Introduction to Version Control using Git

CC-BY

You are free to:

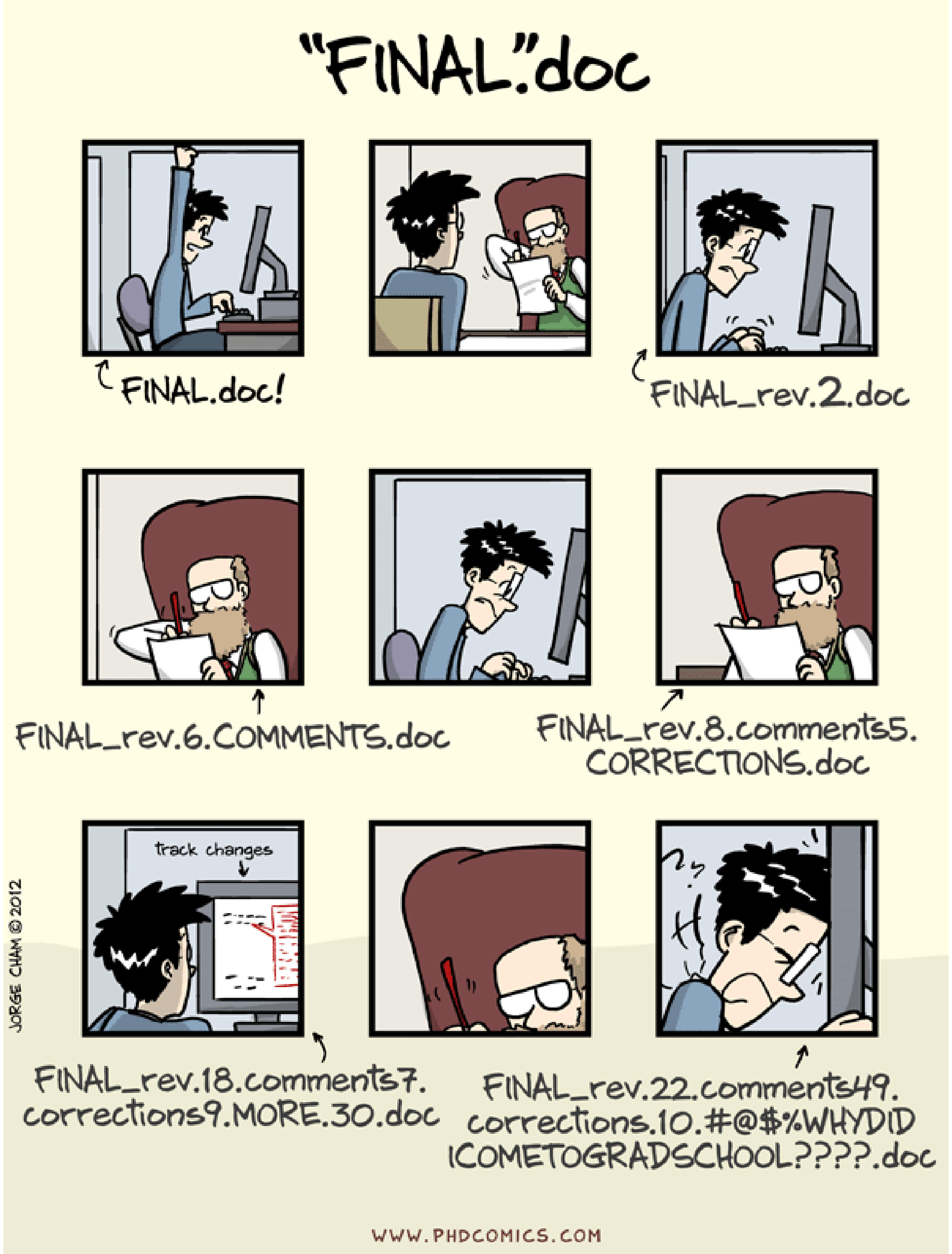
Copy, share, adapt, or re-mix;

Photograph, film, or broadcast;

Blog, live-blog, or post video of;

Provided that:

You attribute the work to its author and respect the rights and licenses associated with its components.


"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.CORRECTIONS.doc
track changes
FINAL_rev.18.comments7.corrections9.MORE.30.doc
FINAL_rev.22.comments49.corrections.10.#@$%WHYDIDICOMETOGRADSCHOOL????.doc
JORGE CHAM © 2012
WWW.PHDCOMICS.COM

Outline

- What is Version Control
- Why we need it in science
- Git as a version control system

