Waipapa
Taumata Rau
University
of Auckland

Introduction to Version Control using Git

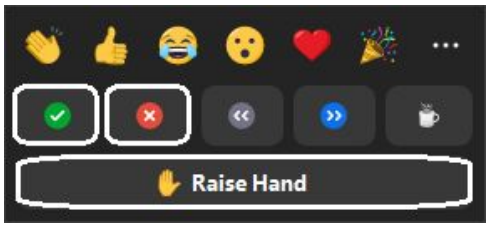
Noel Zeng, James Love and Victor Gambarini

ResBaz
RESEARCH BAZAAR

July 2025

Introduction to Version Control using Git

Setup Instructions:
https://uoa-eresearch.github.io/git-novice/#installing-git
Ask questions in the chat at any time
Use ‘Raise Hand’ ✋ when asked for verbal responses
Use ‘Yes’ & ‘No’ reactions to indicate status
When you are ready to begin select ‘Yes’



He karakia - *Tūtawa* by Professor Scotty Morrison

Tūtawa mai i runga

Tūtawa mai i raro

Tūtawa mai i roto

Tūtawa mai i waho

Kia tau ai

Te mauri tū

Te mauri ora

Ki te katoa

Haumi ē! Hui ē!

Taiki ē!

Come forth from above,

below,

within,

and from the environment

vitality

and well being,

for all.

Strengthened in unity.

Version control

- **Introduction to Automated Version Control**
- Setting up Git
- Creating a Repository
- Tracking Changes
- Exploring History
- Ignoring Things
- Remotes in GitHub
- Collaboration
- Open Research, Licensing, Citation and Other Hosting Options

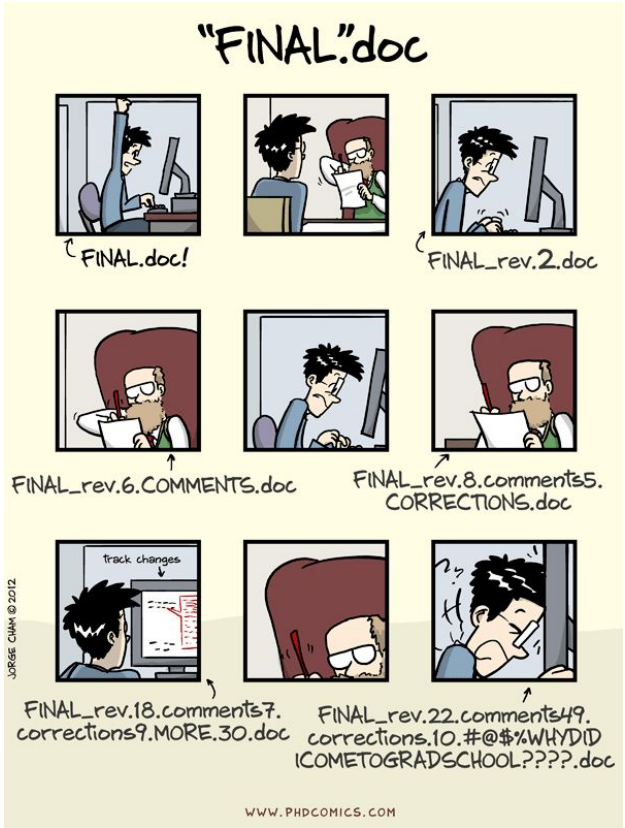
5 minute break at 10:00am

10 minute break at 11:00am

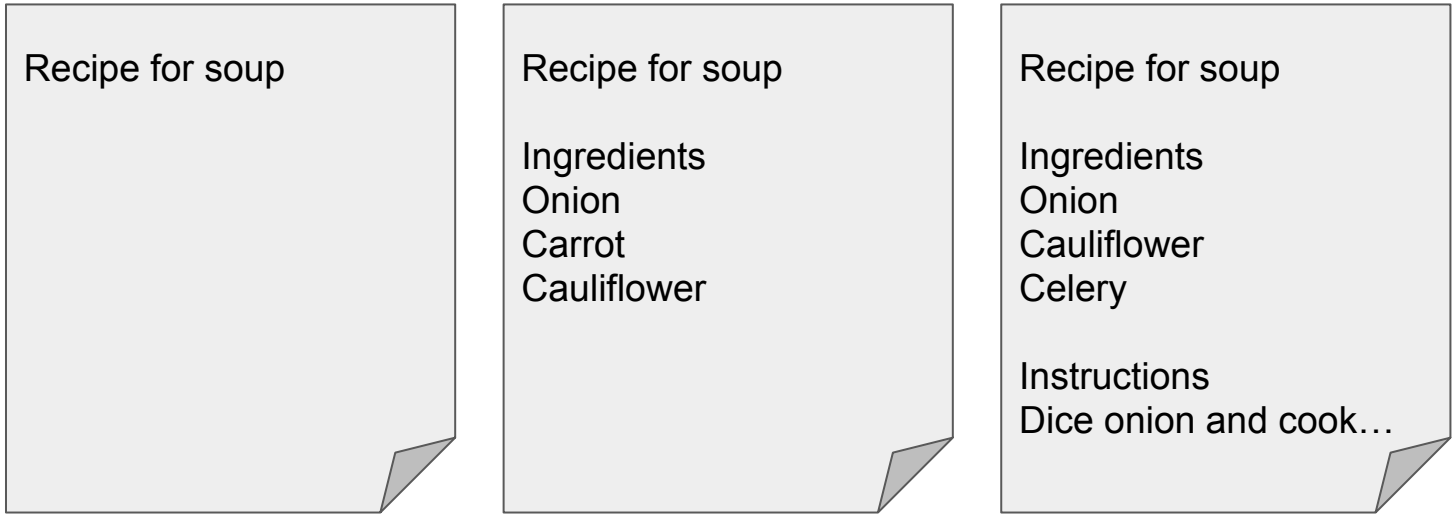
Version control: *Make a copy of file*

- As we work on our project, we need to keep track of changes
- One approach: Make a copy of your code, name it something else (e.g. `script_edited.py`, `script_noels_changes.py`)

- It can work for smaller scripts, but…
  - Naming can become inconsistent.
  - You may forget which is the newest.
  - Folder and files gets messy.
  - Unworkable for larger set of files.
  - Makes collaboration hard

