

Software setup

We will need the following software set up on whatever computer(s) you plan to use for this course. It should all work equally well on Mac, Windows, or Linux. These are somewhat 'bare bones' instructions, so use [Piazza](#) to ask questions if you encounter any problems.

Java SDK 7

The first step is to install the Java Software Development Kit (SDK). You may already have a Java Runtime Environment (JRE) if other applications on your system require it, but **that is not enough**. Unless you have programmed in Java on this system before, chances are you'll need to install a JDK from scratch.

Start at this link: <http://www.oracle.com/technetwork/java/javase/downloads/index.html> and scroll down to **Java SE 7uXX » JDK » Download** — Android might **not work** with the latest Java 8!

Java SE 7u71/72

[Auto-update Notice & End of Public Updates for Oracle JDK 7](#)

Coincident with the January 2015 CPU release users with the auto-update feature enabled will be migrated from Oracle JRE 7 to Oracle JRE 8. Also, please note the April 2015 CPU release will be the last Oracle JDK 7 publicly available update. For more information, and details on how to receive longer term support for Oracle JDK 7, please see the Oracle Java SE Support Roadmap.

These releases includes important security fixes. Oracle strongly recommends that all Java SE 7 users upgrade to one of these releases.
[Learn more](#)

- Installation Instructions
- Release Notes
- Oracle License
- Java SE Products
- Third Party Licenses
- Certified System Configurations

JDK
DOWNLOAD

Server JRE
DOWNLOAD

Figure 1: Java SE 7 JDK Download

On the next page, you'll need to accept the license agreement, and then you can find the download link for your platform — probably Windows x64 or Mac OS X.

Go through the Oracle setup wizard; all the defaults are fine. At this point you may

be redirected to a web browser to register with Oracle. That is completely optional.

Android Studio

Download Android Studio for your platform, at <http://developer.android.com/sdk/index.html>

Run the installer — the defaults should be fine. After you start Android Studio for the first time, it will download some additional components. It may even alert you that a new version is already available! Don't feel you have to update every time a new version is released — in general, if things seem to be working, I'd recommend leaving it alone.

Git and GitLab

Git is a version control tool, that helps developers keep track of any changes that are made to a set of files. We'll also use it to submit and collaborate on assignments. You may have heard of GitHub, a social code-sharing site. We'll be using a similar service called GitLab, which more easily allows **private** code repositories.

Note: if you have already set up Git and GitLab for [CS120](#), then you can just create a new **private** project named cs164 and skip to step 7 below.

1. Install the Git distributed version control application from <http://git-scm.com/downloads>. The default settings are fine.
2. Sign up for an account on GitLab: https://gitlab.com/users/sign_up. Once logged in to your dashboard, create a new **private** project named cs164.
3. Open the **Git bash** application on Windows, or use **Utilities » Terminal** on Mac. Type these commands, replacing your actual name and email address in the double quotes:

```
git config --global user.name "YOUR NAME"  
git config --global user.email "your.address@example.com"
```

4. Create an SSH key pair, assuming you don't already have one on this computer. To do that, in the Git bash terminal, enter ssh-keygen. You'll press enter for all the defaults, including an empty passphrase. It should go something like this:

```
$ ssh-keygen  
Generating public/private rsa key pair.  
Enter file in which to save the key (/c/Users/league/.ssh/id_rsa):  
Created directory '/c/Users/league/.ssh'.  
Enter passphrase (empty for no passphrase):
```

Enter same passphrase again:

Your identification has been saved in /c/Users/league/.ssh/id_rsa.

Your public key has been saved in /c/Users/league/.ssh/id_rsa.pub.

The key fingerprint is:

b7:27:3e:8d:83:df:9a:21:c7:27:0d:fa:43:ac:d0:7e league@WIN7

5. Now you'll want to access the *public* part of the key that it generated. It's easiest to open it in a text editor, using:

- On Windows: notepad .ssh/id_rsa.pub
- On Mac: open -e .ssh/id_rsa.pub

The editor will pop up with a line that begins ssh-rsa AAAA... copy the whole thing onto your clipboard.

6. In your GitLab account, go to **Profile settings » SSH Keys » Add SSH Key** and paste your key in the big box, then **Add Key**.

