

# Introduction to Version Control with a focus on Git

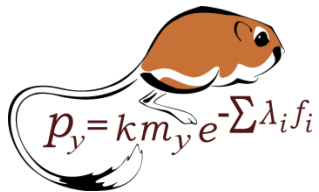
*Go back in time with....*  
**Version Control**



# HELLO

my name is

*Dan McGlinn*



Weecology Lab  
@danmcglinn



danmcglinn@gmail.com

<http://mcglinn.web.unc.edu>

# "FINAL".doc



FINAL.doc!



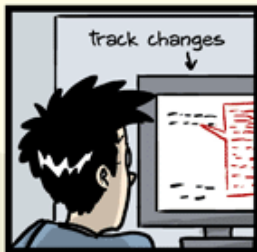
FINAL\_rev.2.doc



FINAL\_rev.6.COMMENTS.doc



FINAL\_rev.8.comments5.  
CORRECTIONS.doc



FINAL\_rev.18.comments7.  
corrections9.MORE.30.doc



FINAL\_rev.22.comments49.  
corrections.10. #@\$%WHYDID  
ICOMETOGRADSCHOOL????.doc

# 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



# Version Control

- System to manage different versions of a single file

## Working Copy

my\_code.txt



Edits

my\_code.txt

## History of Edits

my\_code.txt

my\_code.txt

my\_code.txt

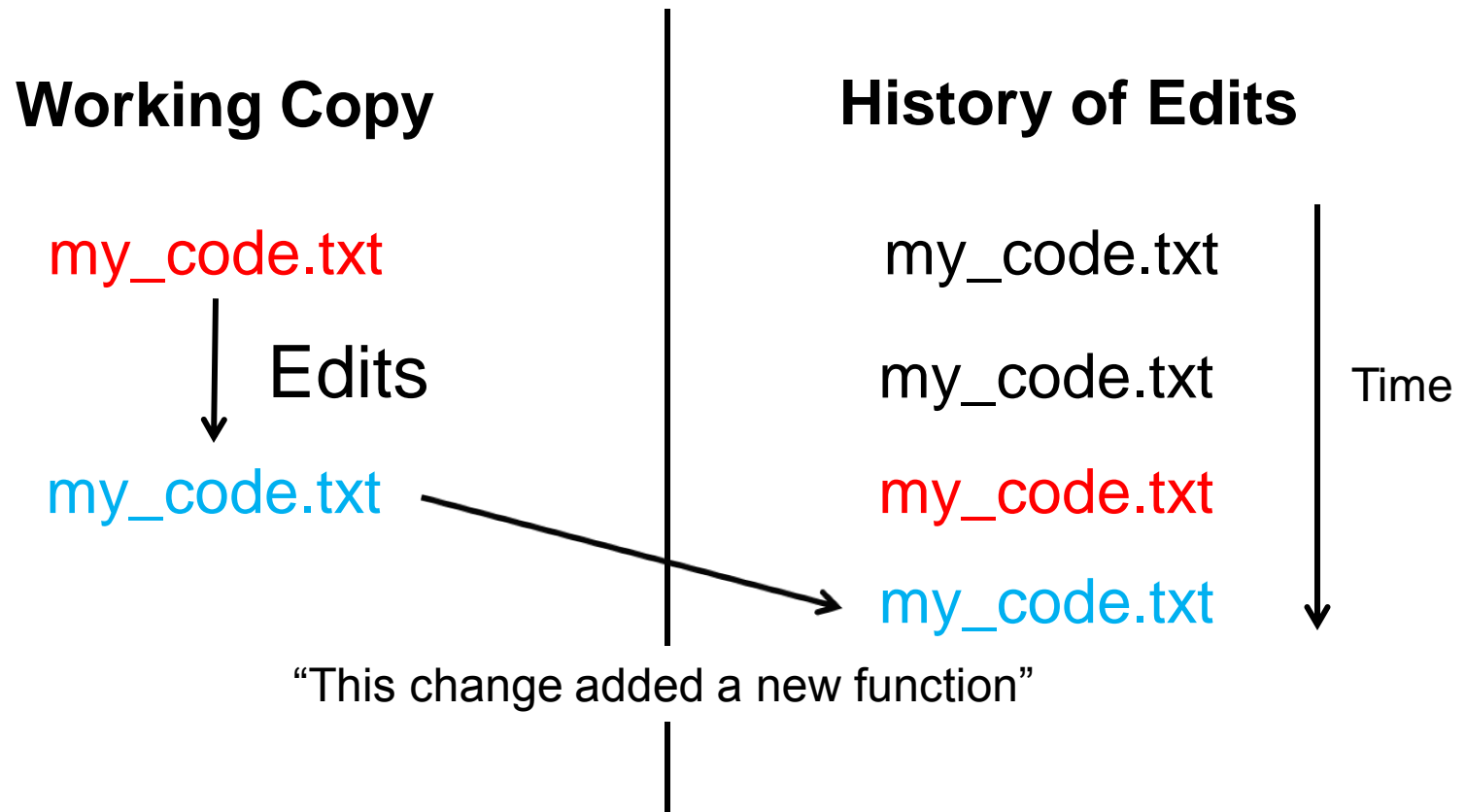
my\_code.txt

Time



# 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.

## Working Copy

my\_code.txt

my\_code.txt

## History of Edits

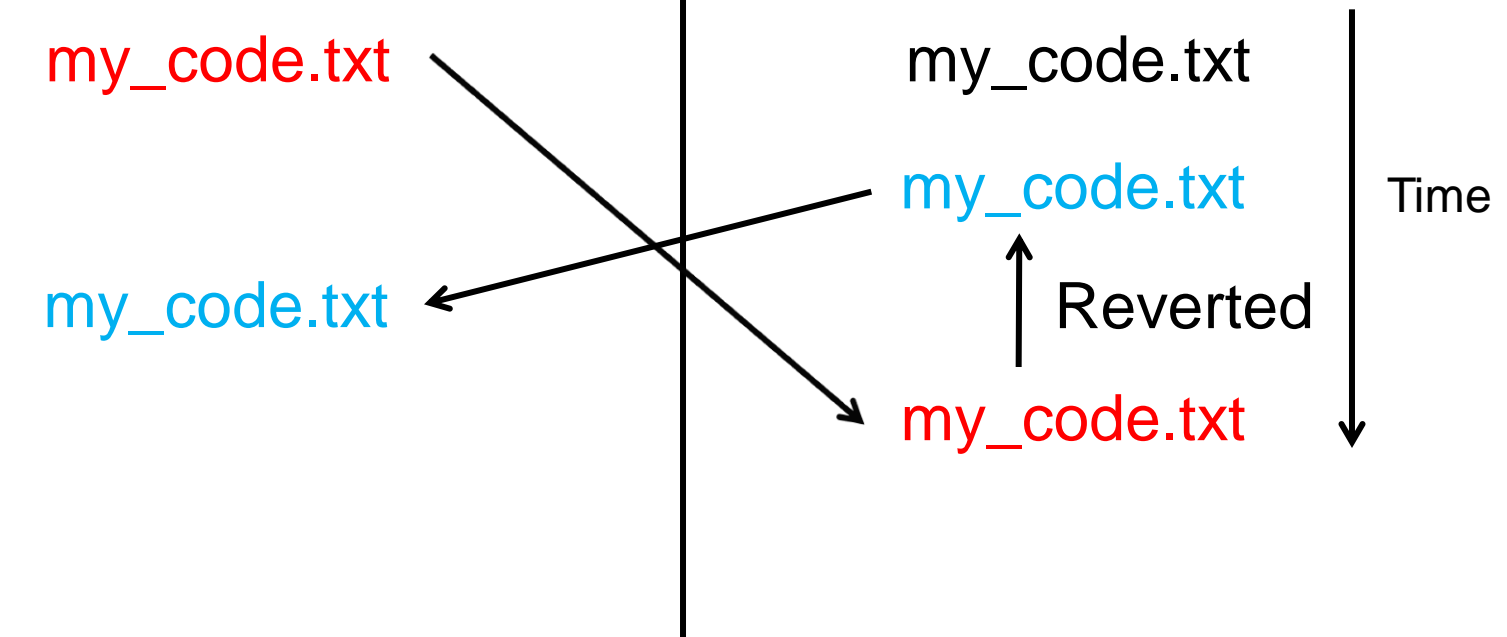
