ECSE 321 Introduction to Software Engineering

Hands-on Tutorials

McGill University

Table of Contents

- HTML version
- PDF version

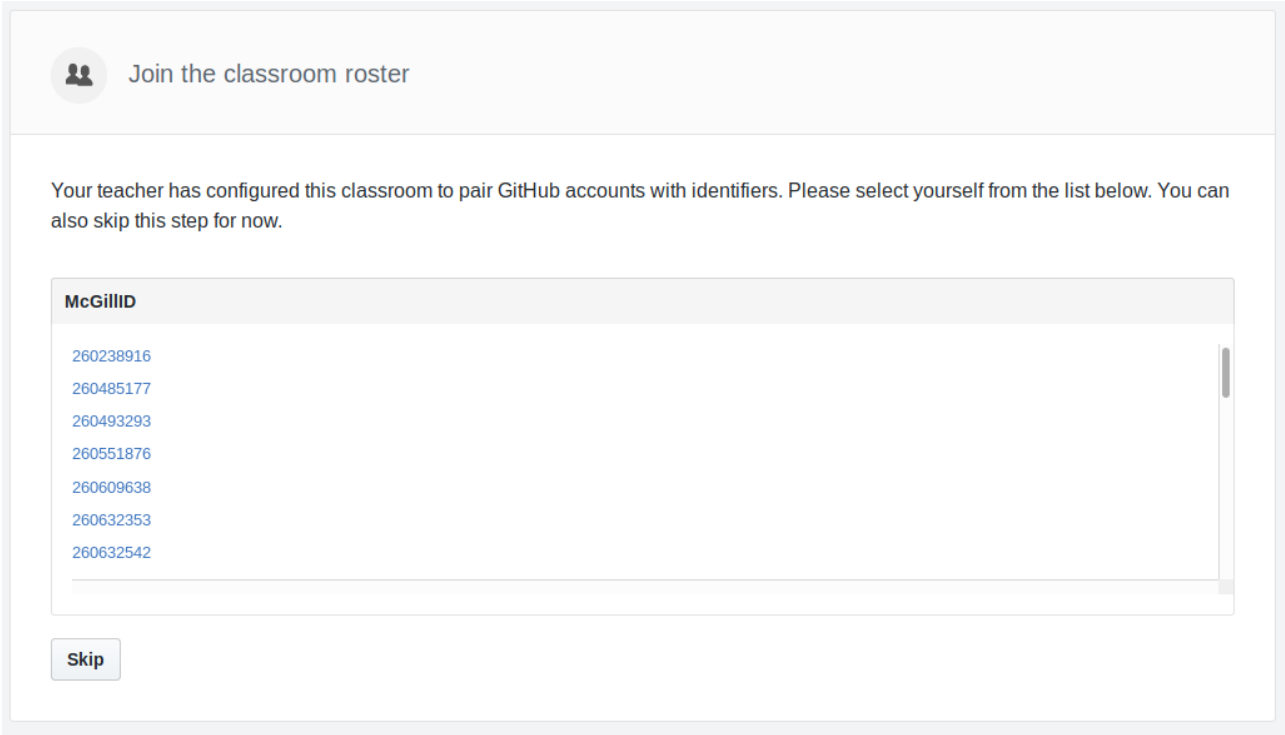
Sections of the tutorial will continuously be published at this web page.

1. Preliminaries

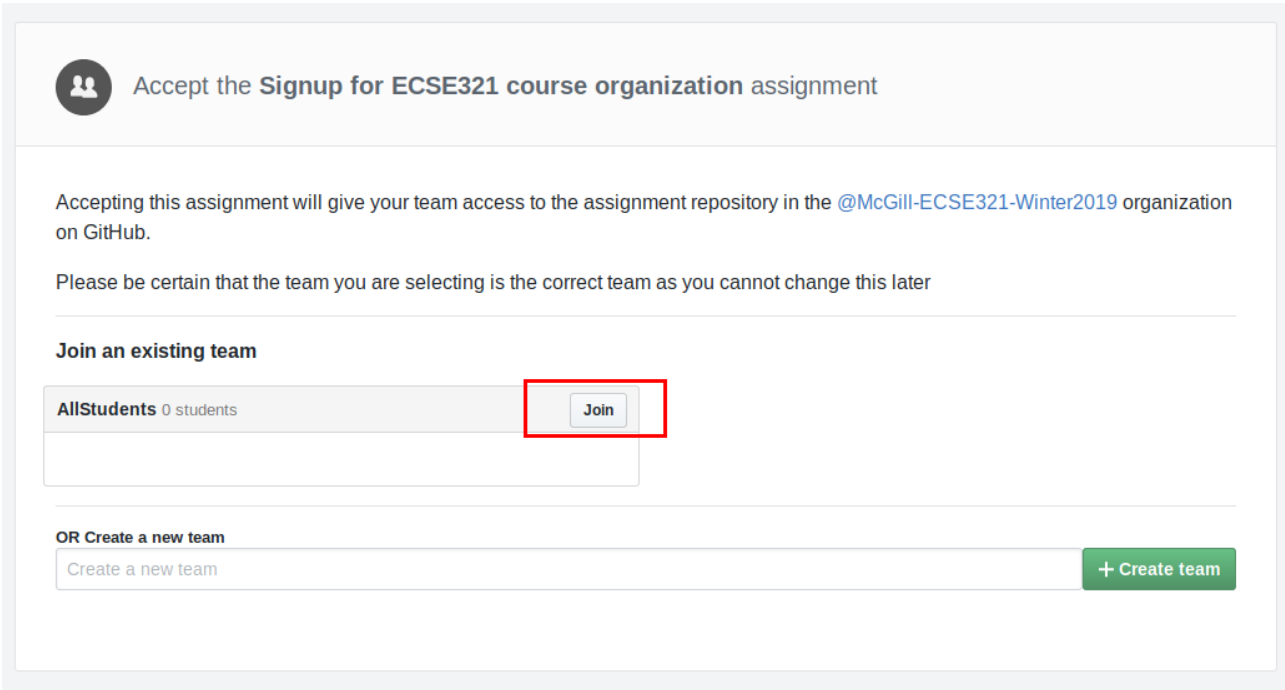
1.1. Getting Started

Steps for signing up for GitHub classroom:

1. Log in/Register on GitHub.
2. Open link https://classroom.github.com/g/o9gWNZis
3. Select your McGill ID from the list



