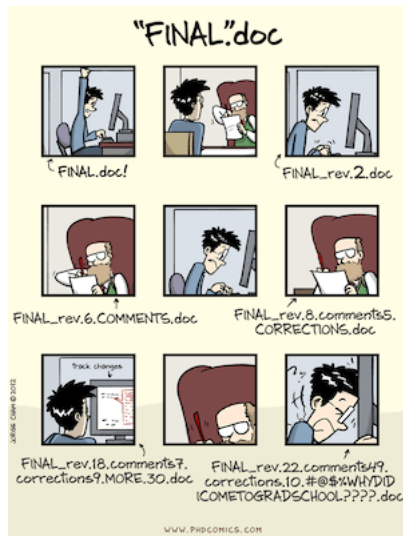# Version Control with Git/GitHub

Ying Jin

March 9, 2023

Introduction

- Version Control
    - Tracking and managing changes
    - Create reproducible analysis
- Git: Free and open source **distributed version control system (DVCS)**
- GitHub: Internet hosting service of Git

"FINAL".doc

Set up Git on your device

- MacOS: open terminal and run the command git −−version
- PC: Git Bash
    - Download and install Git For Windows package:
      `https://gitforwindows.org/`
- Additional resources
  `https://www.atlassian.com/git/tutorials/git-bash`
- Set up your GitHub account

Configuration

- Three levels of configuration: system, global (user), local (repository)
  - Check existing settings: git config −−list
  - Look for configuration files: git config −l −−show-origin
- Associate your local Git configuration with your name and Email
  - git config −−global user.name *your user name*
  - git config −−global user.email *your GitHub email*

Authentication

Protocols

- SSH: Easy, secure and free
- HTTP: Need to renew personal token often
- Git: Not secure. Rarely used alone.

Check existing SSH keys

- cd ~/.ssh: go to where git stores SSH keys
- ls: list all files in the directory
- Look for a pair of files named "id_xxx" and "id_xxx.pub"

Generate new SSH key

- Put in terminal:
  `ssh-keygen -t ed25519 -C "your_email@example.com"`
- Put in a secure passphrase (remember it!)
- Add your key to ssh-agent
    - Start ssh-agent in the background: `eval "$(ssh-agent -s)"`
    - Check if file exists in default location: `open ~/.ssh/config`
   - If doesn't exist, create file: `touch ~/.ssh/config`

Generate new SSH key

- Add your key to ssh-agent
  - Once .ssh/config is open, edit the file to include:

    ```
    Host *.github.com
        AddKeysToAgent yes
        UseKeychain yes
        IdentityFile ~/.ssh/id_ed25519
    ```

  - Add key to agent and store password:
    ```
    ssh-add –apple-use-keychain ~/.ssh/id_ed25519
    ```
- Add key to GitHub:
  - Copy key: `pbcopy < ~/.ssh/id_ed25519.pub`
  - Settings -> SSH and GPG keys
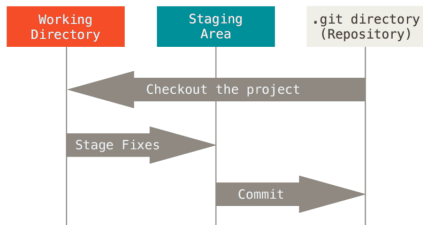
- Working tree: modify files
- Staging area: selected changes (added)
- Repository: store changes permanently (committed)

