git

Rohit Varkey Thankachan
github.com/rohitvarkey

Credits

- This slideshow contains material from the following sources.
- Images from the ProGit book, distributed under the Creative Commons (http://creativecommons.org/licenses/by/3.0/). They are marked with *.
- Images from the Atlassian git tutorials, distributed under the Creative Commons Attribution 2.5 Australia License (https://creativecommons.org/licenses/by/2.5/au/). They are marked with **.
- Image from the xkcd webcomic, licensed under Creative Commons Attribution-NonCommercial 2.5 License. (https://creativecommons.org/licenses/by-nc/2.5/). Marked with a ***.

Roadmap

- 10 mins - Why you need git
- 20 mins - git basics, have a local workflow
- 10 mins - git with remotes (git + internet)
- 10 mins - git branching and merging (very basic intro)
- 10 mins - Quick run-through of GitHub.

Linus Torvalds

- Creator of Linux
- Created git in a weekend!

Why you need `git`!

Manage changes

- Stores snapshots of all your files
- Kind of like checkpoints in games
- Fearlessly modify code!

Collaborate with others

- git is fully distributed.
- Separate local copies can be maintained
- Multiple people can work on a project easily
- git helps manage changes made by multiple people

`git` basics

How to use git for just you?

0. Install `git`

- Install git on your computer.
- Installers for Linux, Mac OSX and Windows
- Google “download git” and you should be set

1. Set up `git` for a project

- For git to start tracking your changes, you need to tell it to do so.

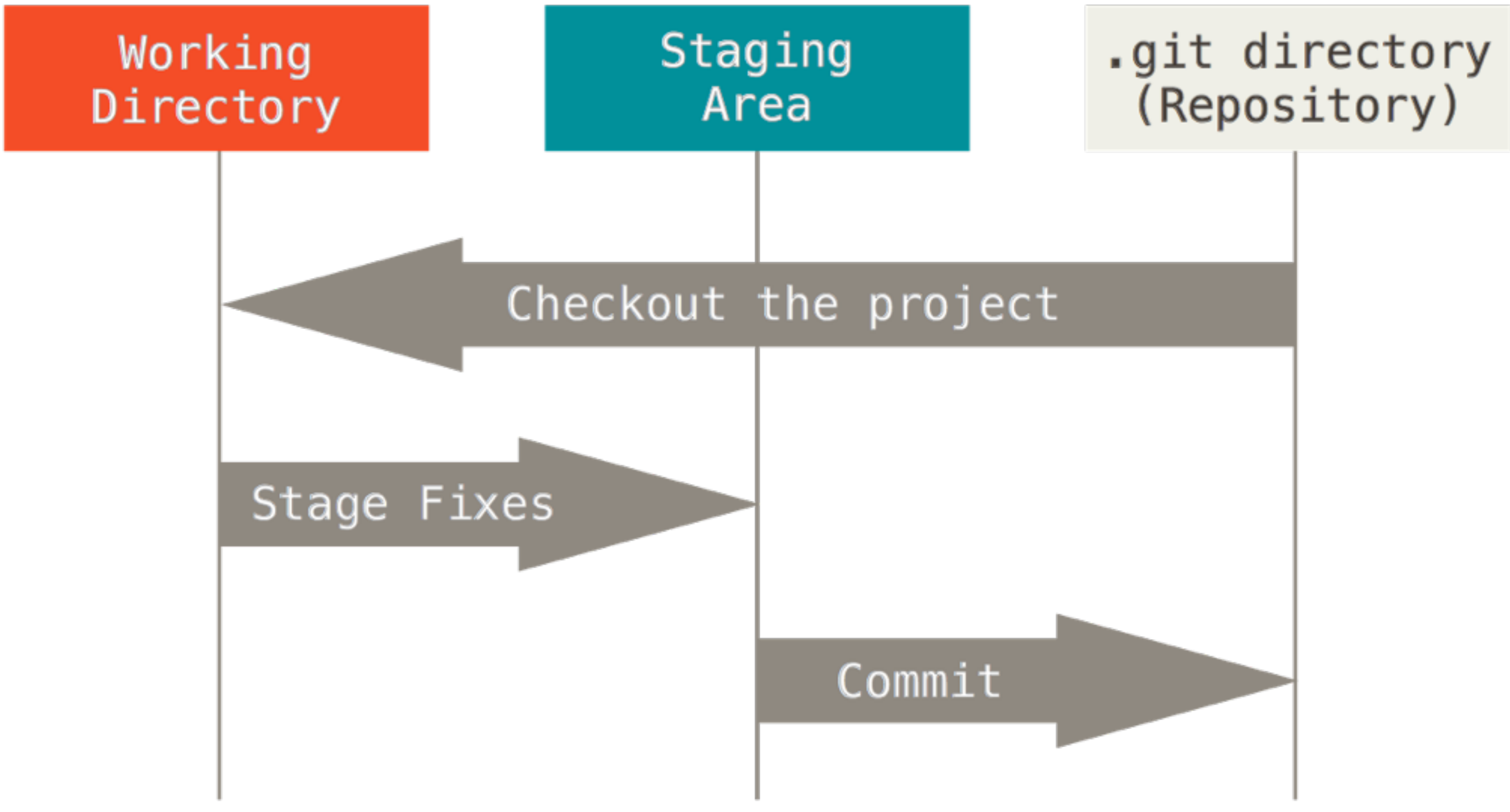
1. Open your Terminal
2. `cd` into the project folder
3. git init

