GitHub & Git Branching

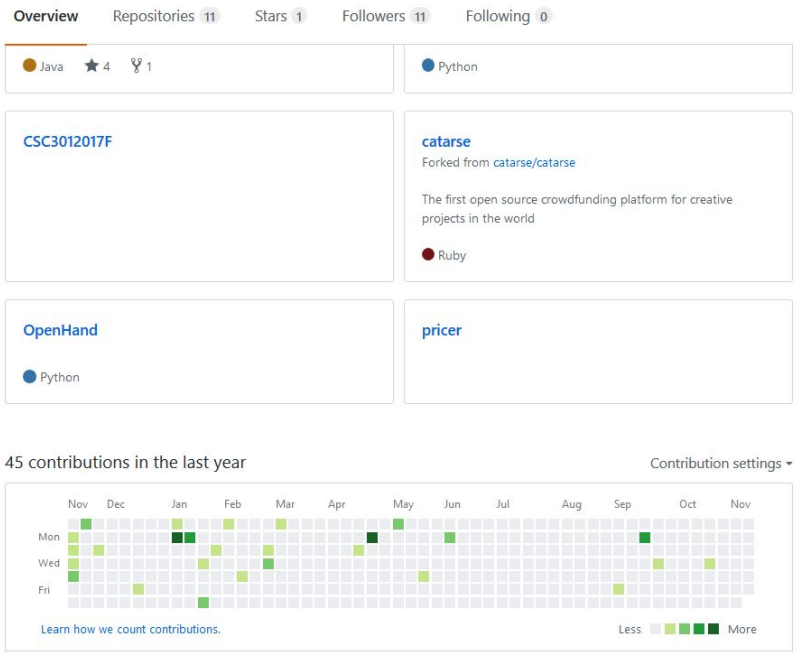
GitHub Overview

- Web-based Git Hosting Service
- Global Standard (BitBucket.org is another one)
- Very helpful to show your work to the world (e.g., future employers)
  - Think of it as your “online portfolio”
  - You can create repositories for the projects you have done
- Free to have a public account and share your code publicly
- You can put your GitHub URL on your resume
  - This is even expected

Git - Review

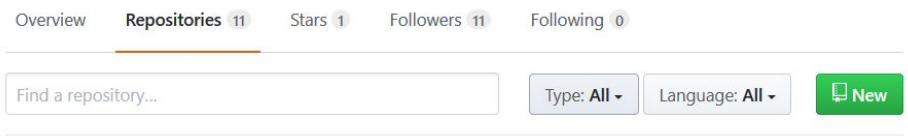
- Basic commands:
  - `git add` - Add a file to the staging area
  - `git commit` - Commit all added changes
- These commands run locally
  - You can commit (i.e. save) changes, even if you are not connected to the Internet.
- Getting info:
  - `git status`
  - `git log`

GitHub Profile Overview

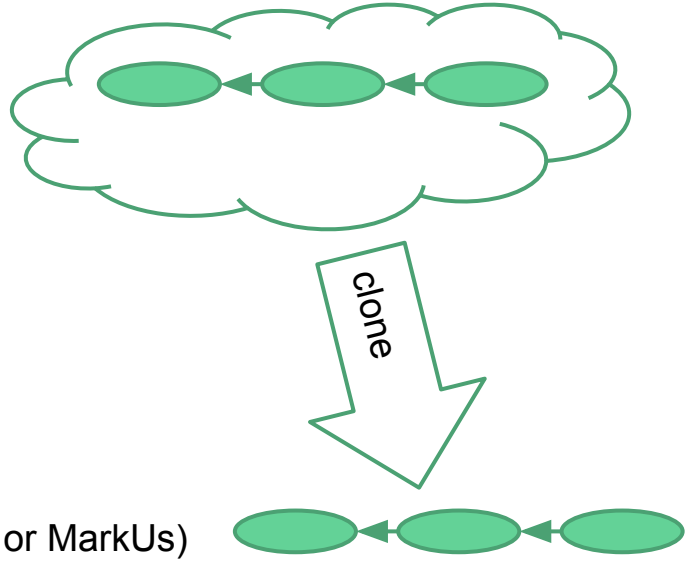


How to create a new repo

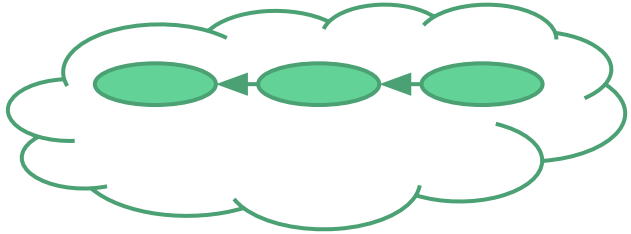
- Go to your profile
- Click on “Repositories” and “New”
- Choose a [descriptive] name
  - Make it public
  - For simplicity, do not initialize the repo
- Create the repo
- You will get a url like the one below:
  - https://github.com/jorjani/SlidingTilesStarter
- You can do “git clone [URL]” and voila! The rest is exactly like MarkUs
- **After the term is over**, you can copy in your A2 or project code



