Lab 1: C++ Primer 1

Instructor: Beom Heyn Kim

beomheynkim@hanyang.ac.kr

Department of Computer Science

Outline

- About Labs
- Set up the Project Environment
- Project Overview
- Assignment

About Labs

- In this semester, you are going to implement solutions for 3 programming projects incrementally during labs
  - Project 1: C++ Primer
  - Project 2: Buffer Pool
  - Project 3: B+Tree
- We will use the BusTub database management system (DBMS) for all the programming projects
- All of the code in this programming assignment must be written in C++ (specifically C++17). It is generally sufficient to know C++11.
- Programming projects are single-person projects that will be completed individually (i.e. no groups).

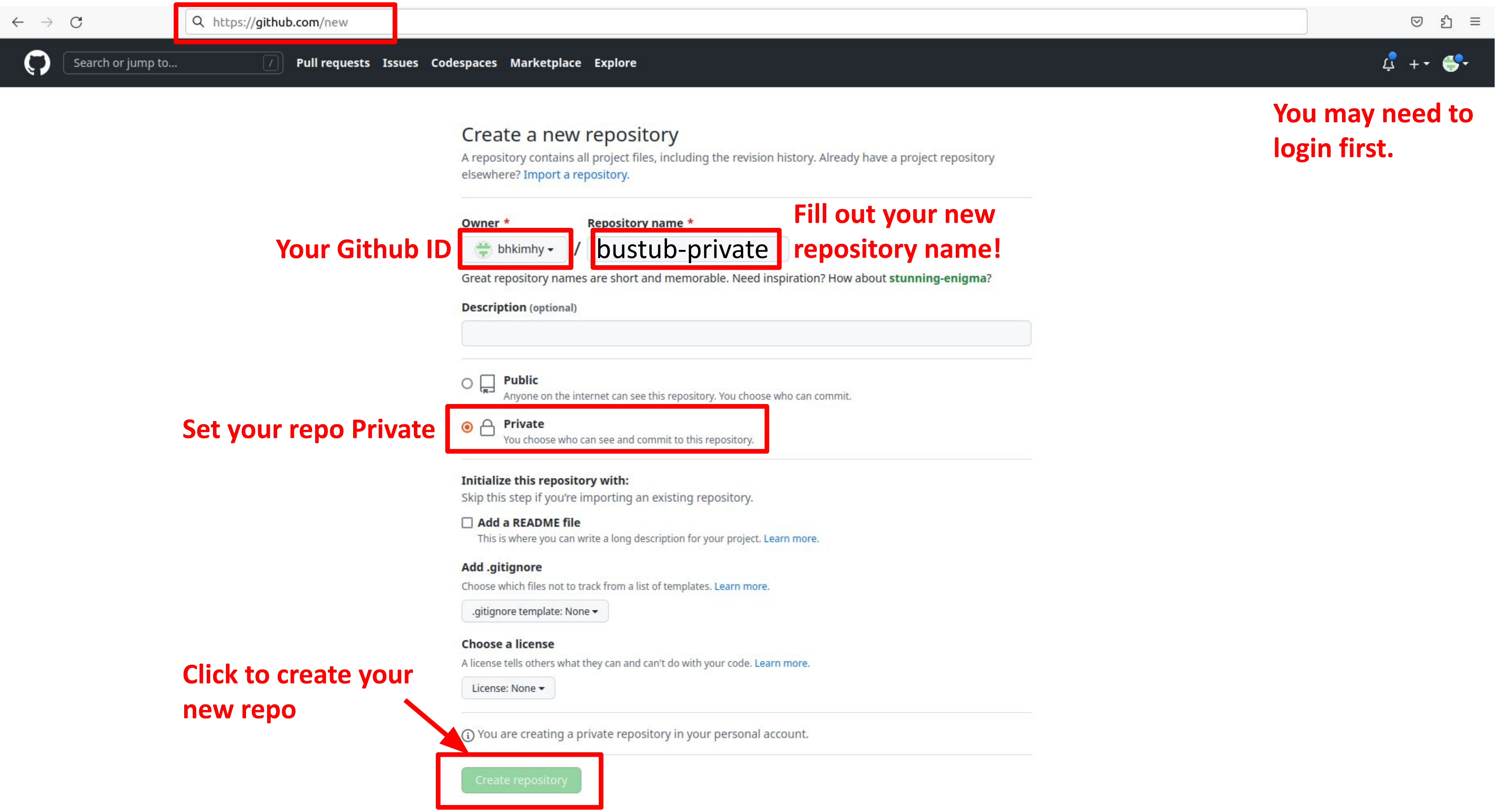
NOTE: You will need to submit your code on LMS by each specified deadline, which is usually 11:59PM on Saturday followed by the last lab of each project.

Outline

- About Labs
- Set up the Project Environment
- Project Overview
- Assignment

Git Repository Setup

Go to https://github.com/new



You may need to login first.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Your Github ID

Owner * bhkimhy / Repository name * bustub-private

Fill out your new repository name!

Great repository names are short and memorable. Need inspiration? How about **stunning-enigma**?

Description (optional)

○ **Public**
Anyone on the internet can see this repository. You choose who can commit.

Set your repo Private

● **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

□ **Add a README file**
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.

.gitignore template: None

Choose a license
A license tells others what they can and can't do with your code. Learn more.

