CODE MANAGEMENT

Software development with GIT
QSciTech-Quantum BC-CMC Virtual Workshop 2024

Tania Belabbas

January 24th, 2023

tania.belababs@usherbrooke.ca

Outline

- Why use code management?
- Description of Git
- Using Git with the command line

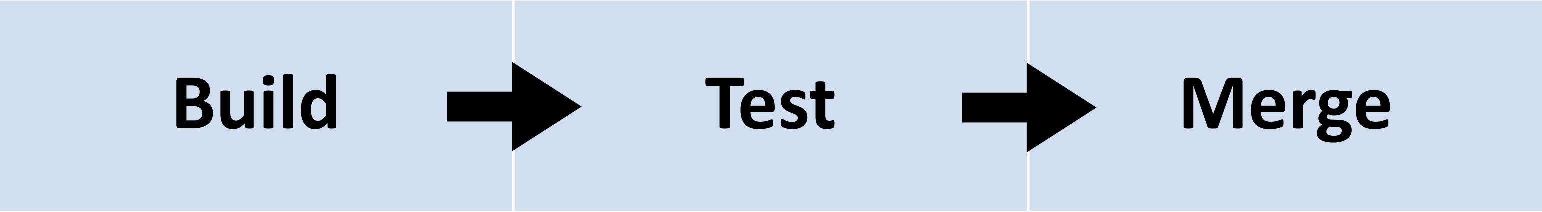
Code management

- Working in big groups
- Keeping a **history**
- Always having a working version
- Insuring code quality through continuous integration

Continuous integration

- **Merging** you own changes into a base version.
- The changes are then thoroughly tested.

***Breaking the build* is costly!**



Build → Test → Merge

Continuous integration - FYI

- **CI/CD** is continuous integration and continuous deployment (or continuous delivery).
- As the name suggests, continuous deployment is the process of scheduling automatic releases of a code base and is not feasible without continuous integration.

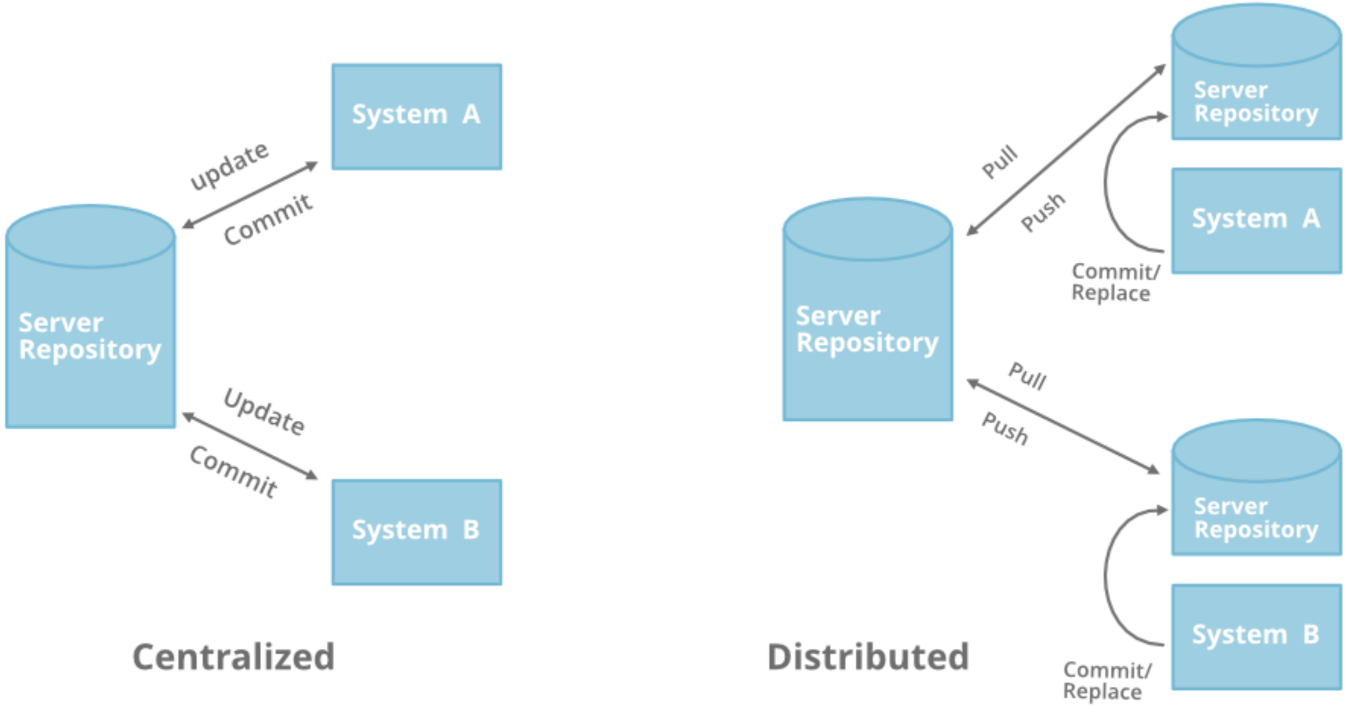
VERSION CONTROL SYSTEM

Version control system

- Version control systems are made for code management.
- Allows developers of a code base to track versions of their code.
- A safety net for developers.

