Week 02 - Getting started with GitHub

Louis Botha louis.botha@tuni.fi

[MISSING
IMAGE: ,]

GitHub classroom

For assignments/exercises we are going to use GitHub classroom and GitHub repositories.

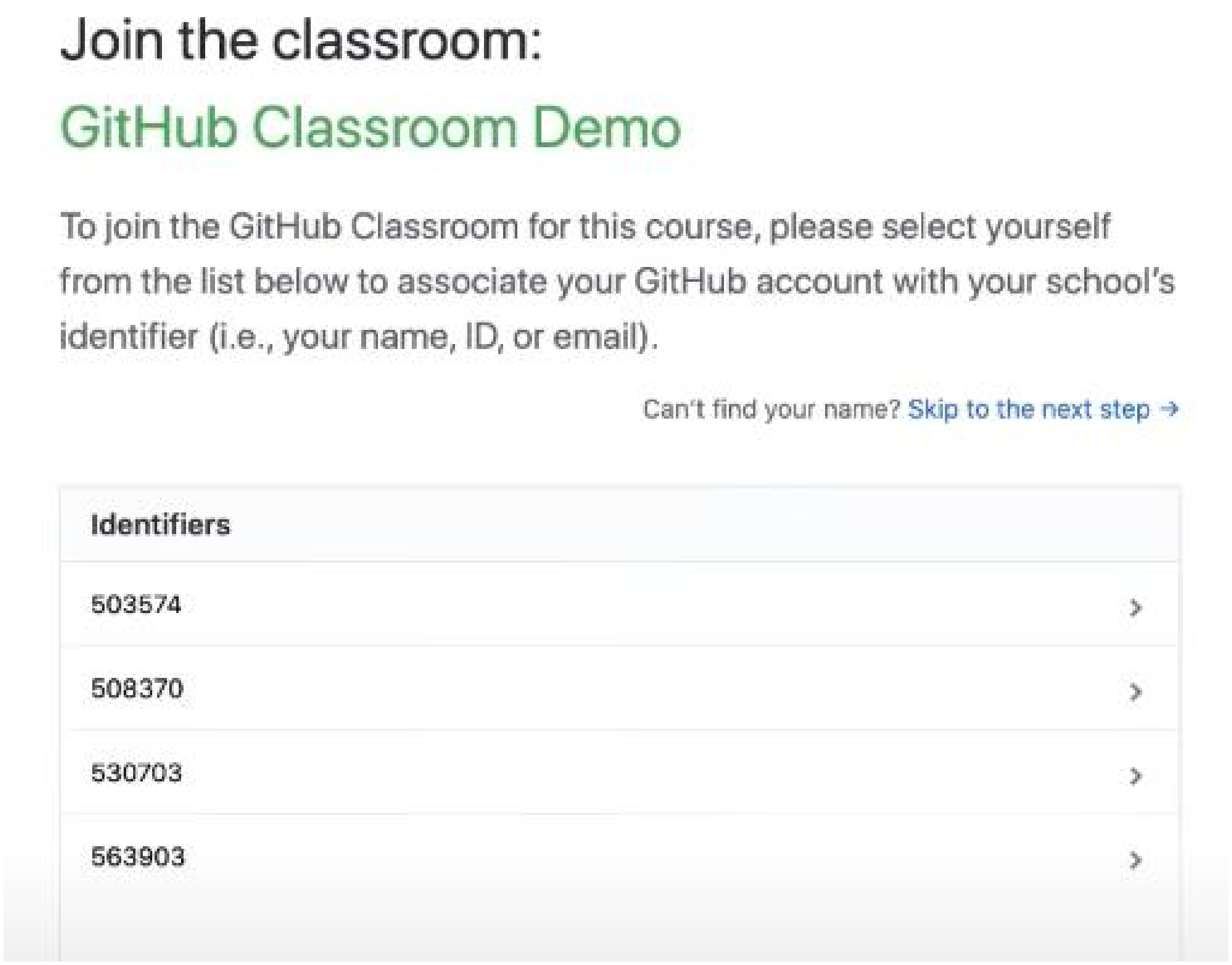
Each weeks assignment will be in its own git repository.

Each GitHub Classroom assignment will start with an invitation that you need to open and accept.

Here is the **2023-wk02 Getting started with GitHub** assignment invitation: https://classroom.github.com/a/7FwdgSmG

Join the Classroom (Once)

Opening an assignment invitation for the first time, you will be requested to join the classroom by linking your Account to a TAMK identifier (most likely your student number or tuni email). You only need to join the classroom once. Find your identifier and click on it to proceed.


Join the classroom:

GitHub Classroom Demo

To join the GitHub Classroom for this course, please select yourself from the list below to associate your GitHub account with your school's identifier (i.e., your name, ID, or email).

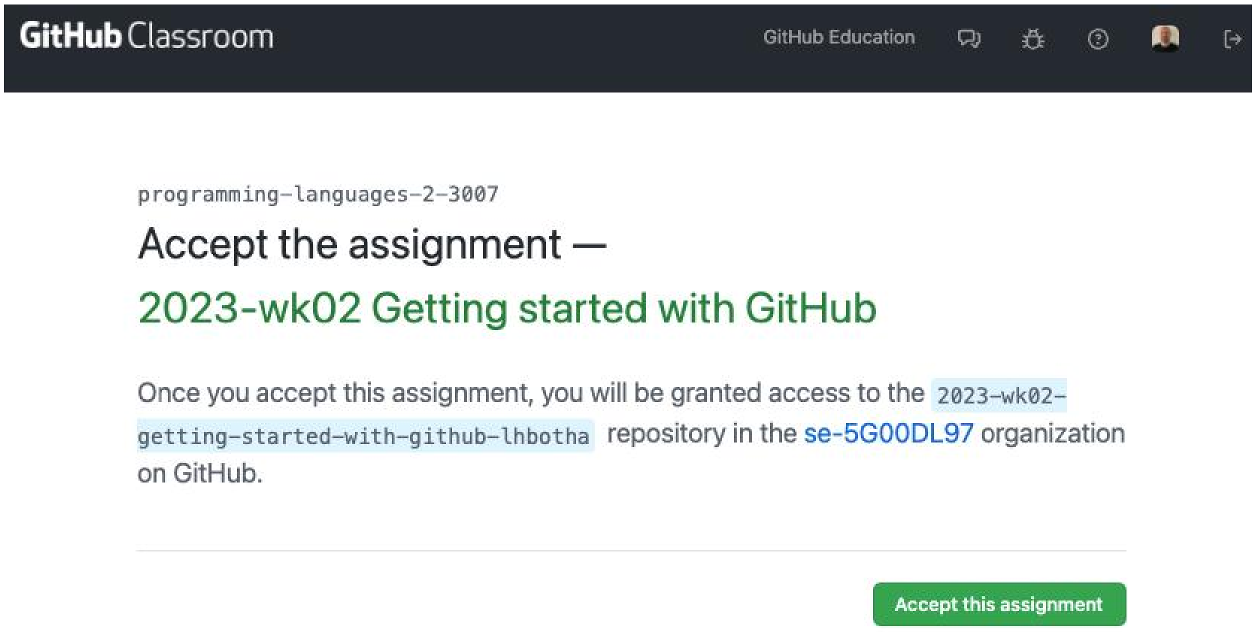
Can't find your name? Skip to the next step →

Identifiers
503574
508370
530703
563903

- ⚠ *Note*
- *The classroom is* ***programming-languages2***
- *Identifier is your* ***TUNI email***
-
- *If your identifier is not there, let me know!*
- *If you linked by accident to the wrong identifier, let me know!*

Accept the Assignment

Accept the assignment by clicking on the "Accept this assignment" button.

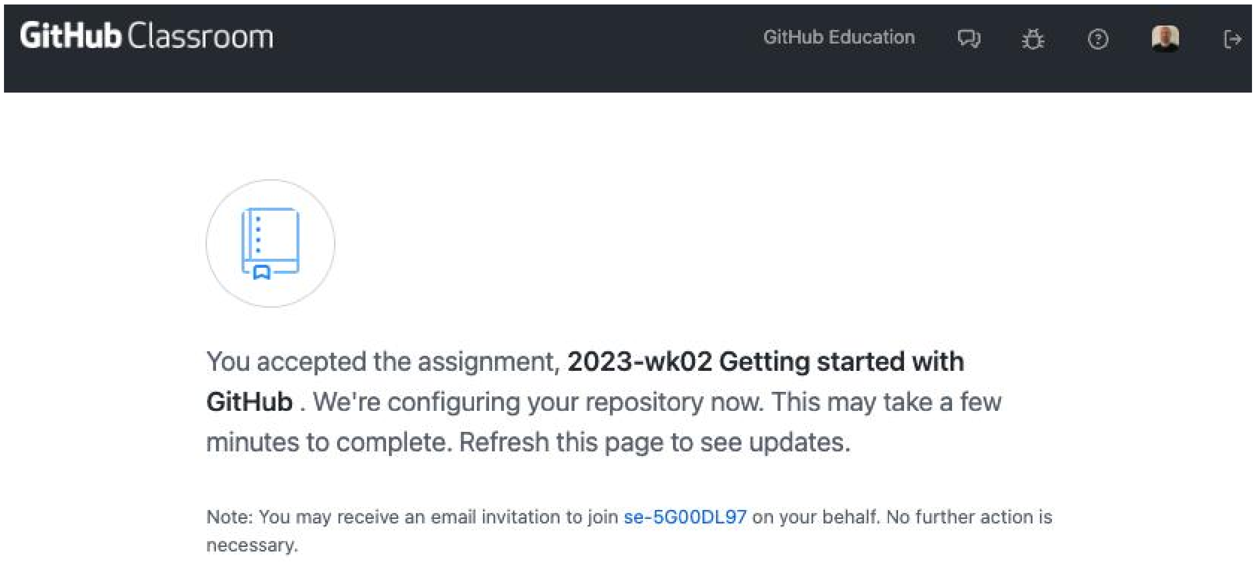


programming-languages-2-3007

Accept the assignment —
2023-wk02 Getting started with GitHub

Once you accept this assignment, you will be granted access to the `2023-wk02-getting-started-with-github-lhbotha` repository in the se-5G00DL97 organization on GitHub.

