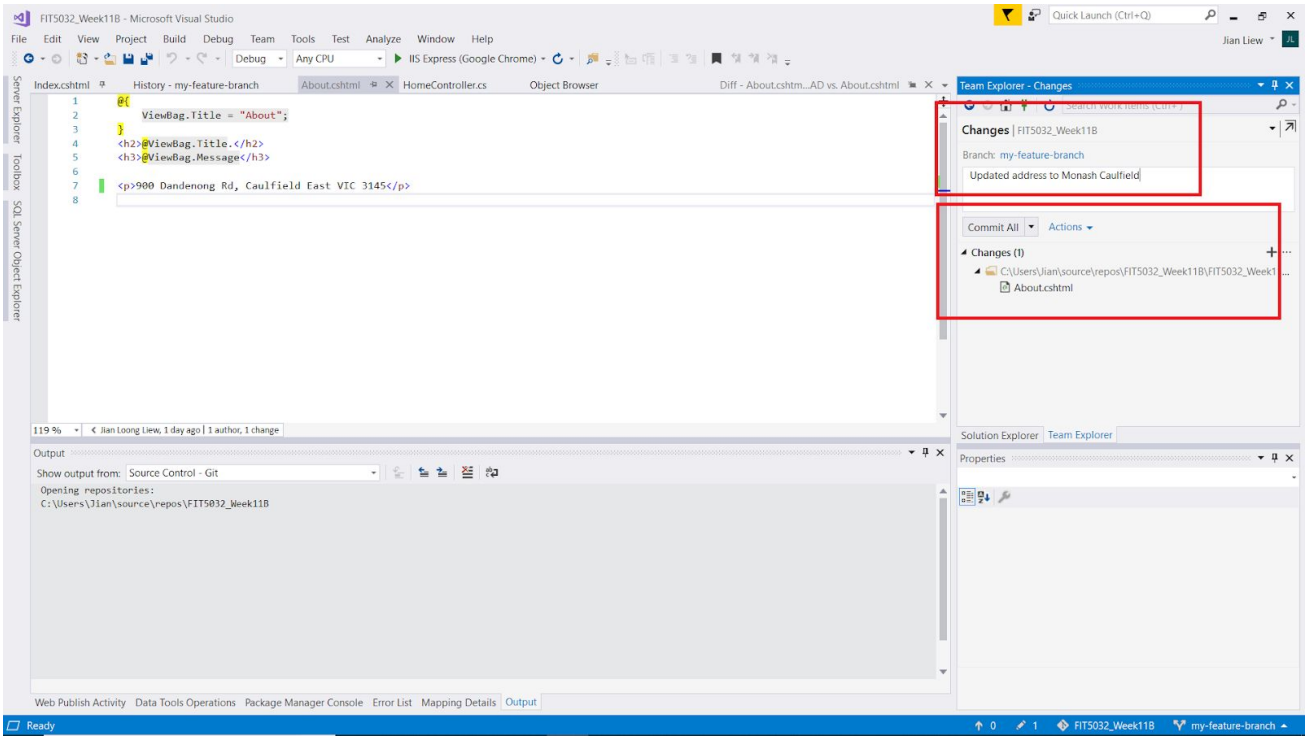
Since we have made changes and introduced a feature on our branch. We will now, merge it back into the main branch (master). There is a concept known as fast forward merging. (But we are not doing that)

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.

