Version Control with Git and Github

...

Job Lindsen

Install git and create a GitHub account

Install git:

https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

And create a GitHub account:

https://github.com/

What is version control?

“Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.”

From:

https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

"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

Why version control?

- Records all changes.
- Easy to go back, and *undo*.
- Multiple version of the project, fast switching.
- Easy and disposable experimentation.
- Automatic merge.
- Conflicts are detected.
- See what has changed: diffs.

GitHub vs. git

GitHub

- Company
- Provides git hosting
- With a web interface (optional)
- By GitHub: Issue tracker, Pull requests, Wikis, Forks, Gists, github.com

git

- Distributed version control system
- Free and open source
- Not owned or developed by GitHub
- By git: Repositories, Branches, Remotes, Commits, Clones, Pushes, Merges, etc.

Some git/GitHub terminology

Repository (‘Repo’ for short): A set of files, with changes tracked over time.

Commit: Record changes to the repository, a ‘snapshot’

Push: Updates remote repo using local repo

Pull: Incorporates changes from a remote repository into the local repo

Branch: A different version of the same repo, e.g. ‘master’ and ‘testing’

Fork: A branch stored in another GitHub account

Merge: Join two or more development histories (branches/forks) together

Let’s get started!

I’ll show you 2 ways to set up a git repository:

1. Start locally on your own machine, upload to GitHub later.
2. Start with a repository (your own or forked) on GitHub and clone it to your local machine.

Command line git

All git commands start with ‘git’:

```
$ git <command> <options>
```

For example:

```
$ git init
$ git push origin master
```

Live Demo Time!

Command line git

git commands we have seen so far:

```
$ git init
$ git status
$ git add <file>
$ git commit -m “Message”
$ git log
$ git branch
$ git checkout -b new_branch
$ git merge
```

Making your local repo available on GitHub

- If you only want to keep track of your code locally, you don't need to use GitHub.
- If you want to work with a team, you can use GitHub to collaboratively modify the project's code.
- GitHub is also a great place to:
  - show the world what projects you are working on
  - share projects that might be of interest to others
  - keep track or contribute to open source software (e.g. check out ‘pandas’ on GitHub)

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner **Repository name**

JPLindsen / repo1

Great repository names are short and memorable. Need inspiration? How about **upgraded-engine**.

Description (optional)

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

Private
You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** | Add a license: **None**

Create repository

JPLindsen / repo1

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Wiki | Pulse | Graphs | Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/JPLindsen/repo1.git

We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# repo1" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/JPLindsen/repo1.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/JPLindsen/repo1.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

JPLindsen / repo1

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Wiki | Pulse | Graphs | Settings

No description or website provided. — Edit

3 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

JPLindsen commit — Latest commit 5167d65 22 hours ago

hello_world.ipynb — commit — 22 hours ago

Help people interested in this repository understand your project by adding a README. Add a README

git push/pull

Push updates to remote repo using branch of local repo:

```
$ git push <remote> <local_branch>
$ git push origin master
```

Pull incorporates changes from a remote repository into the current local branch:

```
$ git pull <remote>
$ git pull origin
```

Clone an existing GitHub repo onto local machine

```
$ git clone https://github.com/YOUR-USERNAME/YOUR-REPOSITORY
```

For example:

```
$ git clone https://github.com/JPLindsen/repo1
```

Command line git

git commands we have seen so far:

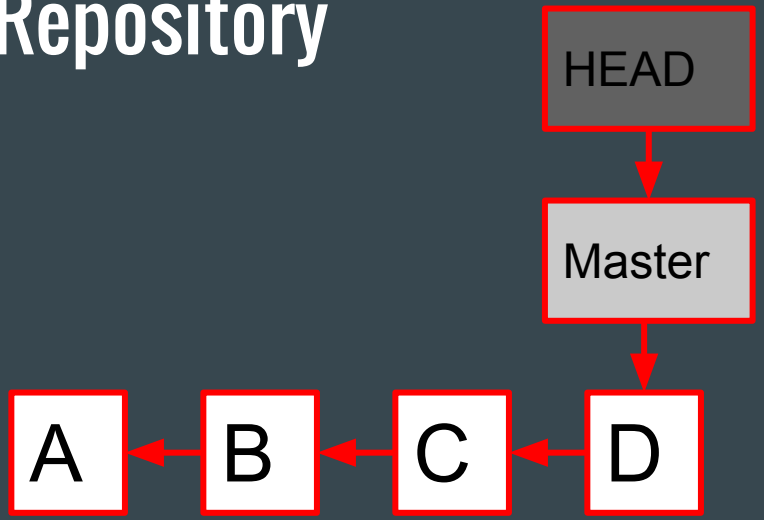
```
$ git init
$ git status
$ git add <file>
$ git commit -m “Message”
$ git log
$ git branch
$ git checkout -b new_branch
$ git merge
```

```
$ git push
$ git pull
$ git clone
```

