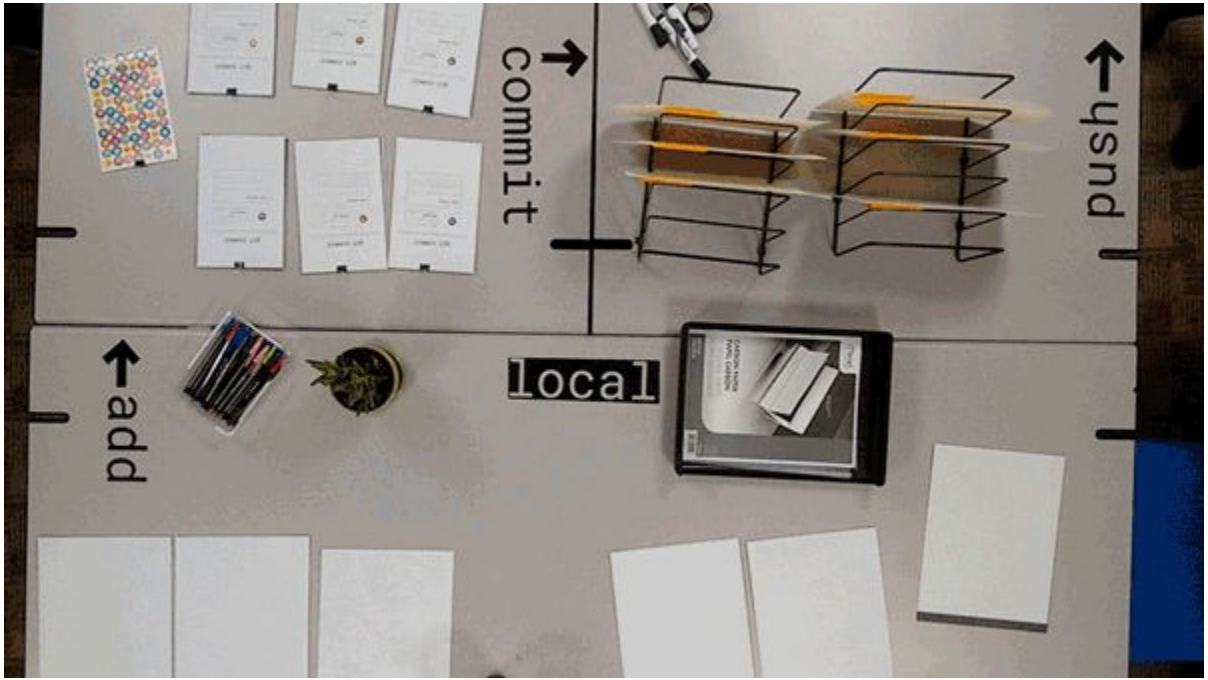
Git Workflow

// FLATIRON SCHOOL

Outline

1. The need for a Git workflow
2. Introducing branches
3. Git feature branch workflow
    a. Walk through the creation of a branch
    b. Pull requests
    c. Merging branches
4. Git fork workflow

Git helps you manage work done on projects



But without a consistent git workflow, collaboration is easier said than done

Without the agreement beforehand, collaboration feels like this

The team must all be in agreement on how the flow of changes will be applied from the beginning

Git makes it easy to experiment with branches

Branches have a short life cycle

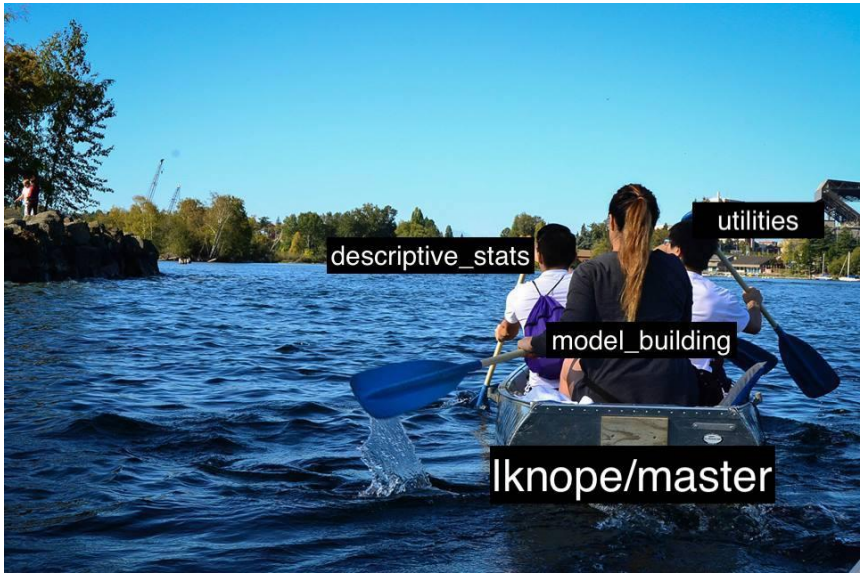
- They are named for a particular feature
- Once that feature branch is successfully pulled into the master branch, it should be left alone or deleted

