

Name: _____

Date: _____

Lab 0 – Getting Started with Azure DevOps and TortoiseGit

Objectives

Part 1 – Setting up TortoiseGit

Part 2 – Working with Azure DevOps

Part 3 – Submitting Lab 0

Background/Scenario

During this class, we will be working with many different programming languages and each one will use a different IDE. Because of that, we will not be using the Visual Studio DevOps integration but instead we will be using another Git tool called TortoiseGit.

This lab will cover setting up an Azure DevOps account and cloning the repository with TortoiseGit. This is where you will submit all of your assignments to me for grading.

Required Resources

- Git
- TortoiseGit
- Internet Connection

Part 1: Setting up TortoiseGit

Git is a powerful source control system that has many features. Because of the many features, it can be pretty complex to work with and have a steep learning curve. One of my favorite tools to help with using Git is TortoiseGit.

If you already have a working Git environment set up and are already comfortable working with your existing tools, you can skip this part and proceed to Part 2.

Step 1: Download Git for Windows

If you don't already have Git for Windows installed (git command line or git bash) download and install Git for Windows from the link below

<https://git-scm.com/downloads>

Step 2: Download TortoiseGit

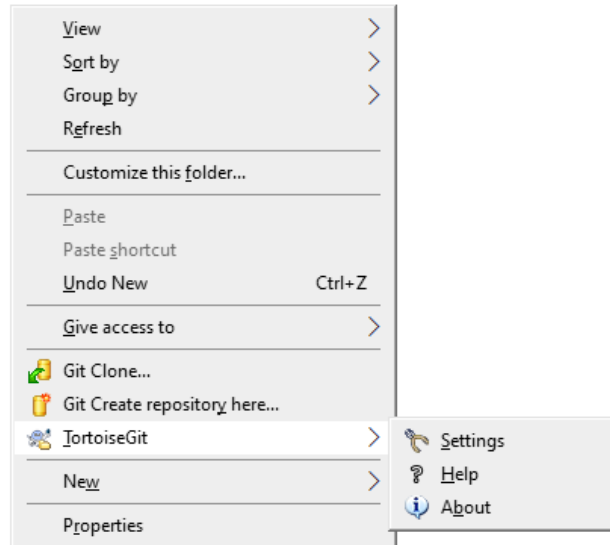
Download and install TortoiseGit from the link below

<https://tortoisegit.org/download/>

Step 3: Verify TortoiseGit is installed and working

TortoiseGit is not a program you typically run from the start menu. Instead, you interact with it completely from the Windows Explorer from the context menu.

Create a new folder and right click it.



If TortoiseGit is installed correctly, you should see these new options in your context menu.

Click Settings.

If the TortoiseGit settings window opens just fine, everything installed correctly. Close the settings window and proceed to Part 2.

Part 2: Working with Azure DevOps

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, Azure DevOps, and we will be using that.

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.

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.

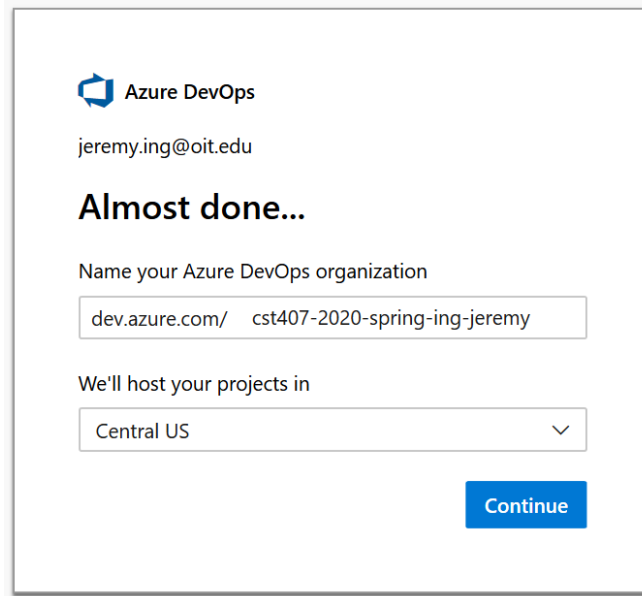
What is your email address that you used when logging in?

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 **cst407-2020-spring-lastname-firstname**. This is so when I log into my dashboard, I can easily identify everyone.



Azure DevOps

jeremy.ing@oit.edu

Almost done...