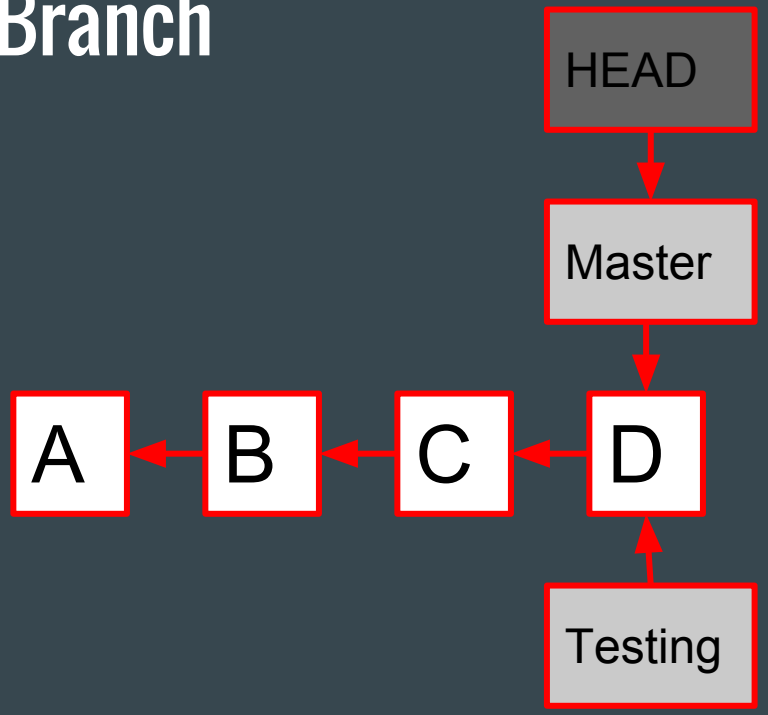
For a full git reference see:

