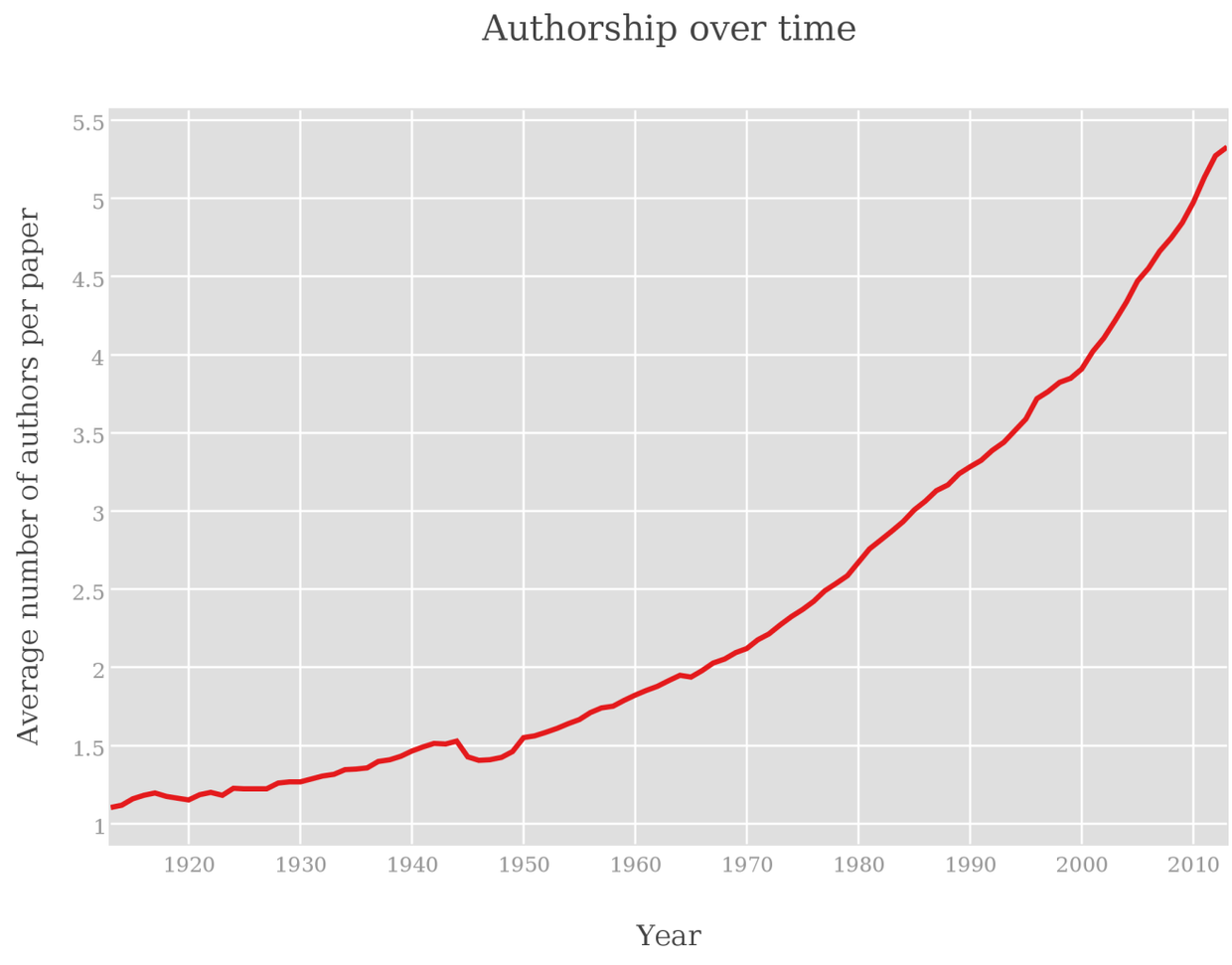
SASBi SC Symposium

Co-Lab training:
Mini Hackathon

Intro:

- Global collaboration is becoming more common



Aims

- Teach skills that will come in handy when collaborating
- Introduction to tools, terms, and concepts
- Get a feeling of what its like to be part of a big project
- Apply in a simulated mini-hackathon!

Waterfall methodology

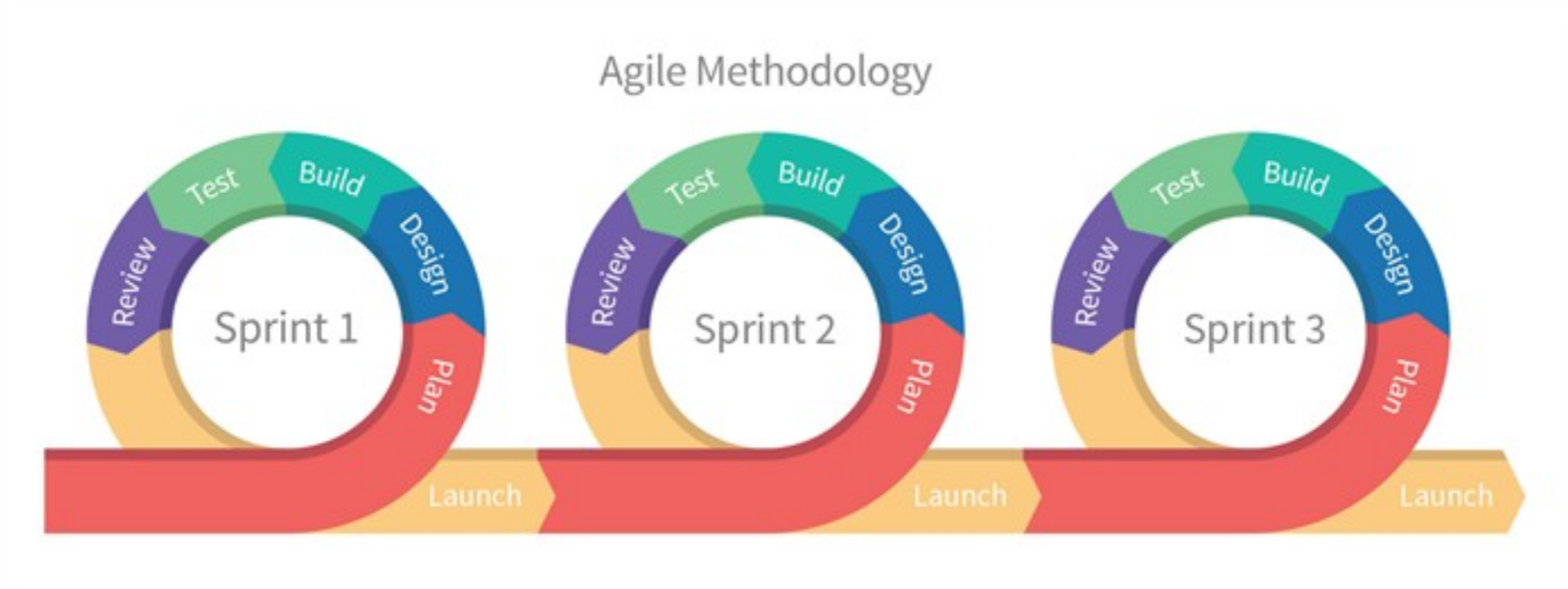
- Traditional development:
  - Long waterfall like process
  - Final product only released at the end
  - Committed to early design choices
  - Assumes all requirements are predicted at start

Enter Agile!