Name your Azure DevOps organization

dev.azure.com/ cst407-2020-spring-ing-jeremy

We'll host your projects in

Central US

Continue

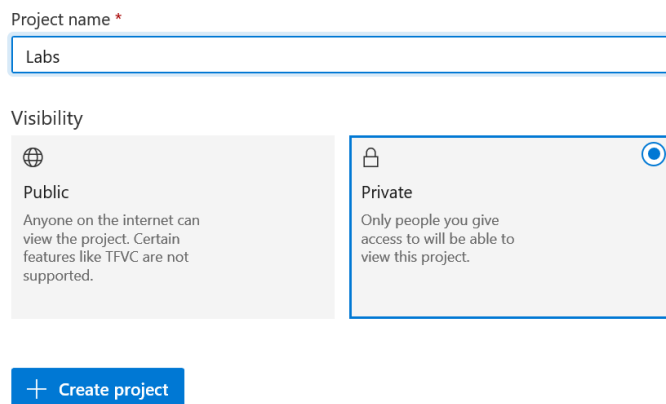
What is the name of your organization?

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



Project name *

Labs

Visibility

☒ Public
Anyone on the internet can view the project. Certain features like TFVC are not supported.

☐ Private
Only people you give access to will be able to view this project.

+ Create project

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 *

JI

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: Clone your Repository

From the left side navigation sidebar, click on Repos. You should be given your cloning URL. Click the Copy button to copy the full URL.

Clone to your computer

HTTPS

SSH

https://cst223-2020-winter-ing-jeremy@dev.azure.com/cst223-2020-

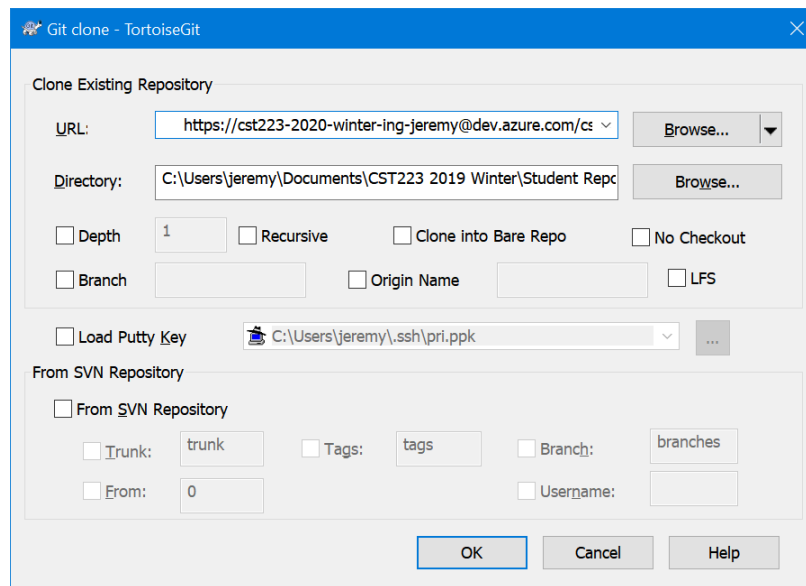
OR

Clone in Visual Studio

Copy clone URL to clipboard

Once copied, minimize the DevOps window because we will come back to it in a bit.

In Windows Explorer, navigate to a folder where you want to save your repository to, then right click it and select Git Clone from the TortoiseGit commands.



TortoiseGit will automatically fill in your URL directly from what you copied from Azure and your current directory. You should not need to change anything. Click OK.

TortoiseGit will now clone the repository from Azure DevOps. Enter your credentials when it prompts you. If successful, you should have your repository cloned and the folder should have a green checkmark.

Step 6: 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?

Go back to the minimized Azure DevOps window.

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 7: Pull your commit

Back in Windows, right click your repository and in the TortoiseGit menu, select Git Pull. Once finished, you should now see your newly created gitignore and readme file in your local repository.

Part 3: Submitting Lab 0

All labs will be saved in your repository and you will submit the commit hash in Canvas.

Step 1: Create a Lab 0 folder

Inside your repository folder in Windows, create a new folder and name it Lab 0.

For each lab, you will create a new folder and save all your documents in their respectively named folders.

Step 2: Create a text file

Using Notepad, create a text file named Lab0.txt and save it inside the Lab 0 folder.