my\_code.txt

my\_code.txt

my\_code.txt

Reverted

Time





# 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

Working

my\_code.txt

Staging

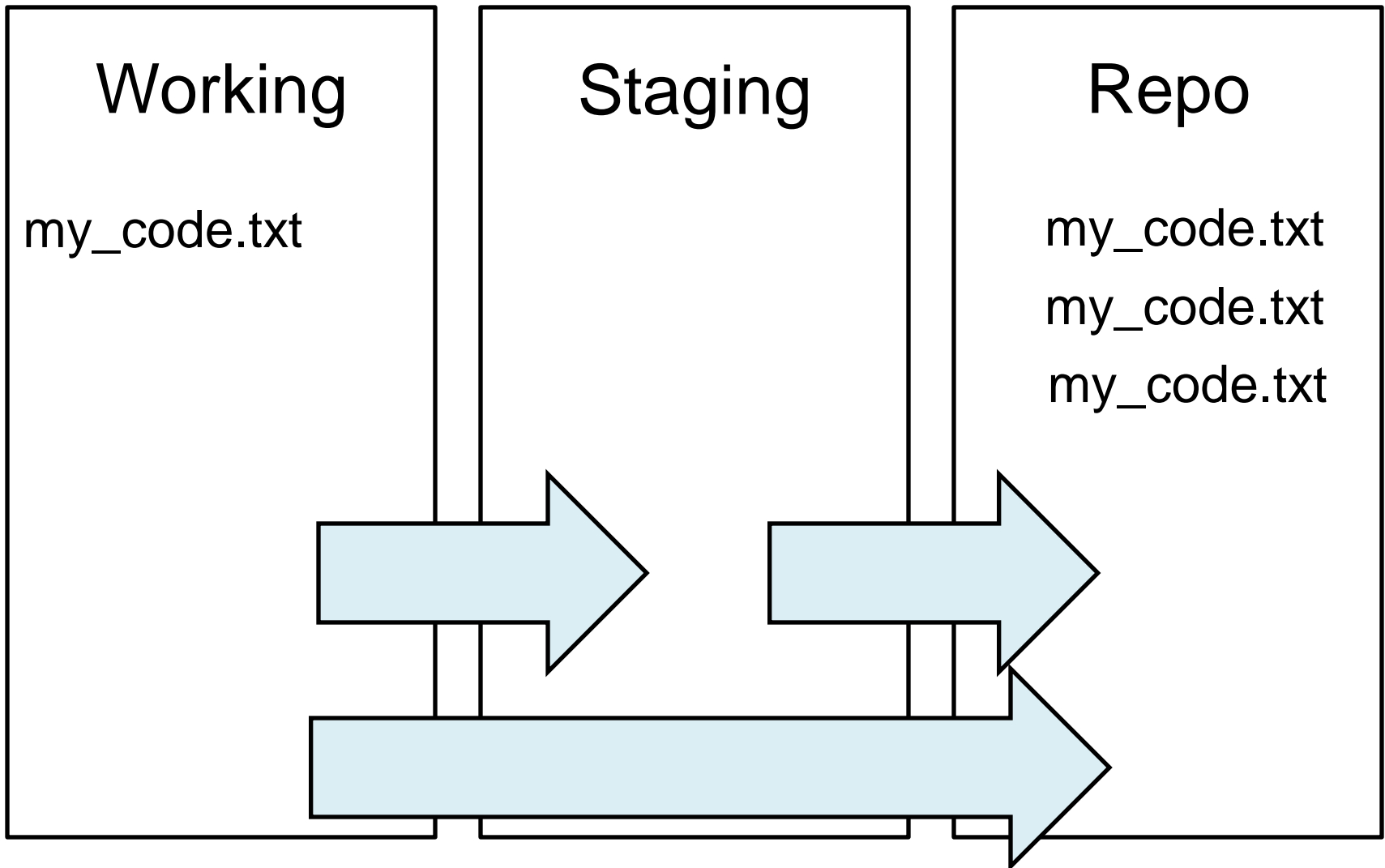
Repo

my\_code.txt

my\_code.txt

my\_code.txt

# Local Git Workflow



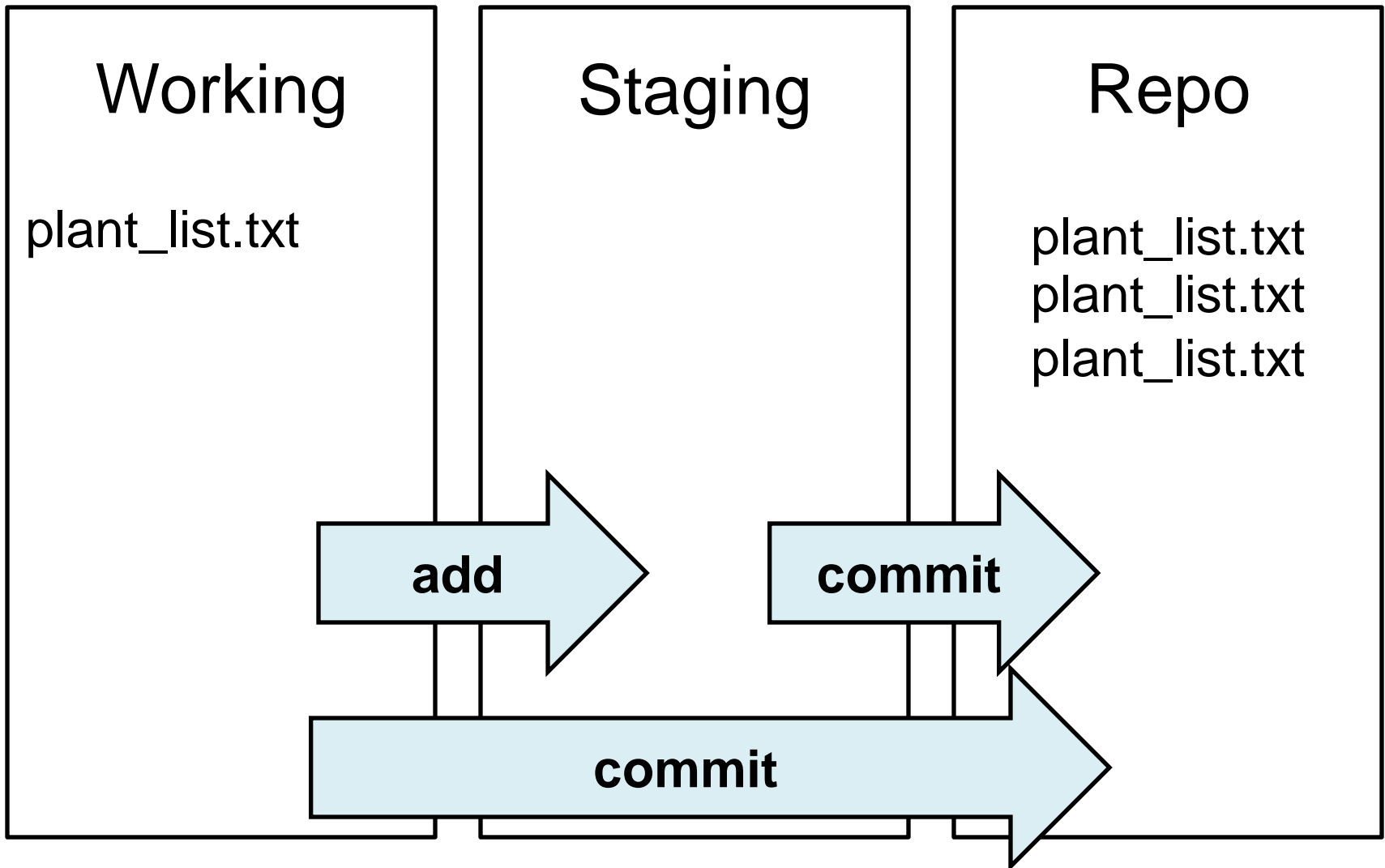


# First Commit: Follow Along Demo





# 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 PB weecology's wiki / Fro... RStudio Sign In danmcglinn — Bitbuc... weecology (Weecology) dmcglinn's Profile - Gi...