- Iterative, incremental software development method

- Get a working version out ASAP

- Work in “Sprints”



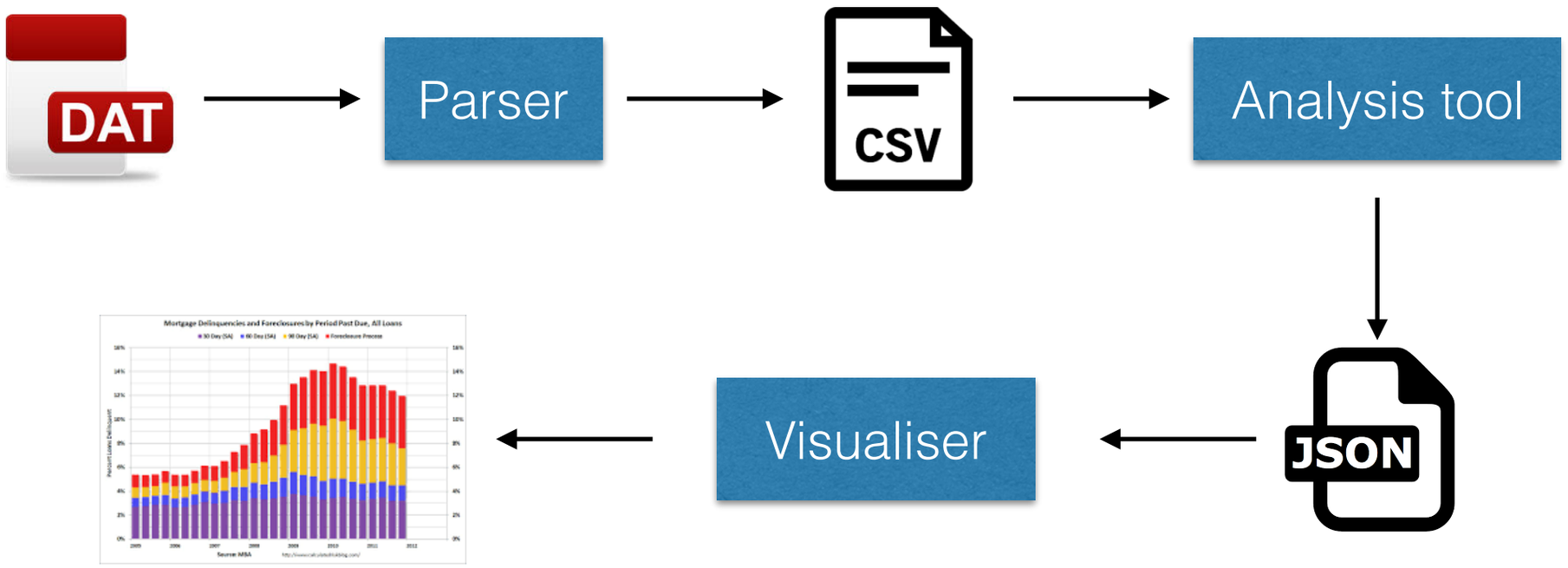
Agile

- Not the best way! Just another way to think about.
- Some reasons yay:
  - Often our hypothesis / method is wrong
  - We are often learning as we go along
  - Second time around it almost always better
  - Agile is fun! (Keeps people interested)

Asynchronous development

- Each part of the project designed in such a way that they may be developed independently

- Defined ways that the different parts interact



Function skeleton

- def some_analysis(file name):

  # Does things

  return list/dict of data

Function skeleton

- def some_visualisation(list/dict, out_name):

  # Does things

  export .svg

Github 

- A place to store projects
- Allows everyone access to the latest versions
- Allows asynchronous development
- Keeps track of changes
- Great for the CV

Brief:

- I have a .pcl file
- It contains gene expression fold change data for multiple experiments for multiple genes
- I would like a tool to analyse it
- Do some stats
- Create some visualisations

Brief:

- The tool requires:
  - Main platform
  - Input parser
  - Tools
  - Visualisers

Brief:

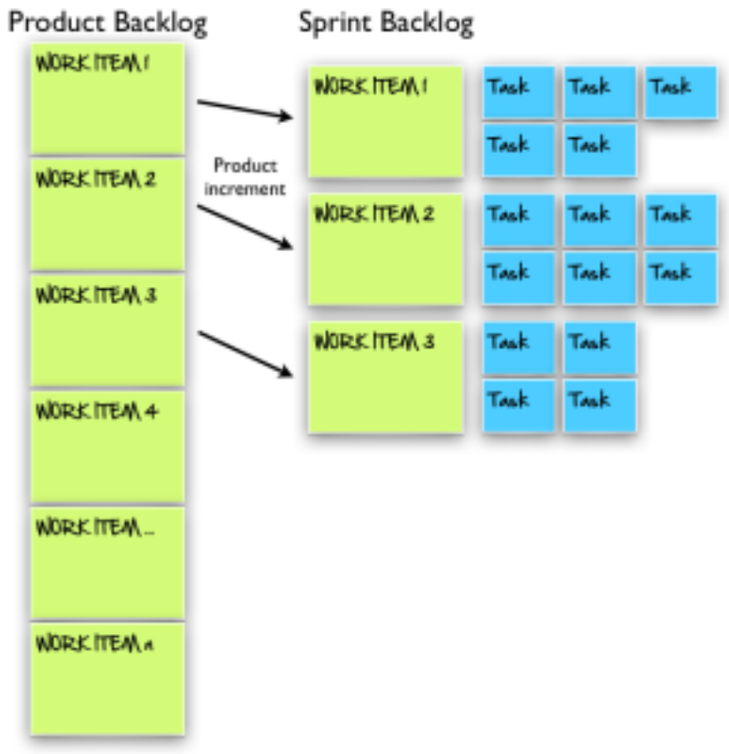
- 35 people
- 7 teams
- Try get a mix of experience
- Each with a spokesperson
    - Updates the backlog
    - Reports at end of sprints
    - Liaises with other group spokespersons

Phase 1: Planning and design
Dev team, assemble!

- Planning
- Design
- Working with Trello

Job boards / Project backlog

- Project backlog -> Prioritised backlog -> Sprint backlog



Divide and Conquer

- Divide into groups
    - One group needs to create the platform
    - Select a spokesperson for the group
    - Discuss what functions your team wants to create
    - Add all of them to the Backlog

Phase 2: Build
Lets make it happen!

- Intro to Git
- Create team branch
- Coding of functions

Intro to Git

What is Git?

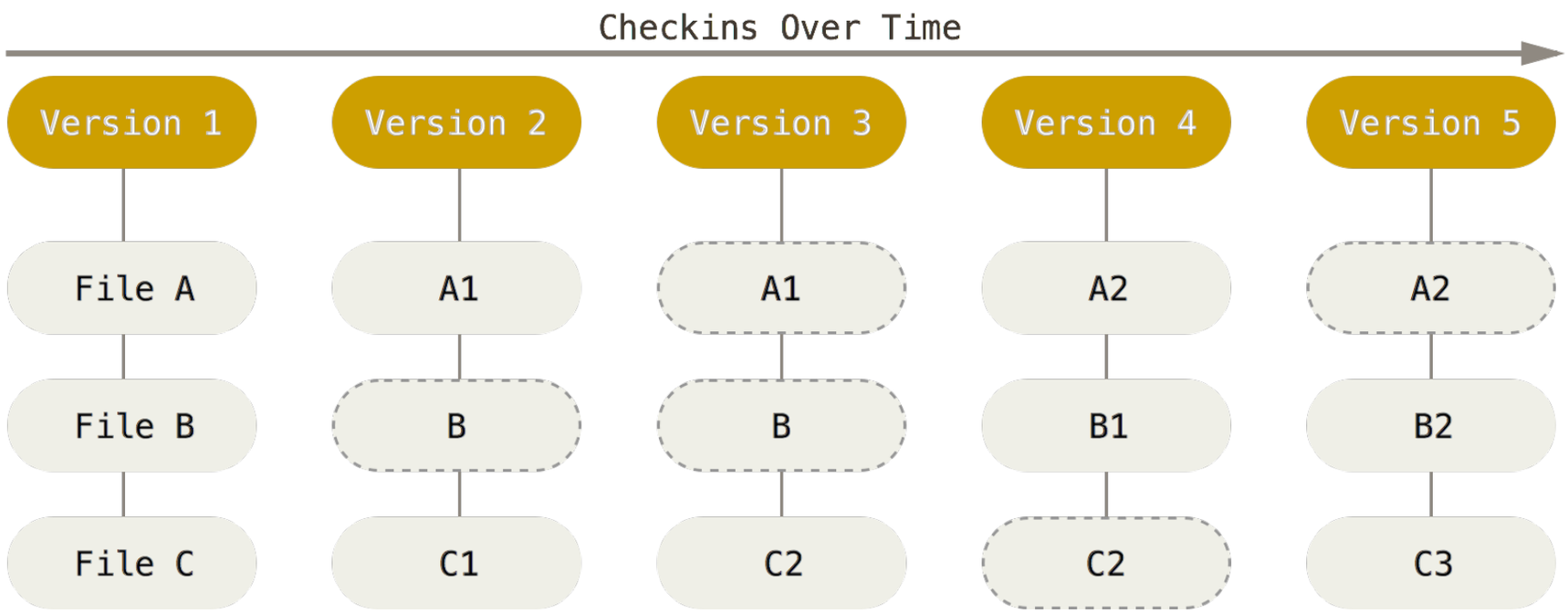
- A version control system (VCS)
  - Git, Subversion, Perforce
- What is a VCS?
  - Keeps multiple version of the same file
  - Allows you to undo changes

What is Git?

