

Name: _____

Date: _____

Lab 0 – Getting Started with Azure DevOps and Visual Studio

Objectives

Part 1 – Setting up Visual Studio 2017

Part 2 – Working with Azure DevOps

Part 3 – Creating Hello World!

Background/Scenario

During this class, we will be using the C# programming language for all of the labs and lecture demonstrations. This lab will cover setting up your programming environment and creating a C# console application to verify everything is working.

When submitting completed lab assignments, you will need to commit the C# solution to a Git repository and push your commit to a remote server. This lab will also walk you through setting up a remote repository with Azure DevOps, which you will be able to push all of your labs to.

Required Resources

- Visual Studio 2017 with C# and .NET Environment
- .NET Framework 4.7.2 SDK
- Internet Connection

Part 1: Setting up Visual Studio 2017

Although Visual Studio 2019 was just released, I will be grading all the labs using Visual Studio 2017. Due to potential project and solution incompatibilities between Visual Studio versions, I highly recommend all students to use Visual Studio 2017 for their labs. While I won't be outright disallowing any student their choice in versions, just be aware that all sample code, lectures, and grading will be done in 2017. If on the (unlikely but nonetheless unfortunate) chance I cannot open your solution or get it to compile due to version issues, your lab will not be graded and I will ask you to convert the solution.

If you already have a working C# environment with .NET 4.7.2 installed, you may skip this part and move onto Part 2.

Step 1: Downloading and installing Visual Studio 2017

If you don't already have Visual Studio installed due to prior classes, download it and install it now. As a student, you are given free access to download Visual Studio Community from Microsoft.

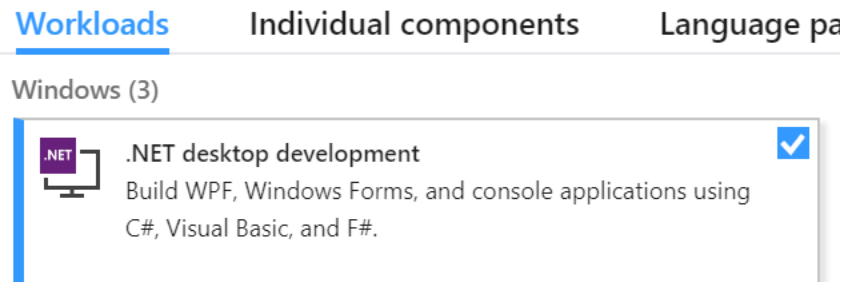
<https://visualstudio.microsoft.com/vs/older-downloads/>

As a student of the CSET department, you are also granted a free license to the other editions of Visual Studio, Professional and Enterprise. Being a student of OIT offers many licenses to different software for educational use. I highly recommend you make use of all of these free resources while you can. That being said, Community edition (which is free for everyone, not just students) will work just fine for the purposes of this class.

Step 2: Installing the C# and .NET components

Visual Studio is a very powerful code editing IDE that supports many languages and frameworks. Because it supports so much, it is also highly modular and you can add and remove any components at any time if you don't want to invest the disk space.

At any time (even after installation), you can launch the Visual Studio Installer (from your Start Menu, not the installer you downloaded) and change anything you want. For now, we are only interested in the .NET development environment. Under Workloads, make sure it is checked.

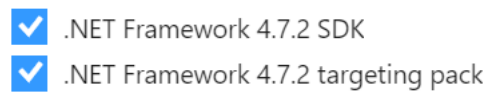


This will install everything you need to develop for .NET and C#.

Step 3: Installing the .NET 4.7.2 SDK

By default, only up to .NET 4.6.1 is installed, so let's install the latest so we can make use of some of latest language and framework features.

Click the Individual components tab and select the .NET 4.7.2 SDK and Targeting Pack.



Click the Modify or Install button at the bottom and grab a coffee while the Visual Studio Installer does its thing.

Part 2: Working with Azure DevOps

Git is a very powerful tool that allows you to track any changes you (or anyone on your team) makes to your projects. For all labs, we will be a "Team" and anything you commit and push to your Git repository I will be able to download on my side.