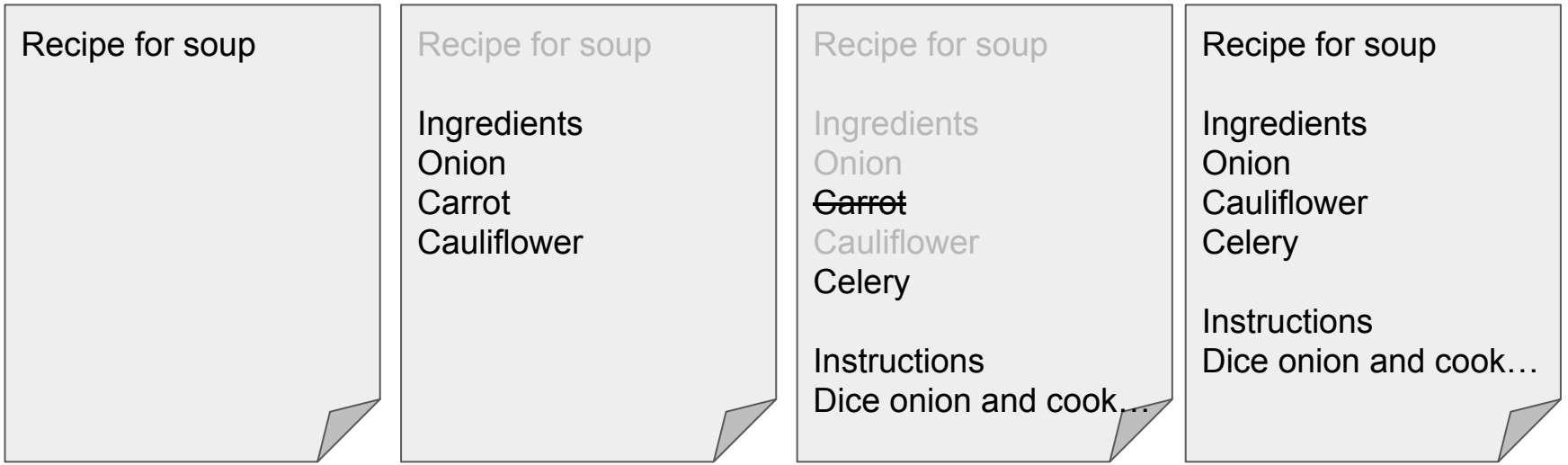


Instead of saving a full copy each time…



recipe.txt recipe2.txt recipe_final.txt

We keep a record of the changes at each step



A better approach - use a version control system

- Version control system keeps track of the steps of changes it took to get to the latest version
- You decide what changes go into the each step (**commit**) of changes.
- A directory of files using a version control system is called a **repository**.
- Makes collaboration easier.

Git: the most popular version control system for code

- Originally created by Linus Torvalds in 2005.
- Made for large software projects with many collaborators.
- Many if not most research and industry software projects use it.
- Originally used for tracking code, but also data and large ***plain text*** documents like theses and papers.

GitHub: most popular Git hosting cloud service

- Git is an open source program, **GitHub** is a commercial hosting service.
- Why use a hosting service? Backup. Collaboration. Discover new projects helpful to you.
- Large number of projects hosted on there.

Scenario: Aspiring cookbook authors

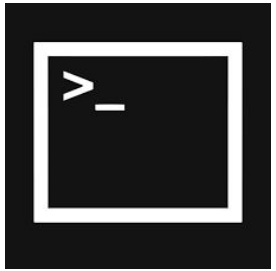
- For simplicity, we won't be working with code today.
- Instead, we're writing recipes!
- Sarah and Kai are each writing their own cookbooks.
- They want to collaborate on some recipes, but back and forth email drafts are becoming unmanageable. So they're giving Git a try.

Megan Bucknall on Unsplash

Version control

- Introduction to Automated Version Control
- **Setting up Git**
- Creating a Repository
- Tracking Changes
- Exploring History
- Ignoring Things
- Remotes in GitHub
- Collaboration
- Conflicts
- Open Research, Licensing, Citation and Other Hosting Options

Let’s jump into the Unix shell!



Windows
Start Menu > Git > Git Bash

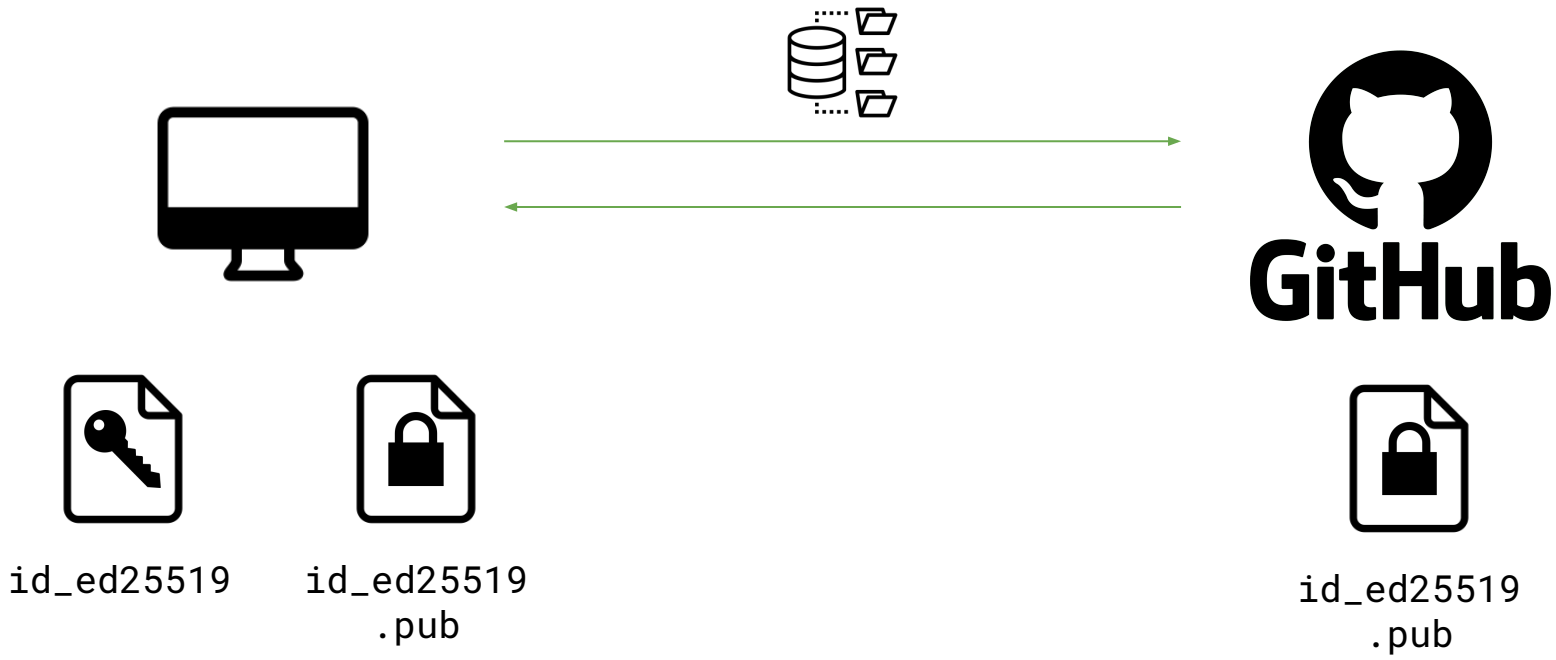
macOS
Dock > Terminal
or
Finder > Applications > Utilities > Terminal
Then, type `bash` and press Enter

Linux
Activities > *type* Terminal
or
Sidebar > Terminal

The git command(s)

```
$ git verb options
```

How do I set up SSH authentication?



git cheatsheet

- `git config` # Change per-repository and per-device settings.
- `ssh-keygen` # Create new SSH key pair files, needed for each new device.

