ADVANCED SOFTWARE ENGINEERING CMP9134

WEEK 1 – DR MOHAMMED AL-KHAFAJIY

Version control in git

(and Github)

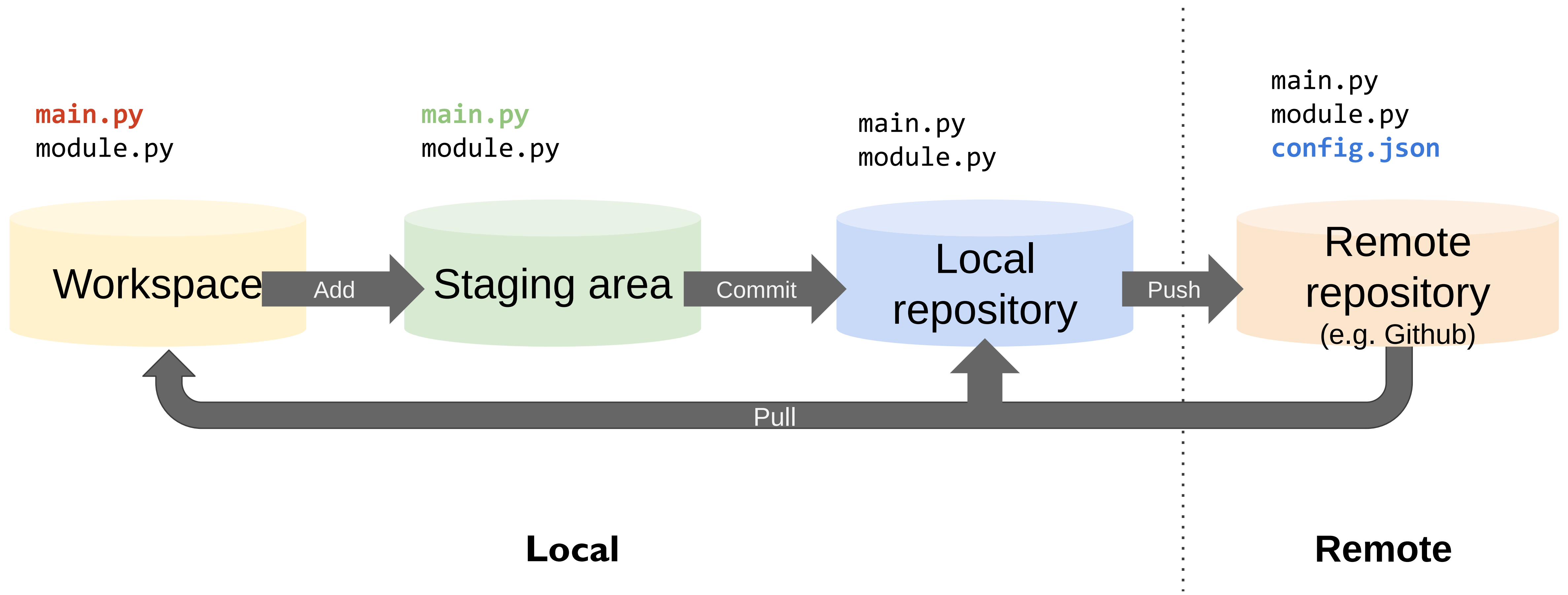
What is `git`?

- A **version control system (VCS)** for tracking changes in source code
  - You can also track changes in LaTeX documents for papers/dissertations
  - Avoid using it to version large files (images, videos) – use Git LFS instead
- It is free, open-source, and cross-platform:
  - Windows: https://git-scm.com/download/win
  - Linux: `sudo apt install git`
  - mac OS: install the Xcode Command Line Tools
- It is an essential tool in the armory of a software developer!

Repositories

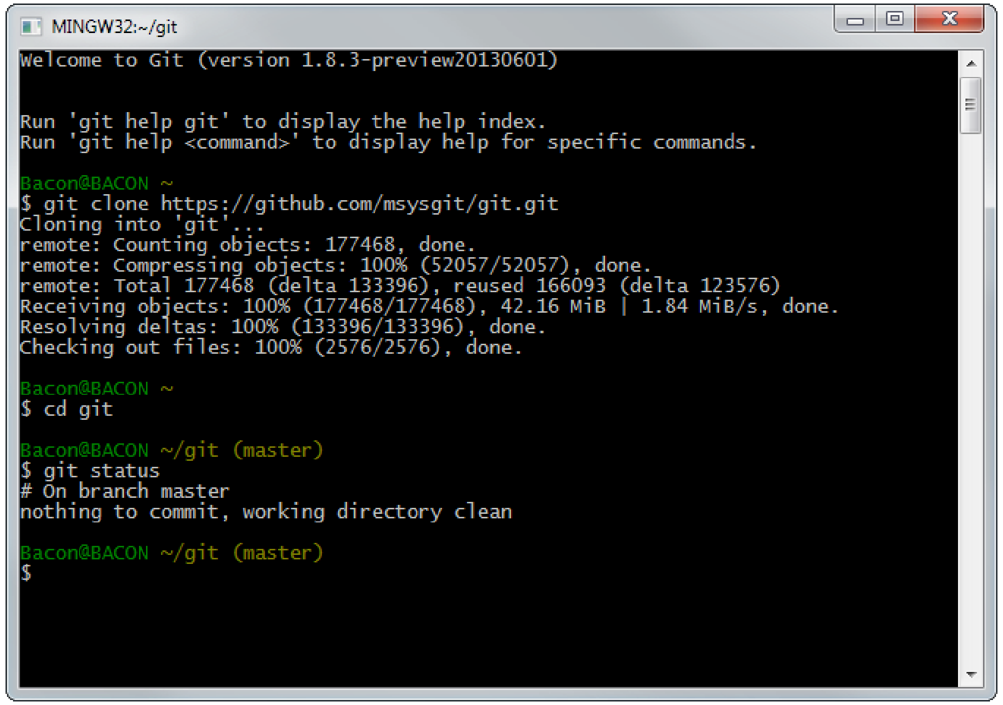
- Individual “projects” are referred as repositories (**repos**)
- By using a git repo, you can:
  - maintain a backup of your source code
  - keep track of changes made by you and others
  - rollback to earlier versions of your software
  - have multiple people work on the code at a same time
  - merge several different working versions, or “branches”
- Let’s take a brief look at how a repository works

Overview


main.py
module.py
main.py
module.py
main.py
module.py
main.py
module.py
config.json
Workspace
Add
Staging area
Commit
Local repository
Push
Remote repository
(e.g. Github)
Pull
Local
Remote