7. On your computer, open the Git GUI application. Select **Create New Repository**. In the **Directory** box, browse to somewhere convenient such as Desktop or Documents, and then manually add /cs164 onto the end (without creating that folder). In my example, the Directory is C:/Users/league/Desktop/cs164

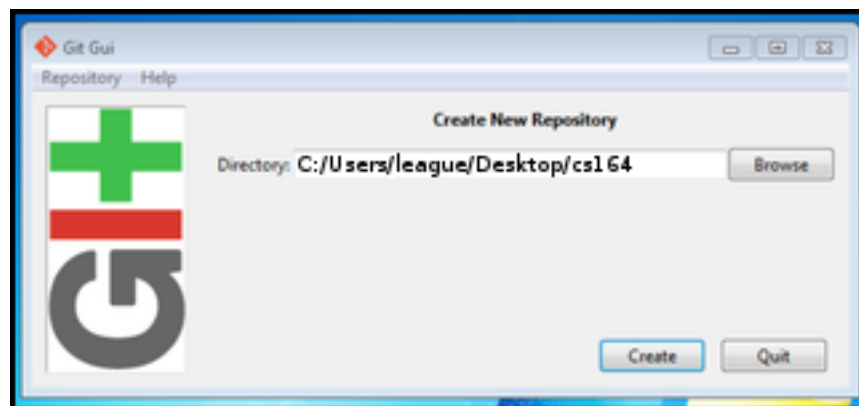


Figure 2: Create New Repository

Mac users: you may not be able to find the Git GUI application. Here is [how to proceed](#).

8. You should now see the repository status viewer. You will interact with this interface whenever you want to save your work and synchronize it to the GitLab server.
9. Create a new file called `readme.txt` and save it within the new `cs164` folder. In the file, just type your name and whatever else you want.
10. Switch back to the repository status viewer.

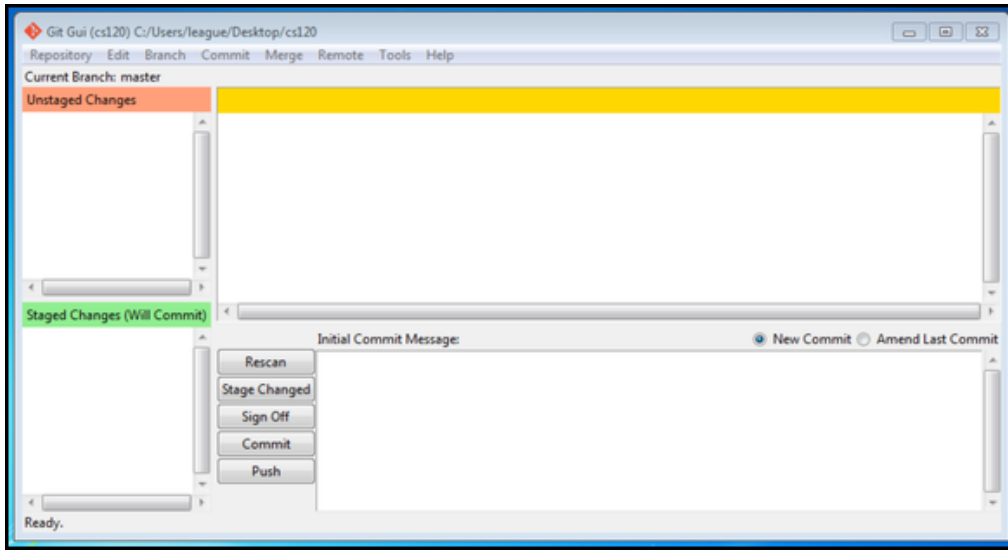


Figure 3: Repository status viewer

- a. Hit the **Rescan** button. Your `readme.txt` should appear in the upper left area, labeled **Unstaged Changes**.
- b. Hit the **Stage Changed** button. The app will confirm and move the `readme.txt` to the lower left area, labeled **Staged Changes**.
- c. Type a message such as “my initial commit” in the lower box labeled **Initial Commit Message**.
- d. Hit the **Commit** button. The status bar at the bottom should say something like “Created commit abcdef: my initial commit.”
- e. Another way to see the commit is to select **Repository » Visualize master’s history** from the menu. Your commit should be highlighted, and the contents of your `readme` file will also appear there.

After confirming the details, you can close the history viewer.

