Version control

L5 Data Engineer Higher Apprenticeship
Module 3 / 2 (“**Programming and Scripting Essentials**”)
Topic 2 / 8

rev. 1 (2024)

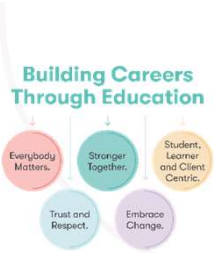
Ice breaker: Discussion

A bit of fun to start…

1. If you were a superhero, what would your superpower be and why?

2. What's the most unusual item within your reach right now, and is there a story behind it?

3. Can you share an instance where you leveraged DevOps, version control, and CI/CD principles in a data engineering project? How did it contribute to the project's success?



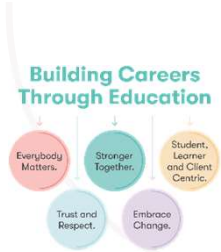
Submit your responses to the chat or turn on your microphone

BPP

Case study discussion

The impact of successful DevOps and version control…

- A well-known streaming service company faced challenges in managing vast data resources
- They implemented Git for version control and GitHub for code reviews
- Established **"CI/CD pipelines"** for efficient data engineering processes



prime video

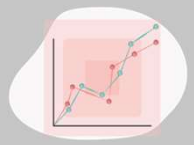
Disney+

NETFLIX

hulu

YouTube

Leading streaming services



The results – achieved efficient data processing, frequent algorithm updates, and reliable data product delivery

BPP

Case study discussion

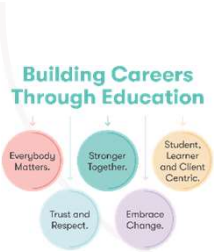
The impact of successful DevOps and version control…

In light of the case study, consider the following questions:

1.How did the use of DevOps principles enhance collaboration, automation, and monitoring in the company's data engineering practices?

2.How did the use of version control and CI/CD processes contribute to the efficient and reliable delivery of data products?

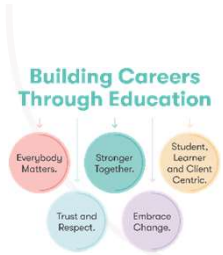
3.How did conducting code reviews on GitHub help maintain code quality and facilitate team collaboration?



Leading streaming services

Discussion activity: Use your knowledge from the topic e-learning to discuss these questions.

Webinar agenda



This webinar will cover the following:

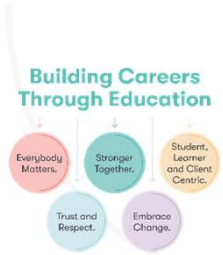
- An introduction to version control
- Git and its applications
- Practical demonstration - Working with repositories
- Git branching, changes and merges

Webinar length: 3 hours

Learning objectives

This webinar supports completion of the following outcomes:

- Understand the principles of DevOps and its application in data engineering to enhance collaboration, automation, and monitoring
- Gain practical knowledge of version control, Continuous Integration and Continuous Deployment (CI/CD) processes to ensure efficient and reliable delivery of data products
- Conduct effective code reviews using GitHub to maintain code quality and facilitate team collaboration



BPP

An introduction to version control

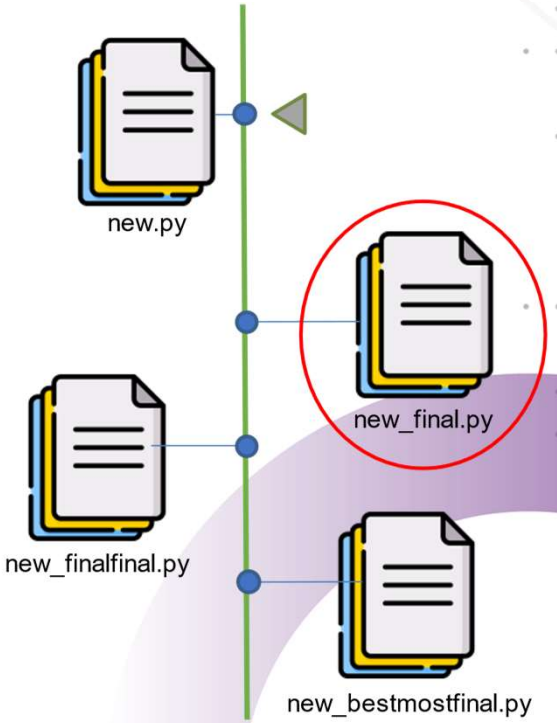
BPP

Introduction to version control

What you need to know…

- Version control keeps record of your changes in codes
- Allows you to revert any changes and go back to a previous state
- It's a time machine

The core fundamentals of DevOps: Automation, tools and culture



An example version control sequence

BPP

Practical example

File for the organisation of a party…

Without Versioning

organization_party_v0.txt

organization_party _v1.txt

organization_party _v2.txt

organization_party _v2_dj.txt

organization_party _v2_orch.txt

organization_party _v3.txt

- Lots of different versions of same file
- Who remember which one is what?
- How about combining the different versions

With versioning

organization_party.txt

.git (stores difference from last commit, date, author, commit ID…)

- Only one working file
- History of the file documented and easy to consult
- Easy to combine

BPP

Different version control types

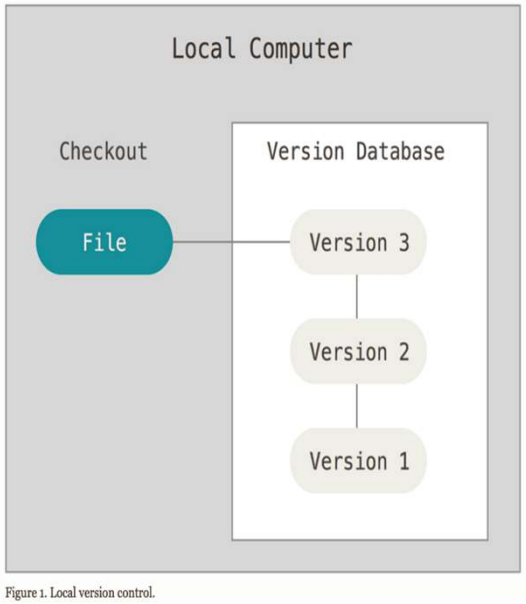
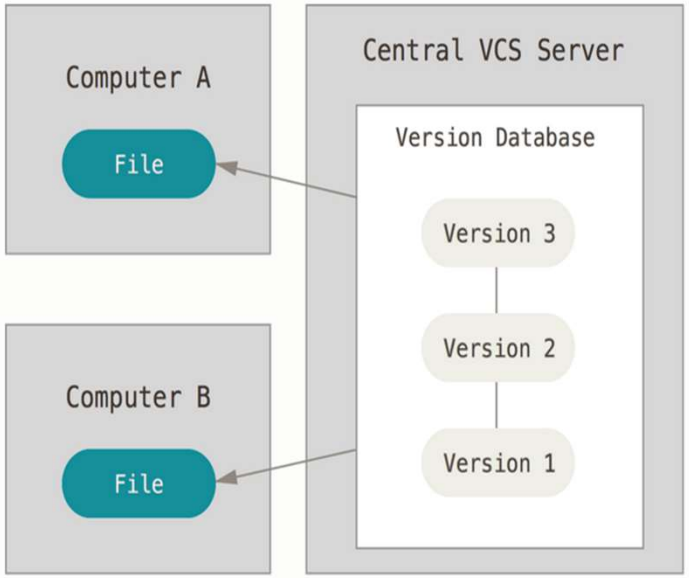
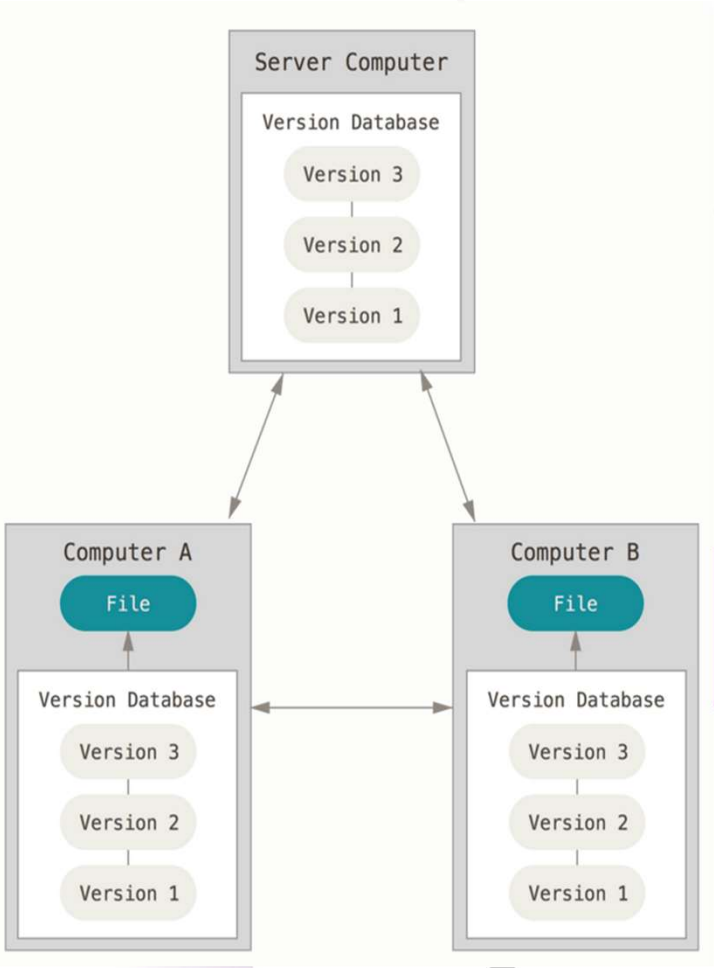


Figure 1. Local version control.

Local version control system (VCS)



Central version control system (eg SVN, …)



Distributed version control system (eg Git, Mercurial,…)

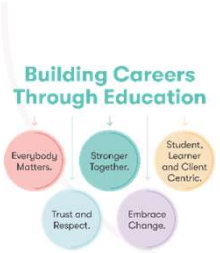
Git and its applications

BPP

Introducing Git

What is it and why use it?

- Free and open-source version control system
- Created by Linus Torvalds in 2005
- Widely used, fast, and offline-capable
- Efficient storage of file changes
- Good at merging code versions
- Enables easy code exchange and backup



BPP

Introducing Git

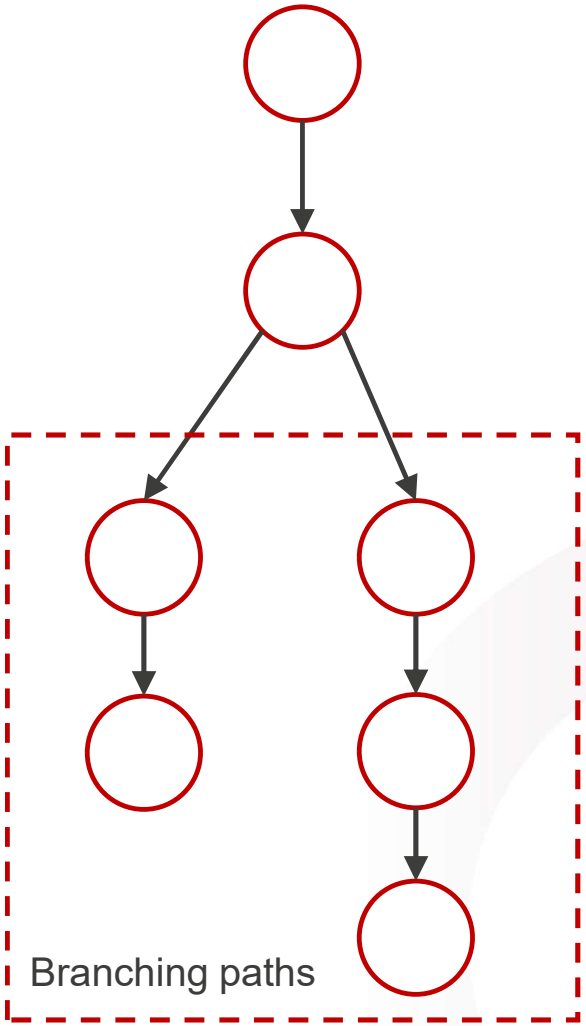
Why bother learning about Git...