Version control system

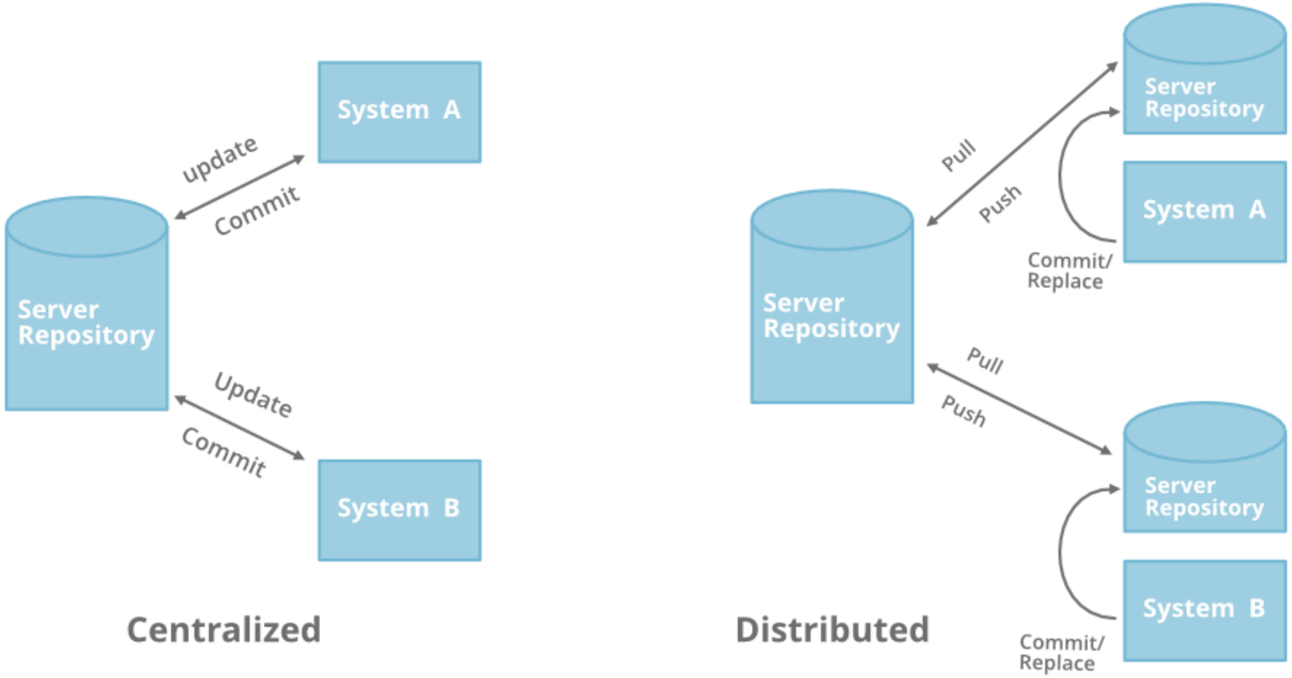
- Three types : localized, centralized and distributed VCS.



centralized-vs-distributed-version-control-which-one-should-we-choose/

Version control systems

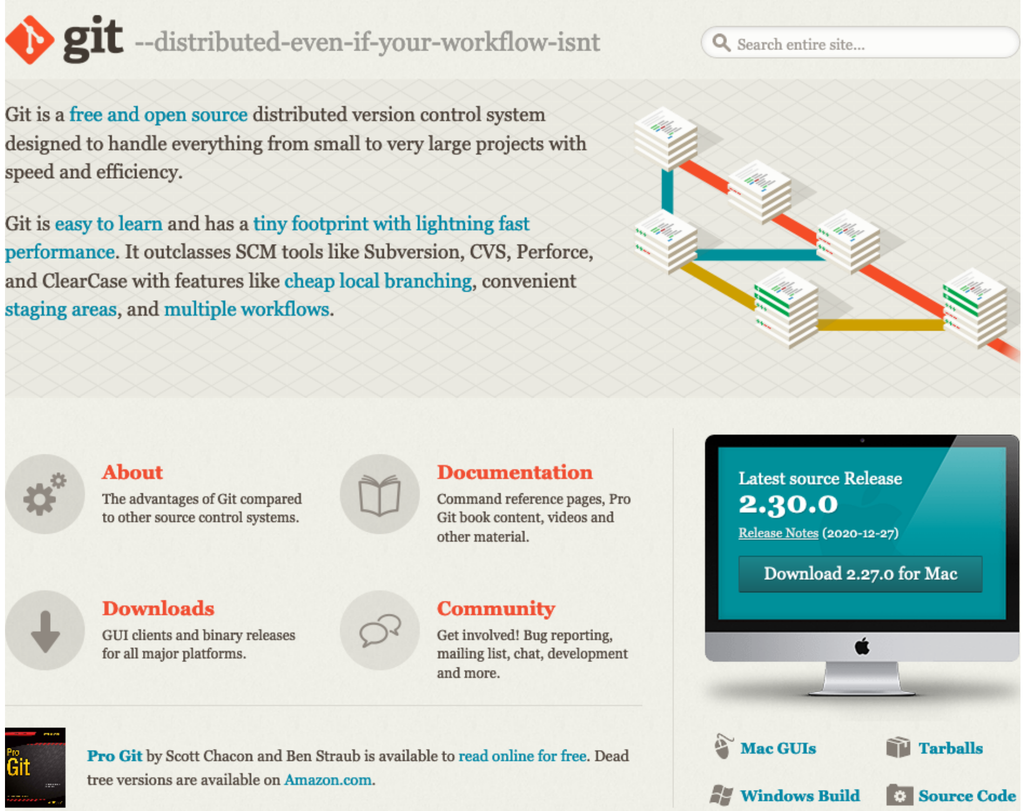
- Subversion (SVN)
  - Centralized system.
- Mercurial
  - Distributed system



centralized-vs-distributed-version-control-which-one-should-we-choose/

Version control systems

- Git
  - Most commonly used system.
  - Distributed system
  - Open source
  - Cross platform
  - Tracks changes in text files



git --distributed-even-if-your-workflow-isnt

Search entire site...

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

About
The advantages of Git compared to other source control systems.

Documentation
Command reference pages, Pro Git book content, videos and other material.

Downloads
GUI clients and binary releases for all major platforms.

Community
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release
2.30.0
Release Notes (2020-12-27)

Download 2.27.0 for Mac

Pro Git by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Mac GUIs
Tarballs
Windows Build
Source Code

https://git-scm.com/

Git Platforms

Git Platforms

- We will be using GitHub.