The terminal

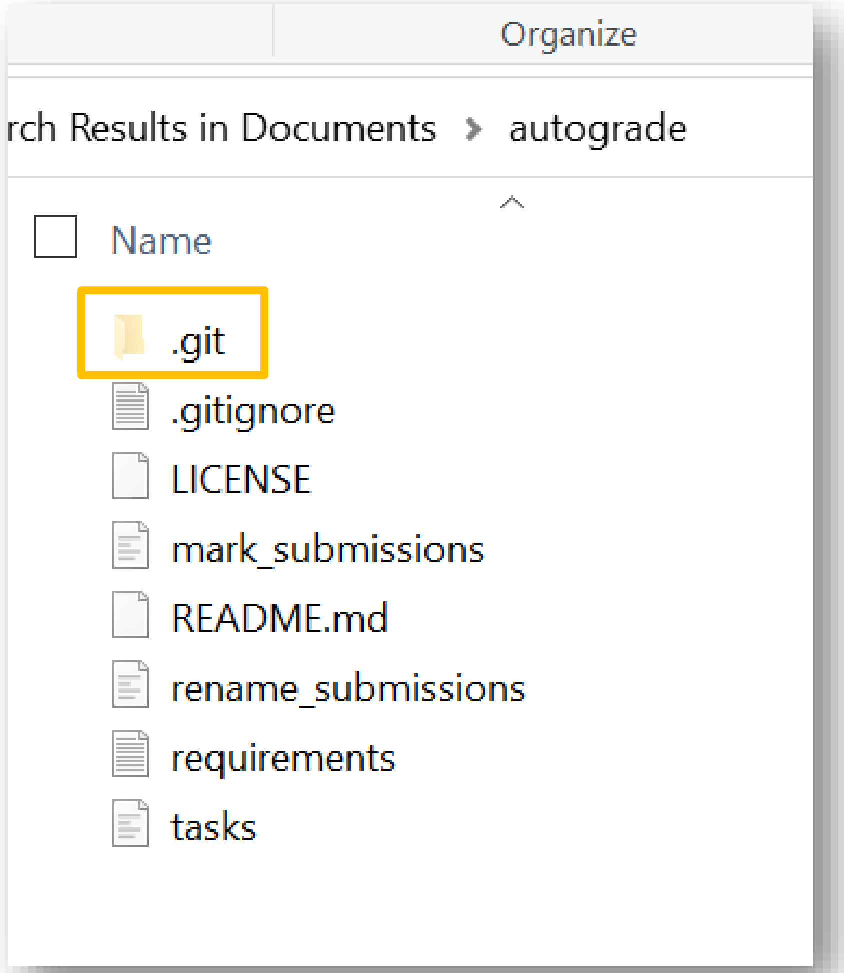
- In this session we'll mostly focus on terminal usage of `git` - but fear not!
- The GUI is perfectly fine, but getting used to the commands helps a lot
- The practical session (later) will show both the command line and GUI-based usage



```
Welcome to Git (version 1.8.3-preview20130601)


Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

Bacon@BACON ~
$ git clone https://github.com/msysgit/git.git
Cloning into 'git'...
remote: Counting objects: 177468, done.
remote: Compressing objects: 100% (52057/52057), done.
remote: Total 177468 (delta 133396), reused 166093 (delta 123576)
Receiving objects: 100% (177468/177468), 42.16 MiB | 1.84 MiB/s, done.
Resolving deltas: 100% (133396/133396), done.
Checking out files: 100% (2576/2576), done.

Bacon@BACON ~
$ cd git

Bacon@BACON ~/git (master)
$ git status
# On branch master
nothing to commit, working directory clean

Bacon@BACON ~/git (master)
$
```

Initialise a repo

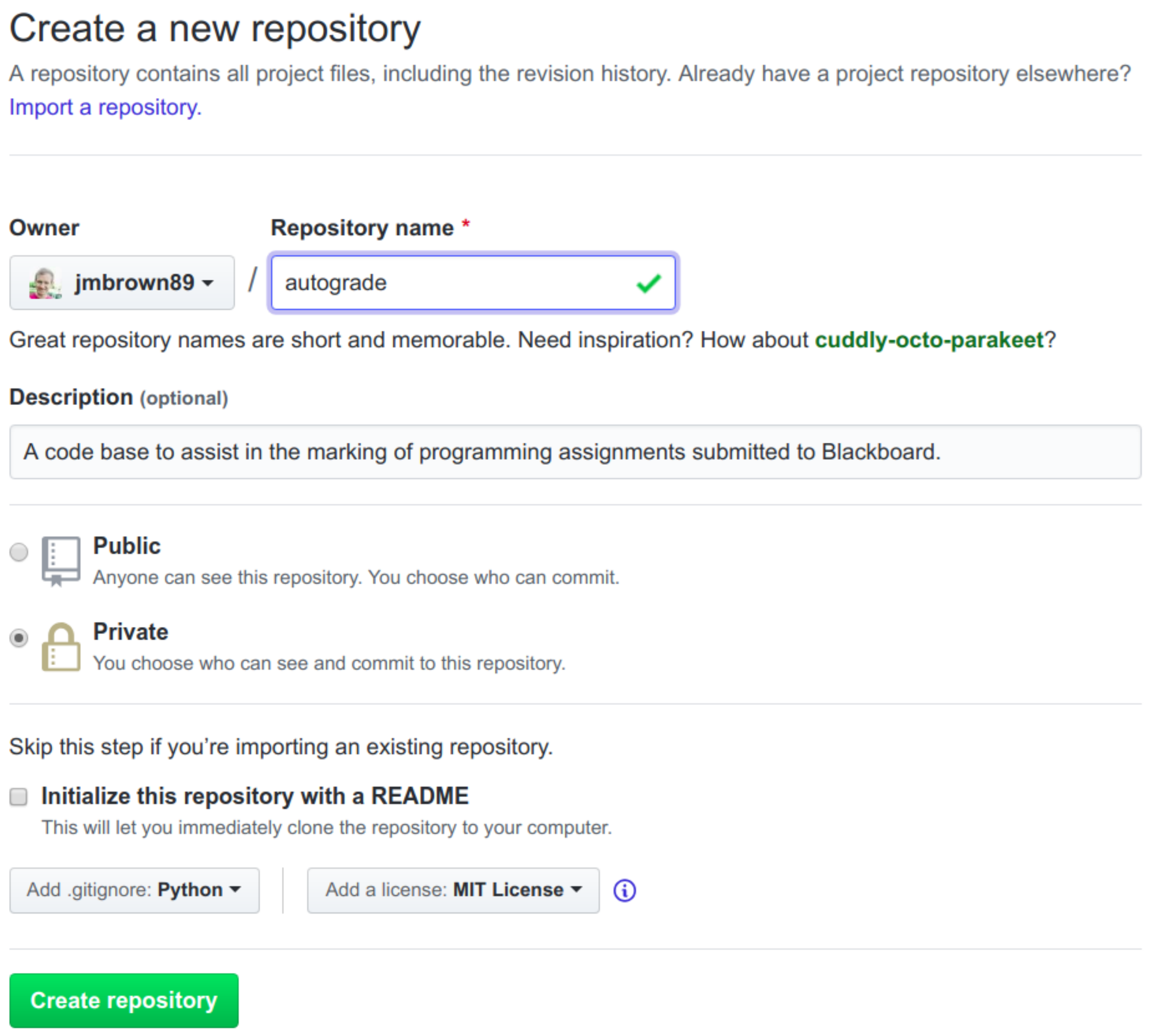
- The `git init` command is used to create an “empty” repository
- You can do this in any empty folder, or in an existing (unversioned) project
- All this does is create a hidden folder called `.git` **– do not touch this!**
- This folder contains all the versioning information for your project



.git
.gitignore
LICENSE
mark_submissions
README.md
rename_submissions
requirements
tasks

Using Github

- GitHub provides free private repositories you can use, plus some other nice tools
- You don’t have to; git itself is independent of the host
- Create an account on Github, and go to https://github.com/new



Create a new repository

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

Owner: jmbrown89 / Repository name *: autograde ✓

Great repository names are short and memorable. Need inspiration? How about cuddly-octo-parakeet?