4. Join team *All students*

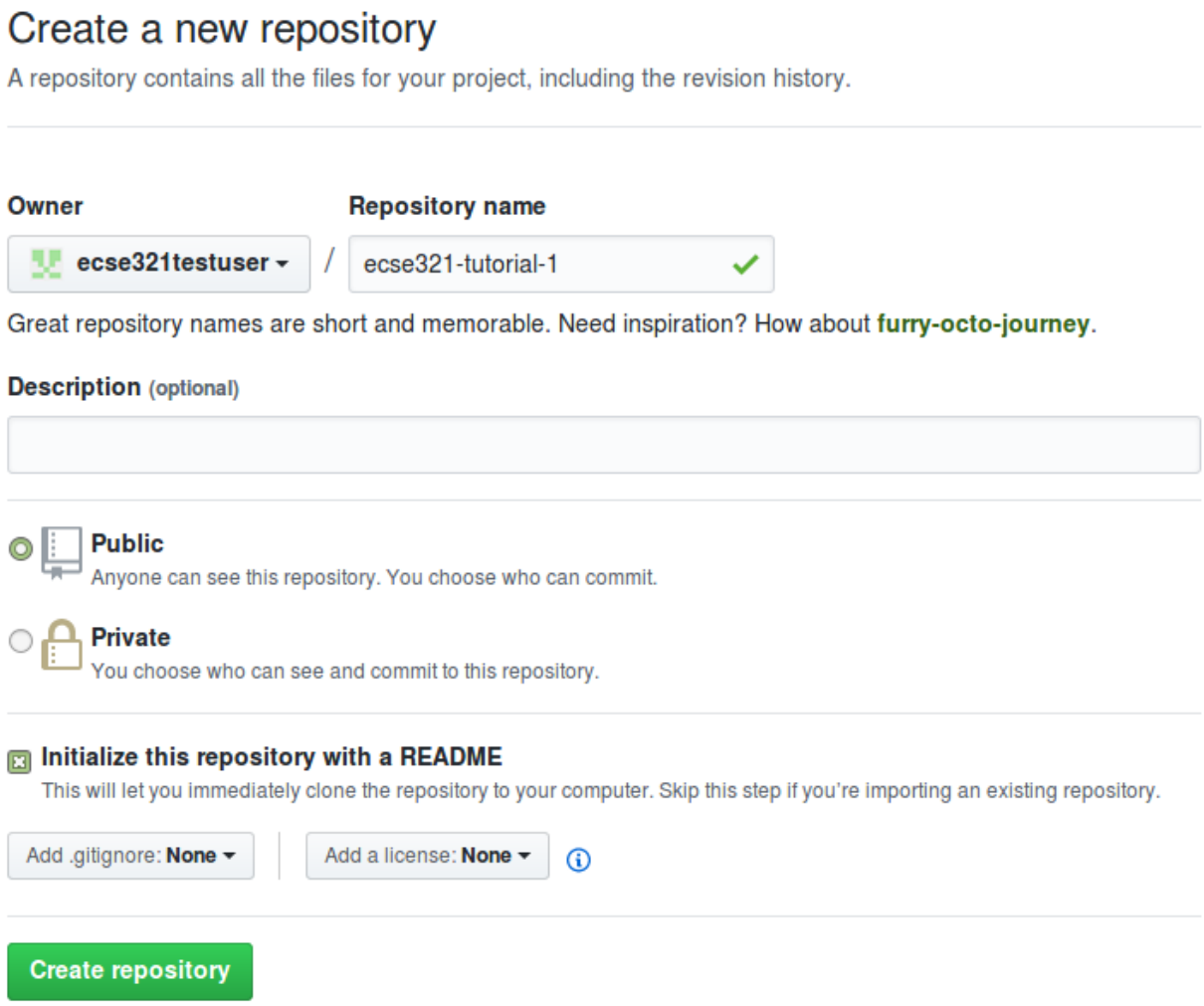


1.2. Project Management Tools for Agile Development

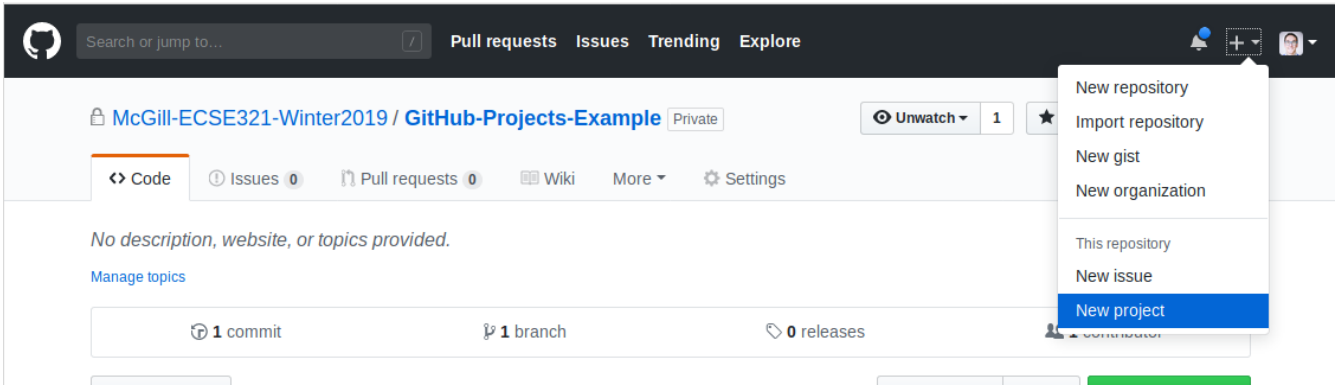
1.2.1. GitHub Projects

First, we create a new repository under everyone's own account to demonstrate the basic features of "GitHub Projects".

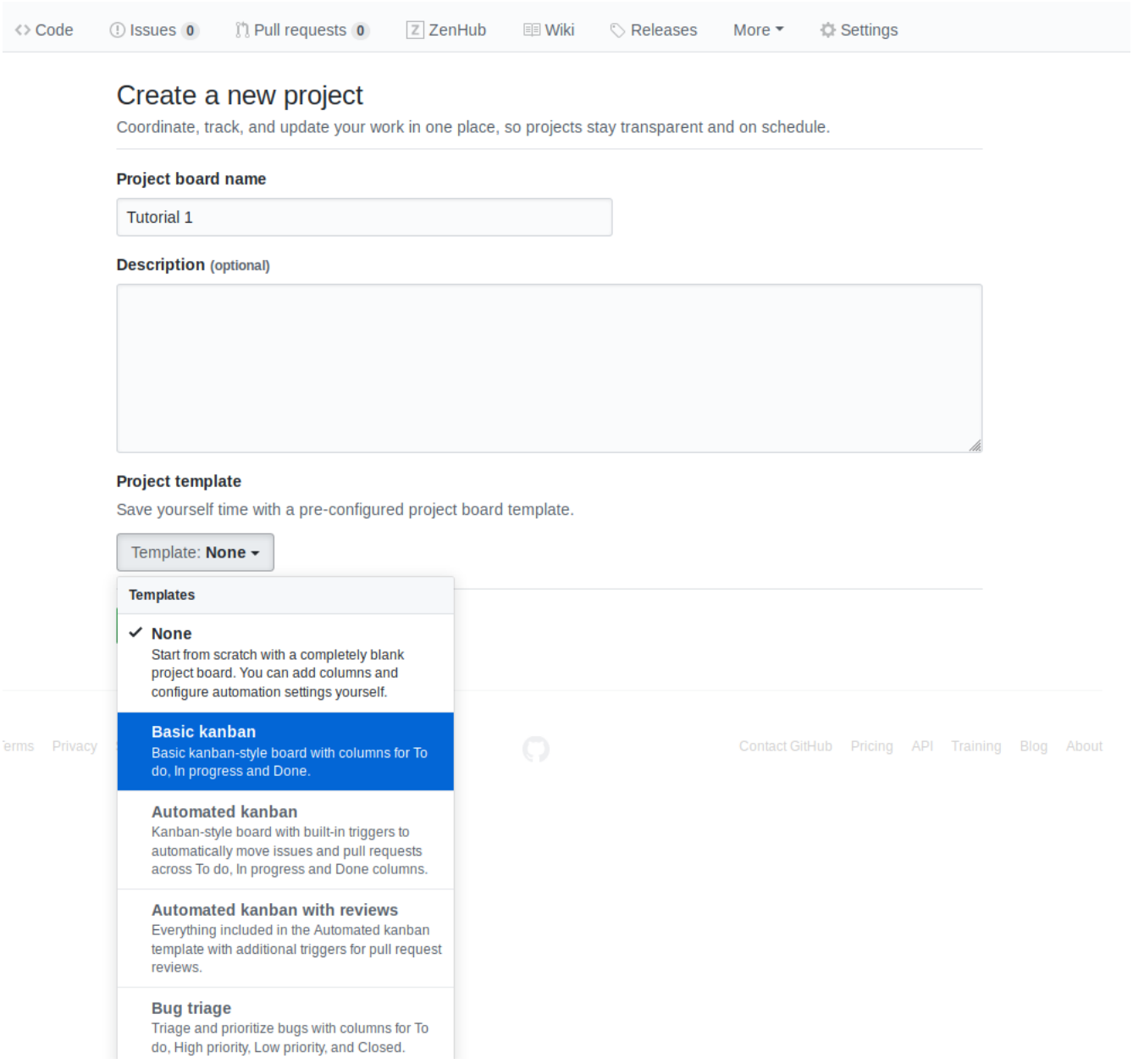
1. Visit https://github.com/ then click on *New repository* (green buttom on the right).
2. Set your user as the owner of the repository.
3. Give a name for the repository (e.g., ecse321-tutorial-1), leave it *public*, then check *Initialize this repository with a README*. Click on *Create repository* afterwards. At this point the remote repository is ready to use.