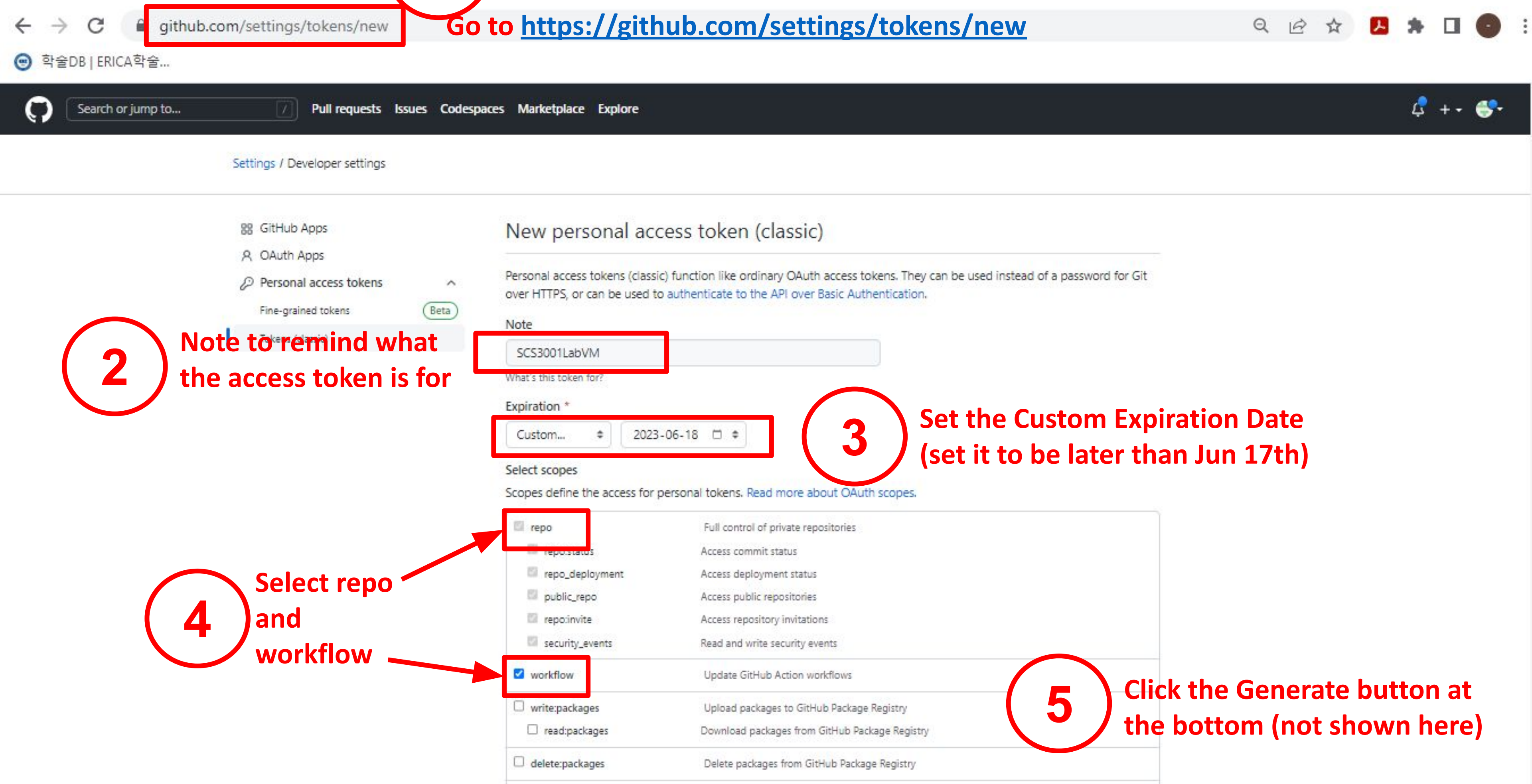
License: None

Click to create your new repo

You are creating a private repository in your personal account.

Git Repository Setup

- Setup your personal access tokens which will be used to authorize your access to your private repo.



github.com/settings/tokens/new

1 Go to https://github.com/settings/tokens/new

Settings / Developer settings

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

SCS3001LabVM

What's this token for?

Expiration *

Custom... 2023-06-18

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

☑ repo	Full control of private repositories
☑ repo:status	Access commit status
☑ repo_deployment	Access deployment status
☑ public_repo	Access public repositories
☑ repo:invite	Access repository invitations
☑ security_events	Read and write security events
☑ workflow	Update GitHub Action workflows
□ write:packages	Upload packages to GitHub Package Registry
□ read:packages	Download packages from GitHub Package Registry
□ delete:packages	Delete packages from GitHub Package Registry