https://git-scm.com/docs

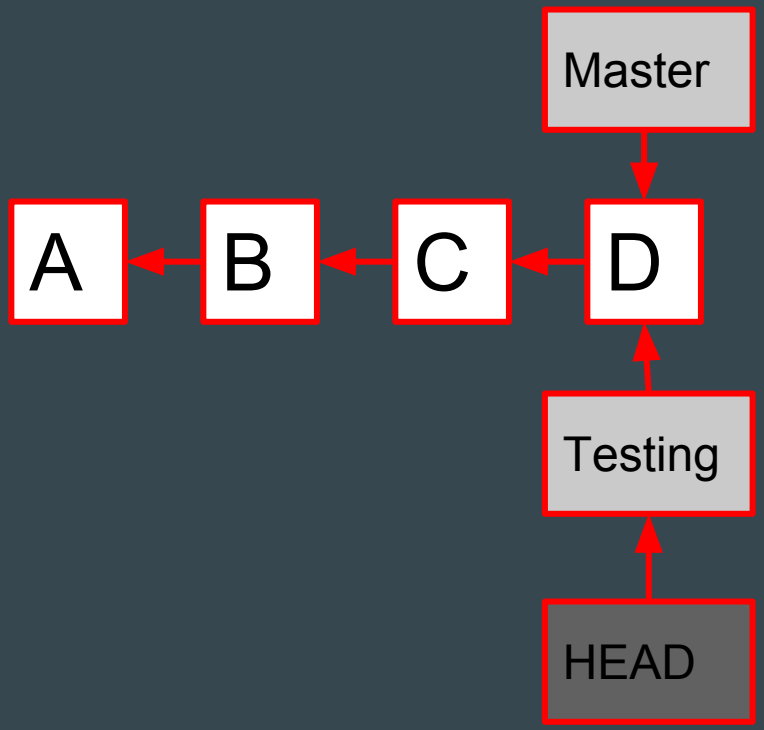
Repository



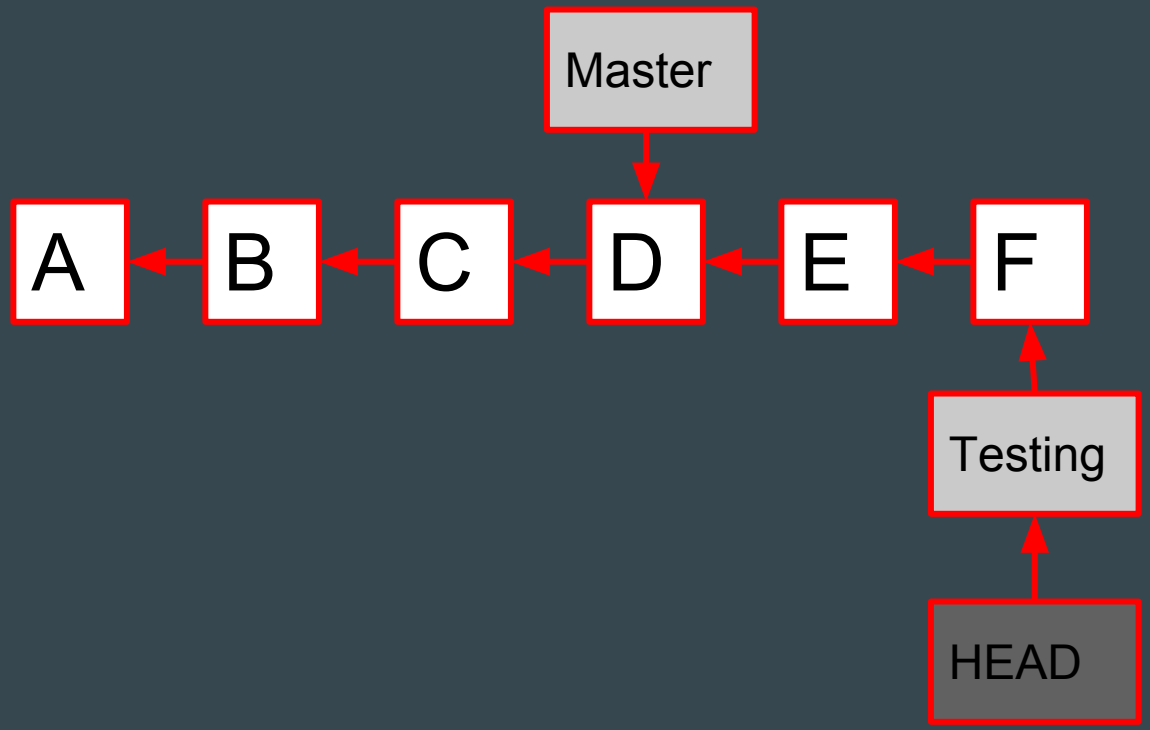