GitHub

Git Platforms

- Let’s first setup your Git options :

1. Open your terminal (command line)

2. Set your username:

```
git config --global user.name “username”
```

3.Set your email address:

```
git config --global user.email “name@em.com”
```

Git repository

- A git repository is the .git/ folder inside a project.
- Tracks all changes made to files in your project.
- Builds a history over time.

Git repository

- It contains
  - **HEAD** points to the branch you currently have checked out
  - **index** staging area information
  - **objects/** stores all the content for your database
  - **refs/** stores pointers into commit objects (e.g. branches, tags, remotes)

Git organizations

- A way for GitHub to organize different code basis.
- What is accessible to you varies according to the organization.
- In the context of this workshop, it is suggested you create your team repository online first.

Git repository

- Create a git repository:

  1. Open your terminal.
  2. Navigate to the new created folder.
  3. Use the following command

```
git init
```

- Or you use the online option by first creating an online repository.
- Other ways are available (VSCode or GitHub desktop).

Git repository

- Create a git repository online:

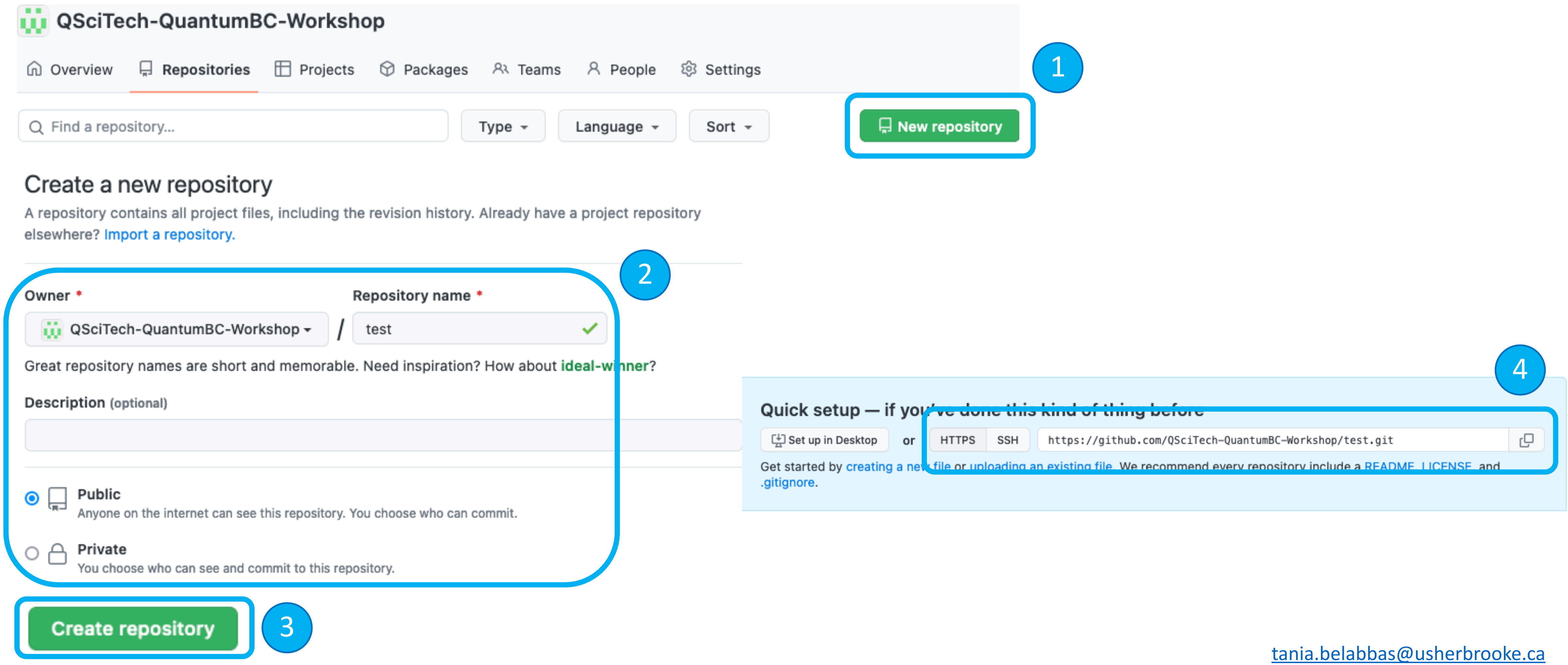
1. Open your Github page and navigate to the repositories menu.
2. Create a new repository.

- Synchronise your repository locally :

1. Open your terminal.
2. Navigate to the folder where you want your repository then use the command :

```
git clone <url to git repo>
```

Create a repository in GitHub



QSciTech-QuantumBC-Workshop

Create a new repository

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

Owner * QSciTech-QuantumBC-Workshop / Repository name * test

Great repository names are short and memorable. Need inspiration? How about ideal-winner?

Description (optional)

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

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

Quick setup — if you've done this kind of thing before

```
https://github.com/QSciTech-QuantumBC-Workshop/test.git
```