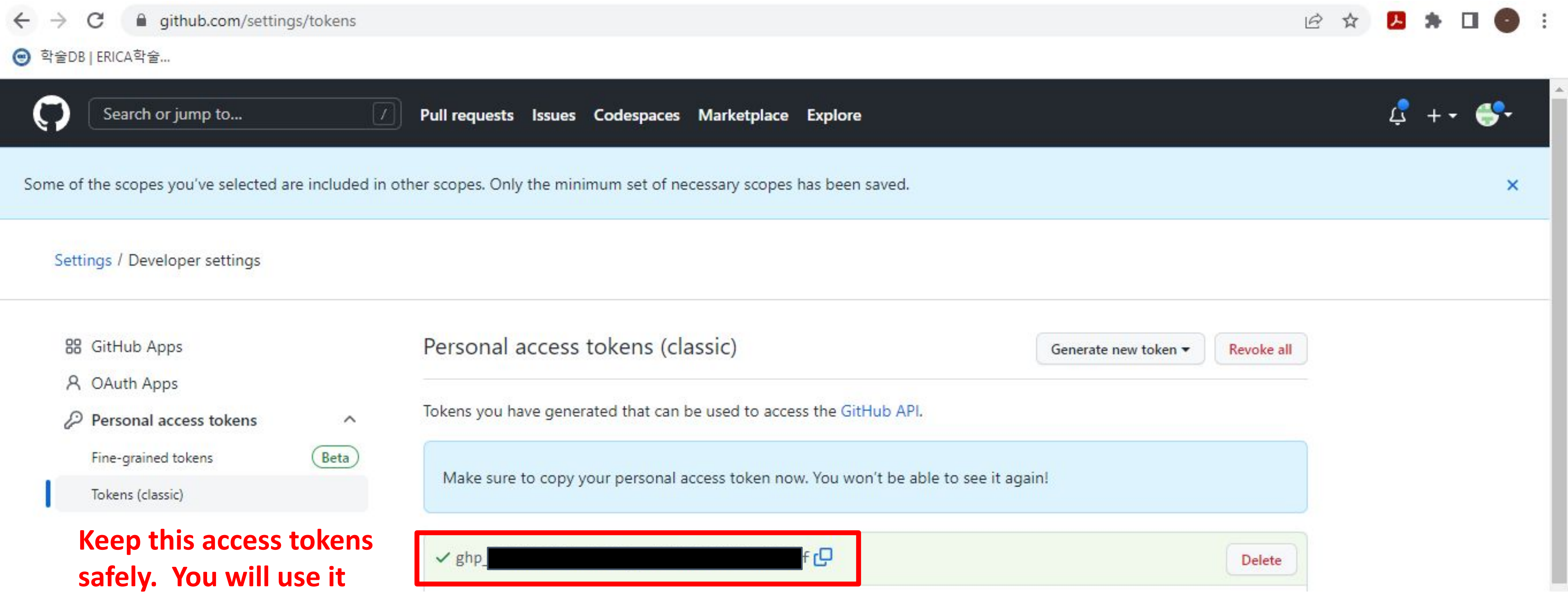
2 Note to remind what the access token is for

3 Set the Custom Expiration Date (set it to be later than Jun 17th)

4 Select repo and workflow

5 Click the Generate button at the bottom (not shown here)

Git Repository Setup



Some of the scopes you've selected are included in other scopes. Only the minimum set of necessary scopes has been saved.

Personal access tokens (classic)

Tokens you have generated that can be used to access the GitHub API.

Make sure to copy your personal access token now. You won't be able to see it again!

✓ ghp_[redacted]f

Keep this access tokens safely. You will use it instead of password.

Git Repository Setup

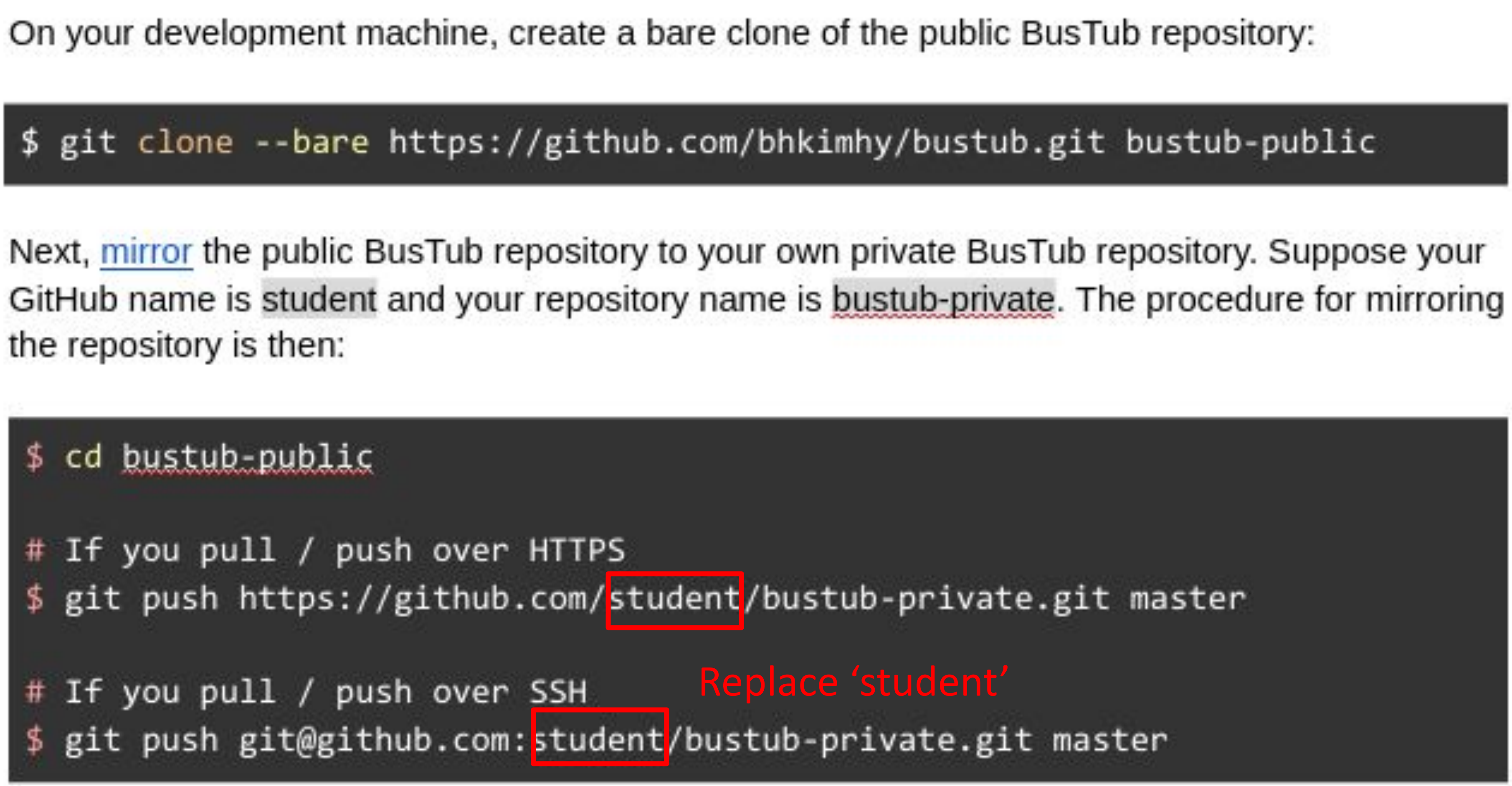
- Start up your VirtualBox VM you installed during the last lecture and log in
- Start the terminal
- Install git

```
$ sudo apt install git
```