Start a repository

- Clone from GitHub with SSH/HTTP link

  ```
  git clone <link>
  ```
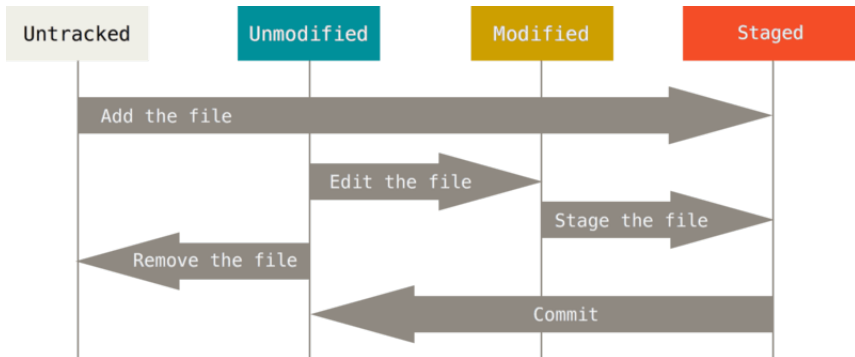
- Initializing a local Repository
  Make sure you are on the right working directory

  ```
  cd <filepath>
  git init
  git add .
  git commit -m <message>
  ```

Track changes in the repository

- Status of files: **Tracked** or **Untracked**
- Status of **tracked** files: **Unmodified, modified, staged**

Track changes in the repository

- Check changes in your repository

  git status / git status -s

- Track new files/stage modified files

  git add *filename*

- Commit staged files to repository

  git commit -m *your message*

Ignore files

- Greate a gitignore file

  touch .gitignore

- Open and edit the gitignore file

  open .gitignore

  Follow standard global patterns, for example:
  - Ignore all .xlsx files: *.xlsx
  - Ignore all files in a directory: Data/

Remove files

- Remove files completely

  git rm *filename*

- Untrack, but do not remove from the hard drive

  git rm −−cached *filename*

  - Move from tracked to untracked
  - Not ignored
  - Can be added back

Undo changes

- View commit history

  git log −−oneline

- Unstage/Unmodity files
  - `git status` will tell us how
  - Unstage files: Does not change content of the file

    git restore −−staged *filename*

  - Unmodify files: Careful! Any local changes will be gone!:

    git restore *filename*

Remote command lines

- Use on initialized local repository
- Check existing remote settings: `git remote` or `git remote -v`
- Clone a remote repository to your local device: `git clone` *url*
- Connect local repository to remote repository:

  git remote add *shortname url*

  - Specify a name for your remote server
  - Use the name (string) on the command line in lieu of the whole URL

Remote command lines

- Download data: git fetch *shortname*
    - Only download. Does not change local files.
    - Need to merge manually
- Download and merge: git pull
- Update remote repository   git push *shortname branch*
- Upstream setting:

    git push −−set-upstream *shortname branch*

    - Set a default remote branch for a local branch
    - fetch/push/pull can be used with no additional arguments

Remote command lines

- Inspect remote settings: `git remote show remote`

- Rename remote `git remote rename oldname newname`

- Remove remote repository: `git remote remove shortname`

- Merging conflicts
  Open the file and look for the following section

  ```
  <<<<<<<
  Current branch version
  =======
  Merged-in branch version
  >>>>>>>
  ```

- Use GitHub to track modifications

Git Branches

- Remember Git stores data in a series of commit
- Every branch has a pointer to a specific commit in this chain
- A "head" point indicates the branch you are currently on
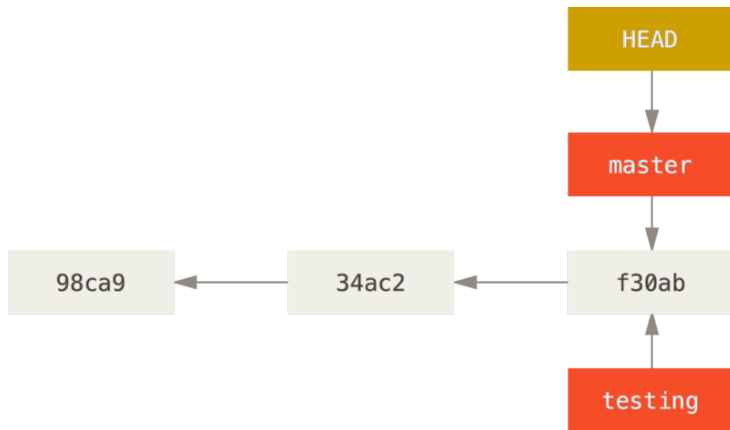
Branch operations