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

tania.belabbas@usherbrooke.ca

Git repository

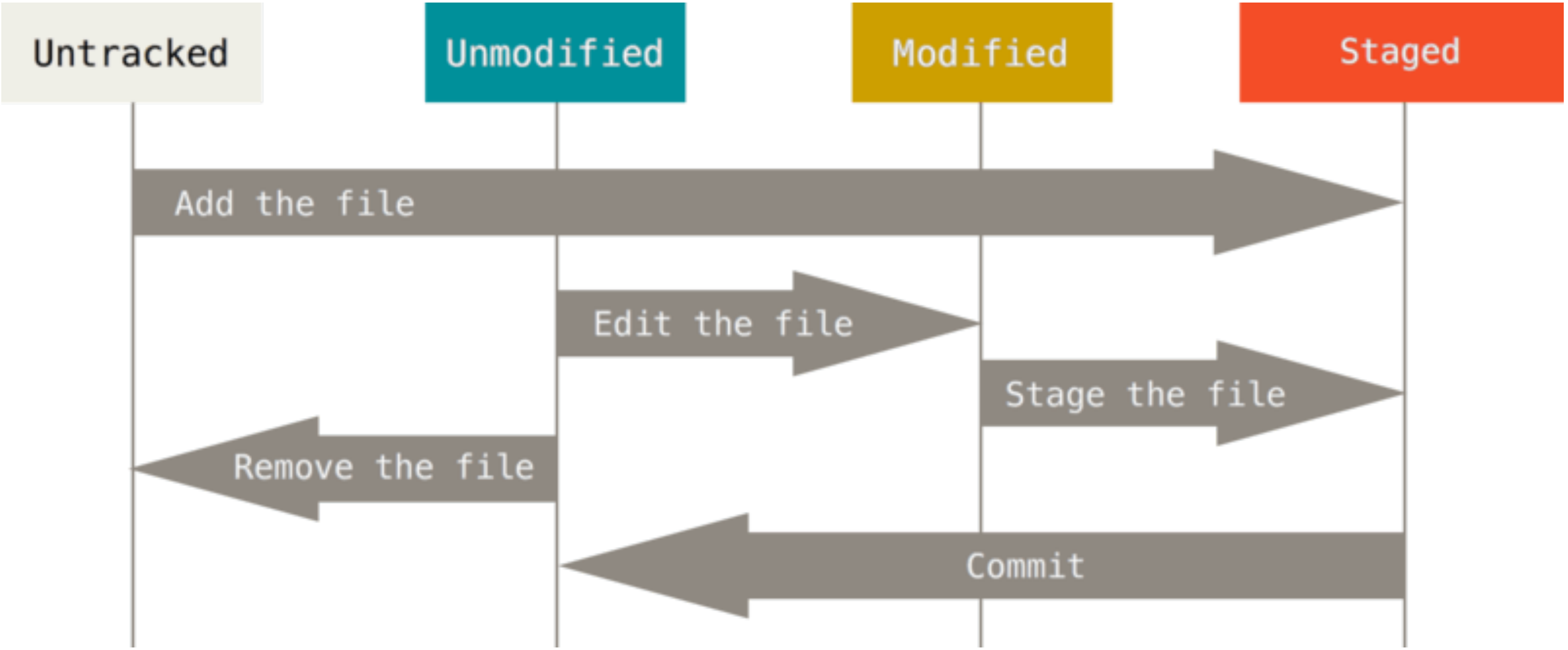
- Check the status of a repository:

```
git status
```

- Add a file to your folder.

- Check the status again.

File status lifecycle



https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository

Managing files status

- Check the status of a repository:

```
git status
```

- Check the modifications made to tracked files

```
git diff [<name of file>]
```

- Add files to staged files

```
git add <name of file>
```

Git Commits

- Snapshots or milestones along the timeline of a project.
- A commit is identified by a hash, a 40-character hexadecimal string like *c93b502f86c81ef5eef444f77c2b8e61a5b2f2e9*

Git Commits

- **You should make new commits often, based around logical units of change!**
- Create a commit :

```
git commit -m <your message>
```

Commit messages

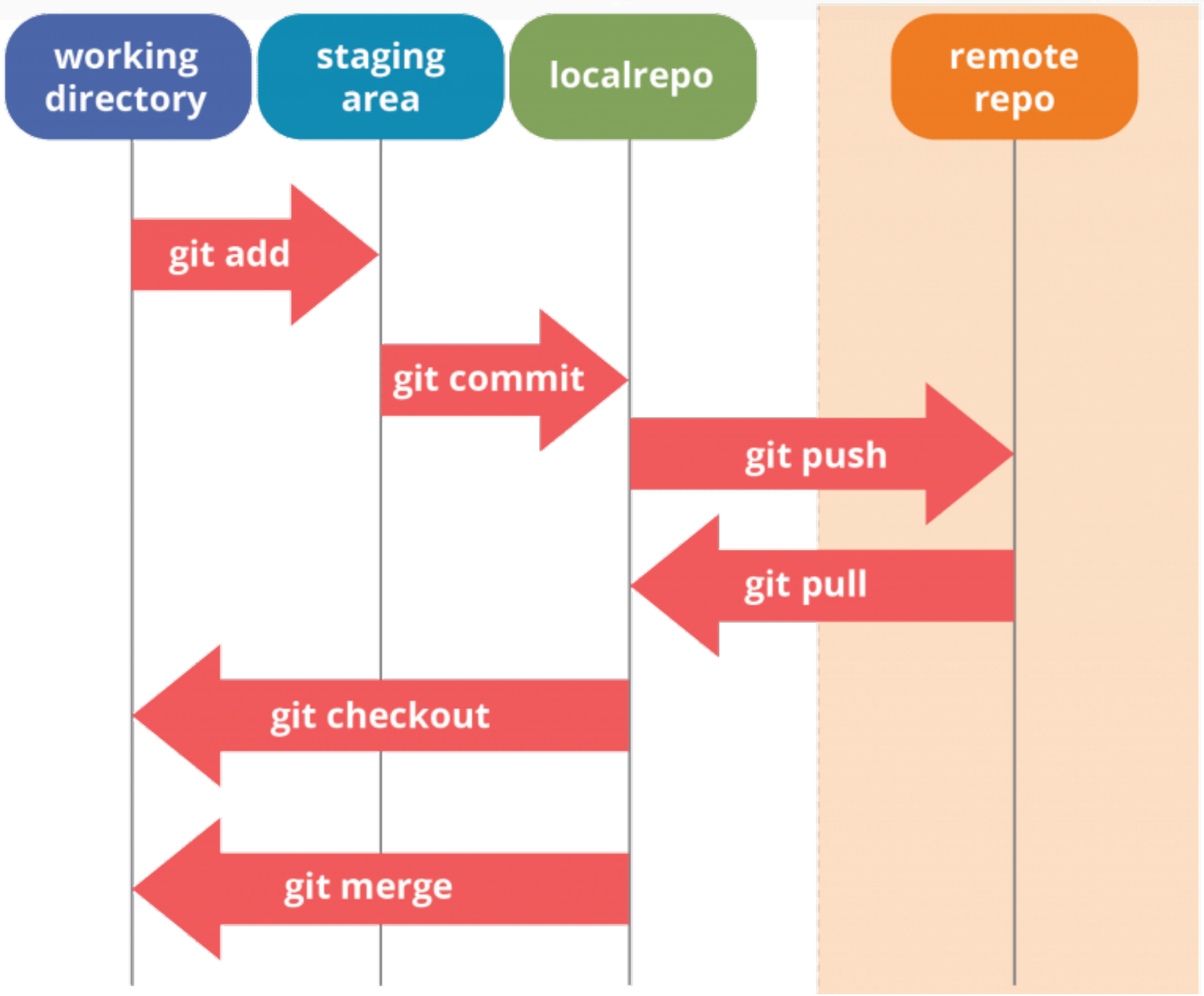
- Title
  - Commit message starts with a single short (less than 50 characters) line summarizing the change, followed by a blank line.
- Description
  - More detailed explanatory text, if necessary.
  - Wraps the body at 72 characters
- Display the history of commits

```
git log
```