**Commit Logs From Last Night**  
because real hackers pivot two hours before their demo

**Fuckin' fork me**

	03/20/13 10:56 PM	fuck cells
	03/20/13 10:48 PM	fuck it, stay with pygments for now
	03/20/13 10:45 PM	Still fucking around with getting fields right
	03/20/13 10:40 PM	Fucking method visibility in PHP!!! At least make it consistent in different versions! Made all public!
	03/20/13 10:39 PM	shits broke

This thing tweets at @CLFLN

Created by @abestanway

# Bookkeeping

\$ `git init <repo_name>`

- Creates git repo

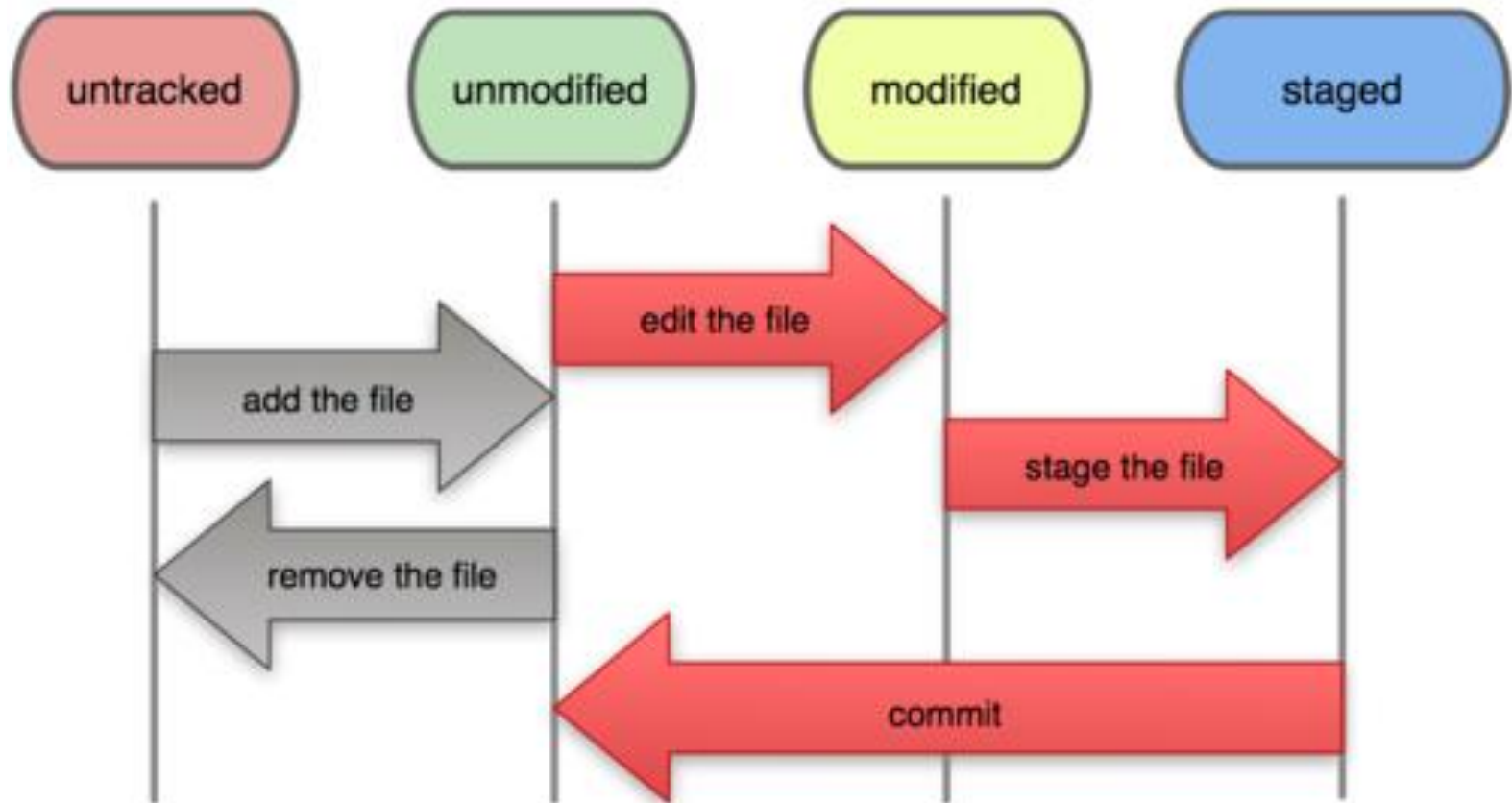
\$ `git status`

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

\$ `git log`

- Provides history of commits to the repo

## File Status Lifecycle



# First Commit: Do It Yourself



- Add new observations to **plant\_list.txt**
- Create a new file called **habitat\_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 **habitat\_list.txt**
- Create a directory called **data**
- Move the file **plant\_list.txt** to the directory **data**