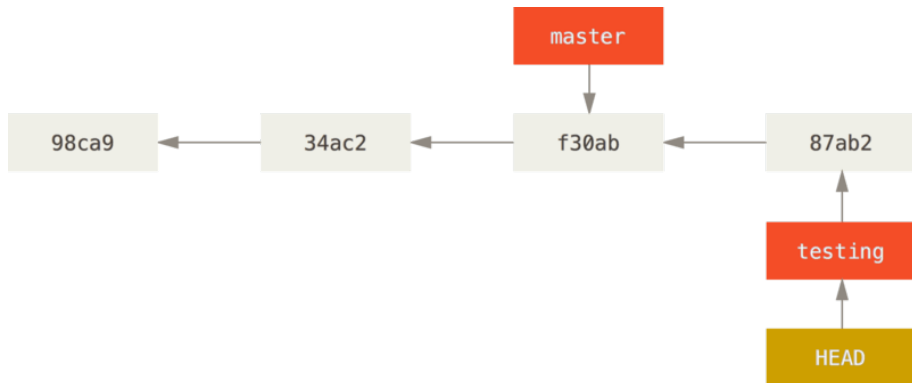- Create a branch

  git branch *branchname*

- Switch to a branch

  git checkout *branchname*

- See current branch
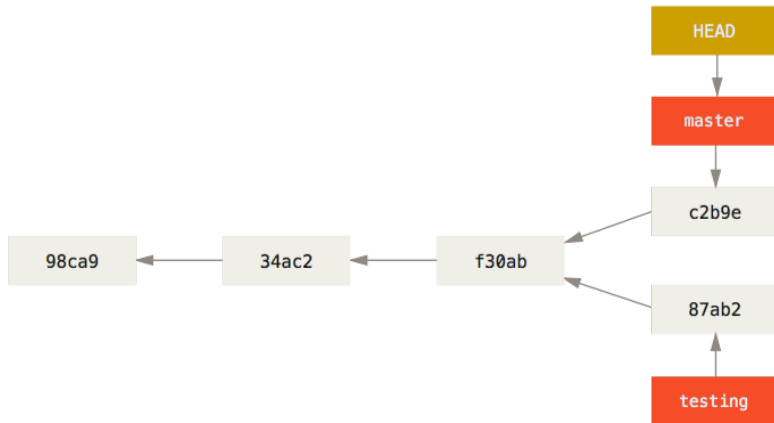
  git branch
  git log −−oneline −−decorate

Branch operations

When commits are made, the "head" pointer moves forward.
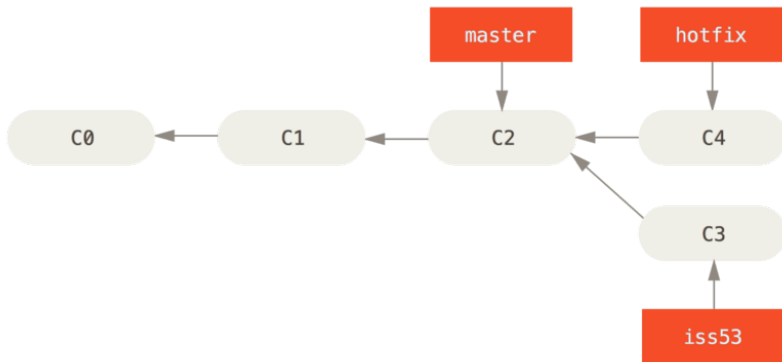
Branch operations

Your Git repository can accommodate divergent history

Basic merge

- Make sure you are currently on the branch to merge into
- Run command `git merge branchname`
- Fast-forward recursive merging

Branch management

Example:

- A long-running branch that is clean and "ready for publication"
- A long-running branch that is messy with all your mess around code
- Occasionally, create "topic" branches for particular fixes

> git branch -d *branchname*
> git push *remote* −−delete *branchname*