Description (optional): A code base to assist in the marking of programming assignments submitted to Blackboard.

Public — Anyone can see this repository. You choose who can commit.

[✓ Private] — You choose who can see and commit to this repository.

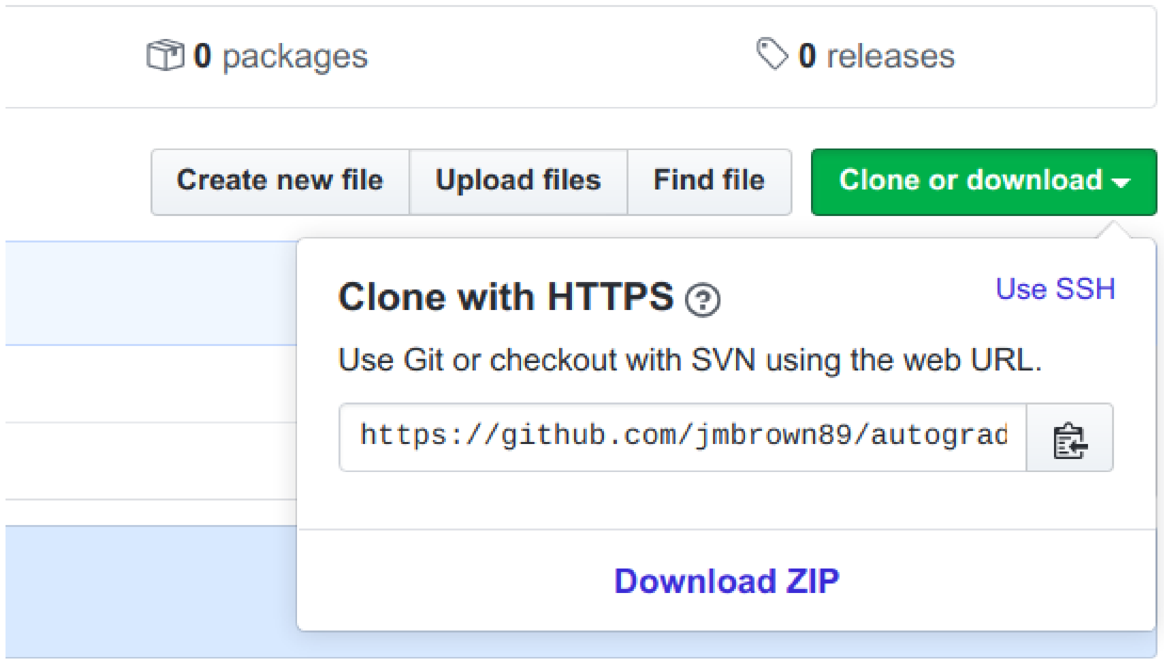
Skip this step if you’re importing an existing repository.

Initialize this repository with a README — This will let you immediately clone the repository to your computer.

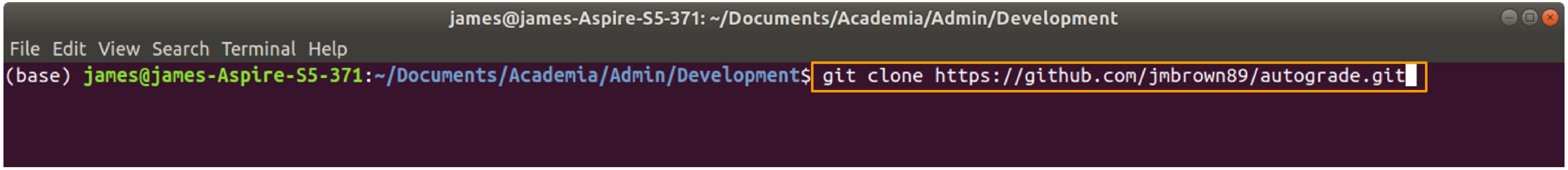
Add .gitignore: Python | Add a license: MIT License

Cloning your repository

- When you create a repo, the version on GitHub is the “remote”
- When you clone a repo, you are creating a local copy of it
- To clone, first copy the web URL and type: `git clone <url>`

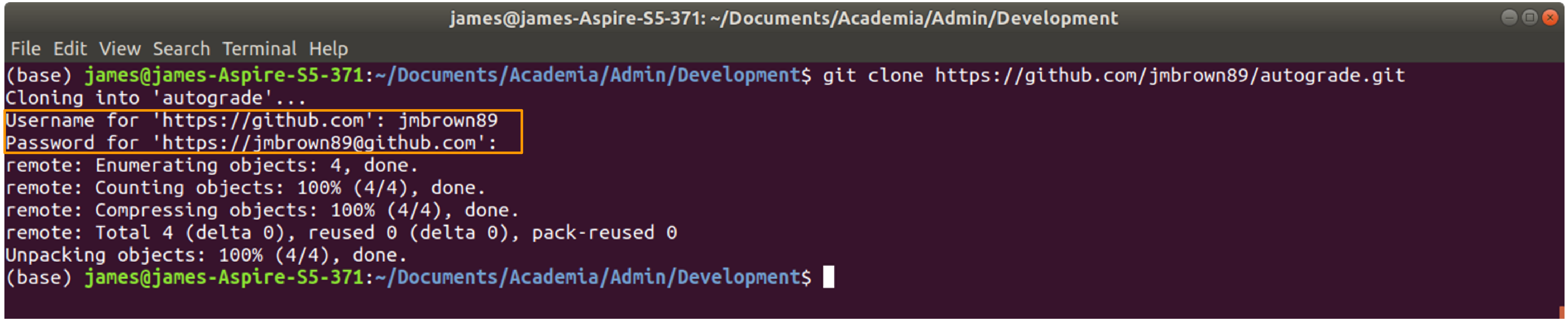

0 packages
0 releases
Create new file
Upload files
Find file
Clone or download
Clone with HTTPS
Use SSH
Use Git or checkout with SVN using the web URL.
```
https://github.com/jmbrown89/autograd
```
Download ZIP

My local repository



james@james-Aspire-S5-371: ~/Documents/Academia/Admin/Development
File Edit View Search Terminal Help

```
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development$ git clone https://github.com/jmbrown89/autograde.git
```



james@james-Aspire-S5-371: ~/Documents/Academia/Admin/Development
File Edit View Search Terminal Help