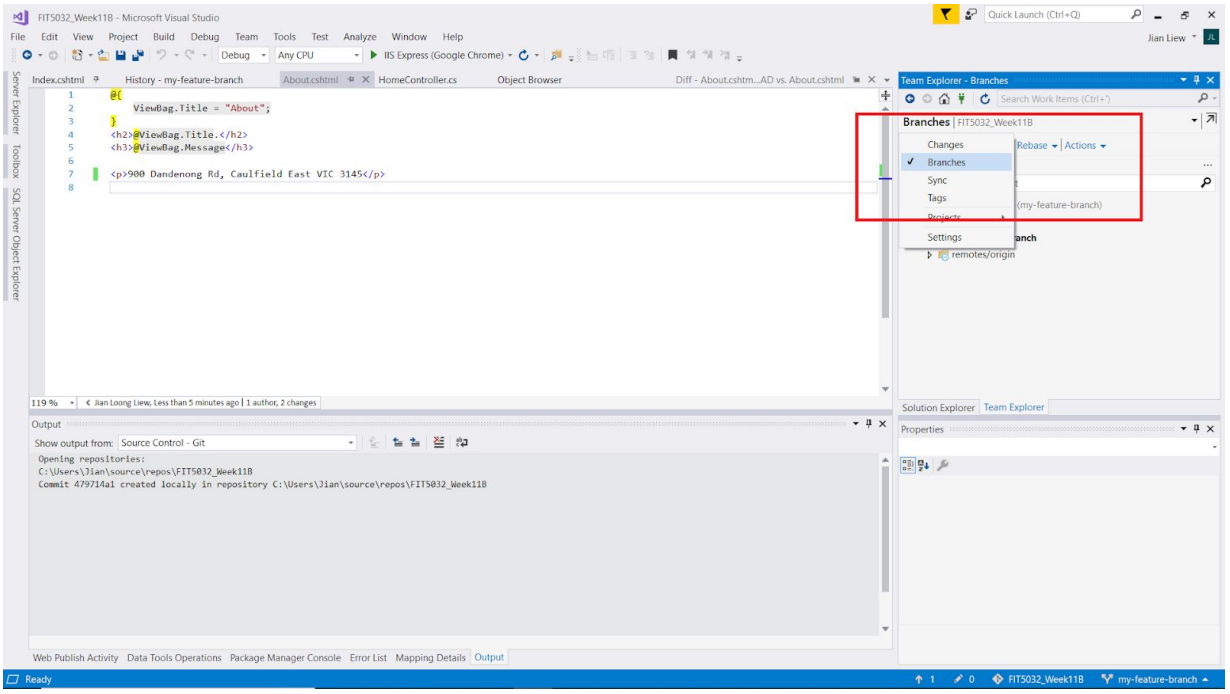
Before you do all the steps, we will need to add & commit the newly made changes to our feature branch. (git status) → a command line command that tells you the status of the git repository

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).