- Used for:
  - Working collaboratively on the same project
  - Keeping work safe
  - Keeping multiple versions of the same file

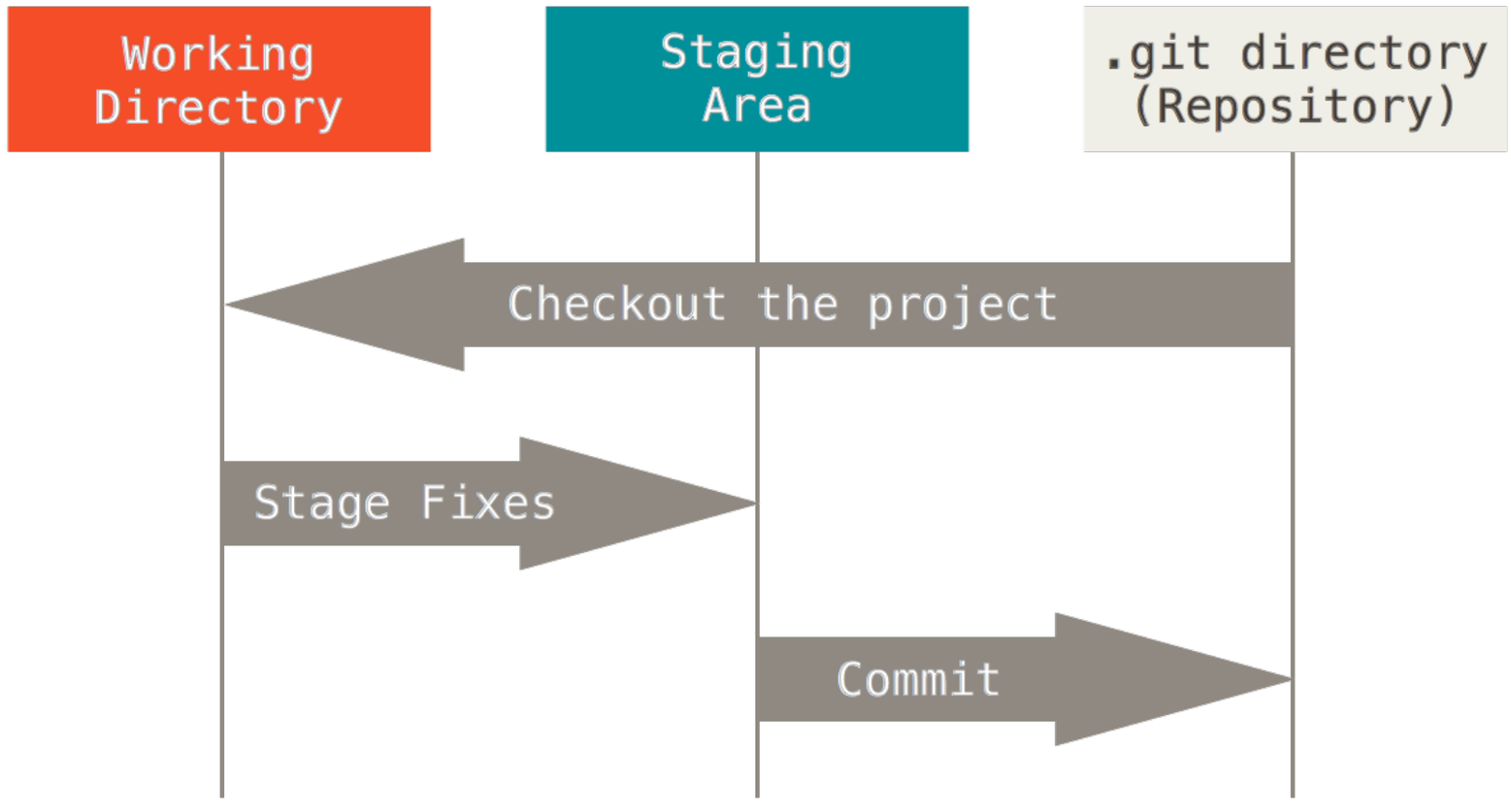
What is Git?

- Git stores streams of snapshots
- Other VCS systems store changes



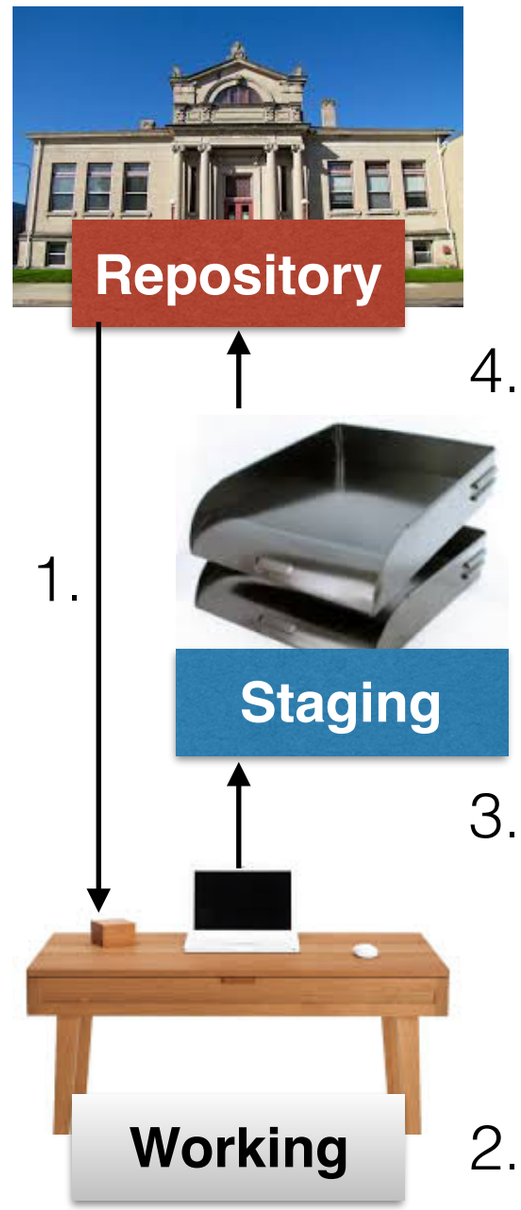
Three states in Git

- A file can be committed, modified, or staged



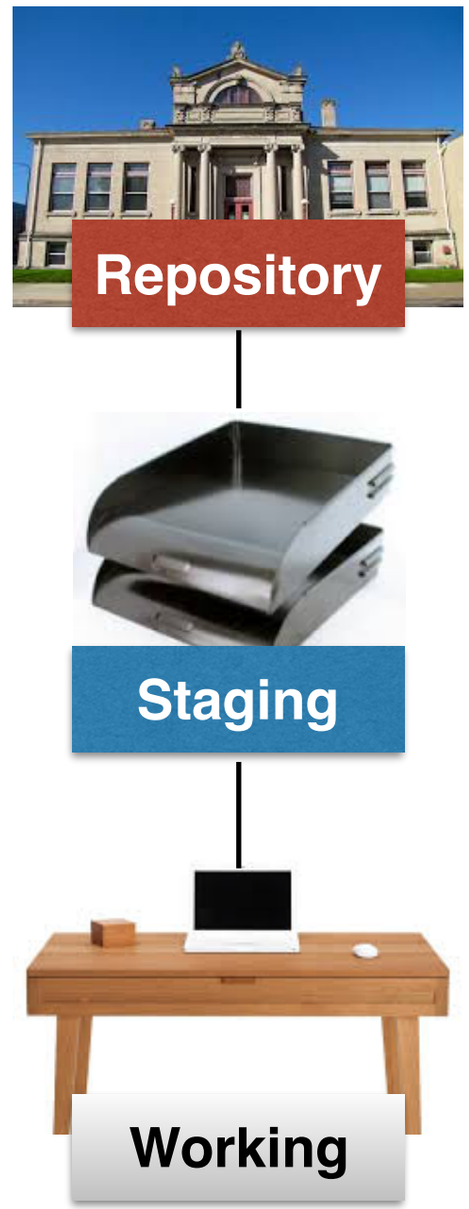
Git workflow

1. Pull latest version from the repository
2. Modify the files in your working directory
3. Stage the files you have modified
4. Commit the files in your staging area to your Git repository