Git ***feature branch*** workflow

Feature Branch Workflow

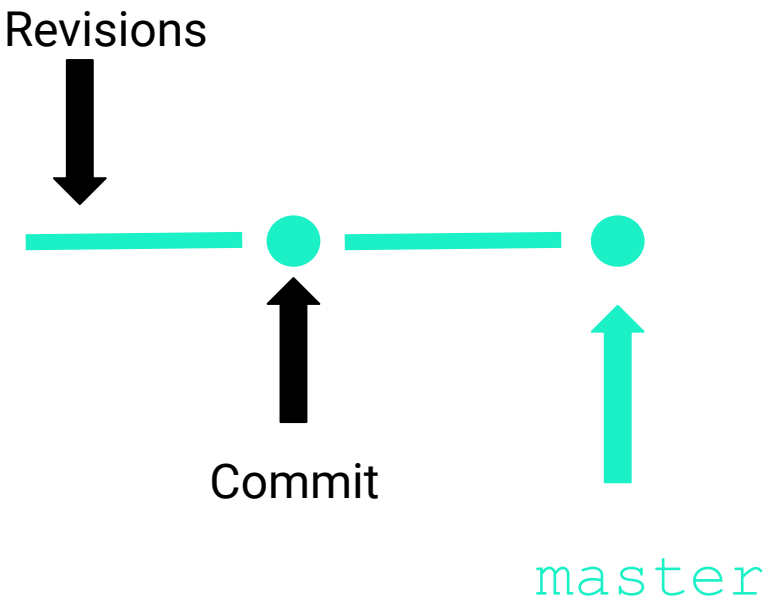
One repository with many collaborators

Add other developers as collaborators to your main repository so they can push their changes to it on the master branch



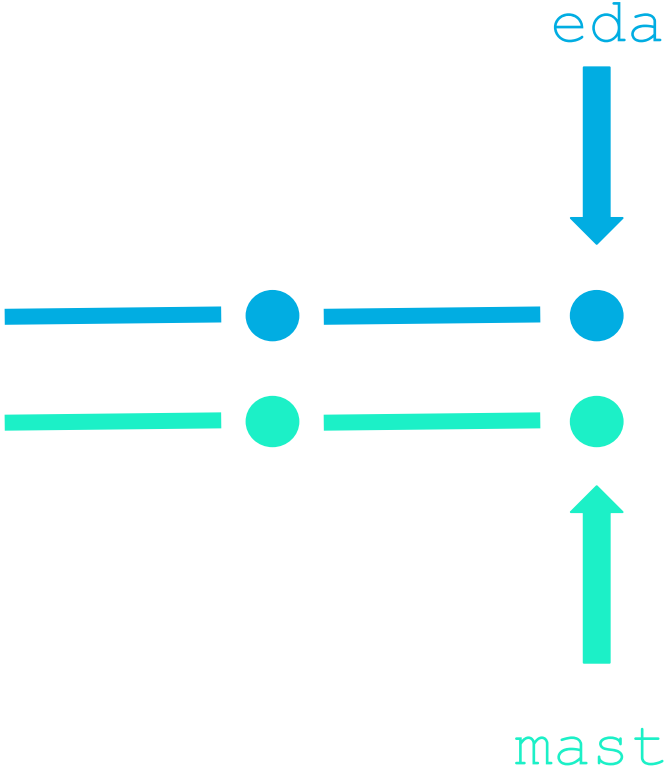
Branches are a labeled series of commits towards building some feature (i.e. task)