Overview of file stages

- Adding an untracked file allows it to be in the working directory.

- Files type mentioned in the **.gitignore** file are **ignored** by git and **untracked**.



https://hboeving.dev/blog/git-graph-p1/

tania.belabbas@usherbrooke.ca

Exercise 1

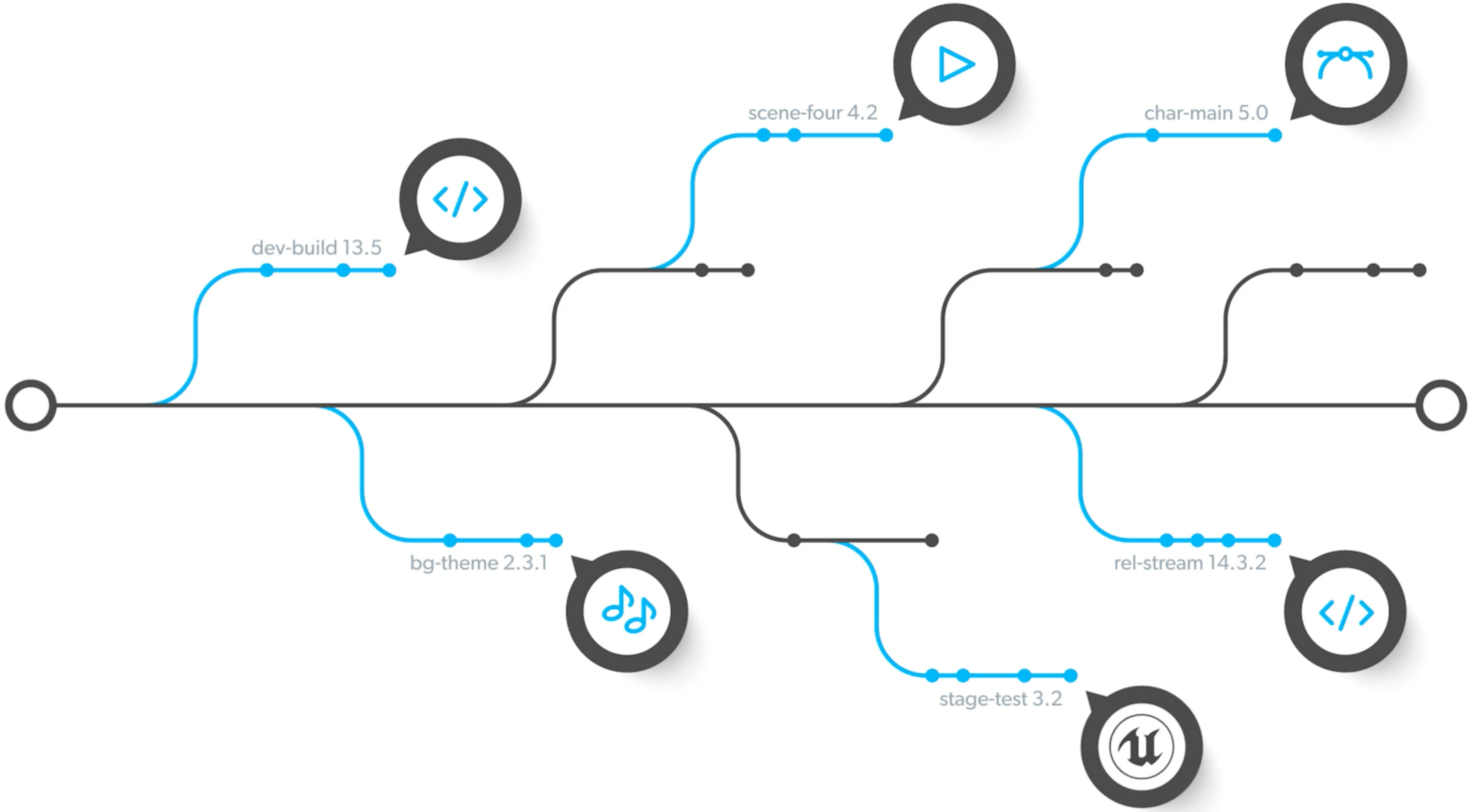
Objectives

- Setup git config (name, email, editor)
- Create a git repo
- Experiment with file lifecycle
- Create a commit and check commits history

Branching

- Default branch name is “main” (or “master”).
- You can create as many branches as you need, merge them and delete them at will.
- Branches isolate your development work from other branches in the repository. For example, you could use a branch to develop a new feature or fix a bug.

