

Introduction to GIT Version Control

James Willis (SciNet)

November 20, 2023

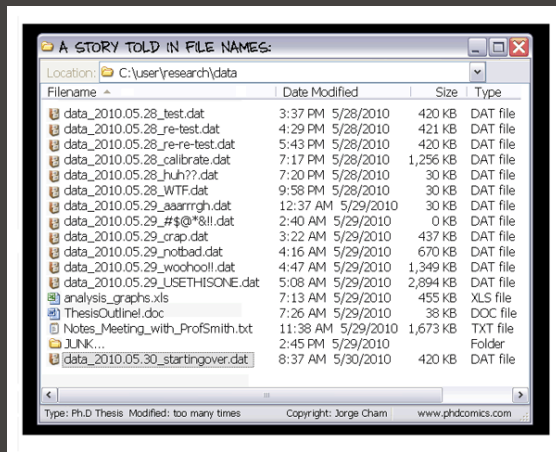
- Why use version control?
- About GIT version control
- GIT commands
 - Hands-on
- Web-based GIT Repos
- GitHub
 - Hands-on

Section 1

Version Control

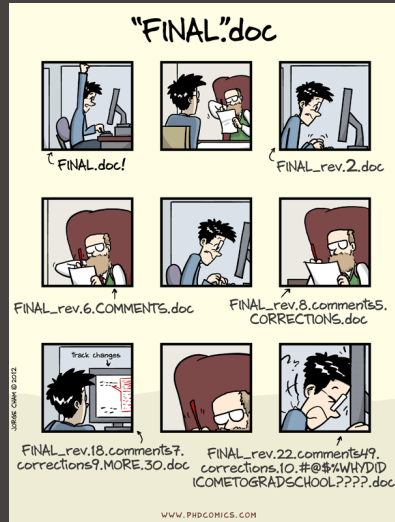
What is version control?

- Version control is a tool (i.e. a program) for managing changes in a set of files
- It allows you to keep a history of file versions and make them easy to track
- Essentially it takes a “snapshot” of the files in a given moment in time
- Can you think of any examples where you may have used/experienced something similar...?



Why use version control?

- Makes collaboration easier
- Helps you stay organised
- Allows you to keep track of changes without keeping duplicated copies of the same file
- Allows reproducibility
- When something goes wrong, you can back up to the last “working” copy
- It can be used for writing code, writing papers, it is especially powerful for text-based documents
- It is considered a **must** in professional software development



You may be familiar with the main features of Version Control already:

- Google Docs/Sheets/Slides
- Overleaf
- Dropbox
- Microsoft Word

These are **not** really Version Control though!



Section 2

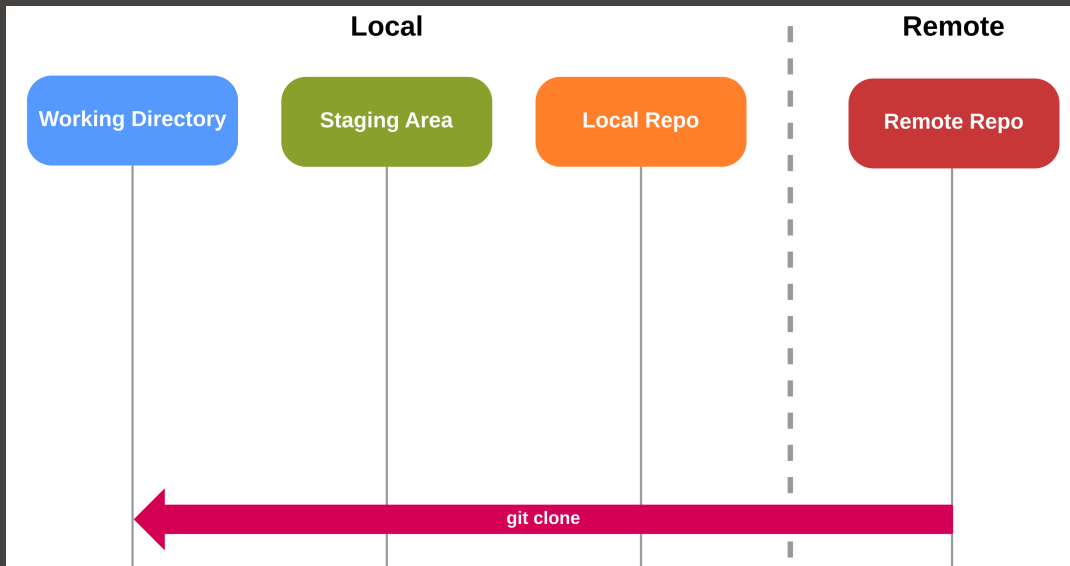
GIT

- Created by *Linus Towalds* in 2005
- What does **GIT** stand for? (<https://en.wikipedia.org/wiki/Git#naming>)
- There are many types and approaches to version control
- GIT is just one implementation, but it has taken over as the most popular and is used all over the world
- Other implementations include: CVS, SVN, Mercurial, etc. . .
- Some IDEs incorporate VC systems in their GUIs (e.g. Rstudio, Visual Studio, etc. . .)
- And of course, as we will discuss later, there are web-based repositories that allow you to use VC/GIT from within a browser

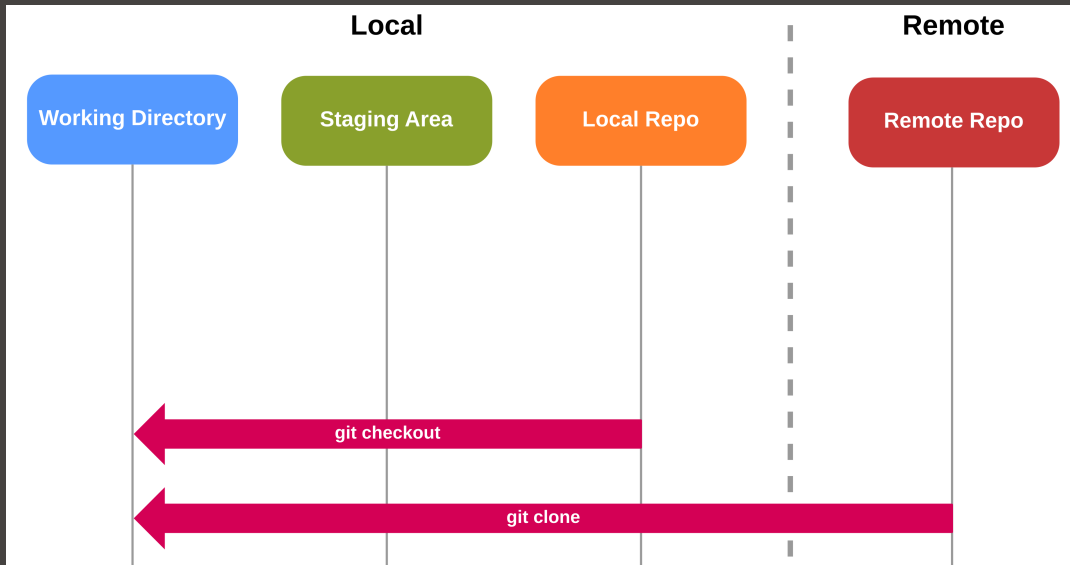
How does GIT work?