Version Control

- System to manage different versions of a single file

Working Copy

my_code.txt

History of Edits

my_code.txt

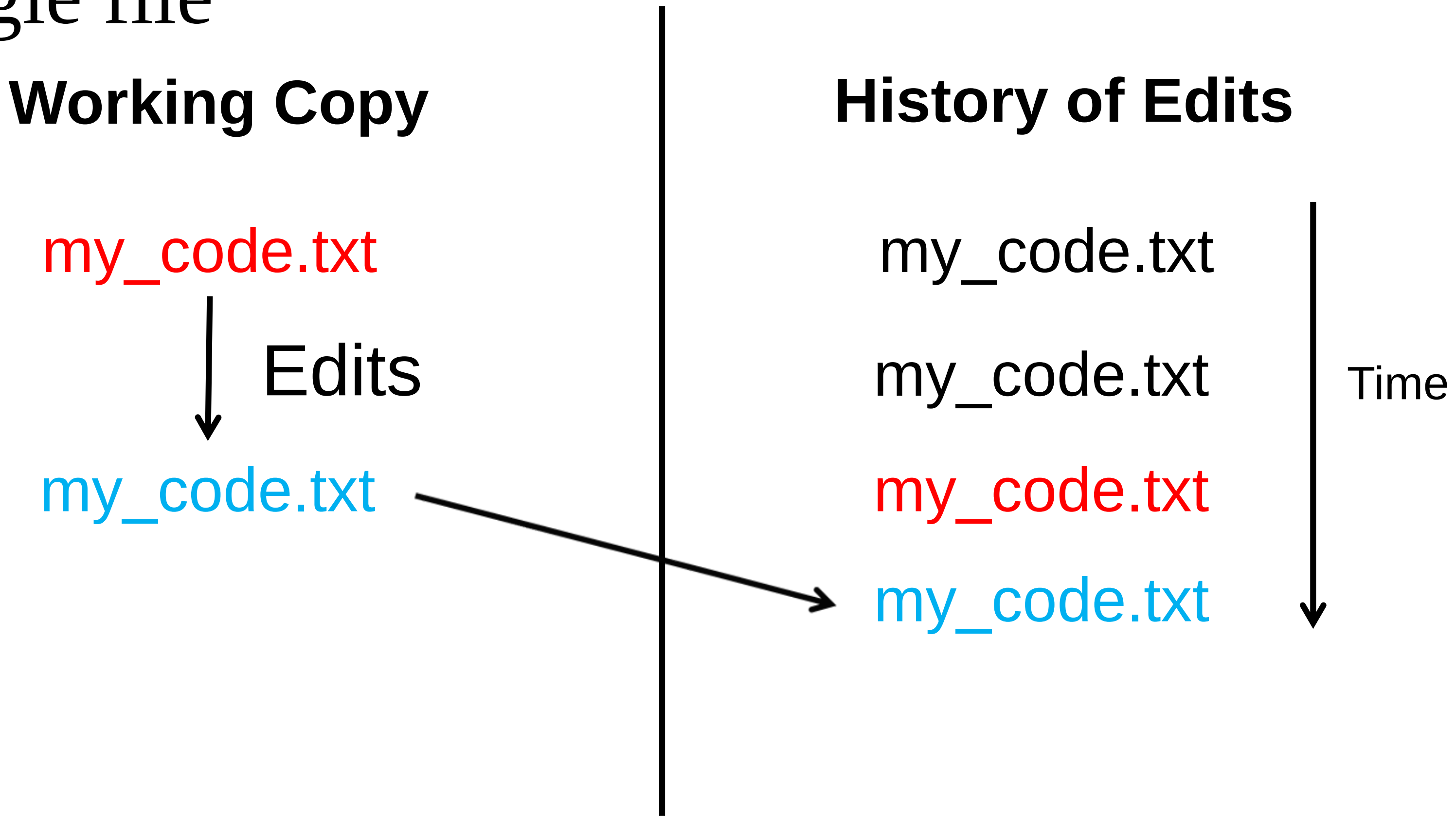
my_code.txt

my_code.txt

Time $\downarrow$

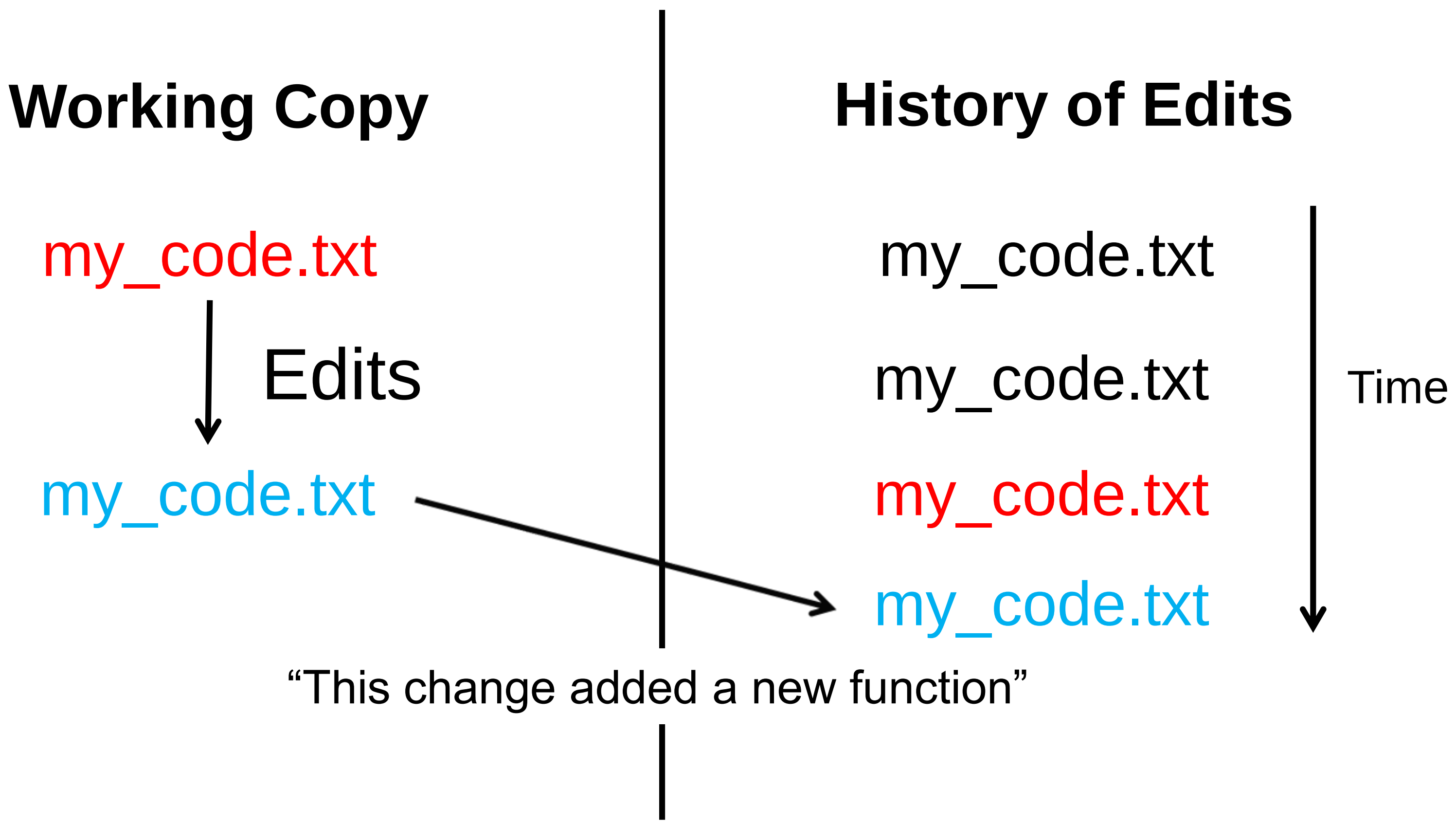
Version Control

- System to manage different versions of a single file



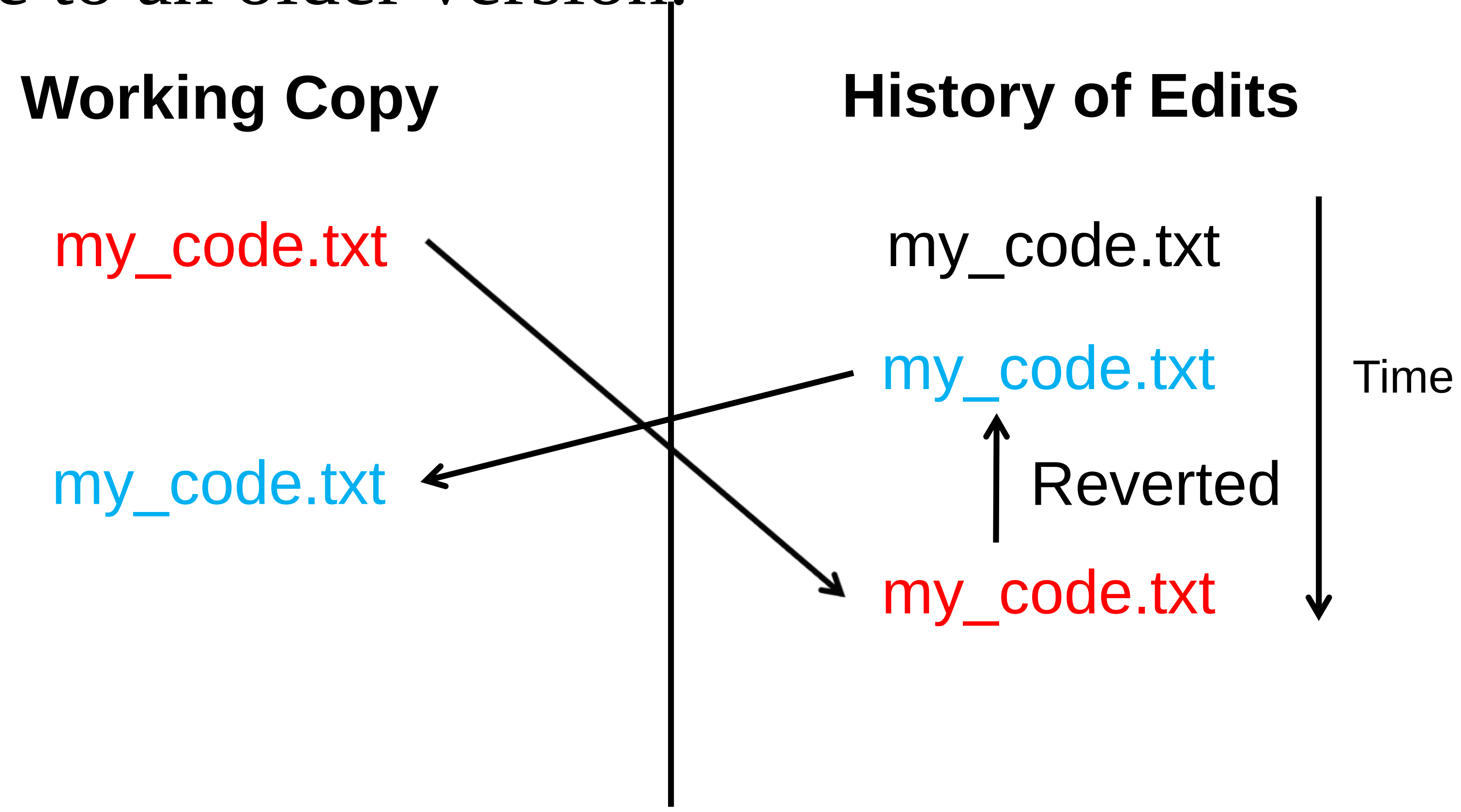
Version Control

- Ability to annotate changes (e.g., Lab Notebook)



Version Control

- Enables the ability to retrace edits to revert the file to an older version.



Why Do We Need This?