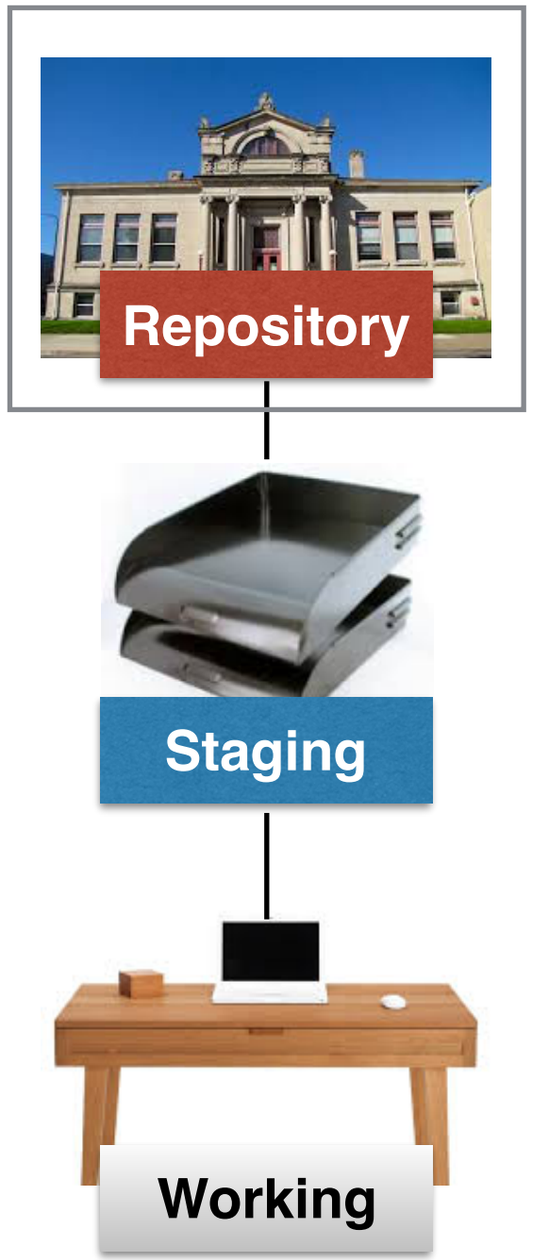
Git usage

- Can be used from:
  - The command line
  - A browser
  - A GUI
  - Built into interactive development environments (IDEs)



Repositories

- Local
  - More control
  - For sensitive data
- Remote
  - Github, Bitbucket
  - Safer backup-wise
  - Files are not private (Free version)



Startup: First time

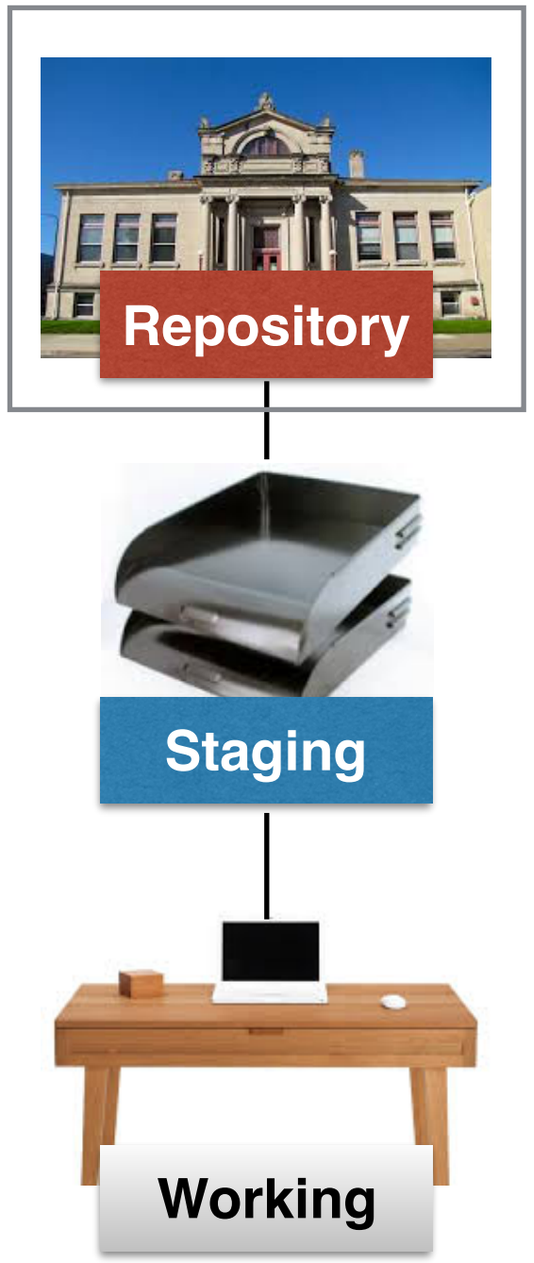
- `$ git config`
  - Configuration stored at 3 levels
    - System `(--system)`
    - User `(--global)`
    - Directory

Startup: First time

- Setting the user identity
  - `$ git config --global user.name "John Doe”`
  - `$ git config --global user.email johndoe@example.com`
  - These must be the same as your Github details

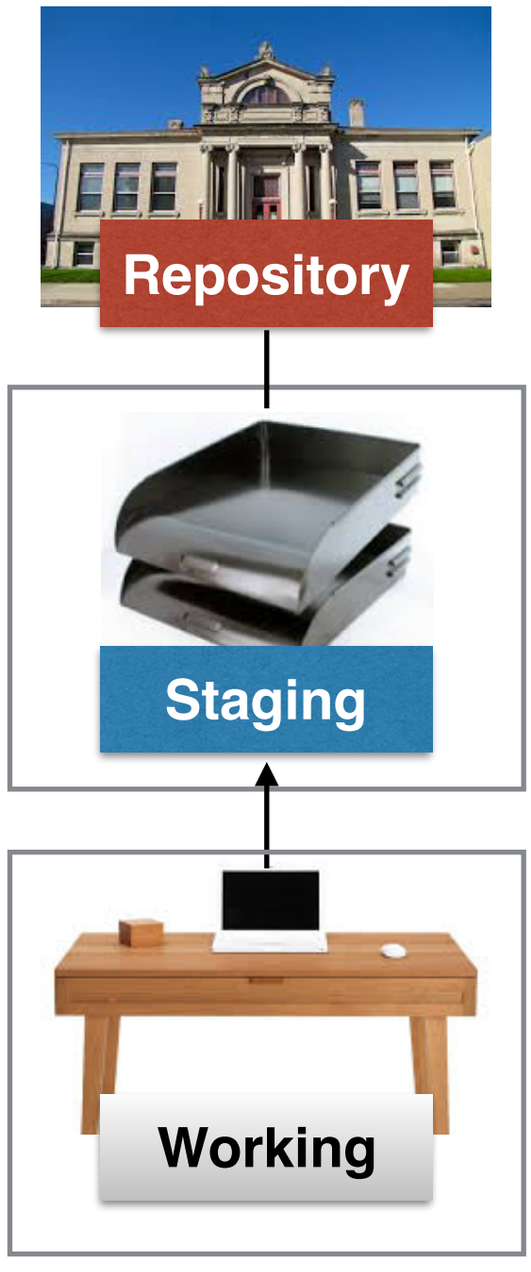
Creating a repository

- Creating a local repo:
  - `$ git init`
- Creating a remote repo:
  - https://github.com