- Create and move to the directory to work with (e.g. /home/student)

```
$ mkdir labs
$ cd labs
```

Git Repository Setup



On your development machine, create a bare clone of the public BusTub repository:

```
$ git clone --bare https://github.com/bhkimhy/bustub.git bustub-public
```

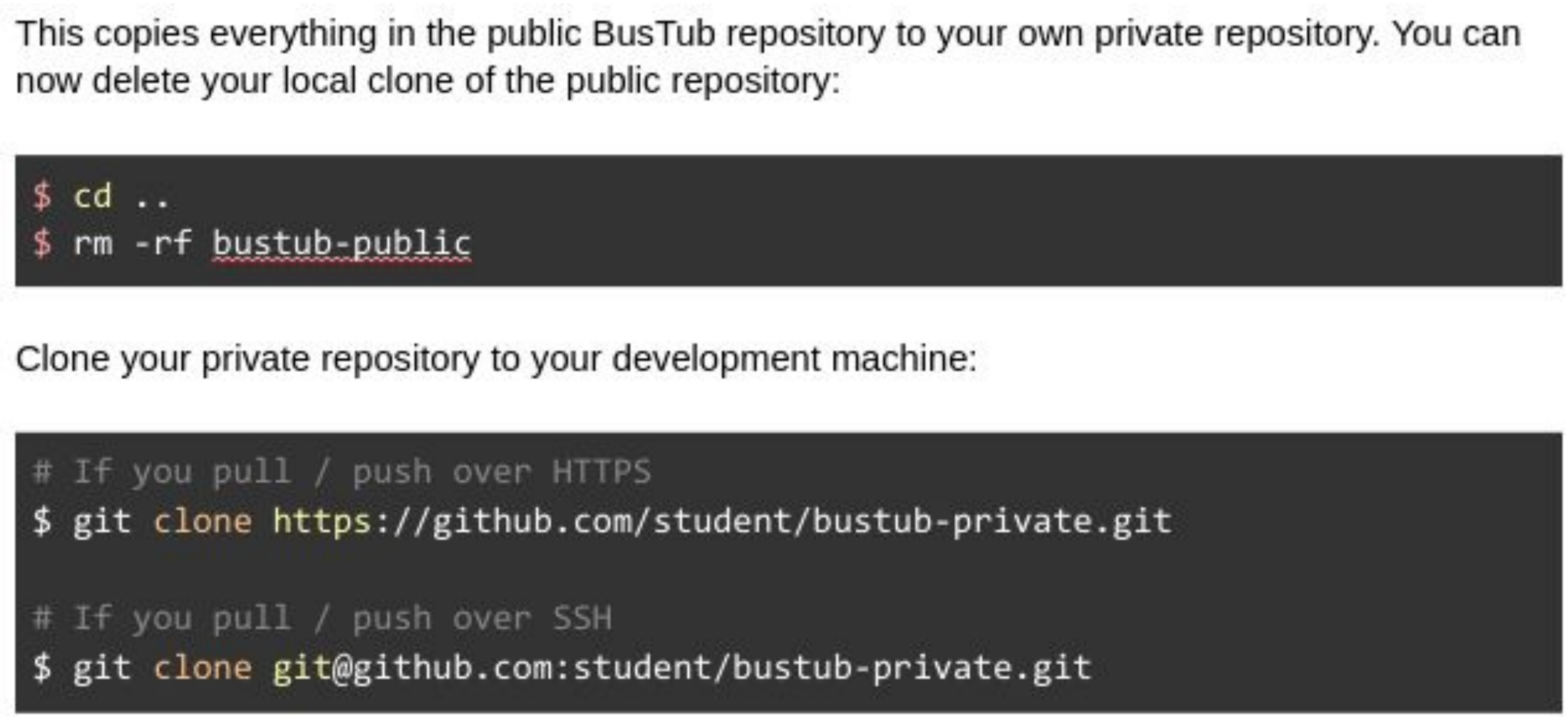
Next, mirror the public BusTub repository to your own private BusTub repository. Suppose your GitHub name is student and your repository name is bustub-private. The procedure for mirroring the repository is then:

```
$ cd bustub-public

# If you pull / push over HTTPS
$ git push https://github.com/student/bustub-private.git master

# If you pull / push over SSH
$ git push git@github.com:student/bustub-private.git master
```

Replace 'student'

To make Git store the username credentials, do:
$ git config –global credential.helper store