Once the repository is ready, associate a new GitHub Project and see how their features work. Create a project:

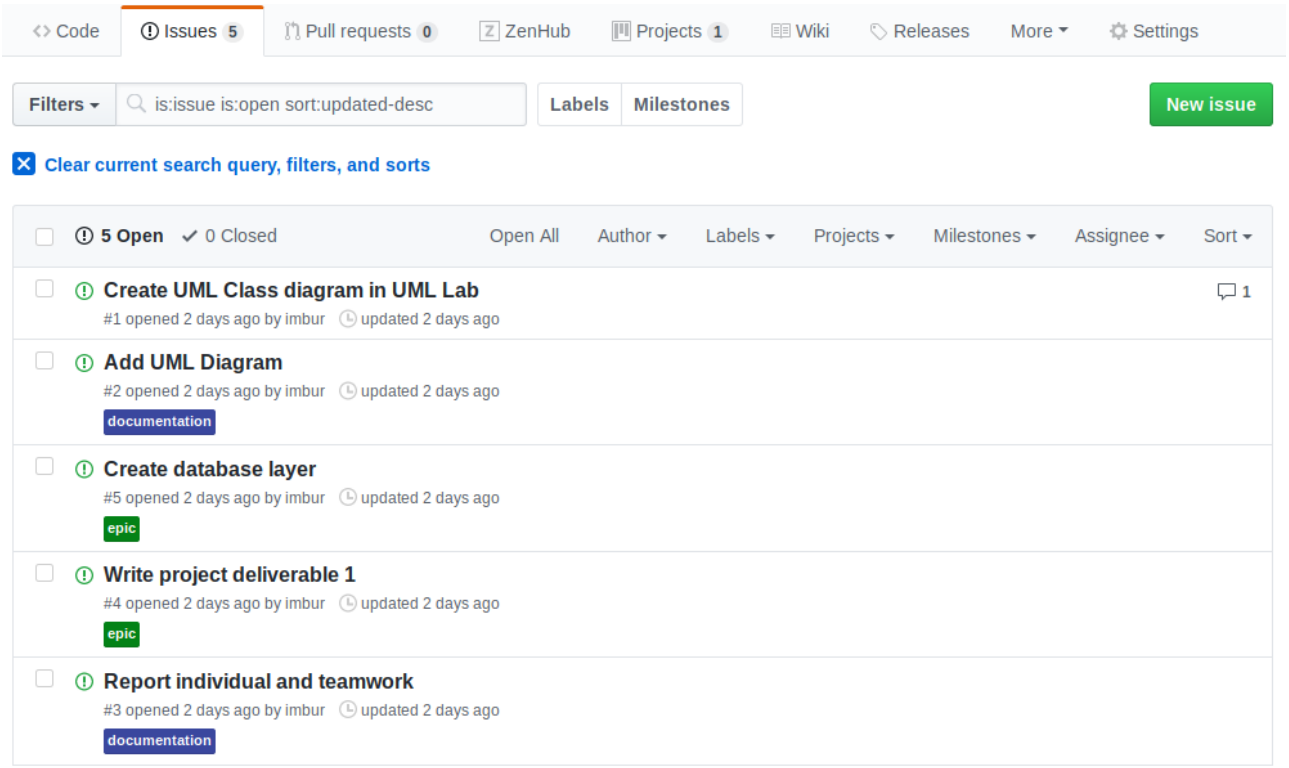


Select Basic Kanban project style:

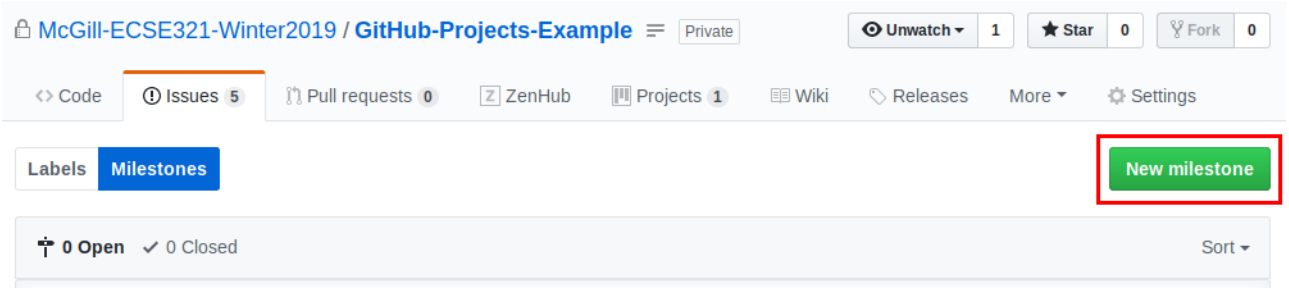


Tasks to complete:

1. Create a few issues to outline the tasks for the first deliverable. Assign them appropriate labels and add yourself as the assignee!



2. Create a milestone for the issues.



3. Create cards from the issues on the project board.
4. See how GitHub track the project progress as you move the cards from the different columns.

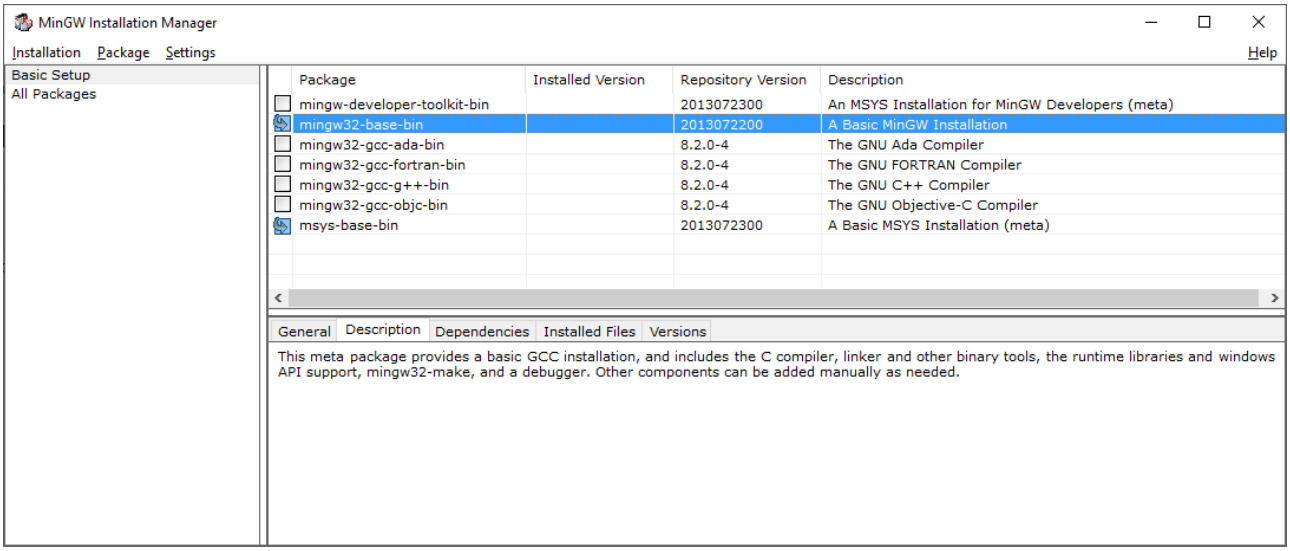
1.3. Command Line Basics

This section shows a few handy terminal commands.

1.3.1. Windows prerequisites

This step can be skipped if you are using MacOS or Linux. However, if you are using Windows, you need to have a terminal that supports the execution of basic Linux commands. Such programs are Git Bash or MinGW, for example. You can find below a few helper steps to get MinGW running on your system.