Version control

- Introduction to Automated Version Control
- Setting up Git
- **Creating a Repository**
- Tracking Changes
- Exploring History
- Ignoring Things
- Remotes in GitHub
- Collaboration
- Conflicts
- Open Research, Licensing, Citation and Other Hosting Options

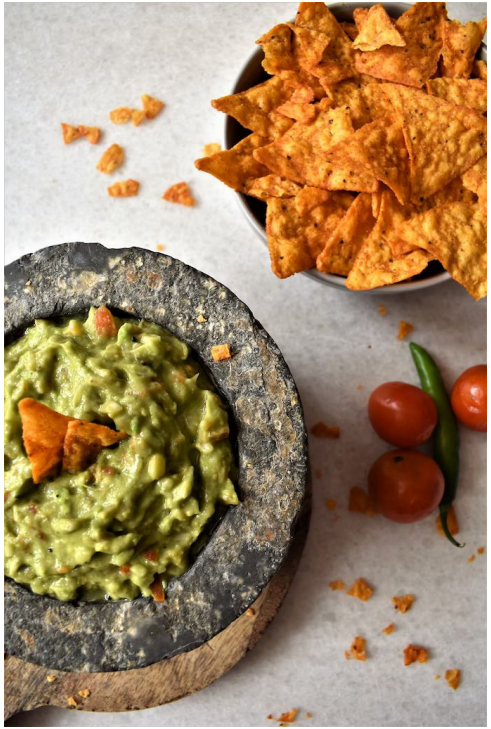
git cheatsheet

- `git config` # Change per-repository and per-device settings.
- `ssh-keygen` # Create new SSH key pair files, needed for each new device.
- `git init` # Initialise the directory into a repository

Version control

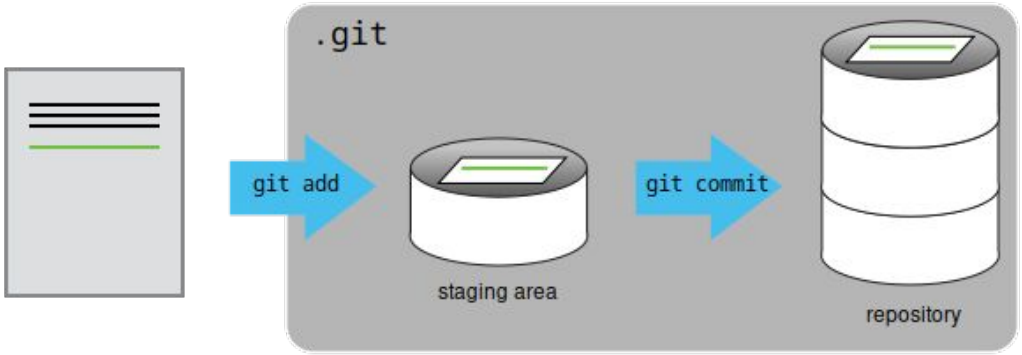
- Introduction to Automated Version Control
- Setting up Git
- Creating a Repository
- **Tracking Changes**
- Exploring History
- Ignoring Things
- Remotes in GitHub
- Collaboration
- Conflicts
- Open Research, Licensing, Citation and Other Hosting Options

Add our first guacamole recipe using nano



yakshi virmani on Unsplash

The staging area



- `git add` specifies what will go into a snapshot/commit, `git commit` *actually* takes a snapshot and creates a permanent record (a **commit**).
- Multiple files can go into the staging area before a commit - allows you to create logical portions, rather than single file changes or a big mixed batch.
- You can add *all* changed files to the staging area (`git add -a`), but it's better to go one-by-one to select only the changes you do want to keep.

A good commit message...

- <50 characters
- Describes what you did and why
- Completes the sentence “If applied, this commit will…”
- If you need to put in more details, add blank line between summary and additional notes.

“Improve recipe with more vegetables”

“Fix bug with incorrect calculations”

“Add capsicum in soup”

“asdf”

https://cbea.ms/git-commit/

Exercise: Choosing a Commit Message

Which of the following commit messages would be most appropriate for the last commit made to mars.txt? **Why do you think it is?**

1. “Changes”
2. “Changed lemon to lime”
3. “Guacamole modified to the traditional recipe”

Exercise: Committing Changes to Git

Which command(s) below would save the changes of myfile.txt to my local Git repository?

1.
```
$ git commit -m "my recent changes"
```

2.
```
$ git init myfile.txt
$ git commit -m "my recent changes"
```

3.
```
$ git add myfile.txt
$ git commit -m "my recent changes"
```

4.
```
$ git commit -m myfile.txt "my recent changes"
```

Challenge: A new repository

- Create a new Git repository on your computer called bio (outside the recipes repository)
- Write three facts about yourself in a file called `me.txt`, commit your changes
- Modify one of the facts, add a fourth fact
- Display the differences between its updated state and its original state.

Hint: Here are some commands we've learned so far. Not all are relevant.

```
$ git diff
$ git log
$ git commit -m "[message]"
$ git init
$ git add [file path]
```

```
$ mv [source] [destination]
$ mkdir [directory name]
$ nano [file name]
$ ls [directory name]
$ cd [directory path]
```

git cheatsheet

- `git config` # Change per-repository and per-device settings.
- `ssh-keygen` # Create new SSH key pair files, needed for each new device.
- `git init` # Initialise the directory into a repository
- `git add` # Adds changes to staging area
- `git commit -m “message”` # Creates a new commit with changes in staging area
- `git log` # Show a log of past commits
- `git diff` # Show difference between commits

Version control

- Introduction to Automated Version Control
- Setting up Git
- Creating a Repository
- Tracking Changes
- **Exploring History**
- Ignoring Things
- Remotes in GitHub
- Collaboration
- Conflicts
- Open Research, Licensing, Citation and Other Hosting Options

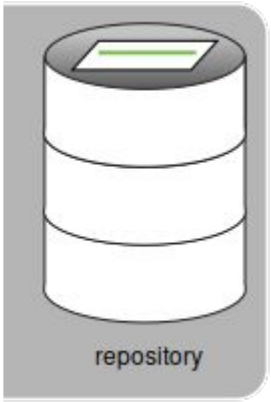
How do we refer to our past commits in git?