Git Repository Setup



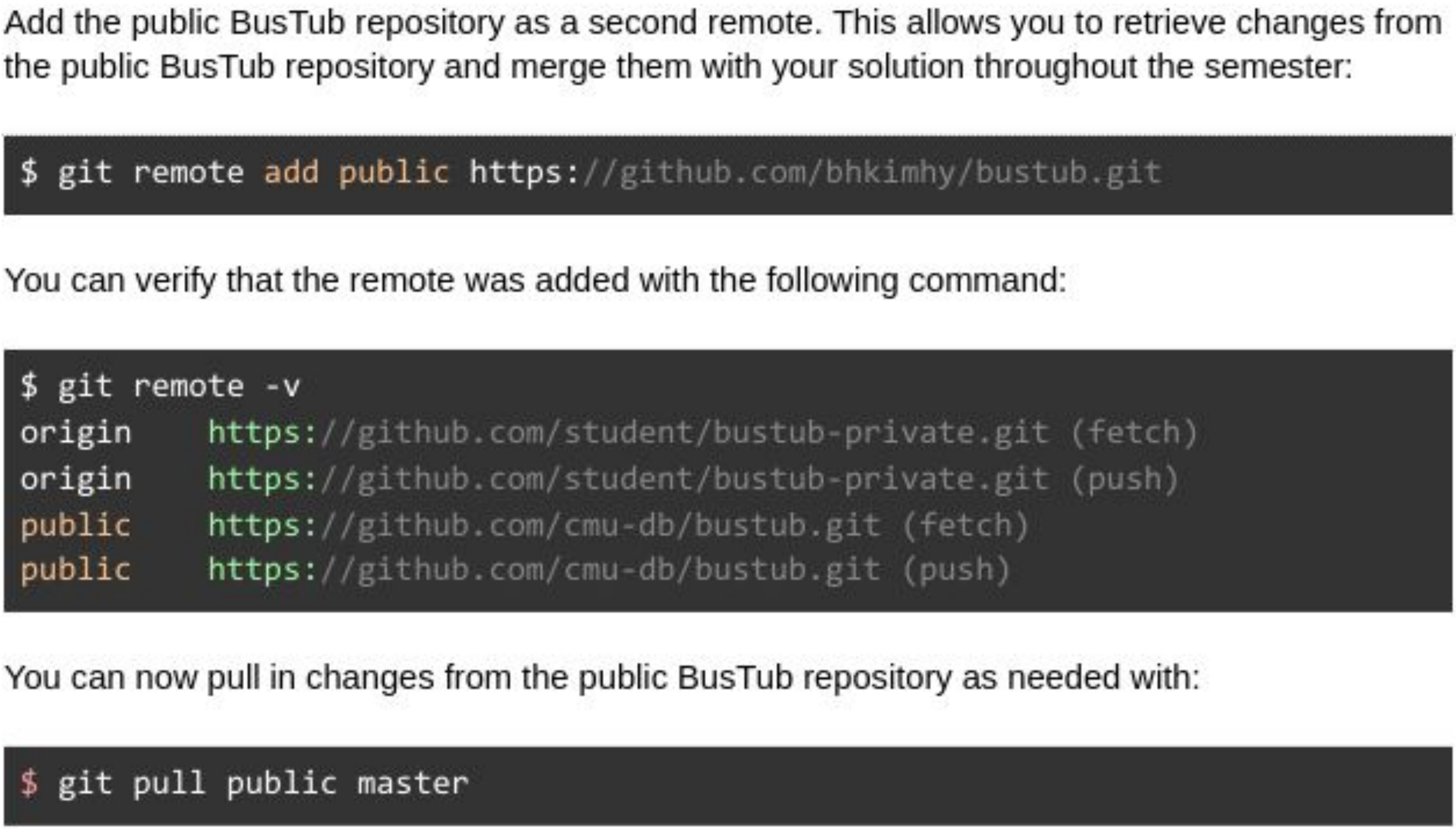
This copies everything in the public BusTub repository to your own private repository. You can now delete your local clone of the public repository:

```
$ cd ..
$ rm -rf bustub-public
```

Clone your private repository to your development machine:

```
# If you pull / push over HTTPS
$ git clone https://github.com/student/bustub-private.git

# If you pull / push over SSH
$ git clone git@github.com:student/bustub-private.git
```

Git Repository Setup



Add the public BusTub repository as a second remote. This allows you to retrieve changes from the public BusTub repository and merge them with your solution throughout the semester:

```
$ git remote add public https://github.com/bhkimhy/bustub.git
```

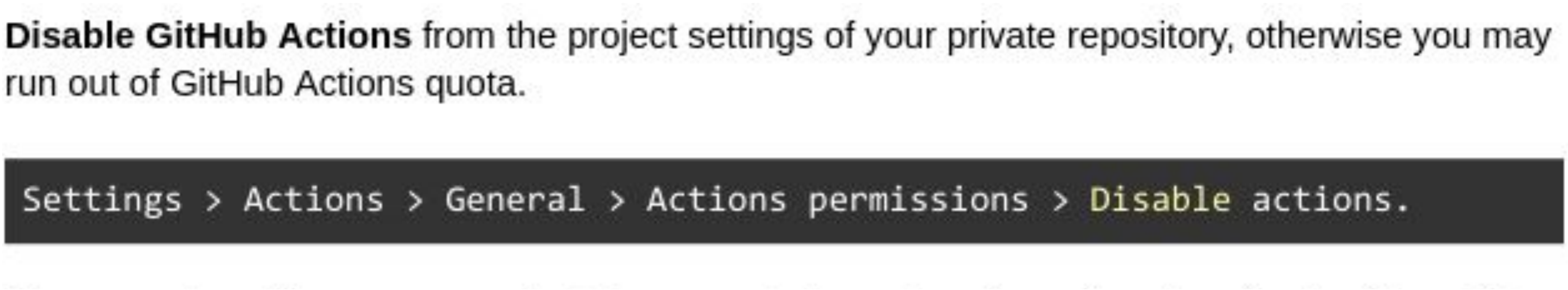
You can verify that the remote was added with the following command:

```
$ git remote -v
origin    https://github.com/student/bustub-private.git (fetch)
origin    https://github.com/student/bustub-private.git (push)
public    https://github.com/cmu-db/bustub.git (fetch)
public    https://github.com/cmu-db/bustub.git (push)
```