1. Get the MinGW installer from here
2. Install it to wherever you like, the default installation folder is *C:\MinGW*
3. Once the setup finishes, open the MinGW Installation Manager
4. Select the two packages for installation as shown in the figure below



5. Click on *Installation/Apply Changes*. This will take a few moments to fetch and install the required packages.
6. You can open a terminal window by running the executable *C:\MinGW\msys\1.0\bin\bash.exe*

1.3.2. Basic file system operaions

1. Open a terminal, and try the following commands:
   - `pwd`: prints the present working directory
     Example:

     ```
     $ pwd
     /home/ecse321
     ```

   - `ls`: lists the content of a given folder
     Example:

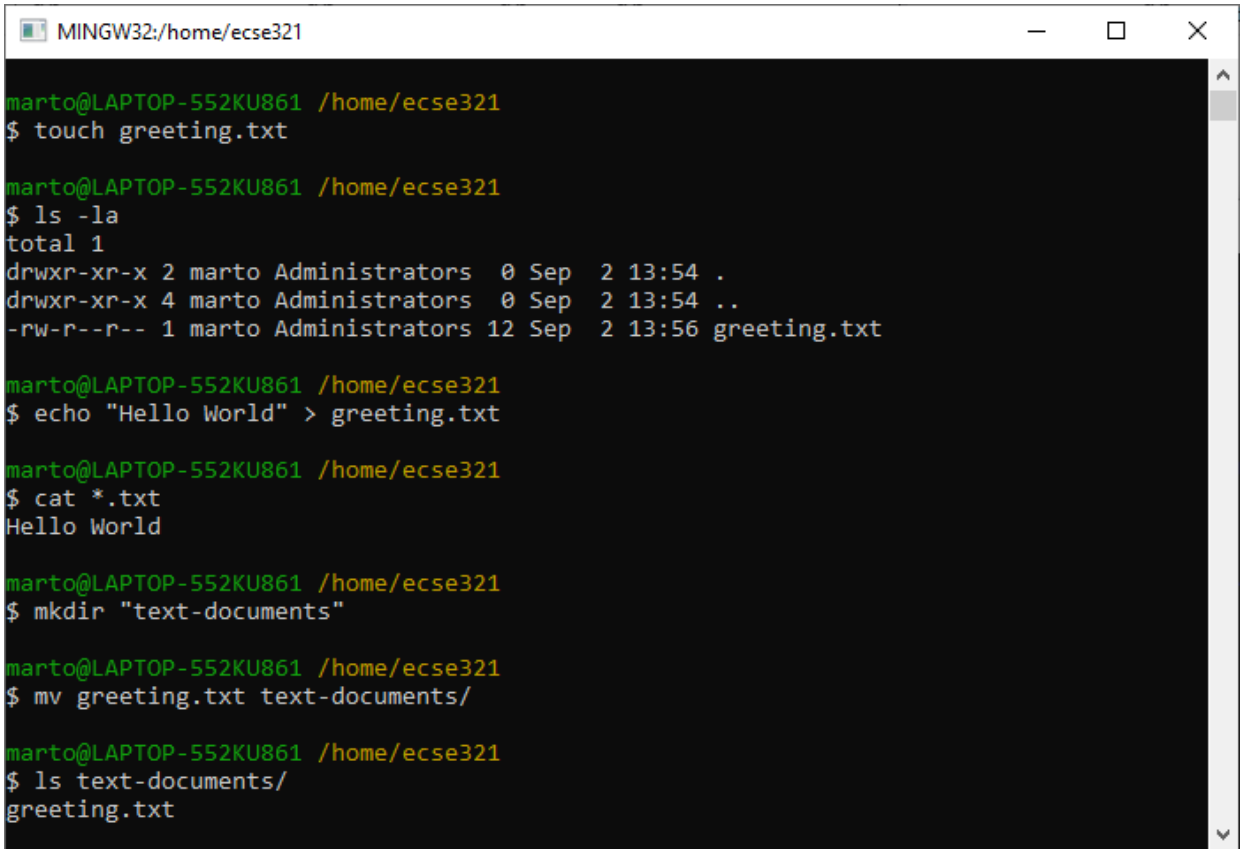
```
$ ls /home
ecse321 guest-user admin
```

- `cd`: navigates the file system
  Example:

```
$ cd ..
$ pwd
/home
$ cd ecse321
$ pwd
/home/ecse321
```

> **NOTE** The following steps will include images that illustrate the commands and their output to prevent easy copy-paste. Sorry! :)

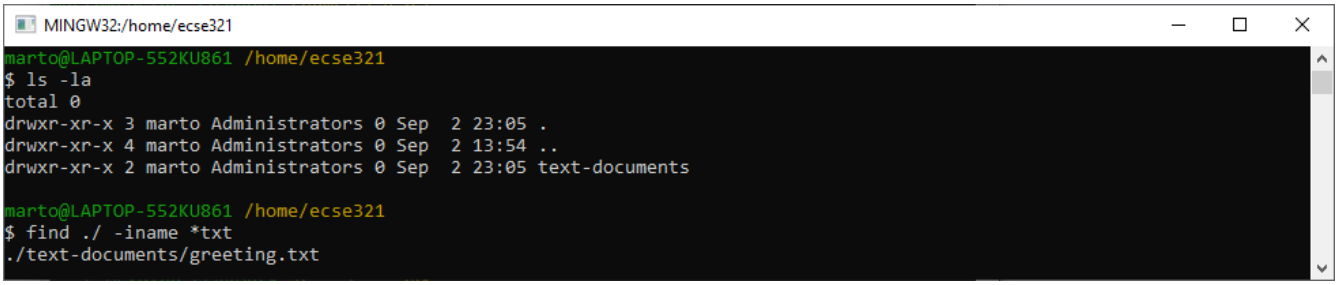
2. Creating files and reading/writing their contents
   - `touch`: creates a file
   - `mkdir`: creates a directory
   - `mv`: moves a file (or directory) from its current location to a target location
   - `echo`: prints a string
   - `cat`: prints the contents of a file
     Example:



1.3.3. Finding files

The versatile `find` command allows us to find files based on given criteria. Take look at its manual page with `man find`!

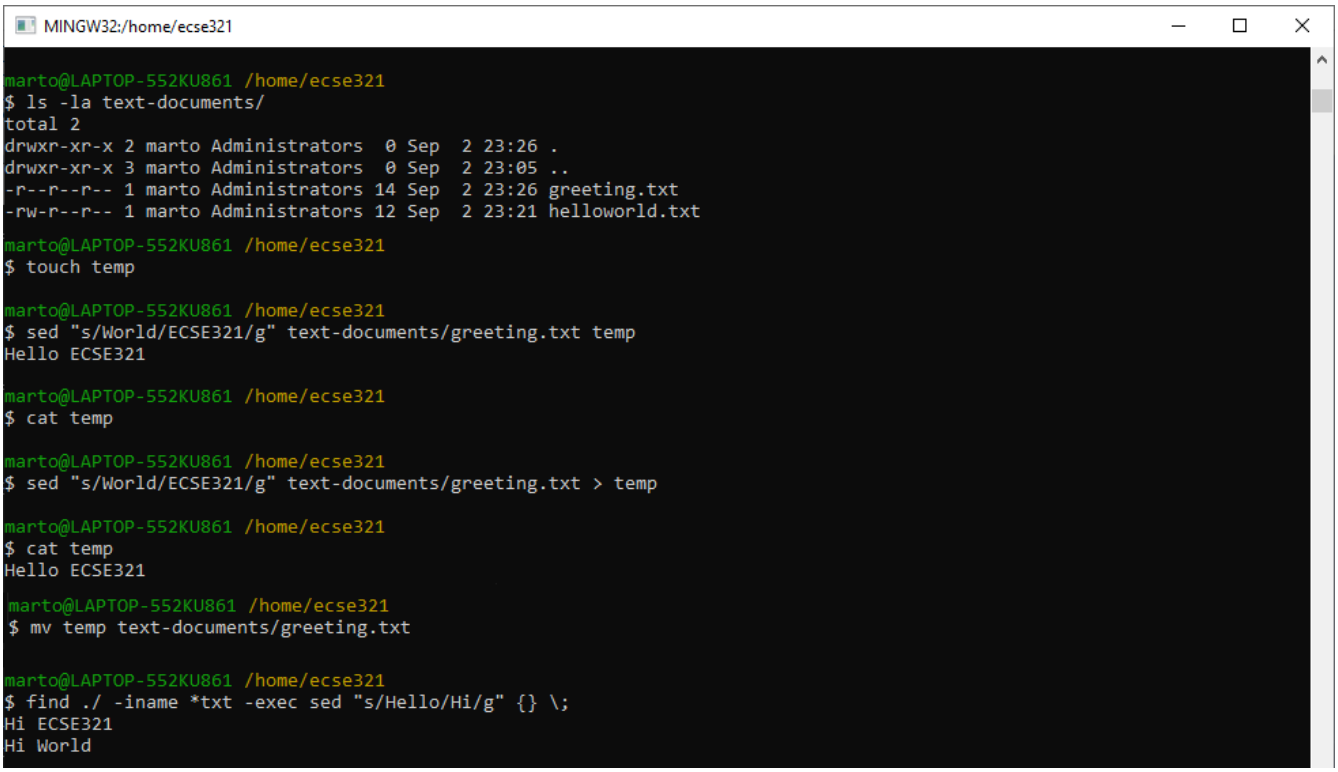
Example:



1.3.4. Batch file operations

- `sed`: stream editor; changes a given string to a replacement

Combining `find` with an additional command (e.g., `sed`) can greatly speed up your repetitive tasks. Example:



NOTE	The file *helloworld.txt* in the example is initially a copy of *greeting.txt*.