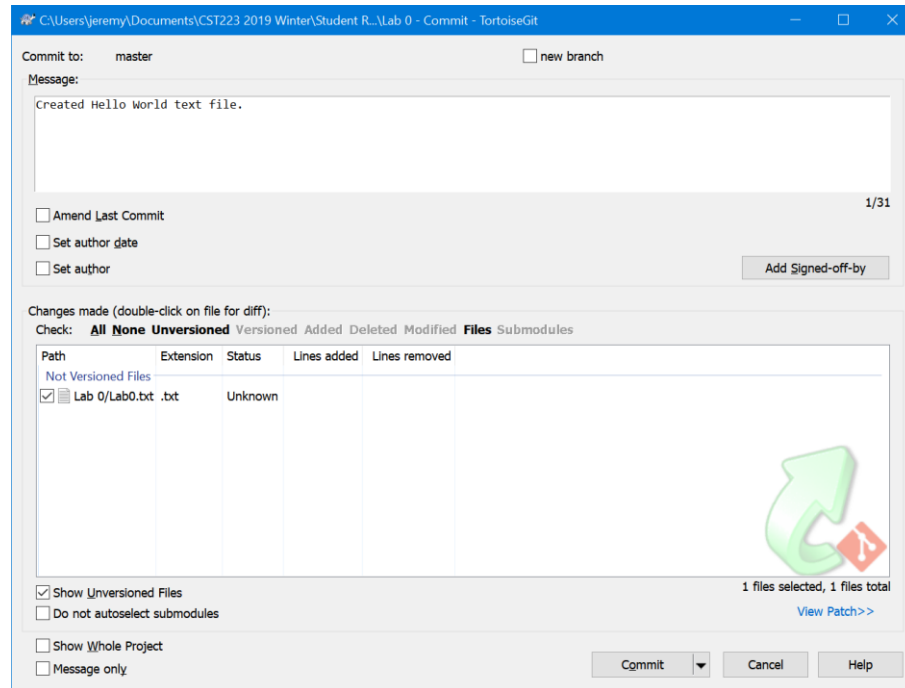
Type the following string into the text file:

Hello World!

Save the text file.

Step 3: Commit your changes

Right click the folder and in TortoiseGit, select Git commit -> “master”



This window will show you all the changes to your repository. Check the box to Lab0.txt to stage the file for commit. Type in a descriptive message that describes what our commit contains and click the Commit button.

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

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.

Step 4: Push your commit

Making any changes to your repository (including committing) does not automatically upload it back to Azure DevOps. To upload your committed changes, you have to do a Push.

Right click again and select Git Push.

TortoiseGit will have the master branch selected for you so you can just leave all the options default and click OK.

Step 5: Verify the Push worked

Back to Azure DevOps, refresh your Repos page. You should see your Lab 0 folder and Lab0.txt file uploaded.

You should also see the commit hash next to the file. Check to make sure the commit hash matches on DevOps as it did from the previous step.

 Lab 0

12 minutes ago

020edf58

Made the console say Hello World!

Step 6: Research other Git Commands

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

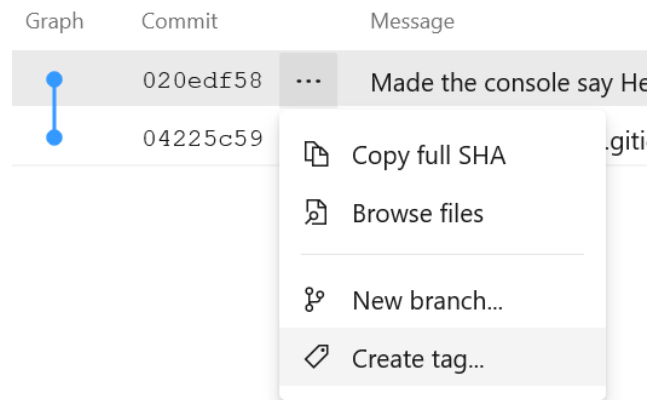
Command	What it does
Sync	
Fetch	
Pull	
Push	

Step 7: Create a tag

Normally you will create a tag once you are finished with the lab you are working on. 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. 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 case of this class, the tag will mark the completion of a lab.

For now, let's create an example tag just to see how it works.

Back in DevOps, 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.

You can optionally create tags in TortoiseGit, but if you do, you have to remember to push after creating the tag. You may need to click the checkbox "Include Tags" if it isn't already checked.

Step 8: Save this PDF

Finally, you will save your filled out version of this lab PDF into your Lab 0 project folder. 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 9: Answer the Reflection Questions

At the end of every lab, there will be a few reflection questions, don't forget to answer them!

Save the PDF, Commit, then Push.

Step 10: Submit to Canvas

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.

For example:

Tag: Lab0

Commit: 020edf58

You don't need to upload or attach anything else to Canvas. I just need the Tag and Commit hash.

Remember, anything you push to your repository, I can see it and pull it to my computer for grading.

If you want to confirm that everything pushed correctly, you can always go back to Azure DevOps and see your repository there. Everything you see on the website is exactly what I will see when I pull your repository.

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?

3. What Git clients have you used in the past (e.g. TortiseGit or Git Bash) and what is your favorite? If this is the first time you are using Git, did you find TortoiseGit easy to use?