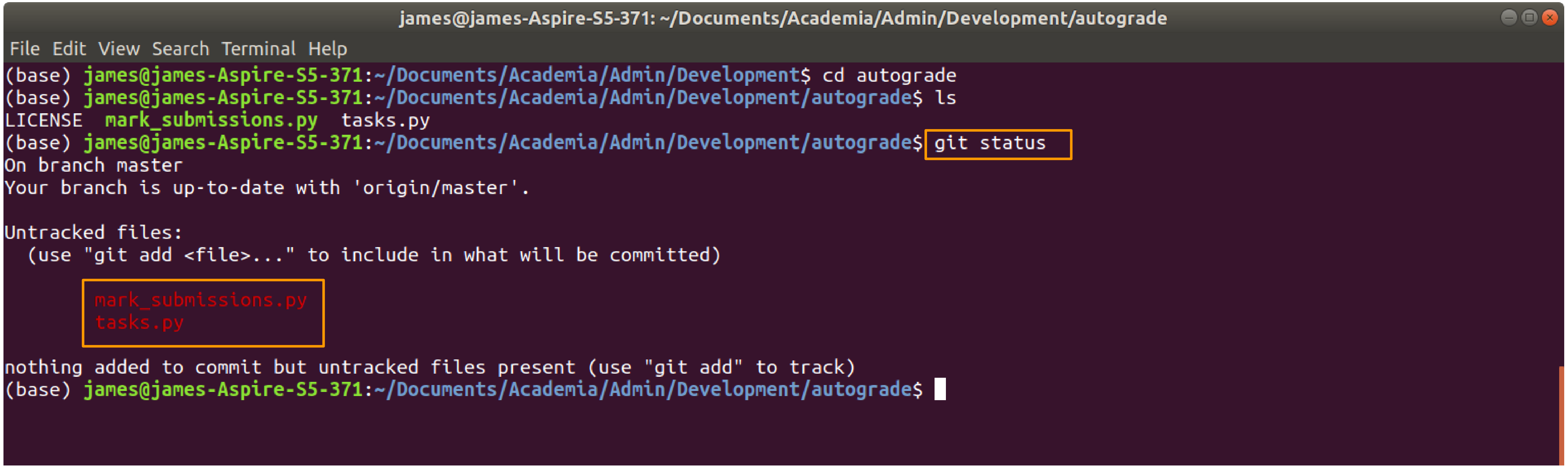
```
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development$ git clone https://github.com/jmbrown89/autograde.git
Cloning into 'autograde'...
Username for 'https://github.com': jmbrown89
Password for 'https://jmbrown89@github.com':
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development$
```

Note: login needs to be via ssh keys nowadays – see later slide for information on setting this up

Making contributions

- Let’s say you’ve added some code, how do you version it?
    1. Stage (add) which files you wish to version
    2. Commit those changes in your local repo
    3. Update your remote repo to reflect your local
- Each steps involves a different command:
    1. `git add <various-options>`
    2. `git commit <various-options>`
    3. `git push <various-options>`

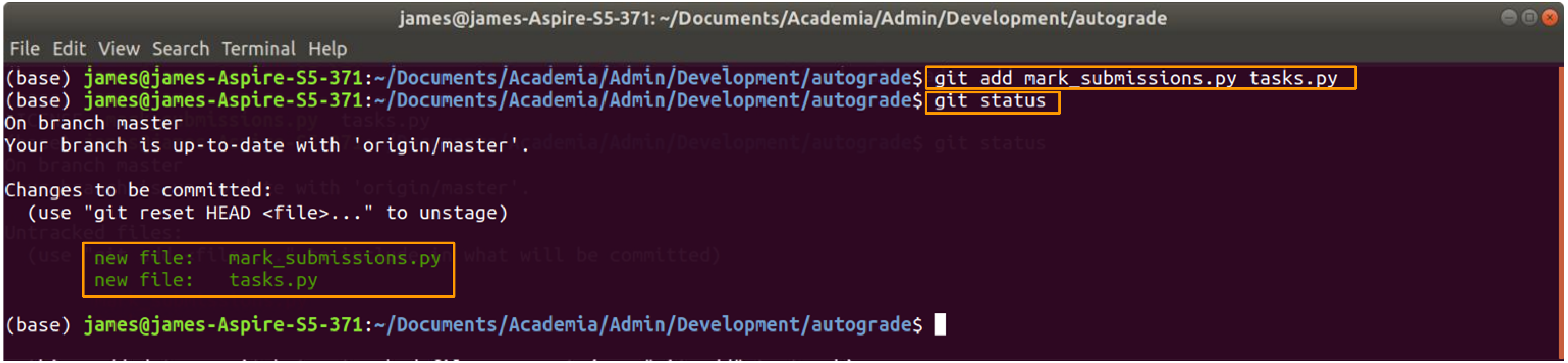
Checking what's been changed


james@james-Aspire-S5-371: ~/Documents/Academia/Admin/Development/autograde
File Edit View Search Terminal Help

```
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development$ cd autograde
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$ ls
LICENSE  mark_submissions.py  tasks.py
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

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

        mark_submissions.py
        tasks.py

nothing added to commit but untracked files present (use "git add" to track)
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$
```

Use `git status` to check the current state of your repo's files

Staging files

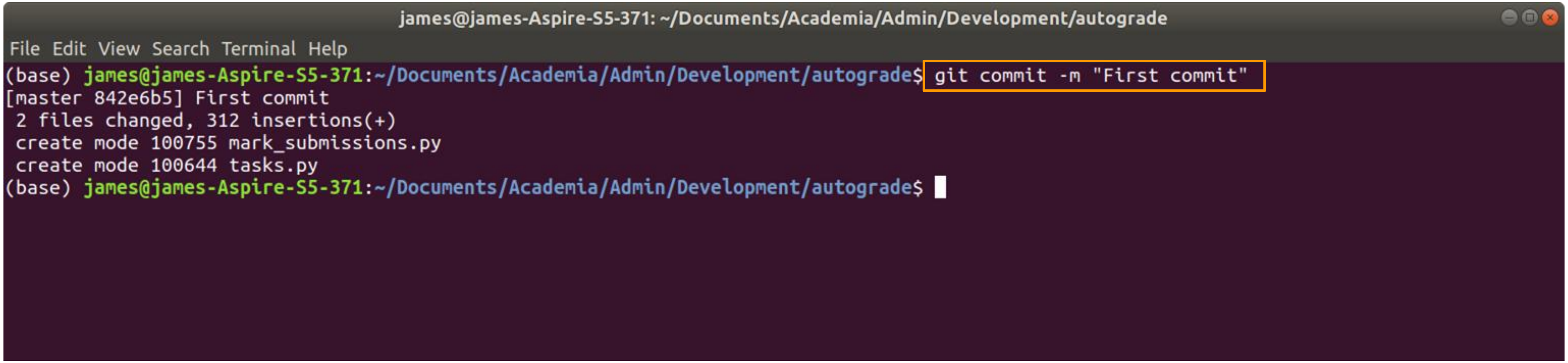

james@james-Aspire-S5-371: ~/Documents/Academia/Admin/Development/autograde
File Edit View Search Terminal Help

```
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$ git add mark_submissions.py tasks.py
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

	new file:   mark_submissions.py
	new file:   tasks.py

(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$
```

Use `git add <file1> <file2>` (or `git add -A` to stage all changes)