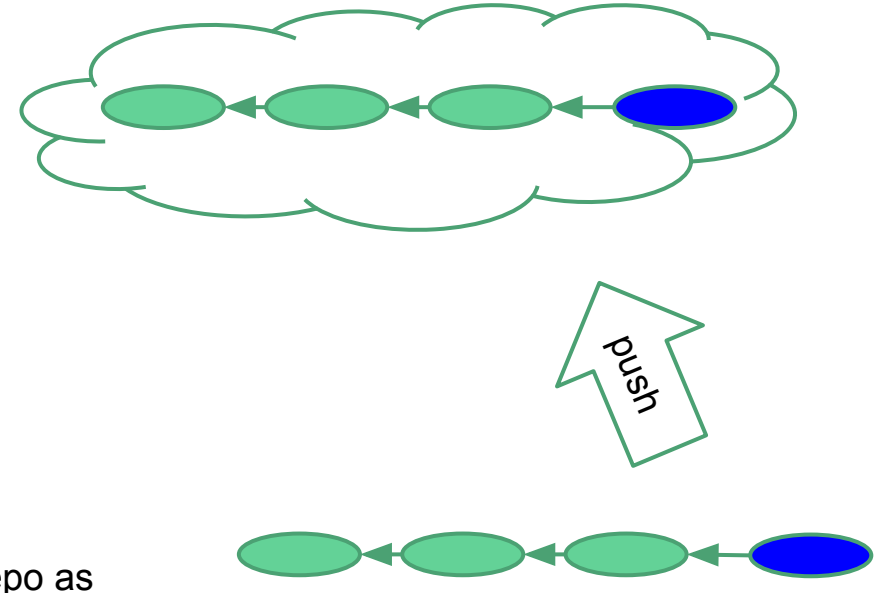
Now let’s review our Git process and learn something new



Local clone of our repo (GitHub or MarkUs)



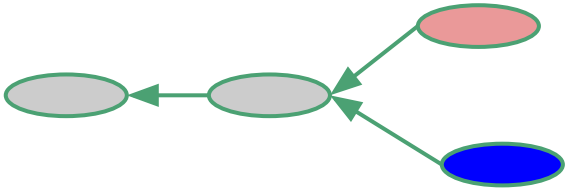
Great! Local commits.
Making progress!



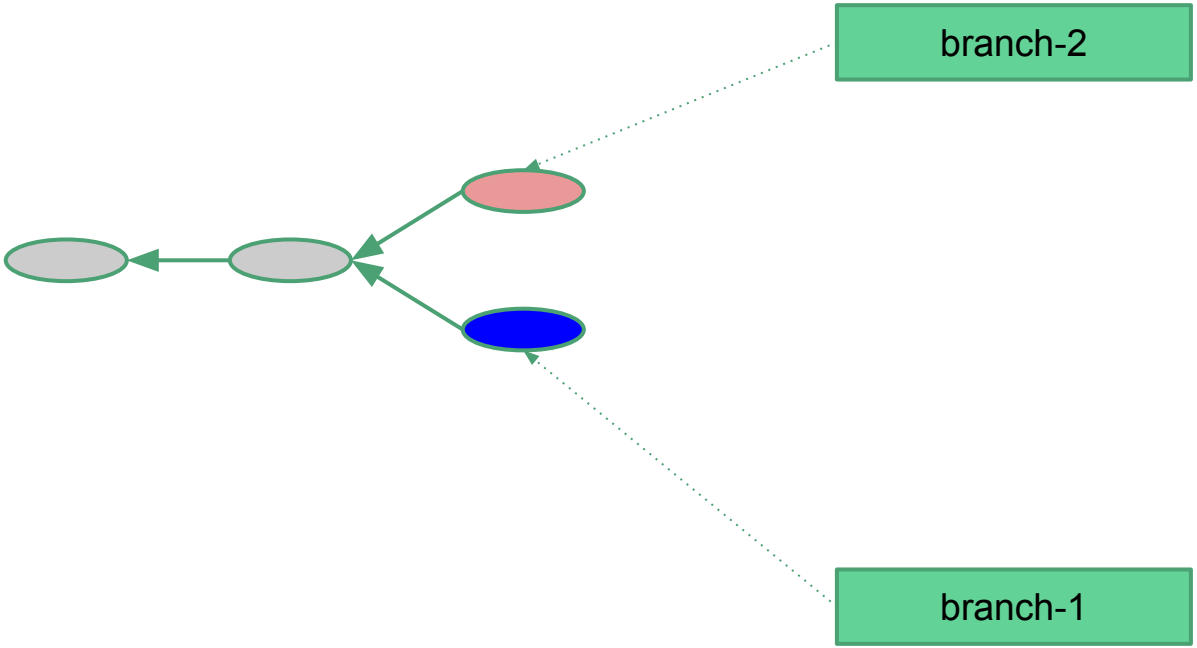
Push WIP (Work-In-Progress) to the main repo as you commit.