1.3.5. Some additional useful commands

- `rm`: removes a file
- `cp -r`: copies a directory recursively with its contents
- `rmdir`: remove an empty directory

- `rm -rf`: force to recursively delete a directory (or file) and all its contents
- `nano`: an easy-to-use text editor (not available by default in MinGW)
- `grep`: finds matches for a string in a given stream of characters
- `ag`: takes a string as argument and searches through the contents of files recursively to find matches of the given string (this tool is included in the *silversearcher-ag* package)

1.4. Git and GitHub

1.4.1. Installing Git

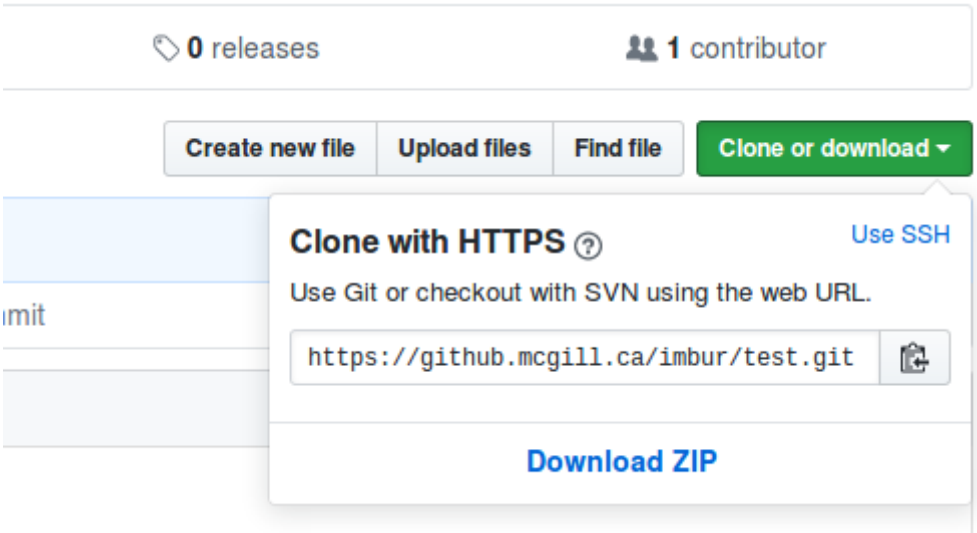
Install the Git version control system (VCS) from https://git-scm.com/downloads.

1.4.2. Creating a remote git repository on GitHub

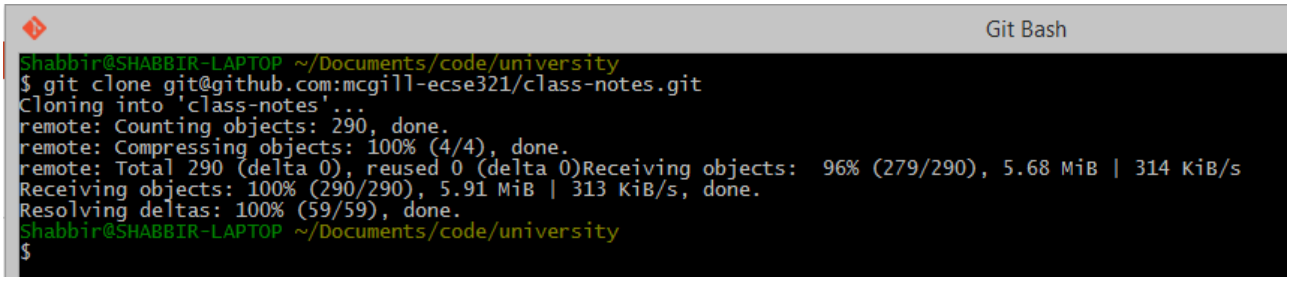
1. Go to https://github.com/new
2. Set *test* as the name of the repository
3. Check the checkbox *Initialize this repository with a README*
4. Click on create repository

1.4.3. Cloning to a local repository

1. Open up a terminal (Git bash on Windows).
2. Navigate to the designated target directory (it is typical to use the `git` folder within the home directory for storing Git repositories, e.g., `cd /home/username/git`).
3. Using a Git client, clone this newly created *test* repository to your computer. First, get the repository URL (use HTTPS for now).



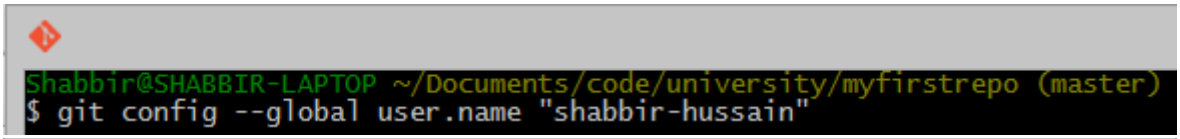
Then, issue `git clone https://url/of/the/repository.git`
You should get an output similar to this:

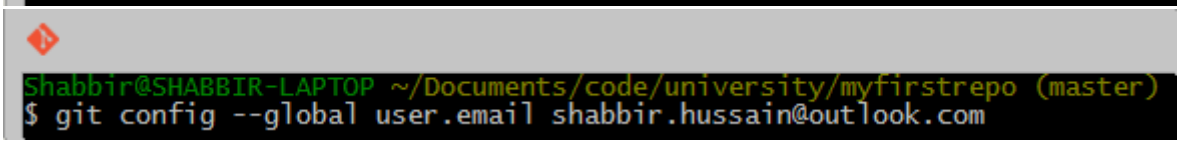


4. Verify the contents of the *working copy* of the repository by `ls -la ./test`. The *.git* folder holds version information and history for the repository, while the *README.md* is an auto-generated text file by GitHub.

1.4.4. Git basics

1. Open up a terminal and configure username and email address. These are needed to identify the author of the different changes.