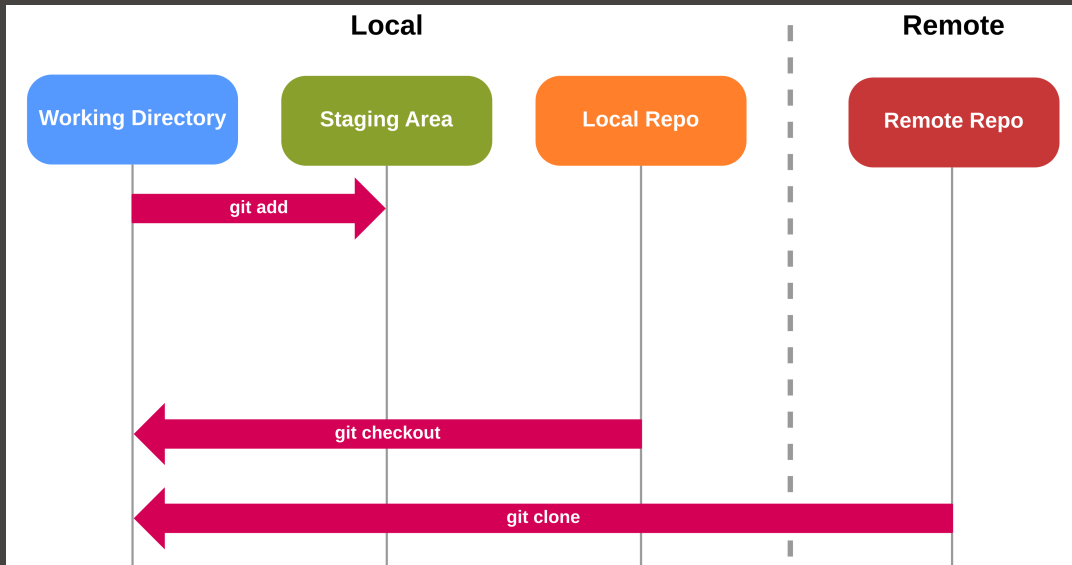
How does GIT work?



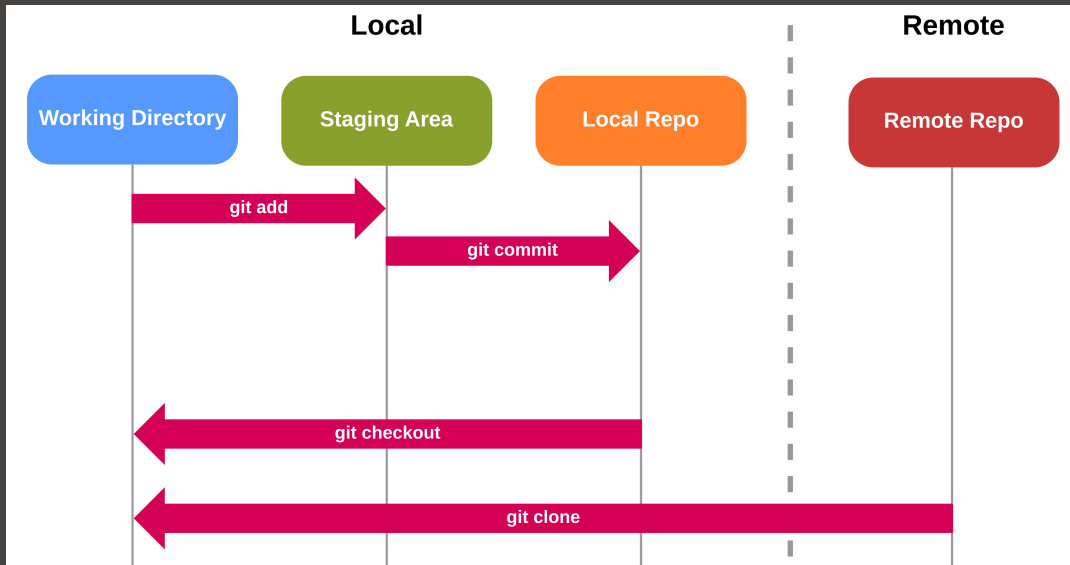
How does GIT work?



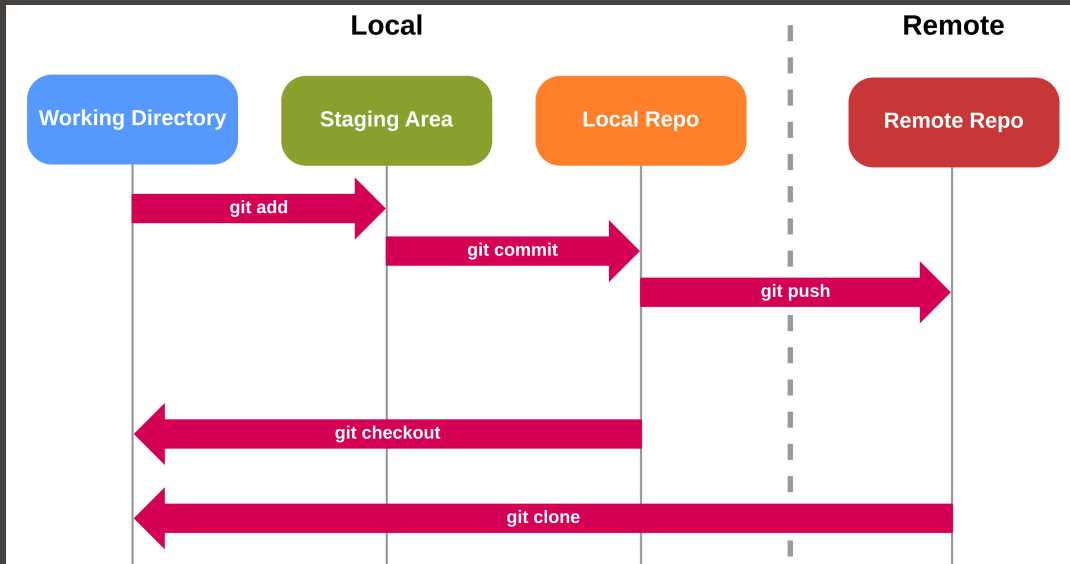
How does GIT work?



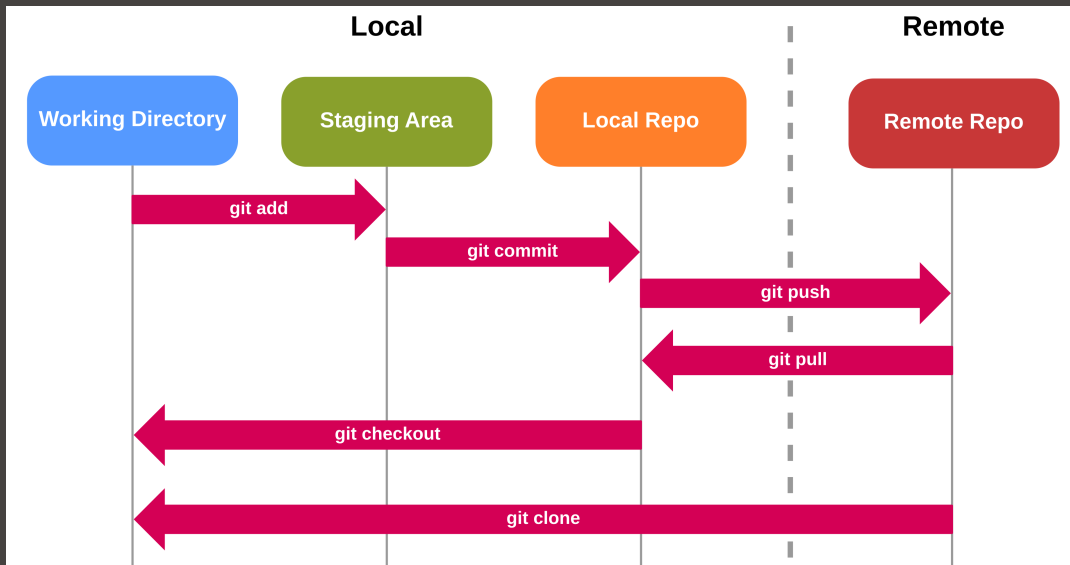
How does GIT work?