Adding files for VC

- Add files to be tracked:
  - `$ git add <file>`
- Checking what is the state of the files in the dir
  - `$ git status`

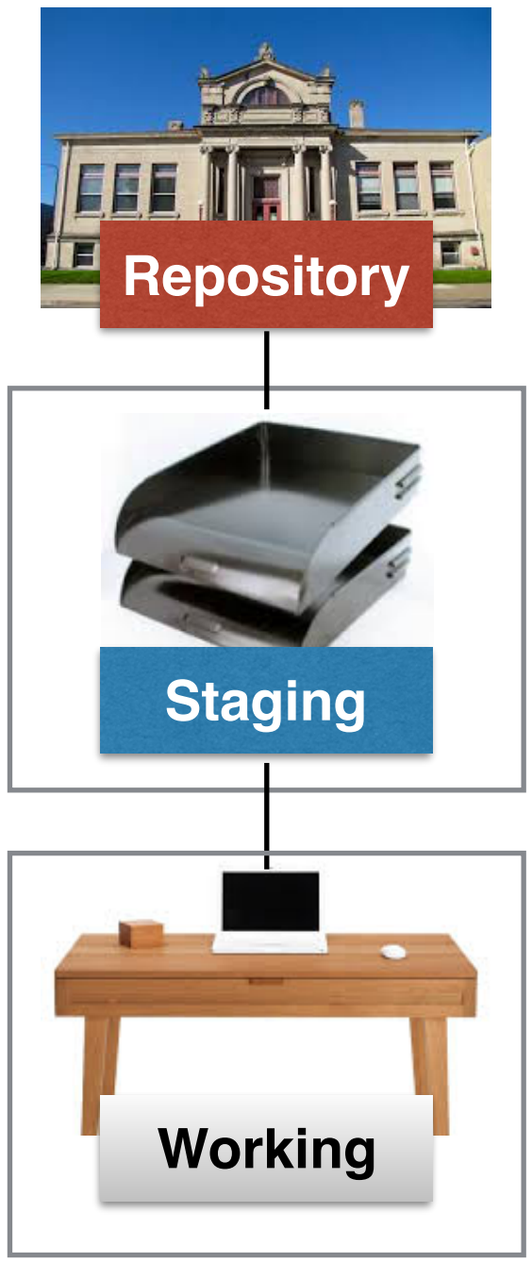


Git usage

- To move a file to the staging area from the working area
  - `$ git add <file>`
- To move a file from the staging area to the local repository
  - `$ git commit`

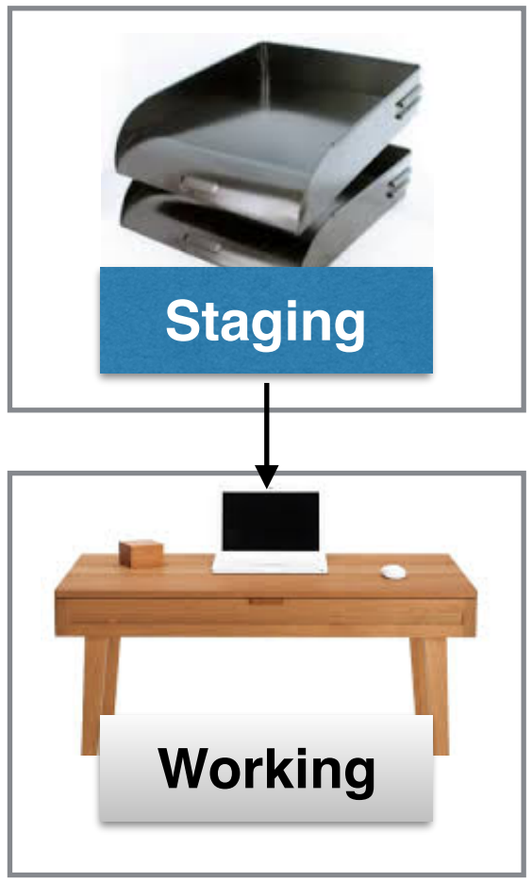
Git usage

- If you start working on a staged file:
  - You will be working on a different file to the one that is staged
  - If you run git commit, the staged file, not the one you are working on will be committed to the repo

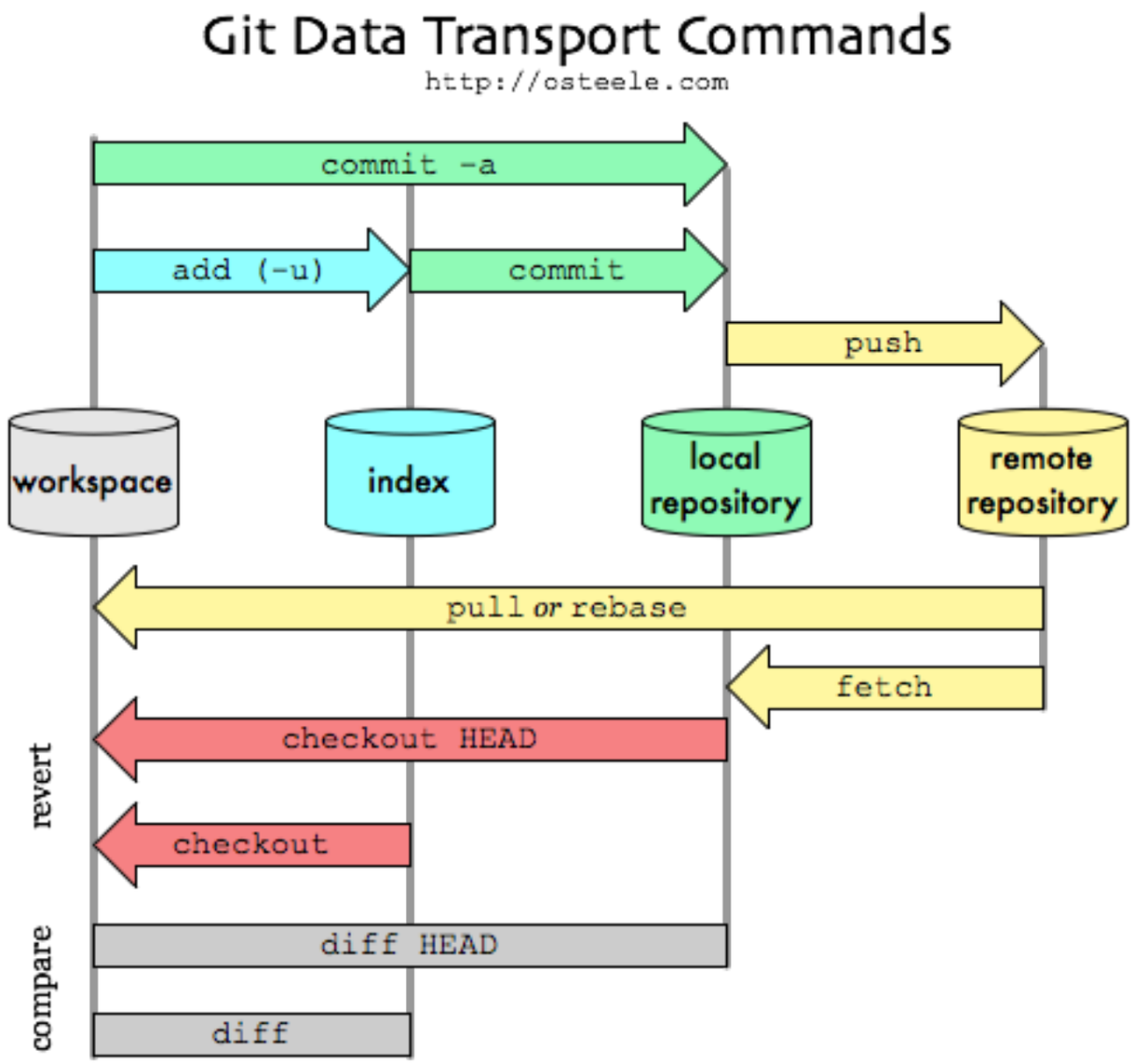


Git usage

- To undo the changes you have made and revert to a previously committed version
  - `$ git checkout`