After accepting the assignment, an assignment repository will be automatically created for you. It will also import all the skeleton code.

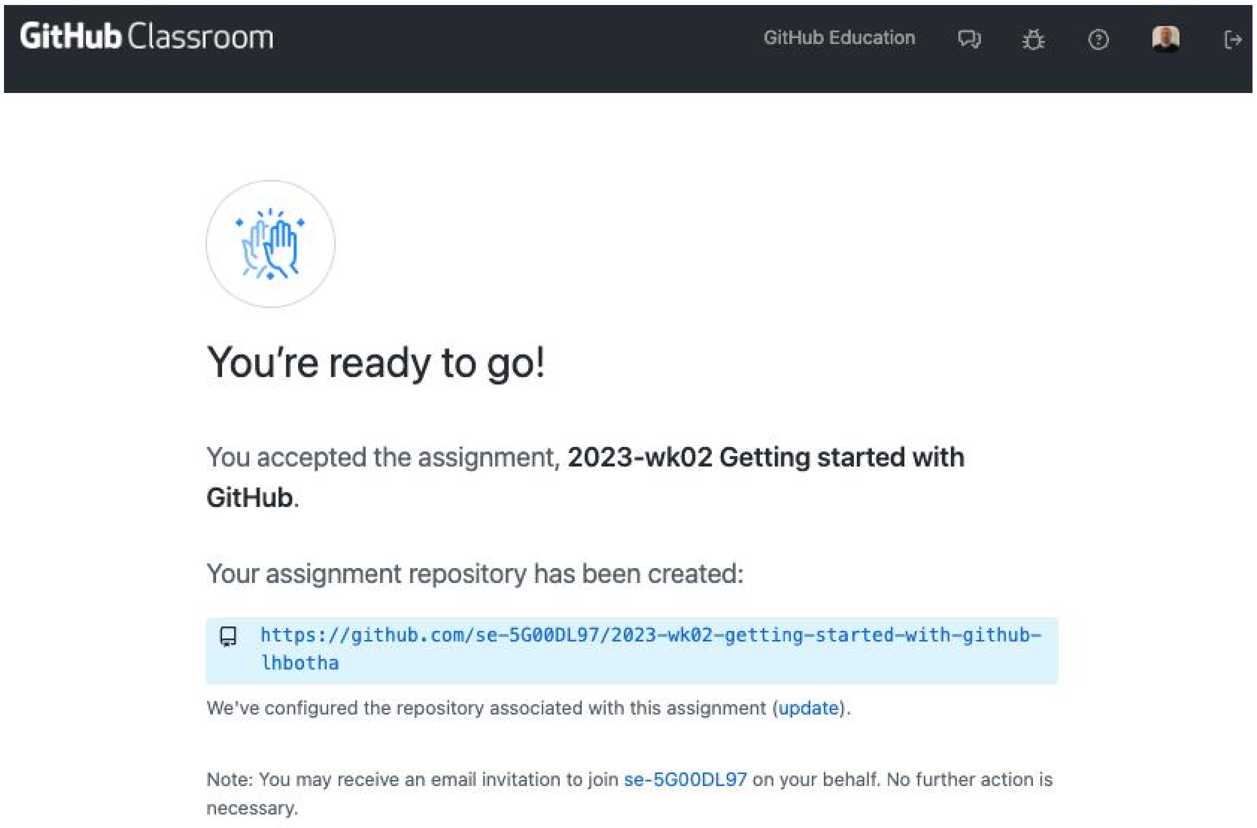


You accepted the assignment, **2023-wk02 Getting started with GitHub** . We're configuring your repository now. This may take a few minutes to complete. Refresh this page to see updates.

Note: You may receive an email invitation to join se-5G00DL97 on your behalf. No further action is necessary.

The page doesn't always automatically refresh so you might need to reload the page.

When the assignment repository is created, you will be provided with a link to the repository.



You're ready to go!

You accepted the assignment, **2023-wk02 Getting started with GitHub**.

Your assignment repository has been created:

`https://github.com/se-5G00DL97/2023-wk02-getting-started-with-github-lhbotha`

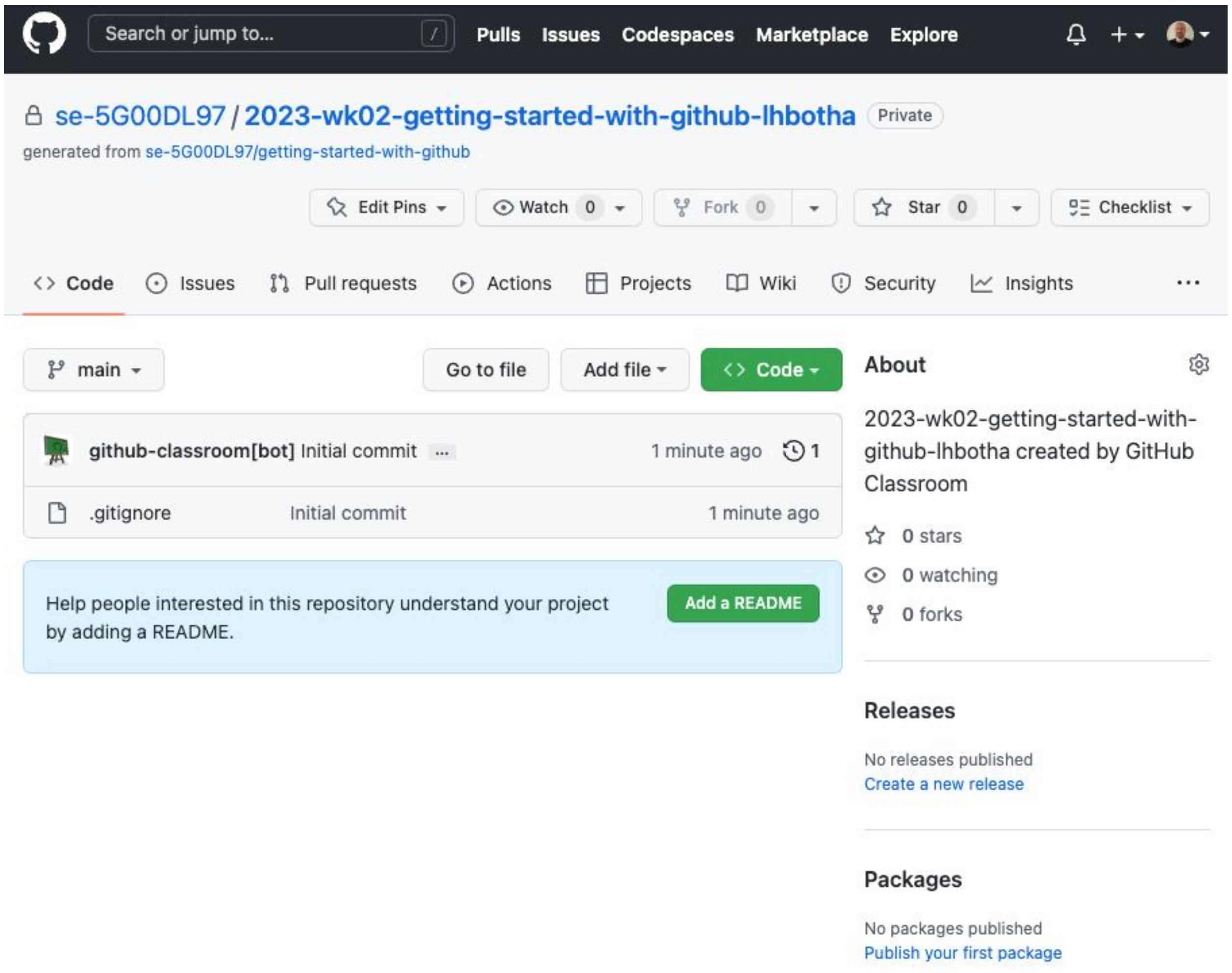
We've configured the repository associated with this assignment (update).

