

Git your **** together

A primer in modern version control







Tim Smith
APA Seminar



What is git?

Git is a distributed version control system that keeps track of snapshots of your code

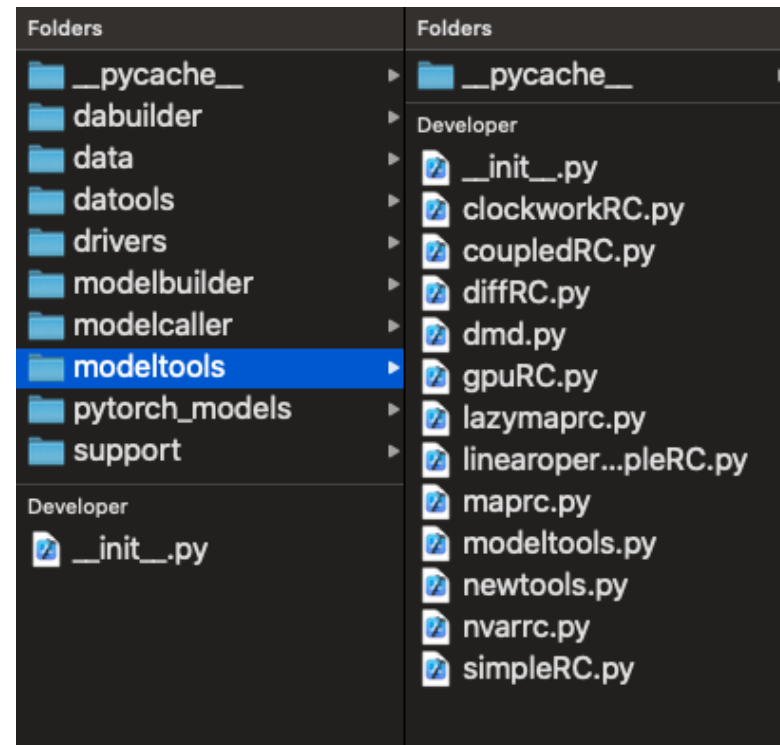
- Manage updates to your files (code, writing, etc)
- Keep track of changes across many machines
- Use and modify existing code
- Contribute to existing projects

Name	
	mycode.py
	mycode_fixed.py
	mycode_fixed_faster.py
	mycode_fixed_faster_final.py
	mycode_fixed_faster_final_really-this-time.py
	mycode_mar31_2022.py

What is git?

Repository (repo) is a collection of files for a single project

- On your computer, a git repo looks familiar: it's just a bunch of files
- But, by running specific git commands we can modify and compare changes with a “main” version



```
@@ -483,7 +483,7 @@ def _genpred(u, mask_in=[slice(None)],
    Win = Win[:, mask_in]

    uT = u.T
-    rT = np.zeros(shape=(spinup_steps, reservoir_dimension))
+    rT = np.ones(shape=(spinup_steps, reservoir_dimension))

    # Generate reservoir history
    for n in range(1, spinup_steps):
```

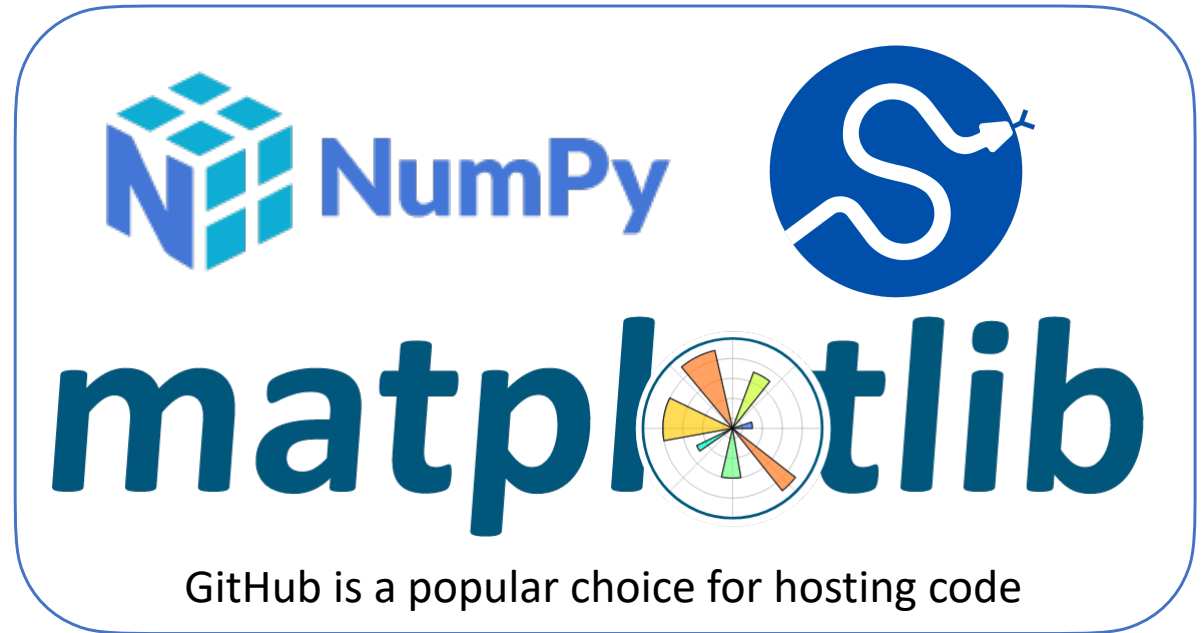
Running `git diff` shows that now the variable `rT` is created with `np.ones`, instead of `np.zeros`

GitHub

GitHub stores your files and provides a web interface that integrates with git functionality

Why use GitHub?

1. It's popular
2. It has nice features that allow for collaboration and discussion
3. It integrates well with automated features like code testing and package publishing



How do we use it?

- Git for personal use
- Git for collaborative use
- Special requests & discussion
 - Authentication
 - Filetype best practices
 - I'm all spread out
 - etc...