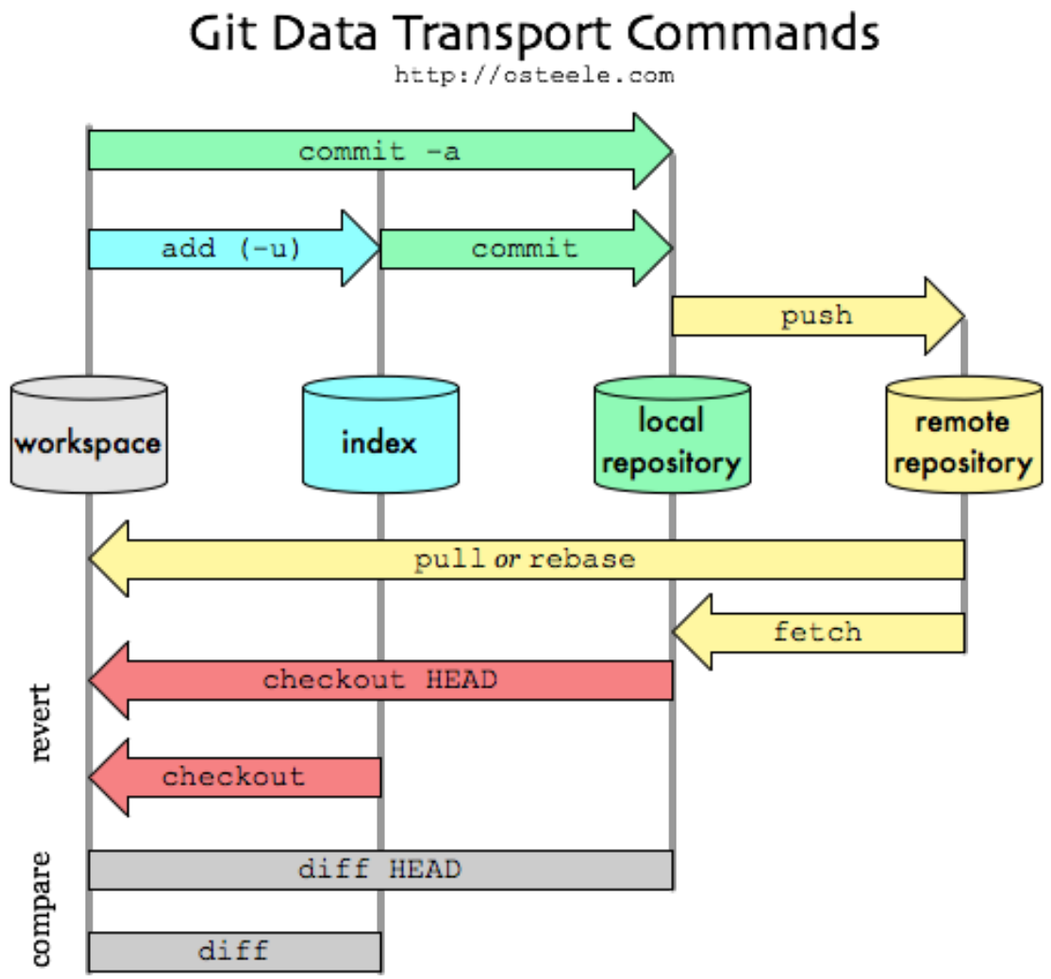


Remote repositories



Remote repositories

- Cloning an existing repo:
  - `$ git clone <url>`

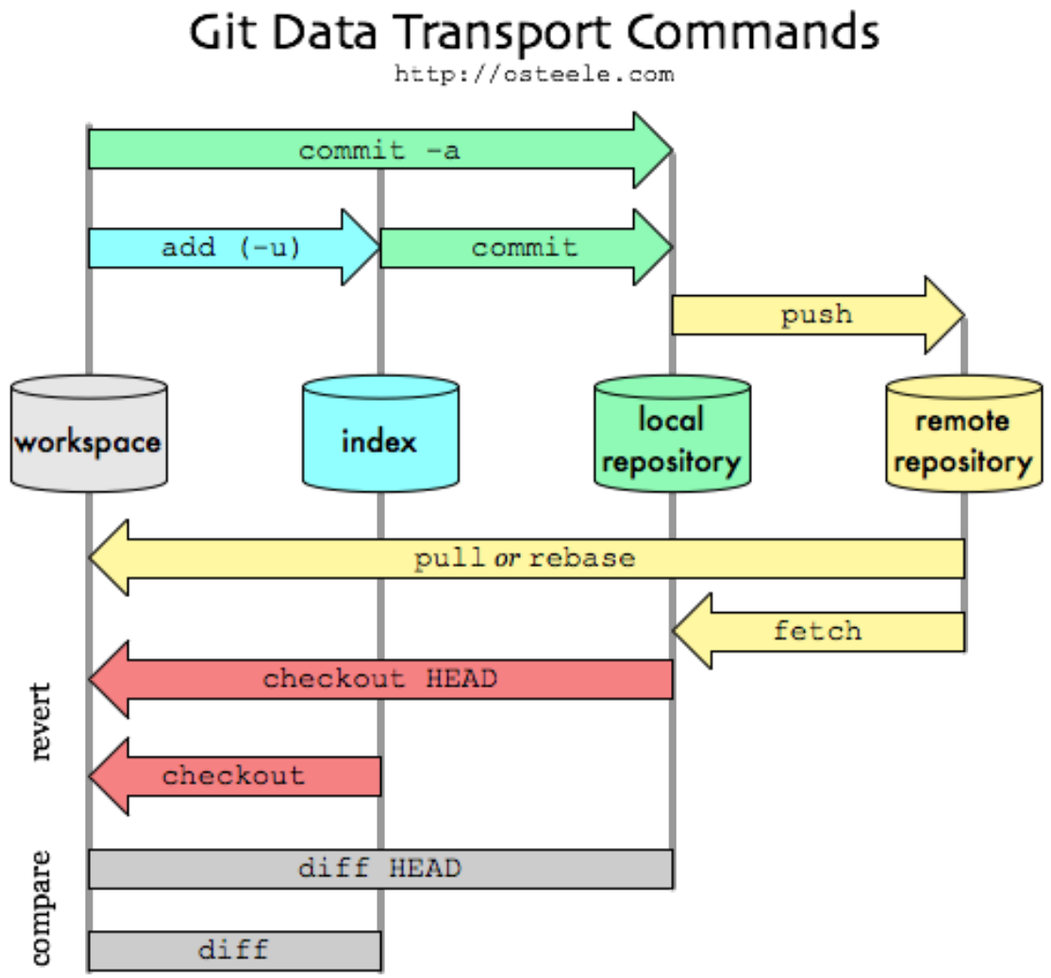


Remote repositories

- Check what remotes you have added:
  - `$ git remote -v`
- Manually add new remote
  - `$ git remote add <optional shortname> <url>`