# Finding What's Different: Follow Along Demo

Go away, Arial.

a

But, Helvetica!

a

# 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>
```



# Finding What's Different: Do It Yourself

Go away, Arial.

a

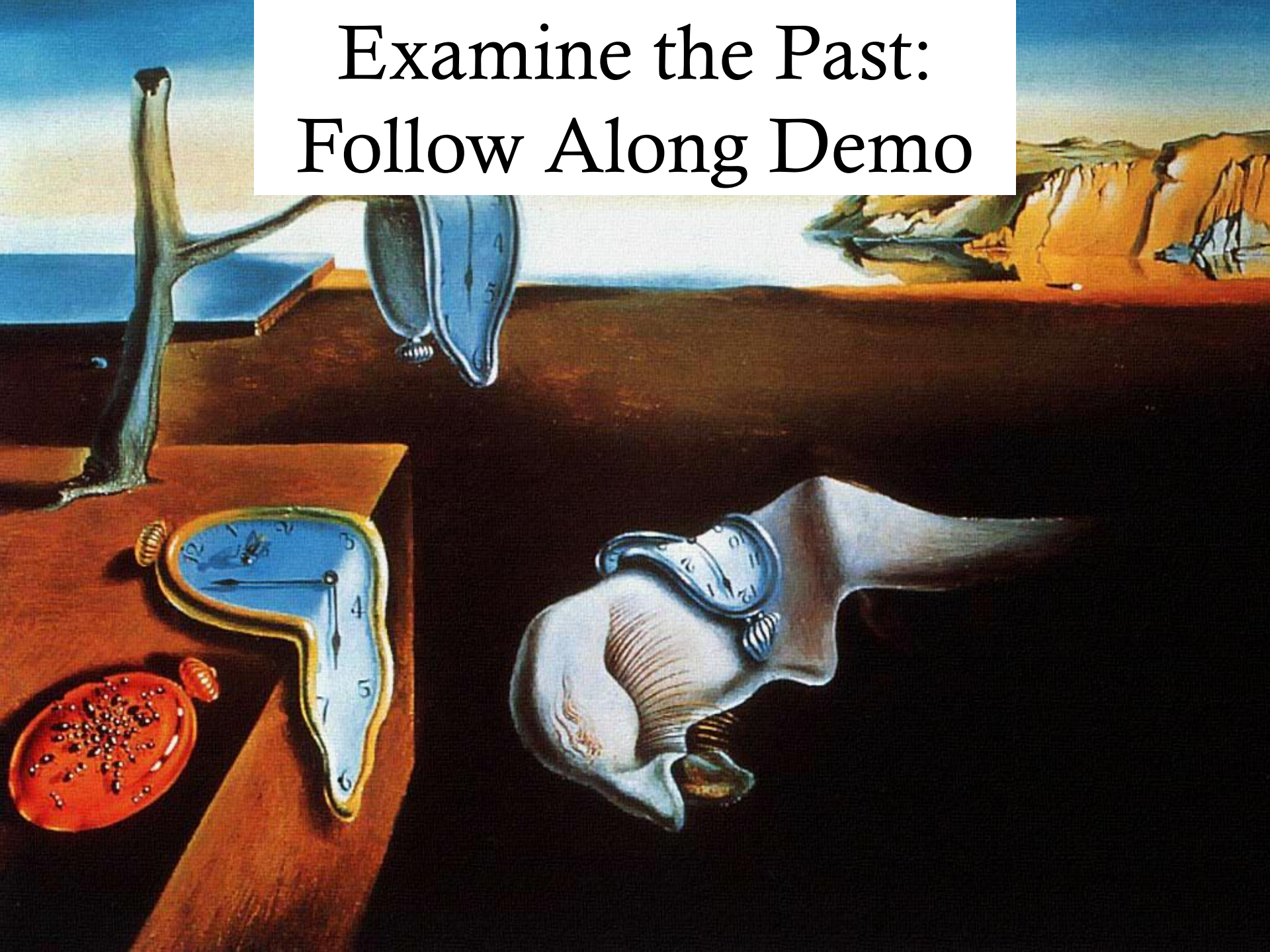
But, Helvetica!

a