Git clone

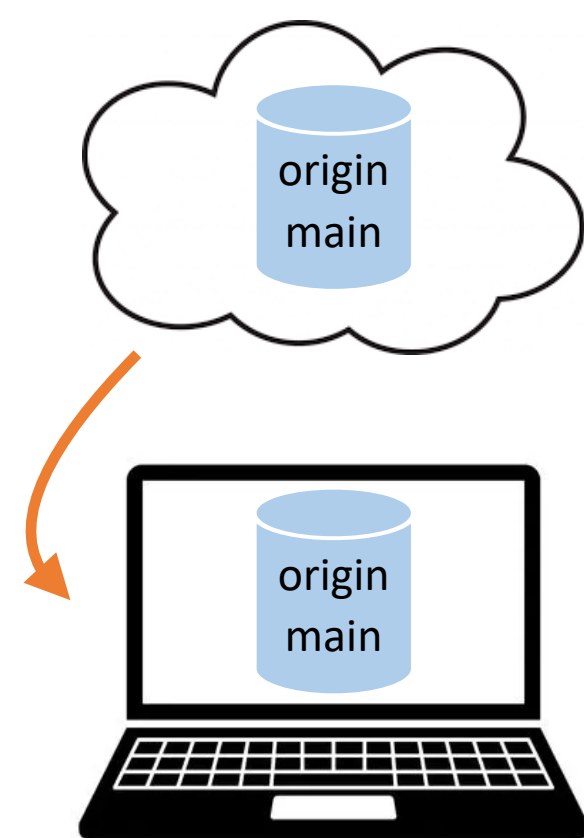
- Easiest to do by going to GitHub.com, and cloning from there

github.com/timothyas/bash-envy

git clone

The screenshot shows the GitHub repository page for 'timothyas/bash-envy'. The repository is public and has 1 branch (master) and 0 tags. The file list includes folders like 'conda', 'env', 'job-prep', 'optfiles', 'stylelib', and 'useful-cmd-line', as well as files like '.gitignore' and 'README.md'. A 'Clone' dropdown menu is open, showing options for cloning via HTTPS, SSH, or GitHub CLI. The SSH URL 'git@github.com:timothyas/bash-envy.git' is highlighted with a blue box. Other options include 'Open with GitHub Desktop' and 'Download ZIP'.

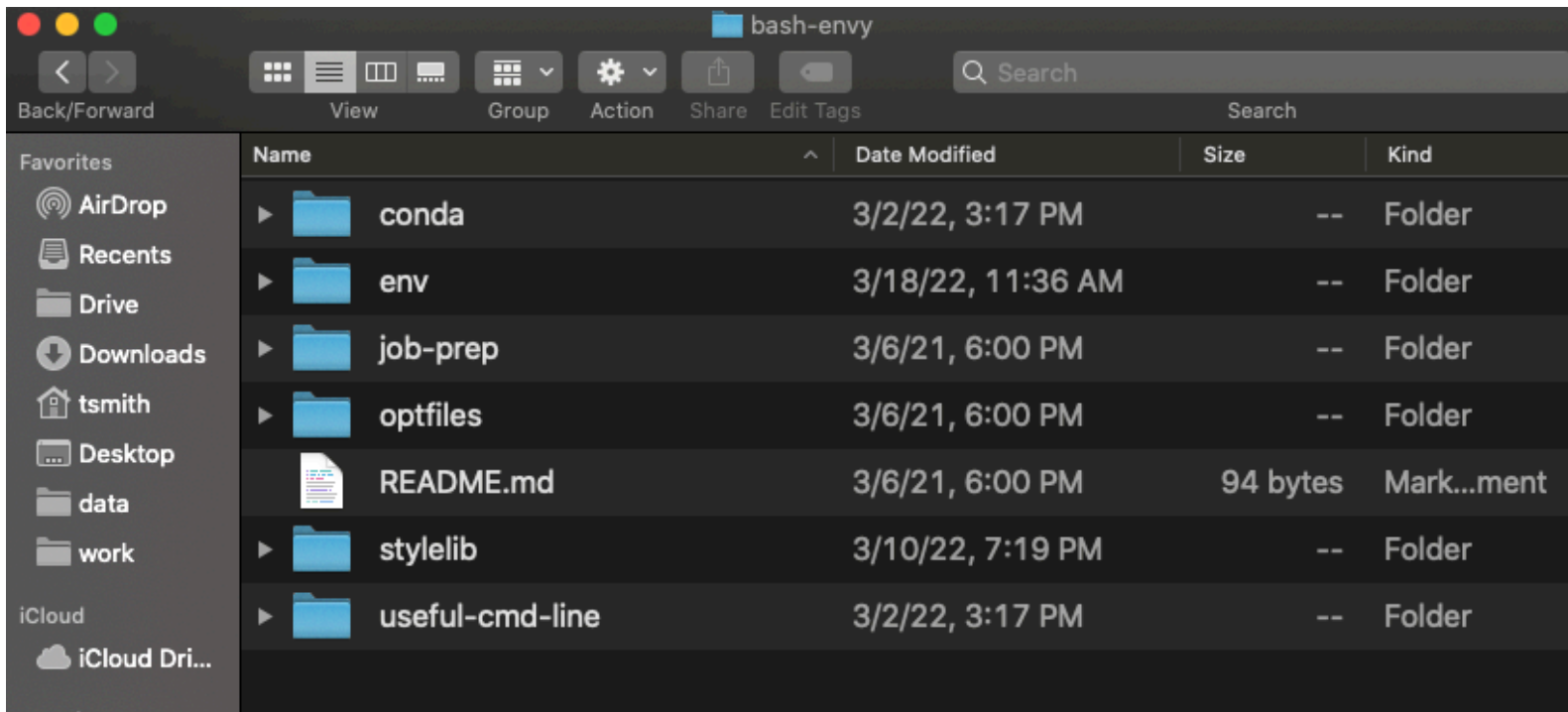
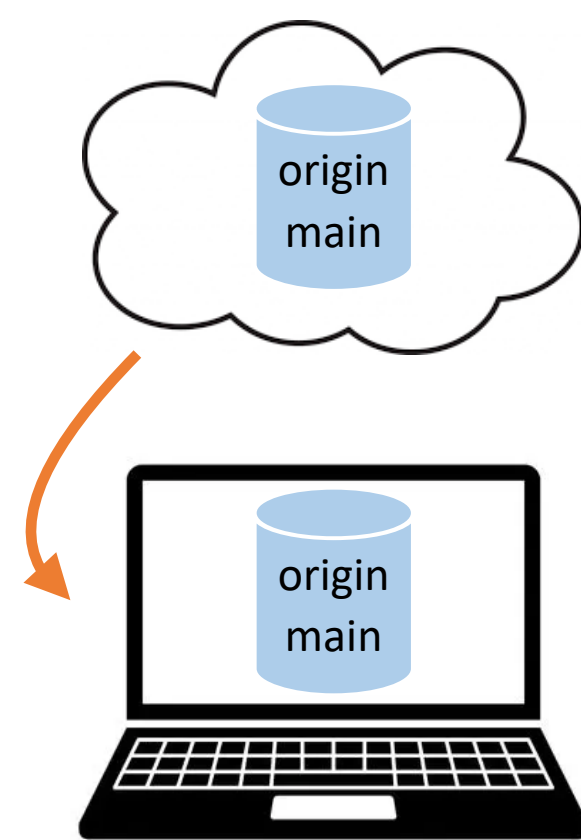
File/Folder	Description	Last Updated
conda	conda dir overhaul	
env	gcp stuff	
job-prep	rearranging	
optfiles	rearranging	
stylelib	small updates	
useful-cmd-line	bash script to enable jupyter vim binding	2 months ago
.gitignore	ignore netcdf	4 years ago
README.md	README.md created online with Bitbucket	4 years ago