The default branch in git is called `master`



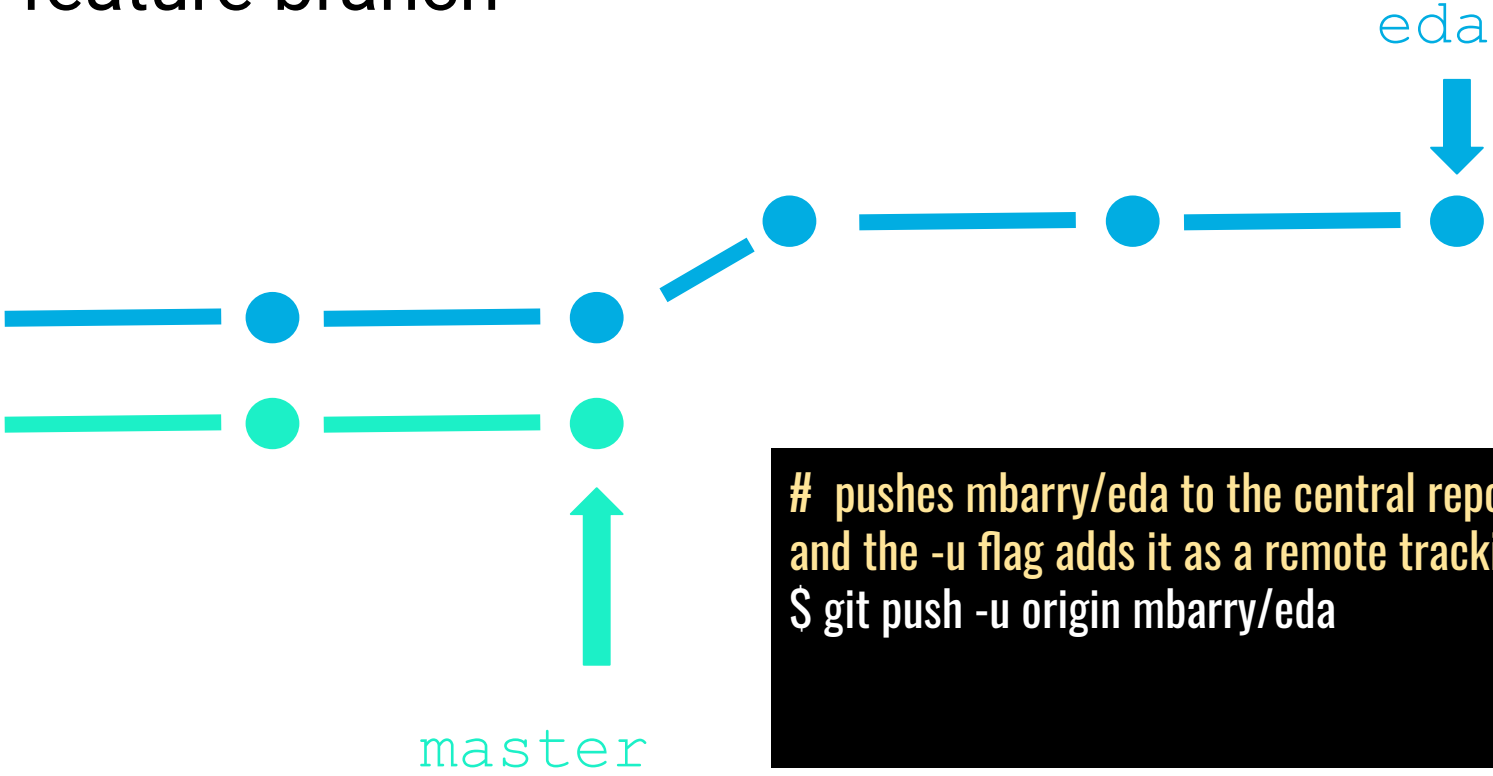

```
# switches the repo to the master branch
$ git checkout master

# pulls the latest commits from remote
$ git fetch origin

# merges the remote changes of master to the  local copy to incorporate latest changes
$ git merge origin/master
```

The name of a new branch takes both your username and the task you hope to accomplish