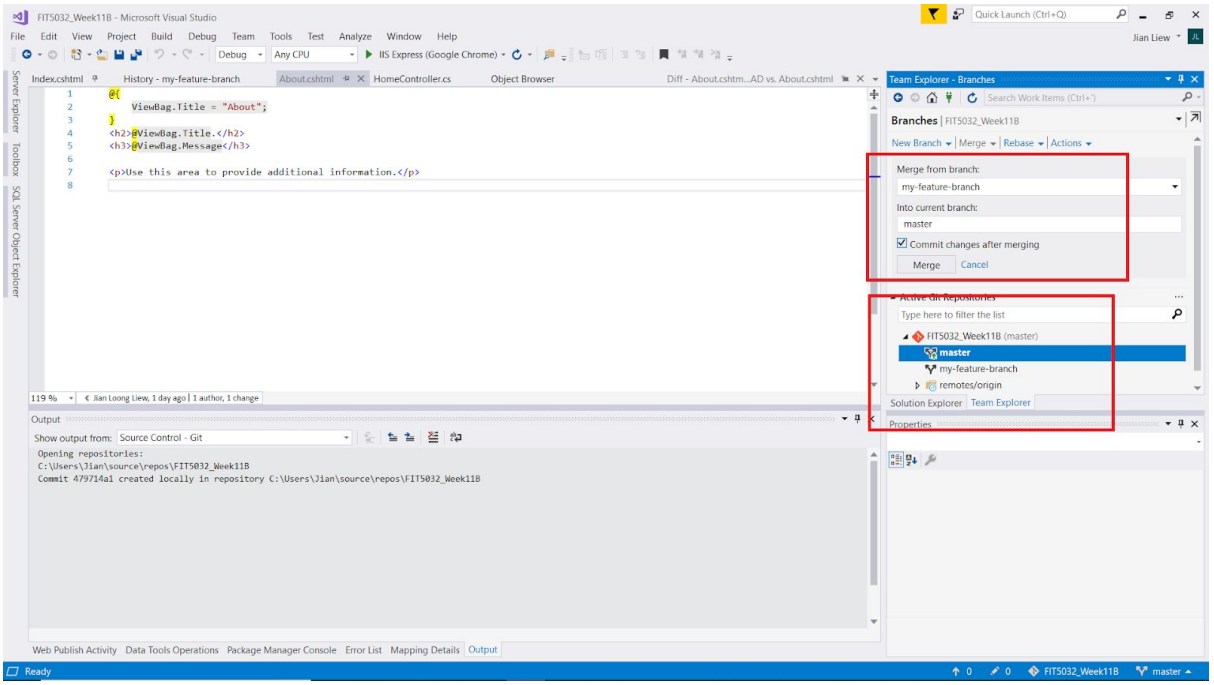


Step 24



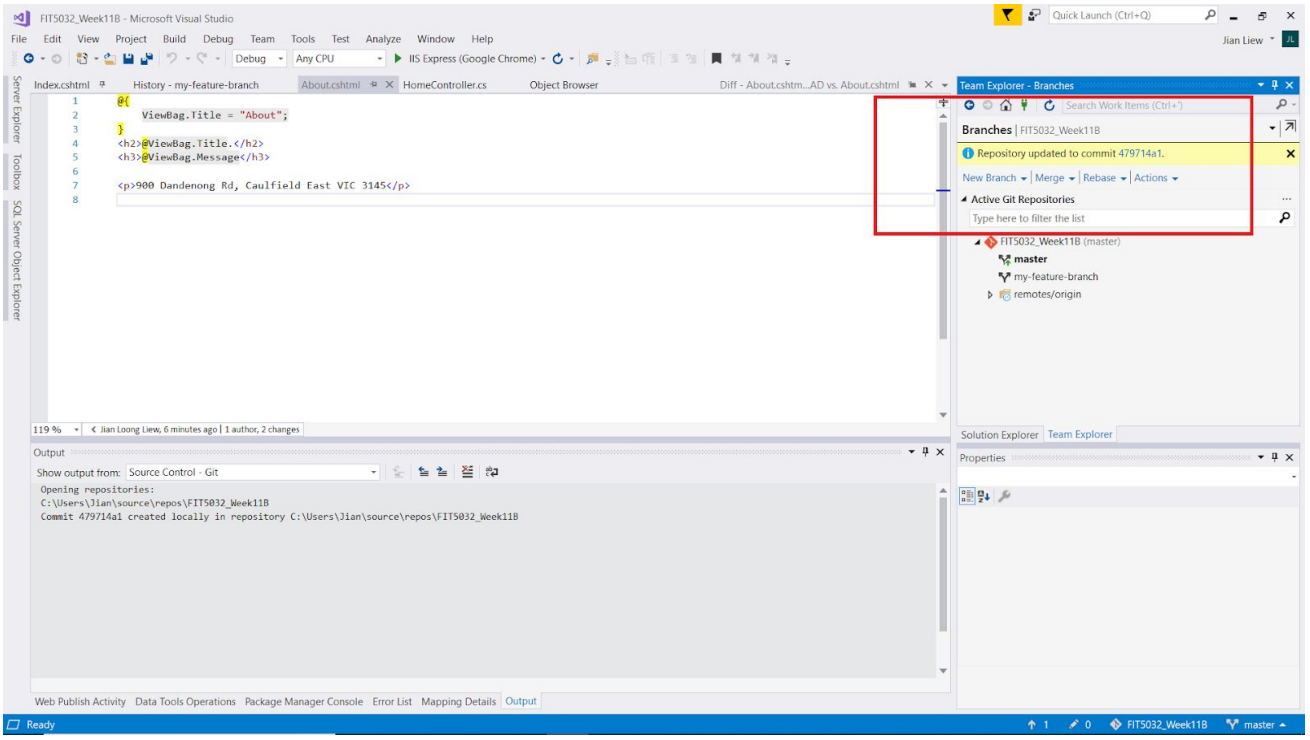
We will now go to the branches, and merge our feature branch into the master. **The reason we are doing it this way is to prevent as merge conflicts. (If our local master is up to date with the remote master from earlier and we merge, and later when we push into the remote, it will not have any merge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