Note: You may receive an email invitation to join se-5G00DL97 on your behalf. No further action is necessary.

Click on the link to open the assignment repository. You should now see the normal repository dashboard.

. ⚠ ***Always start by reading the README.md carefully*** ⚠

The README.md will always contain the instructions for the assignment and exercises. This assignment, probably most, also contains skeleton code that you need to complete.


se-5G00DL97 / 2023-wk02-getting-started-with-github-lhbotha Private

generated from se-5G00DL97/getting-started-with-github

github-classroom[bot] Initial commit ... 1 minute ago 1

.gitignore Initial commit 1 minute ago

Help people interested in this repository understand your project by adding a README.

About

2023-wk02-getting-started-with-github-lhbotha created by GitHub Classroom

0 stars

0 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

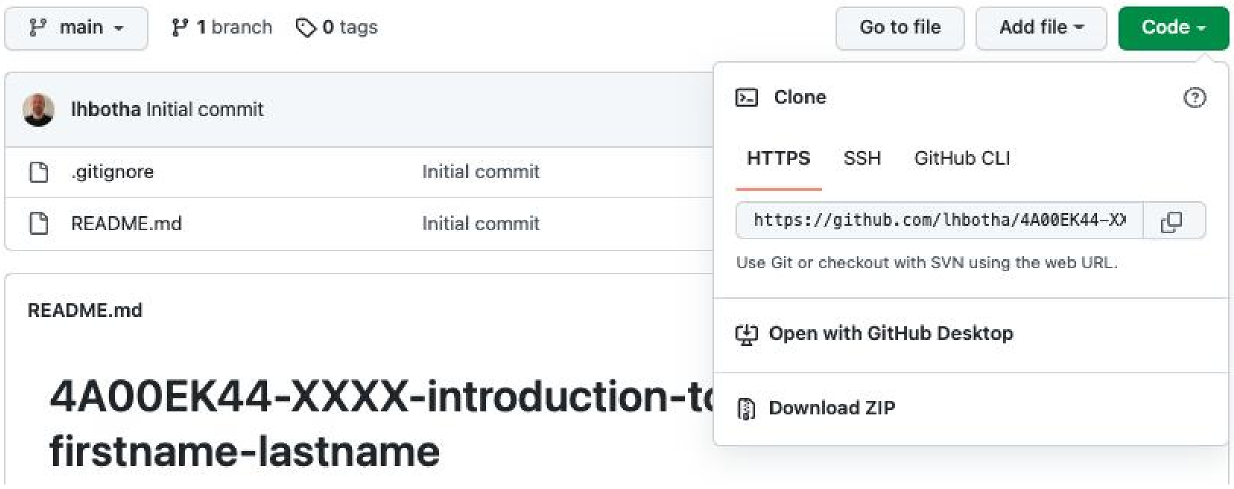
Publish your first package

GitHub and git clone

Git is a **distributed version control system**, which is freely made available as open source. Several well-known names, like the Linux kernel, Eclipse and Android use Git for managing their projects.

There is GUI tools for using git but using the command line will give you much more street cred.

. ⚠ *Repository name is different then in screenshots*

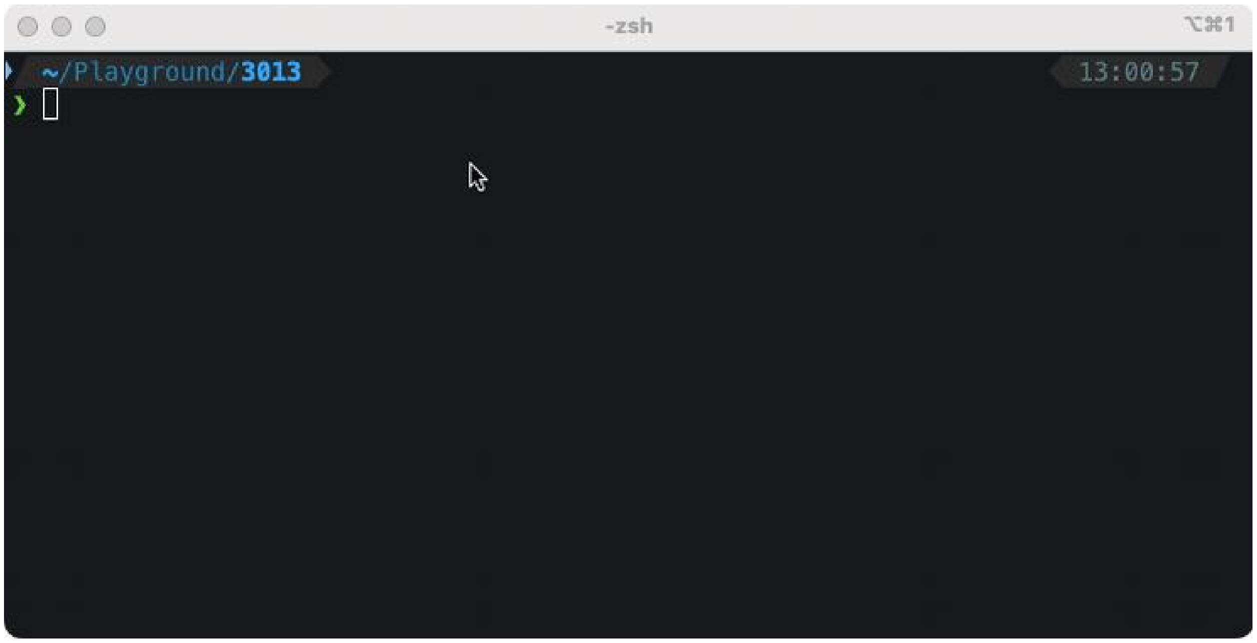


```
https://github.com/lhbotha/4A00EK44-XX
```

Use Git or checkout with SVN using the web URL.

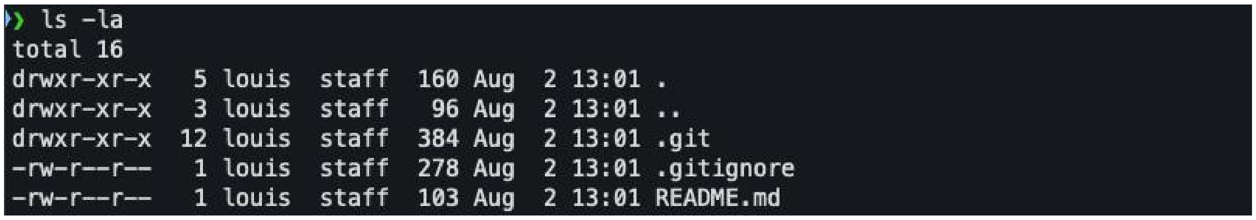
4A00EK44-XXXX-introduction-t
firstname-lastname

Open a terminal and change to the directory to where you want to keep the repository on your local machine. Now use the git clone command to clone the repository to your local machine.



```
~/Playground/3013                                      13:00:57
❯
```

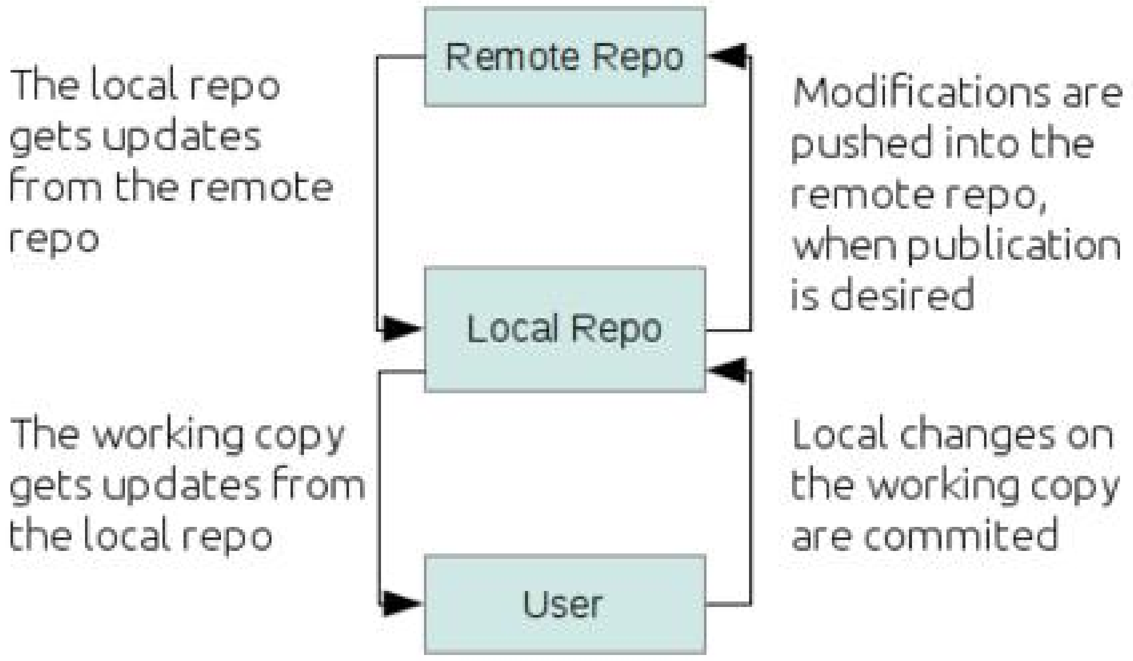
In the directory where the repository is cloned we can see the README.md file. There is also hidden files and directories.



```
❯ ls -la
total 16
drwxr-xr-x   5 louis  staff  160 Aug  2 13:01 .
drwxr-xr-x   3 louis  staff   96 Aug  2 13:01 ..
drwxr-xr-x  12 louis  staff  384 Aug  2 13:01 .git
-rw-r--r--   1 louis  staff  278 Aug  2 13:01 .gitignore
-rw-r--r--   1 louis  staff  103 Aug  2 13:01 README.md
```