Git clone

- Easiest to do by going to GitHub.com, and cloning from there

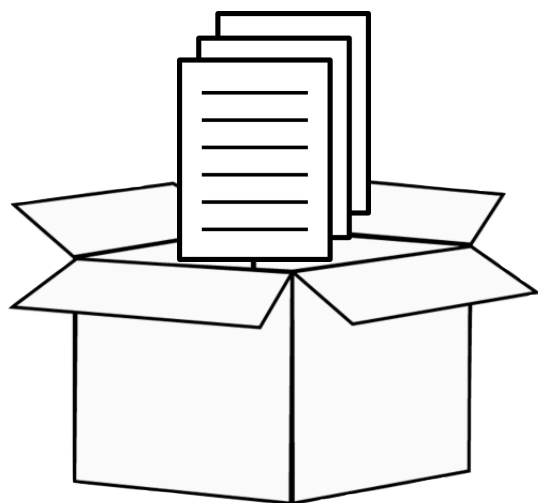
git clone



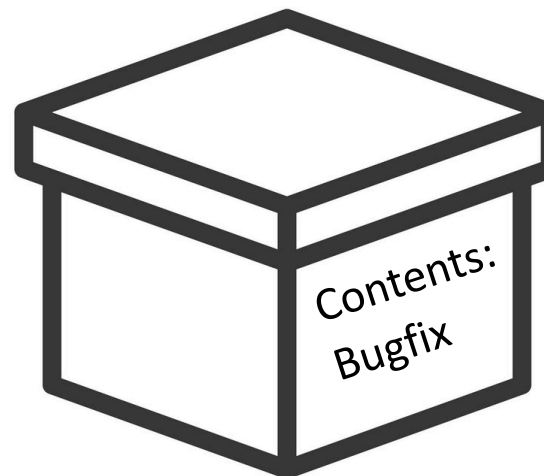
A generic workflow



Make edits



`git add`
Stage (prepare) file changes



`git commit`
Create and label a single
"commit" containing
logically related changes



`git push`
Push the changes to the
cloud

Git log

Note: Here I am showing the web view of commit history on one of my repos. But it is also useful to use this command on your computer, for instance to see where you are in a project's history, and e.g. what happened most recently

🔒 [timothyas / generic-matern-covariance](#) Private

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#) [Settings](#)

[main](#) [1 branch](#) [0 tags](#)

[Go to file](#)

[Add file](#)

[Code](#)



timothyas llc90 vertical correlations are a go

634d4ef 6 days ago

[🕒 16 commits](#)



figures

llc90 vertical correlations are a go

6 days ago

Click here to
See the log
online

Git log

Note: Here I am showing the web view of commit history on one of my repos. But it is also useful to use this command on your computer, for instance to see where you are in a project's history, and e.g. what happened most recently

The screenshot shows the GitHub web interface for a repository named 'timothyas / generic-matern-covariance'. The repository is private and has 1 watcher, 0 forks, and 0 stars. The 'Code' tab is selected. The current branch is 'main'. The commit history is filtered to show commits from March 25, 2022, and March 23, 2022. The most recent commit is 'llc90 vertical correlations are a go' by timothyas, committed 6 days ago, with hash 634d4ef. Other recent commits include 'open_smoothdataset that is super clean... but too many tasks to use f...' and 'minor cleanup', both committed 6 days ago. An older commit from March 23, 2022, is 'much more efficient to just read in eagerly... then to zstore we go', committed 8 days ago, with hash 562edfd.

timothyas / generic-matern-covariance Private Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Security Insights