You can run `git status` to see them change from red to green

Staging

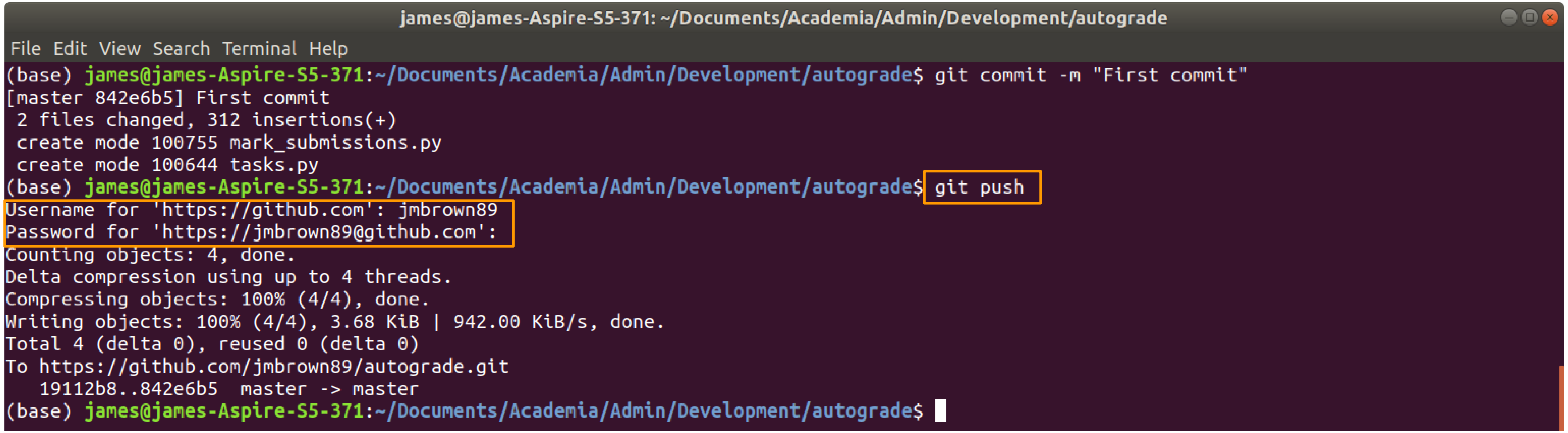

james@james-Aspire-S5-371: ~/Documents/Academia/Admin/Development/autograde
File Edit View Search Terminal Help

```
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$ git commit -m "First commit"
[master 842e6b5] First commit
 2 files changed, 312 insertions(+)
 create mode 100755 mark_submissions.py
 create mode 100644 tasks.py
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$
```

Use `git commit -m “<message>”` to make your commitment

Write a meaningful message to state what you’ve been working on!

Pushing to remote


james@james-Aspire-S5-371: ~/Documents/Academia/Admin/Development/autograde
File Edit View Search Terminal Help

```
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$ git commit -m "First commit"
[master 842e6b5] First commit
 2 files changed, 312 insertions(+)
 create mode 100755 mark_submissions.py
 create mode 100644 tasks.py
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$ git push
Username for 'https://github.com': jmbrown89
Password for 'https://jmbrown89@github.com':
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 3.68 KiB | 942.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/jmbrown89/autograde.git
   19112b8..842e6b5  master -> master
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$
```

Use `git push` to upload your code to the remote repository

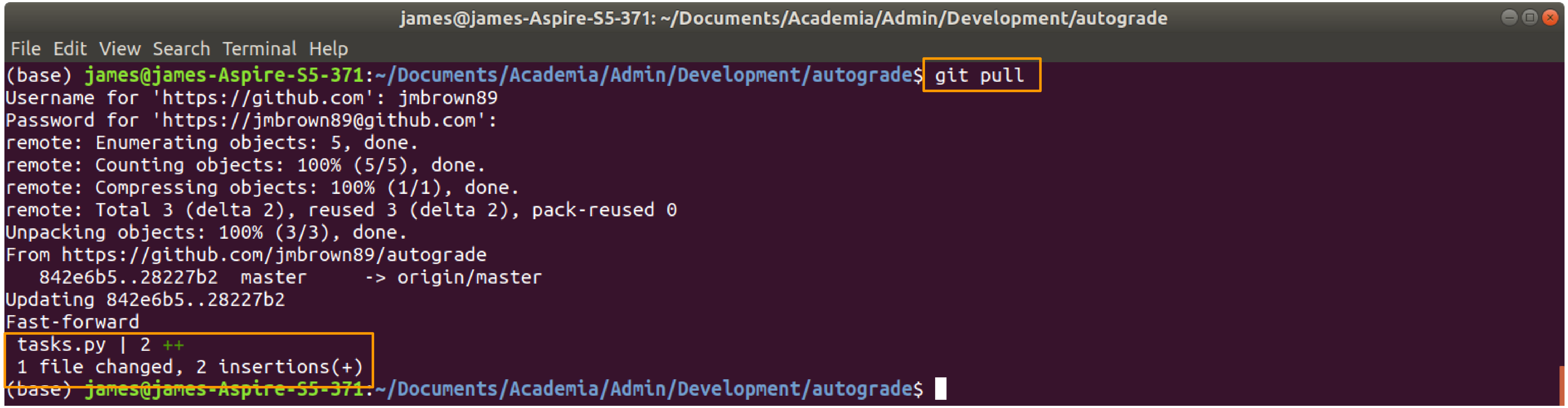
Good and bad practice

- It is probably good practice to do things *like*
  - Making commits per individual feature (with meaningful messages)
  - Writing tests to verify that a feature works before pushing it to remote
  - Only push a few times a day (besides “hotfixes”)
- It is probably bad practice to do things *like*
  - Make one big commit that combines several *major* feature changes
  - Pushing every minor change, rather than just committing locally
  - Making untested changes that break everything

Bringing your code up-to-date

- If someone else makes a change (or you want to work from a different machine, say) you can pull the latest remote repo to your local
- Some things to keep in mind:
    - If you have local unstaged changes then it *might* fail
        - git won’t overwrite your local changes unless you force it
        - there are commands to do so - but be **very** wary of this!
    - Your repo may end up “ahead” if you have unpushed commits
        - You can push your stuff once remote is pulled

Pulling from remote


james@james-Aspire-S5-371: ~/Documents/Academia/Admin/Development/autograde
File Edit View Search Terminal Help

```
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$ git pull
Username for 'https://github.com': jmbrown89
Password for 'https://jmbrown89@github.com':
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 2), reused 3 (delta 2), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/jmbrown89/autograde
   842e6b5..28227b2  master     -> origin/master
Updating 842e6b5..28227b2
Fast-forward
 tasks.py | 2 ++
 1 file changed, 2 insertions(+)
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$ 
```

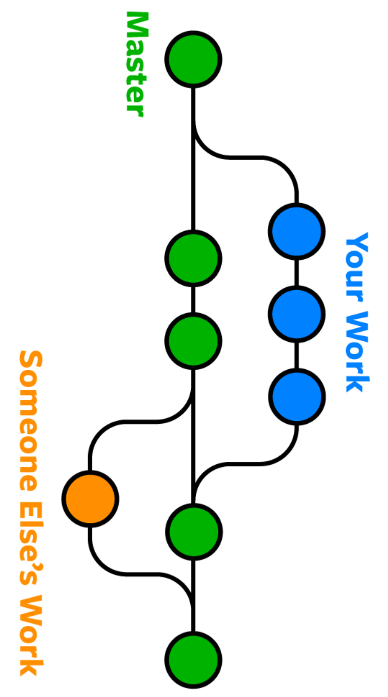
`git` indicates that there have been two lines of code added in `tasks.py`