- Reproducibility → like a lab notebook

- Personal workflow
  - Final.doc example

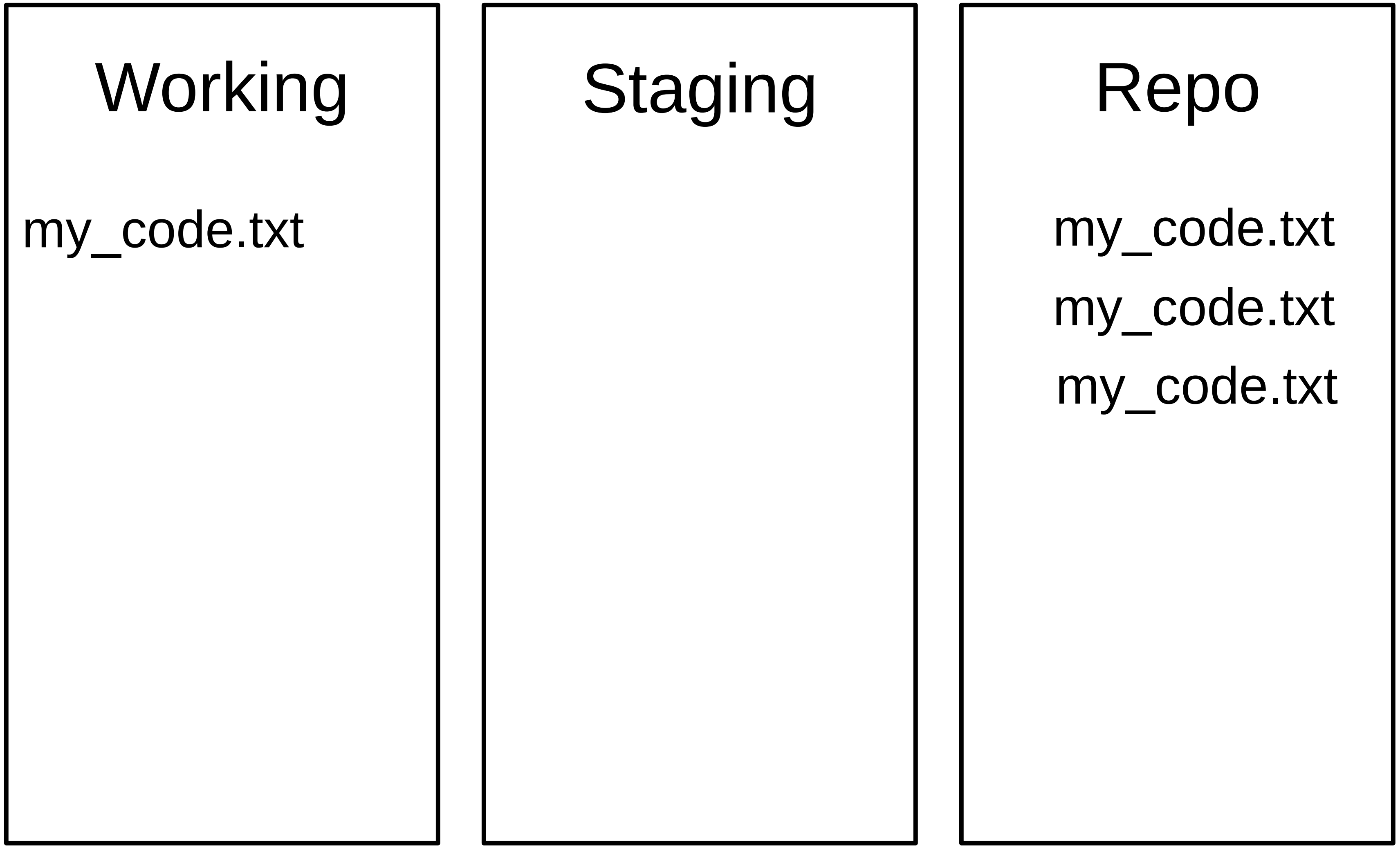
- Collaborative workflow
  - Allows a team of scientists contribute simultaneously to a project without fear of overwriting each other or breaking something

Why Git?

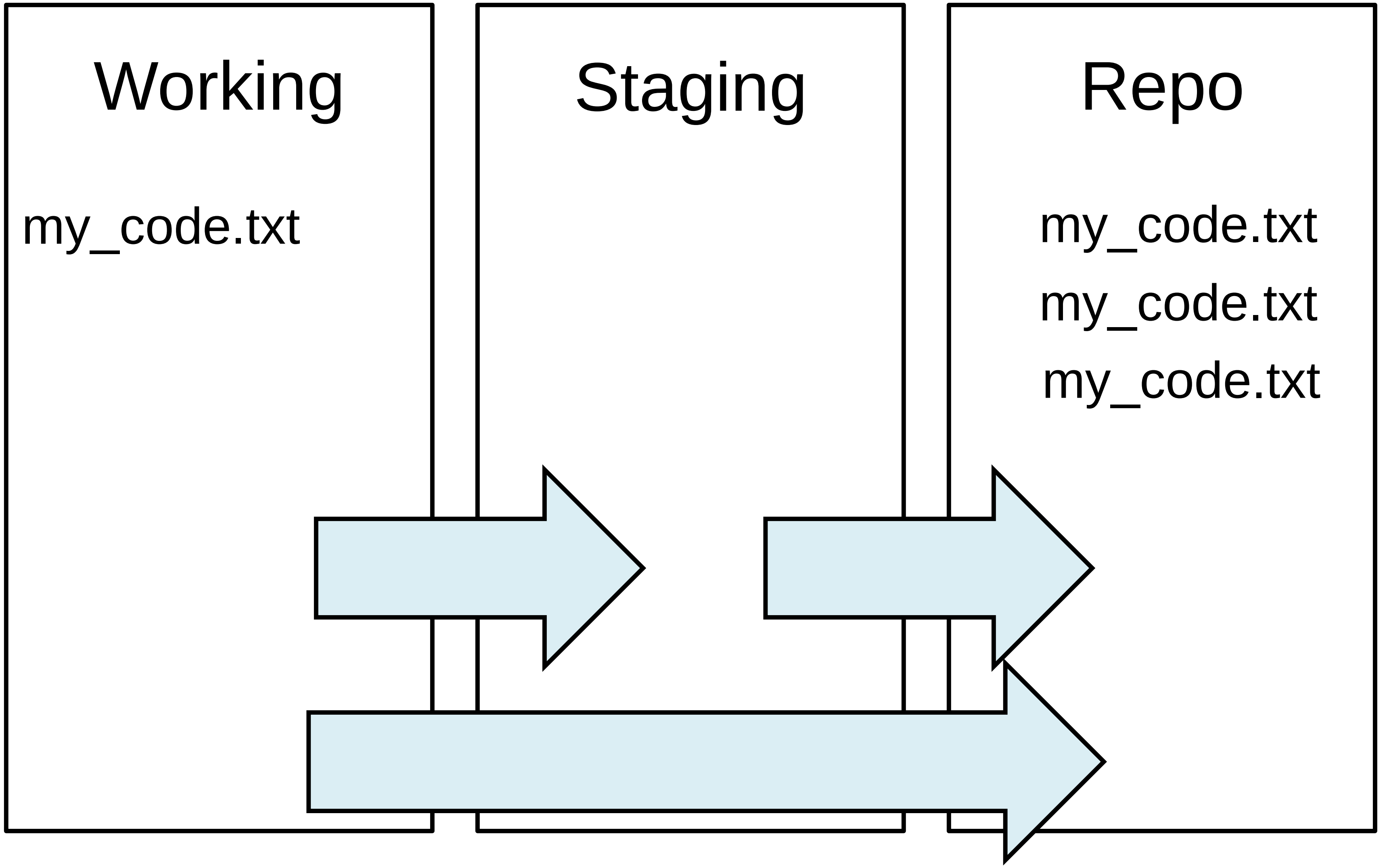
- Free Open Source
- Popular with active user-base
- Decentralized and distributed
- Requires only infrequent connection to server

Ram, K. - Git can facilitate greater reproducibility and increased transparency in science. http://dx.doi.org/10.6084/m9.figshare.153821