Branch



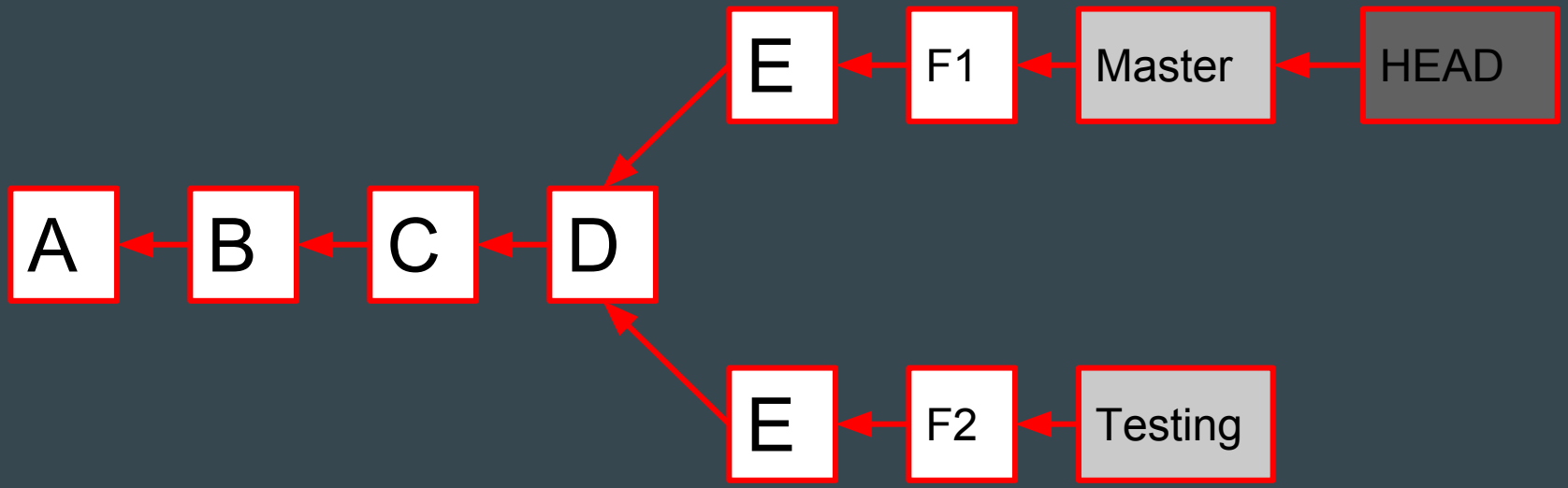
checkout Testing



`commit` in branch

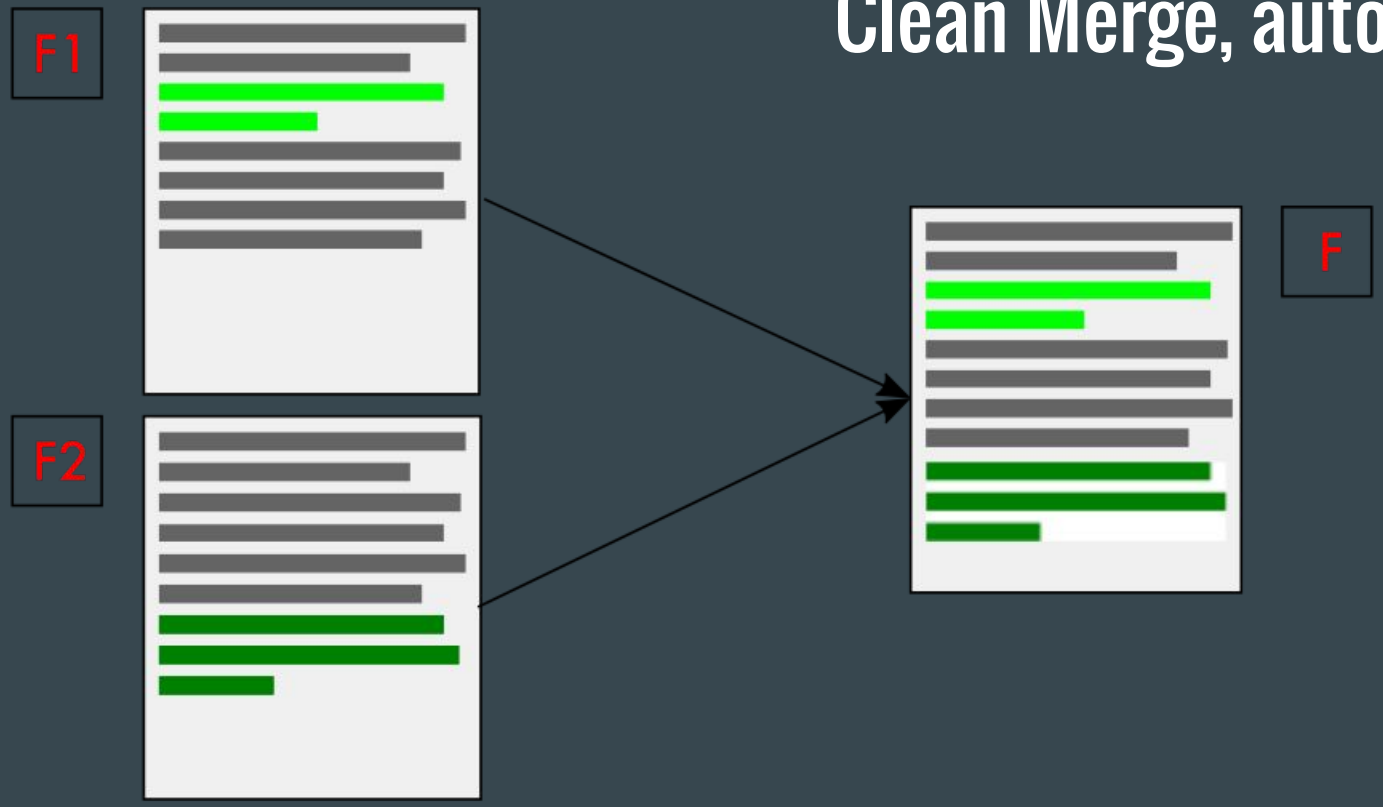