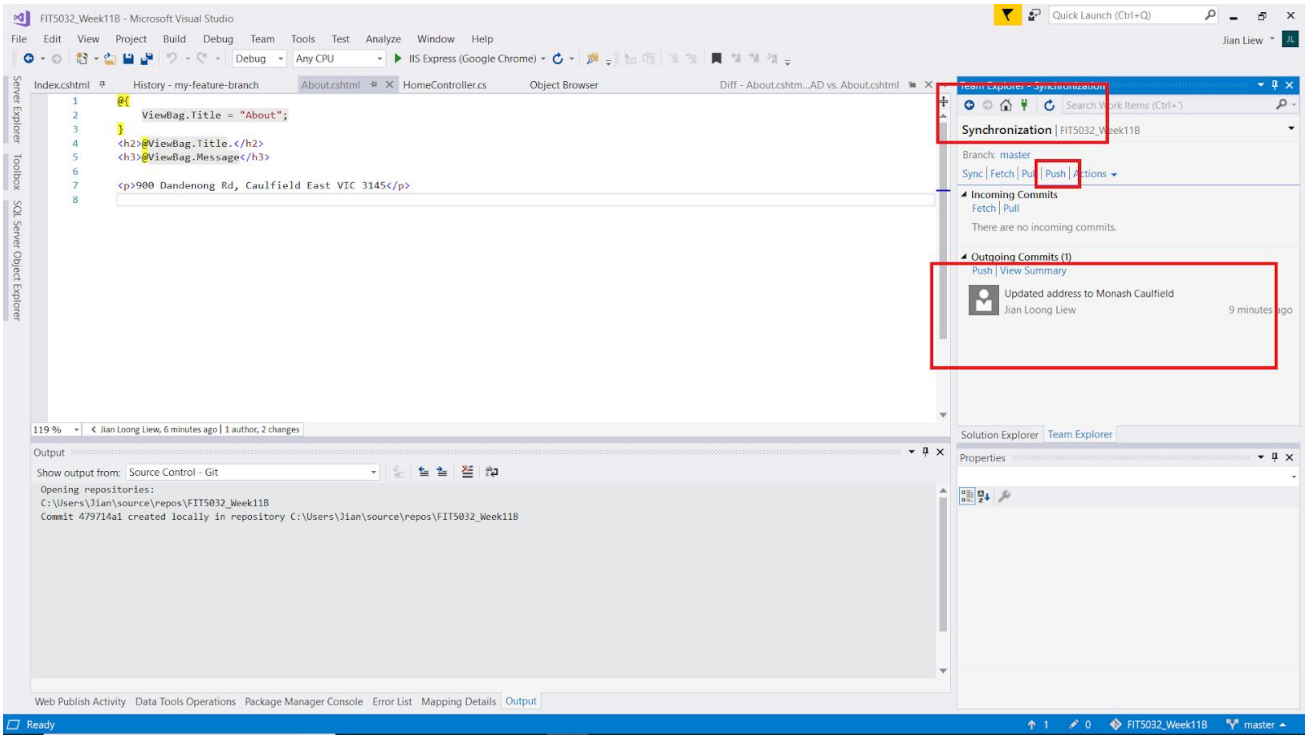
Remember that we are merging from the branch to the master. You might need to switch your active repository.