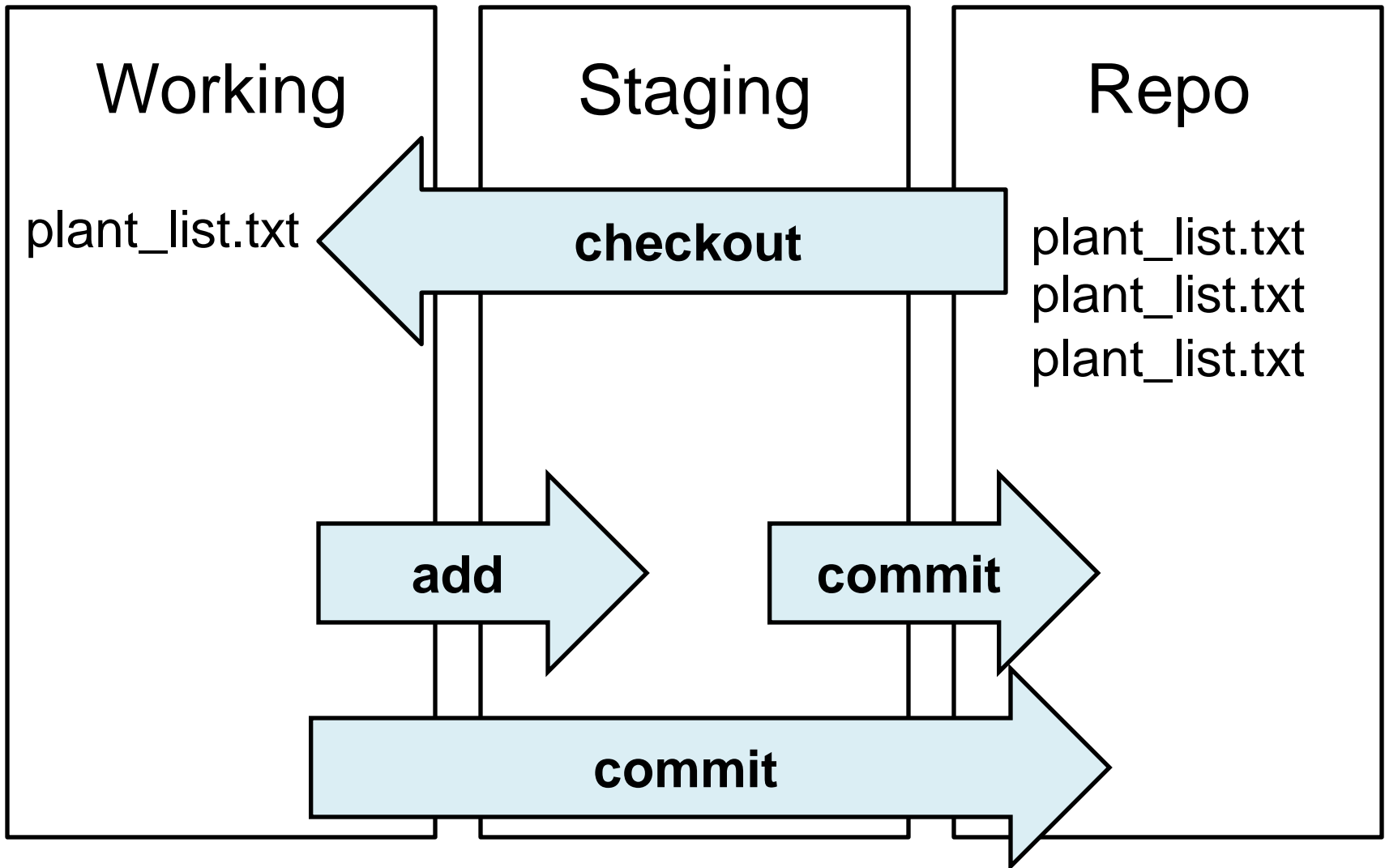
- Modify **plant\_list.txt** by adding a species to it
- Examine using `git diff` to examine what you changed
- Examine how your changes differ from the initial creation of **plant\_list.txt**



# 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

\$ `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>`

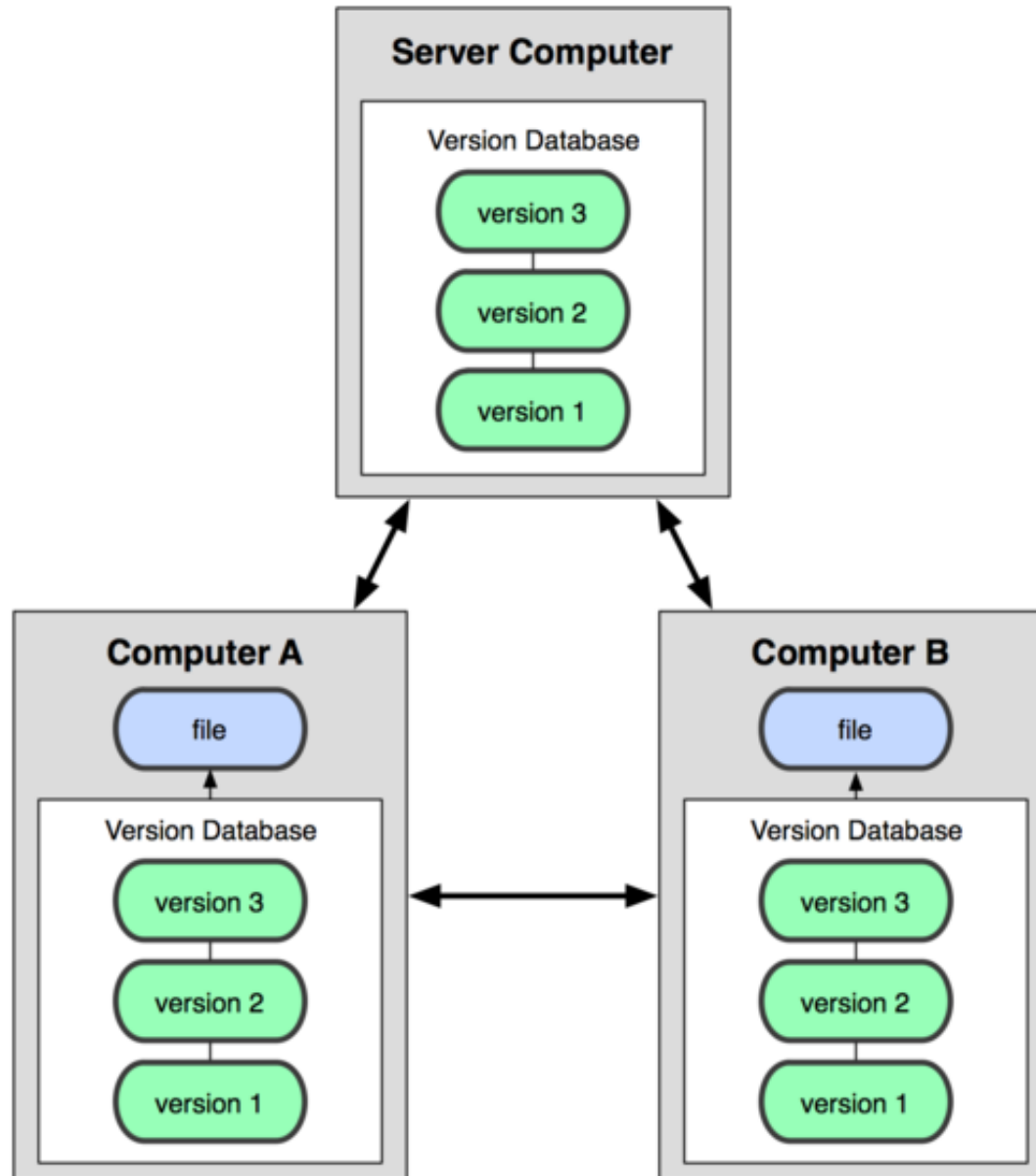
- Rolls back committed changes but does not omit them for all files to a specific version of the repo



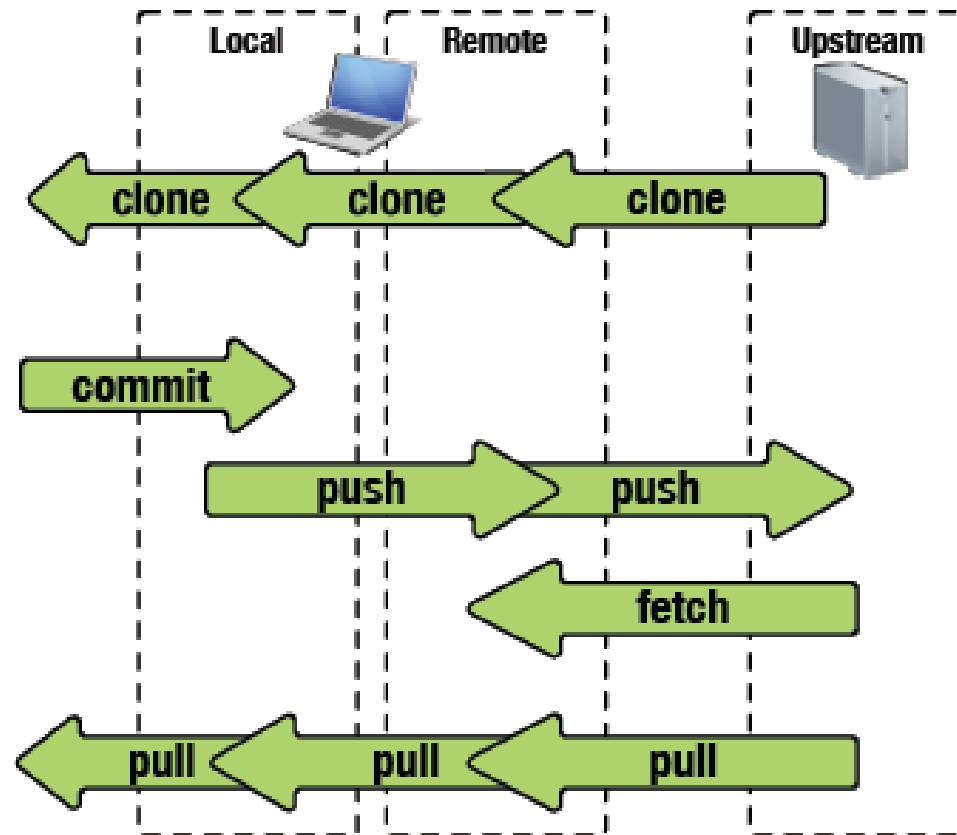
# Turn Back Time: Do It Yourself

- 