(Unique 40-character ID) →

(Unique 40-character ID) →

(Unique 40-character ID) →

…



← HEAD

← HEAD~1

← HEAD~2

…

What's changed since an older commit

```
$ git diff [identifier]
    guacamole.txt
```

Discard changes since the most recent commit

```
$ git checkout HEAD guacamole.txt
```

or

```
$ git checkout -- guacamole.txt
```

Restore an older version of the file

```
$ git checkout [identifier] guacamole.txt
```

- This brings back the older version of the file into **the staging area**.
- You can cancel the restoration by running `$ git checkout HEAD guacamole.txt`.
- Or run `$ git commit` to finalise the restoration!
- If you want to undo a particular change, remember to use the identifier BEFORE when the change happened.

Detached HEAD

What if you run…

`$ git checkout [identifier]`

(without a specific file)

- It does something different! 😱
- Goes into a “detached HEAD” state
- Shows the whole repository’s state at this commit.
- “Look, don’t touch” mode - don’t make any changes here.
- `$ git status` confirms you’re in this mode
- `$ git checkout main` to reattach HEAD.

Exercise: Understanding workflow and history

What is the output of the last command?

```
$ cd recipes
$ (Use nano to create a file called ketchup.txt, adding content "I like tomatoes, therefore I like ketchup")
$ git add ketchup.txt
$ (Use nano to delete the content in ketchup.txt, and replace with "ketchup enhances pasta dishes")
$ git commit -m "my opinions about ketchup"
$ git checkout HEAD ketchup.txt
$ cat ketchup.txt # this will print the content of ketchup.txt on screen
```

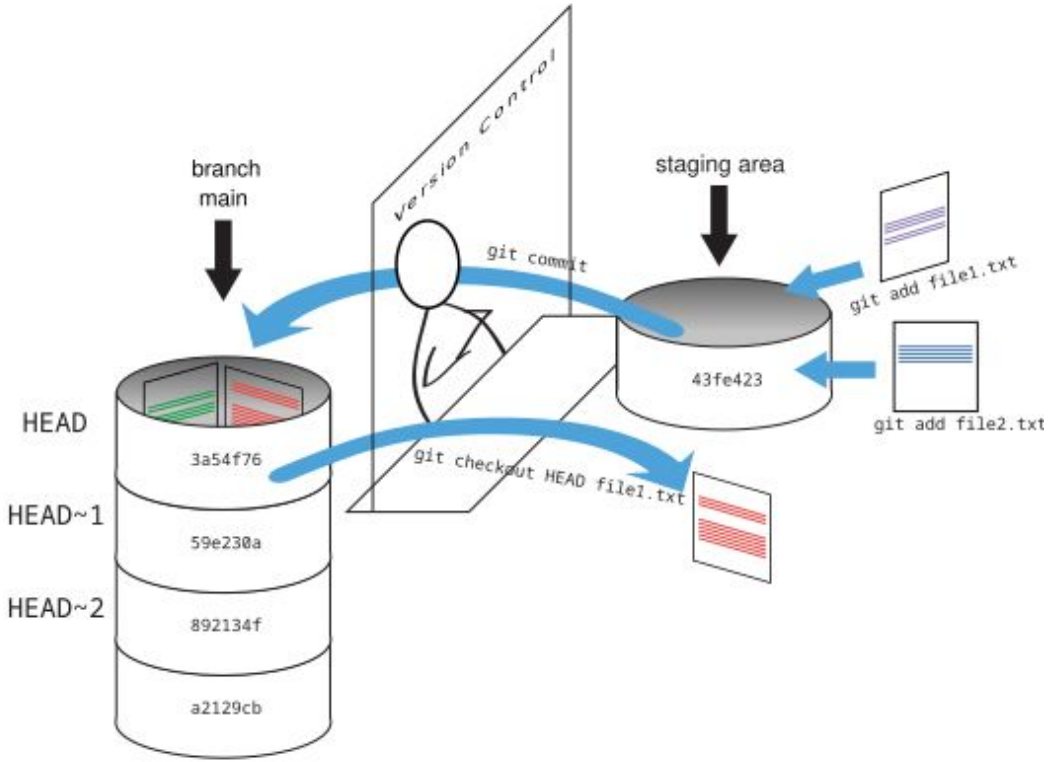
1. ketchup enhances pasta dish
2. I like tomatoes, therefore I like ketchup
3. I like tomatoes, therefore I like ketchup
   ketchup enhances pasta dishes
4. Error because you have changed ketchup.txt without committing the changes

Challenge: git diff

- Consider this command: `git diff HEAD~9 guacamole.txt`. What do you predict this command will do if you execute it? What happens when you do execute it? Why?

- Try another command, `git diff [ID] guacamole.txt`, where [ID] is replaced with the unique identifier for your most recent commit. What do you think will happen, and what does happen?

Putting it all together…



git cheatsheet

- `git config` # Change per-repository and per-device settings.
- `ssh-keygen` # Create new SSH key pair files, needed for each new device.
- `git init` # Initialise the directory into a repository
- `git add` # Adds changes to staging area
- `git commit -m “message”` # Creates a new commit with changes in staging area
- `git log` # Show a log of past commits
- `git diff` # Show difference between commits
- `git checkout` # Discard uncommitted changes or restore an older version of a file

Version control

- Introduction to Automated Version Control
- Setting up Git
- Creating a Repository
- Tracking Changes
- Exploring History
- **Ignoring Things**
- Remotes in GitHub
- Collaboration
- Conflicts
- Open Research, Licensing, Citation and Other Hosting Options

Asking git to ignore files and directories

- Some files don’t need to go into version control
  - Generated files, automatic backup files created by text editors
- Add them to a .gitignore file
- Can use wildcard character (*) to ignore files matching a pattern

Exercise: Including specific files

How would you ignore all .dat files in your root directory except for `final.dat`?

Hint: Find out what ! (the exclamation point operator) does.

git cheatsheet

- `git config` # Change per-repository and per-device settings.
- `ssh-keygen` # Create new SSH key pair files, needed for each new device.
- `git init` # Initialise the directory into a repository
- `git add` # Adds changes to staging area
- `git commit -m “message”` # Creates a new commit with changes in staging area
- `git log` # Show a log of past commits
- `git diff` # Show difference between commits
- `git checkout` # Discard uncommitted changes or restore an older version of a file
- Add a .gitignore file to ignore files git doesn’t need to keep track of.