2. Check status

- Let's check if everything worked!
- `git status`
- This command will let you know what state your git repo is in.
- It's a good idea to keep running this command after every other command!

The 3 areas

*


Working Directory
Staging Area
.git directory (Repository)
Checkout the project
Stage Fixes
Commit

3 . Add your changes

- Add changes made to files to the staging area
- Prepare yourself to create a checkpoint
- `git add <filename>`

4. Create a checkpoint

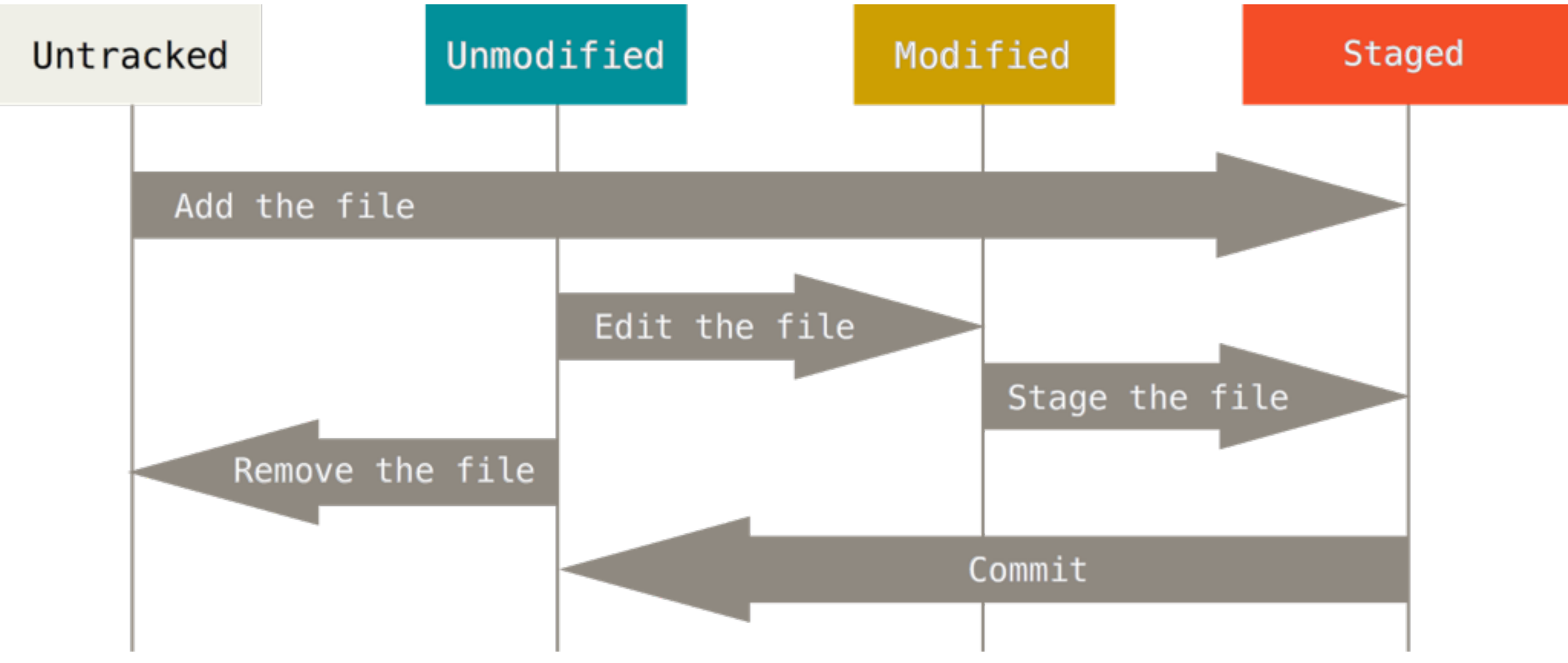
- A checkpoint in `git` is a **commit.**
- When you have all changes you want in the staging area, then commit.
- `git commit -m <description>`
- Commit early, commit often!