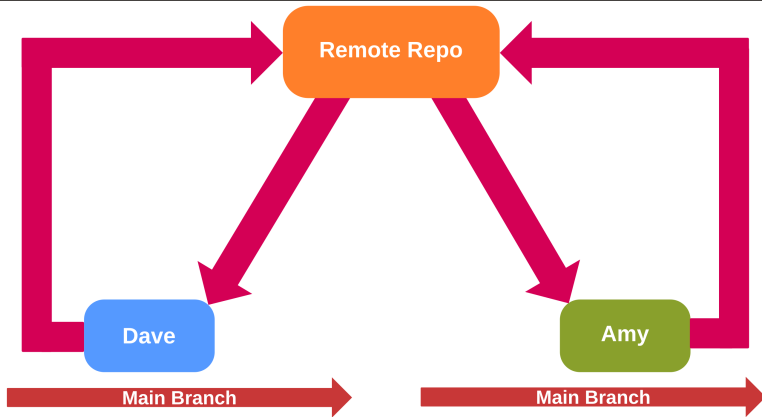


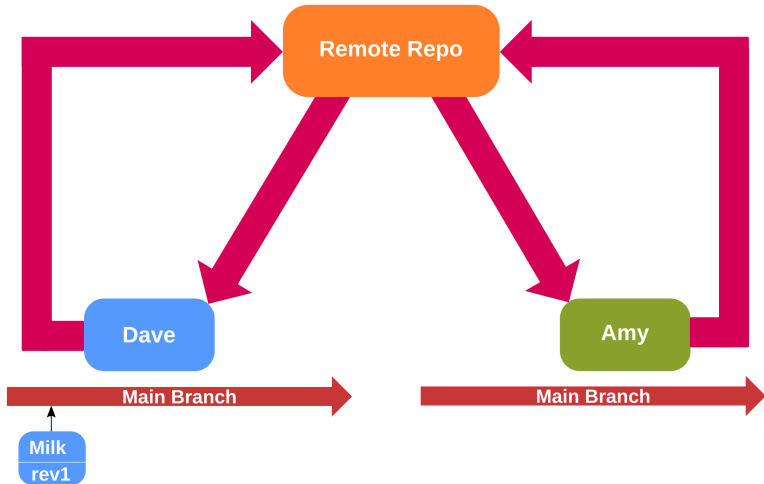
How does GIT work?

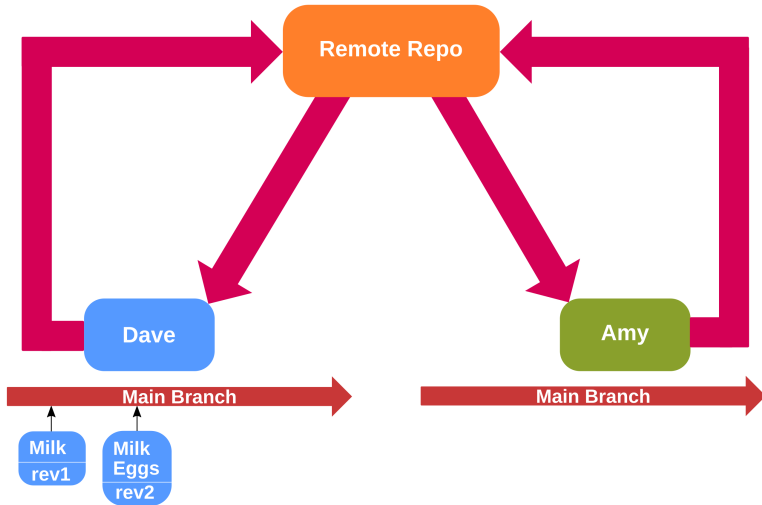


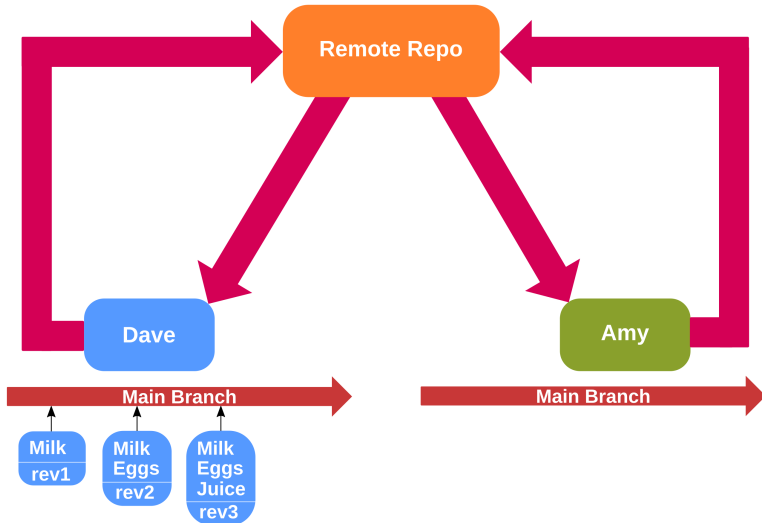
How does GIT work?