Remote repositories

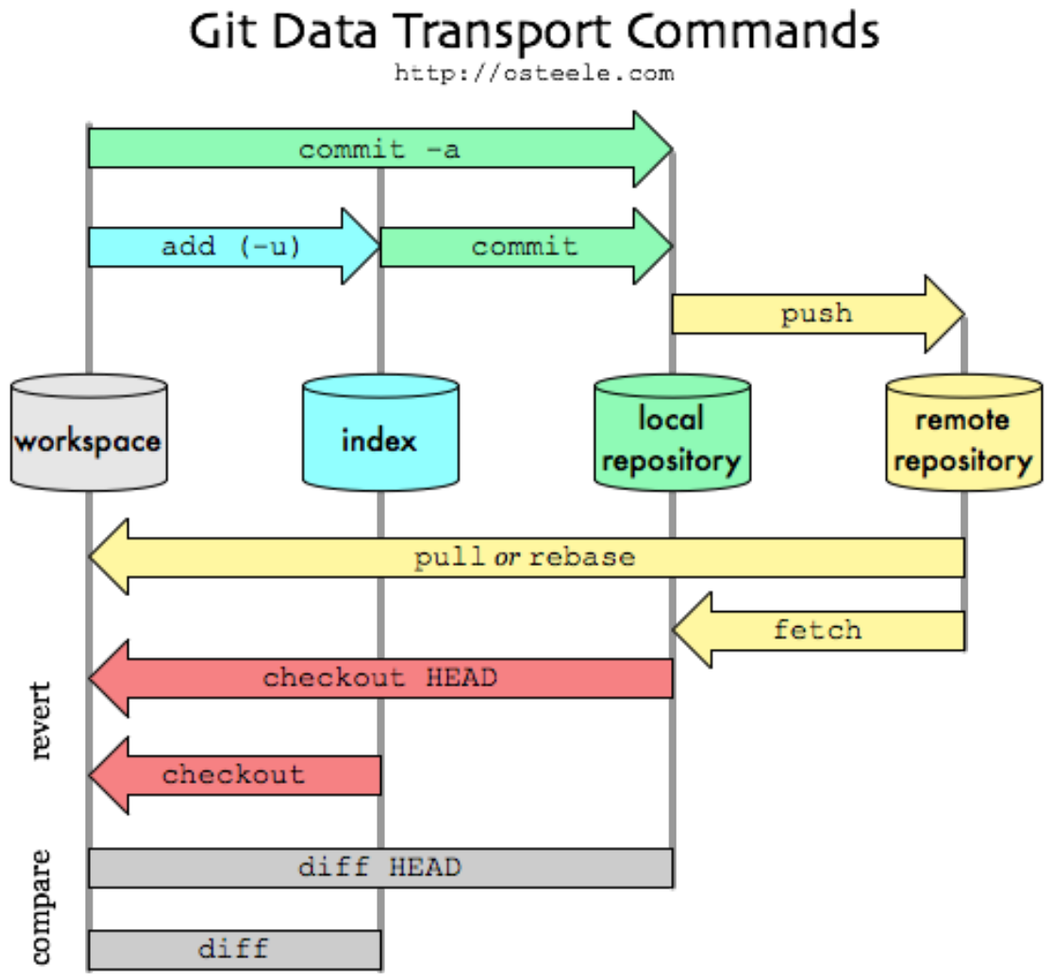
- Get the data from the remote repo, add it to your local repo:
  - `$ git fetch`
- Get the data from the remote repo, and check it out
  - `$ git pull`



Remote repositories

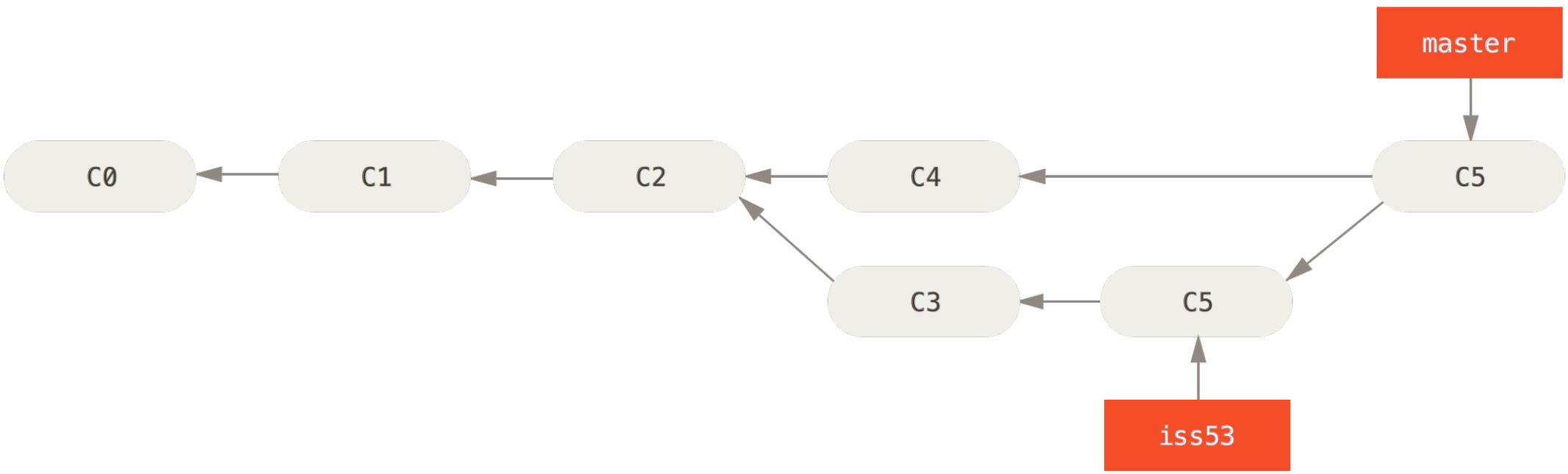
- Send the files from your local repo to the remote repo:

- `$ git push`



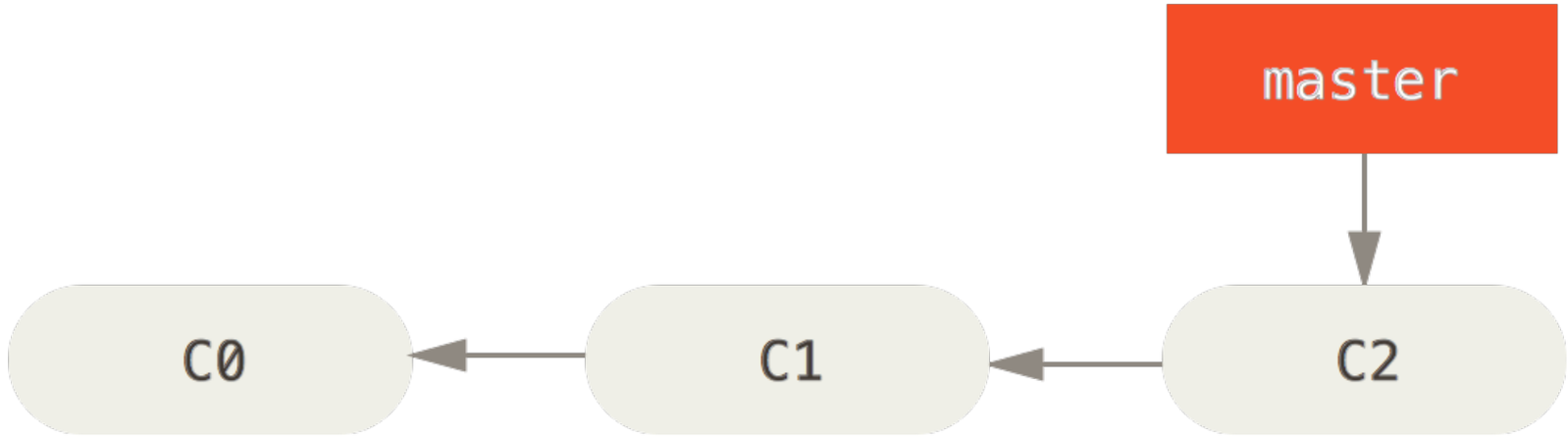
Branches

- Allow you to work on parts of large projects individually



Branches

- So far:

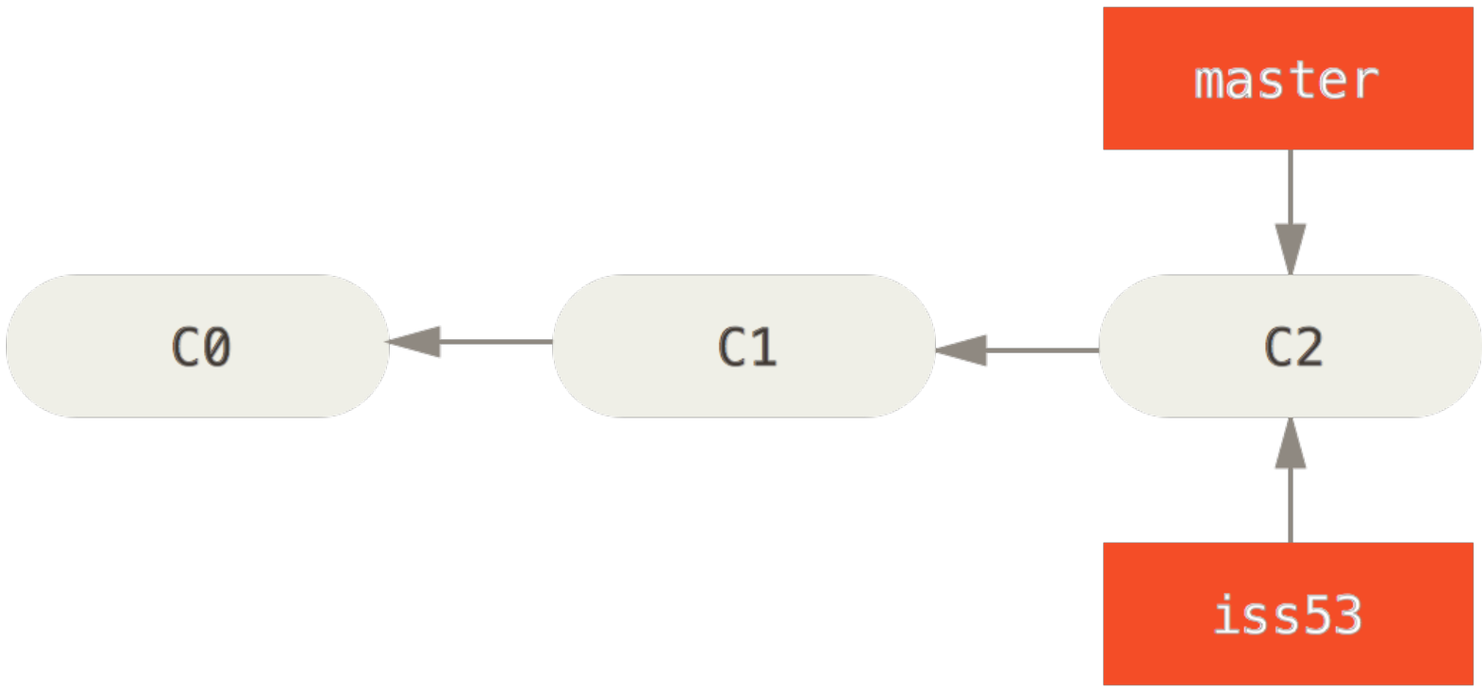


Branches

- Create a new branch:
  - `$ git branch <branch name>`
- Switch to the new branch
  - `$ git checkout <branch name>`
- Create & switch to new branch
  - `$ git checkout -b <branch name>`
- Wait, where am i?
  - `$ git branch`

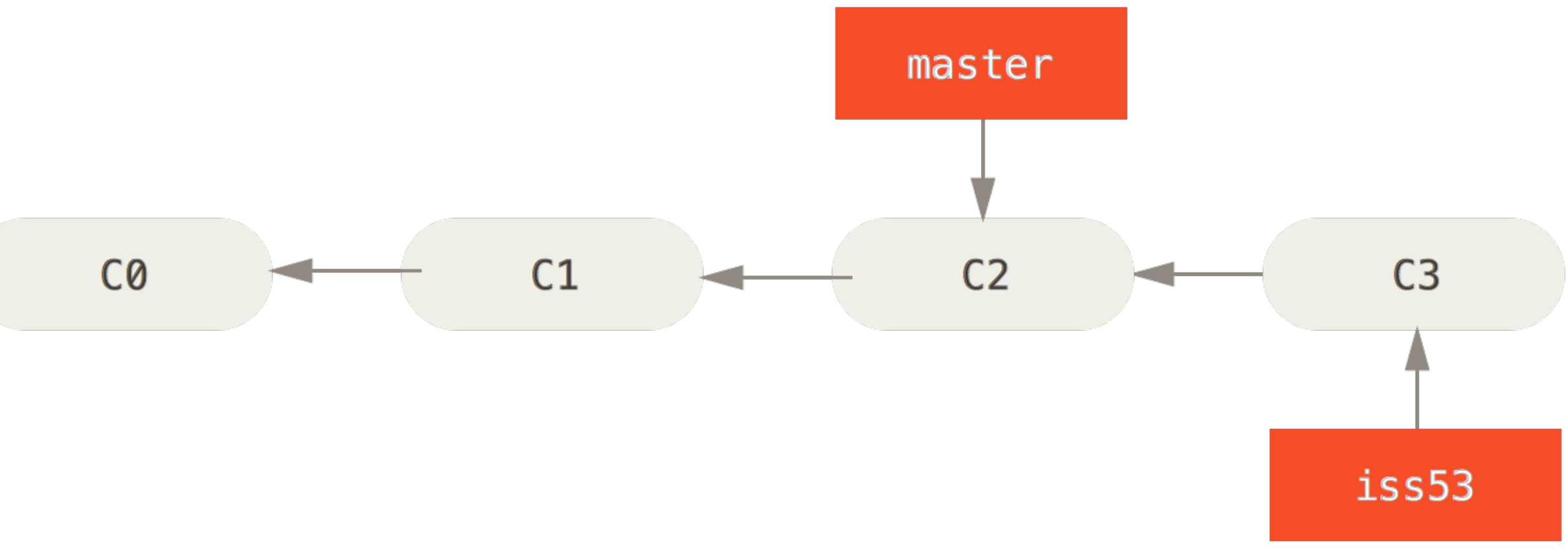
Branches

- Adding a branch does this:



Branches

- Doing a commit on a branch does this:

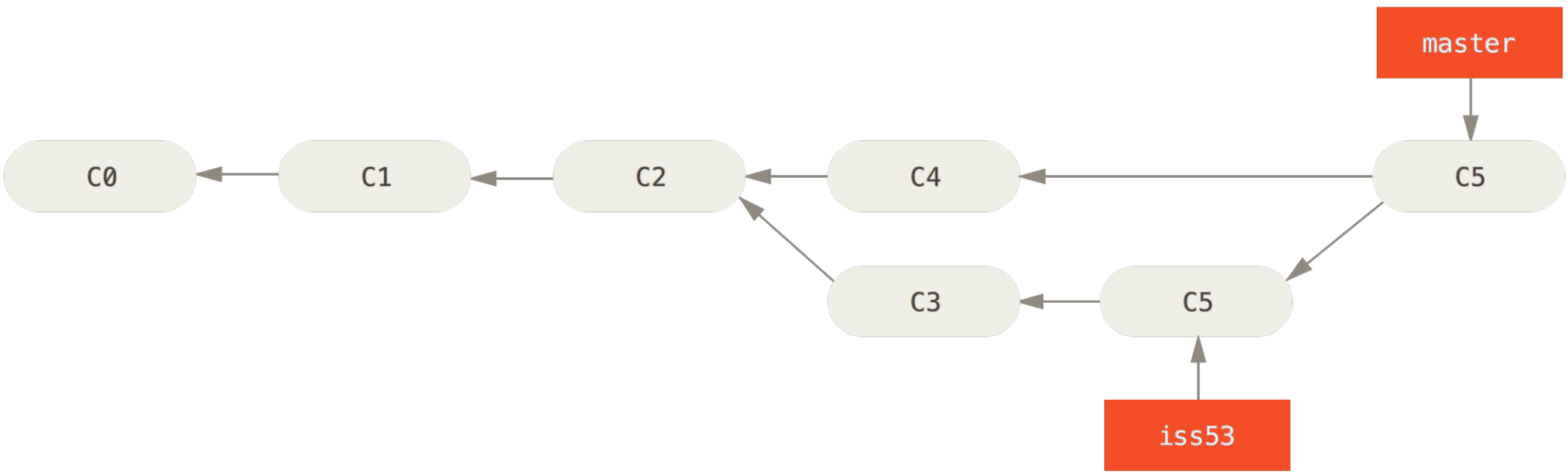


Branches

- Merging branches
  - `$ git checkout master`
  - `$ git merge <branch name>`
- If the merge is successful, and you are down with the branch
  - `$ git branch -d <branch name>`

Branches

- Changes from the branch are merged into the master branch



Sprint 3: Test and review

I love it when a plan comes together.

- Git merge
- Testing
- Deployment!