Step 25

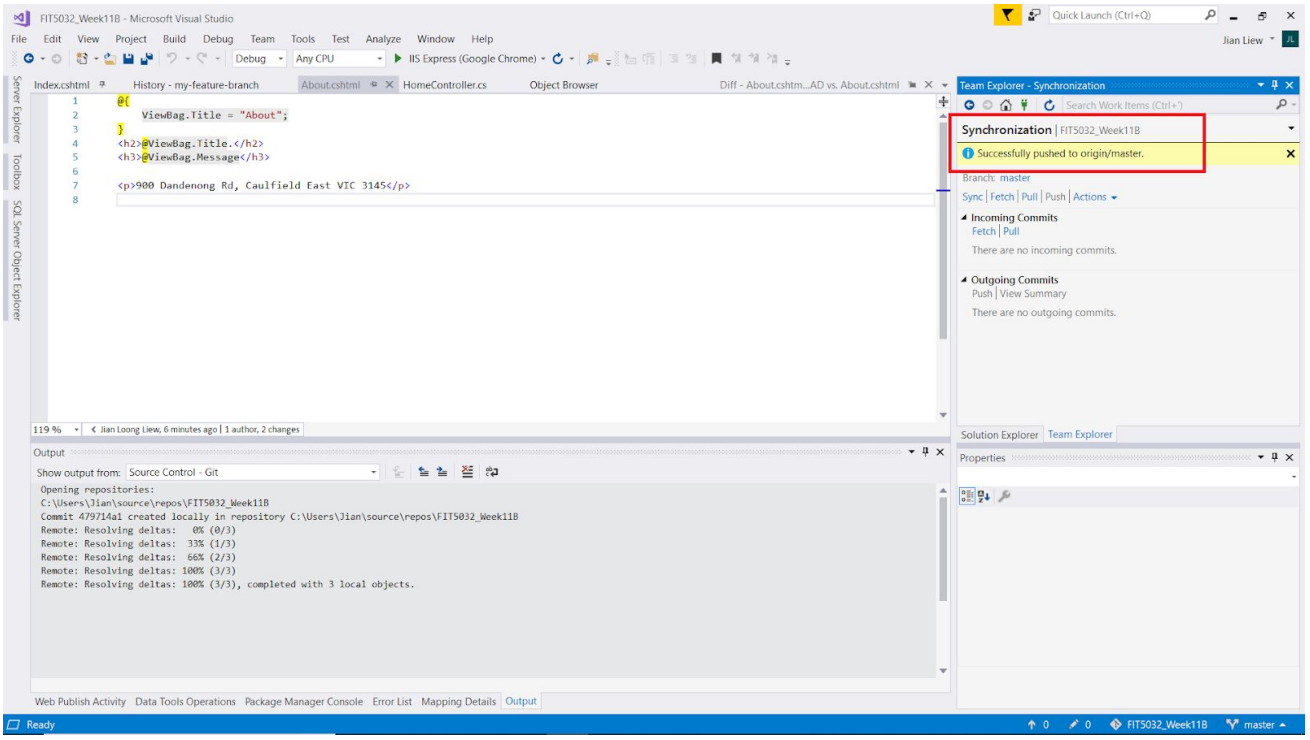
After you have done this, you will need to push the master into the remote master. Shown below.
This will trigger a build on Azure as our service application was set up to do so.



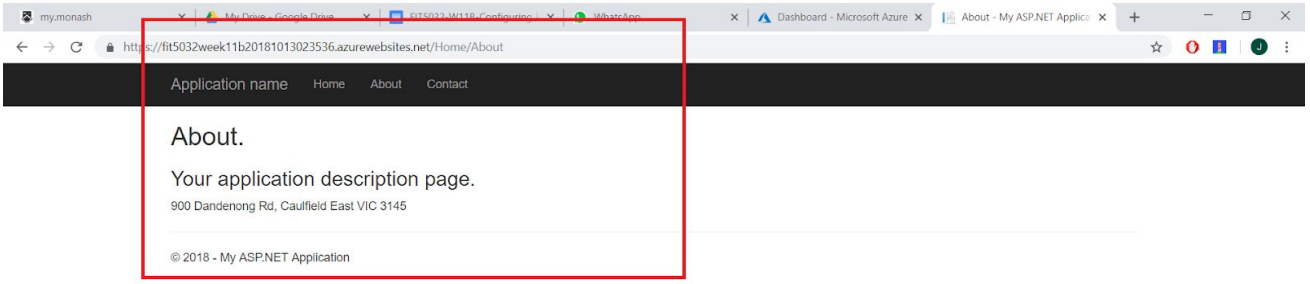


Step 26

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.