5. Seeing the commits

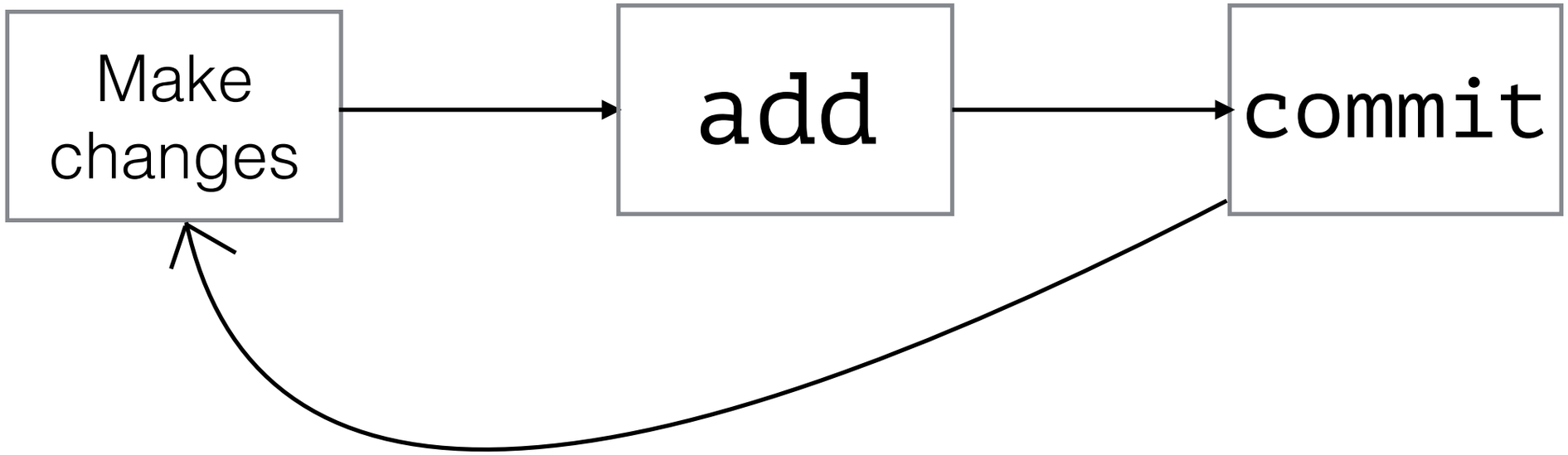
- Take a look through history!
- `git log`
- The hash for every commit is a unique reference to that commit

The `git` File Lifecycle

*



Workflow



- `git diff` - Generates a diff between current working directory and last commmit

- `git reset` - Cleans staging area

- `git reset <hash>` - Resets history to commit represented by hash and cleans staging area