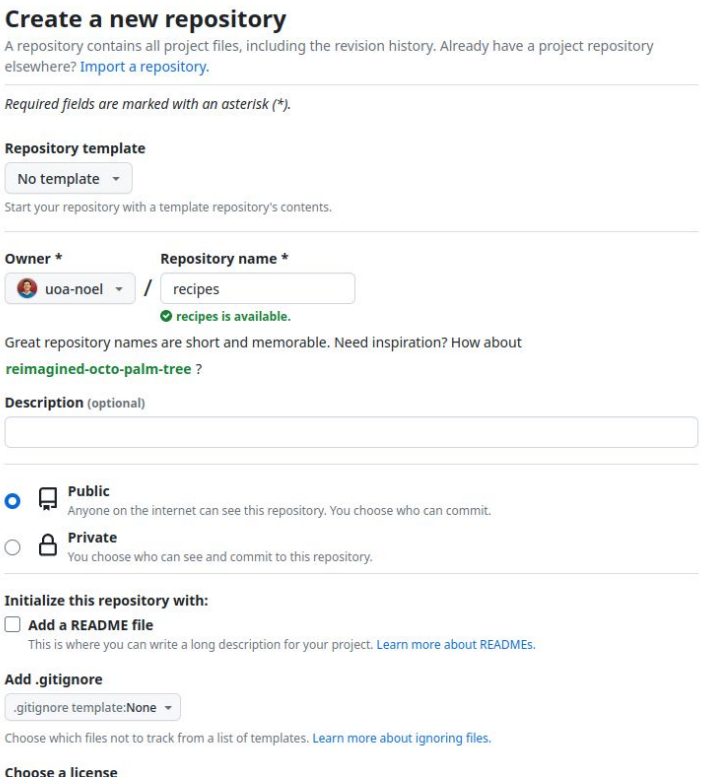
Version control

- Introduction to Automated Version Control
- Setting up Git
- Creating a Repository
- Tracking Changes
- Exploring History
- Ignoring Things
- **Remotes in GitHub**
- Collaboration
- Conflicts
- Open Research, Licensing, Citation and Other Hosting Options

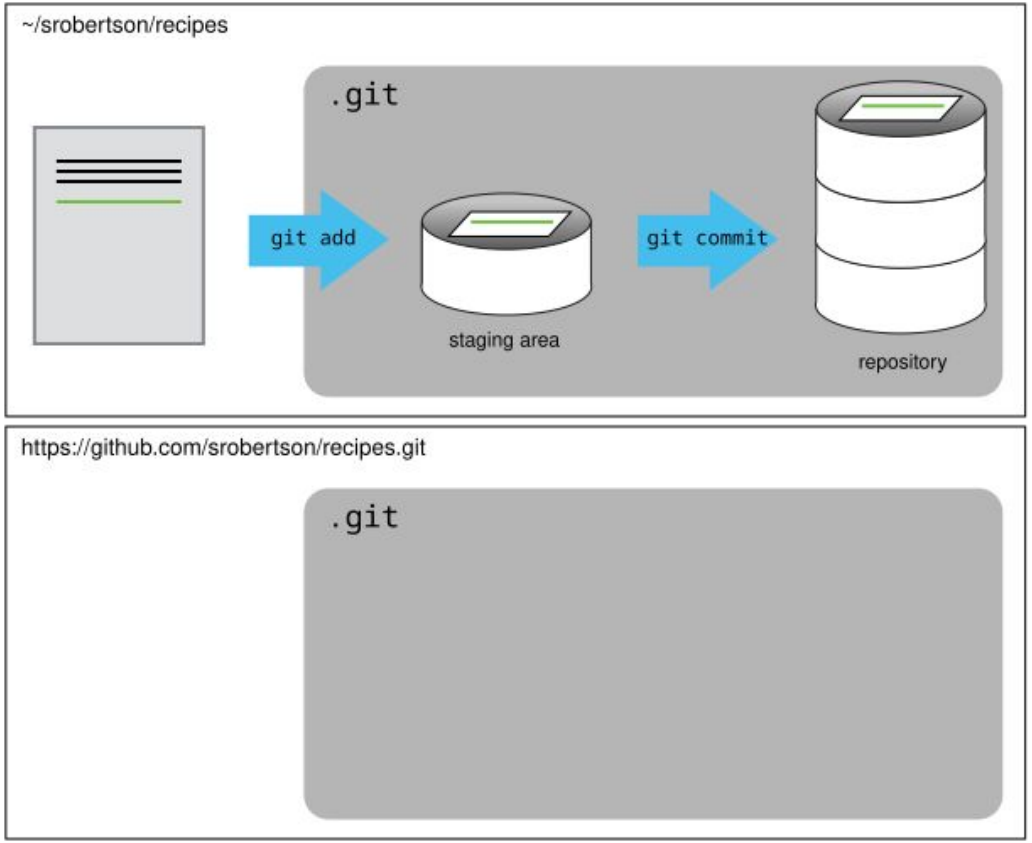
Creating a repository on GitHub

- Equivalent of running this on GitHub's server:

```
$ mkdir recipes
$ cd recipes
$ git init
```