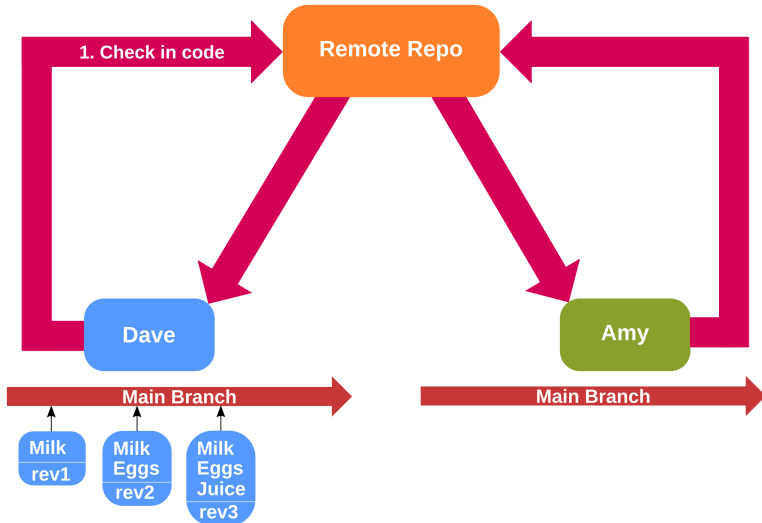


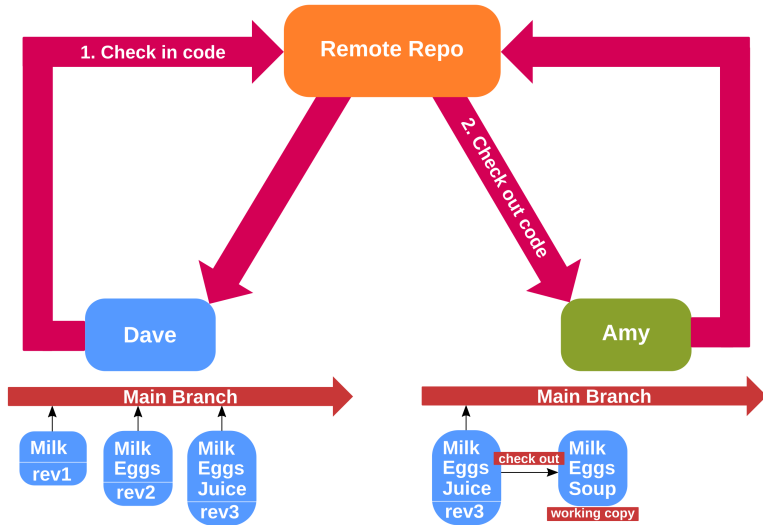


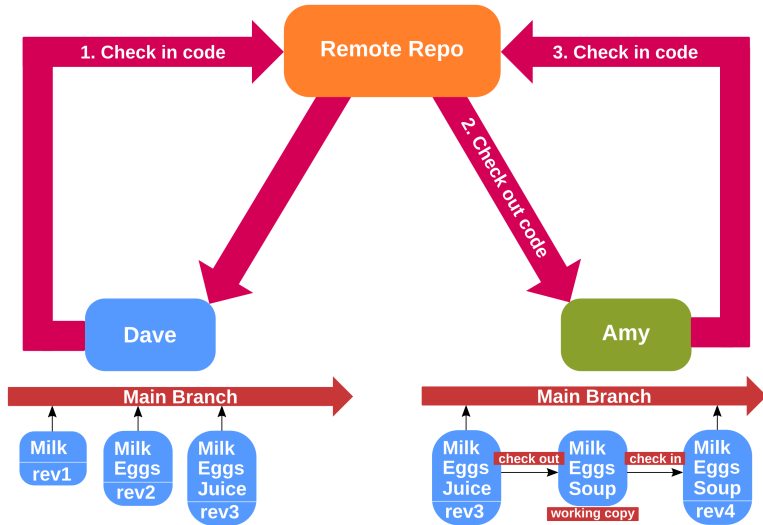


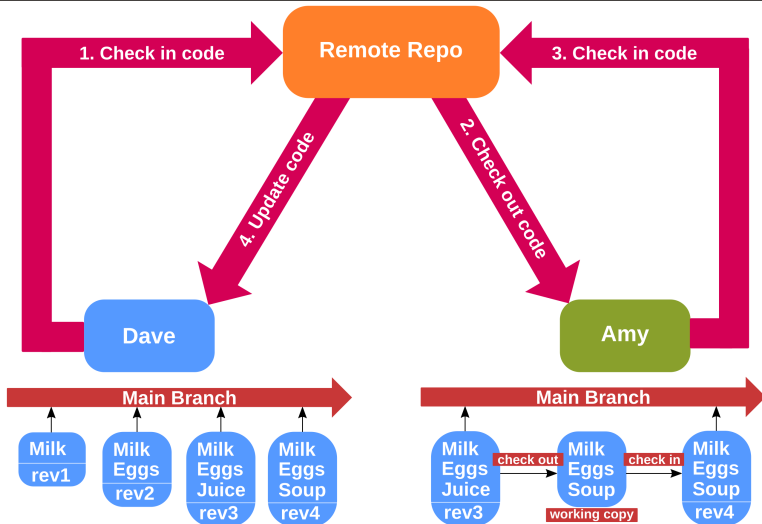












- *Repository*: “A collection of refs together with an object database containing all objects which are reachable from the refs.”
- *Commit*: “A single point in the Git history.”
- *Checkout*: “The action of updating all or part of the working tree with a tree object or blob from the object database.”
- *Branch*: “A branch is used to develop a feature that is merged into the *master* branch upon completion.”
- *Conflict*: “When two branches are merged and one branch overwrites changes from the other. All *conflicts* need to be resolved before completing the merge”.

- Step 0: Setup GIT on your computer
- Step 1: Initialise a GIT repo
- Step 2: Commit files to the repo
- Step 3: Edit/Modify/Add new or existing files
- Step 4: Commit changes
- Step 5: Push changes to remote repo
- Step 6: Repeat from Step 2

- Install GIT on Linux:

```
sudo apt install git-all
```

- Install GIT on MacOS:

```
git --version
```

(It should prompt you to install if it doesn't already exist)

- Install GIT on Windows by downloading packages from here: <https://git-scm.com/download/win>
- More information can be found here:
<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

Section 3

GIT commands

- Find a location for your repo and initialise it:

```
laptop:~$ mkdir my-repo
laptop:~$ cd my-repo
laptop:~/my-repo$ git init
Initialized empty Git repository in /home/willlis/my-repo/.git/
```

- This creates a `.git` repo in the `my-repo` directory, which contains the repo information:

```
laptop:~/my-repo$ ls -a
.  ..  .git
```

Note: The `-a` option for `ls` shows **all** files, which includes *hidden* files that start with `.`

- The first time you try to use git to commit something, it might complain that cannot identify you:

```
*** Please tell me who you are.  
Run  
git config --global user.email "youremail@example.com"  
git config --global user.name "FirstName LastName"  
To set your account's default identity.  
Omit --global to set the identity only in this repository.  
fatal: empty indent name (for <(null)>) not allowed
```

- You can also check in advance using:

```
laptop:~$ git config user.name  
laptop:~$ git config user.email
```

Adding files to a repository, requires two steps:

- Step 1: Add files to the *staging* area:

```
laptop:~/my-repo$ echo "some data" > datafile.dat
laptop:~/my-repo$ cp datafile.dat replicated_data.dat
laptop:~/my-repo$ ls
datafile.dat  replicated_data.dat
laptop:~/my-repo$ git add datafile.dat replicated_data.dat
```

- Step 2: Commit files to the repo:

```
laptop:~/my-repo$ git commit datafile.dat replicated_data.dat -m "Adding data from
experiment X."
[master (root-commit) d67dfb5] Adding data from experiment X.
2 files changed, 2 insertions(+)
create mode 100644 datafile.dat
create mode 100644 replicated_data.dat
```

- Suppose we have to update some data and we would like to compare it with the files already in the repo:

```
laptop:~/my-repo$ echo "updated data" >> datafile.dat
laptop:~/my-repo$ git diff datafile.dat
diff --git a/datafile.dat b/datafile.dat
index 4268632..db1d6b5 100644
--- a/datafile.dat
+++ b/datafile.dat
@@ -1,2 @@
  some data
+updated data
laptop:~/my-repo$ git commit datafile.dat -m "Updating data from new experiments."
```

- Look at the history of the repo:

```
laptop:~/my-repo$ git log
commit 5afbe1a660ba831026542e2df9474213eb42237f (HEAD -> master)
Author: willis <james.willis@scinet.utoronto.ca>
Date:   Wed Mar 2 14:22:09 2022 -0500
```

```
    Updating data from new experiments.
```

```
commit d67dfb567d6d9d92a3a4e0aac1924ab10dda3a61
Author: willis <james.willis@scinet.utoronto.ca>
Date:   Wed Mar 2 12:56:50 2022 -0500
```

```
    Adding data from experiment X.
```

- Recover a specific version:

```
laptop:~/my-repo$ git checkout d67dfb56
```


- Delete file:

```
laptop:~/my-repo$ git rm replicated_data.dat
laptop:~/my-repo$ git commit -m "Removed replicated data."
[master 8ee5a01] Removed replicated data.
1 file changed, 1 deletion(-)
delete mode 100644 replicated_data.dat
```

- Note: when you delete a file from the repo like this, it is also deleted from your computer. To remove it from the repo only use the `--cached` option:

```
laptop:~/my-repo$ git rm --cached replicated_data.dat
```

Command	Scope	Common use cases
<code>git reset</code>	Commit-level	Discard commits in a private branch or uncommitted changes
<code>git reset</code>	File-level	Unstage a file
<code>git checkout</code>	Commit-level	Switch between branches or inspect old snapshots
<code>git checkout</code>	File-level	Discard changes in the working directory
<code>git revert</code>	Commit-level	Undo commits in a public branch
<code>git revert</code>	File-level	N/A

- Check the status of files in the local repo:

```
laptop:~/my-repo$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
   new file:   new_file.txt

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:  replicated_data.dat

Untracked files:
  (use "git add <file>..." to include in what will be committed)
   output.log
```

- Get a comprehensive list of git commands with `git --help`:

These are common Git commands used in various situations:

```
start a working area (see also: git help tutorial)
  clone          Clone a repository into a new directory
  init           Create an empty Git repository or reinitialize an existing one
work on the current change (see also: git help everyday)
  add           Add file contents to the index
  mv            Move or rename a file, a directory, or a symlink
  restore       Restore working tree files
  rm            Remove files from the working tree and from the index
  sparse-checkout Initialize and modify the sparse-checkout
examine the history and state (see also: git help revisions)
  bisect       Use binary search to find the commit that introduced a bug
  diff         Show changes between commits, commit and working tree, etc
  grep         Print lines matching a pattern
  log          Show commit logs
  ...
```

- git commands can be quite long to type repeatedly. They can be shortened with aliases
- For example, to shorten `git checkout` to `git co` run:

```
laptop:~$ git config --global alias.co checkout
```

- Useful aliases:

```
laptop:~$ git config --global alias.br branch  
laptop:~$ git config --global alias.ci commit  
laptop:~$ git config --global alias.st status  
laptop:~$ git config --global alias.d difftool
```

Note: all git configuration options can be found in your HOME directory in `~/.gitconfig`

- It may feel like more work in the short term, but USE IT! It will save you from future headaches
- Commit often!
- Include sensible commit messages
- Do not commit derivative files e.g. log files, executables, compiled modules
- It is useful for different kinds of projects: code development, collaborations, papers etc.
- There are different version control systems: GIT, HG, SVN, CVS

- Install GIT on your local machine
 - `sudo apt install git-all` (Linux)
 - `git --version` (MacOS - It should prompt you to install if it doesn't already exist)
- If that fails, GIT is also installed on Niagara
- Create a local repository
- Add some files
- Experiment with different GIT commands (`git --help` for full list)
- Hints:

```
laptop:~$ git init
laptop:~$ git add file.dat
laptop:~$ git commit file.dat -m "Commit message"
```