There is basically 3 copies of the README.md file now in existence:

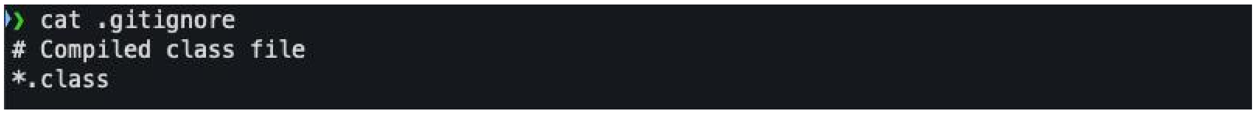
- Remote repository on GitHub
- The cloned local repository
- The working copy (the visible files in the directory)


Remote Repo
The local repo gets updates from the remote repo
Modifications are pushed into the remote repo, when publication is desired
Local Repo
The working copy gets updates from the local repo
Local changes on the working copy are commited
User

The simple workflow is that you make changes to the working copy, that you **commit** to the local repository which is then returned (pushed) to the remote repository.

The local version of the repository is in the .git directory, you should never need to change things there.

In the .gitignore file you place each filename or file type that you want git to ignore when checking for changes. For example, we used the Java template when creating the repository so there is a ignore added already for Java .class files.

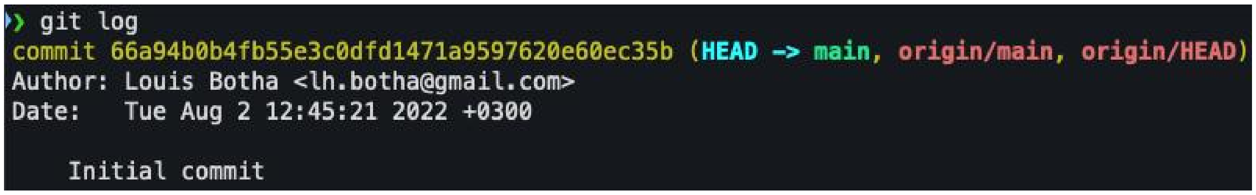


```
❯ cat .gitignore
# Compiled class file
*.class
```

When using git for the first time you will be prompted at some point that you need to set a user name and user email. Setting this configuration values are compulsory as it is needed in the git logs. You set it on the command line with the following commands.

```
git config --global user.name "Your Name"
git config --global user.email "youremail@yourdomain.com"
```

To visualize the point, here is the log of the newly created repository:



```
❯ git log
commit 66a94b0b4fb55e3c0dfd1471a9597620e60ec35b (HEAD -> main, origin/main, origin/HEAD)
Author: Louis Botha <lh.botha@gmail.com>
Date:   Tue Aug 2 12:45:21 2022 +0300

    Initial commit
```

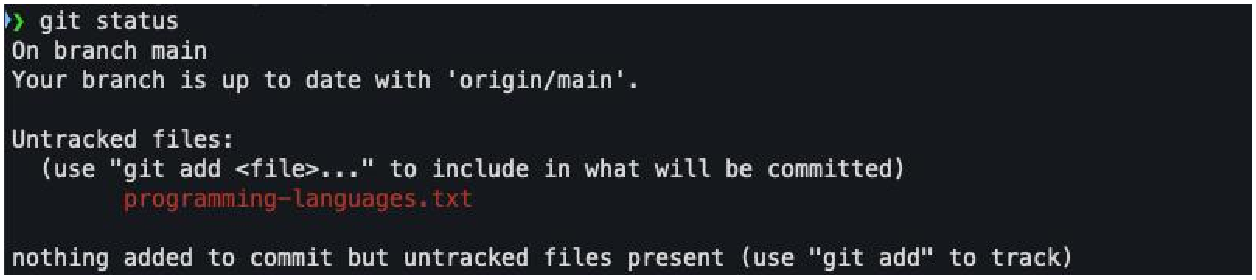
New files

Create a new file called programming-languages.txt in your working directory.

Add some content to the file:

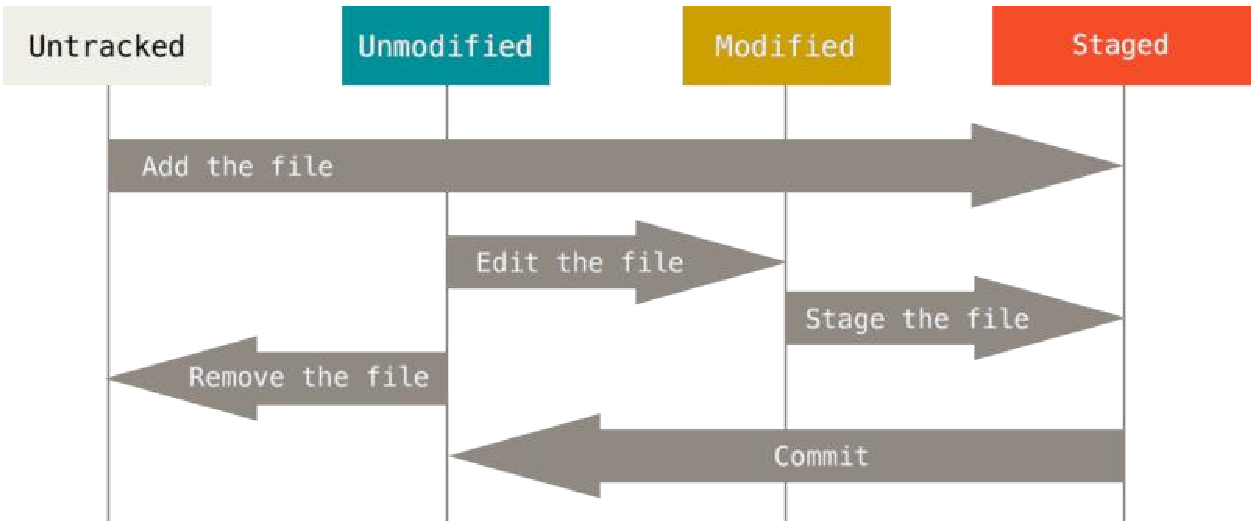
```
- JavaScript
- Java
- C++
```

Now we have a new file in working directory. We can see the state of the working directory with the command git status.

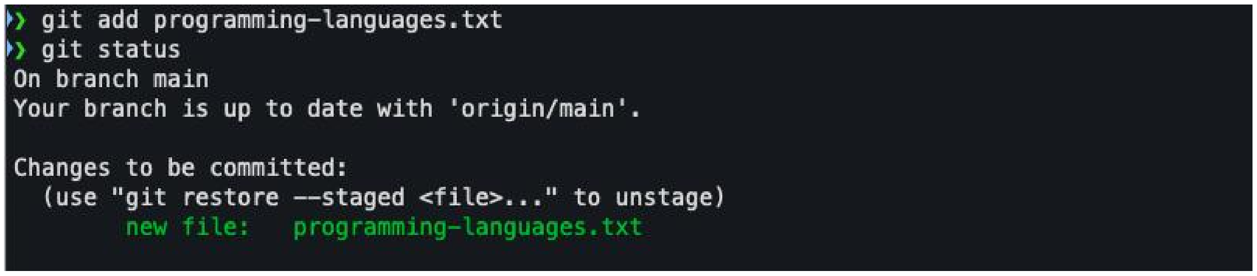


```
❯ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        programming-languages.txt

nothing added to commit but untracked files present (use "git add" to track)
```

We can see the new file reported as an untracked file by git. This means git has noticed or detected the file but the file is not part of the repository yet and it isn't checking the file for modifications.

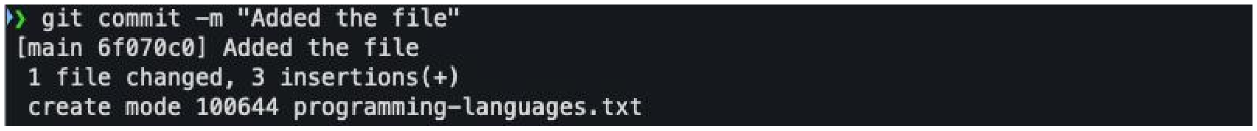


For git to start tracking the programming-languages.txt file for changes we need to **add** it with the git add command. Based on the above diagram, it means that the file went from being *untracked* to being *tracked* state.