Our repository



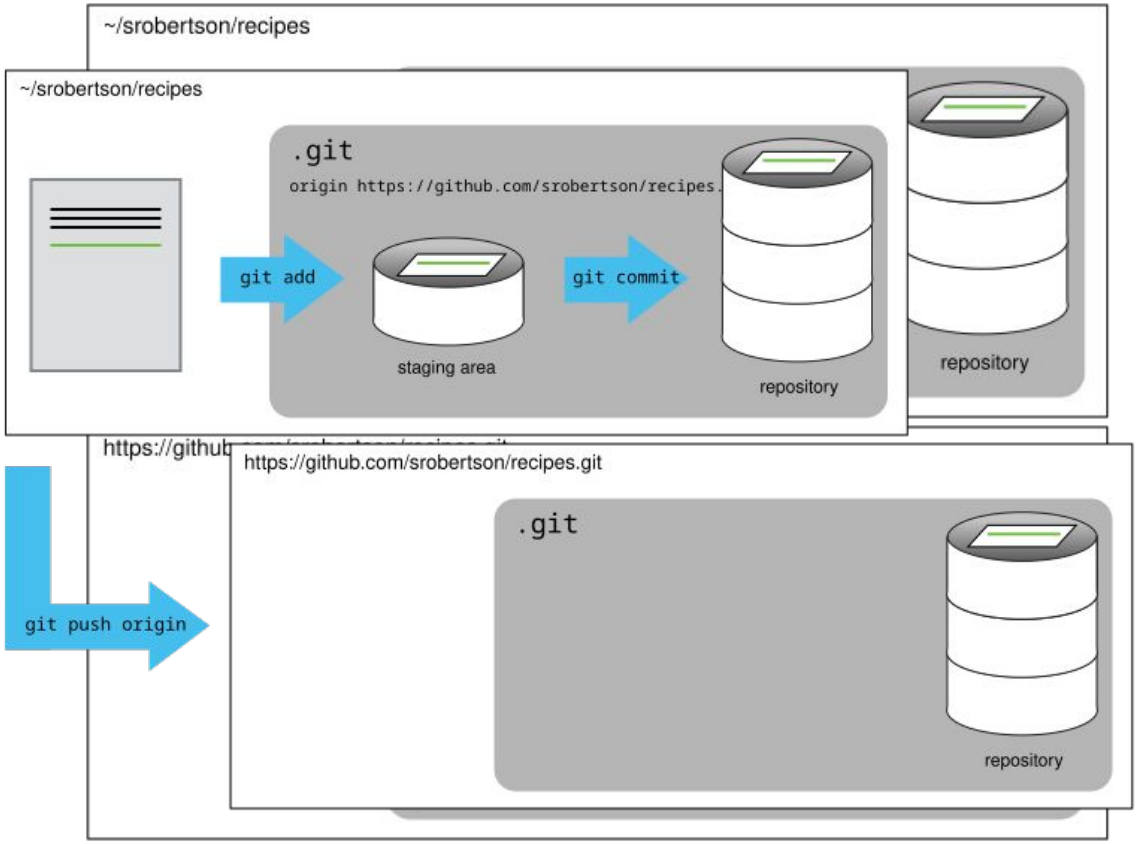
Connect with a remote repository

```
$ git remote add [name] [url]
```

The name is a local label you can use to refer to the remote repository. “origin” is commonly used by convention.

You can also have multiple remotes. This is useful to look into when you are working with a larger project.

Effect of `git push`



Push changes to a remote repository

```
$ git push [remote name] main
```

Add the `-u` option (e.g. `git push -u origin main`) to associate the current branch with a remote branch, so you can use `git pull` without specifying any arguments. This only needs to be done once.

Pull changes from a remote repository

```
$ git pull [remote name] main
```

Exercise: `git push` vs `git commit`

In this episode, we introduced the “git push” command. How is “git push” different from “git commit”?

1. git push records staged changes into your local repository and synchronises all your changes with a remote repository. git commit only does the first part.
2. git push synchronises all your changes with a remote repository. git commit records staged changes into your local repository.
3. git push records staged changes into your local repository. git commit synchronises all your changes with a remote repository.
4. git push only records changes into your local repository. git commit does that and synchronises all your changes with a remote repository.

git cheatsheet

- …
- `git add` # Adds changes to staging area
- `git commit -m “message”` # Creates a new commit with changes in staging area
- `git log` # Show a log of past commits
- `git diff` # Show difference between commits
- `git checkout` # Discard uncommitted changes or restore an older version of a file
- Add a .gitignore file to ignore files git doesn’t need to keep track of.
- `git remote` # connect or disconnect with remote repositories
- `git push and git pull` # push local changes to remote repository and pull remote changes to local repository

For the next section…

For the next section we will be splitting you into pairs. Please add your username and repository address (e.g. https://github.com/AnthonyDShaw/recipes) to the spreadsheet at the following link. The link will also be in the chat.

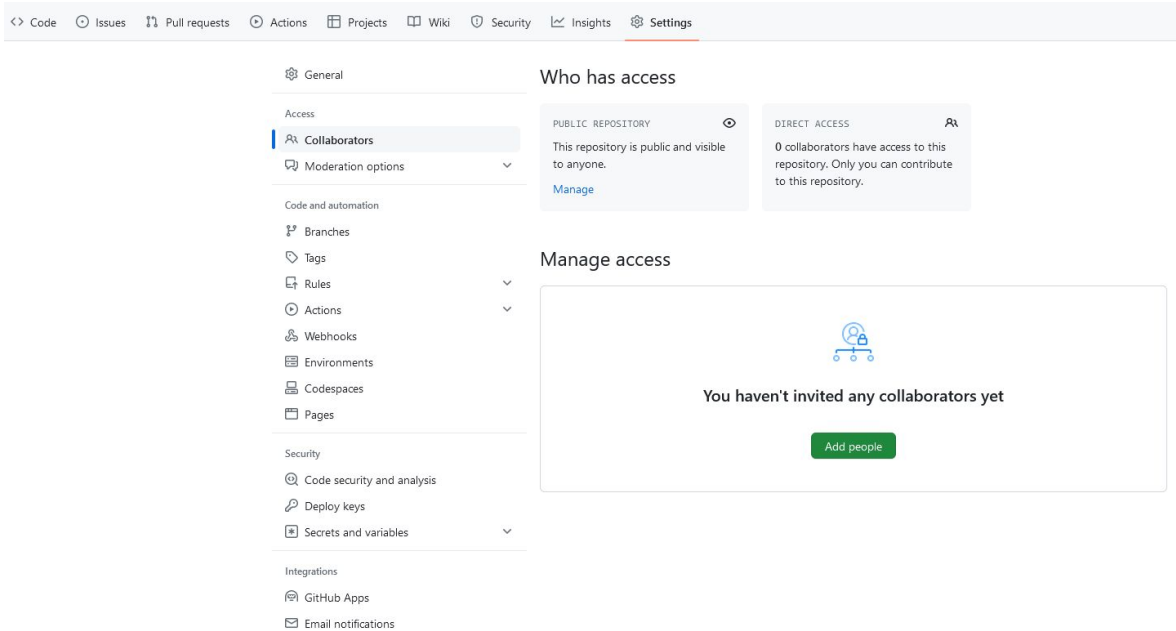
https://tinyurl.com/5x75v69k

Version control

- Introduction to Automated Version Control
- Setting up Git
- Creating a Repository
- Tracking Changes
- Exploring History
- Ignoring Things
- Remotes in GitHub
- **Collaboration**
- Conflicts
- Open Research, Licensing, Citation and Other Hosting Options