Section 4

Web-based GIT Repos

- GitHub: <https://github.com>
- BitBucket: <https://bitbucket.org>
- GitLab: <https://gitlab.com>



Section 5

GitHub

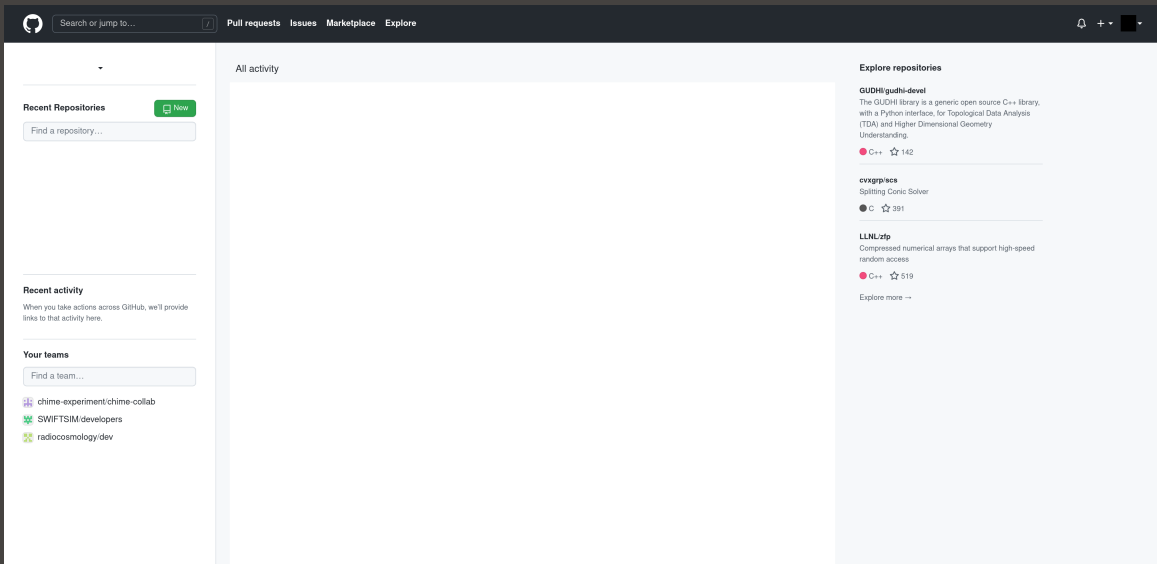
- What is GitHub and why use it?
- Integrating a local repository with GitHub

- Create a repository in GitHub
- *Push* a repository from your computer to GitHub
- *Pull* a repository from GitHub to your computer
- Create and accept a *pull request*

- Git and GitHub *are not the same thing*
- Hosted at github.com
 - Accounts are free
 - Ability to create private repositories
- Heavily used
- Collaborate
- Find code to adapt for you own projects
- Contribute to other code bases

- You have a **local** repository on your computer
- GitHub hosts **remote** repositories
- You can **push** from your local repository to a remote repository
- You can **pull** from a remote repository to a local repository
- You can make a **pull request**, in which you ask someone to **pull** your repository into theirs

GitHub: How to create a repo



The screenshot shows the GitHub homepage. At the top, there is a navigation bar with the GitHub logo, a search bar, and links for Pull requests, Issues, Marketplace, and Explore. The main content area is divided into three columns. The left column contains sections for 'Recent Repositories' with a search bar and a 'New' button, 'Recent activity' with a brief description, and 'Your teams' with a search bar and a list of teams. The middle column is titled 'All activity' and is currently empty. The right column is titled 'Explore repositories' and lists three repositories: GUDHI/gudhi-devel (C++ library), cvxgrp/scs (Conic Solver), and LLNL/afp (compressed numerical arrays).

GitHub: How to create a repo

The screenshot shows the GitHub homepage. At the top, there is a search bar with the text "Search or jump to..." and navigation links for "Pull requests", "Issues", "Marketplace", and "Explore". On the left sidebar, under "Recent Repositories", there is a search box "Find a repository..." and a green "New" button with a plus icon, which is highlighted with a red rectangular box. Below this are sections for "Recent activity" and "Your teams" with a search box "Find a team..." and a list of teams: "chime-experiment/chime-collab", "SWIFTSIM/developers", and "radiocosmology/dev". The main content area is titled "All activity" and is currently empty. On the right sidebar, under "Explore repositories", there are three repository cards: "GUDHI/gudhi-devel" (C++ library with 142 stars), "cvxgrp/scs" (Splitting Conic Solver with 391 stars), and "LLNL/afp" (Compressed numerical arrays with 519 stars).

[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)

Create a new repository

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

[Import a repository.](#)

Repository template

Start your repository with a template repository's contents.

No template ▾

Owner *

Repository name *

 /

Great repository names are short and memorable. Need inspiration? How about [musical-palm-tree?](#)

Description (optional)

 **Public**

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

 **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.](#)

Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository



Search or jump to...



[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Repository template

Start your repository with a template repository's contents.

No template ▾

Owner *

Repository name *

my-repo ✓

Great repository names are short and memorable. Need inspiration? How about [musical-palm-tree?](#)

Description (optional)

My GitHub repo



Public

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



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.](#)

Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Repository template

Start your repository with a template repository's contents.

No template ▾

Owner *

Repository name *

/ ✓

Great repository names are short and memorable. Need inspiration? How about [musical-palm-tree?](#)

Description (optional)

- Public**
Anyone on the internet can see this repository. You choose who can commit.
- 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.](#)
- Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

GitHub: How to create a repo

The screenshot shows the GitHub interface for a new repository named 'my-repo'. At the top, there is a search bar and navigation links for Pull requests, Issues, Marketplace, and Explore. The repository name 'my-repo' is shown as private, with options for Unwatch (1), Fork (0), and Star (0). Below the navigation bar, there are tabs for Code, Issues, Pull requests, Actions, Projects, Security, Insights, and Settings. The main content area is titled 'Quick setup — if you've done this kind of thing before' and provides three methods for creating a repository:

- Quick setup — if you've done this kind of thing before**: Shows a form with 'or' and 'HTTPS' selected. The URL is 'git@github.com: /my-repo.git'. Below the form, it says 'Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).
- ...or create a new repository on the command line**: Provides a terminal command sequence:

```
echo "# my-repo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com: /my-repo.git
git push -u origin main
```
- ...or push an existing repository from the command line**: Provides a terminal command sequence:

```
git remote add origin git@github.com: /my-repo.git
git branch -M main
git push -u origin main
```
- ...or import code from another repository**: States 'You can initialize this repository with code from a Subversion, Mercurial, or TFS project.' and includes an 'Import code' button.

At the bottom of the page, there is a 'ProTip!' icon and text: 'Use the URL for this page when adding GitHub as a remote.'

- Let the local repo know where to find the remote GitHub repo:

```
laptop:~$ cd my-repo/  
laptop:~/my-repo$ git remote add origin git@github.com:username/my-repo.git
```

- Let the local repo know where to find the remote GitHub repo:

```
laptop:~$ cd my-repo/  
laptop:~/my-repo$ git remote add origin git@github.com:username/my-repo.git
```

- Create the main branch and call it main:

```
laptop:~/my-repo$ git branch -M main
```