While Git is a free tool to use, many web hosting services that integrate with Git are not. Fortunately, Microsoft does provide a free and private project hosting service and we will be using that.

Step 1: Sign up for an Azure DevOps account

Go to the Visual Studio website and log in with a Microsoft account.

<https://go.microsoft.com/fwlink/?LinkId=228158>

Like all services by Microsoft, everything is linked to a singular Microsoft account. If you do not have one or wish to not use your personal Microsoft Account, your OIT email address is already an account! You can just simply log with your OIT email and password (if not already logged in) and proceed from there.

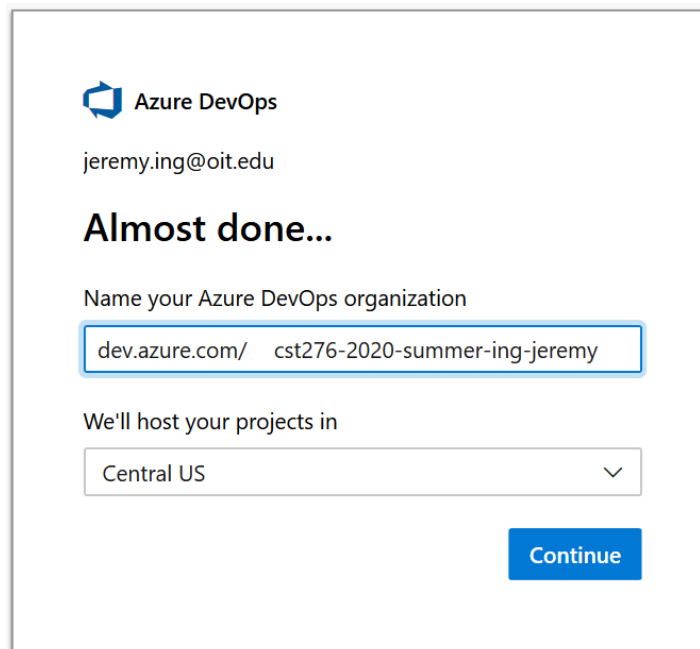
What is your email address that you used when logging in?

Step 2: Create your Organization

Once logged in, you will create a new “Organization” by clicking the Create Organization button in the upper left.

In the terms of Azure DevOps, an Organization is a workspace where you can invite other people and work collaboratively on projects.

For this class, your organization name will be **cst276-2020-summer-lastname-firstname**. This is so when I log into my dashboard, I can easily identify everyone.



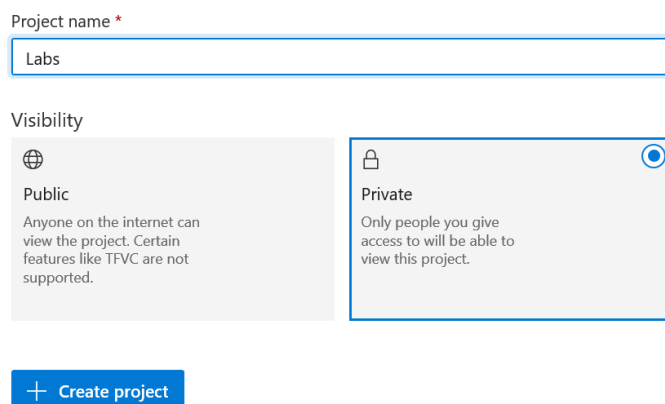
What is the name of your organization?

dev.azure.com/_____

Step 3: Create your Labs Project

This project is where you will be uploading all your labs to so we'll be calling it Labs. Leave the project visibility to private since we don't want the rest of the internet being able to see your code.

Create a project to get started




Step 4: Invite me to join your Organization

Once you have your Labs project created, you can now invite people to your Organization. In the upper right corner, click the Invite button and enter my email address: jeremy.ing@oit.edu

Invite members to Labs

Search and add users to your project

Add users or groups *

 Jeremy Ing

✕ Search users

If you are using your OIT account when you signed up, you should be able to search the OIT directory and find me. Otherwise, just type in my email and click Add.