Local Git Structure

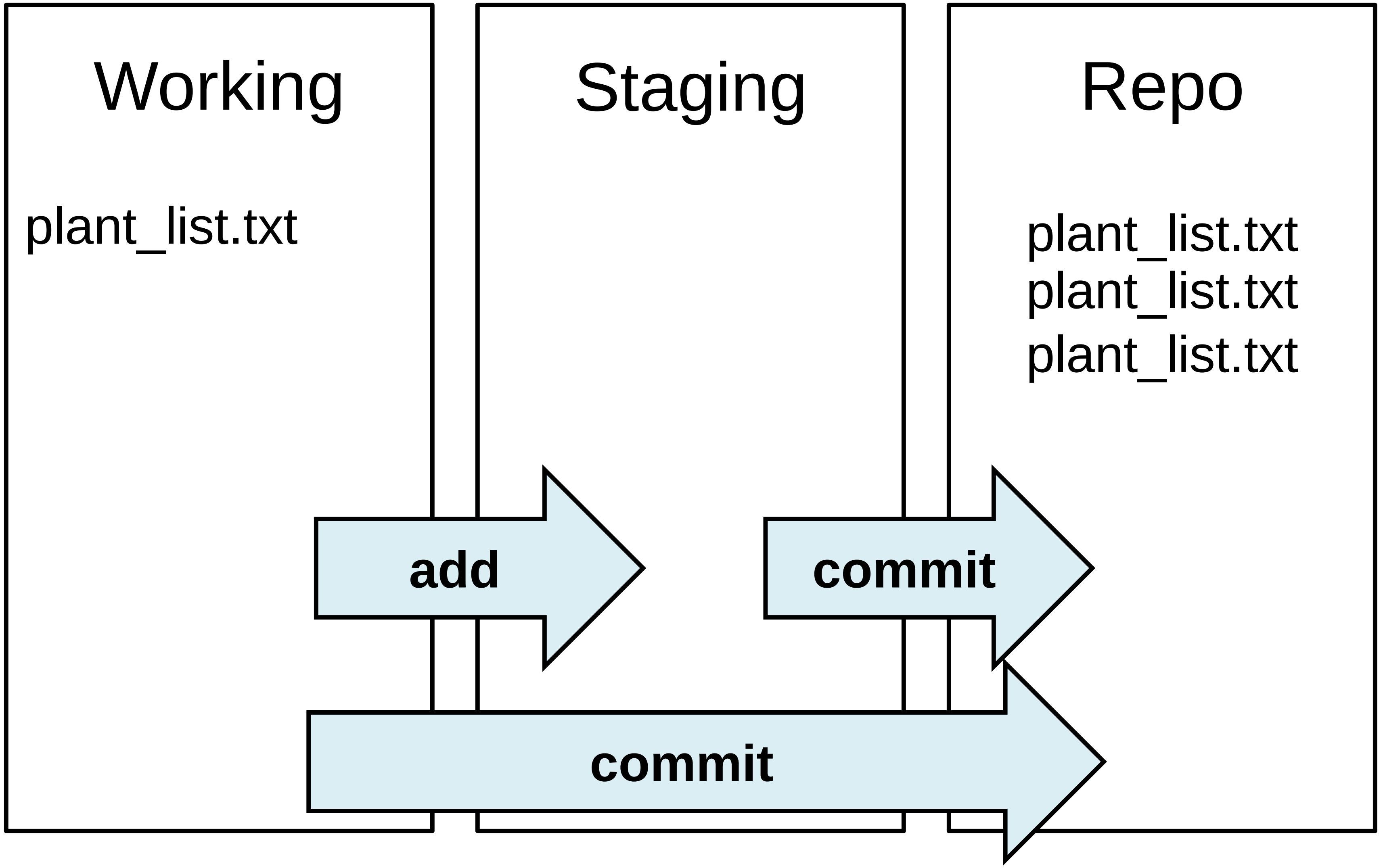


Local Git Workflow



First Commit: Follow Along Demo

Local Git Workflow



Staging Files

```
$ git add .
```

- Stages all new or modified files in the working directory

```
$ git add <file_name>
```

- Stages a specific new or modified file

Committing Files

```
$ git commit –m "Description of changes"
```

- Save a snapshot of all staged files to the repo

```
$ git commit <file_name> -m "Description of changes"
```

- Save a snapshot of a specific file to the repo

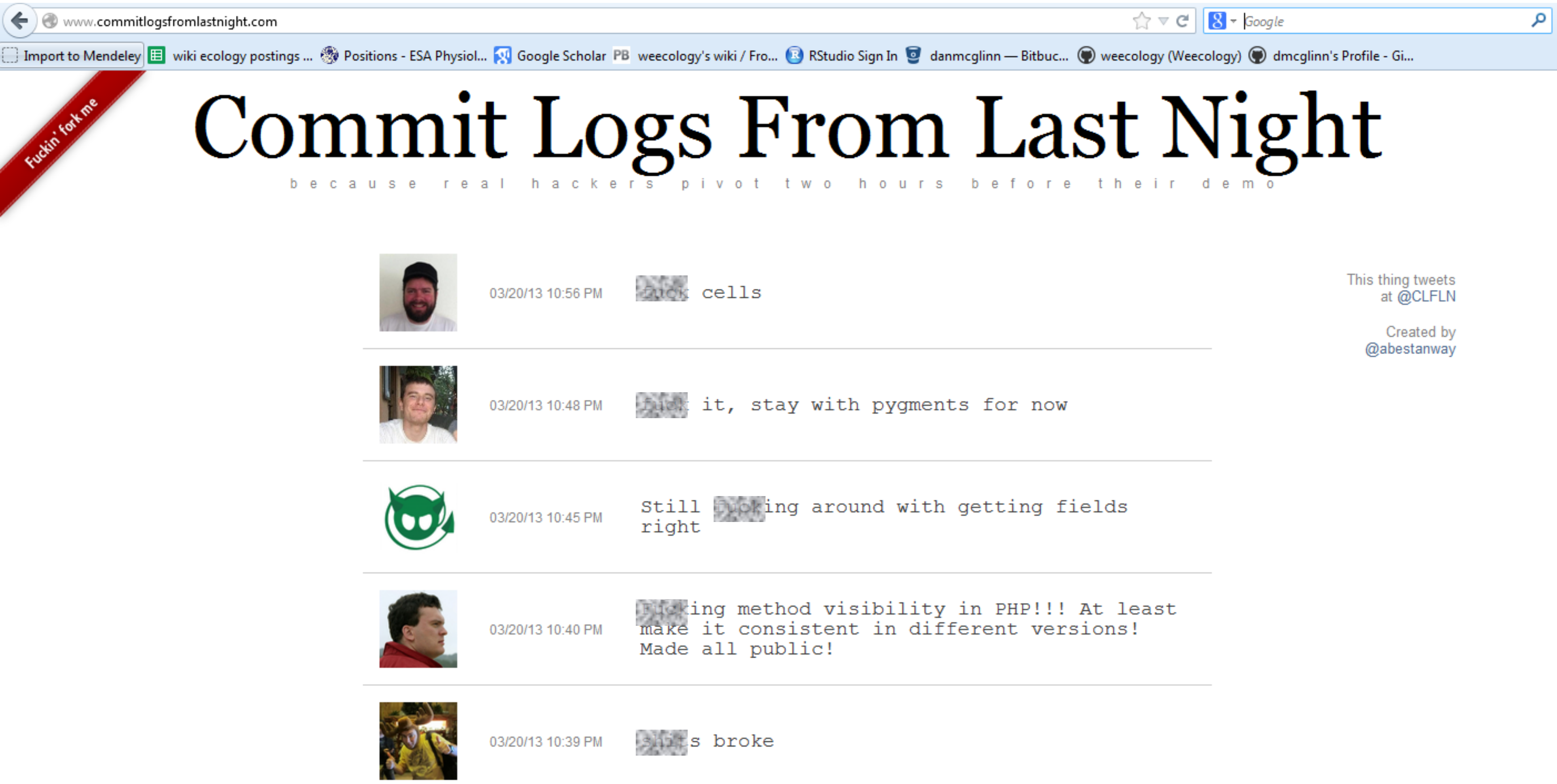
Meaningful Commit Messages

- First line should be a short description
  - 50 characters
  - Informative → what the commit does
  - Like a subject of an email

- Additional text should:
  - include motivation for the change
  - contrast its implementation with previous behavior

- Should be in present tense

Where you don't want to be…



www.commitlogsfromlastnight.com

Import to Mendeley | wiki ecology postings ... | Positions - ESA Physiol... | Google Scholar | weecology's wiki / Fro... | RStudio Sign In | danmcglinn — Bitbuc... | weecology (Weecology) | dmcglinn's Profile - Gi...