`commit` in both branches

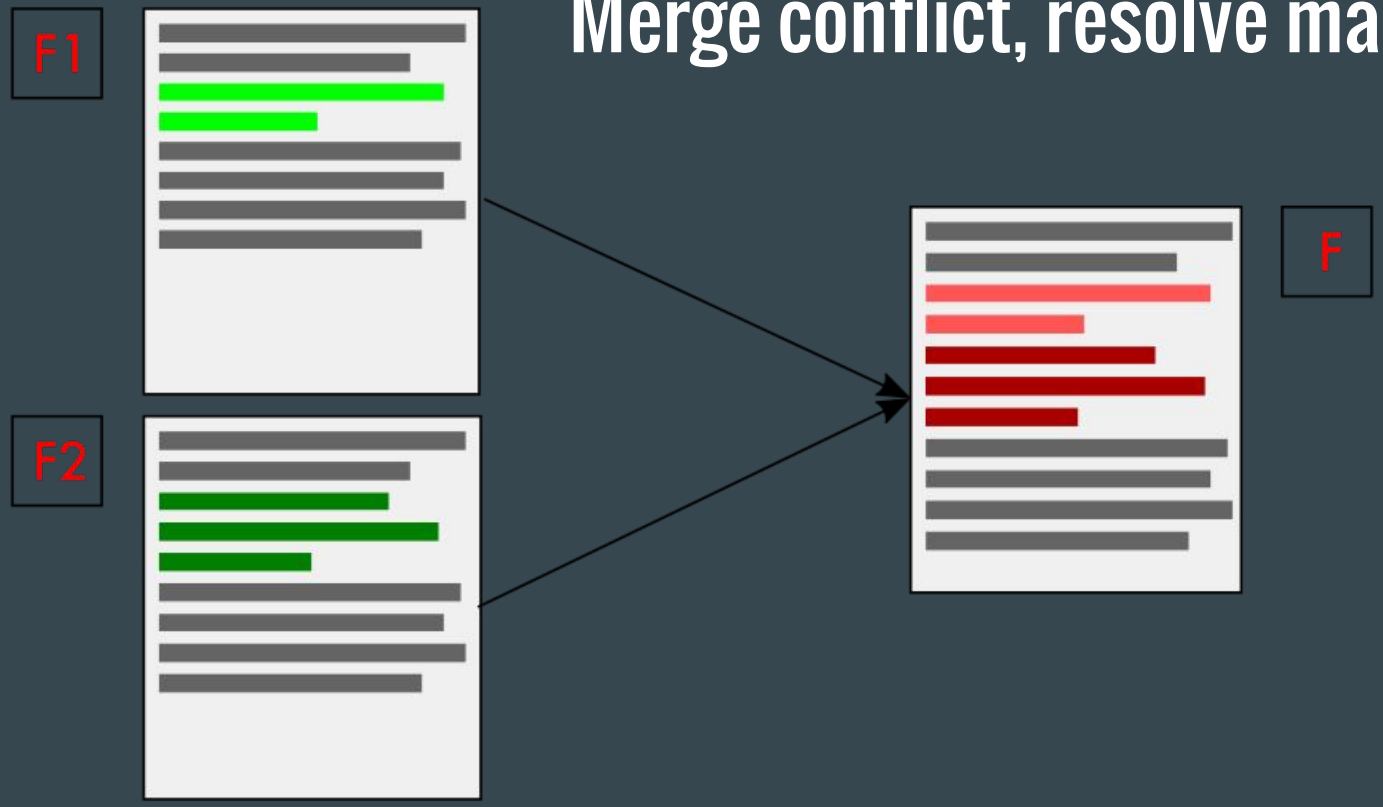


What happens when we `merge`?

