

# Collaborating with Git

Because coding is better with  
friends



# ALTERNATIVES



# Alternatives

1. Email a .zip of your project to yourself and your friends every once in a while
2. (Hackathon only) Everyone works on their own files and then you put everything in the same folder right before you demo
3. Dropbox? I guess??????

# GIT

is better



# What is Git?

- Git is version control software.
- Version control is a way to keep track of a folder over time. User-defined savepoints, called commits, are used to keep track of changes.

# What is Git?

- Important to have a workflow, e.g.:
  - (do work)
  - `git add -A`
  - `git commit -m "Fixed the button again lol"`
  - `git push origin master`
  - (repeat)
- But this workflow must change when you work with others.

# WORK TOGETHER



# Working together


- First of all, there are a lot of different ways to collaborate using Git. A lot!
- To me, there are three main workflows to be aware of.













# Working together

1. Working on a very small (<4 person) project where work doesn't need to be reviewed
  - everyone shares a copy of the centralized repo and does their work on the master branch
  - each contributor is responsible for the quality of their own work





# Small projects #1









 evantarrh / bruhzzfeed


 Unwatch 2  Star 1  Fork 1







 Code  Issues 0  Pull requests 0  Wiki  Pulse  Graphs  Settings

Clarifai API + popular Imgur images + jokes <http://bruhhh.co> — Edit

 45 commits  1 branch  0 releases  2 contributors

Branch: master  New pull request  New file  Upload files  Find file  HTTPS <https://github.com/evan>    Download ZIP

 evantarrh adding google analytics Latest commit 46fe0dd on Oct 18, 2015

|  |   |              |
|--|---|--------------|
|  backend    | general cleanup, refactoring db to work with heroku | 7 months ago |
|  static     | big refactor of app.py                              | 6 months ago |
|  templates  | adding google analytics                             | 6 months ago |
|  .gitignore | adding google analytics                             | 6 months ago |
|  Procfile   | Update Procfile                                     | 7 months ago |
|  README.md  | Update README.md                                    | 7 months ago |



# Working together

## 2. Working on a small or private project where work needs to be reviewed

- everyone shares a copy of the centralized repo
- each contributor has multiple branches on their local machine, keeping track of both the master branch and their own branches at the same time
- if a contributor wants to incorporate a change, they open a pull request from one of their branches to the master branch




## 2. Working on a small or private project where work needs to be reviewed

- everyone shares a copy of the centralized repo
- each contributor has multiple branches on their local machine, keeping track of both the master branch and their own branches at the same time
- if a contributor wants to incorporate a change, they open a pull request from one of their branches to the master branch
- **other contributors** then review the pull request before merging the work into the master branch



# Small projects #2

 **mayankmahajan24 / QL**

Unwatch 5

Unstar 4

Fork 1

<> Code

Issues 3

Pull requests 0

Wiki

Pulse

Graphs

QL Language and Compiler, for Programming Languages and Translators Course Fall 2015

378 commits

1 branch

0 releases

5 contributors

Branch: master

New pull request

New file


Upload files

Find file

HTTPS

https://github.com/maya

Download ZIP

 **evantarrh** Merge pull request #54 from mayankmahajan24/anshul-lastminute

Latest commit c96ad1f on Dec 26, 2015

|            |  |              |
|------------|--|--------------|
| build      | where in progress                            | 4 months ago |
| compiler   | last minute                                  | 4 months ago |
| docs       | last minute                                  | 4 months ago |
| tests      | formatting and compressing integration tests | 4 months ago |
| .gitignore | adding fitbit integration test. remarkable   | 4 months ago |
| Makefile   | More headers                                 | 4 months ago |




# Working together


## 3. Working on a large open-source project


- one centralized repo
- each contributor has their own copy of the repo, called a fork
- if a contributor wants to publish a change, they open a pull request from their fork
- the owner(s) then reviews the pull request before merging the work into the central repo