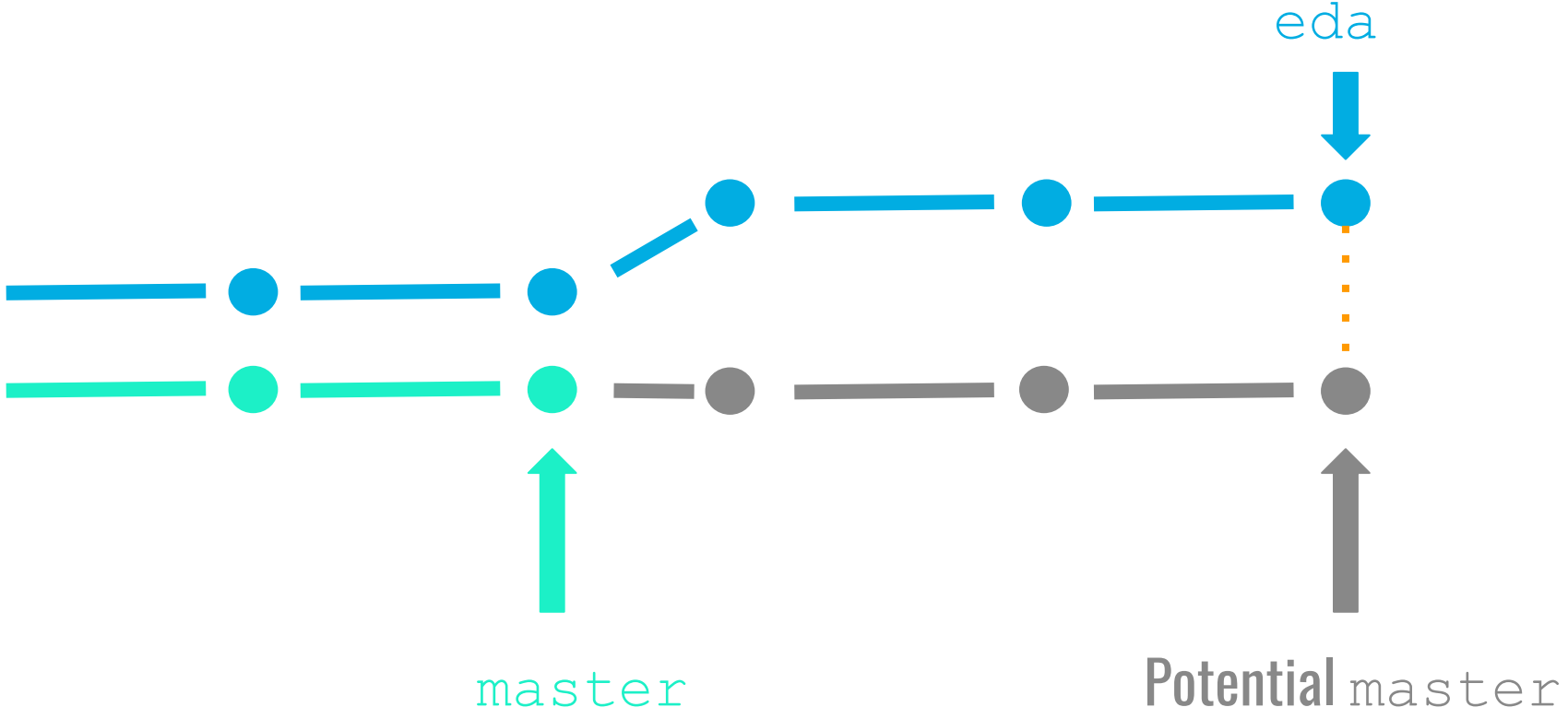

```
# checks out a branch called mbarry/eda based on master and the -b flag tells Git to create the branch if it doesn't already exist.
$ git checkout -b mbarry/eda
```

Update, add, commit, and push changes to your feature branch




```
#  pushes mbarry/eda to the central repository (origin),
and the -u flag adds it as a remote tracking branch
$ git push -u origin mbarry/eda
```

Aside from isolating feature development, branches make it possible to discuss changes via pull requests