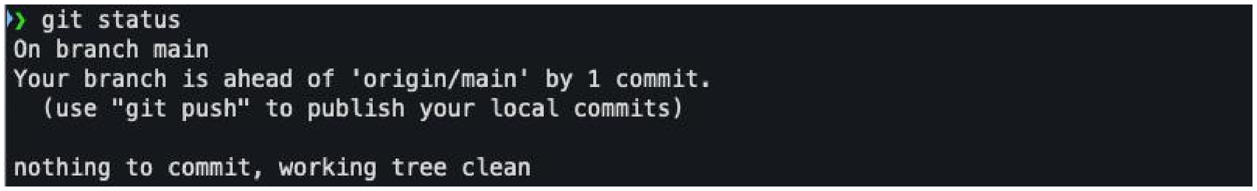
```
❯ git add programming-languages.txt
❯ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   programming-languages.txt
```

Now the programming-languages.txt is staged and you can **commit** it to the local repository with the git commit -m command. The -m is short for message. It is good practice to add a describing message to the commit.



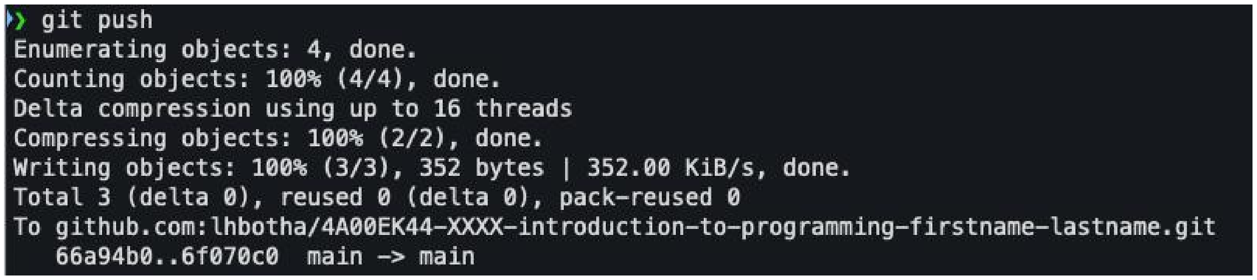
```
❯ git commit -m "Added the file"
[main 6f070c0] Added the file
 1 file changed, 3 insertions(+)
 create mode 100644 programming-languages.txt
```

Now that the new file is in the local repository and the working copy matches the local repository we can see with git status that we have no changes.



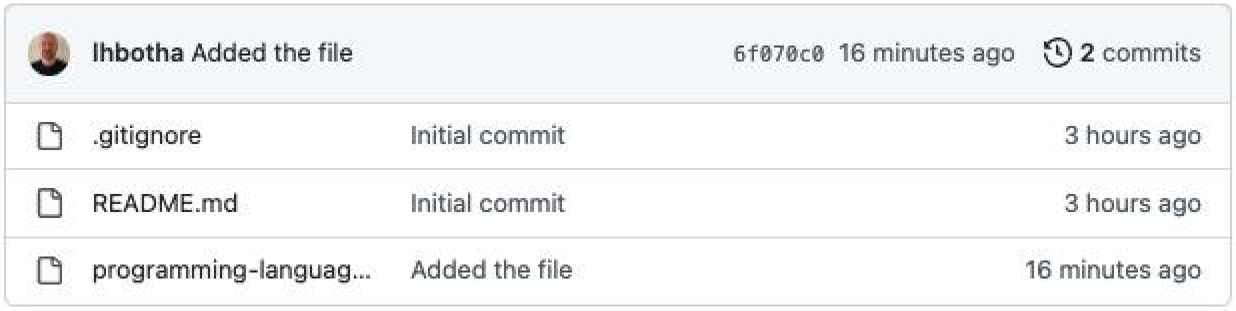
```
❯ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

The status message gives us another hint though. It tells us that our local repository has changes that are not in the remote repository. Next we want to get the changes in local repository to the GitHub repository. We do this by **pushing** our changes to the remote repository with the git push command.



```
❯ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 352 bytes | 352.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:lhbotha/4A00EK44-XXXX-introduction-to-programming-firstname-lastname.git
   66a94b0..6f070c0  main -> main
```

Now we should also see the file in GitHub with our commit message.



lhbotha Added the file · 6f070c0 16 minutes ago · 2 commits

.gitignore	Initial commit	3 hours ago
README.md	Initial commit	3 hours ago
programming-languag...	Added the file	16 minutes ago

Modified files

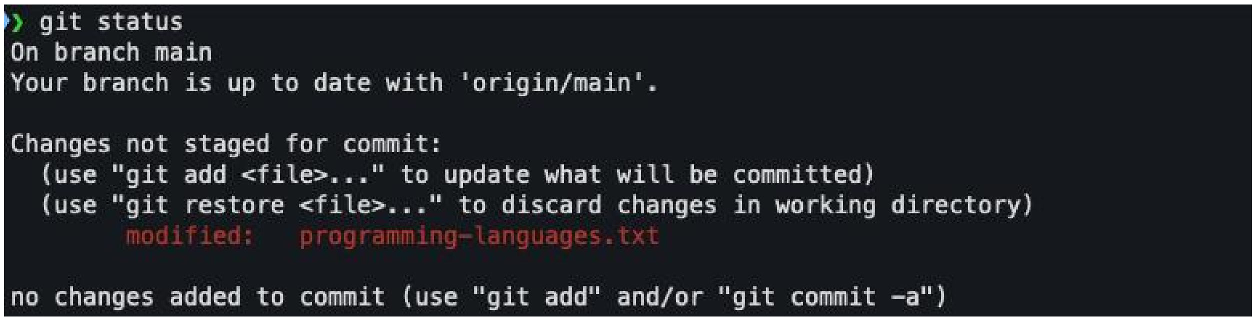
Let’s make a change to the programming-languages.txt file by adding another language. Add the language Go to the end of the file.

Add some content to the file:

```
- JavaScript
- Java
- C++
- Go
```

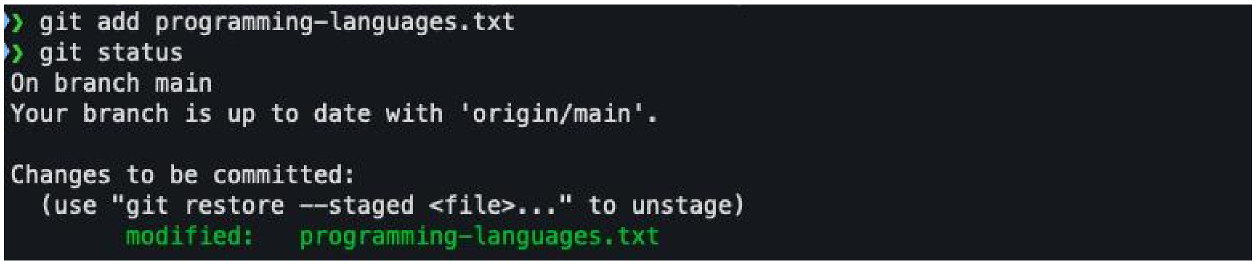
Check the diagram above again. Once a file is tracked, it can have a *unmodified*, *modified* and *staged* state.

Let's check the state of the working directory with git status again now that we made changes to the file.



```
❯ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   programming-languages.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Now git detected that the programming-languages.txt was modified, that there was changes to the file but the changes is not staged to be committed to the local repository. Again we use git add to stage the file.



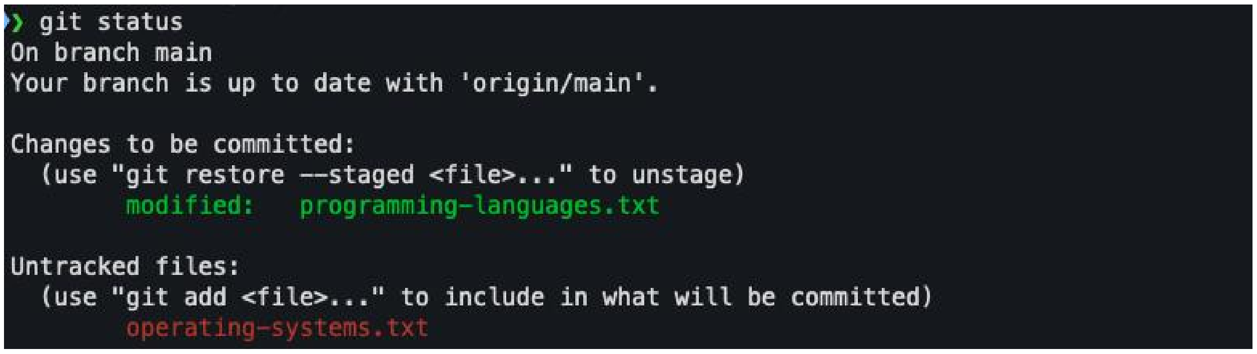
```
❯ git add programming-languages.txt
❯ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   programming-languages.txt
```

Now the changes are staged and we can commit it with the git commit -m command. Let's add another file first and see how that works.