Step 5: Initialize your Repository

You now have an empty repository and before we do anything, we need to initialize it with a .gitignore file first.

What is a .gitignore file and why is it important that we create one before doing anything?

On the left sidebar, click on Repos. You should see that your Labs repo is currently empty. Azure will list a few options to clone or import an existing repo, but we don't want to do that so just ignore those instructions.

At the bottom of the page, there is an option to initialize with a README or a .gitignore. You can optionally include the README if you want. In the pull down menu for the .gitignore file, search for and select Visual Studio. Then click Initialize.

^ or initialize with a README or gitignore

☒ Add a README

Add a .gitignore: VisualStudio ▾

Initialize

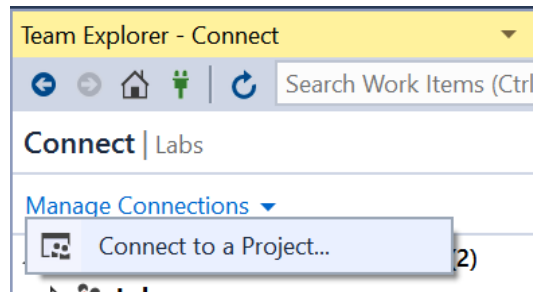
Once you have initialized your repo, you should see your .gitignore (and optionally README) file added to your empty repo. You should see the commit hash and description that Azure created.

What is the commit hash of your .gitignore file and what is the purpose of a commit hash?

Step 6: Connect to Visual Studio

Because we are using Microsoft to host your repository, Azure Dev Ops integrates directly with Visual Studio.

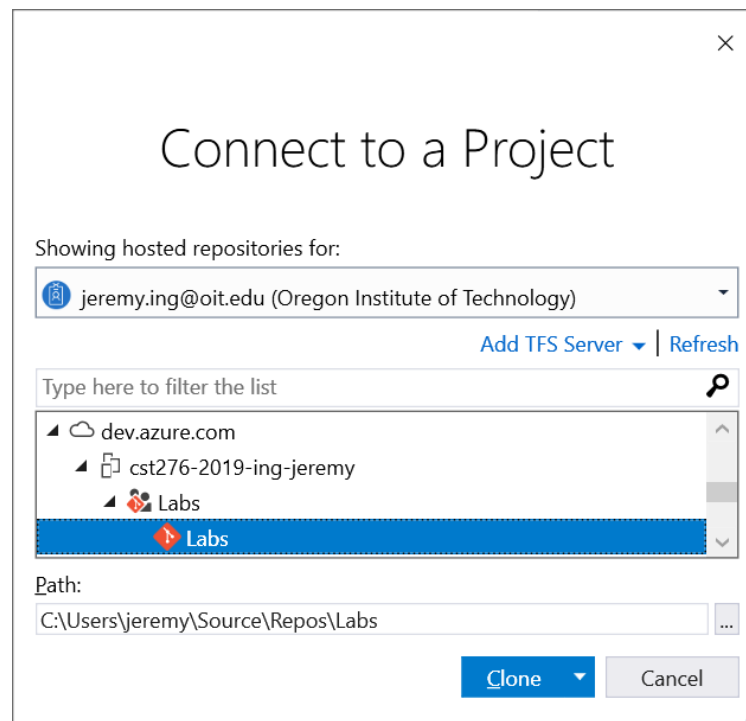
Launch Visual Studio and click the Team Explorer window. Click the Manage Connections link and Connect to a project.