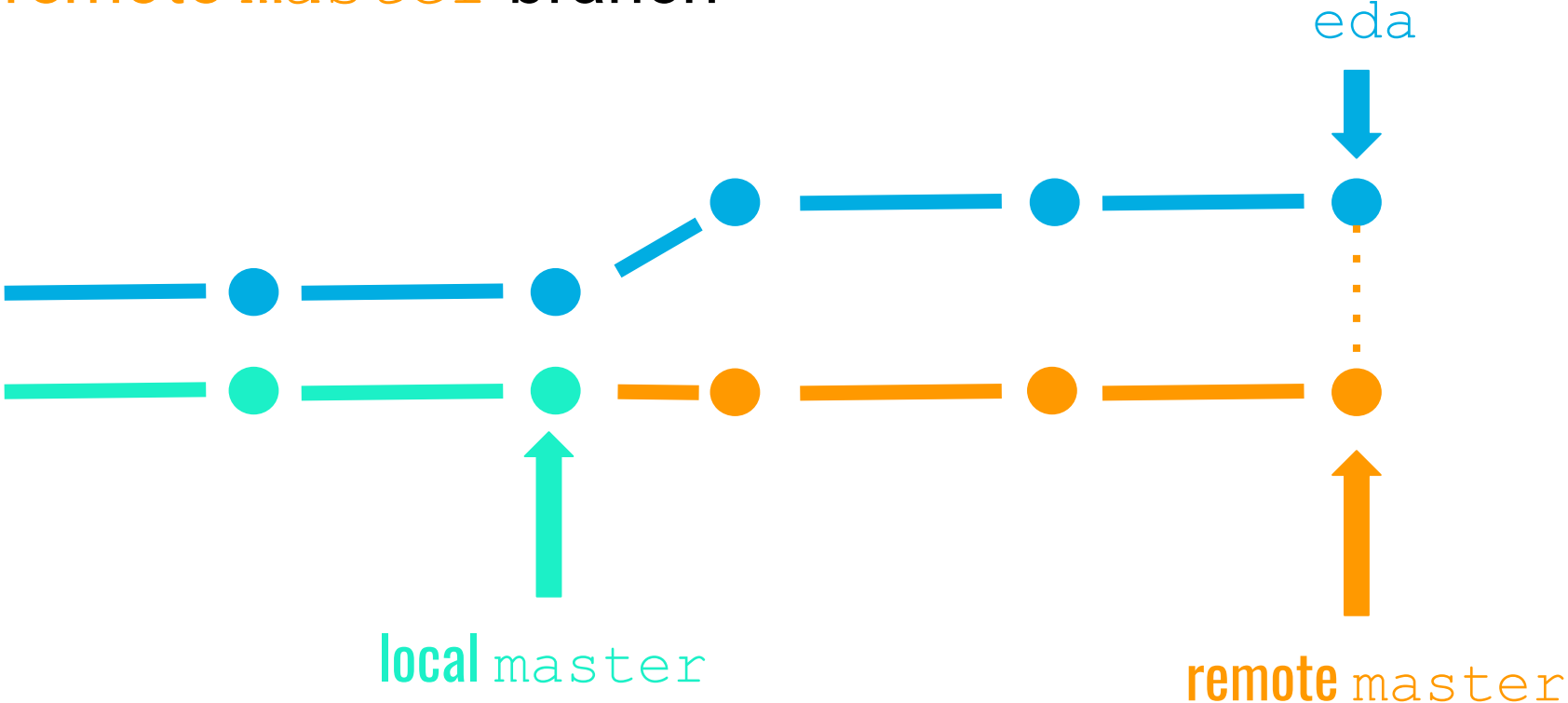


Pull requests give team member the opportunity to review the changes before they become a part of the main codebase

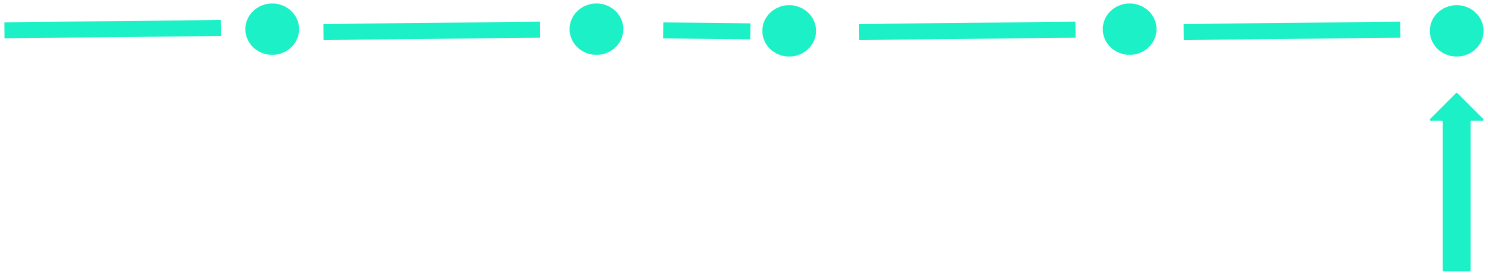
Pull requests that do not follow team guidelines should be declined

On the other hand, once pull requests are accepted, we are nearly done!