Branching



https://www.perforce.com/blog/vcs/what-is-version-control

Managing branches

- List / [create] branch:

```
git branch [<branch name>]
```

- Switch to branch

```
git checkout <branch name>
```

- Merge other branch into current branch

```
git merge <branch name>
```

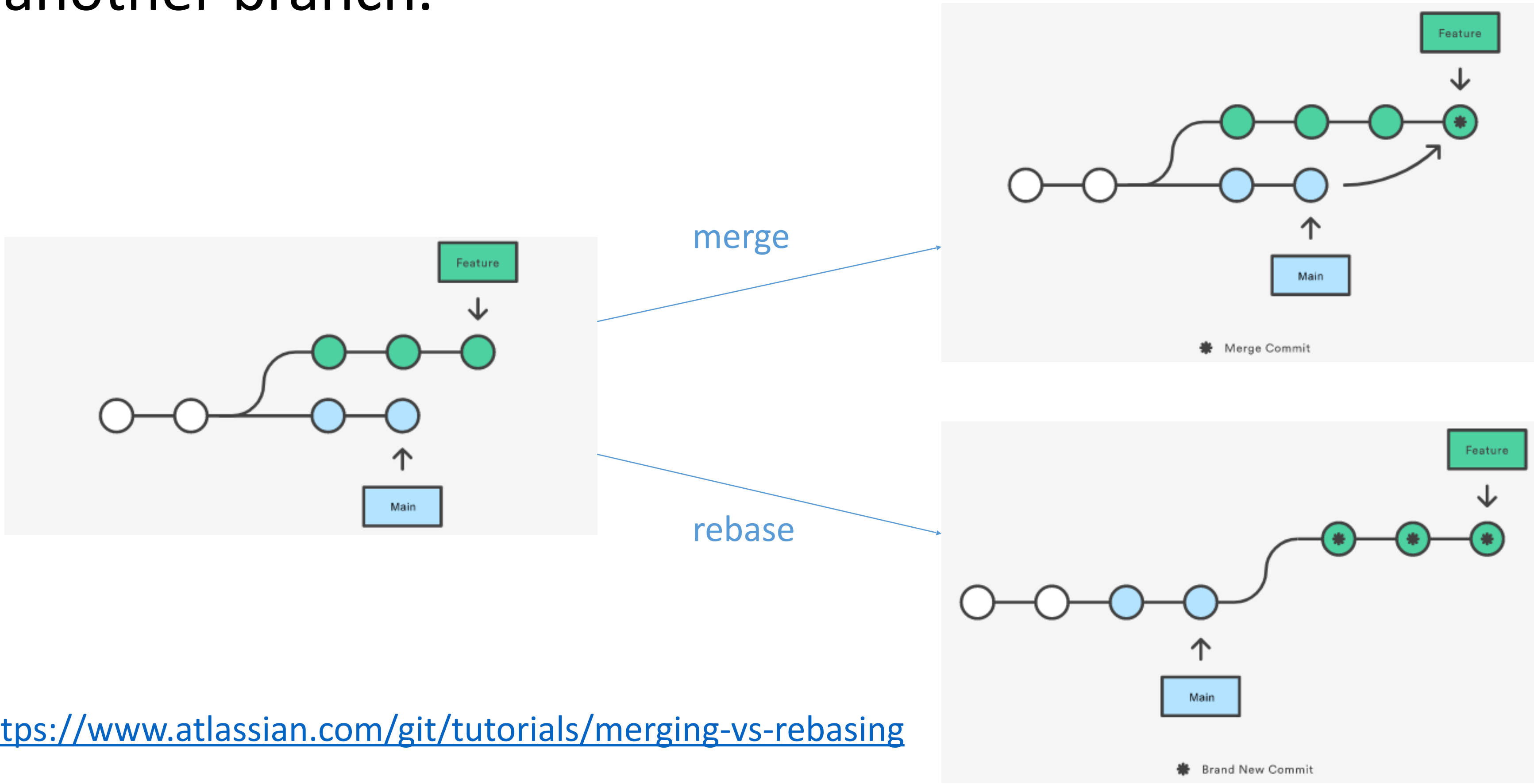
Exercise 2

Objectives

- Create branch B
- Switch to branch B
- Create a new file in branch B and commit
- Back to branch A, merge branch B into A

Merging Vs rebasing

- Both commands are used to integrate commits from one branch into another branch.



https://www.atlassian.com/git/tutorials/merging-vs-rebasing

Managing branches

- Rebase onto another branch

```
git rebase <branch name>
```

Merge conflicts

- Happens when merging two branches (or rebasing) and the same lines are modified.
- Git will ask you to reconciliate the modifications.

```
<<<<<<< HEAD

<the code that is in the current branch>

=======

<the code that is in the incoming branch>

>>>>>>> <other branch>
```

Remote repository

- Remote repositories are versions of your project that are hosted on the Internet or network somewhere.
- Collaborating with others involves managing these remote repositories and pushing and pulling data to and from them when you need to share work.

Working with remote

- Add a link to a remote repository in your local repo

```
git remote add origin <url to git repo>
```

- Show remote

```
git remote -v
```

- Push local branch to remote

```
git push origin <local>:<remote>
```

Git repository

- Create a git repository:

  1. Open your terminal.

  2. Navigate to the new created folder.

  3. Use the following command

```
git init
```

- Or you use the online version.

- Other ways are available (VSCode or GitHub desktop).