# Open source projects


 **evantarrh / rails**  
forked from rails/rails


 Unwatch ▾ 1


 Star 0


 Fork 12,424

 Code


 Pull requests 0


 Pulse


 Graphs


 Settings

Ruby on Rails <http://rubyonrails.org> — Edit

 57,490 commits

 48 branches

 290 releases



 3,044 contributors



Branch: master ▾ **New pull request**



New file








Upload files

Find file

HTTPS ▾ <https://github.com/evan>   **Download ZIP**

This branch is even with rails:master.  Pull request  Compare

 **jeremy** Merge pull request #24511 from lihanli/activemodel-dirty-attribute-ch...  Latest commit 20ffb63 30 minutes ago

|  |  |                |
|--|--|----------------|
|  .github      | fix typo in pull_request_template [ci skip]                              | 2 months ago   |
|  actioncable  | Cable typo: isSupportedProtocol -> isProtocolSupported                   | 6 days ago     |
|  actionmailer | Merge pull request #24497 from vipulnswad/am-changelog-pass              | 8 hours ago    |
|  actionpack   | Merge pull request #24504 from nickmalcolm/master                        | an hour ago    |
|  actionview   | Improved ActionView flows.rb documentation [ci skip]                     | 3 hours ago    |
|  activejob    | Merge pull request #24165 from y-yagi/generate_application_job_when_n... | 2 days ago     |
|  activemodel  | Merge pull request #24511 from lihanli/activemodel-dirty-attribute-ch... | 30 minutes ago |



# Working together

For this workshop, we'll be focusing on the first two scenarios. These workflows are more relevant to both classwork and industry work.

If you have any questions about open source development and working with forks, I'm happy to chat more afterwards!



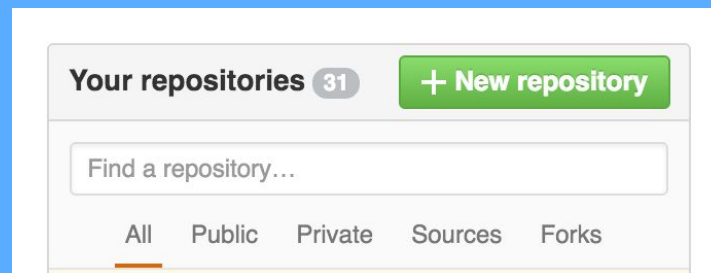


# GETTING STARTED



# Getting started

For either workflow, getting started will be pretty much the same process. Whoever wants to own the repository will create one using GitHub:

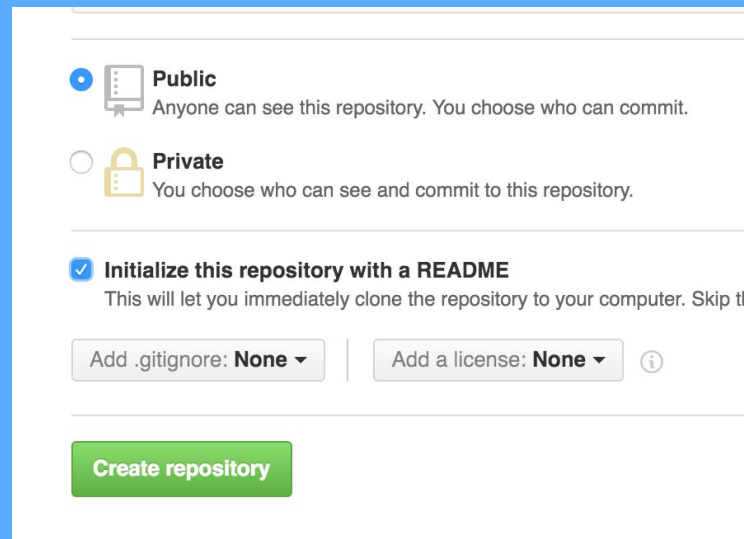


This can be done from the command line as well, but let's stick with GitHub for now.



# Getting started

When you create the repo, check the box that says “Initialize this repository with a README”.



The screenshot shows the GitHub repository creation interface. It features two radio buttons for repository visibility: 'Public' (selected) and 'Private'. Below these, there is a checkbox labeled 'Initialize this repository with a README' which is checked. Underneath the checkbox, a note states: 'This will let you immediately clone the repository to your computer. Skip this step if you want to create the repository without cloning it to your computer.' At the bottom of the form, there are two dropdown menus: 'Add .gitignore: None' and 'Add a license: None', followed by an information icon. A green 'Create repository' button is positioned at the bottom of the form.

☒ **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 want to create the repository without cloning it to your computer.