Git solves data engineering problems related to collaboration and reproducibility (see: link below)

Git has simple – yet powerful – concepts (see branching in the diagram on the right)

Multiple services available for it

https://towardsdatascience.com/git-for-data-engineers-a8b979d8b2ab

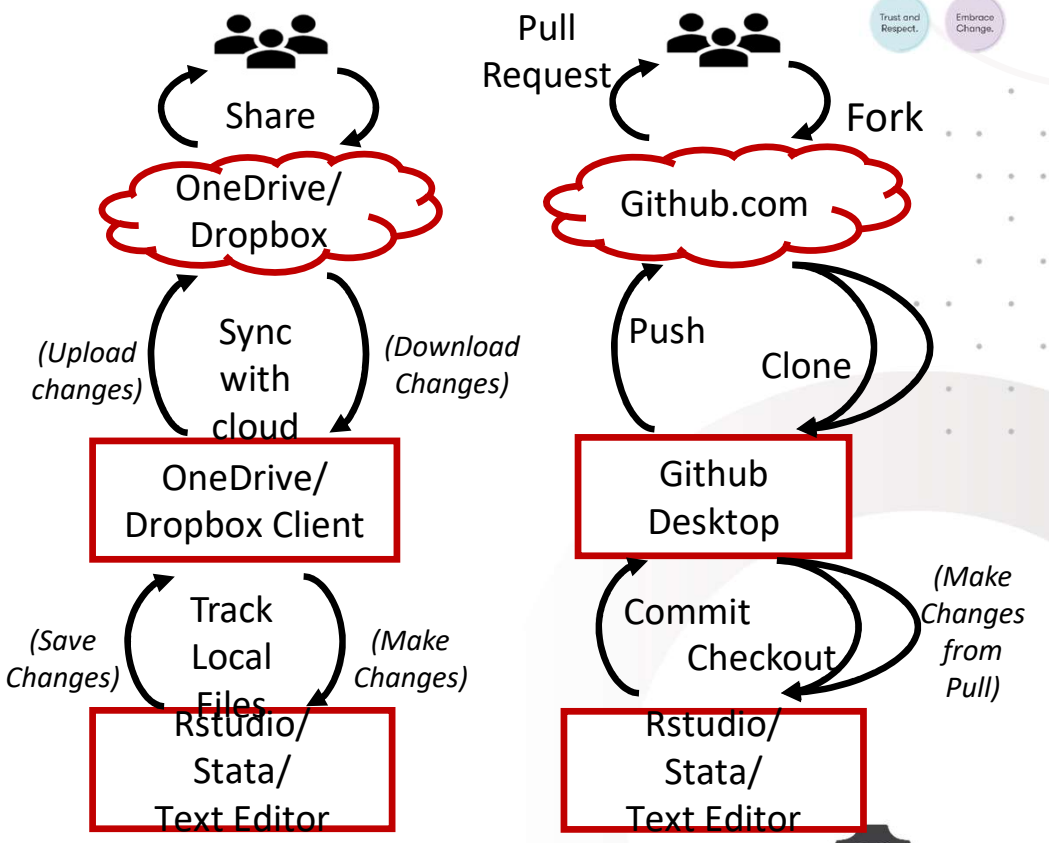


Branching shown using a graph

Git vs. OneDrive/Dropbox

Git works similarly, but with more control/ at each step

Category	Git Jargon	Description
Load/sync	Clone	Download a repository
	Pull	Sync with changes in the repository
	Checkout	Show specific versions of files
Save/sync	Commit	Make a snapshot of changes
	Push	Upload changes to repository
Share	Fork	Create your own copy of someone else's repository to make your own changes with
	Pull request	Request to merge two branches together



OneDrive vs Git functionality

Learning to use Git ‘properly’

Common misconceptions…

- **Usual way to learn Git is via command line**
  - Requires familiarity with using the command line
  - Usually taught in the context of Linux

We don’t have to do any of this as Git has point-and-click interfaces available.

Point-and-click interfaces

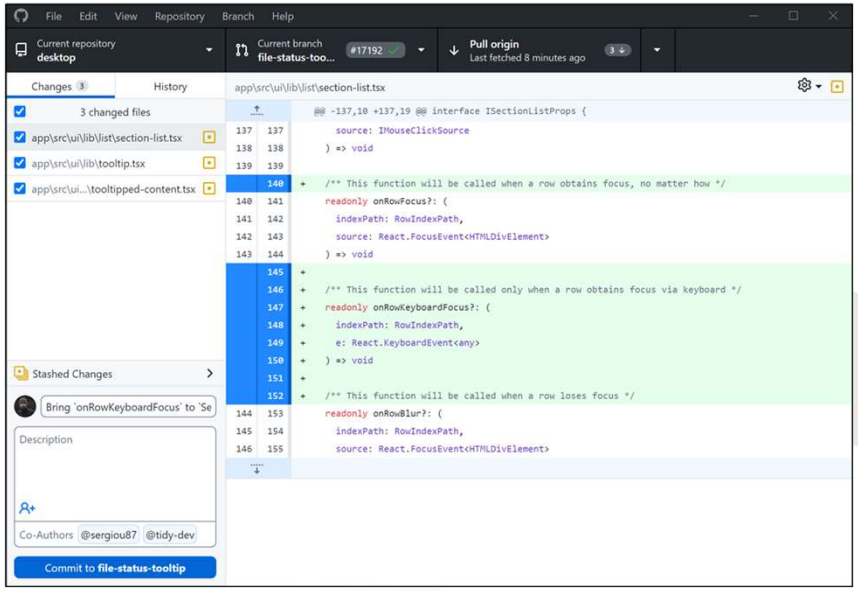
What are they and which one should you use?

- https://desktop.github.com/

Many other options available:
https://git-scm.com/downloads/guis

- Some coding environments come with Git integration, e.g.
  - RStudio
  - PyCharm
  - VSCode

It doesn't matter which one you use, just use whatever you find easiest!

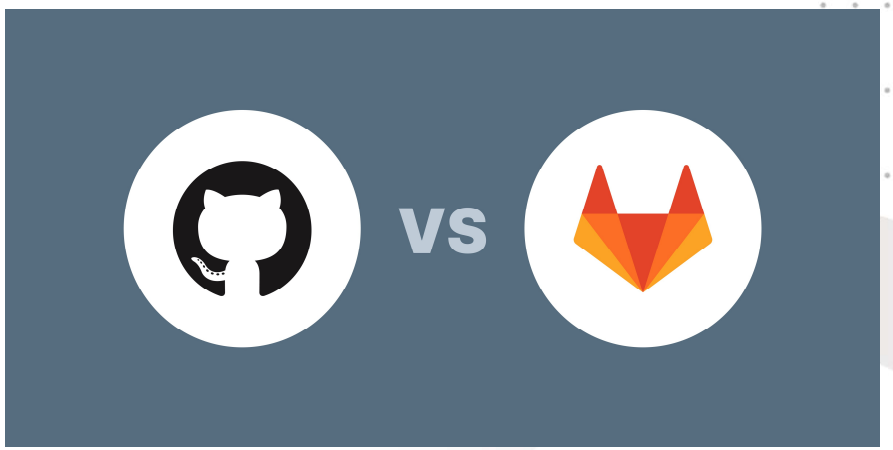


One of Git's visual interfaces

GitLab / GitHub

Do you know the difference?

- These are different websites but both tools are built on top of Git
- You can think of them as online storage of Git repositories
- Repository management that acts as your single source of truth in a distributed version control workflow
- These also includes code review tools with tons of collaboration and flow management features
- Both also have an issue tracker with issue boards and a project wiki.