main

Commits on Mar 25, 2022

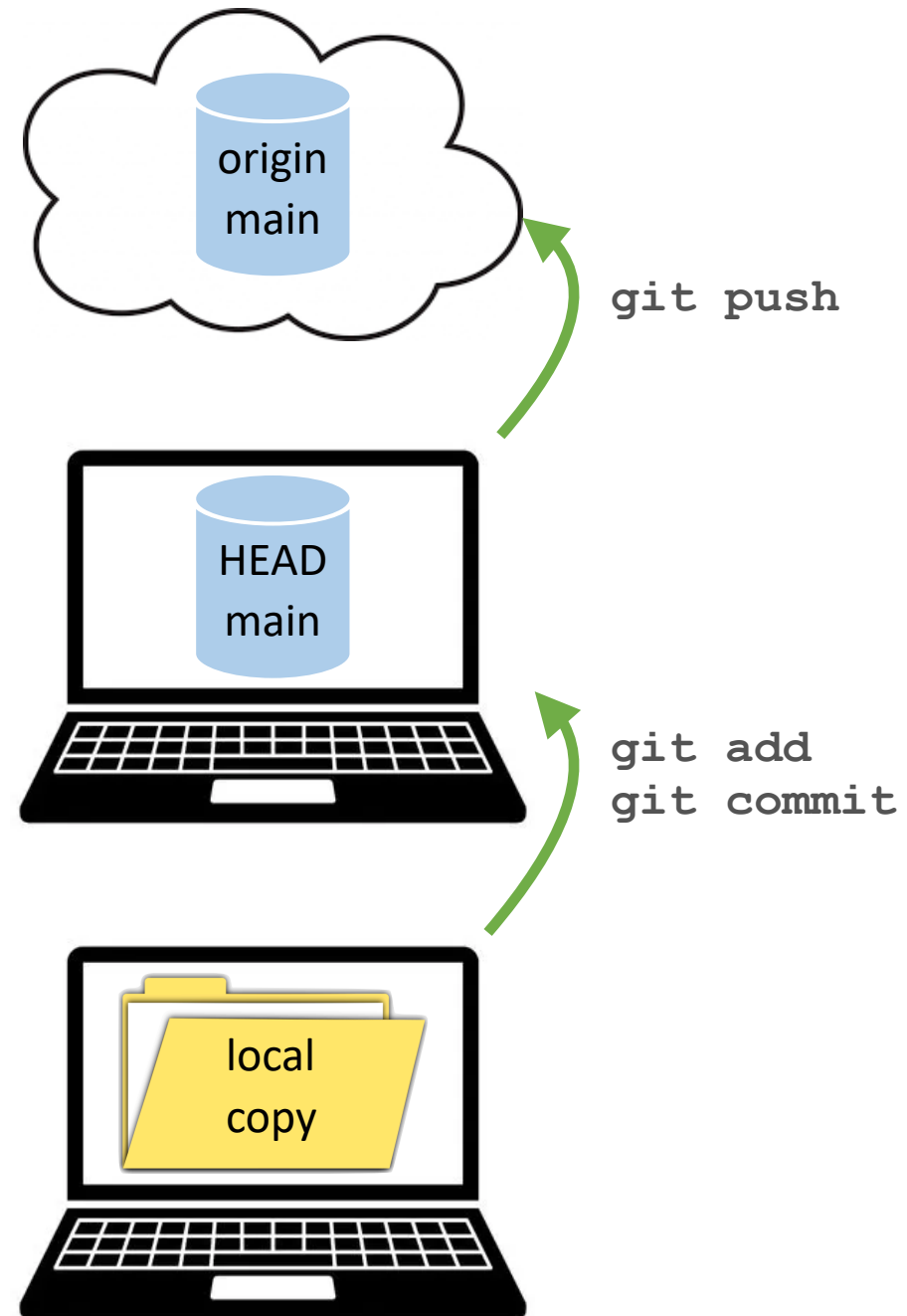
- llc90 vertical correlations are a go
timothyas committed 6 days ago 634d4ef
- open_smoothdataset that is super clean... but too many tasks to use f...
timothyas committed 6 days ago 7da666f
- minor cleanup
timothyas committed 6 days ago 6aeaaa8

Commits on Mar 23, 2022

- much more efficient to just read in eagerly... then to zstore we go
timothyas committed 8 days ago 562edfd

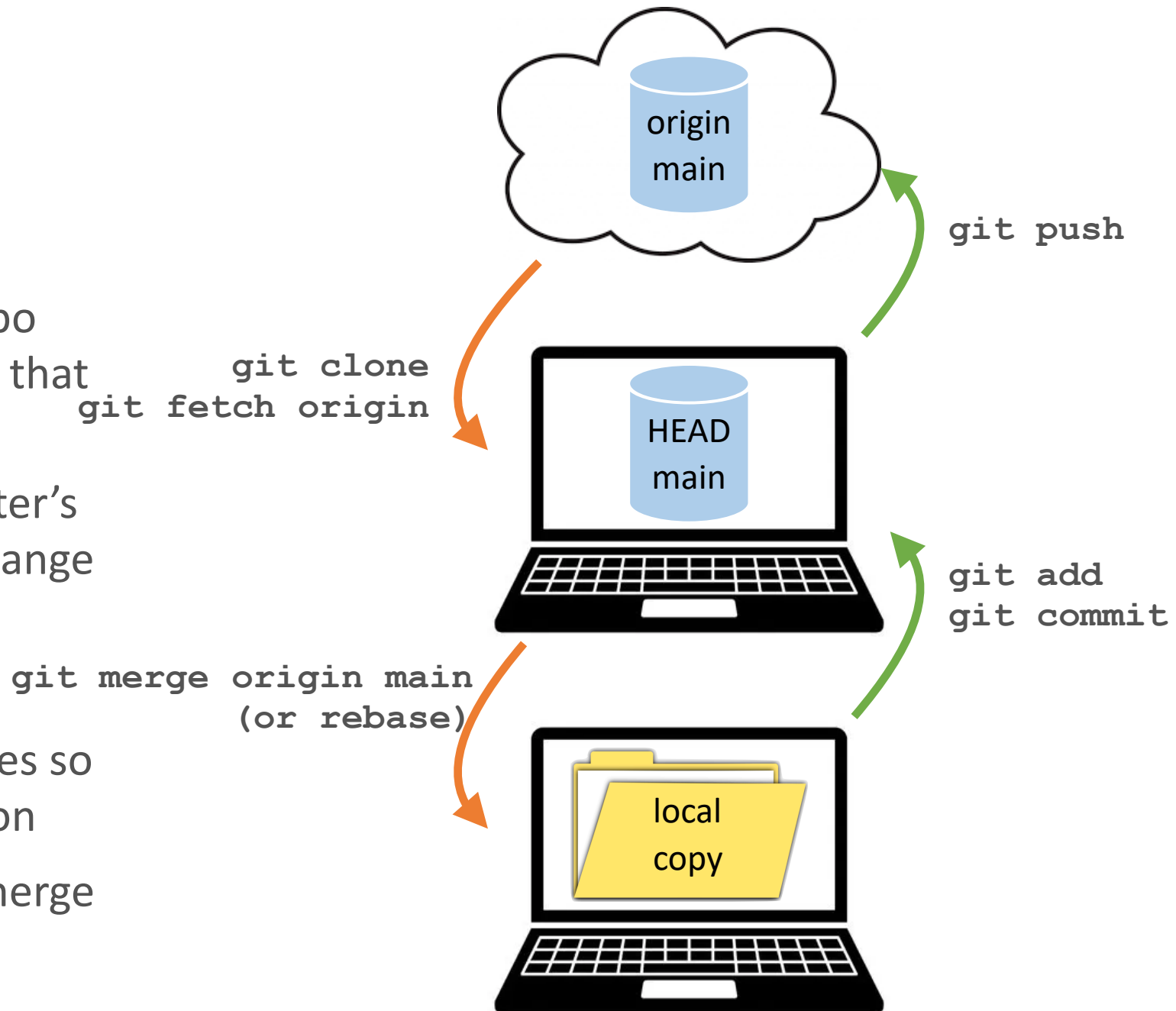
Git-ing personal

- Helpful to think of 3 “views”
 - The version in the cloud (on GitHub)
 - What your computer thinks is in the cloud
 - Your working copy, including any uncommitted changes
- **Origin** is a commonly used “remote” label, indicating your online version of the repo
- **HEAD** is the commit that you are currently working with on your computer
- **Main** is the branch name
- Note: **HEAD** is always the same term, **main** and **origin** are just commonly used names