Avoiding conflicts

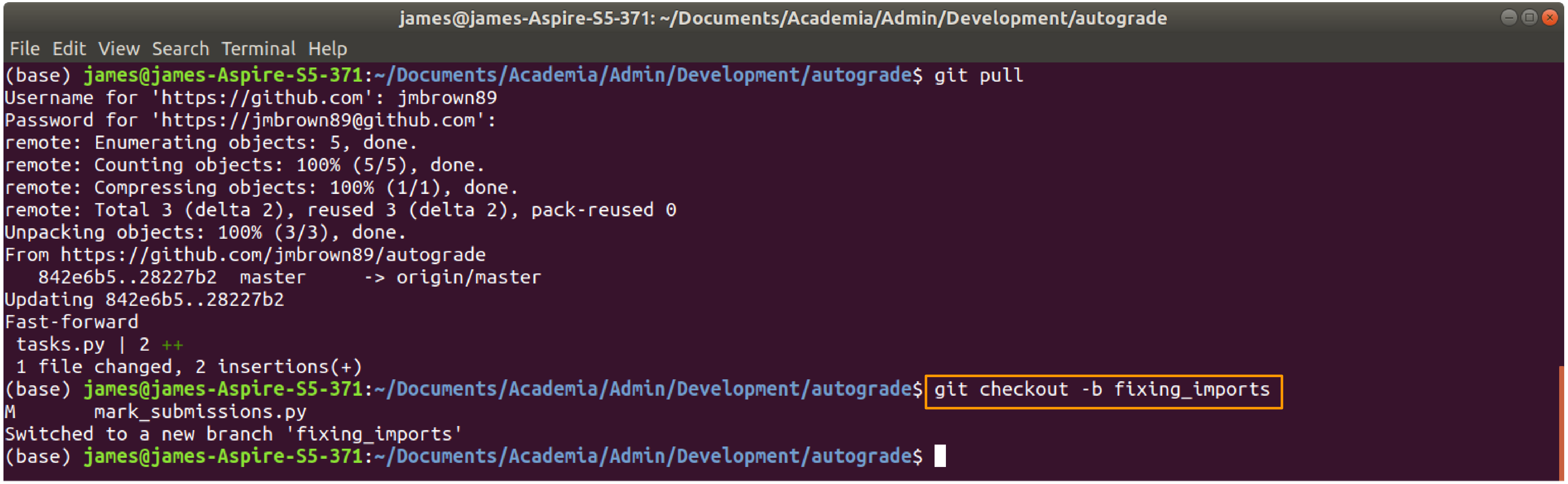
- When working as a team, make it clear who is working on what
  - If you can avoid working on the same files, it helps manage the repo
  - As you grow more confident as a team, it will get easier!
- Another way of avoiding conflicts with each other/yourself is via **branches**
- The default (and only) branch when you create a `git` repo is master
  - The `master` branch should always have your “production code”
  - Try to avoid doing anything too experimental here, it should always *work*

Creating branches

- Let’s say you’re looking to develop a new feature. The master branch has:
    - A main script used to run the software from the command line
    - Several modules with various parts that run the backend
- Here’s what you need to consider:
    - Does my new feature use existing code/files?
    - Will I need to modify other bits of code for it to work?
    - Is there a risk of it breaking other parts of the software?
- If the answer to any of the above is ‘yes’, create a new branch



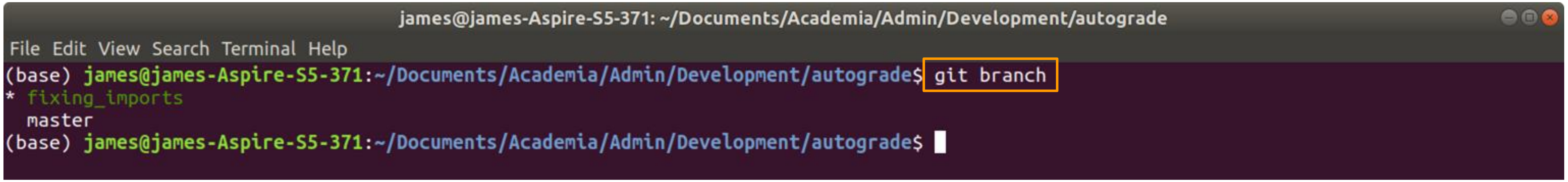
Creating branches


james@james-Aspire-S5-371: ~/Documents/Academia/Admin/Development/autograde
File Edit View Search Terminal Help

```
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$ git pull
Username for 'https://github.com': jmbrown89
Password for 'https://jmbrown89@github.com':
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 2), reused 3 (delta 2), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/jmbrown89/autograde
   842e6b5..28227b2  master     -> origin/master
Updating 842e6b5..28227b2
Fast-forward
 tasks.py | 2 ++
 1 file changed, 2 insertions(+)
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$ git checkout -b fixing_imports
M       mark_submissions.py
Switched to a new branch 'fixing_imports'
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$ 
```

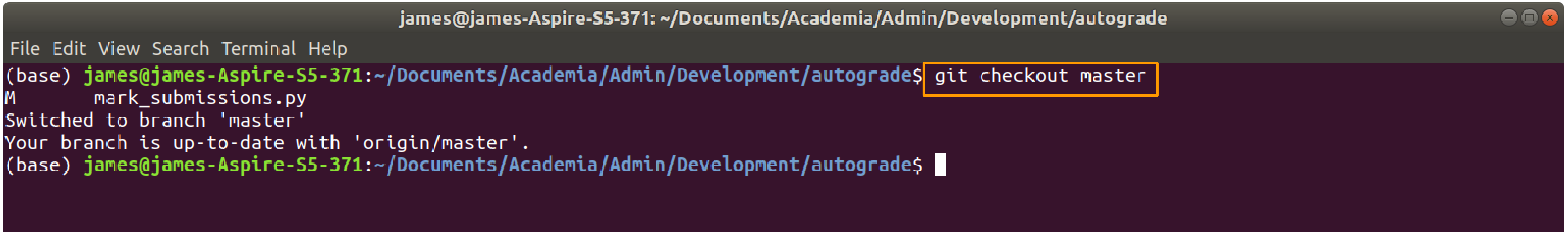
Use `git checkout -b <branch_name>` to create a local branch and automatically start using it (it will switch you automatically)

Creating branches


james@james-Aspire-S5-371: ~/Documents/Academia/Admin/Development/autograde
File Edit View Search Terminal Help

```
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$ git branch
* fixing_imports
  master
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$ 
```

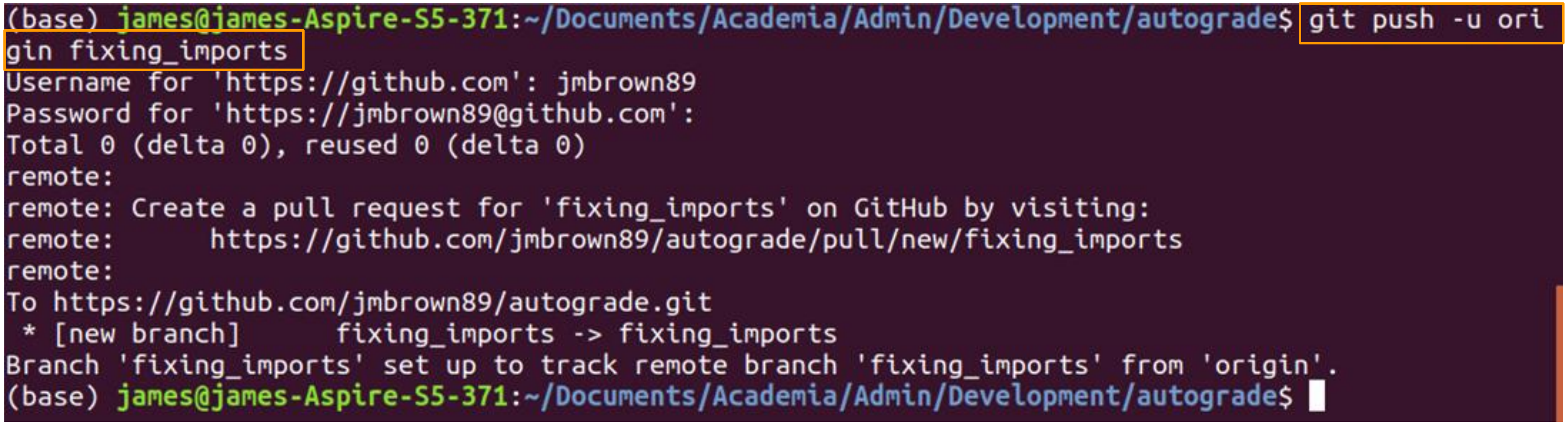
Use `git branch` to check which branches you have and are currently on


james@james-Aspire-S5-371: ~/Documents/Academia/Admin/Development/autograde
File Edit View Search Terminal Help

```
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$ git checkout master
M       mark_submissions.py
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$ 
```