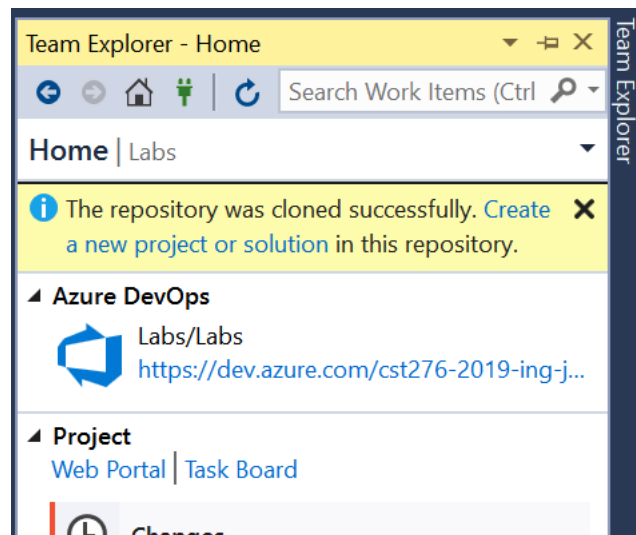
If you are not already logged into Visual Studio with your Microsoft account, you may be asked to log in now.

Step 7: Clone your Labs repository

Find your Labs project in the project explorer window and make sure you select the Labs repository and not the Labs project. You will know if you are cloning the right one if your window matches and the button says Clone and not Connect. Optionally specify a folder path where you want to clone your repo into and select Clone.



If your repository was cloned successfully, you should see this message.

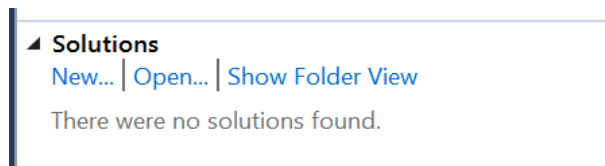


Part 3: Creating Hello World!

Now that we have successfully connected your repository to Visual Studio, we can now actually start with Lab 0!

Step 1: Create your Lab 0 solution

In the Team Explorer window, at the bottom, you will see the list of solutions in your repository. At the moment it is empty because we haven't created any yet, but here is where you will be able to see and access all of your labs.



Click on New to create your first solution. In the New Project window, select Visual C# and Windows Desktop from the left navigation and then Console App (.NET Framework). Name your project and solution Lab 0.

What is the difference between the Project Name and Solution Name?

In the future, you might start off with existing starter code so the project name might be something different every lab, but the solution name must always be "Lab #" where # is the number of the lab.

Step 2: Writing to the Console

In comparison to C++, C# makes writing to the console very easy. Right inside your Main function, type in:

```
Console.WriteLine("Hello World!");
```

Now press the green triangle play button labelled Start to begin debugging.

You should see the familiar console window pop up and immediately close just like your C++ labs did. In many of the labs, we get around this by typing

```
Console.ReadKey();
```

As the last line of code at the end of the Main function. This tells the console to simply wait for the user to press any key. Then once the console receives a key press, the program continues execution which then there is no more code to execute and the console window closes.

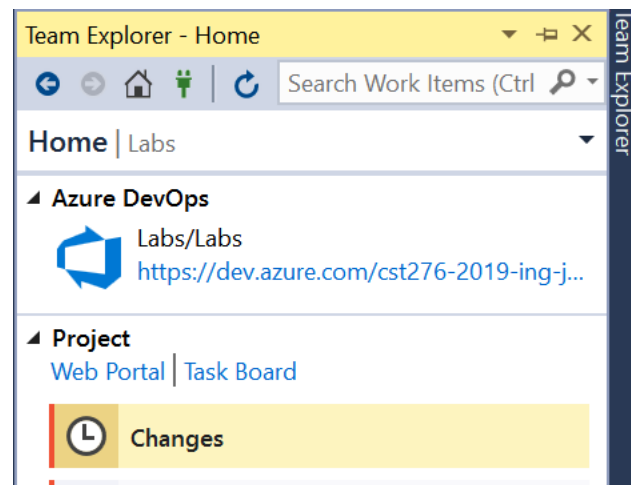
Verify that you can see your Hello World! Message in the console window and then press a key. The console should then close and debugging should stop automatically.