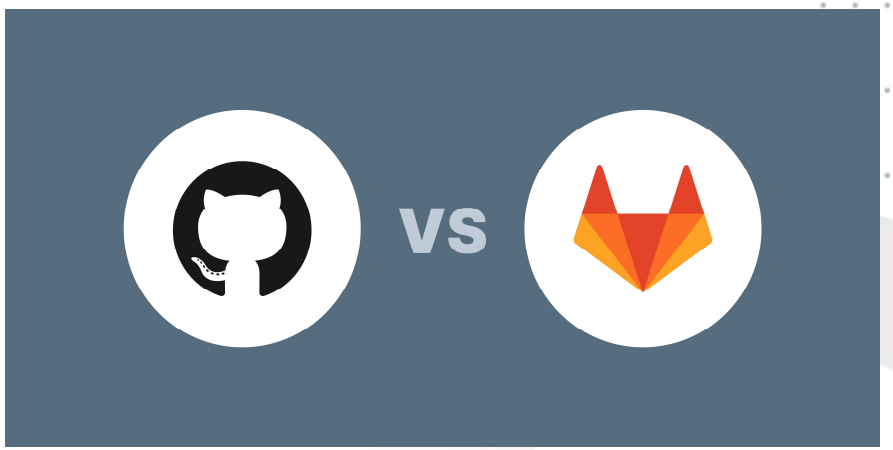


Comparing GitLab with GitHub

GitLab / GitHub

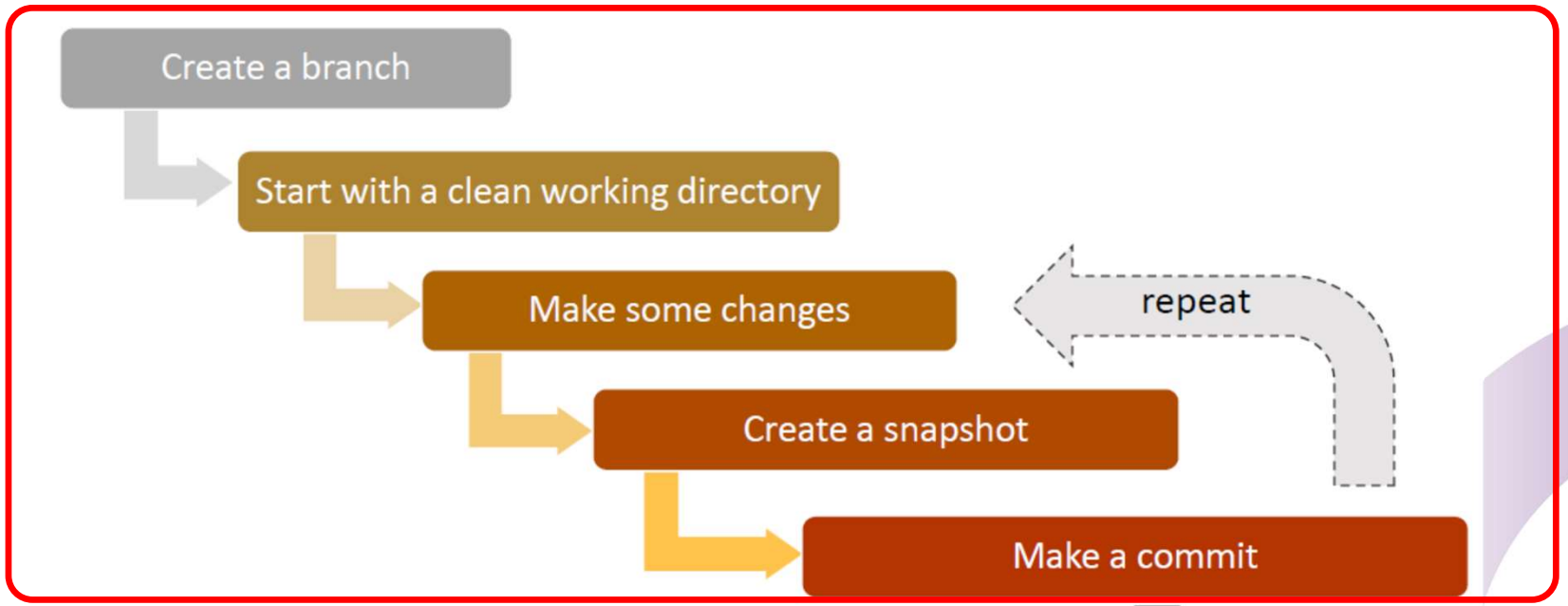
Do you know the difference?

Criteria	GitHub	GitLab
Ownership	Microsoft	GitLab Inc
Community & Publicity	100M+ & high publicity	30M+? & Lower Publicity
Repo storage (public)	Unlimited	5 GB
Repo storage (private)	500 MB	5 GB
CI Compute Minutes (public)	Unlimited	400 minutes (50,000 officially)
CI Compute Minutes (private)	2000 minutes	400 minutes
Max # Collaborators [public]	Unlimited	Unlimited
Max # Collaborators [private]	Unlimited	5



Comparing GitLab with GitHub

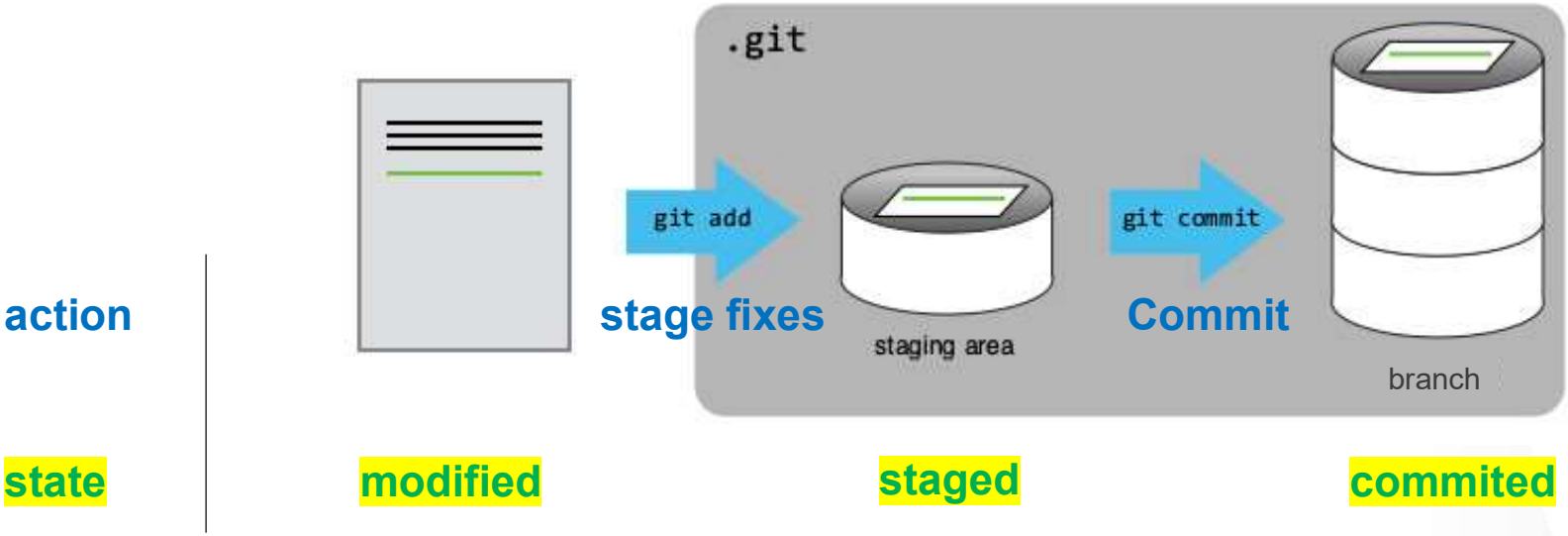
Git workflow



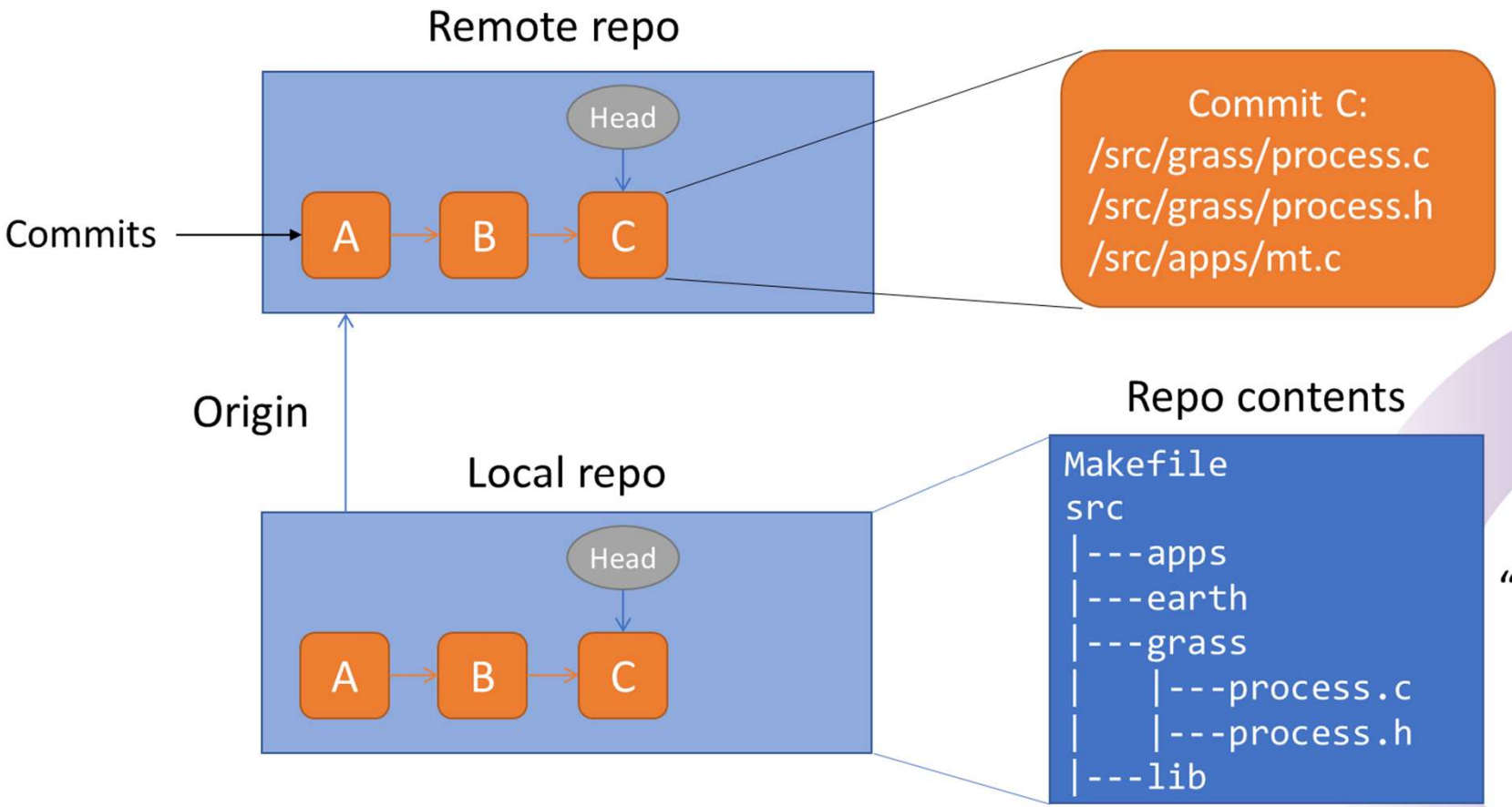
The three states of Git

Modified, staged, and committed

- **modified:** files are changed but have not been committed it to your database yet
- **staged:** modified file in its current version is marked to go into your next commit
- **committed:** data is safely stored in your local database



The Git model



Knowledge Check Poll

What sort of version control system is Git?

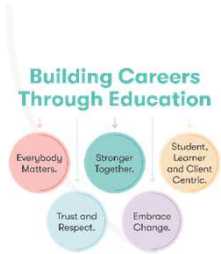
a) Centralised

b) Distributed

c) Check-summed

d) Snapshot-based

e) Free and open-source



Submit your responses to the chat!

BPP

Knowledge Check Poll

What sort of version control system is Git?

a) Centralised

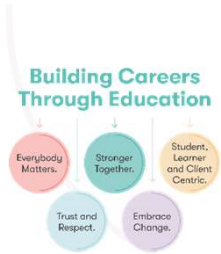
b) Distributed

c) Check-summed

d) Snapshot-based

e) Free and open-source

Feedback: answers b and c are correct. Git has distributed access, check-summed data integrity, snapshot-based change tracking and does not cost you any money to use (apart from infrastructure costs).



Submit your responses to the chat!

BPP

Knowledge Check Poll

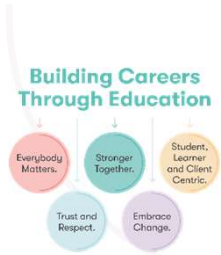
How does Git contribute to DevOps practices?

a) By requiring a central server for all operations

b) By eliminating the need for pull requests as it manages changes

c) By automating all the tests

d) By adding version control, snapshotting and integrity checks to continuous integration and continuous deployment



Submit your responses to the chat!

BPP

Knowledge Check Poll

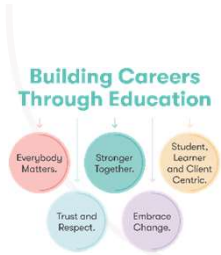
How does Git contribute to DevOps practices?

a) By requiring a central server for all operations

b) By eliminating the need for pull requests as it manages changes

c) By automating all the tests

d) By adding version control, snapshotting and integrity checks to continuous integration and continuous deployment



Feedback: d - because Git provides a robust system for tracking changes allowing teams to trust the code that has been added to the project

Submit your responses to the chat!