- Several more…

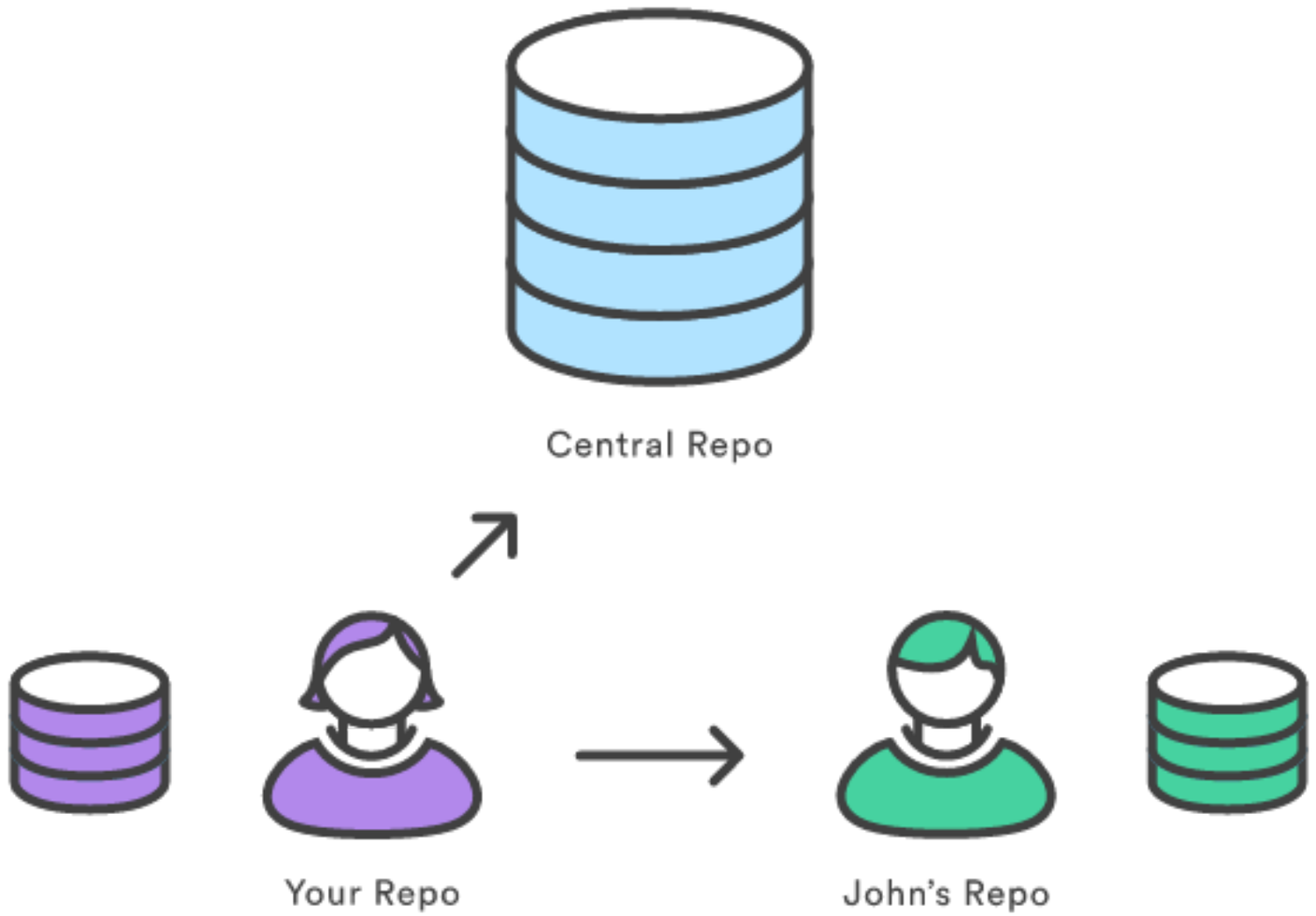
`git` remotes

How to use git to share code?

General situation

- One remote repository (somewhere on the internet)
- Contributors have local copies
- Commits are made by contributors to local git repo
- The remote repo is updated by syncing these commits (**push**)
- Local repos are also synced with the remote repos (**pull**)

**


Central Repo
Your Repo
John's Repo

1. Create a remote repo

- Create a repo on the internet (GitHub is one place!)
- Add a reference to the remote repo to your local repo
- `git remote add <remote_name> <remote_url>`
- Conventionally, the name origin is used for the main remote repo

OR

1. Get a remote repo

- Instead of creating a fresh repo, you might need to work with an existing one.
- `git clone <remote_url>`
- Cloning a repo gets you a local copy of the remote repo
- It’s remotes will already be configured. :)

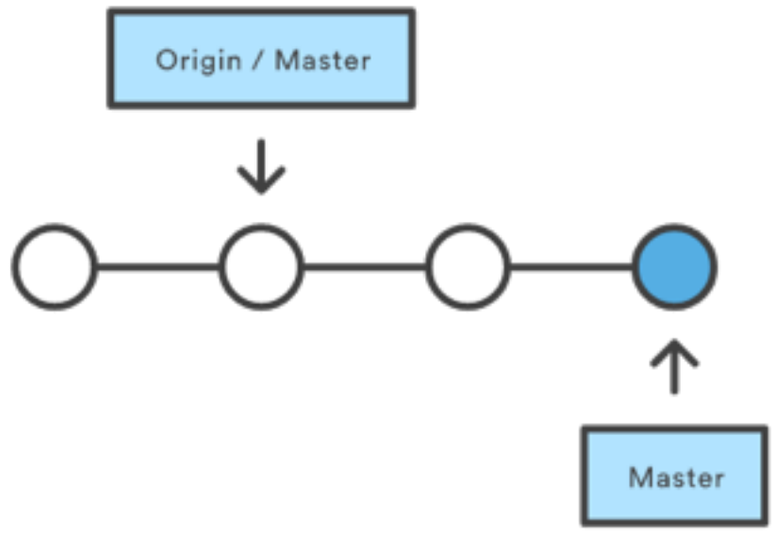
Use the ~~force~~ local workflow and make commits