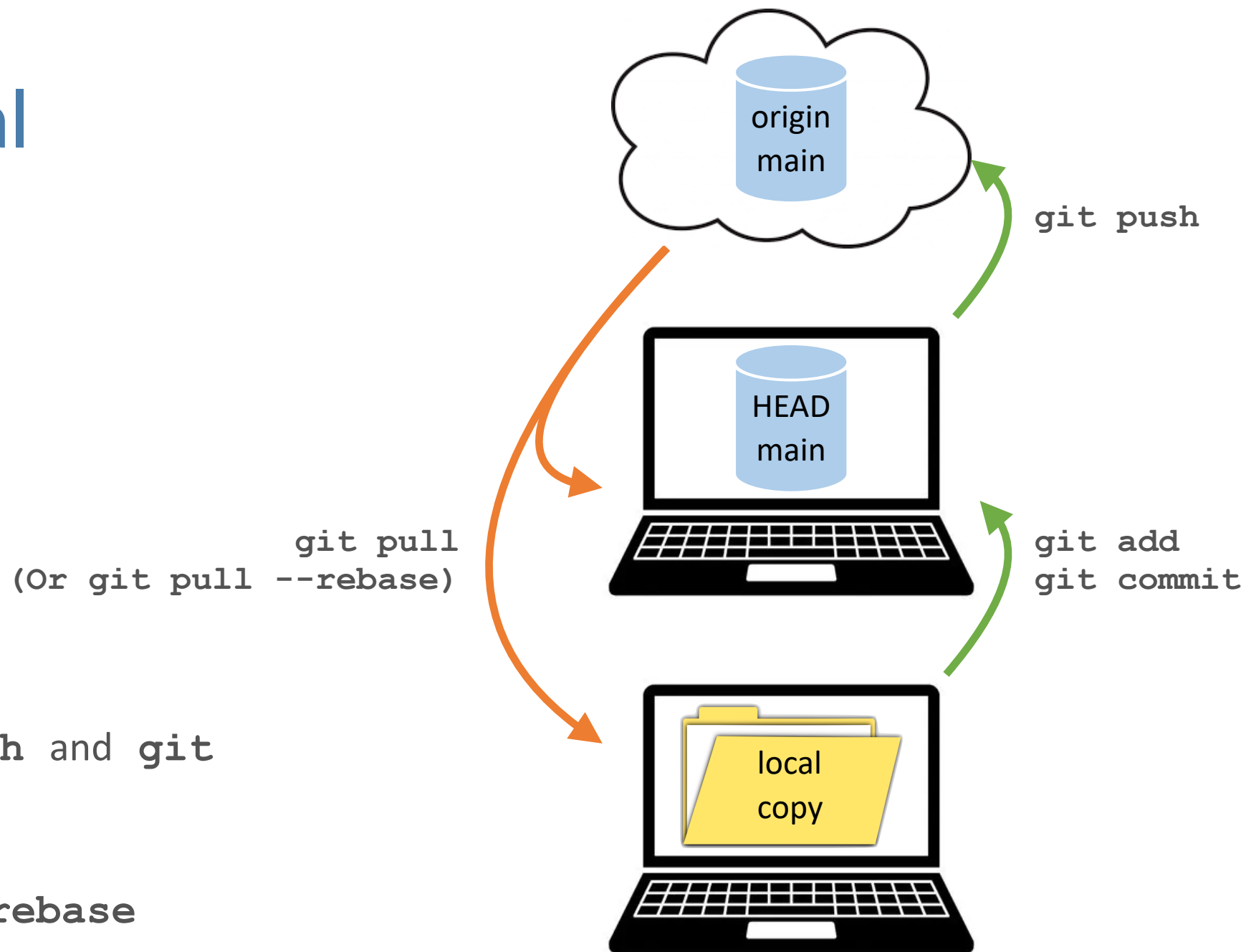


Git-ing personal

- **git clone**: get a copy of the repo on your machine, with special files that manage version control
- **git fetch**: update your computer's view of the cloud version (don't change the files!)
- **git merge**: update your local files so that they look like the online version
- **git rebase**: an alternative to merge



Git-ing personal



Shorthand

- `git pull`: run `git fetch` and `git merge` together
- `git pull --rebase`: run `git fetch` and `git rebase`