- Push your local branch to the remote (origin) GitHub repo:

```
laptop:~/my-repo$ git push -u origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 903 bytes | 903.00 KiB/s, done.
Total 9 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To github.com:username/my-repo.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

View repo on GitHub

The screenshot shows a GitHub repository page for a repository named "my-repo". The page is in a dark theme. At the top, there is a search bar and navigation links for Pull requests, Issues, Marketplace, and Explore. The repository name "my-repo" is displayed with a "Private" label. On the right side, there are buttons for Unwatch (1), Fork (0), and Star (0). Below the repository name, there are tabs for Code, Issues, Pull requests, Actions, Projects, Security, Insights, and Settings. The main content area shows the repository's file structure with a commit history table. The commit history table has columns for commit hash, commit message, and time ago. The most recent commit is "Revert 'Removed replicated data.'" with hash 98cd375, made 5 days ago, and has 4 commits. Below the commit history, there is a button to "Add a README" with a subtext "Add a README with an overview of your project." On the right side, there are sections for "About", "Releases", and "Packages". The "About" section shows "My GitHub repo" with 0 stars, 1 watching, and 0 forks. The "Releases" section shows "No releases published" and a link to "Create a new release". The "Packages" section shows "No packages published" and a link to "Publish your first package". At the bottom, there is a footer with the GitHub logo, copyright information "© 2022 GitHub, Inc.", and various links like Terms, Privacy, Security, Status, Docs, Contact GitHub, Pricing, API, Training, Blog, and About.

Search or jump to... / Pull requests Issues Marketplace Explore

/ my-repo Private Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

Commit Hash	Commit Message	Time Ago	Commits
98cd375	Revert "Removed replicated data."	5 days ago	4 commits
	datafile.dat	Updating data from new experiments.	5 days ago
	replicated_data.dat	Revert "Removed replicated data."	5 days ago

Add a README with an overview of your project. Add a README

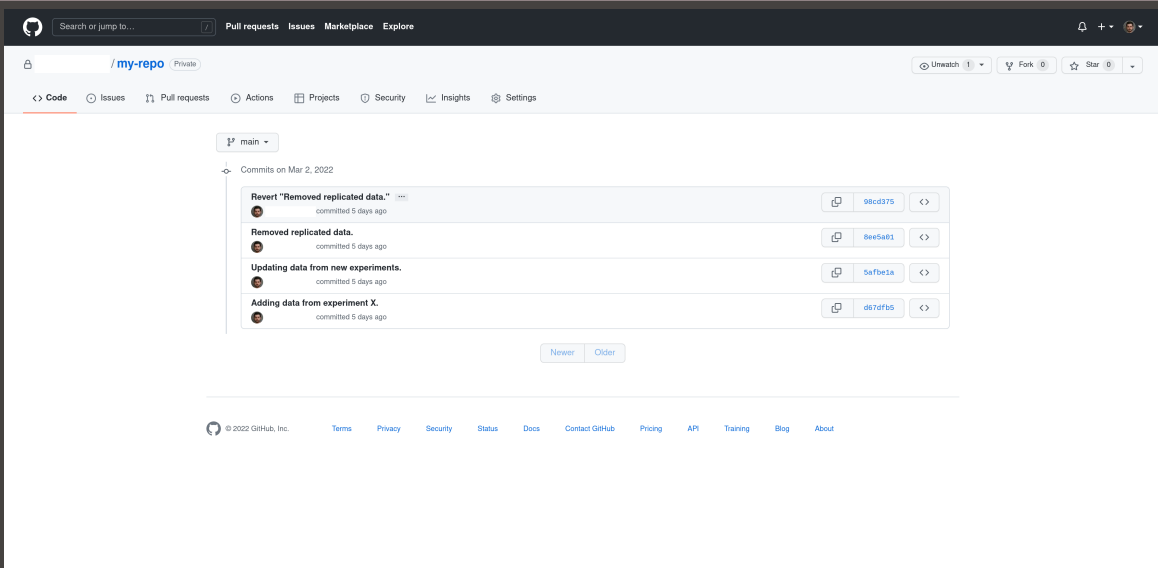
About My GitHub repo
0 stars
1 watching
0 forks

Releases
No releases published
[Create a new release](#)

Packages
No packages published
[Publish your first package](#)

© 2022 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

View repo on GitHub



The screenshot shows the GitHub interface for a repository named 'my-repo'. At the top, there is a search bar and navigation links for Pull requests, Issues, Marketplace, and Explore. The repository name 'my-repo' is displayed with a 'Private' label. On the right, there are buttons for Unwatch (1), Fork (0), and Star (0). Below the repository name, there are navigation tabs for Code, Issues, Pull requests, Actions, Projects, Security, Insights, and Settings. The main content area shows the commit history for the 'main' branch, with a heading 'Commits on Mar 2, 2022'. There are four commits listed, each with a commit message, a commit hash, and a 'committed 5 days ago' timestamp. The commit messages are: 'Revert "Removed replicated data."', 'Removed replicated data.', 'Updating data from new experiments.', and 'Adding data from experiment X.'. The commit hashes are: 98cd375, 8ee5a81, 5afbe1a, and d67dfb5. At the bottom of the page, there is a footer with the GitHub logo, copyright information, and a list of links: Terms, Privacy, Security, Status, Docs, Contact GitHub, Pricing, API, Training, Blog, and About.

Search or jump to... Pull requests Issues Marketplace Explore

/ my-repo Private Unwatch 1 Fork 0 Star 0

<> Code Issues Pull requests Actions Projects Security Insights Settings

main

Commits on Mar 2, 2022

Revert "Removed replicated data." committed 5 days ago	98cd375	<>
Removed replicated data. committed 5 days ago	8ee5a81	<>
Updating data from new experiments. committed 5 days ago	5afbe1a	<>
Adding data from experiment X. committed 5 days ago	d67dfb5	<>

Newer Older

© 2022 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

GitHub navigation: Search or jump to... Pull requests Issues Marketplace Explore

Repository: /my-repo (Private) Unwatch 1 Fork 0 Star 0

Code Issues Pull requests 1 Actions Projects Security Insights Settings

Updating data from new experiments. [Browse files](#)

main committed 7 days ago 1 parent d67dfb5 commit 5afbe1a666ba831626542e2df9474213eb42237f

Showing 1 changed file with 1 addition and 0 deletions. Split Unified

datafile.dat

... 00 -1 +1,2 00	1 some data
1 some data	2 + updated data

0 comments on commit 5afbe1a Lock conversation

Write Preview H B I `↵`

Leave a comment

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

[Comment on this commit](#)

[Unsubscribe](#) You're receiving notifications because you're watching this repository.

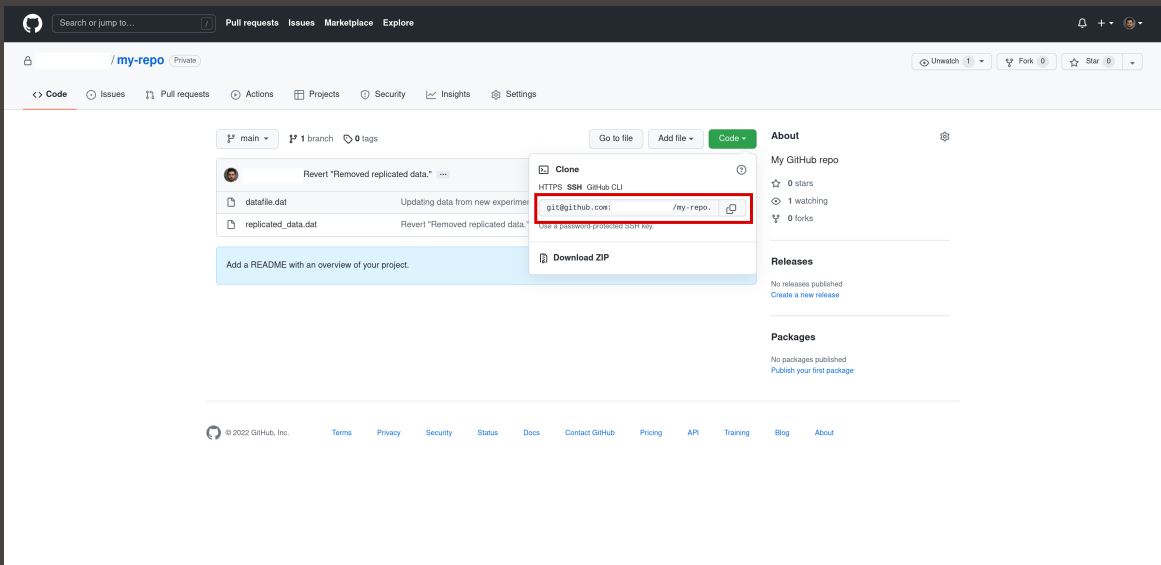
Pull repo from GitHub

The screenshot shows the GitHub interface for a repository named 'my-repo'. At the top, there is a search bar and navigation links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below this, the repository name 'my-repo' is displayed with a 'Private' label. On the right side of the repository header, there are buttons for 'Unwatch' (1), 'Fork' (0), and 'Star' (0). A secondary navigation bar includes links for 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Security', 'Insights', and 'Settings'. The main content area features a file browser with a 'main' branch selected, showing 1 branch and 0 tags. A 'Code' button is highlighted with a red box. Below the file browser, a commit history table is visible:

Commit Hash	Message	Time	Commits
98cd375	Revert "Removed replicated data."	5 days ago	4
	datafile.dat	Updating data from new experiments.	5 days ago
	replicated_data.dat	Revert "Removed replicated data."	5 days ago

Below the commit history, there is a light blue box with the text 'Add a README with an overview of your project.' and a green 'Add a README' button. On the right side, the 'About' section shows 'My GitHub repo' with 0 stars, 1 watching, and 0 forks. The 'Releases' section indicates 'No releases published' with a link to 'Create a new release'. The 'Packages' section indicates 'No packages published' with a link to 'Publish your first package'. The footer contains copyright information for GitHub, Inc. and various links like 'Terms', 'Privacy', 'Security', 'Status', 'Docs', 'Contact GitHub', 'Pricing', 'API', 'Training', 'Blog', and 'About'.

Pull repo from GitHub



The screenshot shows a GitHub repository page for a repository named "my-repo". The repository is private and has 1 branch and 0 tags. The "Code" button is highlighted, and the "Clone" dropdown menu is open, showing the following options:

- Clone (selected)
- HTTPS
- SSH (highlighted with a red box)
- GitHub CLI

The SSH URL is `git@github.com: /my-repo`. Below the SSH option, there is a note: "Use a password-protected SSH key." The "Download ZIP" option is also visible.

The repository page also shows the following information:

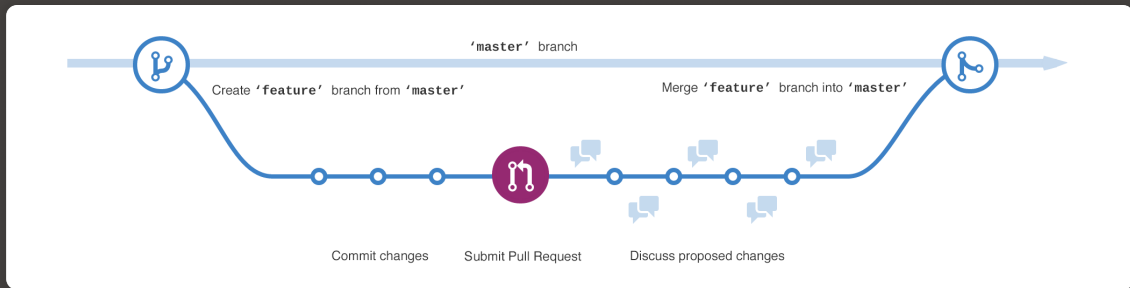
- Repository name: my-repo (Private)
- Buttons: Unwatch (1), Fork (0), Star (0)
- Navigation: Code, Issues, Pull requests, Actions, Projects, Security, Insights, Settings
- Repository details: 0 stars, 1 watching, 0 forks
- Releases: No releases published, Create a new release
- Packages: No packages published, Publish your first package

At the bottom of the page, there is a footer with the following text: © 2022 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

- Pull my-repo from GitHub onto *Niagara*:

```
laptop:~$ ssh -A USERNAME@niagara.computecanada.ca
user@nia-login02:~$ git clone git@github.com:username/my-repo.git
Cloning into 'my-repo'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 9 (delta 1), reused 9 (delta 1), pack-reused 0
Receiving objects: 100% (9/9), done.
Resolving deltas: 100% (1/1), done.
```

- Pull requests are a way to *merge* changes from a new branch into the main branch
- They allow teams to review and either accept or reject new changes
- Powerful tool to help prevent new changes from breaking old code
- Can also run regression tests in GitHub CI (Continuous Integration)
- More info on GitHub CI here:
<https://docs.github.com/en/actions/automating-builds-and-tests/about-continuous-integration>



Source: <https://uoft-oss.github.io/git-workflow/>

- Create a new branch and add some changes locally:

```
laptop:~/my-repo$ git checkout -b new_feature
Switched to a new branch 'new_feature'
laptop:~/my-repo$ git add hello.c
laptop:~/my-repo$ git commit hello.c -m "Hello world program."
[new_feature f3a4091] Hello world program.
1 file changed, 7 insertions(+)
create mode 100644 hello.c
laptop:~/my-repo$ git commit datafile.dat -m "Fixed error."
[new_feature f2b6fe3] Fixed error.
1 file changed, 2 insertions(+), 1 deletion(-)
```


- Push new branch to GitHub:

```
laptop:~/my-repo$ git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 16 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 703 bytes | 703.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'new_feature' on GitHub by visiting:
remote:   https://github.com/username/my-repo/pull/new/new_feature
remote:
To github.com:username/my-repo.git
 * [new branch]      new_feature -> new_feature
```

GitHub: Create a pull request

The screenshot shows the GitHub interface for a repository named 'my-repo'. At the top, there is a search bar and navigation links for Pull requests, Issues, Marketplace, and Explore. The repository name 'my-repo' is displayed with a 'Private' label. On the right, there are buttons for Unwatch (1), Fork (0), and Star (0). Below the repository name, there are tabs for Code, Issues, Pull requests, Actions, Projects, Security, Insights, and Settings. A yellow banner indicates that the 'new_feature' branch has recent pushes 3 minutes ago, with a 'Compare & pull request' button. Below this, there are buttons for 'main', '2 branches', and '0 tags', along with 'Go to file', 'Add file', and 'Code' buttons. A commit history table shows a recent commit titled 'Revert "Removed replicated data."' with the hash '98cd375', pushed 5 days ago, and containing 4 commits. The commit details table lists two files: 'datafile.dat' (Updating data from new experiments, 5 days ago) and 'replicated_data.dat' (Revert "Removed replicated data.", 5 days ago). A blue banner prompts the user to 'Add a README with an overview of your project.' with an 'Add a README' button. On the right side, there are sections for 'About' (My GitHub repo, 0 stars, 1 watching, 0 forks), 'Releases' (No releases published, Create a new release), and 'Packages' (No packages published, Publish your first package). The footer contains the GitHub logo, copyright information for 2022, and various links like Terms, Privacy, Security, Status, Docs, Contact GitHub, Pricing, API, Training, Blog, and About.

GitHub: Create a pull request

Search or jump to... Pull requests Issues Marketplace Explore

/my-repo Private Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Security Insights Settings

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base: main ← compare: new_feature ✓ Able to merge. These branches can be automatically merged.

New feature

Write Preview H B I

Leave a comment

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

Create pull request

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

Reviewers
No reviews

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Development ⓘ
Use [Closing keywords](#) in the description to automatically close issues

Helpful resources
[GitHub Community Guidelines](#)

2 commits 2 files changed 1 contributor

GitHub: Create a pull request

The screenshot shows a GitHub pull request interface for a repository named 'my-repo'. The pull request title is 'New feature #1' and it is currently in an 'Open' state. The pull request description indicates that the user wants to merge 2 commits into the 'main' branch from the 'new_feature' branch. The interface includes a navigation bar with options like 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Security', 'Insights', and 'Settings'. Below the title, there are tabs for 'Conversation', 'Commits', 'Checks', and 'Files changed'. A comment from the user is visible, stating 'commented now' with 'No description provided.' Below the comment, a commit history is shown with two commits: 'Hello world program.' and 'Fixed error.'. A green box highlights the status: 'Continuous integration has not been set up' and 'This branch has no conflicts with the base branch'. A 'Merge pull request' button is visible. On the right side, there are sections for 'Reviewers', 'Assignees', 'Labels', 'Projects', 'Milestone', 'Development', and 'Notifications'.

GitHub: Create a pull request

The screenshot shows a GitHub pull request interface. At the top, there's a navigation bar with 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The repository name is 'my-repo' (Private). The pull request title is 'New feature #1'. It shows 'wants to merge 2 commits into main from new_feature'. There are 2 files changed: 'datafile.dat' and 'hello.c'. The diff for 'datafile.dat' shows a deletion of 'updated data' and additions of 'fixed data' and 'new data'. The diff for 'hello.c' shows the addition of a C program that prints 'Hello World!'. The footer contains copyright information for GitHub, Inc. and a list of links: Terms, Privacy, Security, Status, Docs, Contact GitHub, Pricing, API, Training, Blog, About.

- Create an account on GitHub
- Add an SSH key to your GitHub account (<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>)
- Create a repository on GitHub
- *Push* a repository from your computer to GitHub:

```
laptop:~/my-repo$ git remote add origin git@github.com:username/my-repo.git
laptop:~/my-repo$ git branch -M main
laptop:~/my-repo$ git push -u origin main
```

- *Pull* your repository from GitHub to your computer:

```
laptop:~$ git clone git@github.com:username/my-repo.git
```

- Create and accept a *pull request*

- GitHub Skills: <https://skills.github.com/>

Support

Questions? Need help?

Don't be afraid to contact us! We are here to help.

- Email to support@scinet.utoronto.ca or to niagara@computecanada.ca
- Contact me directly at: james.willis@scinet.utoronto.ca.

Section 6

Extra Slides

- Imagine a bug has been discovered in your codebase but you don't know when or where

- Imagine a bug has been discovered in your codebase but you don't know when or where
- Git provides a utility to track down which commit first introduced the bug
- `git bisect` performs a binary search of the commit history
 - Give it a *bad* commit and a *good* commit
 - It searches all of the commits in between

- Let's use it to find a bug

- Let's use it to find a bug
- Here is a simple repo with two files: `file.txt` and `file_2.txt` containing text over multiple commits:

```
laptop:~/bisect-test$ git log --oneline
f58c76c (HEAD -> master) 6th commit.
2395b1a 5th commit.
a2923e9 4th commit.
1698376 3rd commit.
6b78593 2nd commit.
5c1b9ad 1st commit.
```

- We know the latest version of the repo contains the bug and let's assume that the first commit doesn't

- Now that we know a *bad* commit and a *good* commit we can begin the bisect:

```
laptop:~/bisect-test$ git bisect start
```

- Now that we know a *bad* commit and a *good* commit we can begin the bisect:

```
laptop:~/bisect-test$ git bisect start
```

- Let it know about the bad and good commits:

```
laptop:~/bisect-test$ git bisect good 5c1b9ad
laptop:~/bisect-test$ git bisect bad f58c76c
Bisecting: 2 revisions left to test after this (roughly 1 step)
[1698376e623e4f05801c5db36f952f142764a04a] 3rd commit.
```

- That last command checks the code out at a previous commit halfway between the good and bad commit

- Looking at file.txt:

```
laptop:~/bisect-test$ cat file.txt
The
quick
brown
fox
jumps
over
the
bug
```

we see the bug is still there

- Looking at file.txt:

```
laptop:~/bisect-test$ cat file.txt
The
quick
brown
fox
jumps
over
the
bug
```

we see the bug is still there

- We tell bisect that this commit is still bad:

```
laptop:~/bisect-test$ git bisect bad
Bisecting: 0 revisions left to test after this (roughly 0 steps)
[6b78593f95878a124fe66bc17bece55e15924c14] 2nd commit.
```


- It then checks the code out at another commit halfway between the **latest** bad commit and the good commit
- When we look at `file.txt` now:

```
laptop:~/bisect-test$ cat file.txt
The
quick
brown
fox
jumps
```

we see there is no bug anymore

- Let bisect know and it will tell you the first commit which introduced the bug:

```
laptop:~/bisect-test$ git bisect good
1698376e623e4f05801c5db36f952f142764a04a is the first bad commit
commit 1698376e623e4f05801c5db36f952f142764a04a
Author: willis <james.willis@scinet.utoronto.ca>
Date: Thu Jun 8 13:25:51 2023 -0400
```

3rd commit.

```
file.txt | 3 +++
1 file changed, 3 insertions(+)
```

- This commit can now be analysed to see which code was edited to create the bug and a fix can be applied

- Finally exit bisect with:

```
laptop:~/bisect-test$ git bisect reset  
Previous HEAD position was 6b78593 2nd commit.  
Switched to branch 'master'
```

- Now that we have found the bug we have to apply the fix to the repo

- Now that we have found the bug we have to apply the fix to the repo
- There are multiple ways to do this:
 - If your commits have been pushed to a remote repo and are **public** use `git revert`; or
 - If your commits are purely **local** use `git reset`
- Let's look at each case

- `git revert` undoes a commit by applying the inverse of it as a new commit
- This avoids rewriting the commit history of the repo which is important to maintain integrity and reliable collaboration

- `git revert` undoes a commit by applying the inverse of it as a new commit
- This avoids rewriting the commit history of the repo which is important to maintain integrity and reliable collaboration
- Let's use it to fix our bug
- Remember the bug was first introduced in the 3rd commit:

```
1698376e623e4f05801c5db36f952f142764a04a is the first bad commit
commit 1698376e623e4f05801c5db36f952f142764a04a
Author: willis <james.willis@scinet.utoronto.ca>
Date: Thu Jun 8 13:25:51 2023 -0400
```

```
3rd commit.
```

```
file.txt | 3 +++
1 file changed, 3 insertions(+)
```

- We need to give `git revert` the commit we wish to undo:

```
laptop:~/bisect-test$ git revert 1698376
Auto-merging file.txt
CONFLICT (content): Merge conflict in file.txt
error: could not revert 1698376... 3rd commit.
hint: after resolving the conflicts, mark the corrected paths
hint: with 'git add <paths>' or 'git rm <paths>'
hint: and commit the result with 'git commit'
```

- However, this has caused what is known as a CONFLICT
- Git has undone the changes of the commit to `file.txt` but that file has been changed in subsequent commits (4th, 5th and 6th)

- `file.txt` now looks like this:

```
The
quick
brown
fox
jumps
<<<<<<< HEAD
over
the
bug
lazy
dog.
=====
>>>>>>> parent of 1698376... 3rd commit.
```

- We edit the file to remove the bug and any lines starting with `<<</>>>/===`

- We edit the file to remove the bug and any lines starting with <<</>>>/===
- Then let git know that the CONFLICT has been resolved and to continue with the revert:

```
laptop:~/bisect-test$ git add file.txt
laptop:~/bisect-test$ git revert --continue
[master eb03ecb] Revert "3rd commit." and fix bug.
1 file changed, 1 deletion(-)
```

- The git revert --continue command will open a new window where you can edit the commit message
- Save and close the file and the revert will be complete

- Now looking at the git log:

```
laptop:~/bisect-test$ git log --oneline
eb03ecb (HEAD -> master) Revert "3rd commit." and fix bug.
f58c76c 6th commit.
2395b1a 5th commit.
a2923e9 4th commit.
1698376 3rd commit.
6b78593 2nd commit.
5c1b9ad 1st commit.
```

- We see that we have retained the commit history and the bug fix has been applied!

- `git reset` undoes changes by rolling the repo back to a specific commit

- `git reset` undoes changes by rolling the repo back to a specific commit
- However, it does this by rewriting the commit history
- All commits and changes after that specified commit will be **deleted**

- `git reset` undoes changes by rolling the repo back to a specific commit
- However, it does this by rewriting the commit history
- All commits and changes after that specified commit will be **deleted**
- **So be very careful with this command**
- Also, if after a `git reset` you try to push those changes to a *public* repo that contains the commit you removed it will fail

- Let's look at an example:

```
laptop:~/bisect-test$ git log --oneline
eb03ecb (HEAD -> master) Revert "3rd commit." and fix bug.
f58c76c 6th commit.
2395b1a 5th commit.
a2923e9 4th commit.
1698376 3rd commit.  << Bug found
6b78593 2nd commit.
5c1b9ad 1st commit.
```

- We shall reset to the commit *before* the bug was introduced:

```
laptop:~/bisect-test$ git reset --hard 6b78593
HEAD is now at 6b78593 2nd commit.
```


- Let's look at the contents of file.txt:

```
laptop:~/bisect-test$ cat file.txt
The
quick
brown
fox
jumps
```

- And the git log:

```
laptop:~/bisect-test$ git log --oneline
6b78593 (HEAD -> master) 2nd commit.
5c1b9ad 1st commit.
```

- Experiment with:
 - `git bisect`
 - `git revert`
 - `git reset`
- Try and revert a previous commit