Congratulations! You can now rewrite all your C++ labs in C#! (Just kidding)

Step 3: Commit your Hello World program to Git

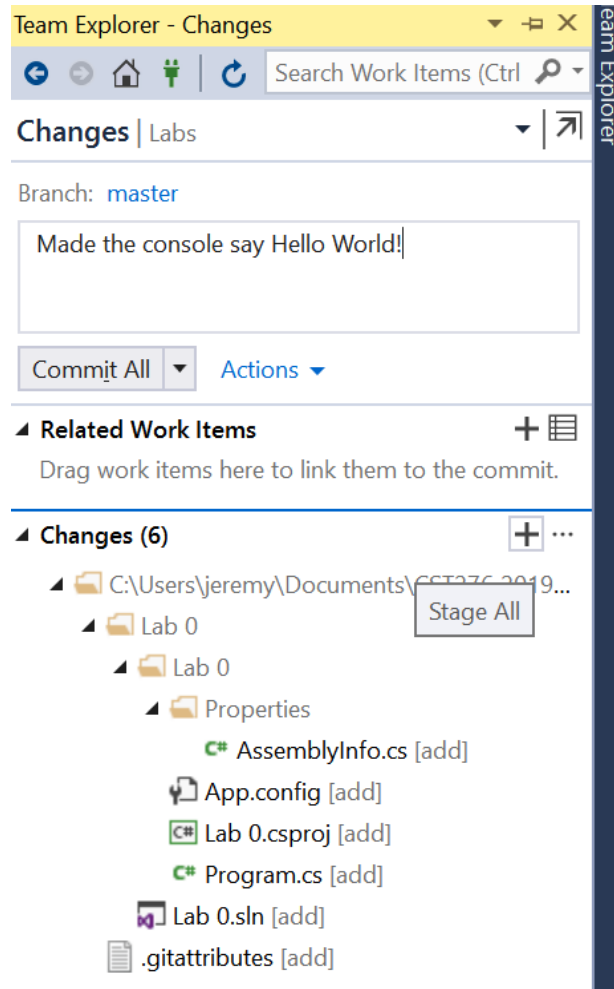
Now that you have made changes to your Lab 0 project. You will now commit it to your local Git repository.

Open the Team Explorer window again and click on Changes.



Once you click the changes, you should now see all the files in your solution that has been modified since the last commit. This is where we will select which files we want to commit and leave a description. The description is mandatory for all Git commits and it is helpful to not only yourself but to everyone on your team to see a summary of what you just committed in words rather than looking through source code.

Enter a good description for your commit and take a look at your file Changes. You will see a plus button at the top of your changes that will let you stage all changed files for the commit so you don't have to manually mark each one individually. Click the plus button now.



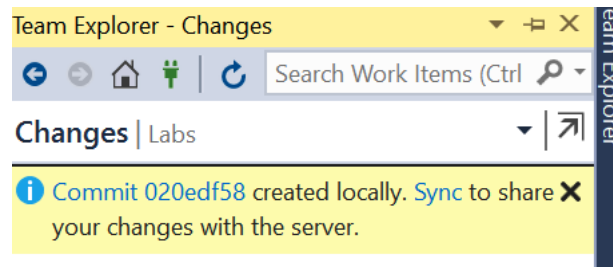
Once you do, all your files have now been moved up and marked as staged. Now click the Commit Staged button to finish the commit.

What does staging files for commit do? What happens to the unstaged files after the commit?

Previously, we just hit the stage all button. When would there be a time where you would want to not stage all and only stage one or two files?

Step 4: Sync your changes.

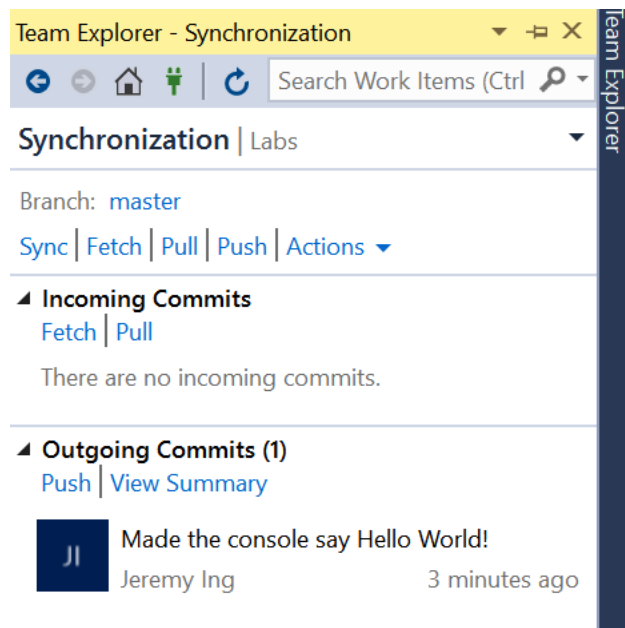
Once you have committed your changes, you should see a success message with your commit hash.