Exercise 5

Objectives

- Create a **public** repository on GitHub
- Add remote to local repo
- Push code to remote repository

Working with remote (2)

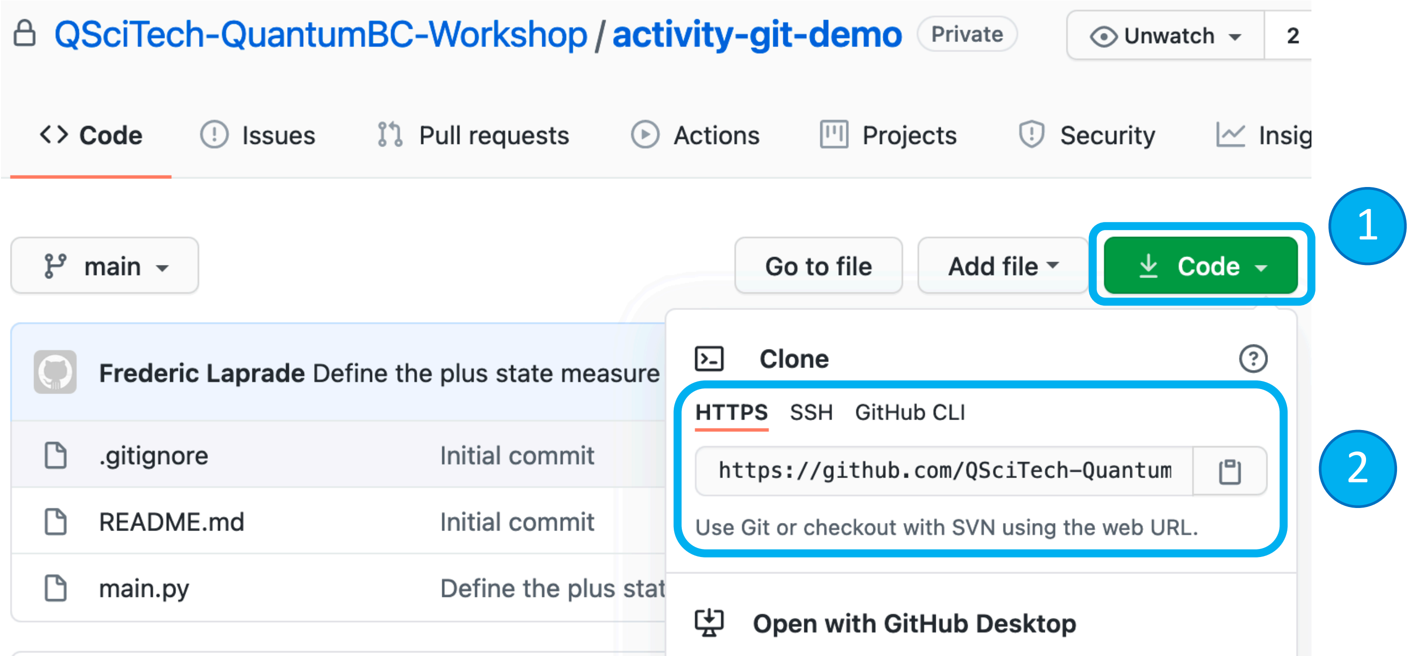
- Cloning a git repo

```
git clone <url to git repo>
```

- Pulling changes from remote repository to local repo

```
git pull origin <remote>:<local>
```

Clone a git repository



QSciTech-QuantumBC-Workshop / activity-git-demo Private

Frederic Laprade Define the plus state measure

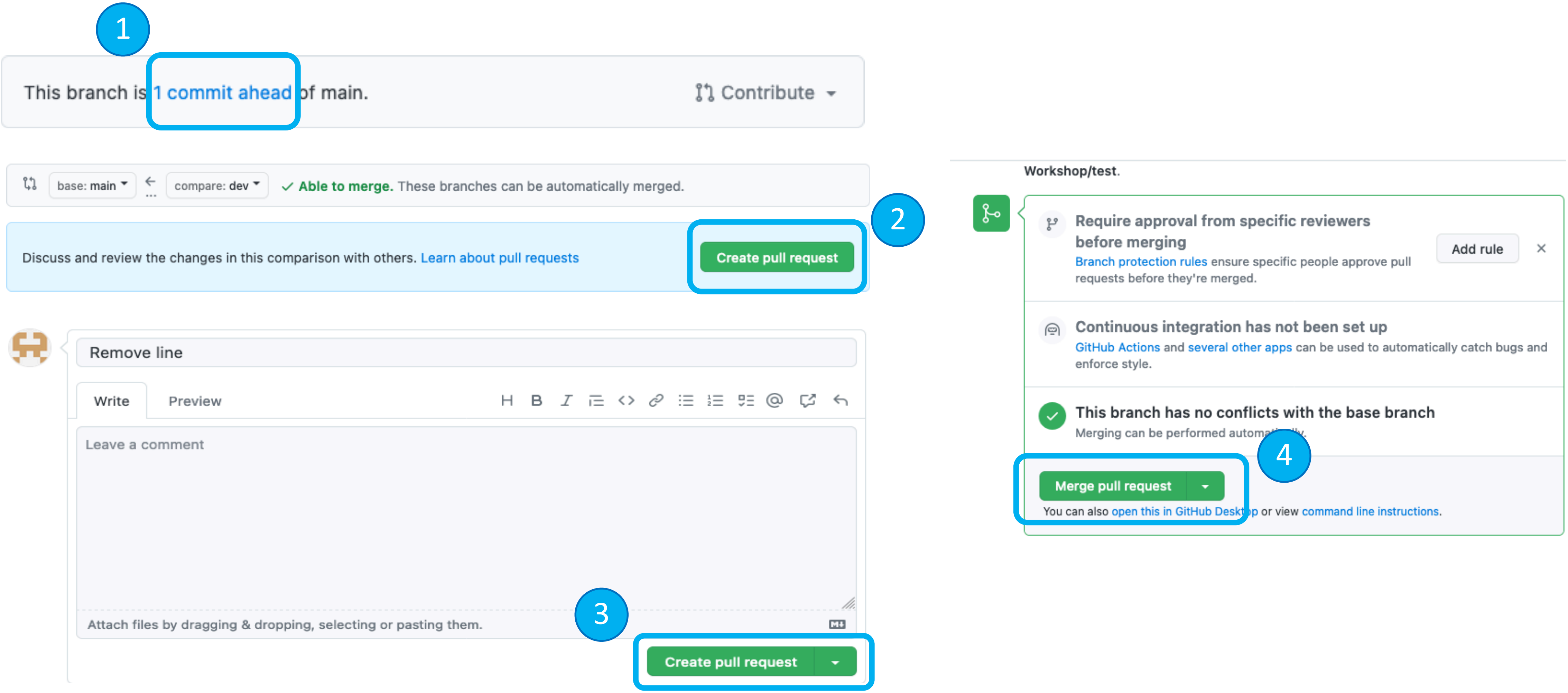
.gitignore	Initial commit
README.md	Initial commit
main.py	Define the plus stat

```
https://github.com/QSciTech-Quantum
```

Use Git or checkout with SVN using the web URL.

tania.belabbas@usherbrooke.ca

Create a pull request



1

This branch is 1 commit ahead of main.