2. Send your commits🚀

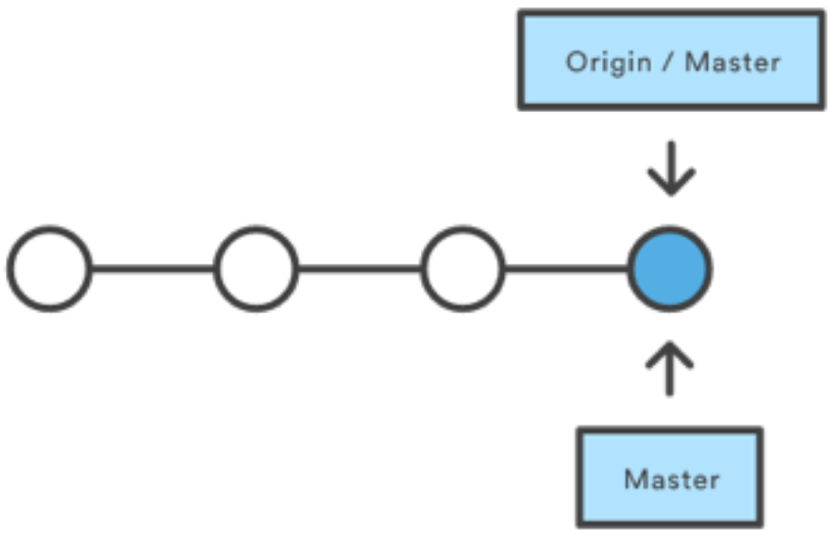
- Currently, your remote repo doesn't have your new local commits!
- You need to send your commits across
- `push` is done to send local changes to remote repo
- `git push <remote_name> <branch_name>`
- Repeat after more commits to send those also!

Before Pushing

**



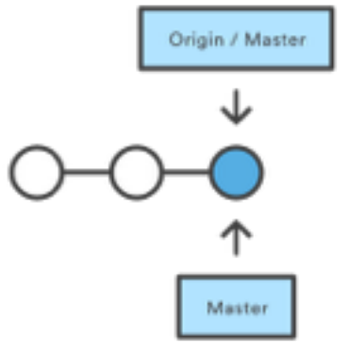
After Pushing

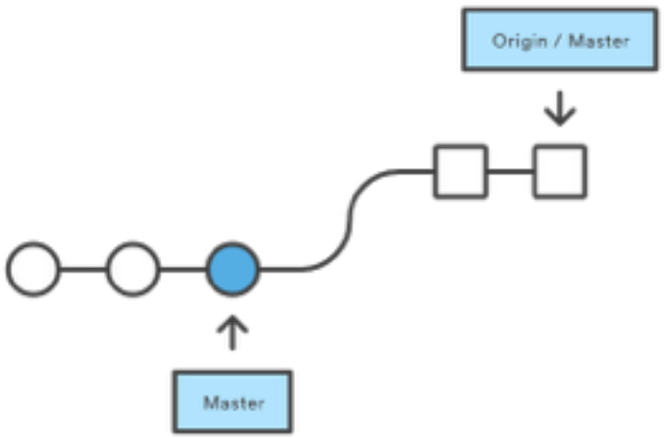


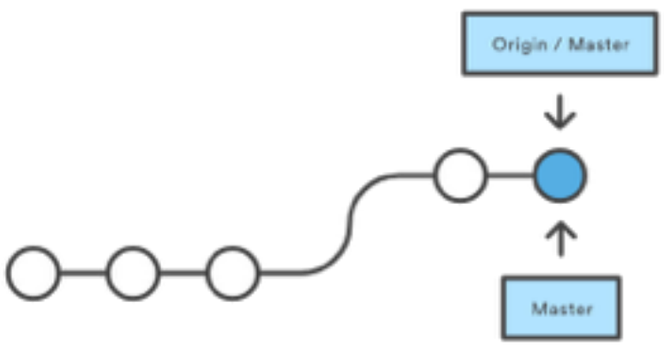
3. Get your remote commits

- git lets you have more than one local repo.
- You need to get your new remote commits onto a unsynced local repo!
- `git pull <remote_name> <branch_name>`
- `pull` is done to get remote repo to local changes