Collaborating: Adding a collaborator

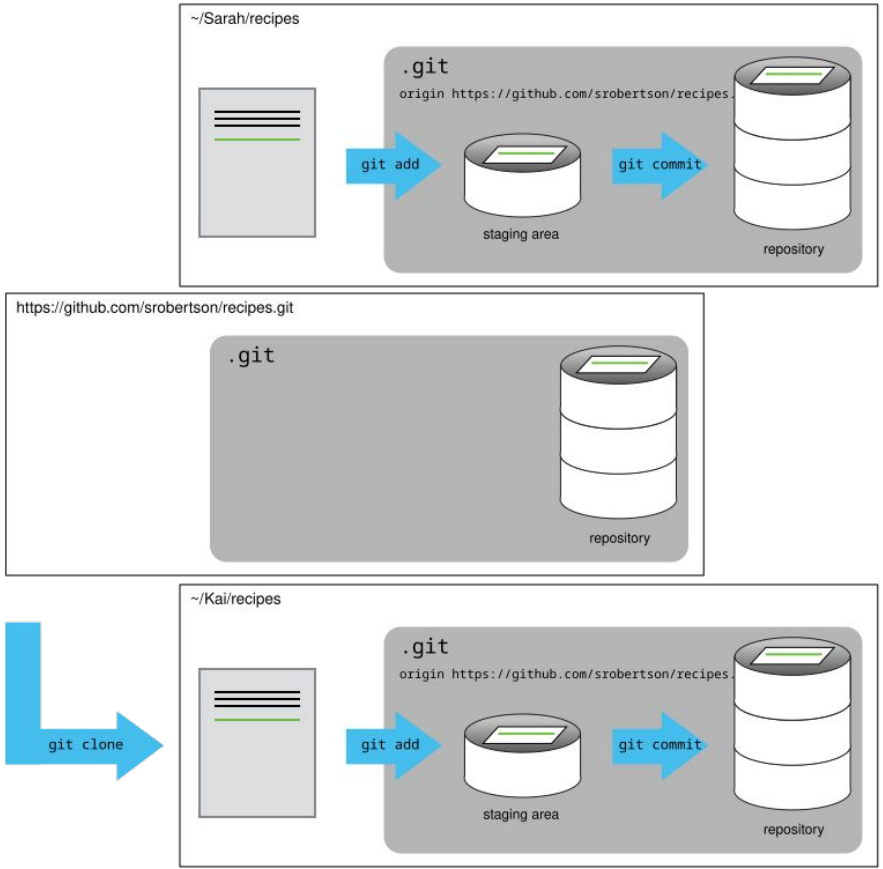
1. Once partners are made give your partner Collaborator access to your repository. On GitHub, click the “Settings” button on the right, select “Collaborators”, click “Add people”, and then enter your partner’s username.
2. To accept access to the Owner’s repo, the Collaborator needs to go to their partner’s repository, or check for email notification



Collaborating: Making a change as the Collaborator

1. `$ git clone git@github.com:AnthonyDShaw/recipes.git ~/Documents/anthony-recipes` #Clone your partners repository and rename it
2. `$ cd ~/Documents/anthony-recipes`
   `$ nano soup.txt` #Create a file with a new recipe
3. `$ git add soup.txt` #Stage the file
   `$ git commit -m "Add ingredients for soup"` #Commit the file
4. `$ git push origin main` #Push the file to the owners repository

Our repository after cloning



Collaborating: Checking the change as the Owner

1. `$ git pull origin main` `#Pull the file from your repository`
2. `$ cat soup.txt` `#Check the changes to the file`

Collaborating: Reviewing changes

If the commit message is not clear on the changes you can use `git fetch origin main` to get the remote changes and then use `git diff main origin/main` to compare. Alternatively, you can go to the repository, click commits and view the most recent commits.

Collaborating: Summary

In this section we:

- Learned how to clone someones repository with `git clone`
- Used `git push` to make a change to this repository
- Used `git pull` to retrieve a change to your local repository
- Noted how to use `git fetch` and `git diff` or the webpage to check changes between commits when the commit message is unclear

git cheatsheet

- …
- `git log` # Show a log of past commits
- `git diff` # Show difference between commits
- `git checkout` # Discard uncommitted changes or restore an older version of a file
- Add a .gitignore file to ignore files git doesn't need to keep track of.
- git clone # Copies an online repository to your local machine
- git pull # Updates your local copy of the repository with any changes made to the online repository
- git push # Updates the online repository with the changes you committed in your local repository

Version control

- Introduction to Automated Version Control
- Setting up Git
- Creating a Repository
- Tracking Changes
- Exploring History
- Ignoring Things
- Remotes in GitHub
- Collaboration
- **Conflicts**
- Open Research, Licensing, Citation and Other Hosting Options

Conflicts

To simulate a Collaborator making a change independently from you:

1. Go to your Github repository (e.g. https://github.com/AnthonyDShaw/recipes)
2. Select the guacamole.md file
3. Select “edit this file” (the pencil near the top right corner)
4. Add a new line
5. Commit the changes with the commit message:
   First step on the instructions

Conflicts: Creating a conflict

Now let’s create a conflict in our local repository. Each of us will make a change to the guacamole.txt file in our local repository associated with `git@github.com:AnthonyDShaw/recipes.git`

1. `$ nano guacamole.txt` #add a line
2. `$ git add guacamole.txt` #Stage the file
3. `$ git commit -m "First step on the instructions"` #Commit the file
4. `$ git push origin main` #Try to push the file to git

Git rejects the second person to push because it detects that the remote repository has new updates that have not been incorporated into the local branch. Now let’s resolve the conflict.

Conflicts: Resolving conflicts

1. `$ git pull origin main` #Attempt to pull the conflict repository from GitHub
2. `$ git config pull.rebase false` # Set default merge divergent branch strategy (if not already set)
   `$ git pull origin main` #Pull the conflict repository from GitHub
3. `$ nano guacamole.txt` #Resolve the conflicted areas
   `$ git add guacamole.txt` #Stage the merged file
4. `$ git commit -m "Merge changes from GitHub"` #Commit the merged file
5. `$ git push origin main` #Push the merged file to GitHub
6. `$ git pull origin main` #Pull the merged file from Github
7. `cat guacamole.txt` #Confirm the conflict is resolved

Conflicts on non-text files

- When trying to merge non-text files such as images you will get a warning saying “Cannot merge binary files”. Meaning git cannot insert conflict markers as it did for the text file example.
- The available version of the file with be HEAD (your version) and an alphanumeric code (the alternative conflicting version) shown in the warning message.
- You can use `git checkout HEAD conflicting_file.jpg` or `git checkout [identifier] conflicting_file.jpg` to chose which file to keep.
- Alternatively, you can keep both images by checking out and renaming itching file with the `mv` command (e.g. `mv conflicting_file.jpg new_name.jpg`.