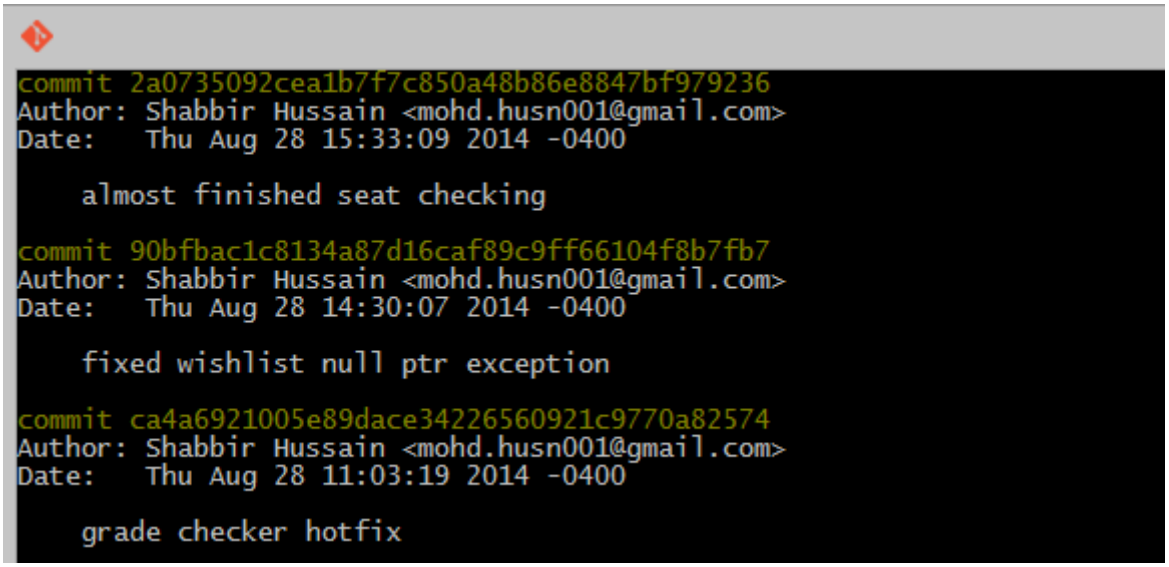




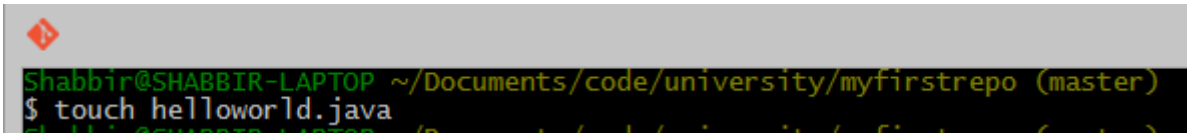
Glossary — Part 1:

   - **Git** is your version control software
   - **GitHub** hosts your repositories
   - A **repository** is a collection of files and their history
   - A **commit** is a saved state of the repository

2. Enter the working directory, then check the history by issuing `git log`. Example output:



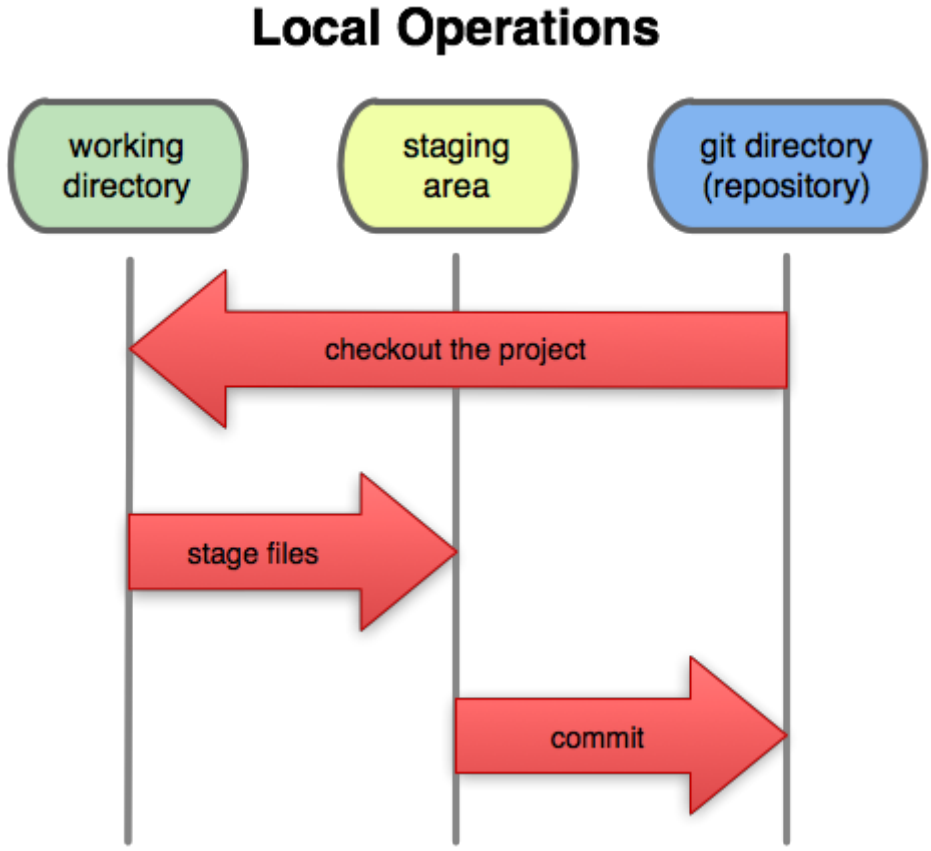
3. Adding and commiting a file: use the `git add` and `git commit` commands.



```
Shabbir@SHABBIR-LAPTOP ~/Documents/code/university/myfirstrepo (master)
$ git add helloworld.java
```

```
Shabbir@SHABBIR-LAPTOP ~/Documents/code/university/myfirstrepo (master)
$ git commit -m 'added hello world file to the project'
[master (root-commit) f4a1ddc] added hello world file to the project
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 helloworld.java
```

The effect of these commands are explained on the figure below:



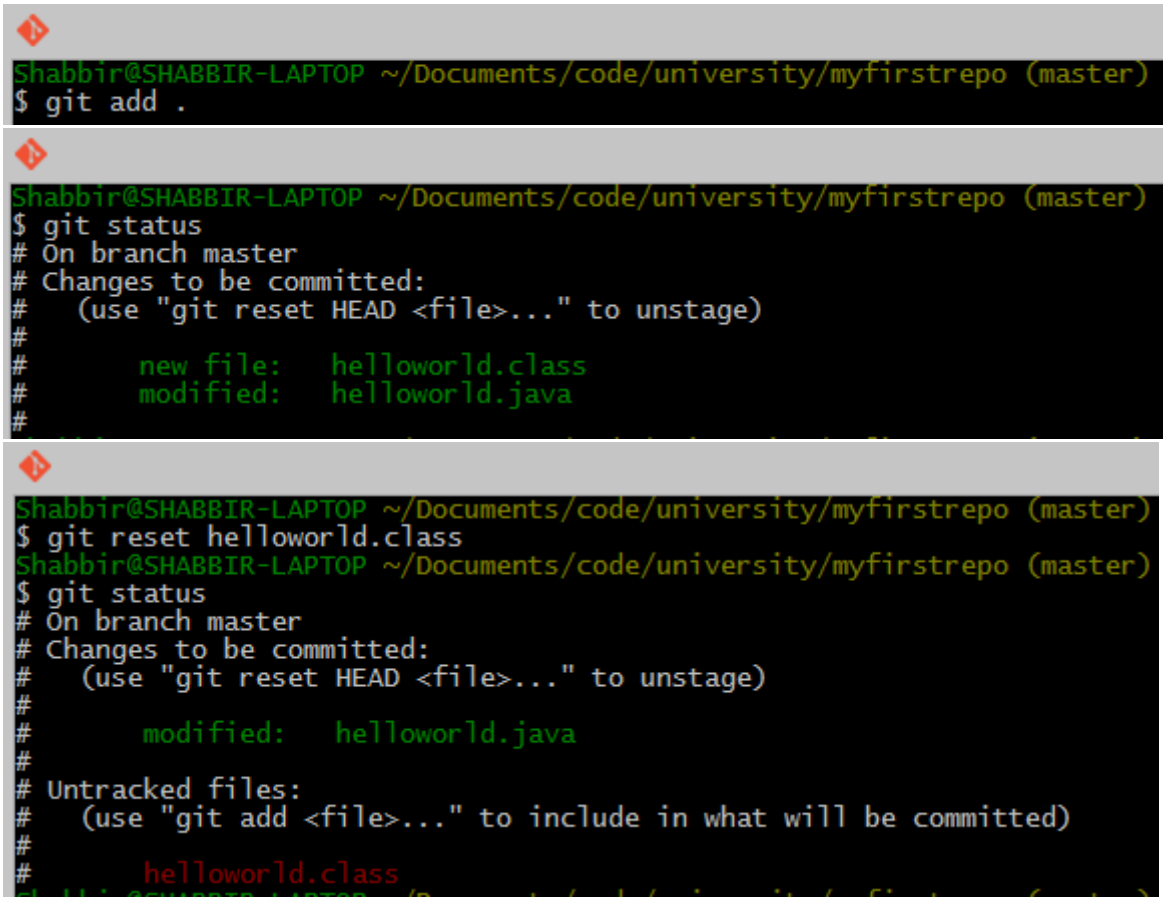
Glossary — Part 2:

- **Working Directory**: files being worked on right now
- **Staging area**: files ready to be committed
- **Repository**: A collection of commits

4. Checking current status is done with `git status`.

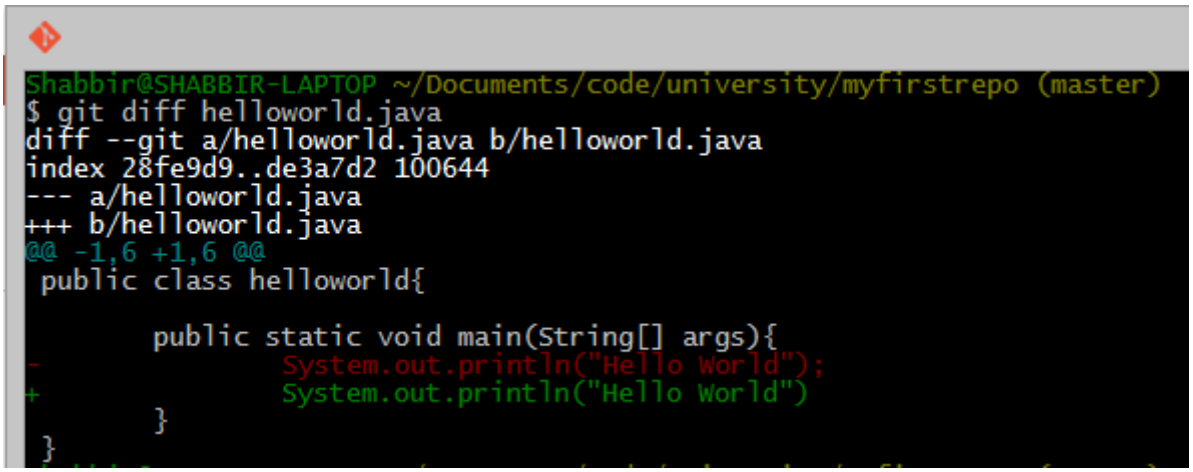
```
Shabbir@SHABBIR-LAPTOP ~/Documents/code/university/myfirstrepo (master)
$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   helloworld.java
#
no changes added to commit (use "git add" and/or "git commit -a")
```

5. Staging and unstaging files: use `git add` to add and `git reset` to remove files from the staging area.

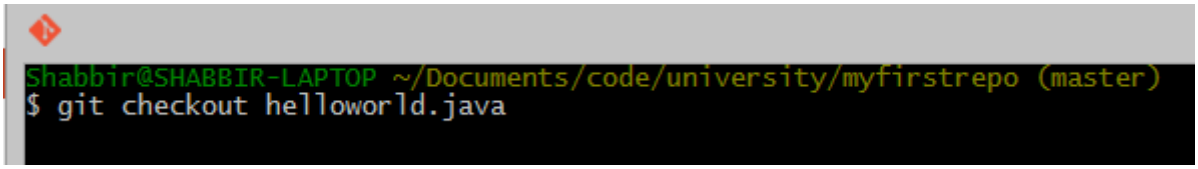


CAUTION	Only staged files will be included in the next commit.

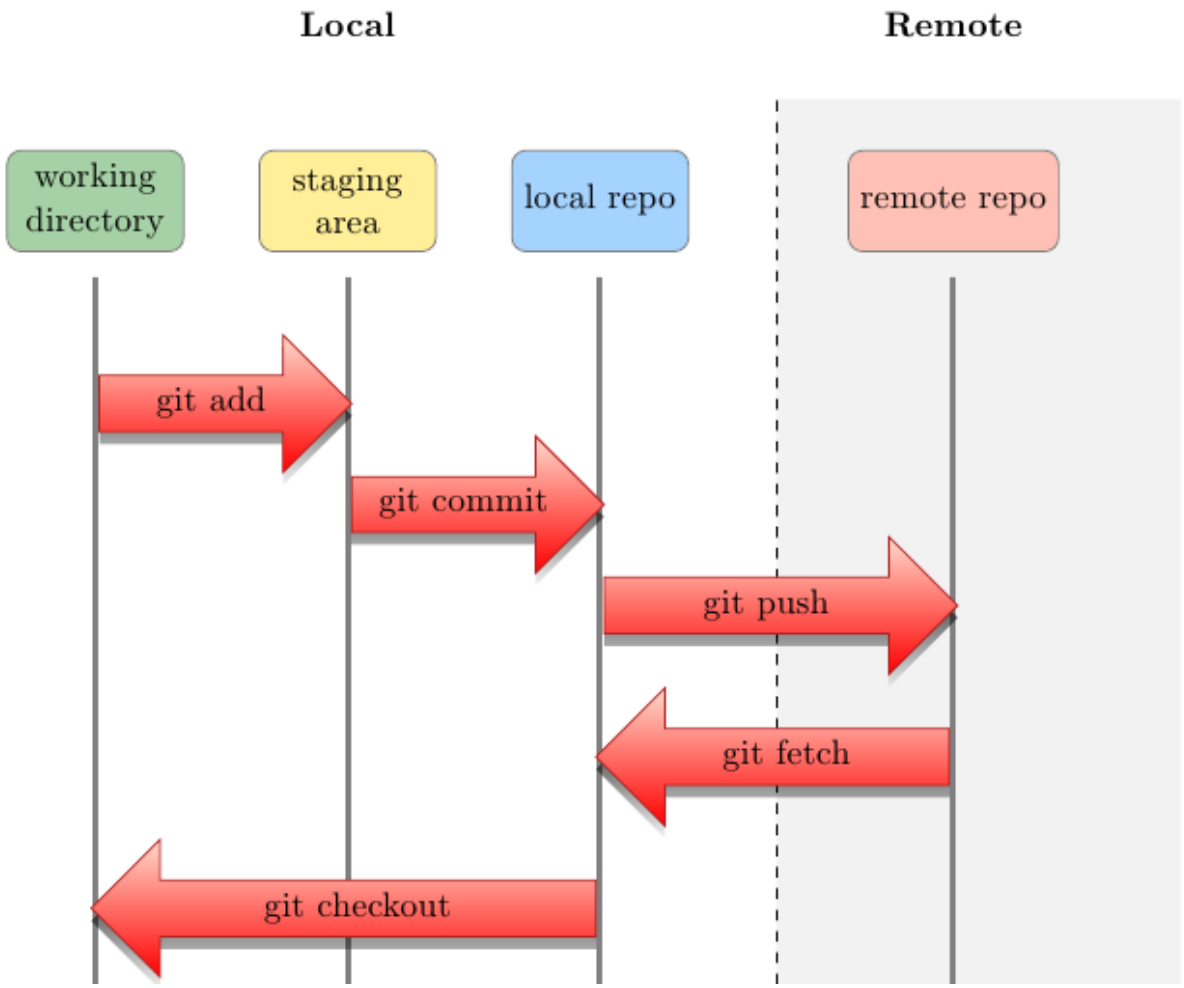
6. To display detailed changes in unstaged files use `git diff`, while use `git diff --staged` to show changes within files staged for commit.



7. Reverting to a previous version is done using `git checkout`.

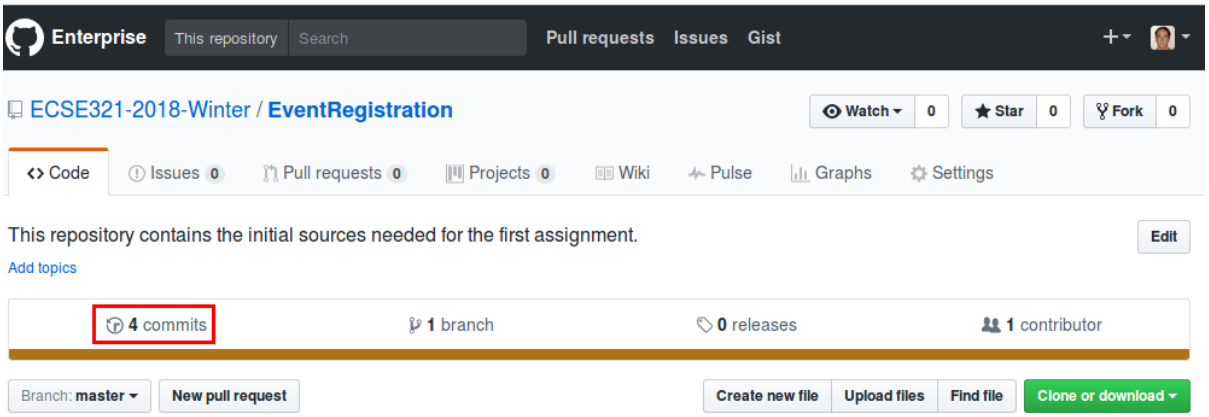


8. The commands `git pull` (or the `git fetch` + `git rebase` combination) and `git push` are used to synchronize local and remote repositories.