**


Origin / Master
Master

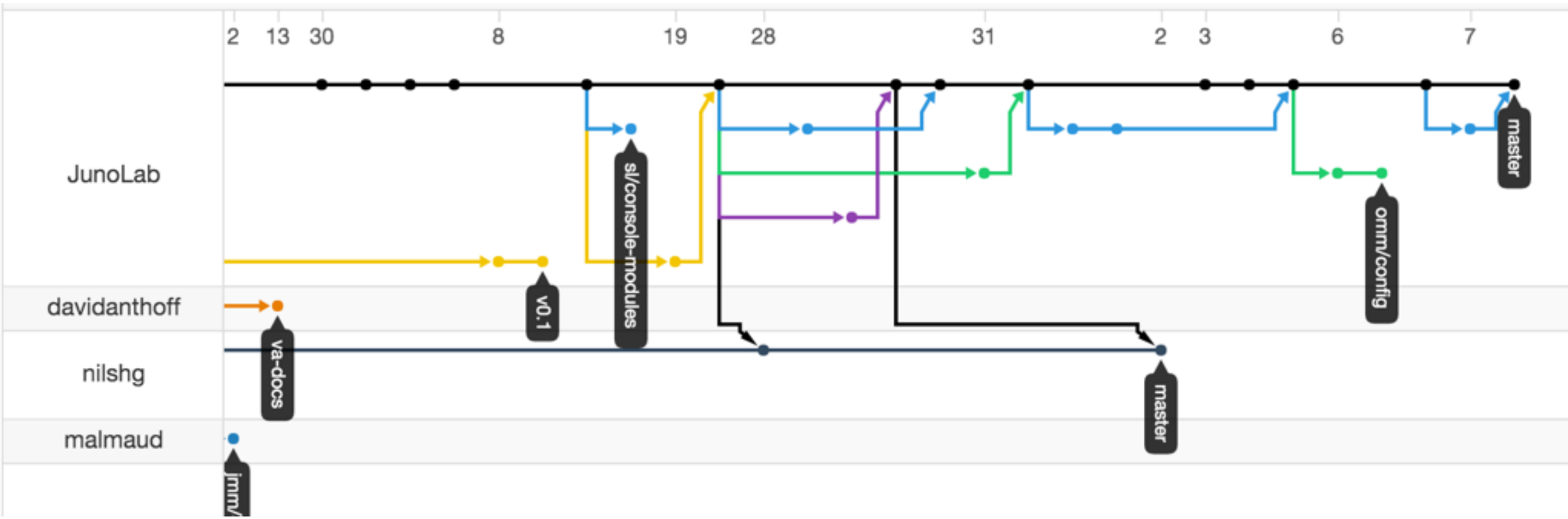

Origin / Master
Master


Origin / Master
Master

`git` branching & merging

How can we use git to develop several features at once?

Real world example

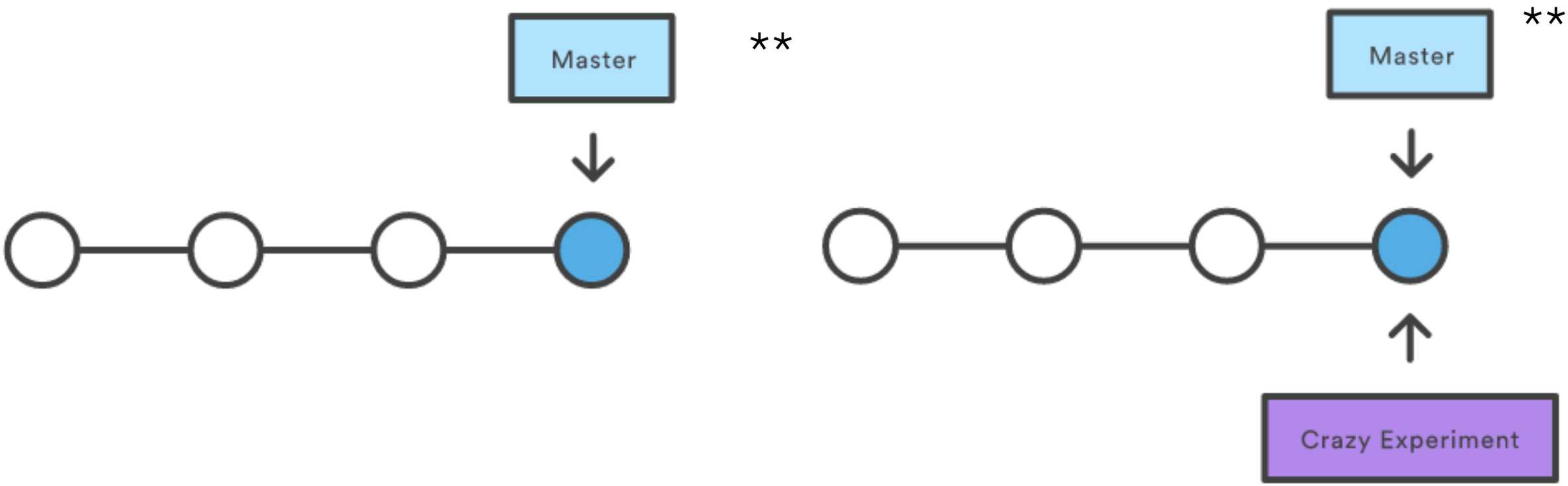


Branches

- Every git repo has a default branch called **master**.
- Branches are a way to have separate commit trees
- Once branched, only commits made on that branch effect it.
- The developer can work on the feature in isolation
- The commits can be merged back to **master** when completed