Once reviewed, pull requests are merged into the remote `master` branch



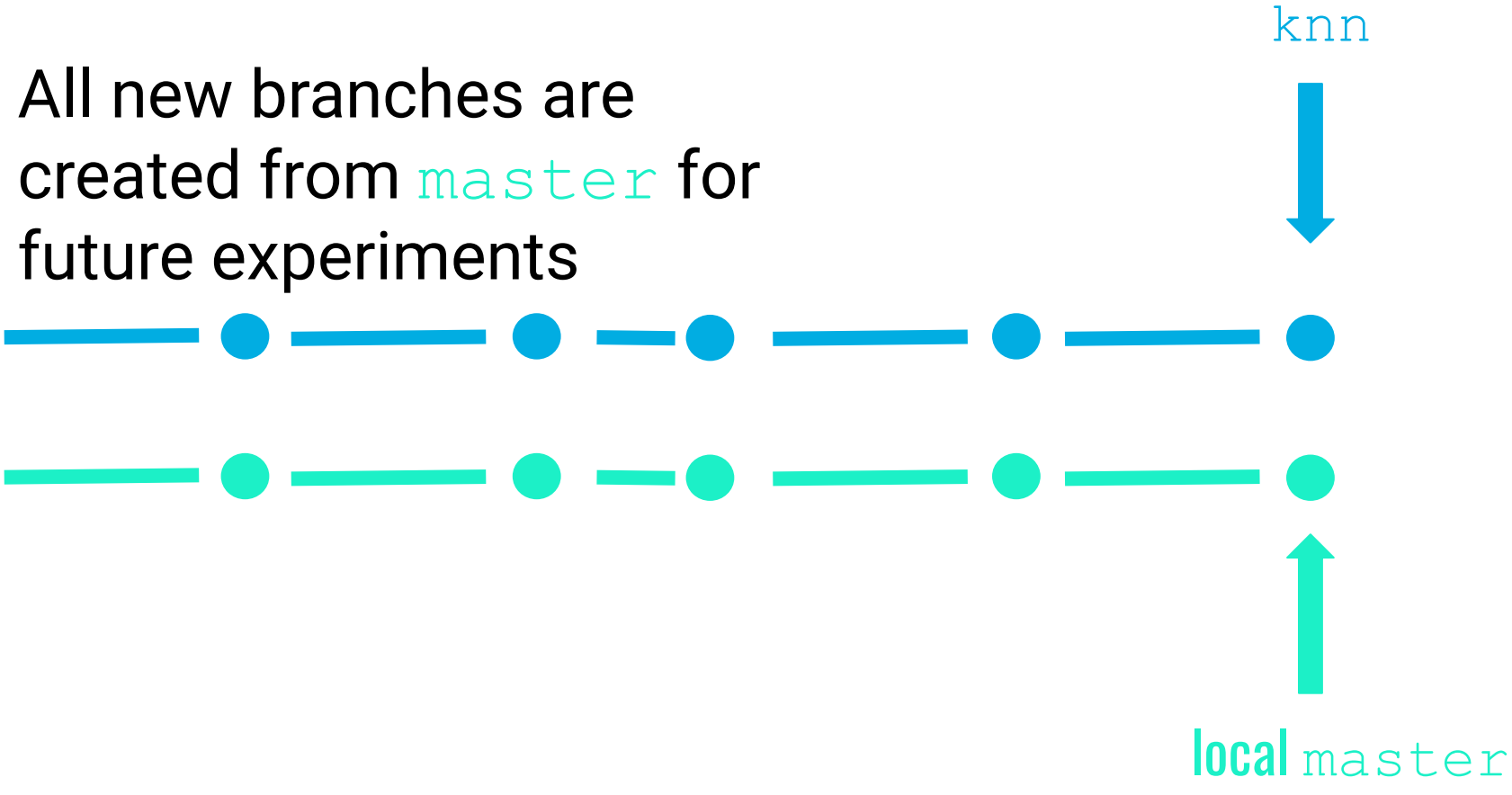
Reset your local copy of `master` to match the remote `master`




```
# switches the repo to the master branch
$ git checkout master
# pulls the latest commits from remote
$ git fetch origin
# resets the repo's local copy of master to match the latest version
$ git merge origin/master
```