BPP

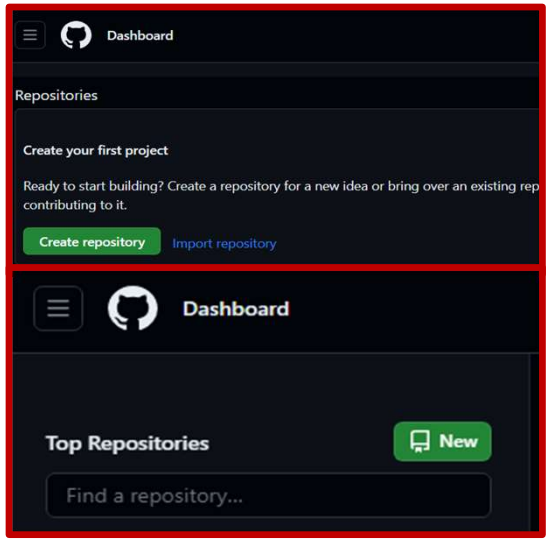
Practical demonstration 1

Working with repositories

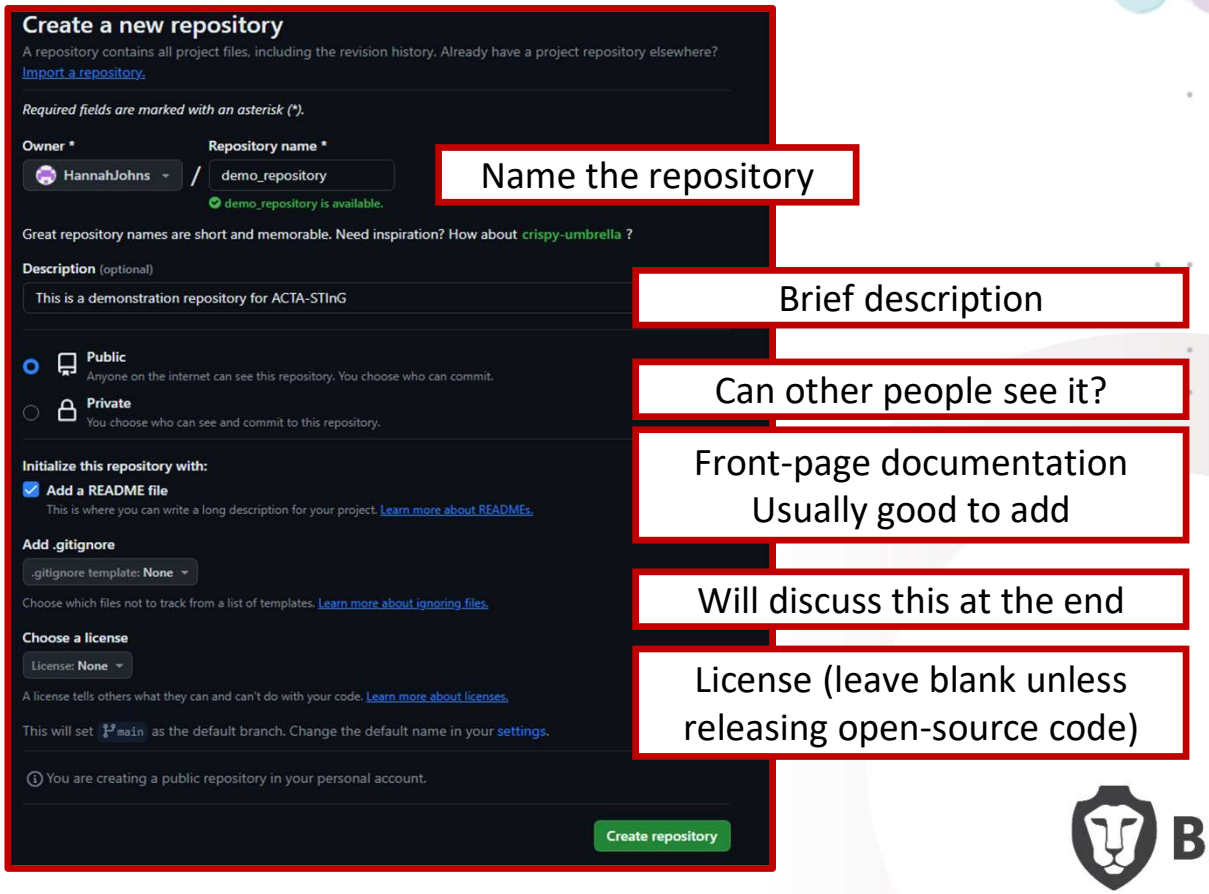
BPP

Working with repositories

A step-by-step guide



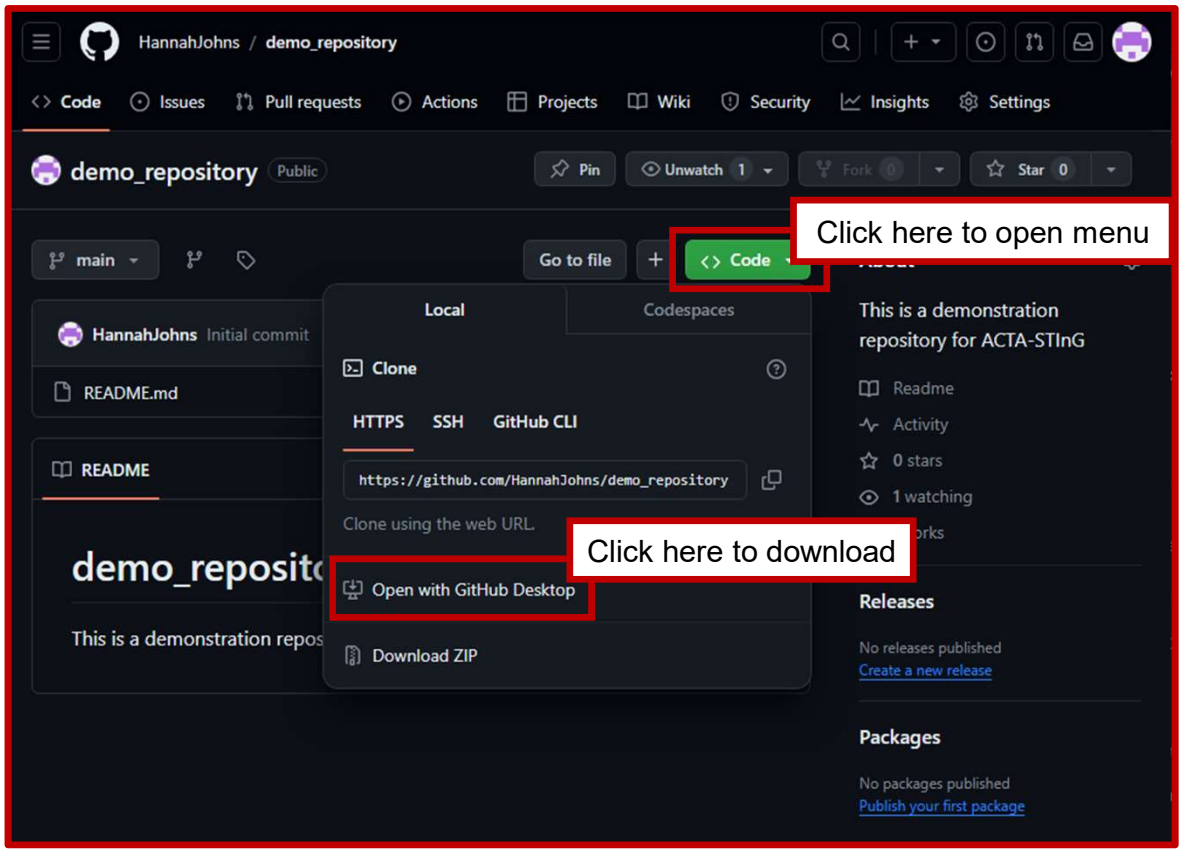
Step 1: *Click "Create repository" or "New"*



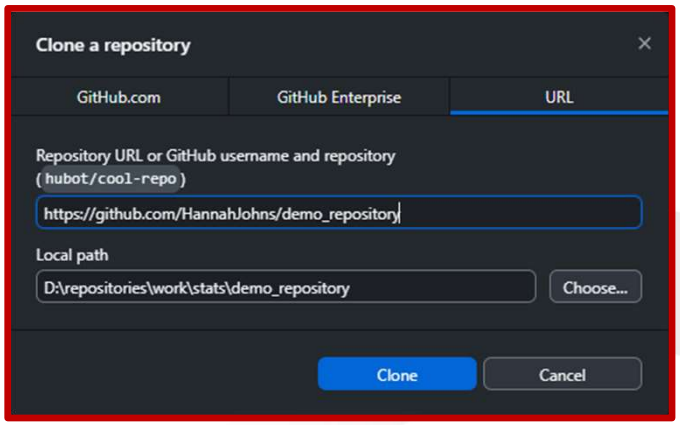
Step 2: *Complete the wizard*

Working with repositories

A step-by-step guide



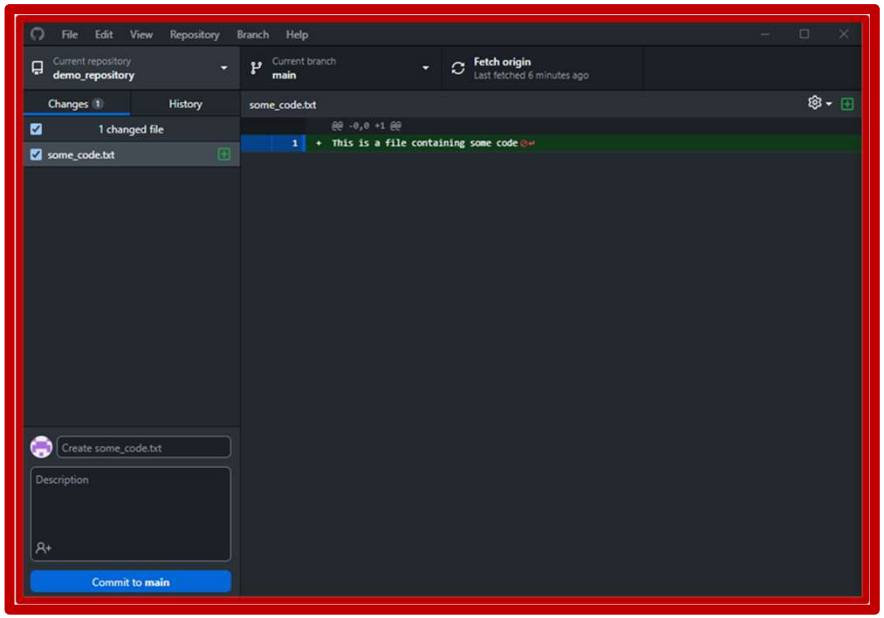
Step 3: *Clone (download) the repository*