Git status

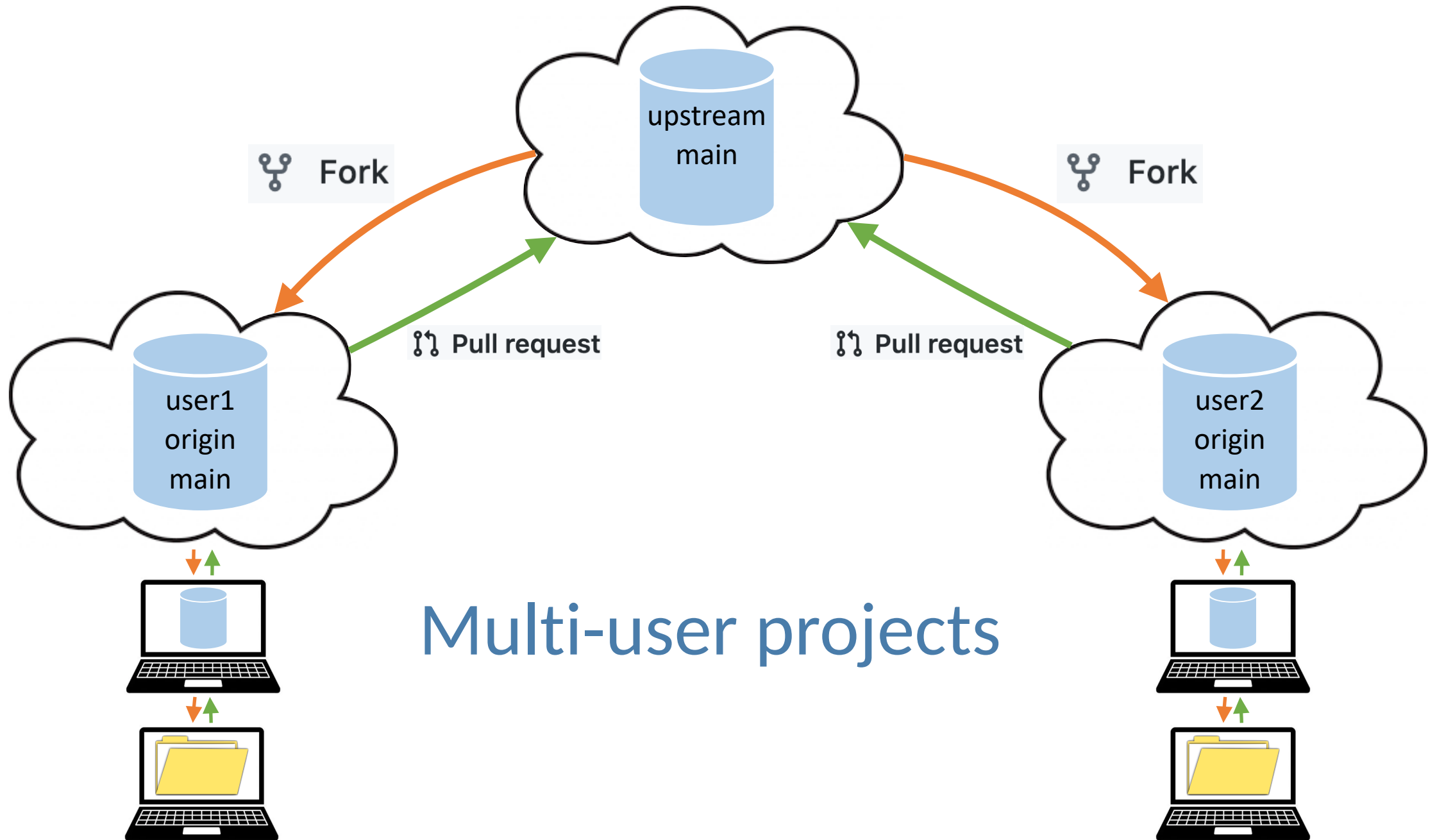
```
(base) [tim env]$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   bashrc_hera

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:   remote_alias_noaa

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

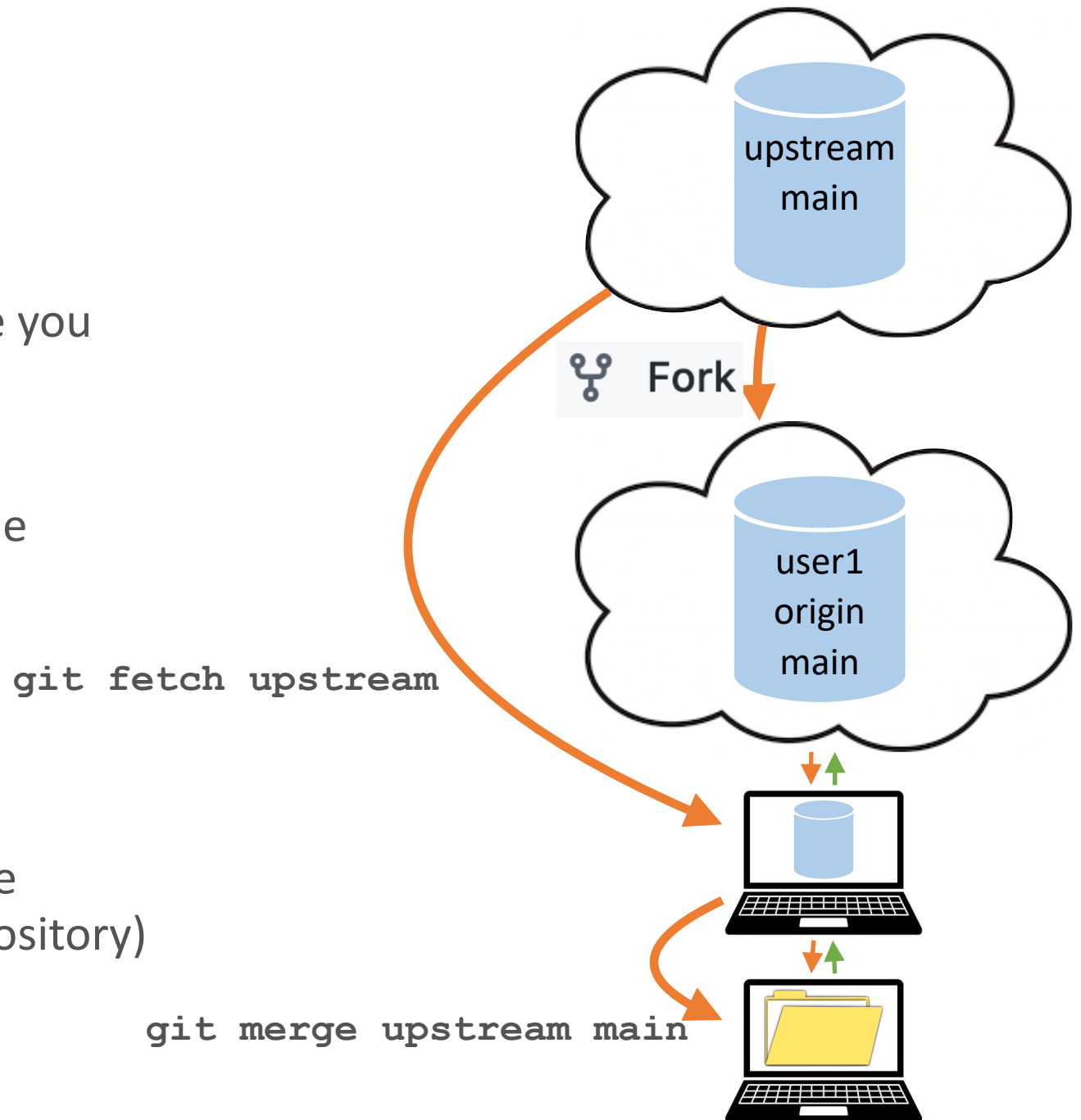
Here I have run `git status` immediately after making modifications and running `git add bashrc_hera`. Git status shows us that the file `bashrc_hera` has been staged to be committed, and there are other changes in the file `remote_alias_noaa` which is not staged for this commit. There is also an untracked file which we can see. Lastly, notice at the top, we see that “your branch is up to date with origin/master” - this is where we see if we are ahead, behind, or up to date with the online version.



Multi-user projects

Forking

- Make your own “version” of a repo, where you have all the editing power
- Once forked, the rest of the workflow is the same as before
- Easiest to do on [GitHub.com](https://github.com)
- **Upstream** is a commonly used label for the overarching/group **remote** (i.e. online repository)



Forking

github.com/mitgcm/mitgcm

MITgcm / MITgcm Public

Unwatch 36

Fork 188

Starred 232

Code Issues 104 Pull requests 24 Actions Projects Wiki Security Insights

master

1 branch 423 tags

Go to file

Add file

Code

About

M.I.T General Circulation Model master code and documentation repository

jm-c Merge pull request #604 from jm-c/deep_gmredi

59ccdec 4 days ago 19,637 commits

github.com/timothyas/mitgcm

timothyas / MITgcm Public

forked from MITgcm/MITgcm

Watch 0

Fork 188

Star 0

Code Pull requests Actions Projects Wiki Security Insights Settings

master

36 branches 389 tags

Go to file

Add file

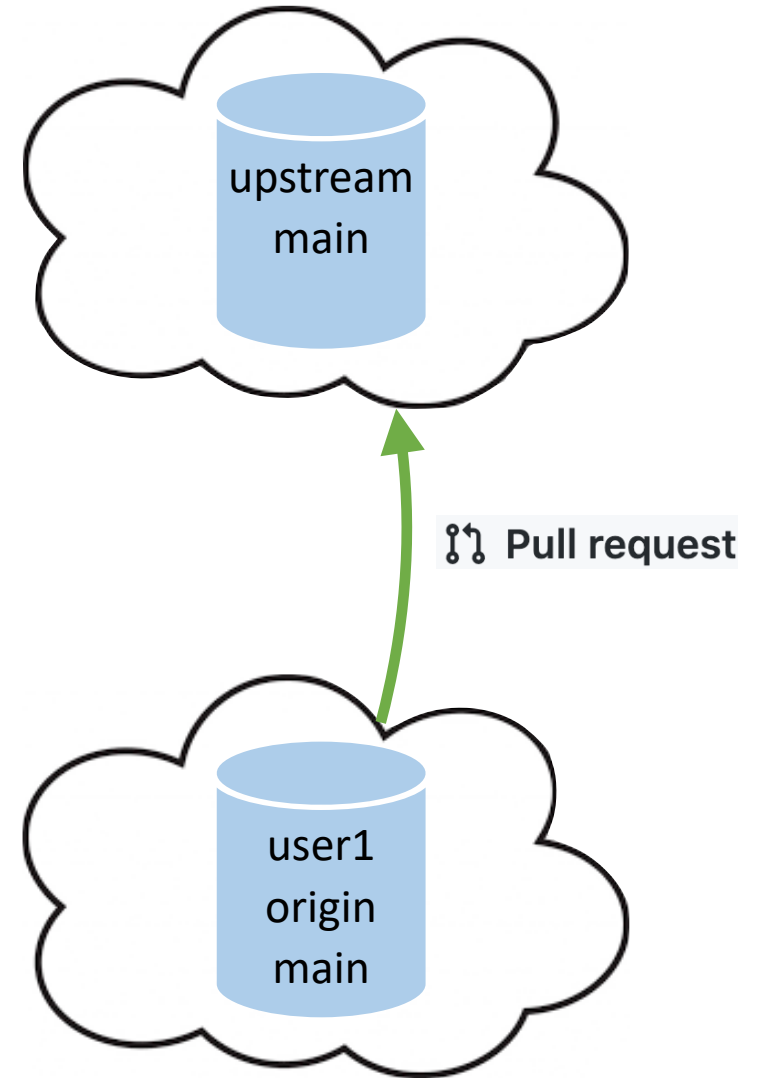
Code

About



Pull Requests

- Merge your modifications into the upstream version
- You are requesting that they pull your changes
- Easiest to do on [GitHub.com](https://github.com)



Raising Issues

- Open a dialogue about unexpected behavior, feature requests
- Anyone can do it on public projects
- Collaborators/maintainers can use the issue tracker as a todo list
- Easiest to do on [GitHub.com](https://github.com)
- JEDI issue recommendations [here](#)



MITgcm / MITgcm Public

Unwatch 36 Fork 188 Starred 232