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

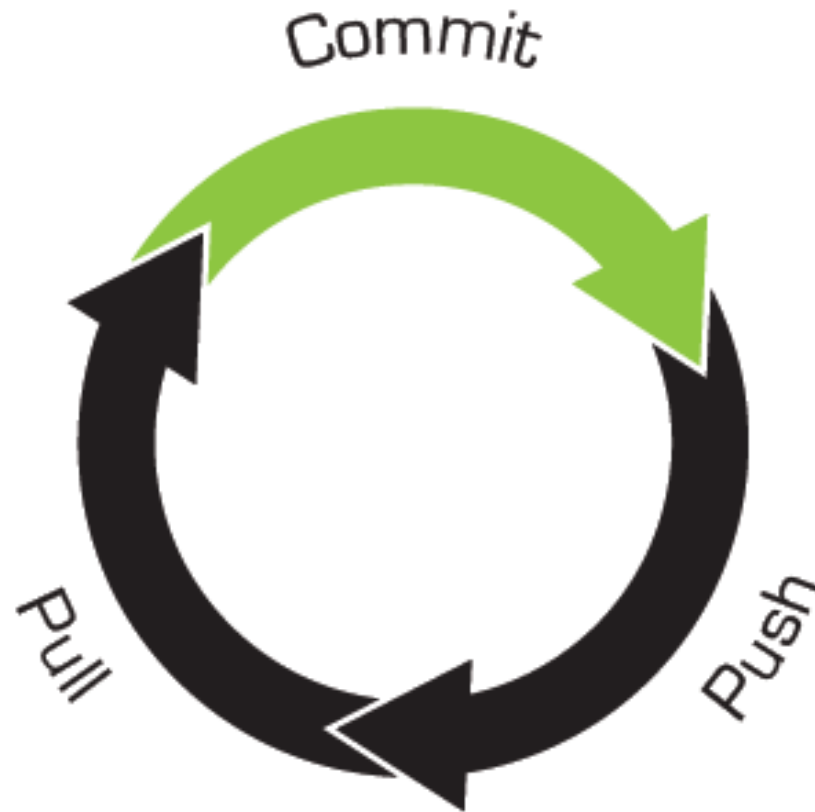
# Git Local & Remote Structure



# Git Remote Workflow



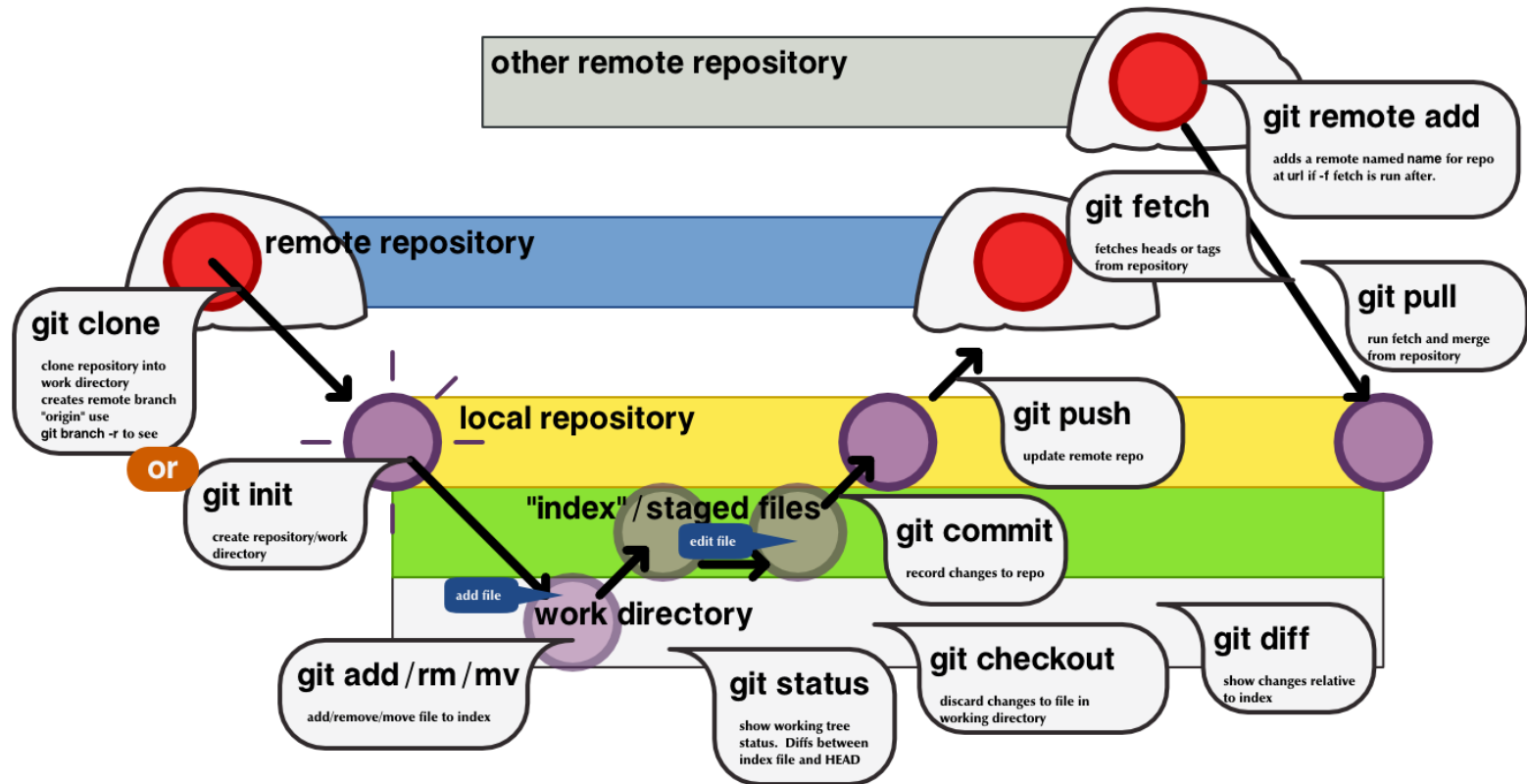
# Git Workflow Cycle



# Git Hub Demo

**github**  
SOCIAL CODING





## GIT Visual cheat sheet