Create a new file called operating-systems.txt and add some content to it:

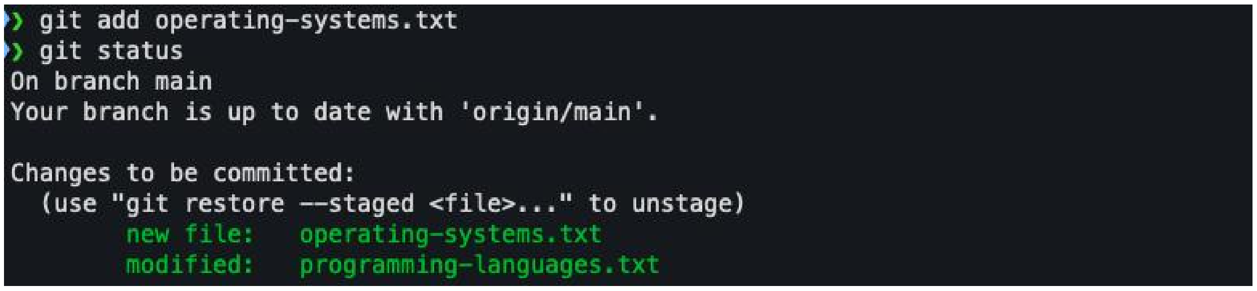
```
- Windows 11
- Ubuntu Linux
- MacOS Monterey
```

We can use git status to check the state of the working directory



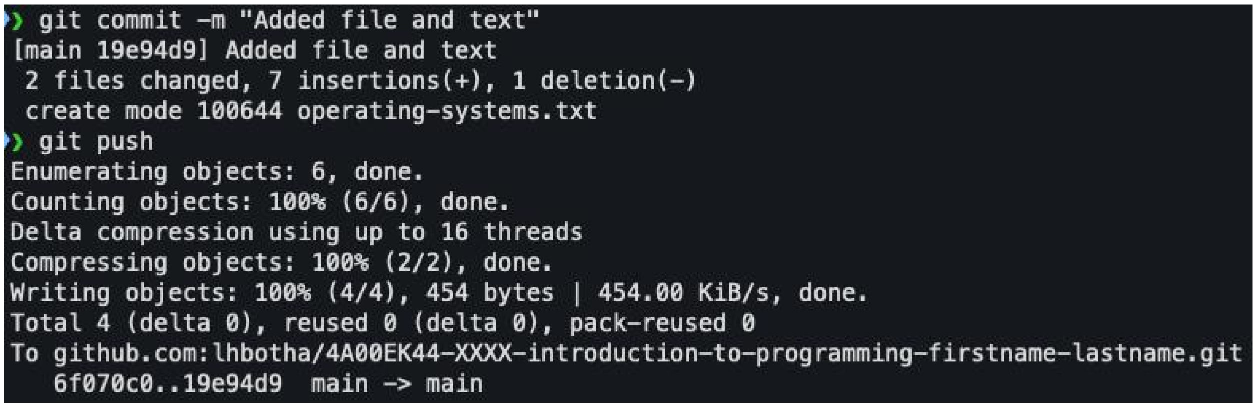
```
❯ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   programming-languages.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        operating-systems.txt
```

Like expected we have the modified programming-languages.txt file and now the untracked operating-systems.txt file. We can still add the operating-systems.txt to the commit with the git add operating-systems.txt command.



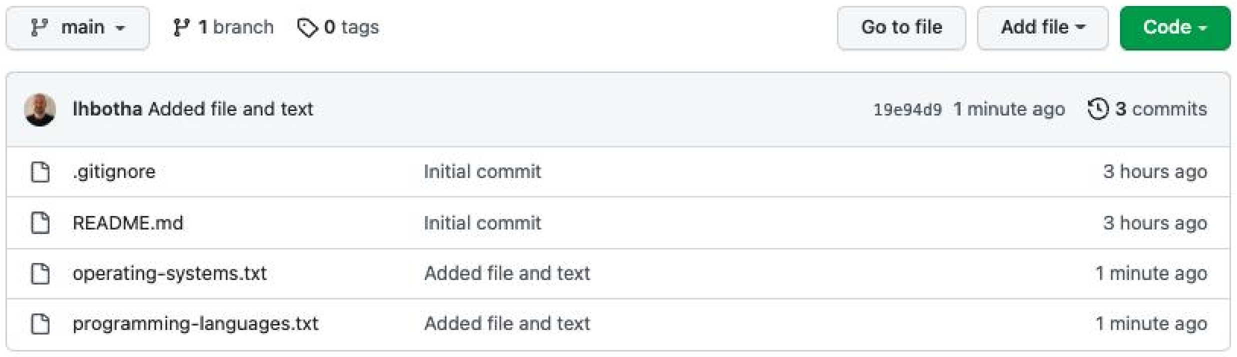
```
❯ git add operating-systems.txt
❯ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   operating-systems.txt
        modified:   programming-languages.txt
```

Let's commit and push these changes to GitHub.



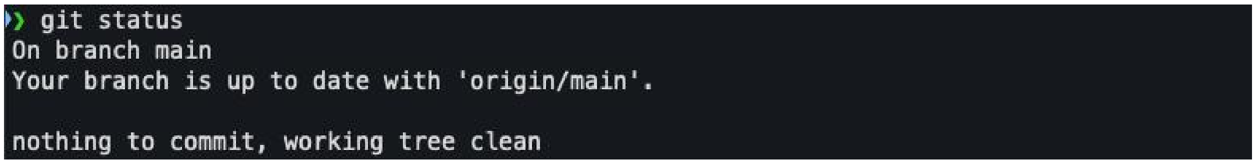
```
❯ git commit -m "Added file and text"
[main 19e94d9] Added file and text
 2 files changed, 7 insertions(+), 1 deletion(-)
 create mode 100644 operating-systems.txt
❯ git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 454 bytes | 454.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:lhbotha/4A00EK44-XXXX-introduction-to-programming-firstname-lastname.git
   6f070c0..19e94d9  main -> main
```

We should see change now also in the GitHub repository with our last commit message.


main · 1 branch · 0 tags · Go to file · Add file · Code

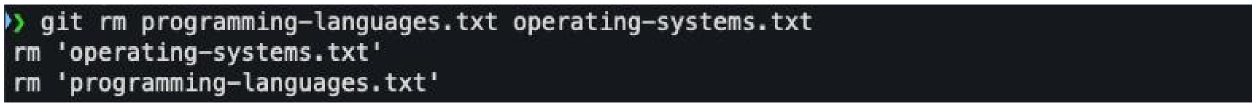
lhbotha Added file and text	19e94d9 1 minute ago	3 commits
.gitignore	Initial commit	3 hours ago
README.md	Initial commit	3 hours ago
operating-systems.txt	Added file and text	1 minute ago
programming-languages.txt	Added file and text	1 minute ago

We should see that the working directory is clean and that the local and remote repositories are in sync with git status.

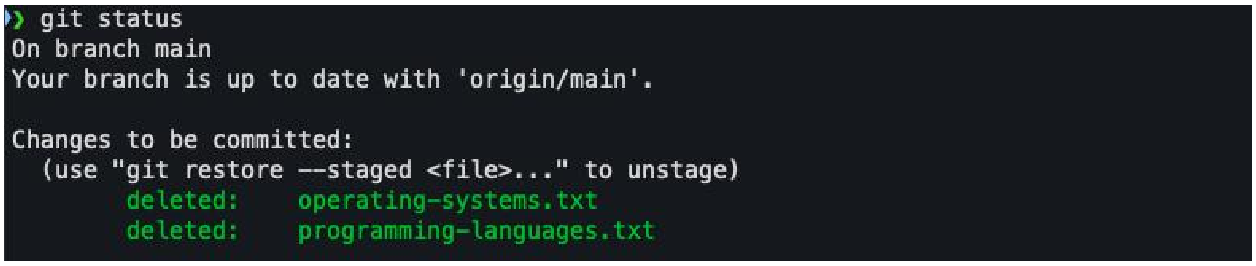

```
❯ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Removing files

When you want to delete files from the working directory, the safest is to use the git rm command. Let’s remove the 2 files we created.

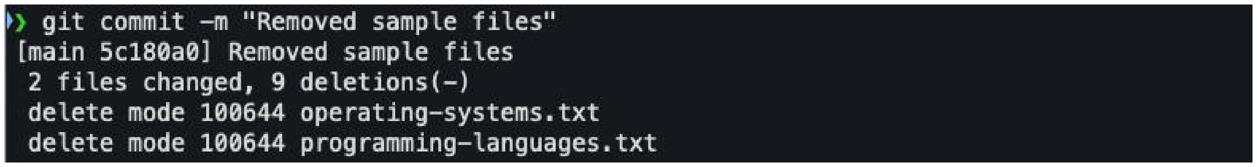

```
❯ git rm programming-languages.txt operating-systems.txt
rm 'operating-systems.txt'
rm 'programming-languages.txt'
```

Let’s check the change with git status.