Fuckin' fork me

Commit Logs From Last Night

because real hackers pivot two hours before their demo

```
03/20/13 10:56 PM    [redacted] cells

03/20/13 10:48 PM    [redacted] it, stay with pygments for now

03/20/13 10:45 PM    Still [redacted]ing around with getting fields
                     right

03/20/13 10:40 PM    [redacted]ing method visibility in PHP!!! At least
                     make it consistent in different versions!
                     Made all public!

03/20/13 10:39 PM    [redacted]s broke
```

This thing tweets at @CLFLN

Created by @abestanway

Bookkeeping

```
$ git init <repo_name>
```

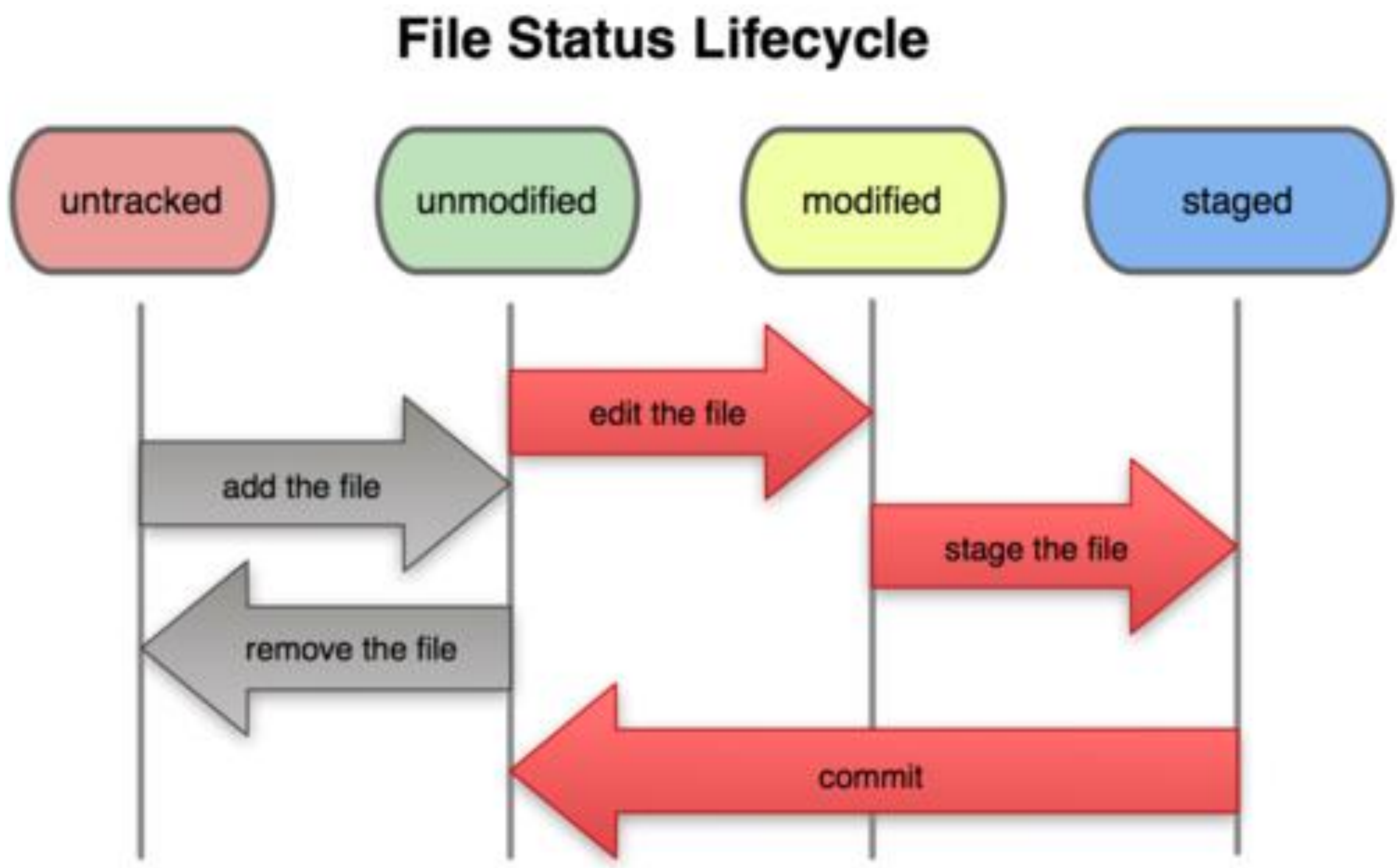
- Creates git repo

```
$ git status
```

- Describes current status of any files in the git repo
  - Untracked
  - Modified
  - Staged

```
$ git log
```

- Provides history of commits to the repo



http://git-scm.com/book

First Commit:
Do It Yourself

- Add new observations to **plant_list.txt**

- Create a new file called **site_list.txt**
  - Add this to the repo

Move and Remove Files: Follow Along Demo

Moving and Removing Files

```
$ git mv <old_file_name> <new_file_name>
```

- Renames or moves a file in the repo to a new location

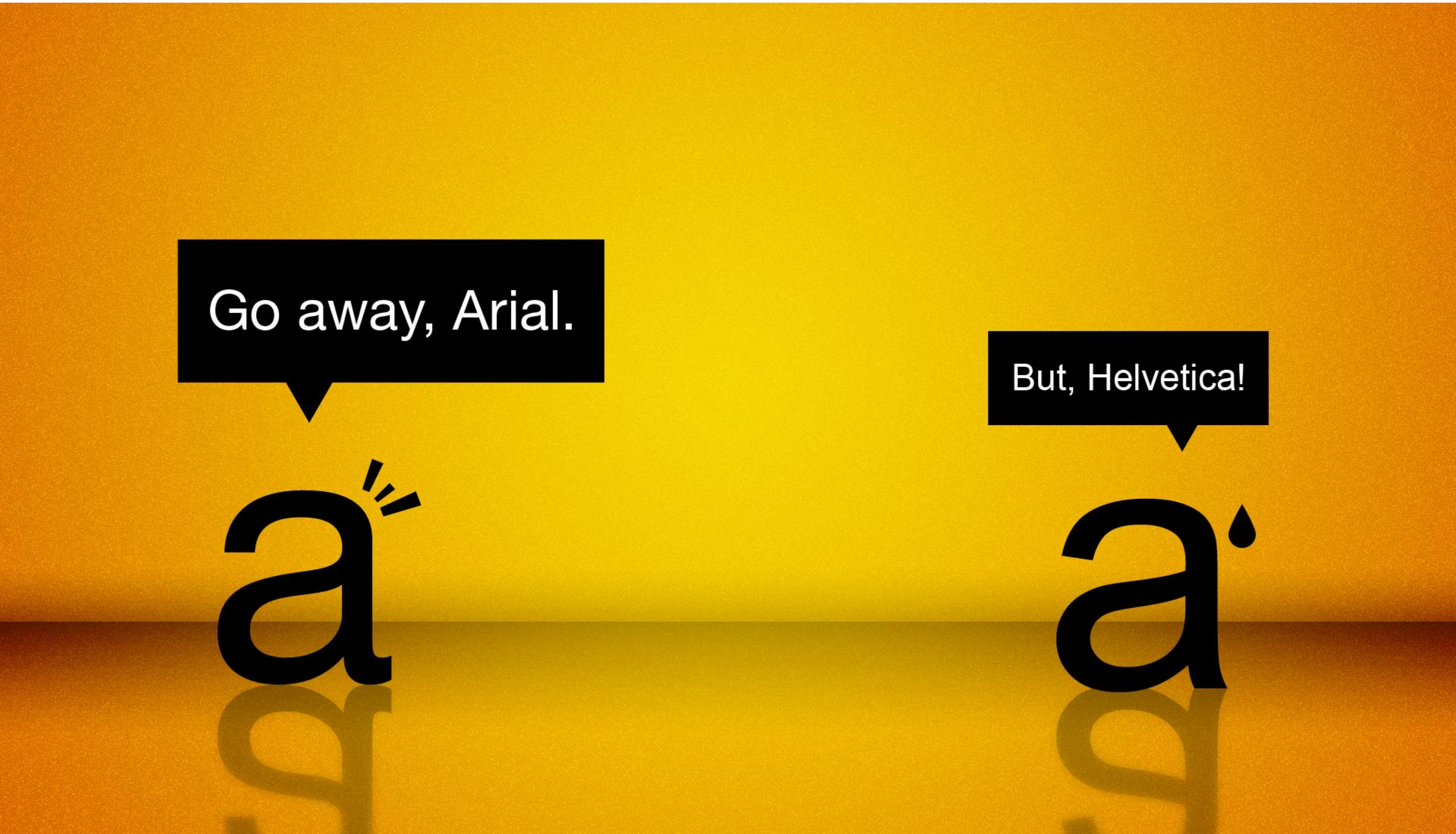
```
$ git rm <file_name>
```

- Removes a file from the repo

Move and Remove Files: Do It Yourself

- Remove the file **site_list.txt**
- Create a directory called **data**
- Move the file **plant_list.txt** and **habitat_list.txt** to the directory **data**