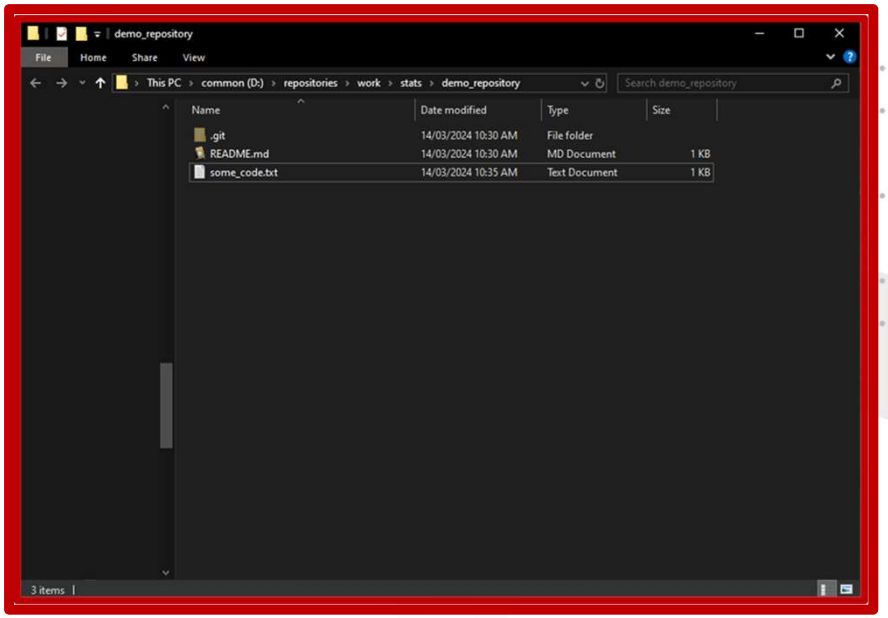


Choose where you want the repository to be located then click “Clone”

Working with repositories

A step-by-step guide



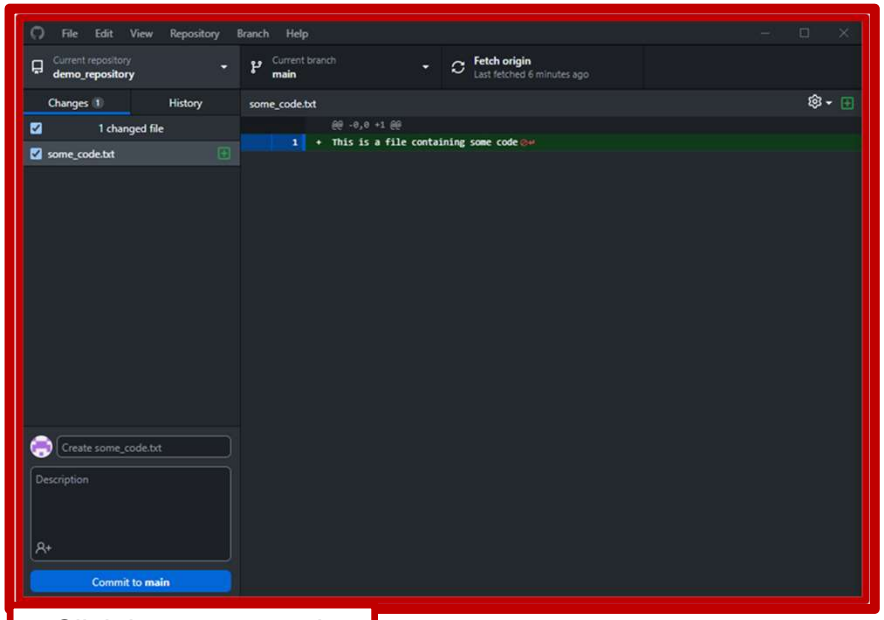


Step 5: *Create a file called some_code.txt*

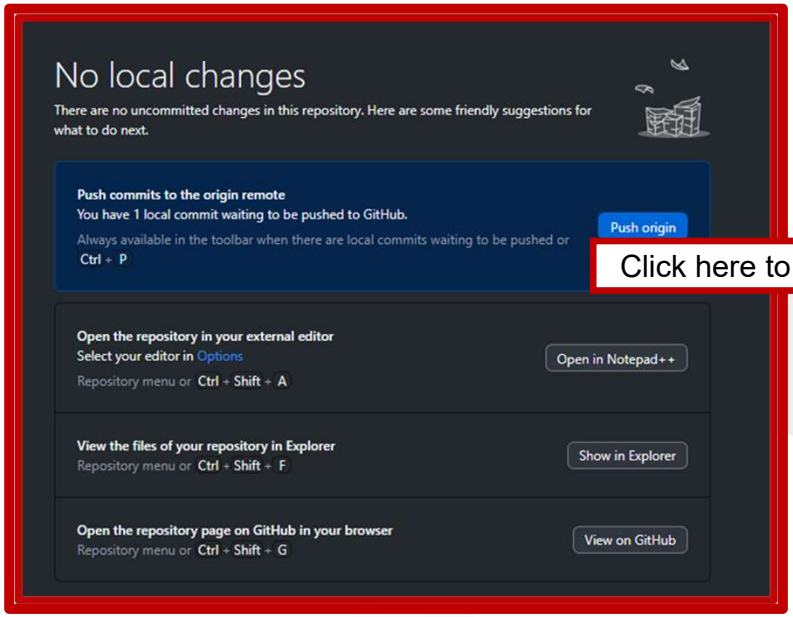
(GitHub desktop sees this challenge)

Working with repositories

A step-by-step guide



Click here to commit



Click here to upload

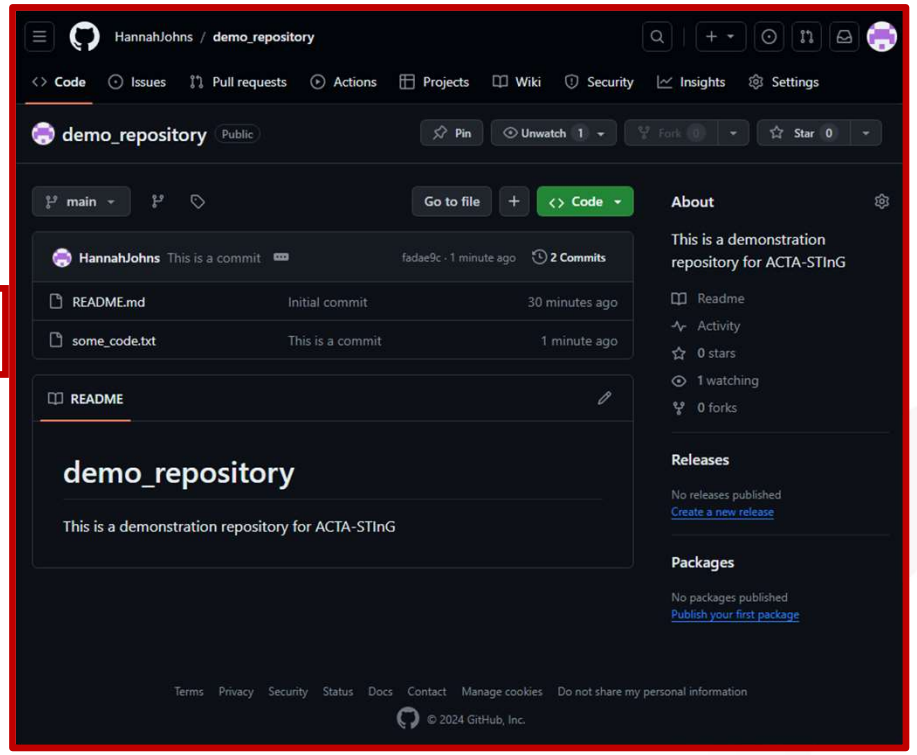
Step 6: *Commit this change and push it to GitHub*

- *'Committing' code means you are creating a snapshot of the codebase at this point in time*
- *'Pushing' code uploads these snapshots to GitHub*

Working with repositories

A step-by-step guide

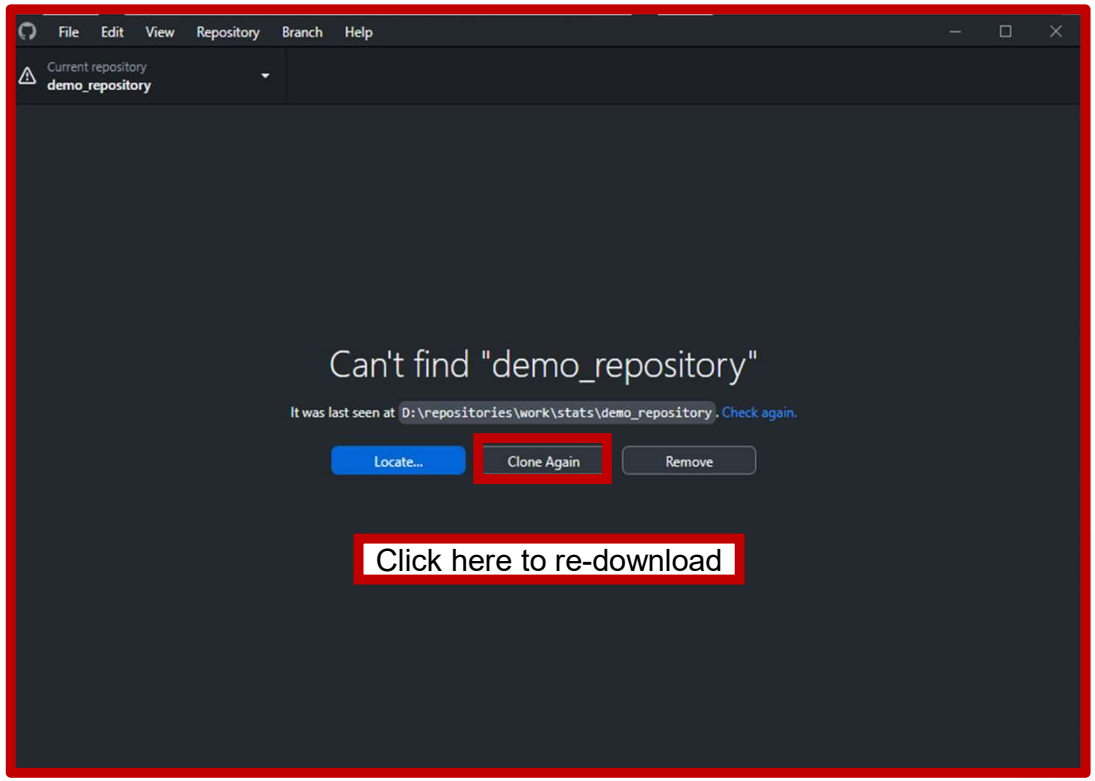
Changes have been uploaded to the github repository