```
❯ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    operating-systems.txt
        deleted:    programming-languages.txt
```

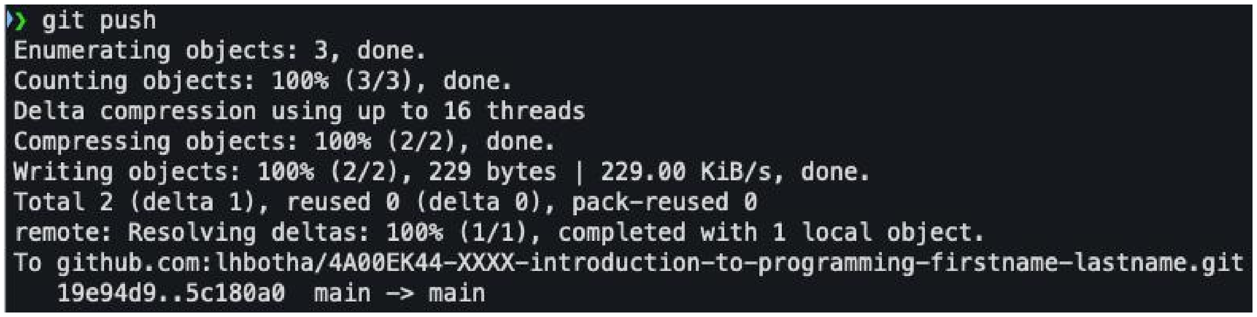
Like expected we see that the files are deleted and once we commit the changes the files will be removed.

When we make the commit the files are deleted locally.



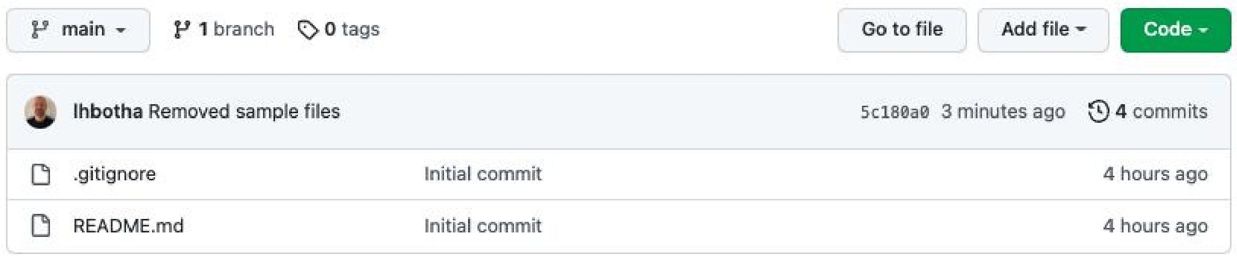
```
❯ git commit -m "Removed sample files"
[main 5c180a0] Removed sample files
 2 files changed, 9 deletions(-)
 delete mode 100644 operating-systems.txt
 delete mode 100644 programming-languages.txt
```

The remote repository still have the files, we can now push the change to the remote repository.



```
❯ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 229 bytes | 229.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:lhbotha/4A00EK44-XXXX-introduction-to-programming-firstname-lastname.git
   19e94d9..5c180a0  main -> main
```

We should now see that the files are removed from the GitHub repository.

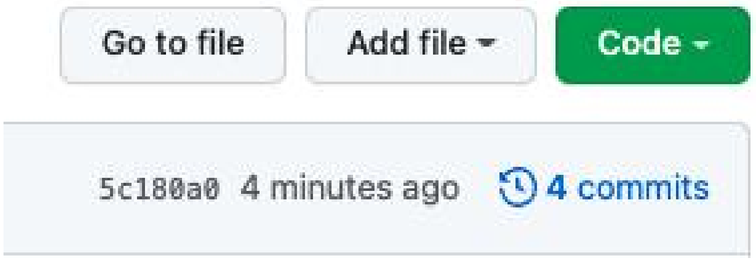


main · 1 branch · 0 tags · Go to file · Add file · Code

lhbotha Removed sample files — 5c180a0 3 minutes ago — 4 commits

.gitignore	Initial commit	4 hours ago
README.md	Initial commit	4 hours ago

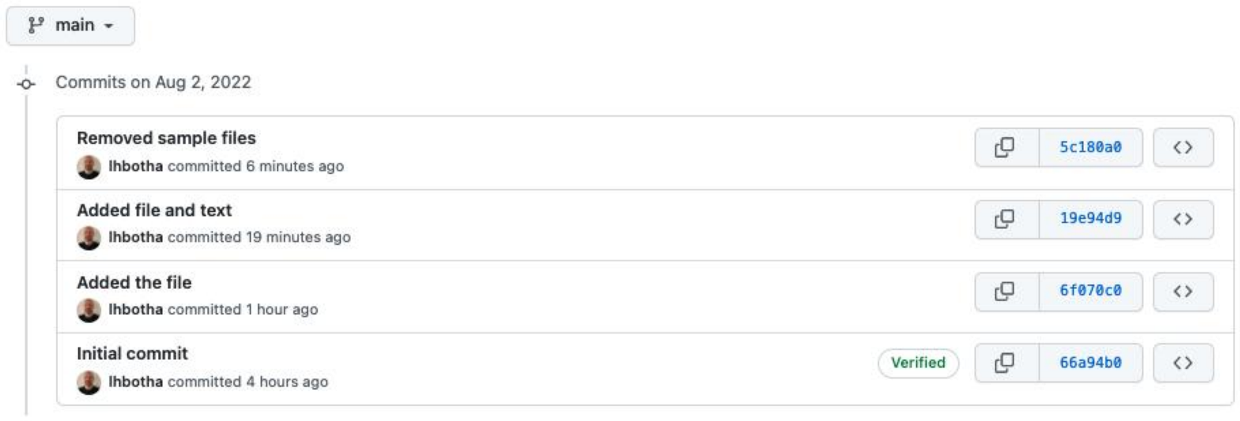
FYI. We can see the repository changes when we click on the 4 commits text.



Go to file · Add file · Code

5c180a0 4 minutes ago · 4 commits

It will open a view where we will see a history of what was done in the repository.



Commits on Aug 2, 2022

Removed sample files
lhbotha committed 6 minutes ago — 5c180a0

Added file and text
lhbotha committed 19 minutes ago — 19e94d9

Added the file
lhbotha committed 1 hour ago — 6f070c0

Initial commit — Verified
lhbotha committed 4 hours ago — 66a94b0

Mark Week 02 - 5 as done on the exercise and attendance sheet

---

git - the simple guide

just a simple guide for getting started with git. no deep shit ;)

GitHub Classroom Assignment Workflow

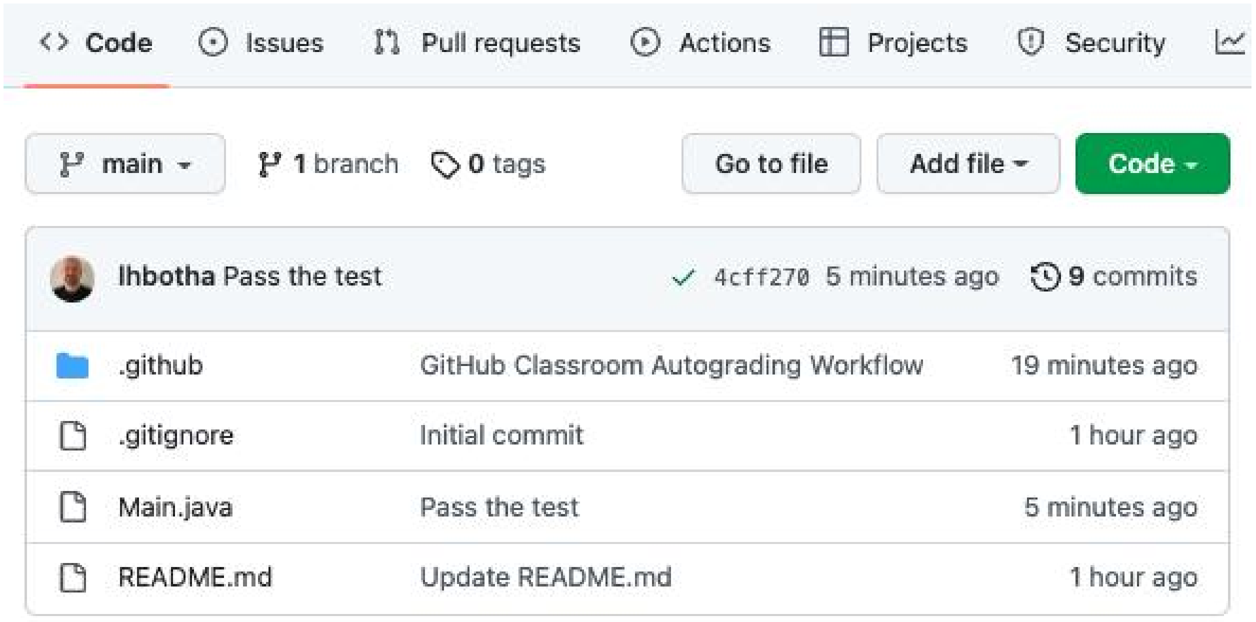
You can now use the normal flow to work on the assignment.