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

**Create repository**

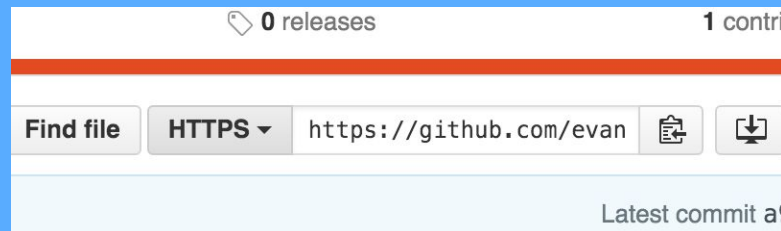


# Getting started

Once the repository has been created, everyone (including the owner) should clone the repo onto their own machine with the following command:

```
git clone https://github.com/evantarrh/cool-repo-name.git
```

(the URL can be copied and pasted from this box on GitHub)



# WORKFLOW #1



# Workflow #1

Good for:

- Hackathons
- Small/quick projects
- Projects where you know your teammates really well



# Workflow #1

- Not too different from working on your own.  
The core workflow is the same: after doing work,
  - `git add -A`
  - `git commit -m "Fixed the button again lol"`
  - `git push origin master`
- But what happens when you and your teammates are making changes at the same time?

## Local Repository

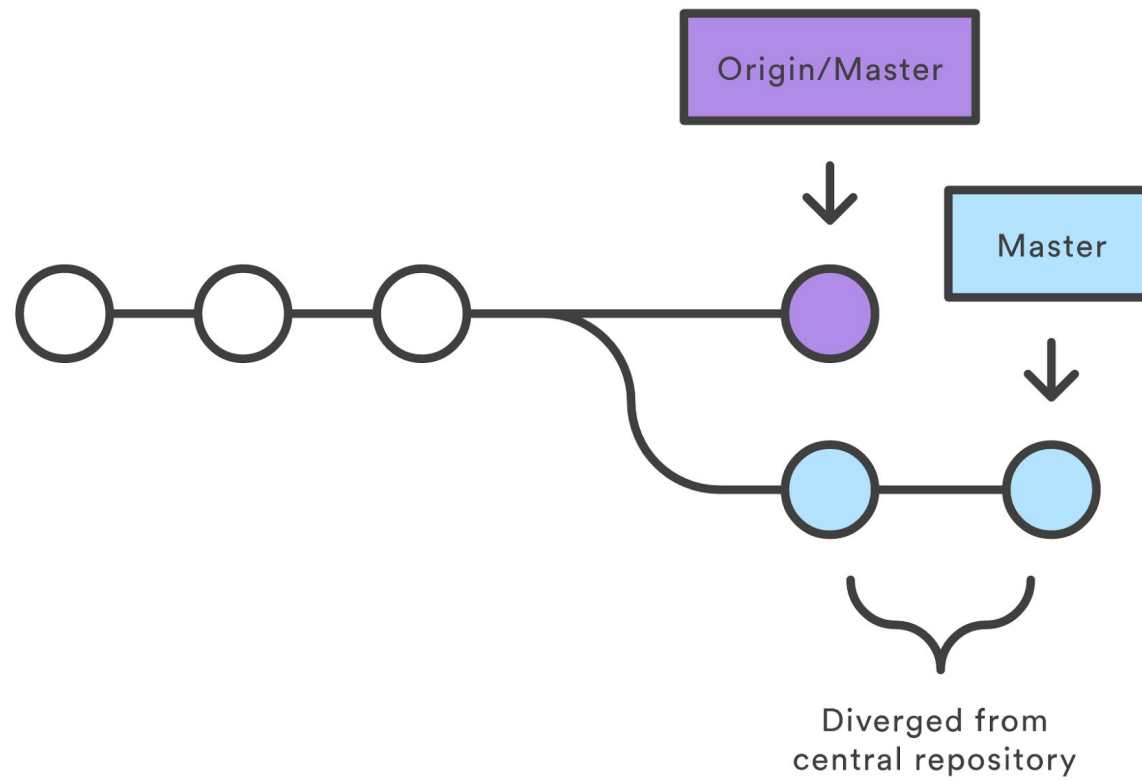


Image: <https://www.atlassian.com/git/tutorials/comparing-workflows/centralized-workflow>





# Workflow #1

If you try to push conflicting work to GitHub, it will give you an error.

If that happens, you'll need to run the following command:

- `git pull --rebase origin master`

This attempts to integrate the new commits from GitHub with the new commits on your computer.