✓ Able to merge. These branches can be automatically merged.

2

Discuss and review the changes in this comparison with others. Learn about pull requests

Remove line

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

3

Workshop/test.

Require approval from specific reviewers before merging

Branch protection rules ensure specific people approve pull requests before they're merged.

Continuous integration has not been set up

GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch

Merging can be performed automatically.

4

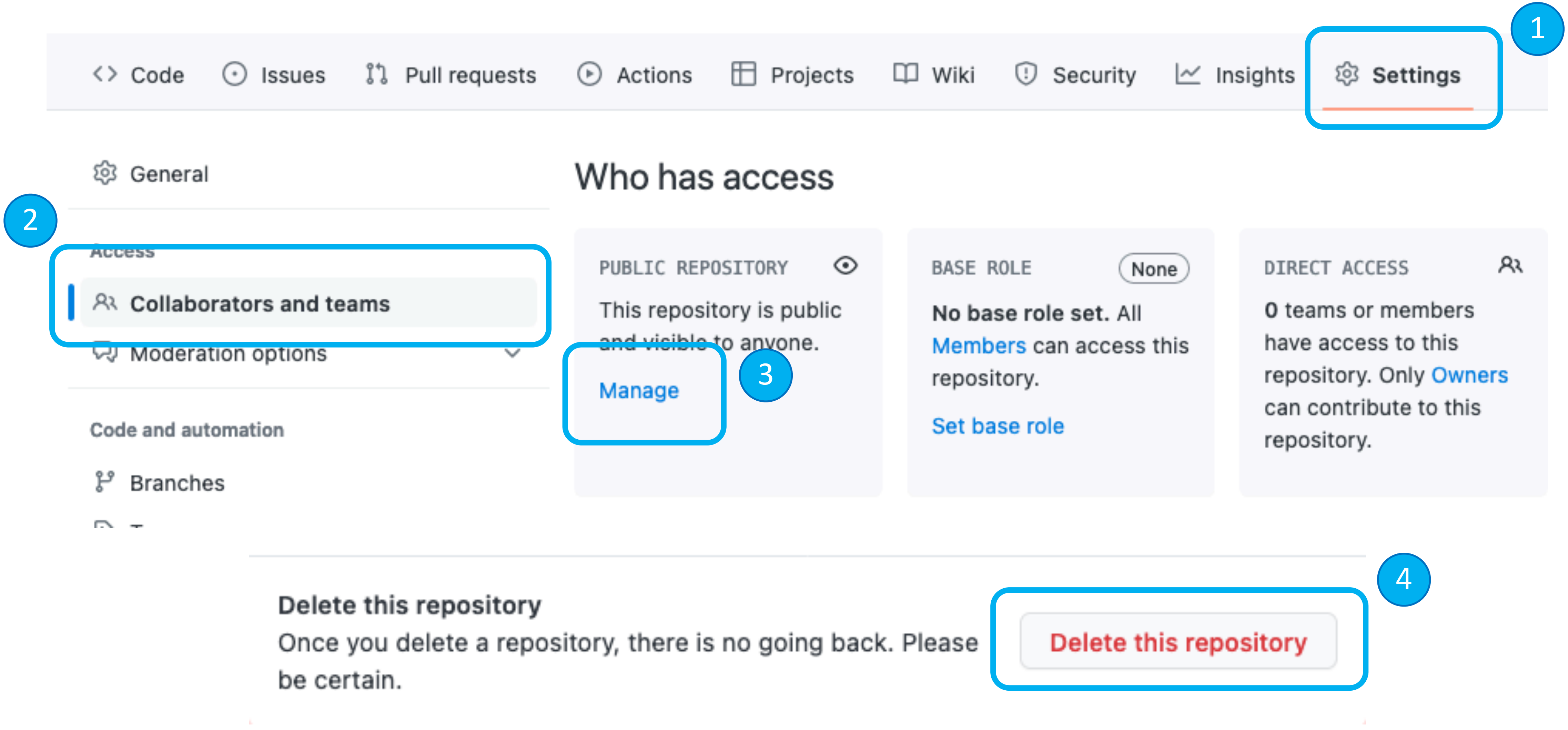
You can also open this in GitHub Desktop or view command line instructions.

Exercise 5

Objectives

- Create a pull request

Delete the repository in GitHub



Who has access

PUBLIC REPOSITORY

This repository is public and visible to anyone.

BASE ROLE

No base role set. All Members can access this repository.

DIRECT ACCESS

0 teams or members have access to this repository. Only Owners can contribute to this repository.

Delete this repository

Once you delete a repository, there is no going back. Please be certain.

tania.belabbas@usherbrooke.ca

Git best practices

- Each task should be developed in its own branch
- Frequently merge the target branch or rebase on target branch. This will reduce headaches from merge conflicts
- Commits in your branch are cheap, do it often!
- When merging with target branch, re-write your dev history first (can be done automatically).

Other useful git commands

- git blame
- git cherry-pick
- git reset
- git tag (lightweight/annotated)
- git stash
- git checkout <*commit*>
- git branch –d (-D) <*branch_name*>

References

Git

- https://git-scm.com/book/en/v2
- https://guides.github.com/
- https://www.atlassian.com/git/tutorials
- https://githowto.com/
- https://learngitbranching.js.org/