Code Issues 104 Pull requests 24 Actions Projects Wiki Security Insights

master 1 branch 423 tags

Go to file Add file Code

About

jm-c Merge pull request #604 from jm-c/deep_gmredi ...

59ccdec 4 days ago 19,637 commits

M.I.T General Circulation Model master code and documentation repository

Refactoring Updated on Feb 26

Filter cards Add cards Fullscreen Menu

1 To do

- Build automation to test coding style and standards for all pull requests. #145 opened by StevePny

Automated as To do Manage

1 In progress

- ddc/modelbuilder #324 opened by timothyas

Automated as In progress Manage

3 Review in progress

- ddc/modelcaller refactor #327 opened by japlatt
1 linked pull request
- ddc/dabuilder refactor #325 opened by tse-chunchen
- ddc/data/data.py refactor #328 opened by StevePny

Automated as In progress Manage

0 Reviewer approved

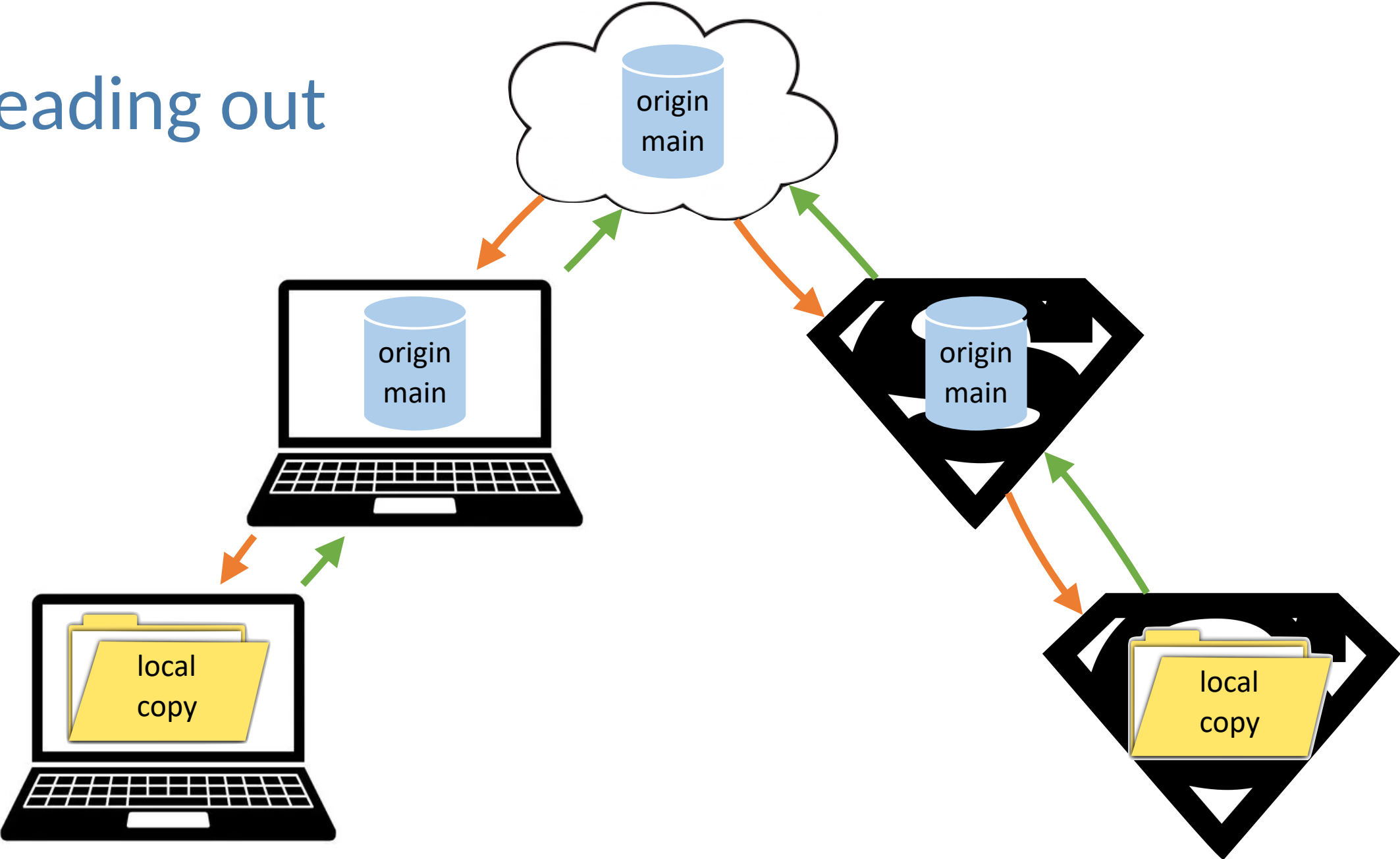
Automated as In progress Manage

15 Done

- Create environment yml file for repository. #24 opened by StevePny (bug)
- ddc/datools/etkf_4d.py refactor #336 opened by mikegoodliff
- ddc/datools/letkf_4d.py refactor #335 opened by mikegoodliff
- ddc/datools/etlm.py refactor #333 opened by mikegoodliff
- ddc/datools/etkf.py refactor #332 opened by mikegoodliff
- ddc/datools/var4d.py refactor #331 opened by mikegoodliff
- tests/test_class_datools*.py refactor #330 opened by mikegoodliff
- ddc/modeltools.py refactor #326 opened by hsinyilin19
- ddc/datools/ossesetup.py refactor #323 opened by mikegoodliff

Automated as Done Manage

Spreading out



Authentication

- My recommendation, use ssh keys
- Add each machine's public ssh key to github and never look back
- See guide [here](#)

[GitHub.com/settings/keys](https://github.com/settings/keys)



Timothy Smith

Your personal account [Switch to another account](#)

Public profile

Account

Appearance

Accessibility

Notifications

Access

Billing and plans

Emails

Password and authentication

SSH and GPG keys

Organizations

Moderation

SSH keys

This is a list of SS



Tim'
SHA2
Add
Last

SSH



Ekm
SHA2
Add
Last

SSH



Stan
SHA2
Add

SSH

HPC+Git tips

Dealing with large files, slurm output, etc:

- Don't do: `git add *`
- Maintain a `.gitignore` file

When code is spread across many machines, using separate branches can be helpful