Finding What's Different: Follow Along Demo



Finding Differences

```
$ git diff
```

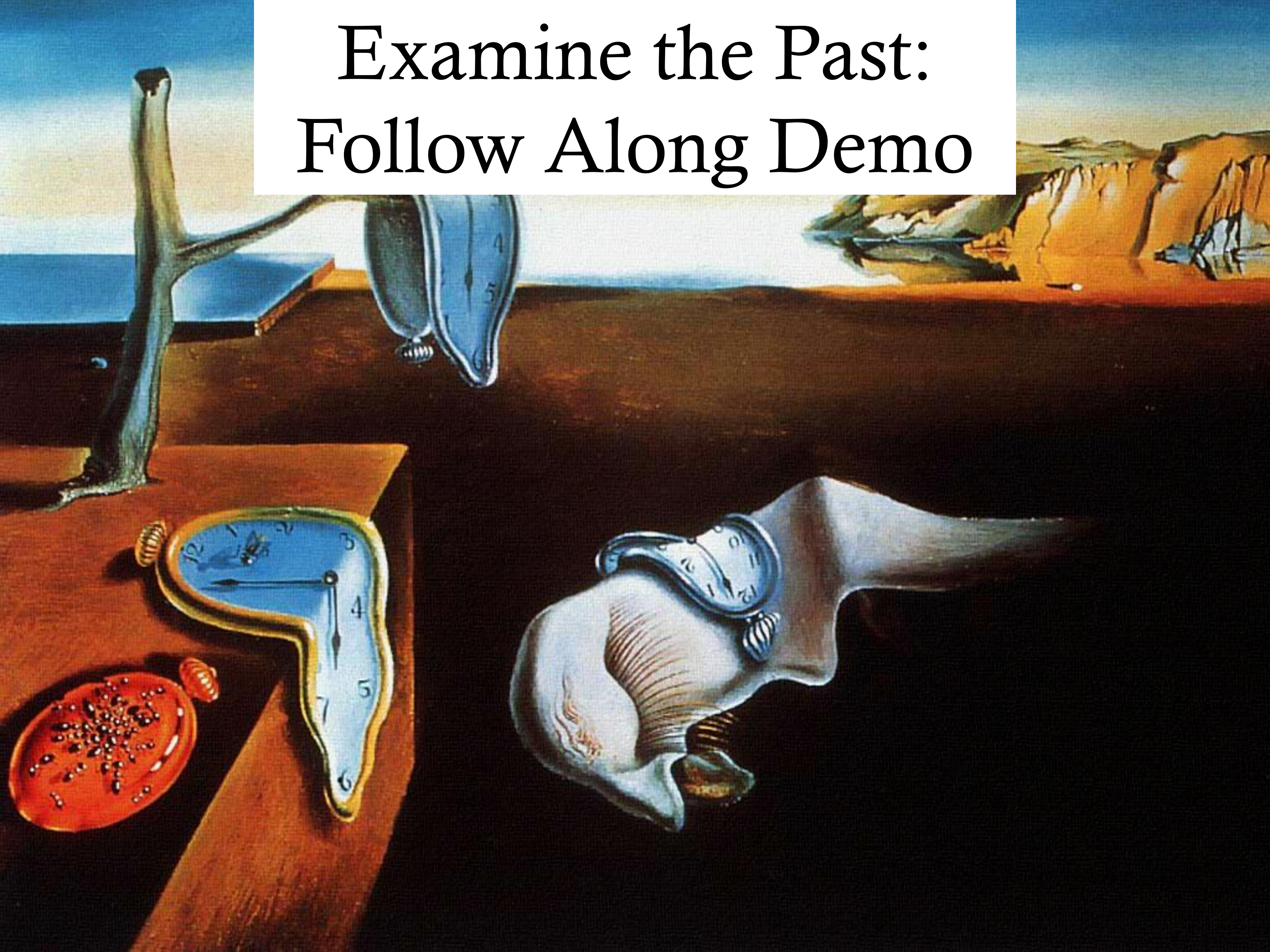
- Describes all differences for all modified files with respect to the last commit

```
$ git diff <file_name>
```

```
$ git diff <commit_hash_id>
```

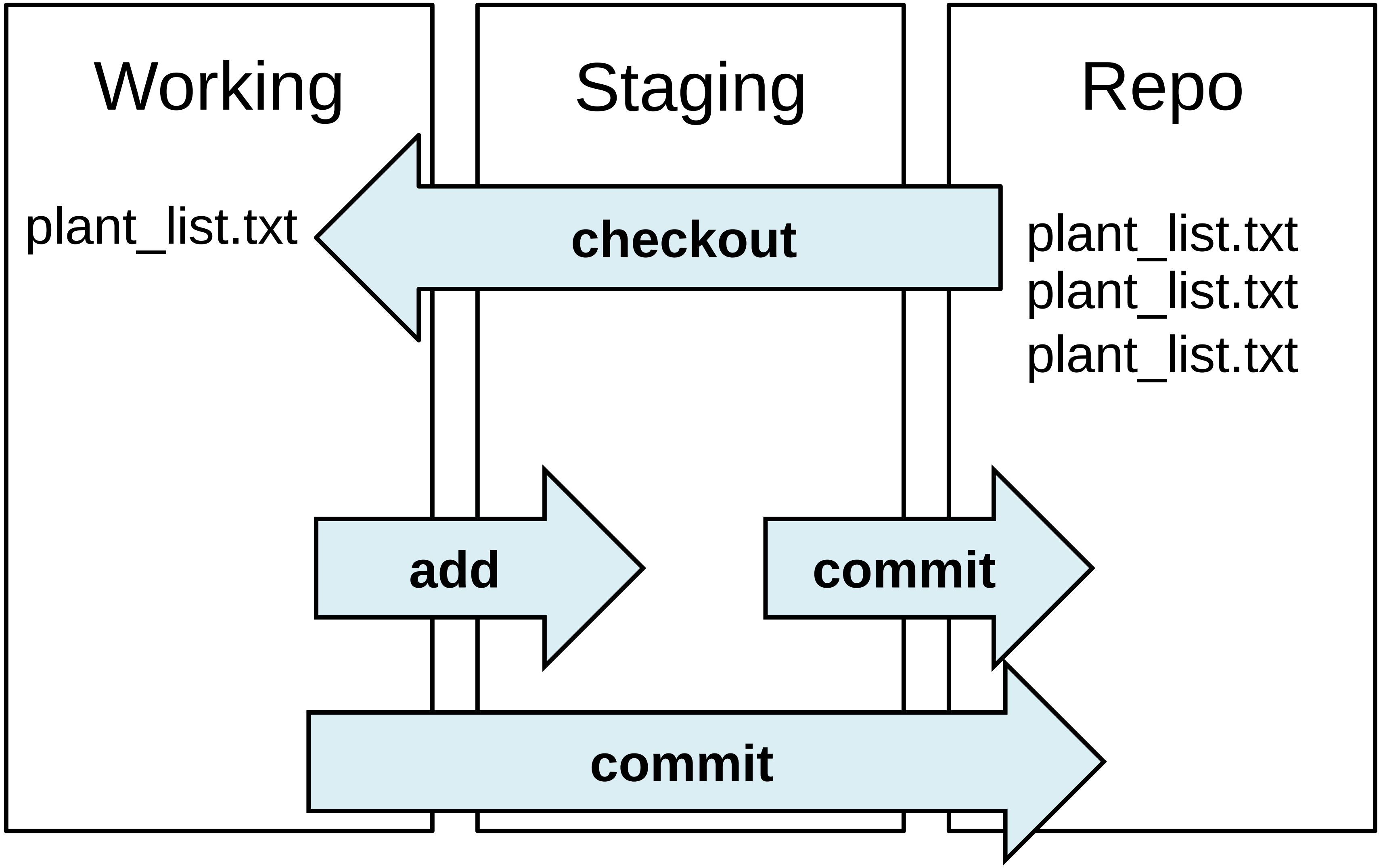
```
$ git diff <commit_hast_id> <file_name>
```

- Describes all differences between current state of file and a specific commit for a specific file



Examine the Past: Follow Along Demo

Local Git Workflow



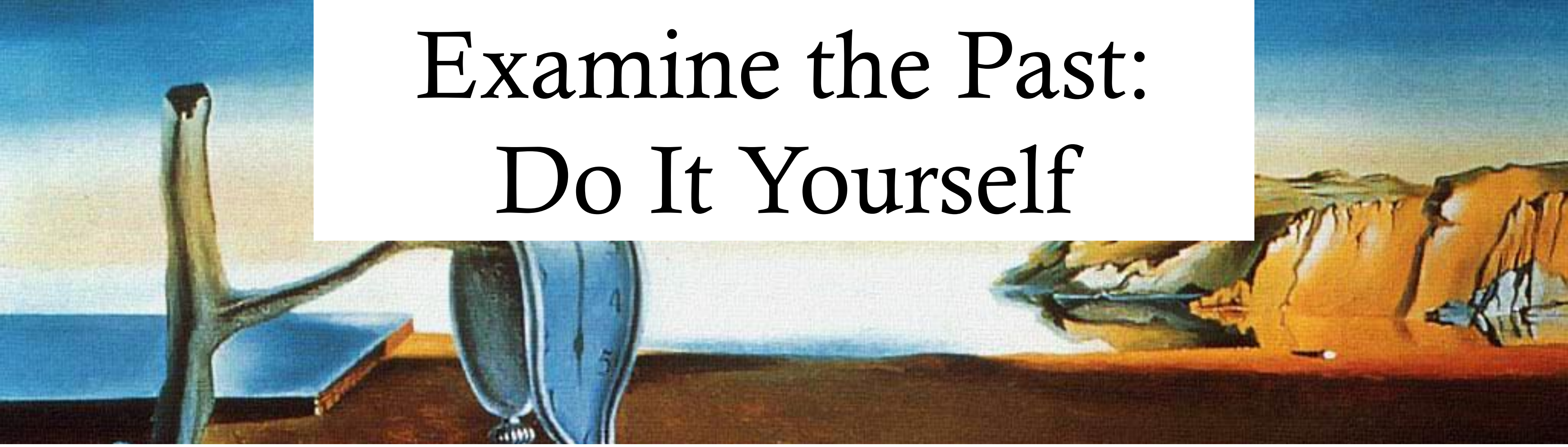
Checkout

```
$ git checkout <commit_hash_id>
```