# Workflow #1

If you're working on similar features, you may experience a merge conflict. Git will tell you what files are affected, and you'll be able to see something strange like this:

```
1 <div class="content landing">
2   <div class="hero-wrapper">
3 +<==== HEAD
4     <div class="hero-text">Turn your audience into a task force.</div>
5 +=====
6 +   <div class="hero-text">Transform your network into a task-force
    designed to solve local public issues.</div>
7 +>>>>>> 45dd2b10f80ee31b308524d85c0b38351fc17ddc
8
9     <% if !user_signed_in? %>
10      <%= link_to "Get Started", new_user_registration_path, class: "hero-
```



# Workflow #1

The area between “<<<<HEAD” and “====” will always be your work. Between the “====” and some commit hash (“45dd2b01f...”) is the work someone else has done.

```
1 <div class="content landing">
2 <div class="hero-wrapper">
3 +<<<<<<<< HEAD
4 <div class="hero-text">Turn your audience into a task force.</div>
5 +=====
6 + <div class="hero-text">Transform your network into a task-force
  designed to solve local public issues.</div>
7 +>>>>>>> 45dd2b10f80ee31b308524d85c0b38351fc17ddc
8
9 <% if !user_signed_in? %>
10 <%= link_to "Get Started", new_user_registration_path, class: "hero-
```



# Workflow #1

It's your job to patch things up and decide which change should stay—ideally, you'll communicate with whoever made the conflicting change and make sure you're on the same page!

Once you've resolved the conflict, you must `git add` the files you've updated, and then run the following command:

```
git rebase --continue
```



# Workflow #1

Whether or not you have to deal with a merge conflict after running

```
git pull --rebase origin master,
```

you'll still need to run

```
git push origin master
```

afterwards to publish your changes.



# Workflow #1: a summary

- (do work)
- `git add -A`
- `git commit -m "Changed something lol icr"`
- `git push origin master`
  - if necessary:
    - `git pull --rebase origin master`
    - resolve any merge conflicts & run:
    - `git rebase --continue`
    - `git push origin master`



# WORKFLOW #2



# Workflow #2

Good for:

- Class projects
- Real work
- Projects where your team is distributed





# Workflow #2

- A little more complicated.
- Requires familiarity with Git branches.



# Workflow #2

Before doing any work, you should identify what task you're actually trying to accomplish, and create a new branch specifically for the task. For example,

```
git checkout -b fix-mobile-nav
```

will create a new branch called “fix-mobile-nav”.

- Any work you commit will be added to the fix-mobile-nav branch, but not to master.
- You can switch between branches with `git checkout`. To switch to master, run `git checkout master`.



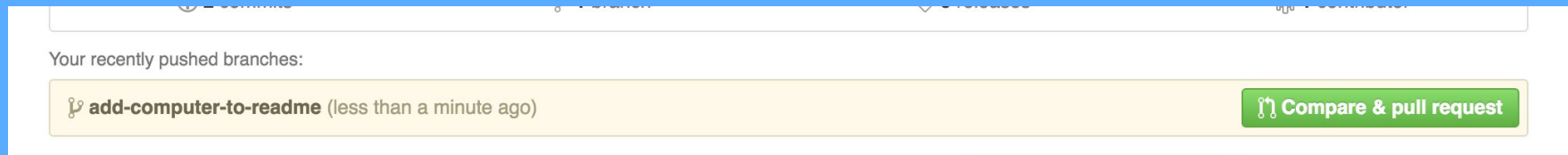
# Workflow #2

While on your own branch, you commit work as normal. Once you think your work is ready to be merged into master, you'll do 2 things:

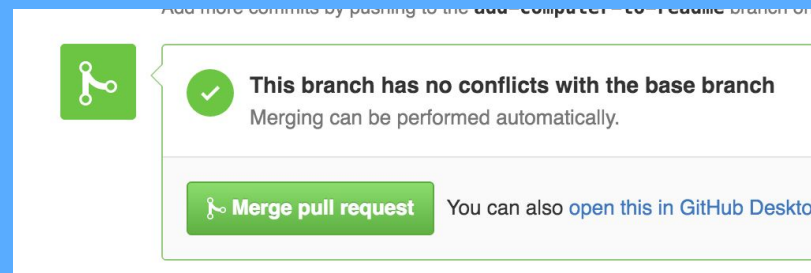
1. Push your branch to GitHub so that your teammates can see the work you've done: `git push origin fix-mobile-nav`
2. Open a pull request on GitHub. Your teammates will review your work, and you'll merge the pull request when at least one of them have given it a thumbs-up.



# Workflow #2: Pull requests



- Push dat green button
- Engage in a healthy comment dialogue with your teammates about your pull request
- Once they sign off on it, push dat other green button



# Workflow #2: a summary

- `git checkout master`
- `git pull --rebase origin master`
- `git checkout -b name-of-feature`
- Until your feature is done:
  - (do work)
  - `git add -A`
  - `git commit -m "Very good programming"`
- `git push origin name-of-feature`
- Make a pull request & merge when ready