What is your commit hash?

Note, you are only committing to your local repository on your hard drive. You have not yet uploaded your changes to Azure yet. That is what this message means when it says Sync. Press the Sync button now. This will bring you to the sync page.

At the Synchronization page, Visual Studio will show you all of your commits that you have done since the last time you have synced with the server that have yet to be synced (Outgoing Commits). If you are working on a team or have made any changes to your repo on another computer, you will see any commits that the server has that are not downloaded to your own local repository (Incoming Commits).



Looking at the top row of the Sync window, you see a bunch of Git commands you can run. Sync, Fetch, Pull, and Push. What do they all do?

Command	What it does
Sync	
Fetch	
Pull	
Push	

Click Sync now.

Step 5: Verify the Sync worked.

Go back to the Azure window you had where you got the commit hash of your .gitignore file. If you have closed the window, you can easily get back to your Azure DevOps page by clicking the Home icon of your Team Explorer and clicking the Web Portal link.

Click Repos and you should now see your Lab 0 project folder inside your repo. Check the commit hash on your Lab 0 commit. Verify that it matches the hash you saw in Visual Studio.

Everything you see here in the Web Portal is everything I will be able to see as a member of your Organization. Just remember, even if you commit, you have to sync before it is uploaded. If you are ever unsure if your sync worked, just come back to this page and you can see if your files are there.

 Lab 0

12 minutes ago

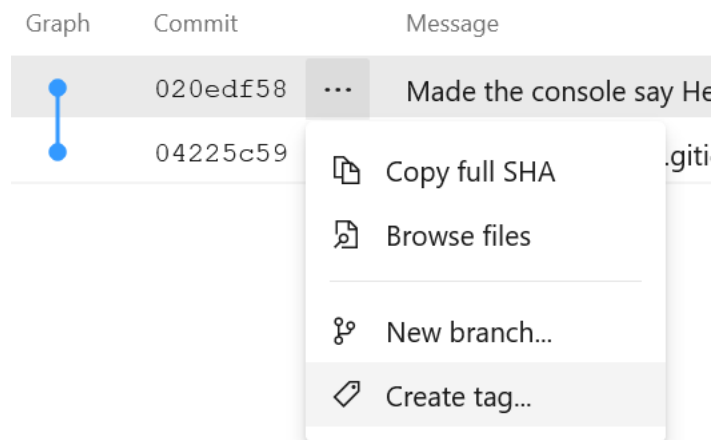
020edf58

Made the console say Hello World!

Step 6: Create a tag

Normally only once you are finished with the lab you are working on, you will create a tag. Tagging a commit usually means you hit some important milestone in the development of your project. Some milestones could be a major version increment or more commonly a software release. In the case of the class, meaning you finished a lab. When working with large projects and teams, you can sometimes have hundreds or thousands of commits between milestones. Tagging a specific commit is an easy way of identifying those milestones.

In the left side navigation bar, select Commits. You should see a listing of all your commits that you have synced with the server. Click the three dots menu on your latest commit (verify the hash matches what you answered in the prior question) and click Create Tag.



Name the tag ExampleTag and enter a sample description. Note, tags cannot contain spaces. Once you have created your tag, click on Tags in the left navigation bar to verify it was created.

Remember, when submitting labs, you will always tag them Lab# where the # represents which lab number you are working on. Once you finish Lab 0, you will name that commit's tag as Lab0.