Branching

- A repo is a graph of commits.
- Branches are just a (named) reference to a commit in the graph.



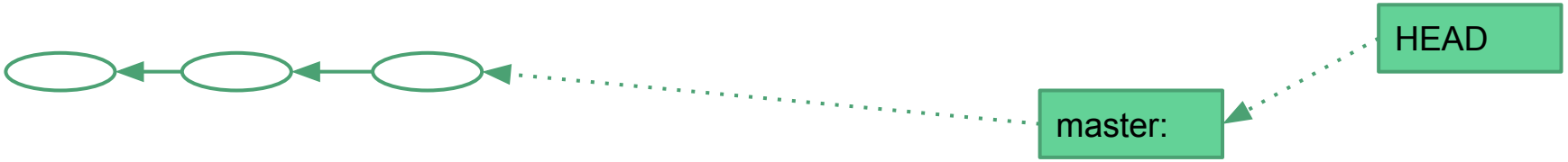
Branching



Branching is very handy for WIP

- Branching in Git is very often used to enable a fancy undo
  - Checkout a (new) branch
  - Fool around a bit
  - Merge it or discard
- Jargon warning:
  - Git “checkout” switches branches.

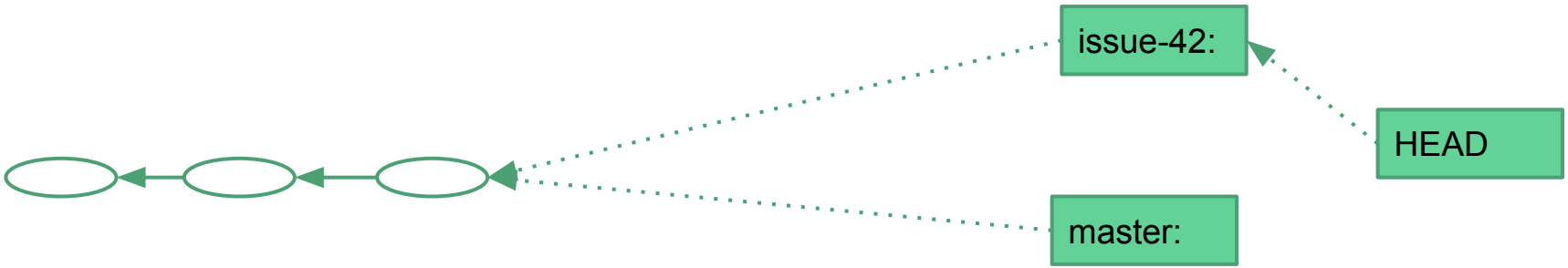
Branching



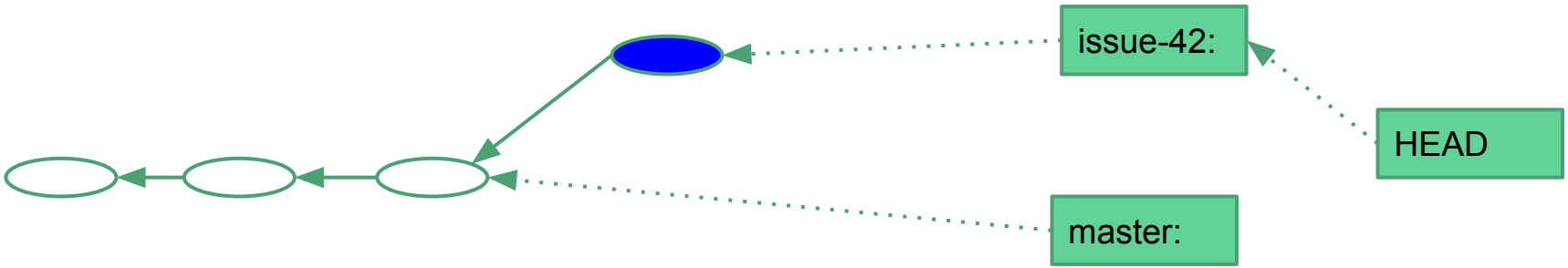
Branching - Creating A Branch

```
git checkout -b issue-42
```