Step 6 (cont): *Commit this change and push it to GitHub*

Working with repositories

A step-by-step guide

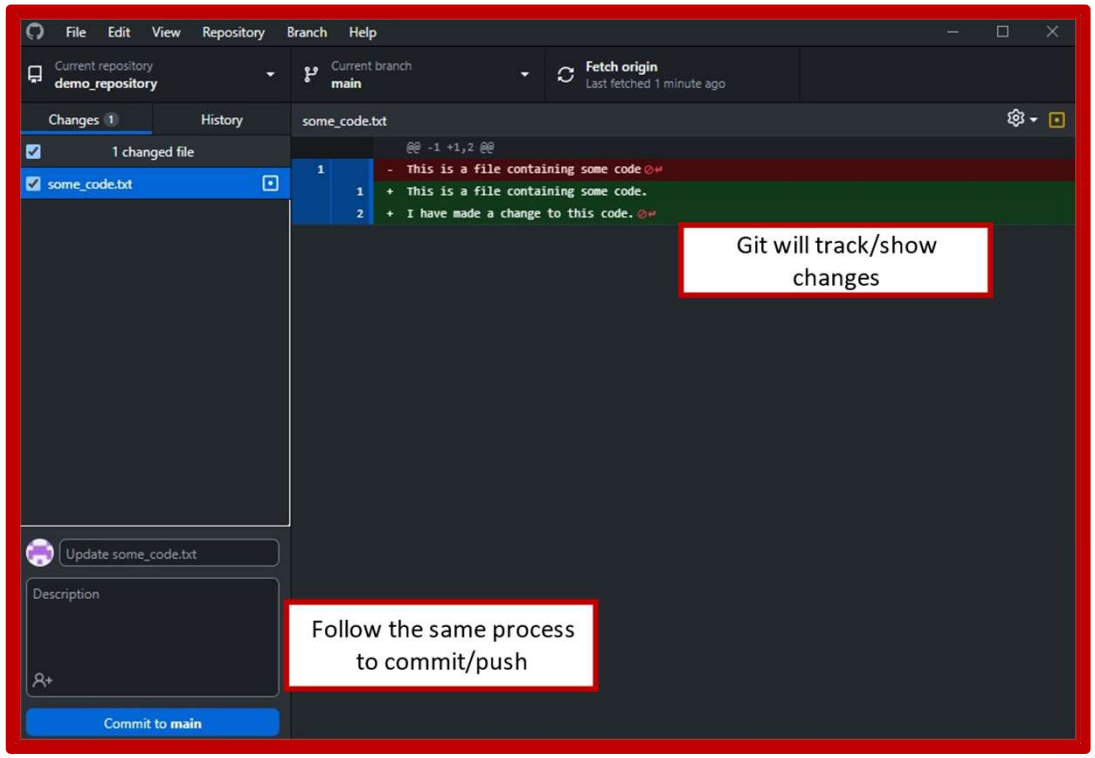


Step 7: *Delete the folder containing the repository and redownload it*

- You can recover any code saved on github by re-downloading the repository
- This ONLY works if the code has been pushed

Working with repositories

A step-by-step guide

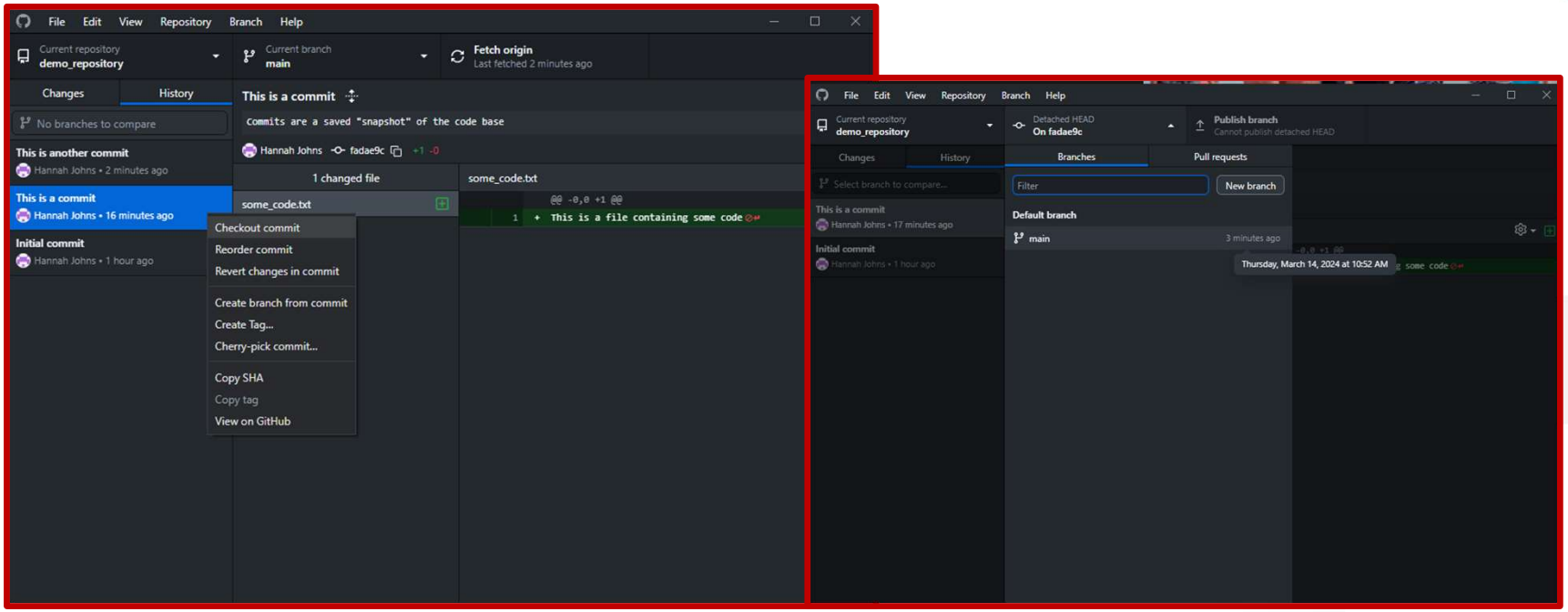


Step 8: *Make a change to some_code.txt*

- *Use the same process to commit/push these changes*

Working with repositories

A step-by-step guide

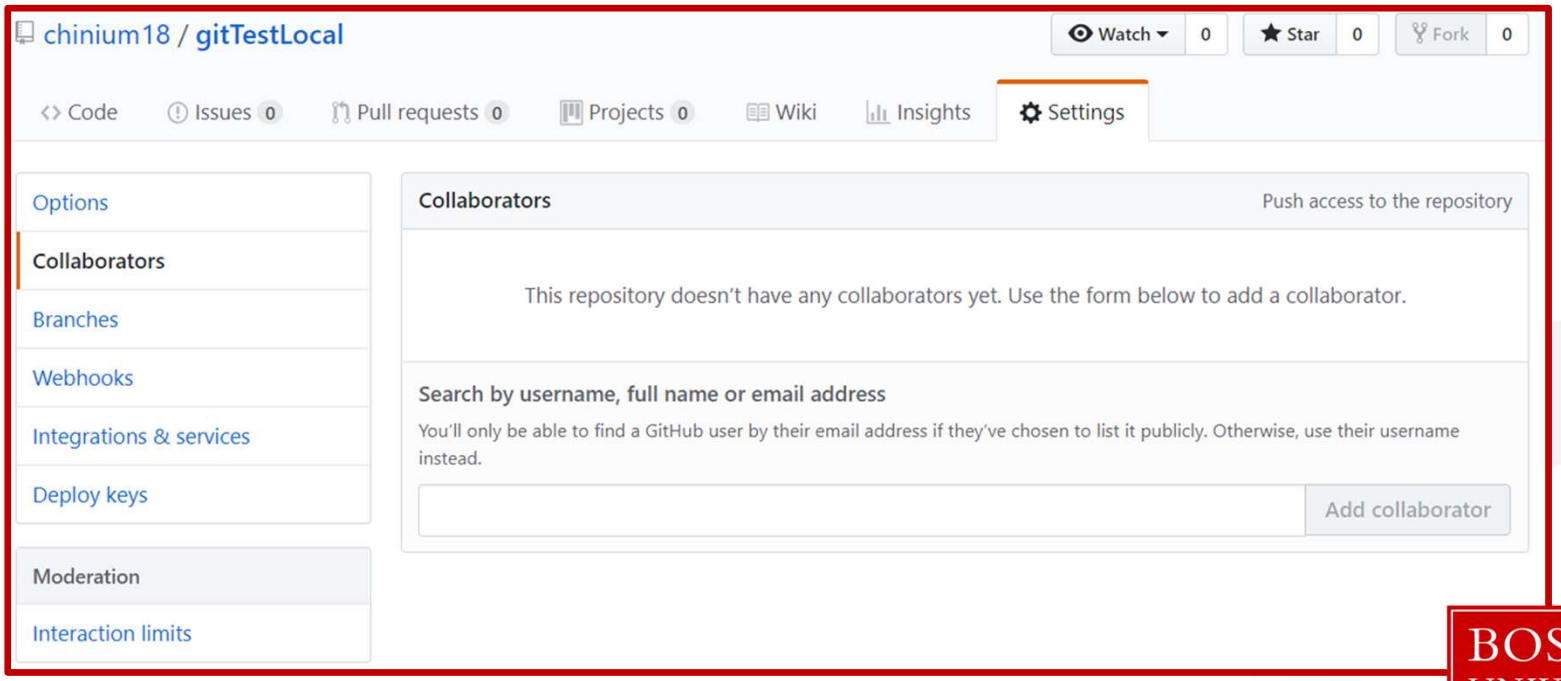


Step 9: *Revisit what the code base looked like previously*

- *You can revisit old commits at any point using "checkout"*
- *The project folder's code will be updated to match the state it was in at this commit*
- *To return to the current code state, select "main" from the "current branch" menu*

Working with repositories

Further guidance



Adding a collaborator

Building Careers Through Education

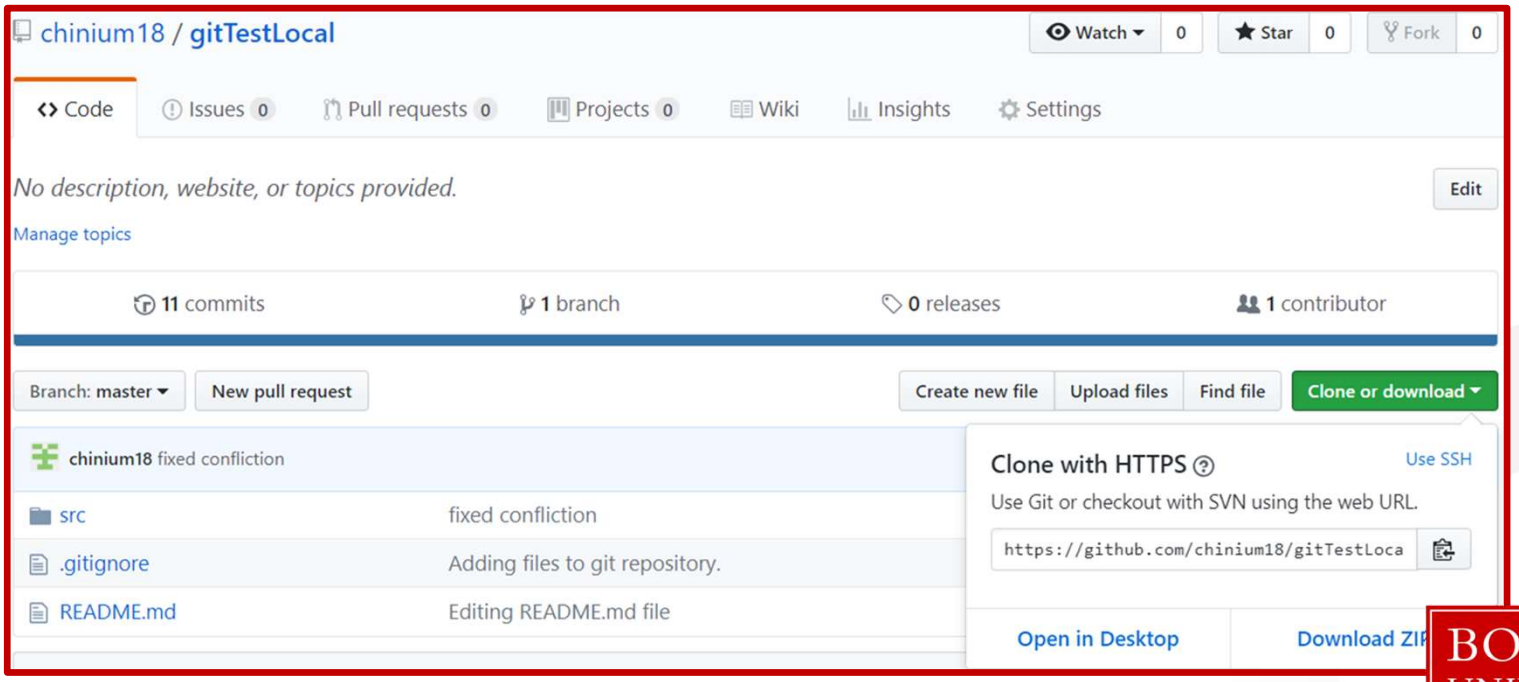
Everybody Matters. Stronger Together. Student, Learner and Client Centric. Trust and Respect. Embrace Change.