Tagging your lab submissions identifies to me which commits I should be viewing to grade your labs. You could have potentially a hundred commits but, I will easily be able to see which commit you intend to submit by the tag.

When submitting the lab to Canvas, you will provide me with the tag name, Lab0 in the case of this lab, and the commit hash, 020edf58 in this example screenshot.

Step 7: Create a Class Library

For every Lab you will work on for this class, you will either be creating new class libraries or modifying existing ones.

What is a Class Library?

Why should we use them?

Right click your solution in your Solution Explorer and click Add -> New Project.

In the New Project dialog, select Class Library (.NET Framework).

Name your project HelloLib and create the project.

Step 8: Reference the Class Library

Now that you have added a new Class Library project, you will have to reference it before you can use it.

Right click the References of your Console project and click Add Reference.

Under the Projects category, select Solution. Then click the checkbox for your HelloLib project and click Add.

Step 9: Use the Class Library

By default, Visual Studio adds a premade class for you in your library project called Class1.cs. Open the file and locate the Class1 class.

Create a public function named Hello() that returns a string and have the function simply return “Hello world!”.

Back in your main console program, instantiate a new instance of Class1 at the beginning of the main function.

```
Class1 class1 = new Class1();
```

Note, Visual Studio is probably red underlining Class1 indicating a build error.

Why is Class1 red underlined?

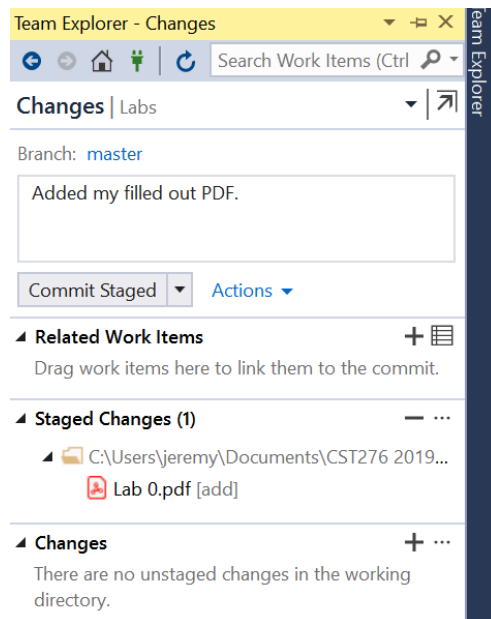
Add the using statement for your library's namespace at the top along all the other using statements.

```
using HelloLib;
```

Replace the “Hello world!” string literal inside the WriteLine function with a call to the Hello() function and run your code. It should print out “Hello world!”.

Step 10: Save this PDF

Finally, you will save your filled out version of this lab PDF into your Lab 0 solution folder. (Don't forget to answer the Reflection questions at the bottom!) Once you have done that, you should now see in Visual Studio that there is now a PDF document in the changes list. Stage it, commit it.



Remember, there is absolutely no harm in committing after every small change you do. Actually it's quite the opposite; it will only benefit you in the long run. If you ever break your code or accidentally delete something, you can easily roll back your code to a previous commit and continue from there.

Step 11: Submit to Canvas

Once you have ensured your solution builds with no error, has the expected output, and have answered all the questions including the Reflection question section, it is now time to submit to Canvas.

Make sure everything has been committed and synced to Azure.

Create the Lab0 tag for your latest commit and copy the SHA1 hash of the commit.

In Canvas, open the Lab 0 assignment and when you are ready to submit, you will be presented with a textbox. Type in the tag and hash like so:

Tag: Lab0

Hash: efa89b90ecff7cf7b6b406db211cfb3ec2bb85b4

Then you can submit the assignment. That's it!

Reflection

The Reflection section of the lab is for questions with not really any right or wrong answer. It is a way for you to reflect on some of the questions of the lab and it gives me a small one-on-one time with your thoughts for you to explain some things in your own words about the lab.

1. Have you encountered any issues when completing the lab?

2. Have you had any prior experience with Git before? If so, what Git clients have you used in the past?

3. Have you had any prior experience with C# before?