You can now pull in changes from the public BusTub repository as needed with:

```
$ git pull public master
```

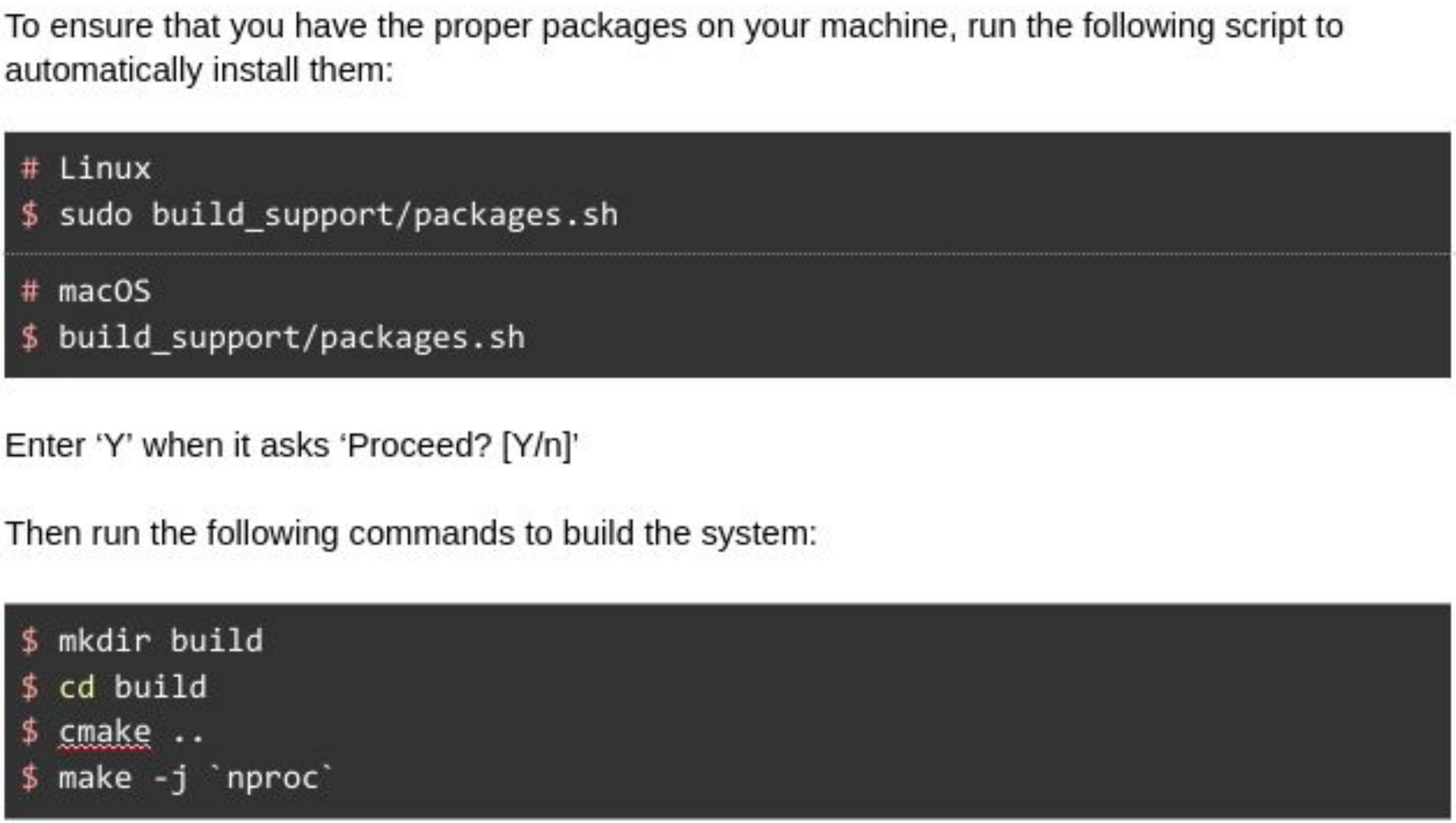
Git Repository Setup



Disable GitHub Actions from the project settings of your private repository, otherwise you may run out of GitHub Actions quota.

```
Settings > Actions > General > Actions permissions > Disable actions.
```

Build from the Source Code



To ensure that you have the proper packages on your machine, run the following script to automatically install them:

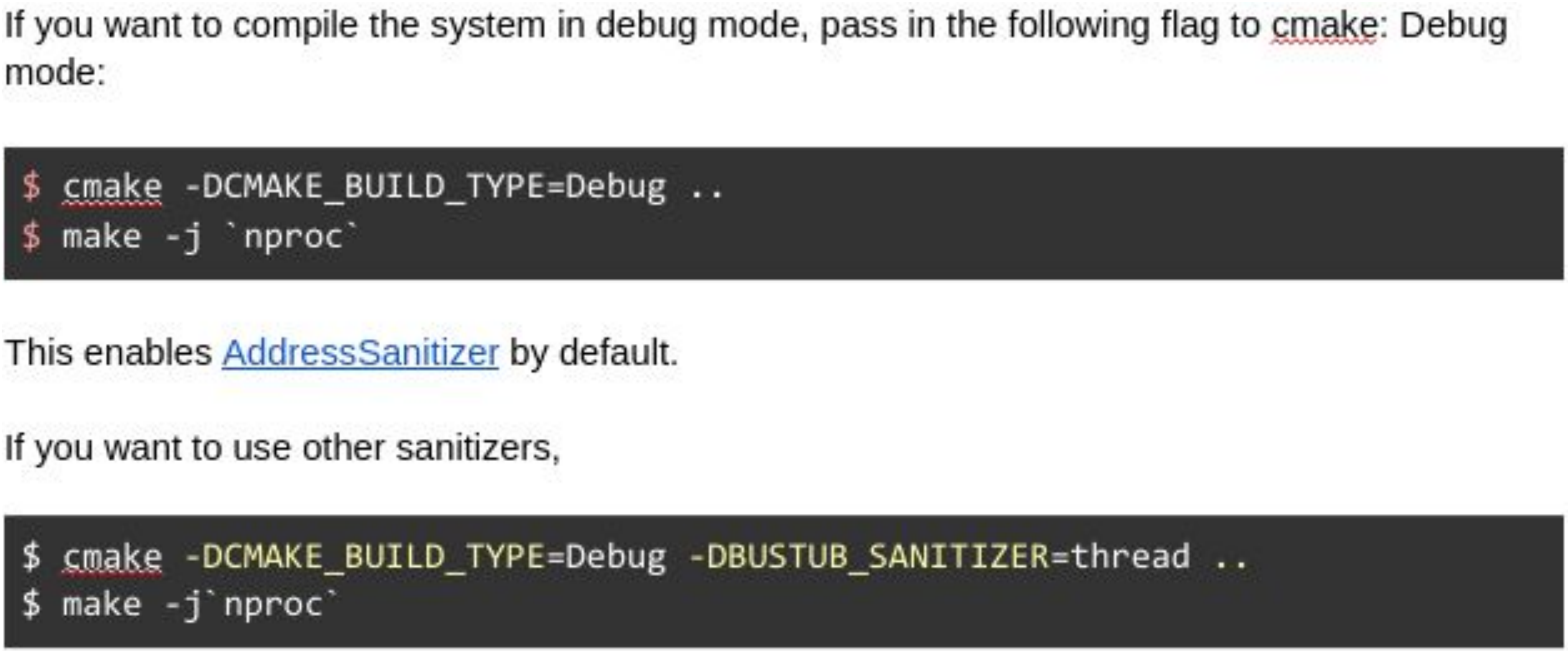
```
# Linux
$ sudo build_support/packages.sh

# macOS
$ build_support/packages.sh
```

Enter ‘Y’ when it asks ‘Proceed? [Y/n]’

Then run the following commands to build the system:

```
$ mkdir build
$ cd build
$ cmake ..
$ make -j `nproc`
```

Build from the Source Code



If you want to compile the system in debug mode, pass in the following flag to cmake: Debug mode:

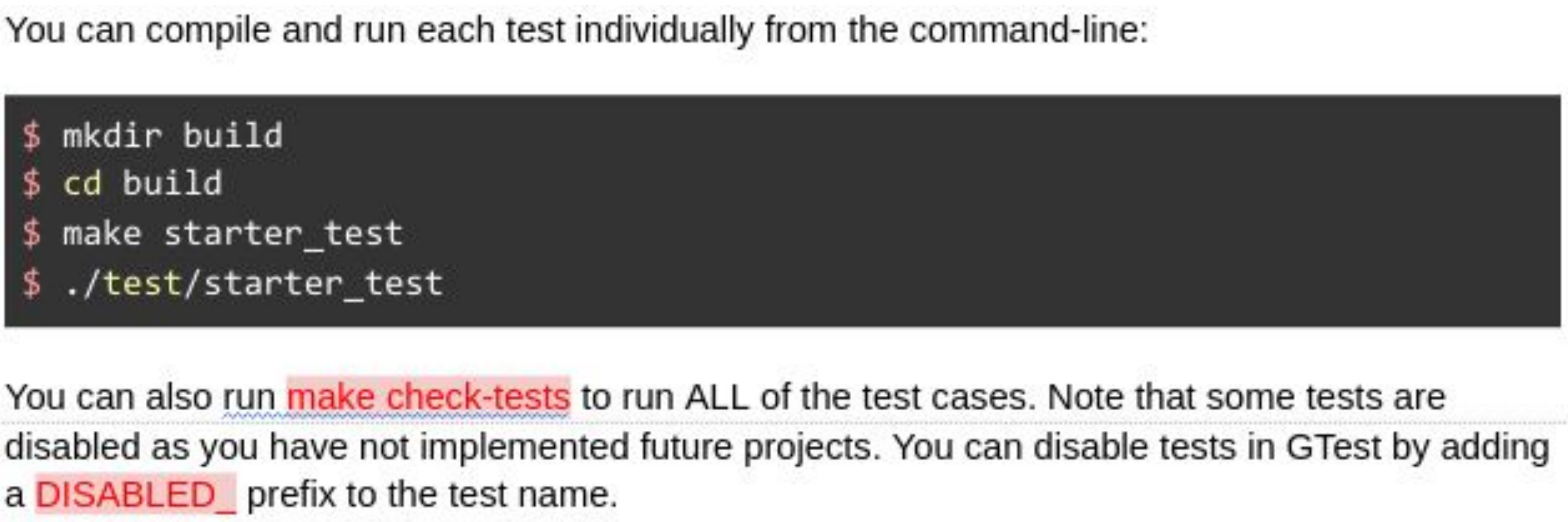
```
$ cmake -DCMAKE_BUILD_TYPE=Debug ..
$ make -j `nproc`
```

This enables AddressSanitizer by default.