```
1. git clone <repository>
2. Add/Edit files
3. git add <files>
4. git commit -m "Made some changes"
5. git push
```

Some of the GitHub classroom assignments will have automated tests that will evaluate the assignment.

- ***Always start by reading the README.md***
- *If you don't follow the instructions in the README.md, the automated evaluation might not evaluate the code correctly.*

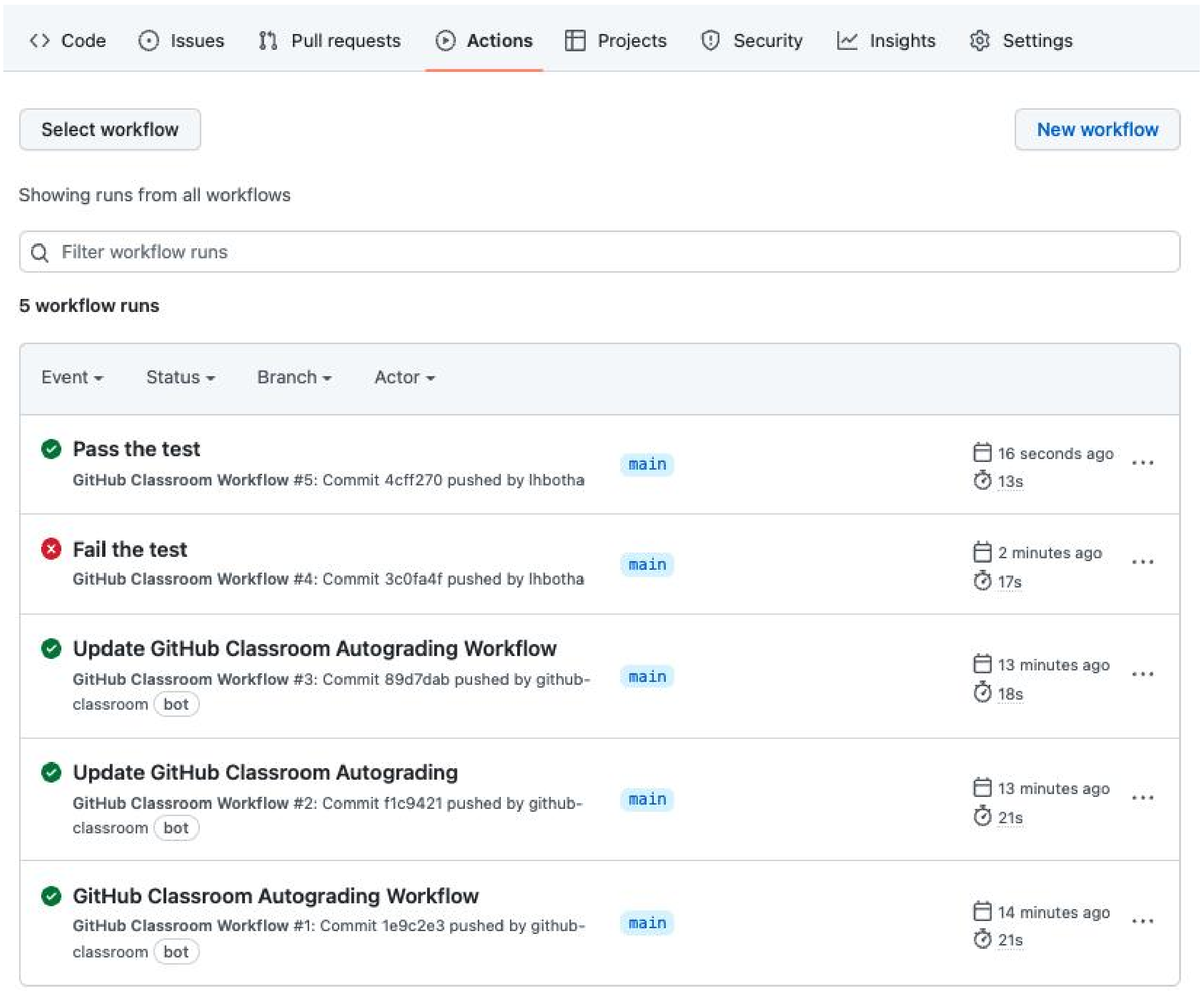
On the dashboard you will see a ✓ if your code passed the auto-grading tests or a ✗ if it failed.



lhbotha Pass the test ✓ 4cff270 5 minutes ago 9 commits

.github	GitHub Classroom Autograding Workflow	19 minutes ago
.gitignore	Initial commit	1 hour ago
Main.java	Pass the test	5 minutes ago
README.md	Update README.md	1 hour ago

You can see the testing tasks under the Actions tab, if your implementation has passed the automated tests.



Showing runs from all workflows

Filter workflow runs

5 workflow runs

✓ Pass the test
GitHub Classroom Workflow #5: Commit 4cff270 pushed by lhbotha · main · 16 seconds ago · 13s

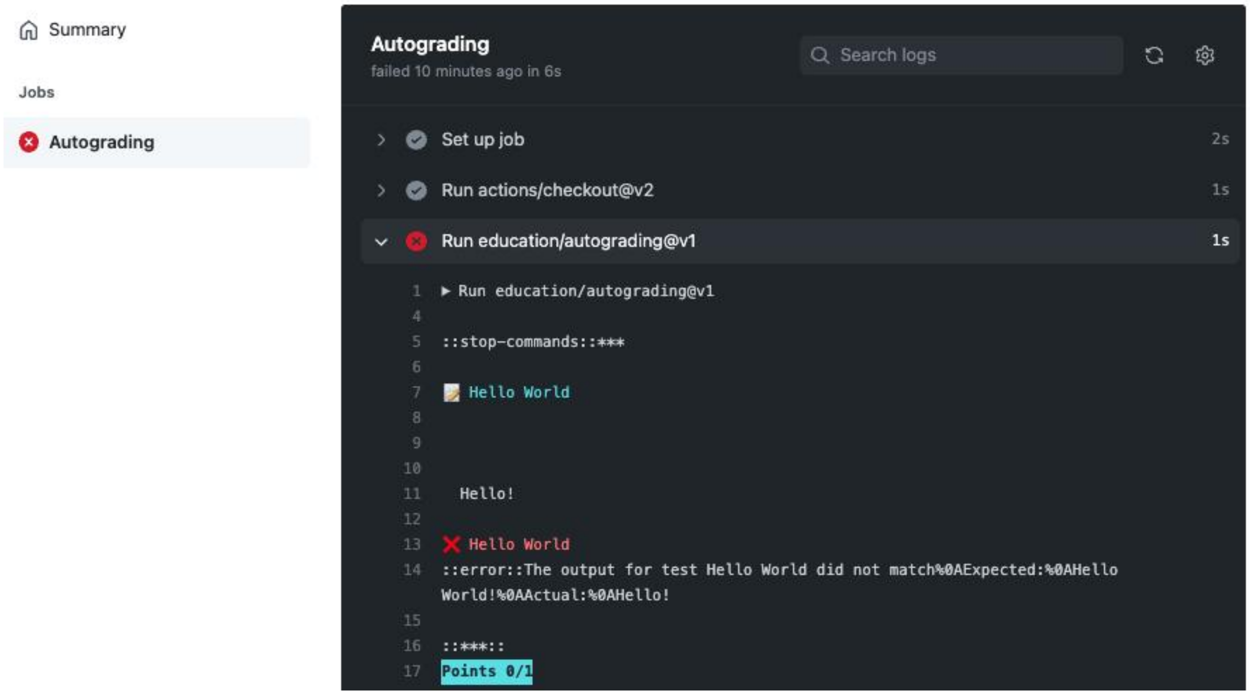
✗ Fail the test
GitHub Classroom Workflow #4: Commit 3c0fa4f pushed by lhbotha · main · 2 minutes ago · 17s

✓ Update GitHub Classroom Autograding Workflow
GitHub Classroom Workflow #3: Commit 89d7dab pushed by github-classroom bot · main · 13 minutes ago · 18s

✓ Update GitHub Classroom Autograding
GitHub Classroom Workflow #2: Commit f1c9421 pushed by github-classroom bot · main · 13 minutes ago · 21s

✓ GitHub Classroom Autograding Workflow
GitHub Classroom Workflow #1: Commit 1e9c2e3 pushed by github-classroom bot · main · 14 minutes ago · 21s

You can drill down into the action to see exactly what test failed.


Summary
Jobs
Autograding
Autograding
failed 10 minutes ago in 6s
Search logs
Set up job 2s
Run actions/checkout@v2 1s
Run education/autograding@v1 1s

```
▶ Run education/autograding@v1

::stop-commands::***

📝 Hello World



  Hello!

❌ Hello World
::error::The output for test Hello World did not match%0AExpected:%0AHello
World!%0AActual:%0AHello!

::***::
Points 0/1
```

You can make changes to fix the assignment and push it to the assignment repository.

- ⚠ *What is that points? In the future I might use the points from the tests.*
- *Currently, until further notice 1 exercise equals 1 point on the sheet, also if you didn't get it to pass.*
-
- *Trying is the most important thing!*

Completing the Assignment

The assignment is considered submitted once you have pushed some changes to the repository before the assignment deadline. The deadline is on the course page. The automated system will stop evaluating the code, once the deadline for the assignment has passed. The last commit that was evaluated before the deadline is the commit linked to classroom and considered the final submission.