BOSTON UNIVERSITY

BPP

Working with repositories

Further guidance



Now your collaborator with be able to access your repository

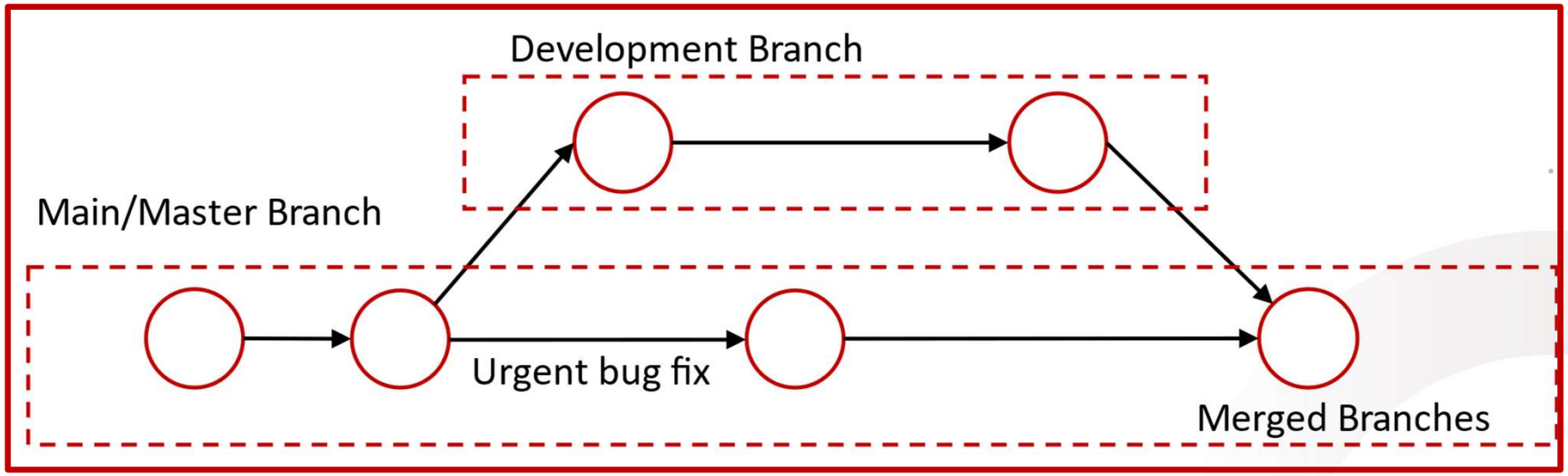
Working with repositories

General tips

- **NEVER** place a git repository in e.g. OneDrive
- **NEVER** place a git repository inside another git repository
- Don’t get lazy with commit messages!
- For code that other people will use, use Semantic Versioning for version numbers - Major.Minor.Patch
    - E.g. “Version 1.12.103:
    - **Major** – changes that break things that others have done (e.g. a complete rewrite)
    - **Minor** – changes that don’t break anything (e.g. feature additions)
    - **Patch** – bug fixes, etc

Working with repositories

What about working on larger projects?

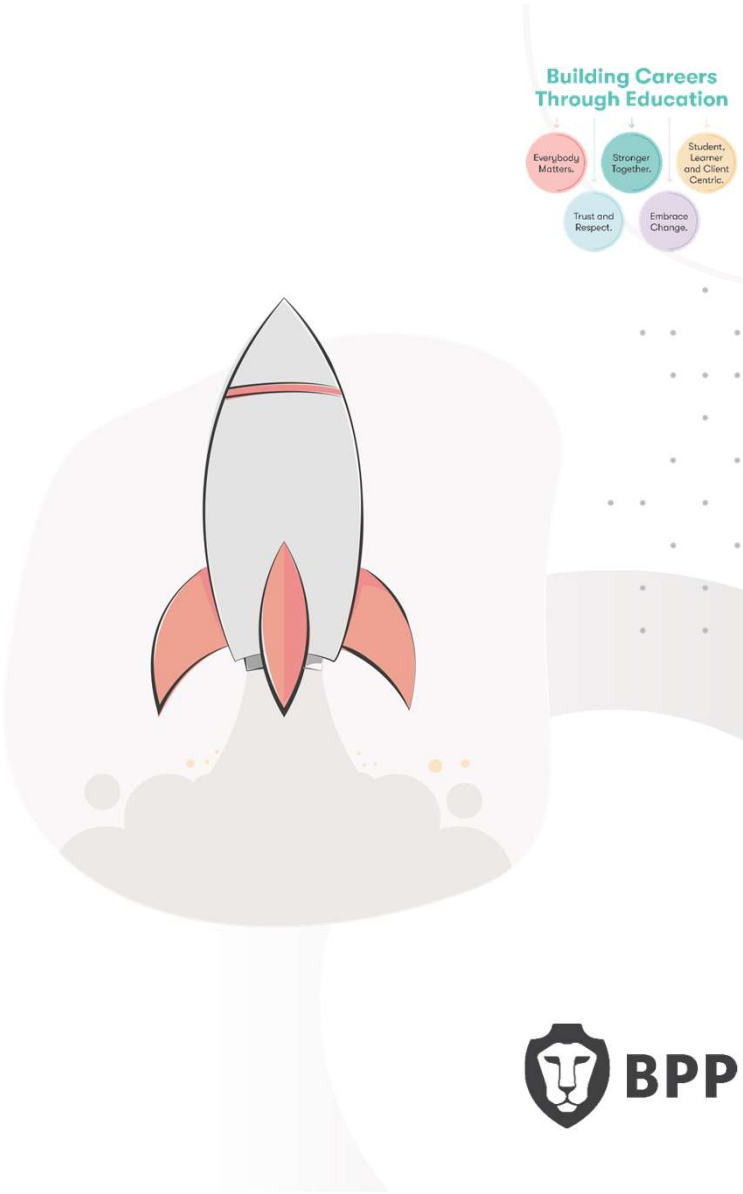


Use branches!

Working with repositories

Pro tips

- Use branching in GitHub for group projects
- Branches are deviations in the commit tree
- Branches are useful for feature management
- Be aware of merge conflicts



Resource:
https://www.atlassian.com/git/tutorials/comparing-workflows

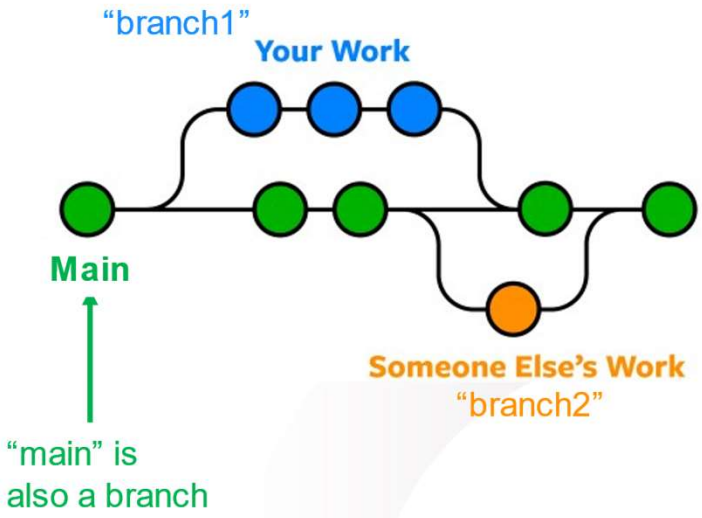
Git branching, changes and merges

BPP

Branching

What is a branch?

- **git branch -** lists all the branches in your repo and show your current branch
- **git checkout –b <branchname>** - creates a new branch and switch to it
- **git checkout <branchname>** - switch from one branch to another
- **git merge <branchname1> <branchname2>** - adds a file to the staging area

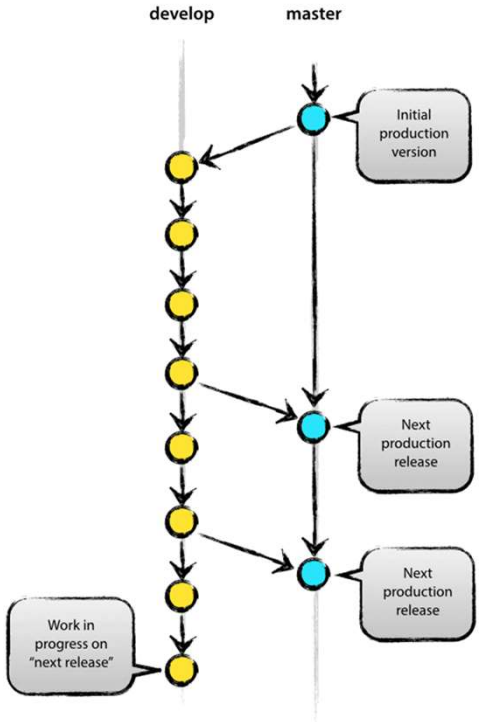


A simple branch

Branching

A Git branching model

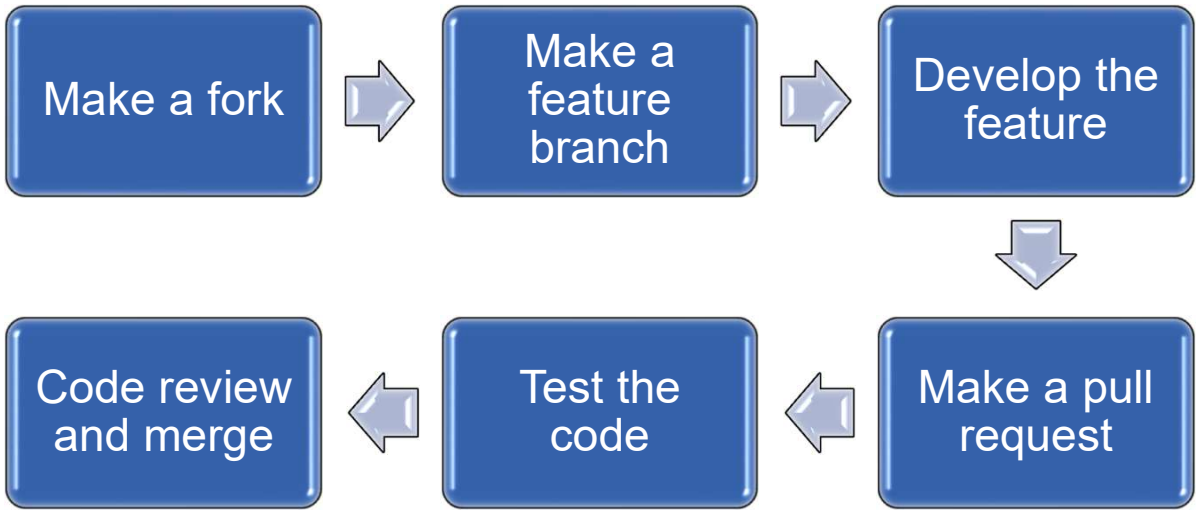
- Concept to make changes, without affecting the master-version
- Multiple branches can co-exist in parallel
- Used to implement new features
- Can be merged into each other