Use `git checkout <branch_name>` to switch between existing branches

Creating branches



```
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$ git push -u ori
gin fixing_imports
Username for 'https://github.com': jmbrown89
Password for 'https://jmbrown89@github.com':
Total 0 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'fixing_imports' on GitHub by visiting:
remote:      https://github.com/jmbrown89/autograde/pull/new/fixing_imports
remote:
To https://github.com/jmbrown89/autograde.git
 * [new branch]      fixing_imports -> fixing_imports
Branch 'fixing_imports' set up to track remote branch 'fixing_imports' from 'origin'.
(base) james@james-Aspire-S5-371:~/Documents/Academia/Admin/Development/autograde$ 
```

Make sure you **explicitly** push new branches - not done by default!

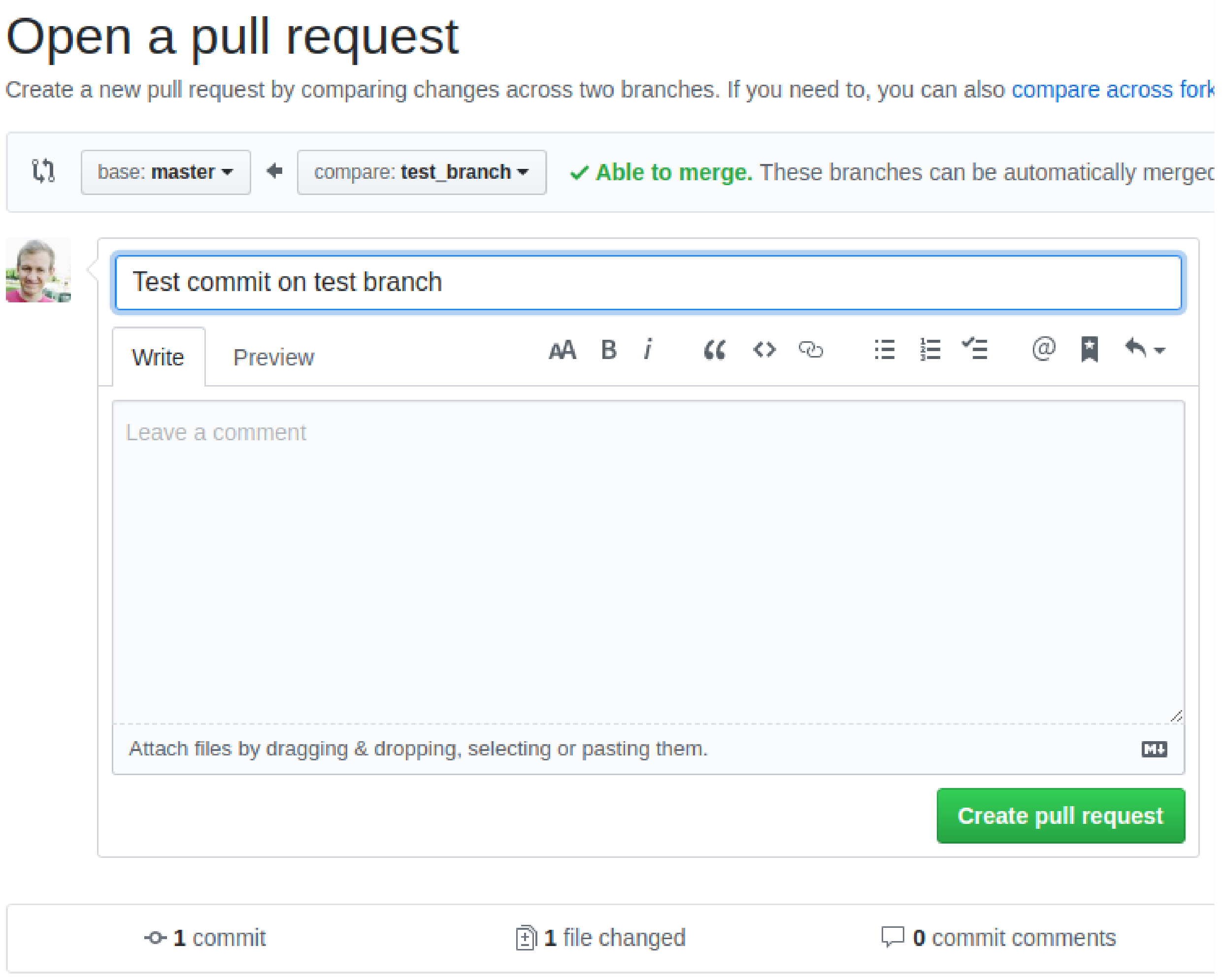
`git push -u origin <branch_name>`

Being careful with branches

- Be sure to check which branch you’re on before starting any work
  - You can always commit what you’re working on to “save it” and then switch to another branch to work on something else
  - It takes careful management which only comes through experience
- Check that everyone is happy before you incorporate a branch into master
  - Can be done and discussed via “pull requests” in GitHub
  - You can discuss these at your meetings - always a good idea

Pull requests (Github)

- Open pull requests are used to discuss proposed merging of branches prior to doing so
- You can keep making commits to open pull requests order to resolve potential conflicts