11. The next step is to push the new commit up to the GitLab server, which we will configure as a ‘remote’. Go back to your GitLab `cs164` project page and find the SSH URL. It should look something like `git@gitlab.com:USERNAME/cs164.git`. Copy that onto the clipboard.

12. In the repository status viewer, select **Remote » Add** from the menu.

Type `origin` as the **Name**, and paste your `git@gitlab...` URL as the **Location**. Select **Initialize Remote Repository and Push**, and hit **Add**.

A successful push should look something like this:

```
Not allowed command [ignore this]
Pushing to git@gitlab.com:USERNAME/cs164.git
To git@gitlab.com:USERNAME/cs164.git
```

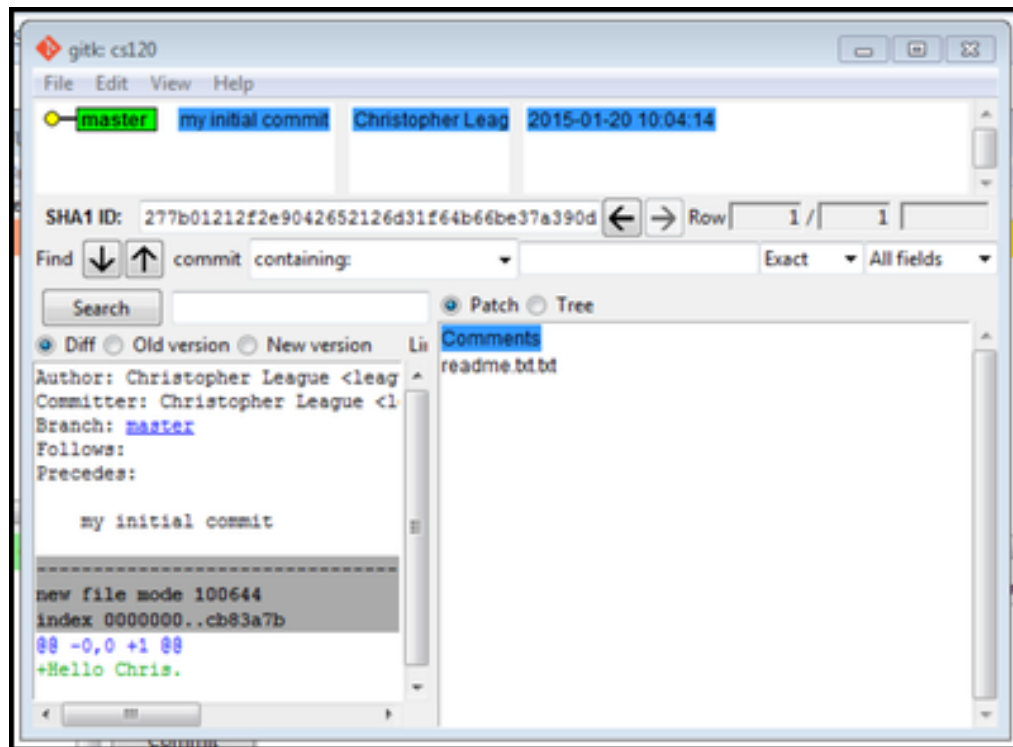


Figure 4: History viewer

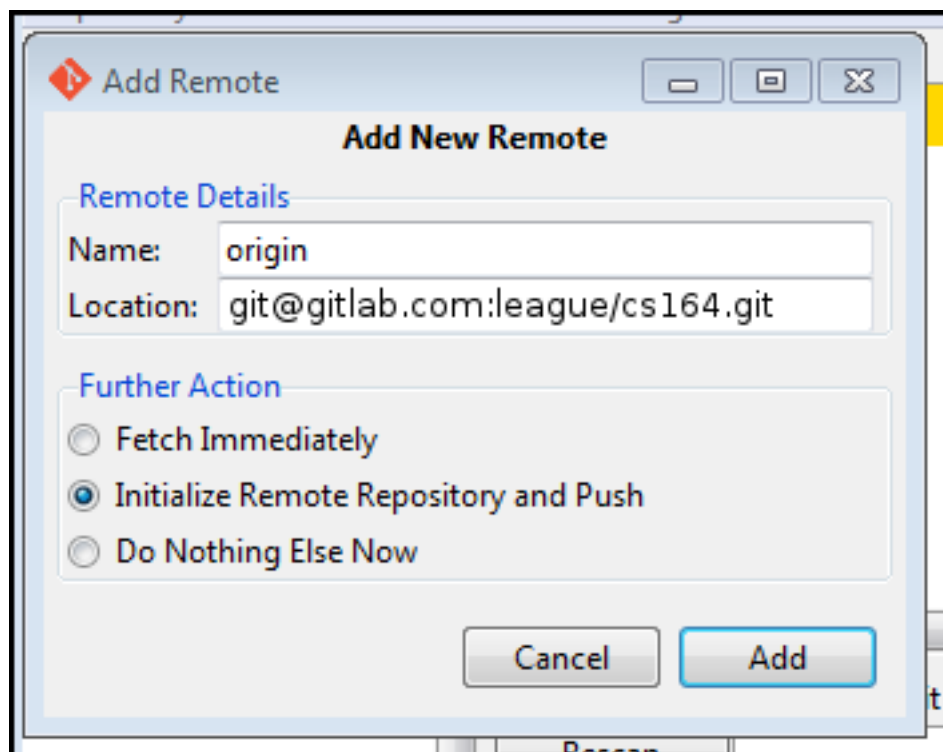


Figure 5: Add New Remote

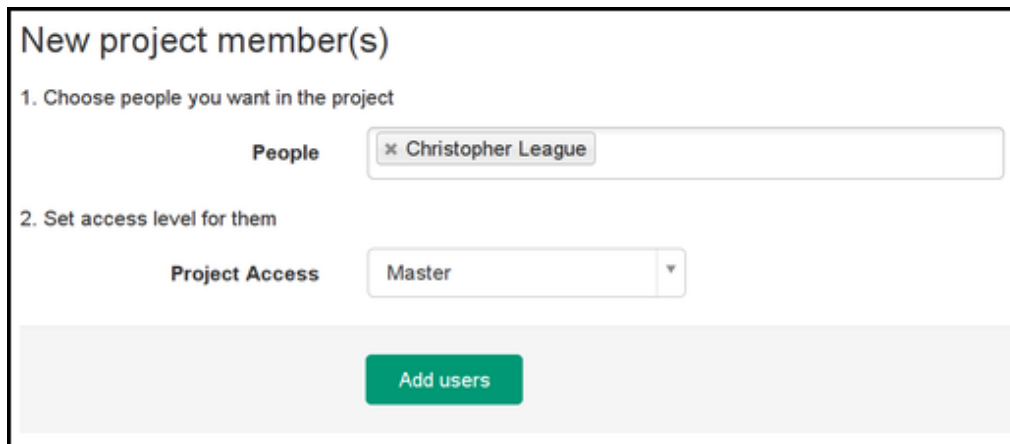
```
* [new branch]      master -> master
updating local tracking ref 'refs/remotes/origin/master'
Success
```

Mac users: if the push seems to be taking a while, switch back to the terminal where you started the `git gui`. If it's asking a yes/no question, type `yes` and press `enter`; then the process should continue.

13. Now, in some text editor, make a small change to the `readme` file you saved in `cs164` – just add a one-line message to it. In the repository status viewer, repeat these steps. We'll use these whenever you have something new to submit or synchronize:
 - a. **Rescan** button
 - b. **Stage Changed** button
 - c. Type a **Commit Message**, such as “added one line to `readme`”
 - d. **Commit** button
 - e. **Push** button
 - f. Confirm the **Push** in the next dialog.

If you reload the project page on GitLab, you should see a summary of the history, with your most recent commit first: “Christopher League pushed to branch `master` at Christopher League / `cs164` ... added one line to `readme`.”

14. This last step will allow me to see your work. On your GitLab repository page, open up **Settings** in the left sidebar, and then select **Members**. Hit the green **New project member** box, and search for the account named `league`. My name should appear. Select that, and add me with access **Master**, as shown below.



The screenshot shows a web form titled "New project member(s)". It is divided into two sections. The first section, "1. Choose people you want in the project", has a "People" label and a text input field containing "x Christopher League". The second section, "2. Set access level for them", has a "Project Access" label and a dropdown menu currently set to "Master". At the bottom of the form is a green button labeled "Add users".

Figure 6:

Confirm by hitting **Add users**.

15. Congratulations, your Git is working!