If you want to use other sanitizers,

```
$ cmake -DCMAKE_BUILD_TYPE=Debug -DBUSTUB_SANITIZER=thread ..
$ make -j`nproc`
```

Run Tests



You can compile and run each test individually from the command-line:

```
$ mkdir build
$ cd build
$ make starter_test
$ ./test/starter_test
```

You can also run make check-tests to run ALL of the test cases. Note that some tests are disabled as you have not implemented future projects. You can disable tests in GTest by adding a DISABLED_ prefix to the test name.

Outline

- About Labs
- Set up the Project Environment
- Project Overview
- Assignment

Project Overview

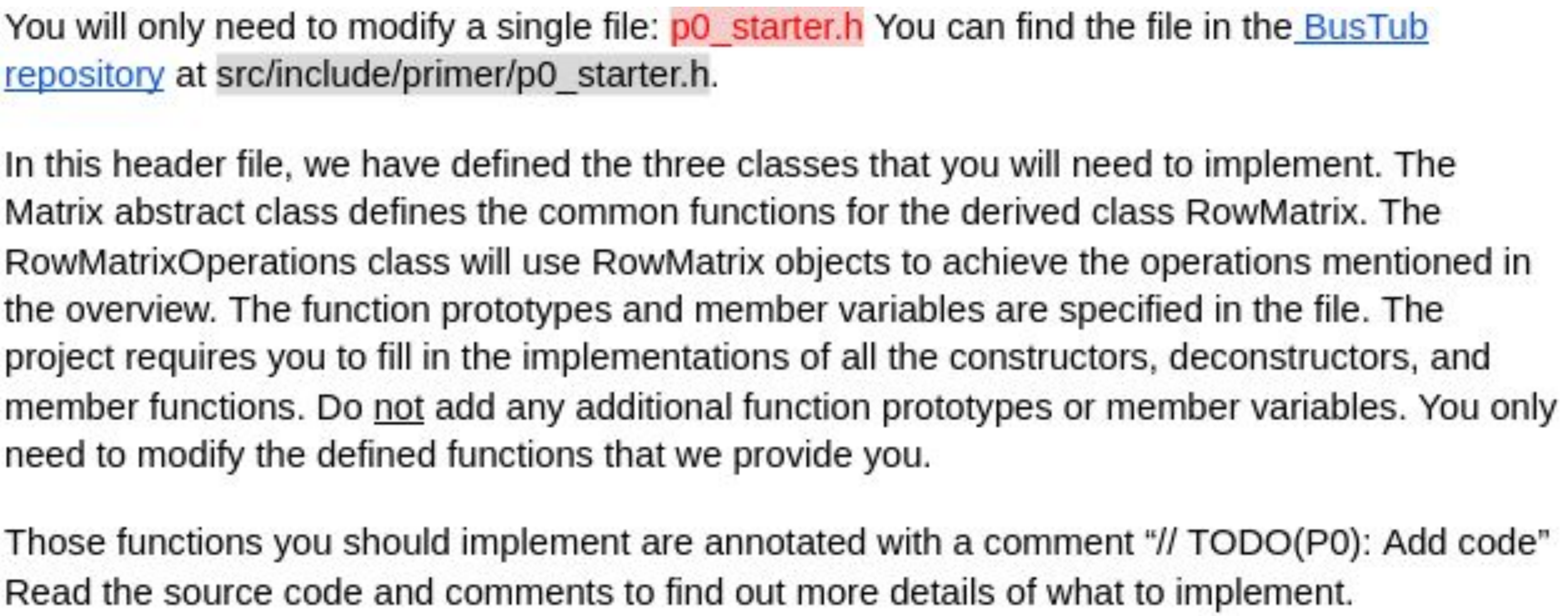
- Implement three classes:
    - Matrix
    - RowMatrix
    - RowMatrixOperations
- These matrices are simple two-dimensional matrices that must support addition, matrix multiplication, and a simplified General Matrix Multiply (GEMM) operations.
    - The description on GEMM operations is here: https://en.wikipedia.org/wiki/Basic_Linear_Algebra_Subprograms#Level_3

Outline

- About Labs
- Set up the Project Environment
- Project Overview
- Assignment

Assignment

- As described below, read the source code and comments to understand what it is that you need to implement.
  - We will start programming in the next lab!



You will only need to modify a single file: p0_starter.h You can find the file in the BusTub repository at src/include/primer/p0_starter.h.

In this header file, we have defined the three classes that you will need to implement. The Matrix abstract class defines the common functions for the derived class RowMatrix. The RowMatrixOperations class will use RowMatrix objects to achieve the operations mentioned in the overview. The function prototypes and member variables are specified in the file. The project requires you to fill in the implementations of all the constructors, deconstructors, and member functions. Do not add any additional function prototypes or member variables. You only need to modify the defined functions that we provide you.

Those functions you should implement are annotated with a comment "// TODO(P0): Add code" Read the source code and comments to find out more details of what to implement.

The End