Conflicts: Avoiding conflicts

- Pull from upstream more frequently, especially before starting new work
- Use topic branches to segregate work, merging to main when complete
- Make smaller more atomic commits
- Push your work when it is done and encourage your team to do the same to reduce work in progress and, by extension, the chance of having conflicts
- Where logically appropriate, break large files into smaller ones so that it is less likely that two authors will alter the same file simultaneously
- Clarify who is responsible for what areas with your collaborators
- Discuss what order tasks should be carried out in with your collaborators so that tasks expected to change the same lines won't be worked on simultaneously
- If the conflicts are stylistic churn (e.g. tabs vs. spaces), establish a project convention that is governing and use code style tools (e.g. htmltidy, perltidy, rubocop, etc.) to enforce, if necessary

Conflicts: Summary

In this section we:

- Intentionally pushed two conflicting changes of the same file to the GitHub repository
- Edited the conflicting file and pushed the file back to the GitHub repository after resolving the conflict
- Noted how this method won't work with non-textual files, such as binaries, and that you will need to use `git checkout` to choose a conflicting file to keep, or rename one of the conflicting files to keep both

git cheatsheet

- …
- `git log` # Show a log of past commits
- `git diff` # Show difference between commits
- `git checkout` # Discard uncommitted changes or restore an older version of a file
- Add a .gitignore file to ignore files git doesn't need to keep track of.
- git clone # Copies an online repository to your local machine
- git pull # Updates your local copy of the repository with any changes made to the online repository
- git push # Updates the online repository with the changes you committed in your local repository

Git can enable open science and research

- Publishing data and code make research reproducible, transparent and open for reuse.
- In one study, publications with publicly available data had a 69% increase in citations. (Piwowar et al 2007)
- You can test how reusable your results are by asking a colleague to try reproducing it based on what you've published.
- Check out University of Auckland's Figshare space and Zenodo.

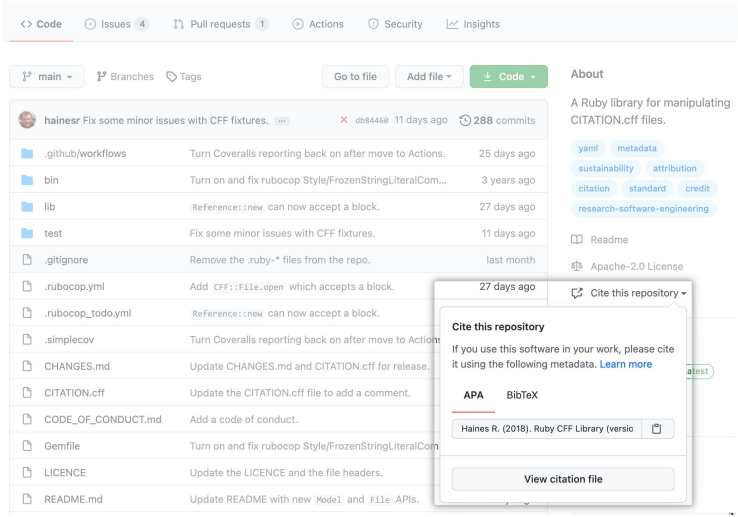
Licensing your code

- To make your code reusable, choose a free and open source copyright license and use it!
- Varying conditions ranging from requiring attribution, requiring users make their changes open source, to requiring the code be used ethically.
- Add a `LICENSE` or `LICENSE.txt` file in your Git repository to tell others.

www.choosealicense.com

Make your code citable

- You can include a `CITATION` or `CITATION.txt` in the Git repository with instructions on how to cite your software.
- GitHub shows a citation popup if you include a `CITATION.cff` file.
- More information: https://swcarpentry.github.io/git-novice/12-citation.html

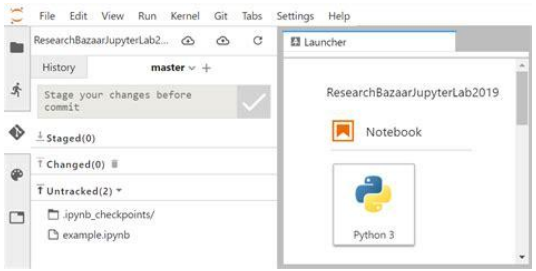


Where to host your Git repository?

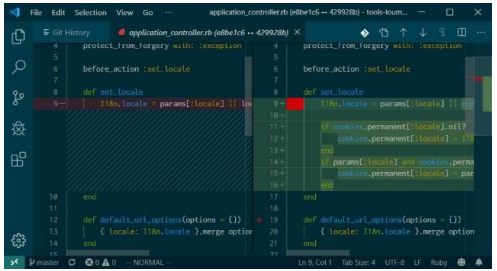
- We’ve looked at GitHub today, but there are other Git hosting services. You can also run a Git hosting server yourself.

Git integration in code editors

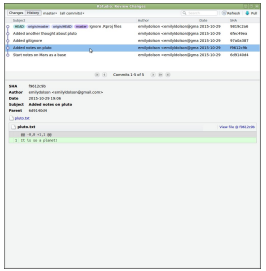
- Many GUI code editors provide graphical Git integrations, so you don't have to use the `git` command directly.
- However, they may be missing more advanced features.



JupyterLab



VSCode



RStudio

Version control: Summary

- Automated version control can help with larger projects and collaboration on code.
- For each computer you use git on, you need to set up authentication and preferences.
- A directory using version control is called a **repository**. Change history (i.e. the list of **commits**) is stored in a .git folder.
- You can choose the changes that go into a commit, adding a description with each.
- You can look at past commits and restore an older version of your files.
- **GitHub** is an online Git hosting service. You can set up a **remote** repository on there and sync it with local ones.
- People can **collaborate** by connecting to a remote repository. Git provides tools for detecting and resolving **conflicts**.
- You can make research better for everyone by **sharing your code**, also by **choosing a license that enables reuse**, and by making it **easier to cite your code**.
- There are alternatives to GitHub, and **graphical interfaces** that make it easier to work with Git. However, commandline git is still useful for trickier situations.

Congratulations, you finished the Git lesson!

He karakia whakakapi - Professor Scotty Morrison

Te whakaeatanga e	*It is completed,*
Te whakaeatanga e	*it is done,*
Tēnei te kaupapa ka ea	*we have achieved our purpose,*
Tēnei te wānanga ka ea	*completed our forum,*
Ko te mauri o te kaupapa ka whakamoea	*let the purpose of our gathering rest for now,*
Ko te mauri o te wānanga ka whakamoea	*let the vitality of our discussions replenish,*
Koa ki runga, koa ki raro	*we depart with fulfilled hearts & minds,*
Haumi e, hui e, tāiki e	*bonded in our common goal & unity*

Waipapa
Taumata Rau
University
of Auckland

Thanks

ResBaz
RESEARCH BAZAAR

Questions?