```
(base) [tim myrun]$ ls -alh
total 0
drwxr-xr-x  3 tsmith  staff   96B Mar 30 21:14 .
drwxr-xr-x 11 tsmith  staff  352B Mar 30 21:14 ..
-rw-r--r--  1 tsmith  staff   0B Mar 30 21:14 really_big_data.nc
```

1. We have the `really_big_data.nc` file that we want to ignore

```
(base) [tim myrun]$ git status .
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

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

2. Git sees the file, and will happily scoop it up (and eventually run into trouble with large files)

```
1 # gitignore file
2 *.nc
3 █
```

3. We create a file called `.gitignore`, and tell it to ignore all files with the suffix `.nc`

```
(base) [tim myrun]$ git status .
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

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

nothing added to commit but untracked files present (use "git add" to track)
```

4. Git is now ignoring the `really_big_data.nc` (and now we should commit our `.gitignore` file!)

Filetype best practices

- Git is most useful for text based files (code and scripts, LaTeX, markdown, yaml files,...)
- Jupyter notebooks: no problem keeping them in repos, but are difficult to version control directly. Tools like [nbdime](#) help.
- Log, compiler output, error files (clutter) generally not recommended
- Pdf's and images: I keep them if necessary to compile, but don't expect nice version control
- Microsoft Powerpoint, Word, Excel documents (and apple equivalents): I don't recommend storing in GitHub, because usually large. I prefer keeping these in e.g. Google Drive, Dropbox, etc.
- Binaries, data, NetCDF, zarr: No. See [Git Large File Storage](#) or [figshare.com](#) for publicly storing large files
- See GitHub's [.gitignore templates](#)

What to make repos for?

My personal usage...

- [bash-envy](#): just environment stuff for easy setup on many machines (.bashrc, conda recipe files, vimrc, ...)
- [pych](#): PYthon scratCH work, my own scratch python package I can import anywhere
- New repo for each new project/paper that contains latex, jupyter notebooks, some python code to explore a contained idea
- Every time I have created “one repo to rule them all”, things got messy

Happy hacking!