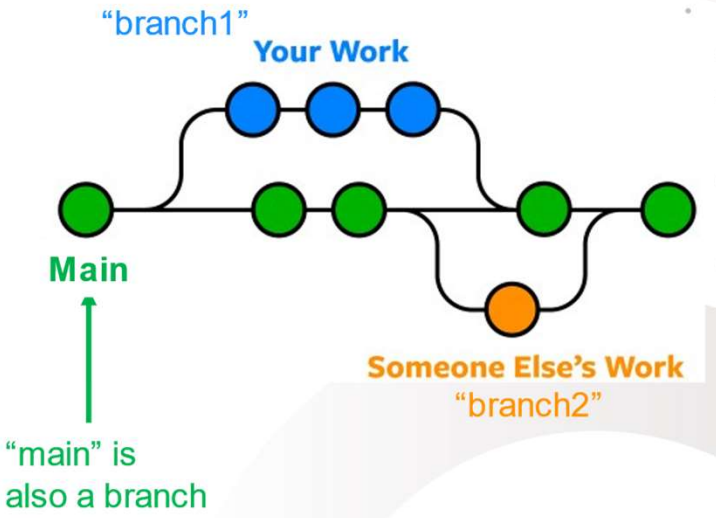


An illustration of a Git branching model

Branching

Branching workflow summary





A simple branch

Pro Tip: .gitignore files are present in the root directory of your repository.

Resource: https://help.github.com/articles/ignoring-files/

Git functionalities

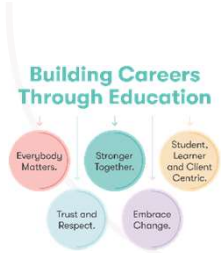
Group practice

Your tutor will navigate with you to the following site:

https://learngitbranching.js.org/

We will try the following git functionalities:

1. Git commit
2. Git branch
3. Git checkout
4. Git merge

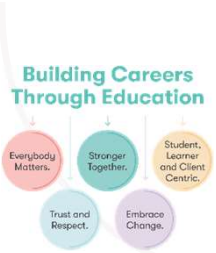


Group practice

BPP

Git status after a commit

A guide




```
~/egos$ git status
On branch master
Your branch is 1 commit behind 'origin/master'

Changes not staged for commit:
  (use "git add <file>…" to update what will be committed)

        modified:    src/grass/process.c

Untracked files:
  (use "git add <file>…" to include in what will be committed)

        src/apps/tags
        src/grass/tags
```

You made a commit, but haven't pushed

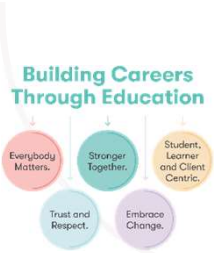
After your last commit, you continued editing this file

These files still haven't been added to any commit

BPP

Diff

What am I committing?




```
~/egos$ git status
On branch master
Your branch is 1 commit behind 'origin/master'

Changes not staged for commit:
  (use "git add <file>…" to update what will be committed)

        modified:   src/grass/process.c

Untracked files:
  (use "git add <file>…" to include in what will be committed)

        src/apps/tags
        src/grass/tags
```

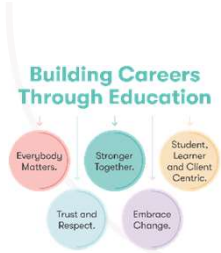
You made a commit, but haven't pushed

After your last commit, you continued editing this file

These files still haven't been added to any commit

BPP

Diff details




```
~/egos$ git diff src/grass/process.c
```

- Shows differences only for that file

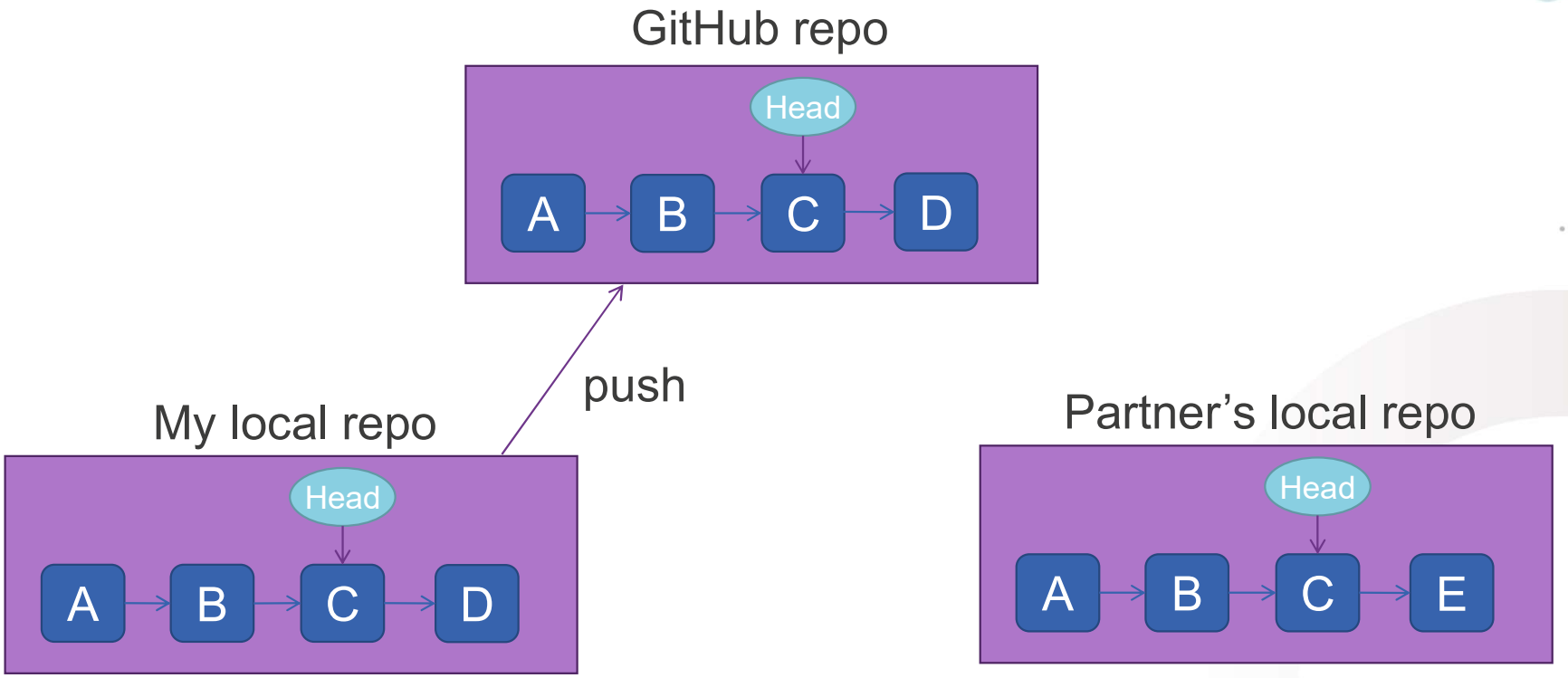
```
~/egos$ git diff
~/egos$
```

- Why does it give no results? I know I made changes!
- Answer: you have already git added your changes

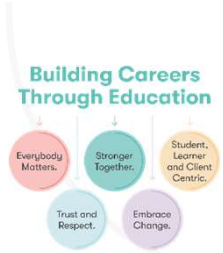
```
~/egos$ git diff --staged
diff --git a/src/lib/queue.c b/src/lib/queue.c
index c638853..19106c3 100644
--- a/src/lib/queue.c
+++ b/src/lib/queue.c
@@ -19,7 +19,7 @@ struct element {
```

BPP

Concurrent changes



Possible outcomes

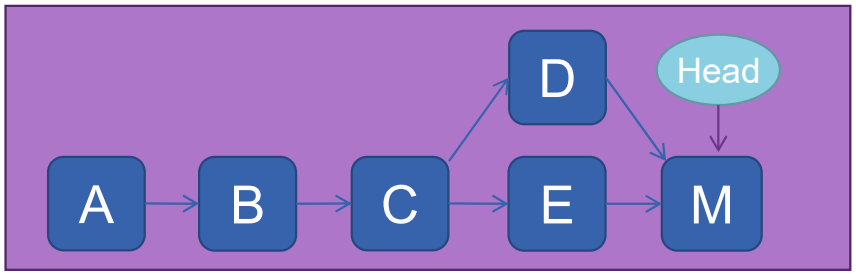


- No conflicts, just merge

```
~/egos$ git pull
# Editor pops up
Merge made by the 'recursive' strategy
  src/lib/queue.c
  1 file changed,
```

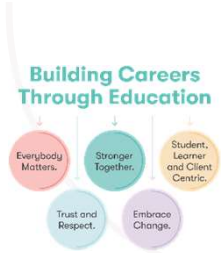
```
Merge branch 'master' of https://github.coecis.cornell.edu/etremel/egos.git

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit
```

  - Partner's local repo



BPP

Possible outcomes



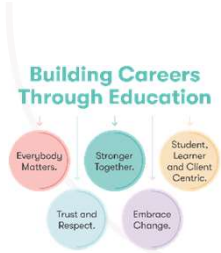
- Conflicting changes to the same file(s)

```
~/egos$ git pull
Auto-merging src/lib/queue.c
CONFLICT (content): Merge conflict in src/lib/queue.c
Automatic merge failed; fix conflicts and then commit the result
~/egos$ git status
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 1 different commits each, respectively.
…
Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   src/lib/queue.c
```

BPP

Finishing the merge

Edit the file, choose a single file and resolve conflicts



- Edit the file to choose a single version of the conflicting lines
- Make sure to delete the <<<<<<< and ======= lines!
- When you have resolved the conflict:

```
~/egos$ git add src/lib/queue.c
~/egos$ git commit
# Write a message for the merge commit
~/egos$ git push
```

BPP

Git functionalities

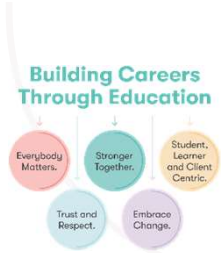
Group practice

Your tutor will navigate with you to the following site:

https://learngitbranching.js.org/

We will try the following git functionalities:

1. Branches and Merging
2. HEAD
3. Reversing changes



Group practice

BPP

Thank you

Do you have any questions, comments, or feedback?