- New branch points to the same commit
- `HEAD is` now the tip of the current branch we're working on



Branching - Commit Changes In A Branch



Merging Branches

- When merging (one branch into another), there are (usually) 3 cases:
  - Fast forward
    - Changes only happened in one branch (since the branching point)
    - Git performs the merge by moving a pointer
  - Recursive
    - The changes in the two branches are NOT conflicting
    - Git performs the merge by creating a new commit
      (with two parents, and all changes from both branches)
  - Fix conflicts manually
    - Changes in the branches are conflicting
    - Git adds markers to source files to indicate where the conflicts are
    - The developer fixes the conflicts by hand and commits the changes

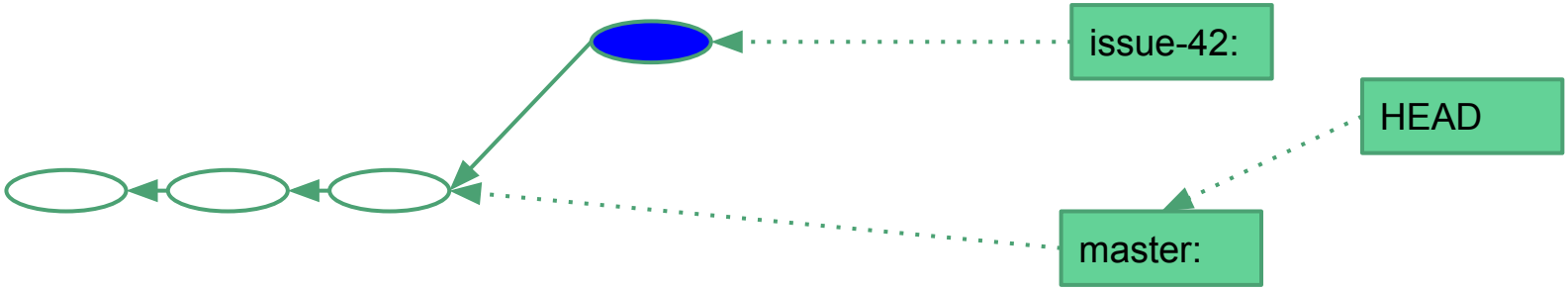
Options to Merge

1. Use branches locally, merge locally, and push to remote repo
    - This is not how it is usually done
2. Use branches remotely
    - with the full set of collaboration tools available
    - submit “pull requests” to the project owner

General advice: keep master as a stable source of truth and use a “develop branch” to integrate and test new changes

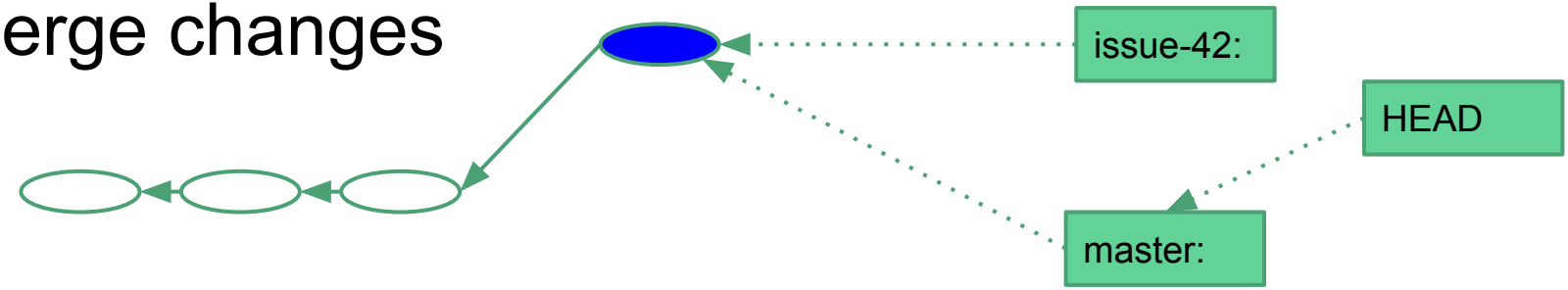
Option 1: local branching workflow

- Create a new branch:
  - `git checkout -b issue-42`
- Make your changes in branch
- Checkout master