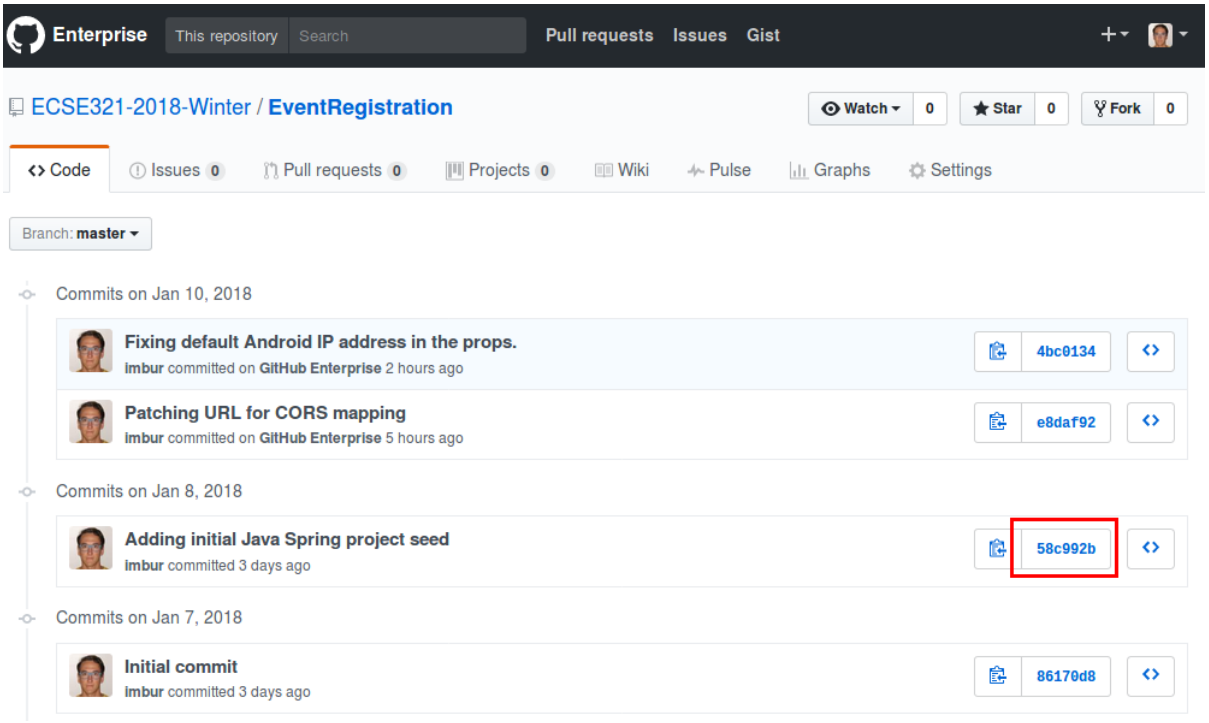


1.4.5. Browsing commit history on GitHub

1. You can browse pushed commits in the remote repository online using GitHub. You can select the *commits* menu for a repository.



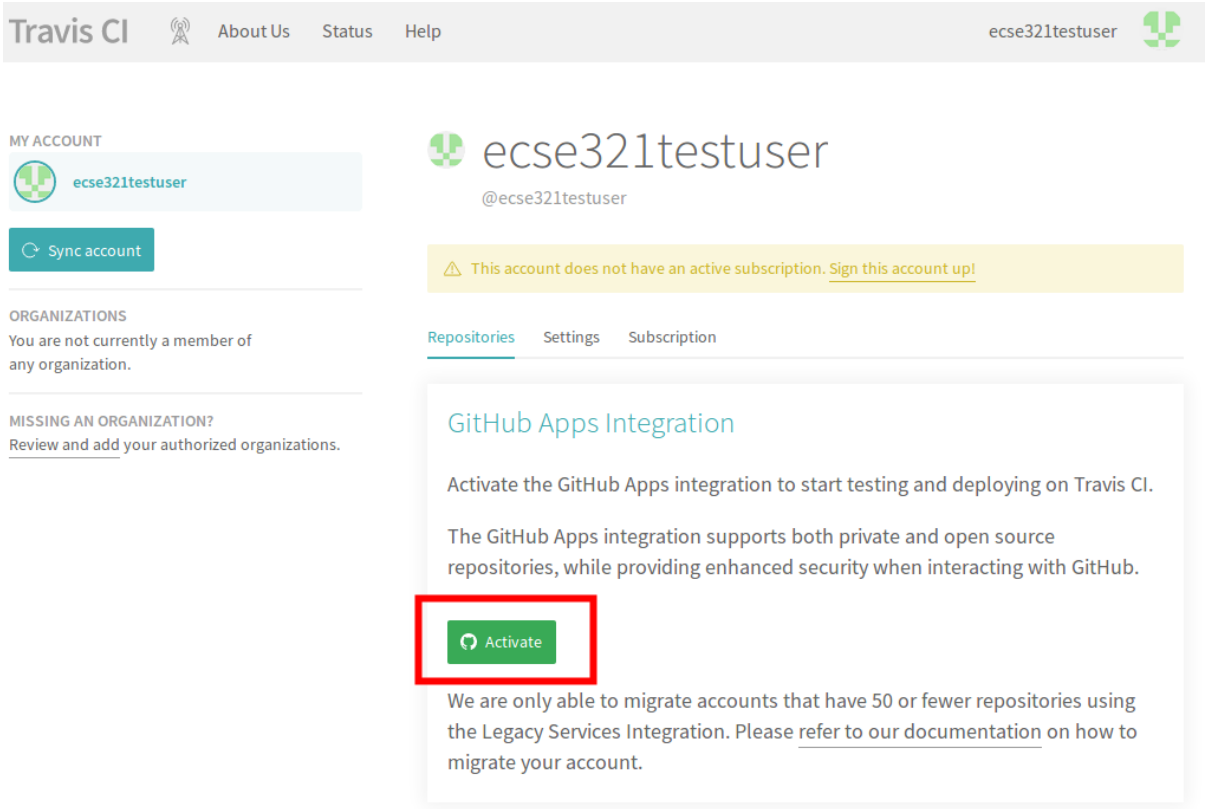
To get a link for a specific commit, click on the button with the first few characters of the hash of the commit.



The source for most of the images in the Git documentation: https://github.com/shabbir-hussain/ecse321tutorials/blob/master/01-githubTutorial1.pptx

1.5. Travis CI

1. Go to https://travis-ci.com/, click on Sign up with GitHub.
2. Click on the green authorize button at the bottom of the page.
3. Activate Travis-CI on your GitHub account



4. Select the repositories you want to build with Travis (make sure to include your repository that you created for this tutorial). You can modify this setting anytime later as well.
5. In your working copy of your repository, create a default Gradle java project.
   - Make sure you have Gradle installed (`gradle --version`).
   - Issue `gradle init --type java-library`
   - Add a `.gitignore` to ignore generated resources by Git:

     ```
     .gradle/
     build/
     ```

   - Make sure your application is compiling by running `gradle build`
6. Create a file called `.travis.yml`:

   ```
   language: java
   script:
   - gradle build
   ```

7. Commit and push your work. If everything is set up correctly, the build should trigger and

Travis should run your build using Gradle.