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across fork

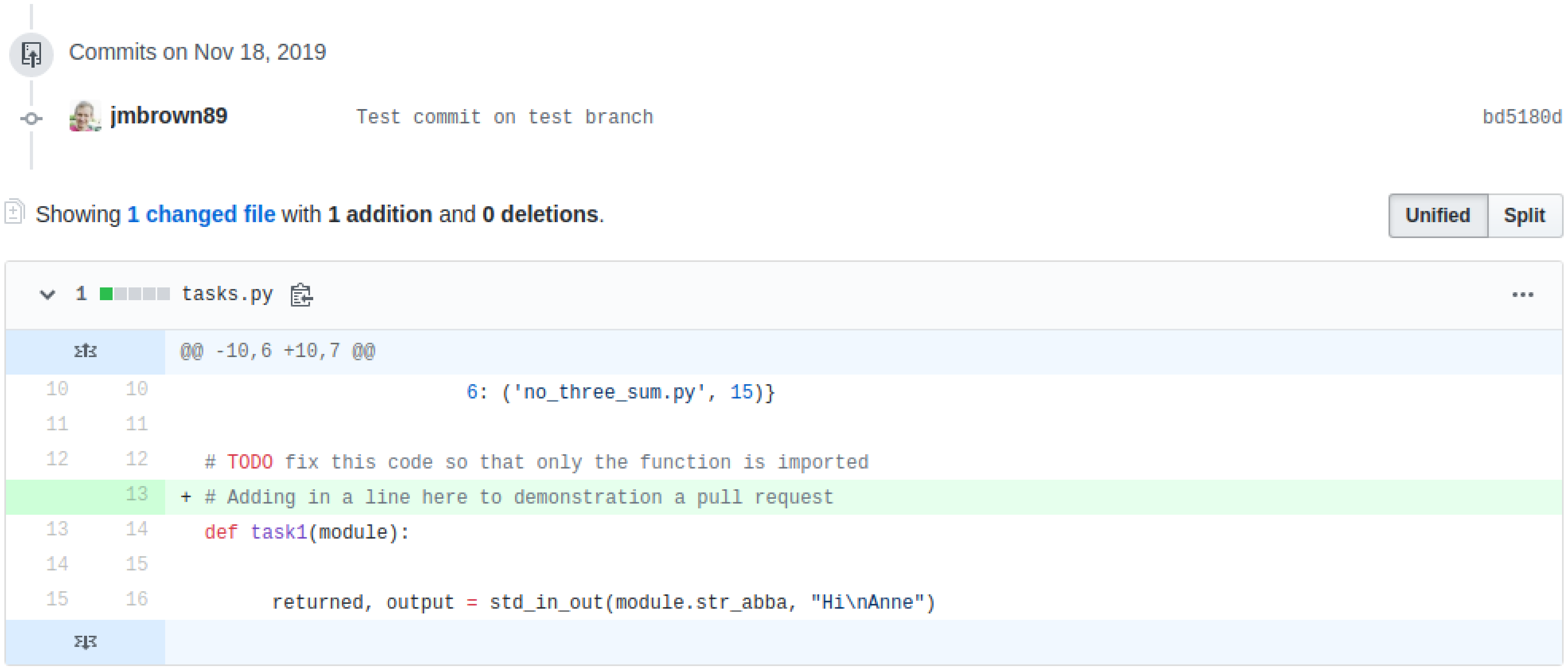
base: master ← compare: test_branch ✓ Able to merge. These branches can be automatically mergec

Test commit on test branch

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

1 commit 1 file changed 0 commit comments

Pull requests


Commits on Nov 18, 2019

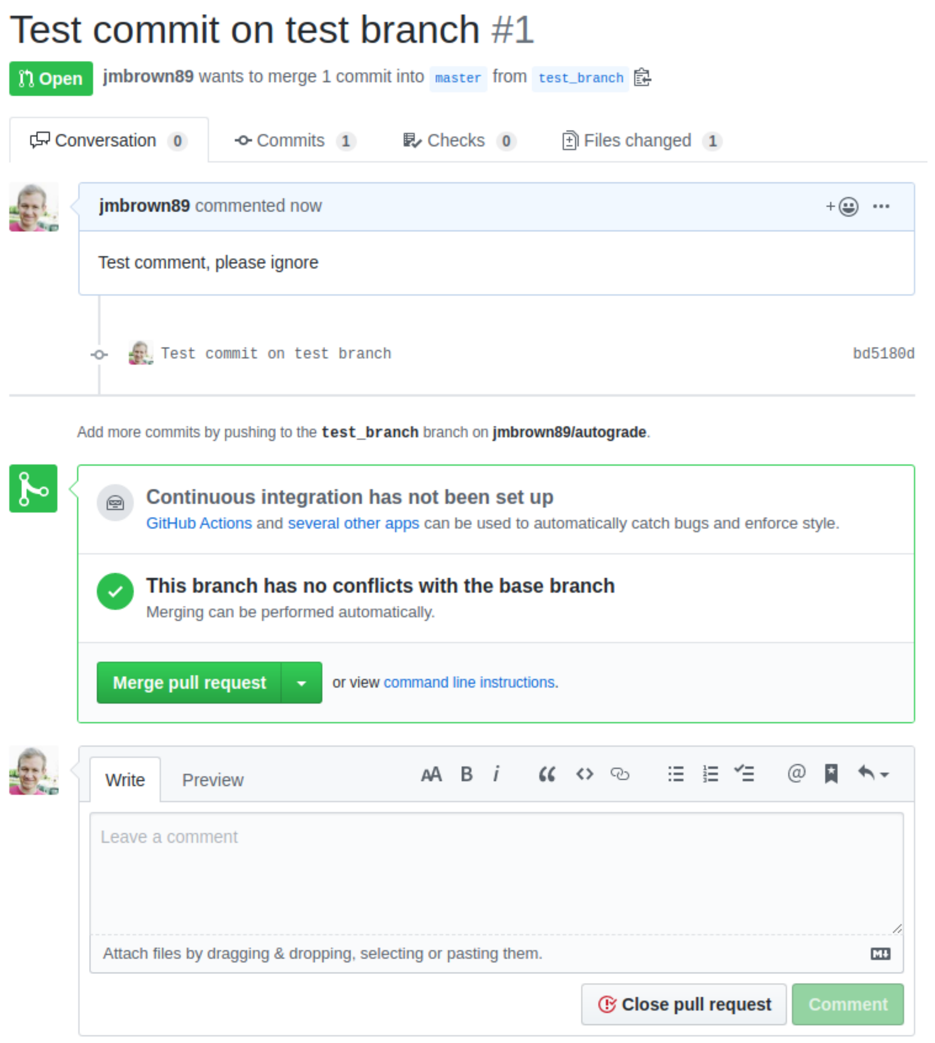
jmbrown89 Test commit on test branch bd5180d

Showing **1 changed file** with **1 addition** and **0 deletions**.

```
1 tasks.py
@@ -10,6 +10,7 @@
                         6: ('no_three_sum.py', 15)}

  # TODO fix this code so that only the function is imported
+ # Adding in a line here to demonstration a pull request
  def task1(module):

        returned, output = std_in_out(module.str_abba, "Hi\nAnne")
```

Merging

- Once everyone is happy, you can merge the pull request
- This will incorporate the “feature branch” into the master branch
- This is generally safer than just using git merge - conflict resolution can be a nightmare!



Test commit on test branch #1

Open jmbrown89 wants to merge 1 commit into `master` from `test_branch`

jmbrown89 commented now

Test comment, please ignore

```
Test commit on test branch                                   bd5180d
```

Add more commits by pushing to the **test_branch** branch on **jmbrown89/autograde**.

Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request
or view command line instructions.
Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting or pasting them.
Close pull request
Comment

Getting to grips with git

- Practice, practice, practice!
  - Try things out with a “fake” repository, there’s no risk involved!
  - Keep in mind it’s actually quite hard to ruin a repo beyond repair
- Rule: *never* use a command unless you know what it’s doing
  - You can mostly get away with just knowing the basics, though
- If all else fails, use a cheat sheet:

  https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf

Stuff you might want to study yourself

- `.gitignore` files
  - Super useful if you want to avoid accidentally versioning certain file types
- Some other useful commands:
  - `git merge` – merge a development branch into your master branch
  - `git log` - show version history for your current branch
  - `git diff` - visualise differences between old/new version of code
  - `git config` - avoid having to type your username/password all the time
  - `git stash` - temporarily shelf your changes to work on something else
  - ssh keys – necessary nowadays for authentication (link below)

https://help.github.com/en/enterprise/2.17/user/authenticating-to-github/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent

END

MALKHAFAJIY@LINCOLN.AC.UK