Option 1: local branching workflow

- Create a new branch:
  - `git checkout -b issue-42`
- Make your changes in branch
- Checkout master
- Merge changes



Option 2: Pull Request

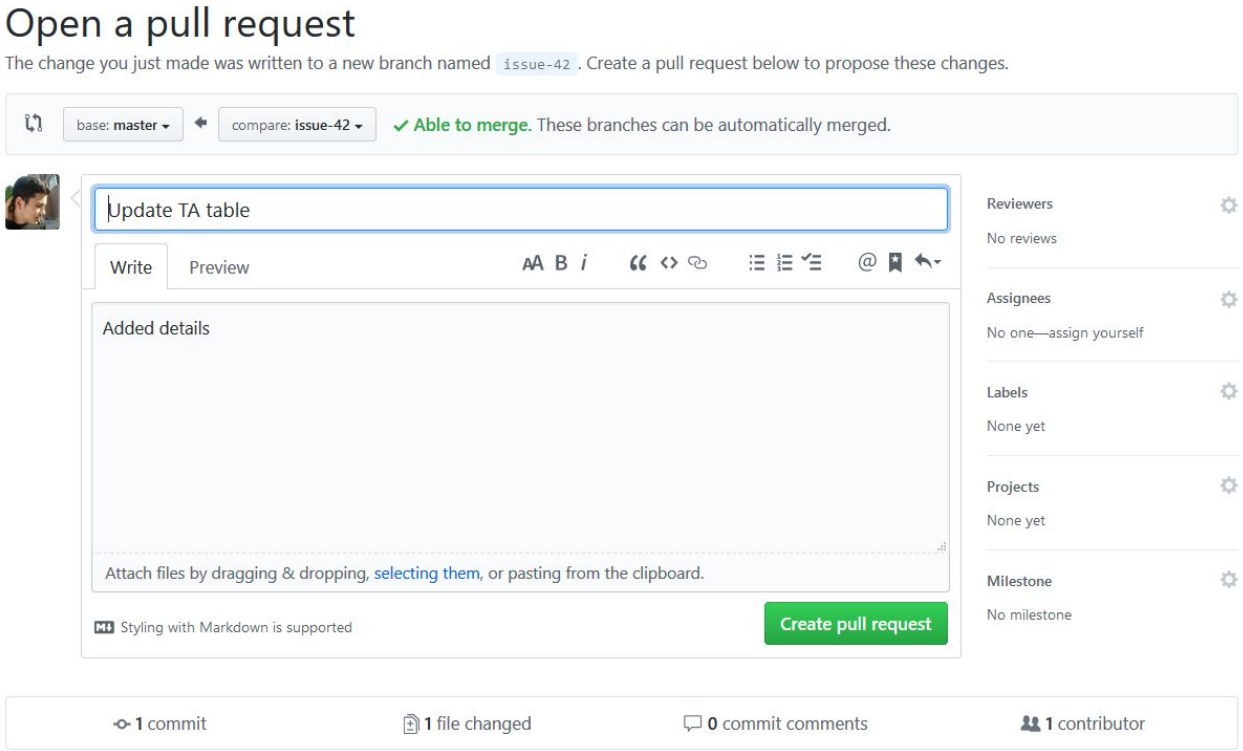
1. Rather than simply merging your feature branch locally, push the branch “upstream” to GitHub
2. Submit a pull request from the feature branch to master (or develop) branch
    - If you are not a project owner, you often won’t even have permission to push directly

This enables a proper code review process

Option 2: Pull Request

- Common workflow:
  - You create a PR between two branches
  - Your teammate(s) review the PR, makes comments, etc.
  - You fix whatever needs to be fixed, based on your teammates' review
  - Once everybody is happy, someone on the team merges the PR
- Pull requests are "open requests" to pull changes from one branch to another

Here is what it looks like on GitHub when you go to create the pull request



On GitHub, the “destination branch” is on the left, and the “source branch” is on the right.

If there are no conflicts, the project owner can merge the PR using GitHub's web UI:

If there *are* conflicts, they will need to be resolved

- Generally, it's on the developer to resolve the conflict
- This happens when the master branch has moved on while you were working locally
- Locally, pull and merge the master branch changes into your local branch
- Pushing will automatically update the PR
- GitHub has a nice UI for this