1. Create a branch

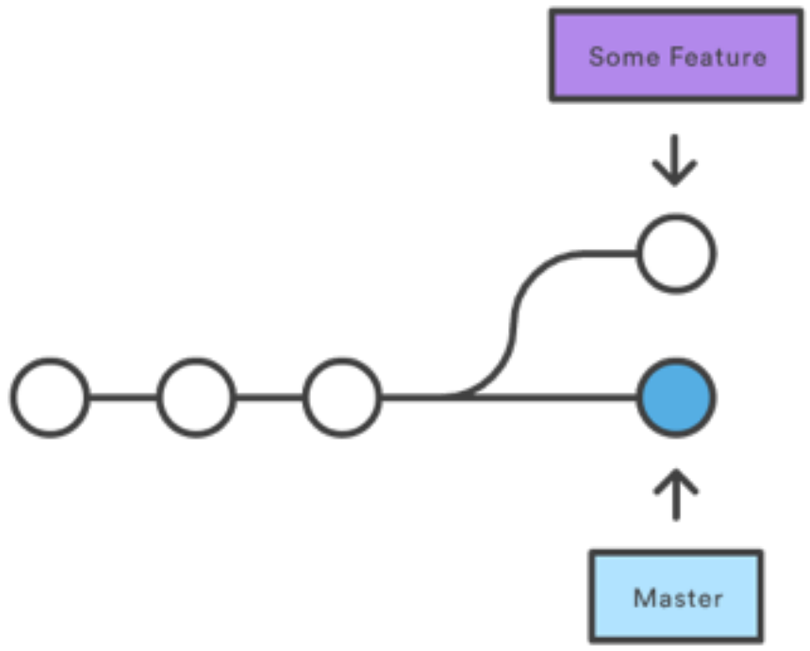
- `git branch <branch_name>`



2. Move to the branch

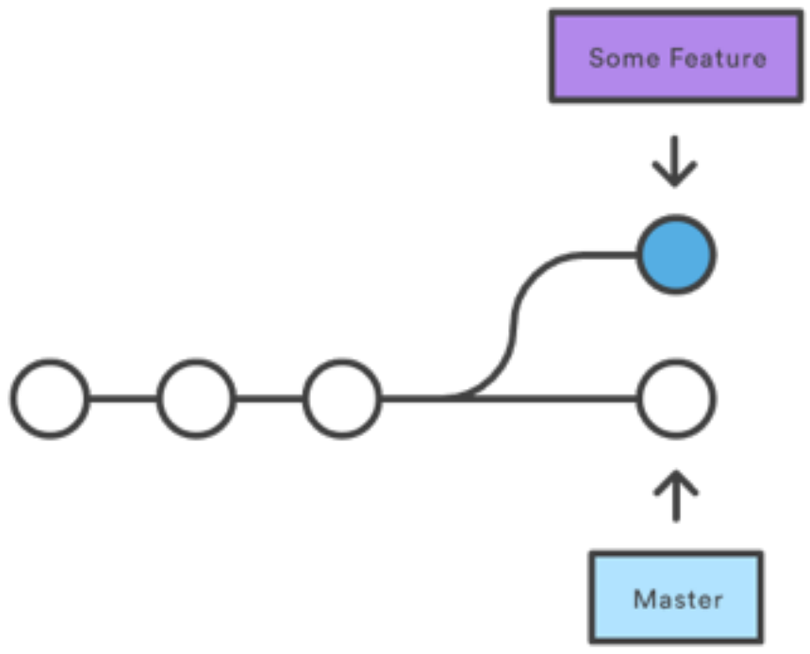
- `git checkout <branch_name>`
- Use this to move between branches.
- Essentially, you will be changing commit histories here.
- A `git log` will show you the differences!