Clean Merge, automatic



Merge conflict, resolve manually



Master branch after `merge`

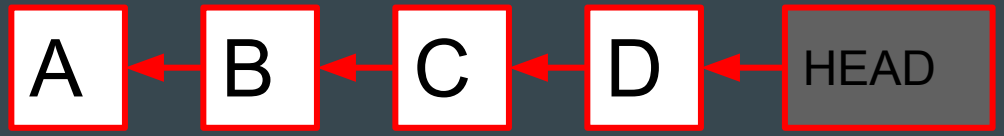
Collaborating on GitHub: Forks and pull requests

- Create a fork of somebody else's repo.
- Make changes, commit
- Create a pull request
- Owner of repo reviews pull request
- If there are no issues with request, merge forks and close request

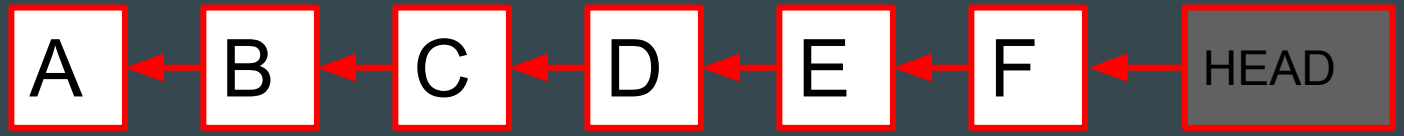
Demo Pull Request

Forks

JPLindsen/repo1:

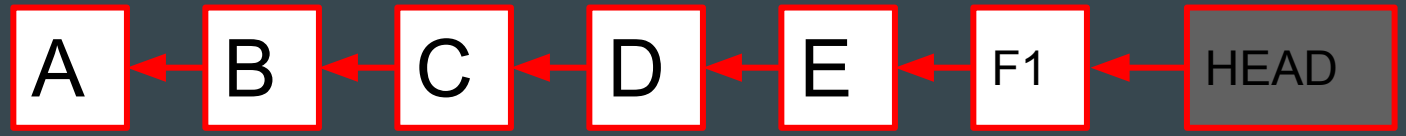


SomeoneElse/repo1:

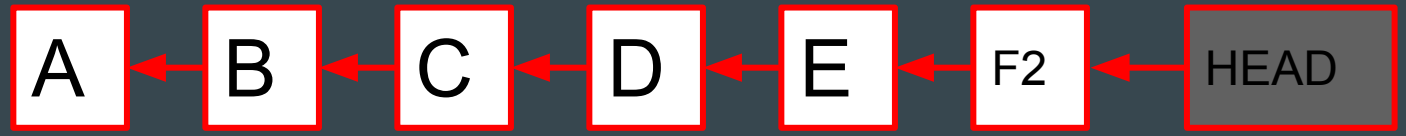


Forks

JPLindsen/repo1:



SomeoneElse/repo1:



Issues Demo

Writing good issues

- Short and specific subject line. Not “pandas bug”!
- Labels: *bug*, *enhancement*, *help wanted wontfix*, etc. custom labels
- If it is a bug, provide a reproducible example
- Code formatting: use Markdown, indicate language
- Include relevant people: @JPLindsen

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Good Commit Messages

- First line <50 characters, concise summary
- Then empty line, and more detailed discussion
- Refer to issues, related commits, users involved
- Close issues with commit message: “This commit `closes #1`”
- Try answering the questions:
    - Why is this change necessary?
    - How does it address the issue?
    - What side effects does this change have?

Exercise 1: getting started with git and GitHub

- Make new directory on your local machine
- Initialise this directory as a git repo
- Add a file to your new directory
- Stage the file for a commit
- Commit the changes

- Set up a new repo on GitHub with the same name as your local directory
- Add the remote repo to your local machine
- Push the local contents of your repo to the remote Github repo

Excercise 2: Issues

- (Enable issues in your repo)
- Create a new issue in your repo
- Add a label to your issue: e.g. bug or enhancement
- Assign the issue to yourself
- Edit your repo to fix the issue, with `closes #{issue}`
- Check that the issue is closed

Exercise 3: Pull requests

- In pairs, fork each other's repo's
- Make a change to the forked repo, commit
- Open a pull request to merge your changes
- Review the changes and merge.