- Superficially returns the working directory to its state from a previous commit
- Visibly rolls back all intermediate commits, but does not delete those changes
- Detaches HEAD

```
$ git checkout master
```

- Returns HEAD to the master branch and returns the working directory to the state of the last commit



Examine the Past: Do It Yourself

- Change **habitat_list.txt** and commit your change.
- You decide you would like to see the old **habitat_list.txt**
- Checkout the commit just prior to when you removed it
- Examine the status of your repo and **habitat_list.txt**
- Return to the master branch

Turn Back Time: Follow Along Demo

Recovering Previous File States

```
$ git checkout <file_name>
```

- Rolls back any unstaged changes to the last commit state and omits them

```
$ git reset <file_name>
```

- Unstages changes but does not omit them

```
$ git reset –hard <file_name>
```

- Unstages changes and omits them

```
$ git reset <commit_hash_id>
```

- Rolls back committed changes but does not omit them for all files to a specific version of the repo

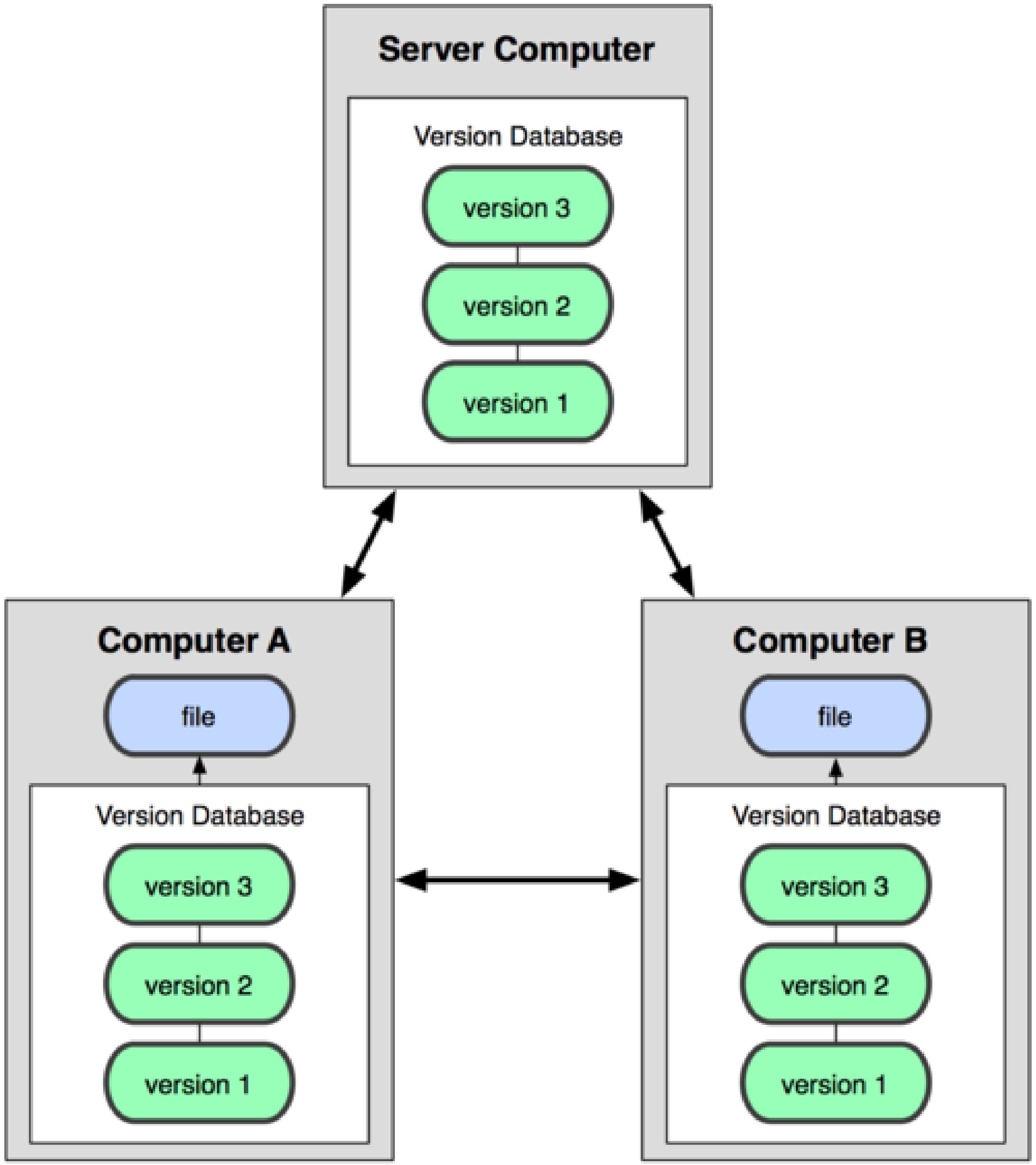
Turn Back Time: Do It Yourself

- Modify **plant_list.txt**
- Return **plant_list.txt** to the last commit state

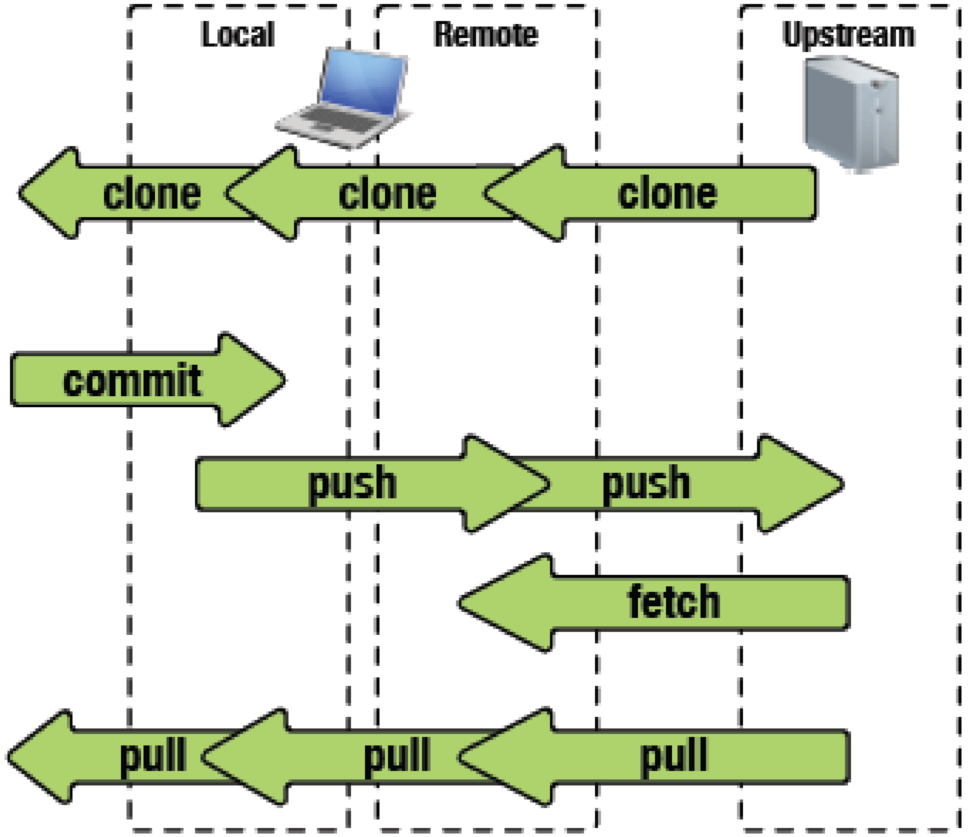
- Modify **plant_list.txt** and stage it
- Return **plant_list.txt** to the last commit state