Checking Out Master



**

Checking Out Some Feature

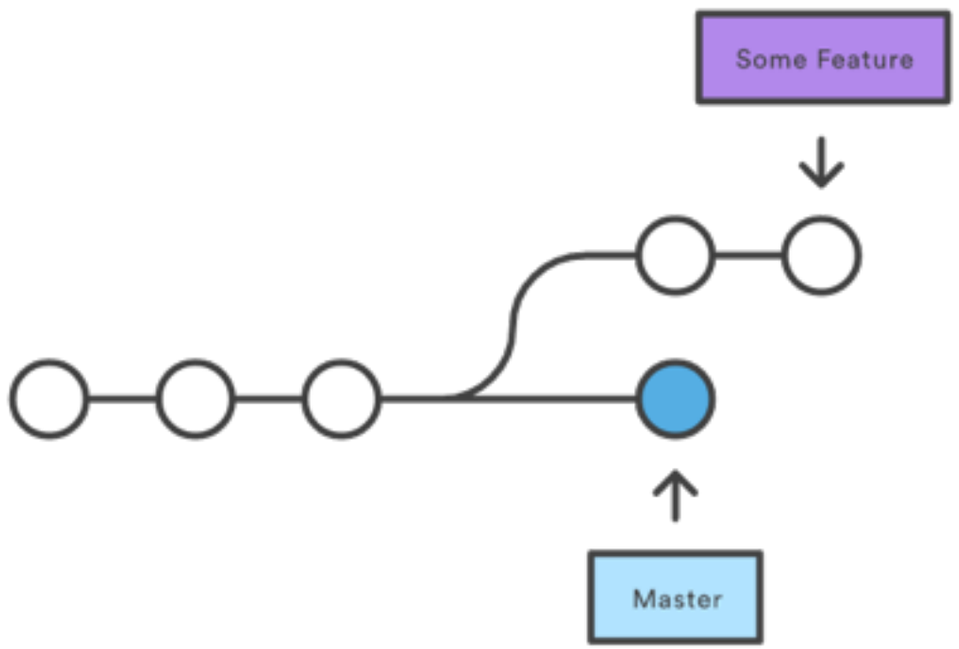


3. Merge

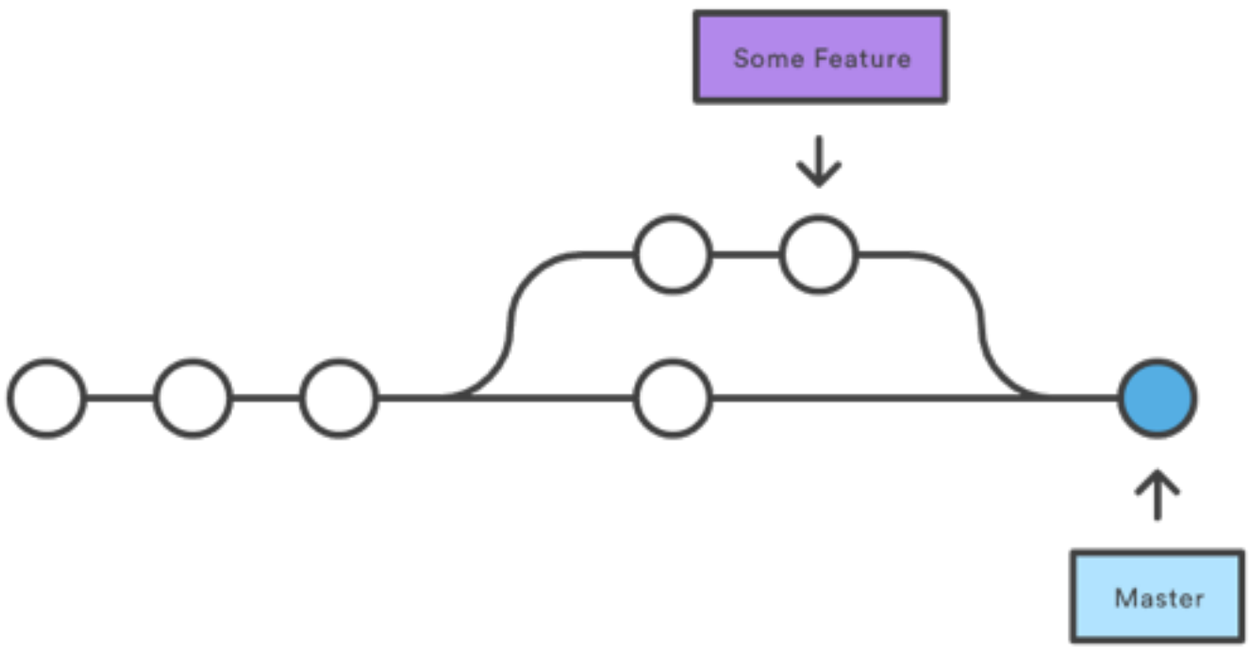
- Use this to merge the 2 trees.
- Get your feature into master!
- Conflicts can arise here.

Before Merging

**



After a 3-way Merge



General branching advice

- Do not commit to master.
- Make branches. Commit. Merge back.
- Makes development clean
- Great way of managing many people working on one project

GitHub!

How to use GitHub?

GitHub ≠ `git`

- GitHub is a company that let's you host your code
- GitHub is currently the most popular code hosting website.
- Lot's of major open source projects are on GitHub now.
- GitHub has good tutorials on most of its features.

Getting an existing project

- GitHub repositories are `git` repos
- `git clone` works!
- `git clone <url>`

…You do your magic, make changes and commits!…

…But! No write permissions to origin! How do I contribute now?…

Forks!

- Forking means you create a copy of the original repository in your profile.
- You have write access to the fork!
- So add your fork as a remote, and push and pull to that remote!

So how do I get the original repo to see my contributions?

Pull Requests!

- Create a pull request in the original repo with a short description of the changes you have made.

- Maintainers will comment on it, make you refine it till they are happy with it and then merge it!

Issues

- GitHub has an issues facility for their repositories.
- As a user you can file your bug reports/worries/ ideas about the repository in the issues.
- As a developer, you can look through the issues and try and fix some of them!
- Look for labels to figure out beginner level ones or ones in your area of interest.

Markdown

- Writing is a very important part of software development.

- GitHub let's you use it's form of Markdown to have really nice formatting and several other tricks to make the experience better.

- Do go through their cheatsheet and learn to write well!

Questions?


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

***