- Return the working directory to when **habitat_list.txt** existed

Git Local & Remote Structure

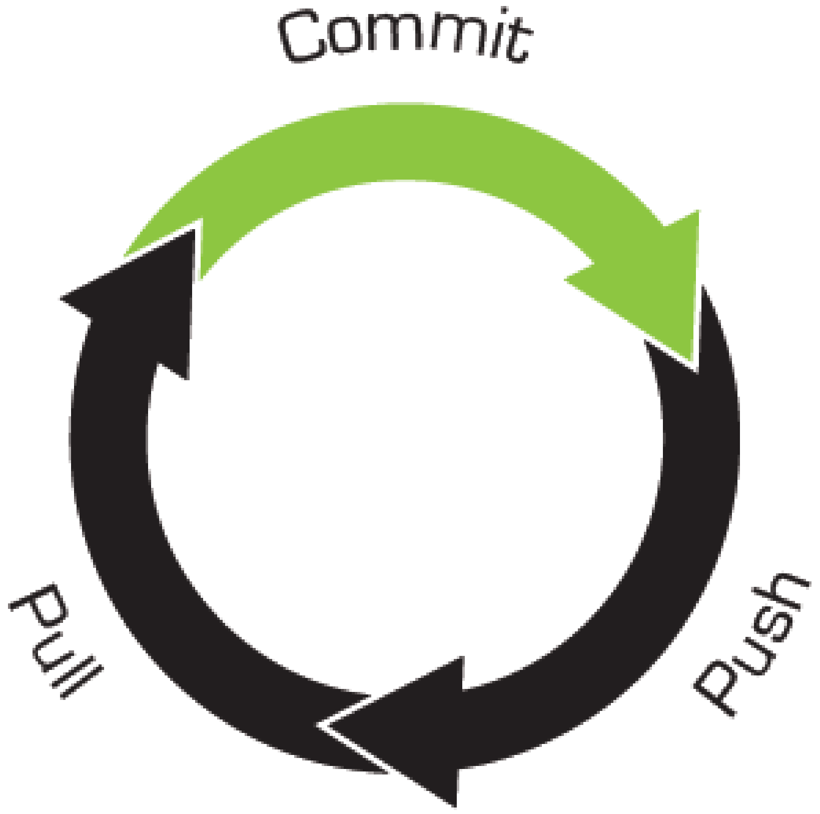


Git Remote Workflow



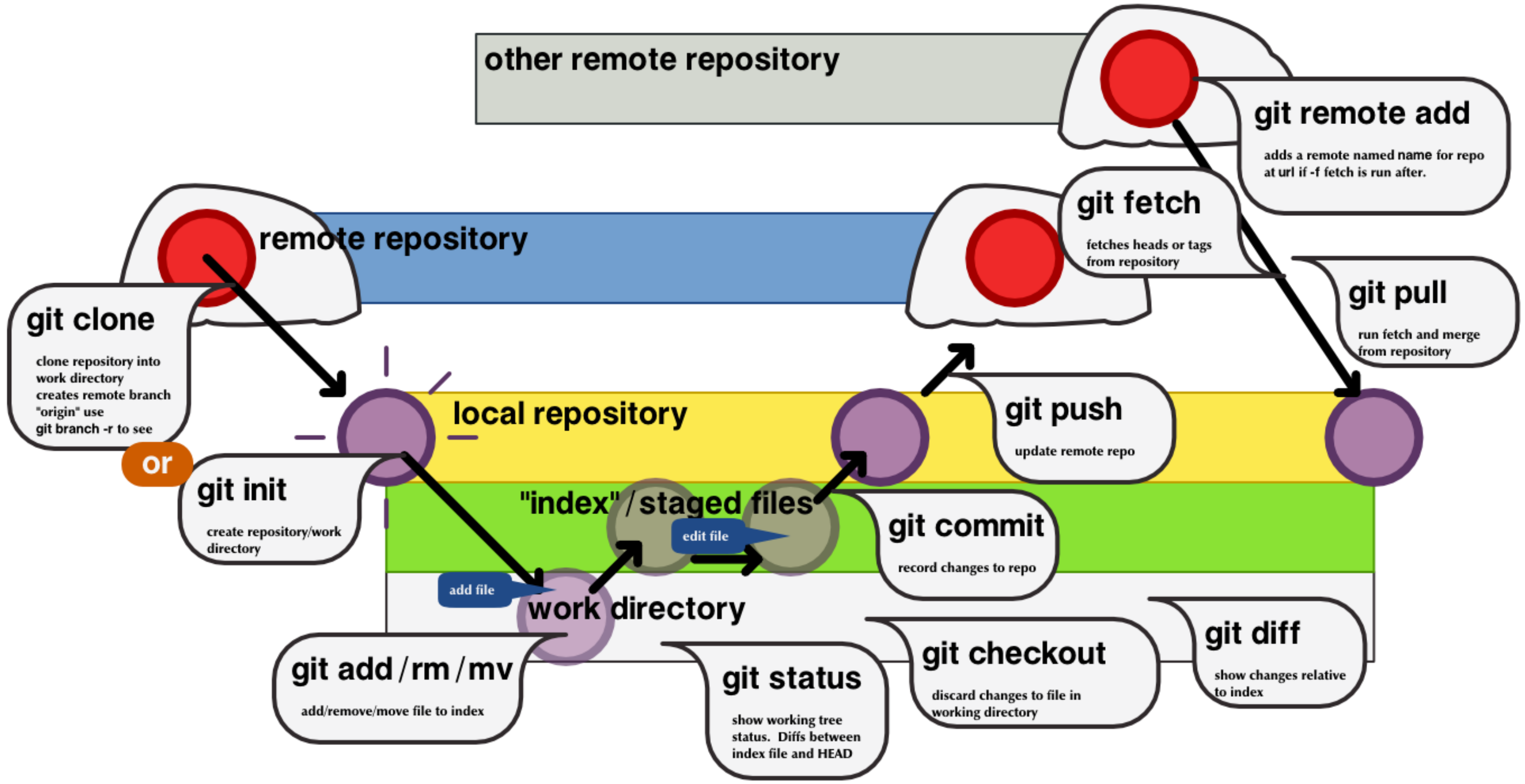
McCullough 2012

Git Workflow Cyle

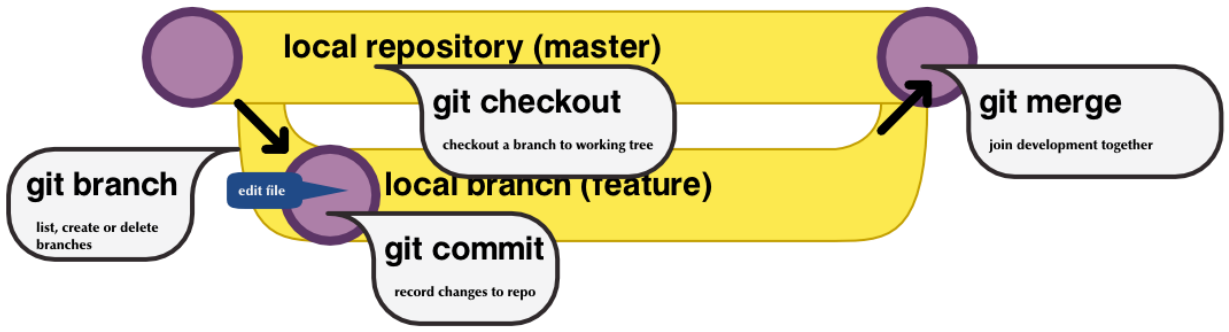


McCullough 2012

Git Hub Demo


other remote repository
git remote add
adds a remote named name for repo at url if -f fetch is run after.
remote repository
git fetch
fetches heads or tags from repository
git pull
run fetch and merge from repository
git clone
clone repository into work directory creates remote branch "origin" use git branch -r to see
or
local repository
git push
update remote repo
git init
create repository/work directory
"index" / staged files
edit file
git commit
record changes to repo
add file
work directory
git diff
show changes relative to index
git checkout
discard changes to file in working directory
git add / rm / mv
add/remove/move file to index
git status
show working tree status. Diffs between index file and HEAD

GIT Visual cheat sheet


local repository (master)
git checkout
checkout a branch to working tree
git merge
join development together
git branch
list, create or delete branches
edit file
local branch (feature)
git commit
record changes to repo

https://raw.github.com/mattharrison/Git-Supervisual-Cheatsheet/master/gitcheat.png