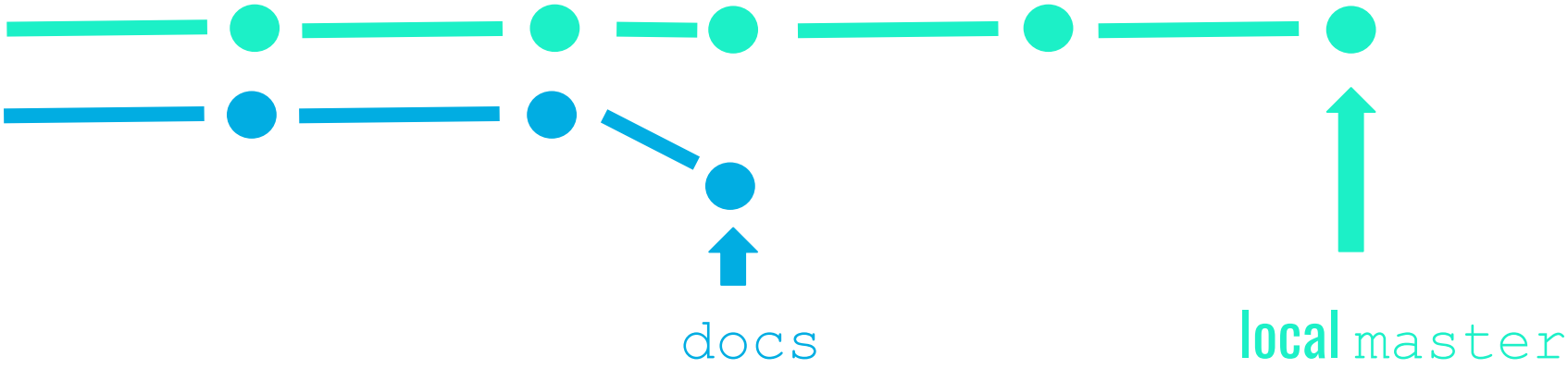
Now your local copy of `master` matches the remote `master`

All new branches are created from `master` for future experiments



...but what happens when you have other branches?

Once `master` has been updated, you might want to merge its changes into another branch



Use `$ git merge <branch>` to merge changes from one branch into another

When do I merge branches?

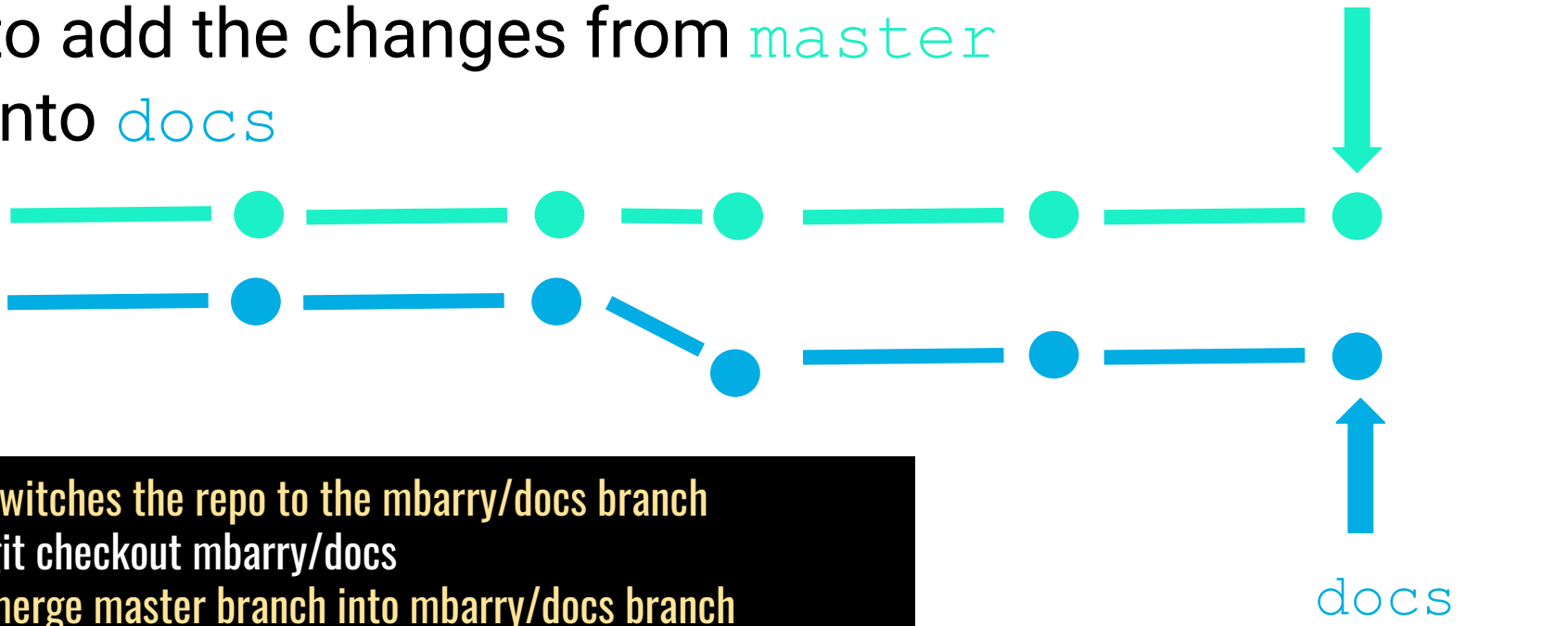
I should merge when:

An existing branch (e.g. `docs`) needs the changes from another branch (e.g. `master`)

I don't need to merge when:

The feature branch does not rely on any changes from another branch

Merge `master` into `docs` to add the changes from `master` into `docs`




```
# switches the repo to the mbarry/docs branch
$ git checkout mbarry/docs
# merge master branch into mbarry/docs branch
$ git merge master
```

By isolating features into separate branches, everybody can work independently and share changes with other developers

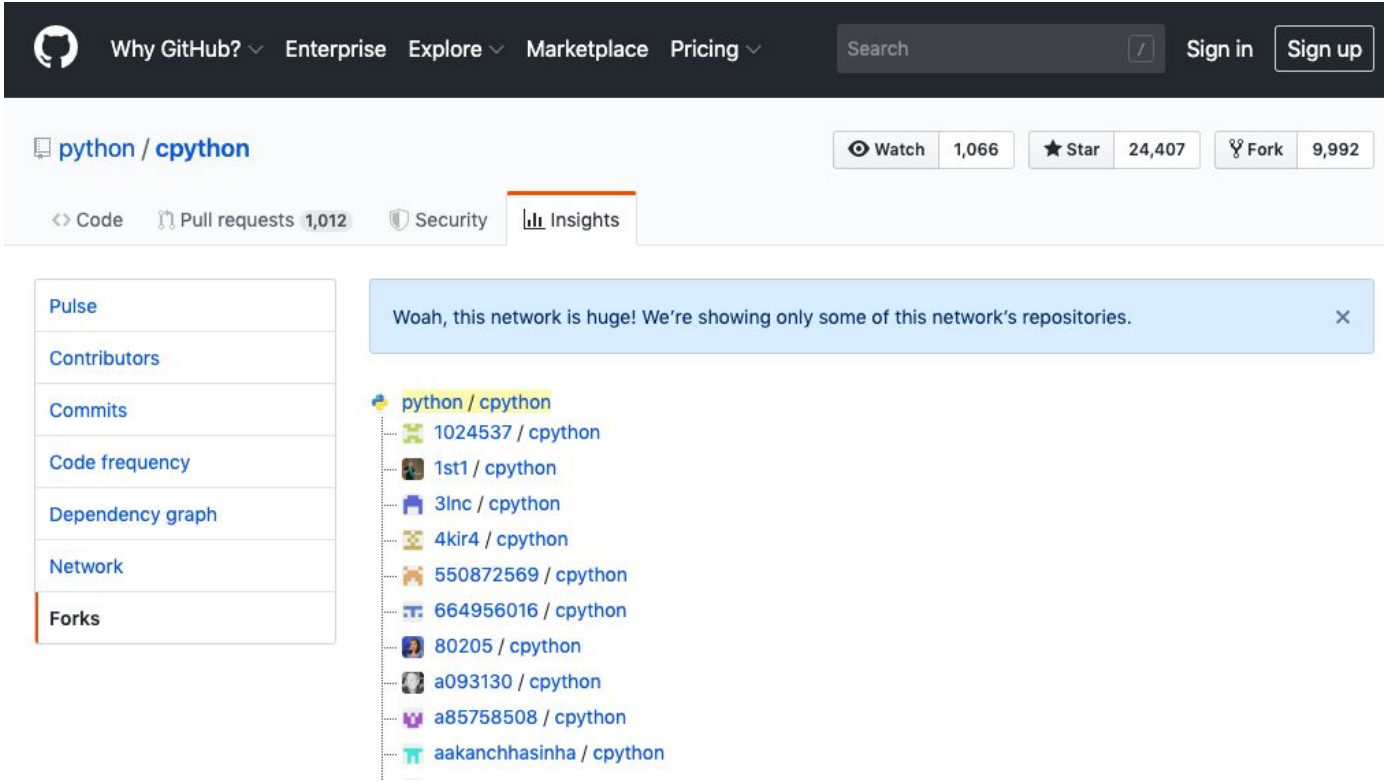
Git ***fork*** workflow

Fork Workflow

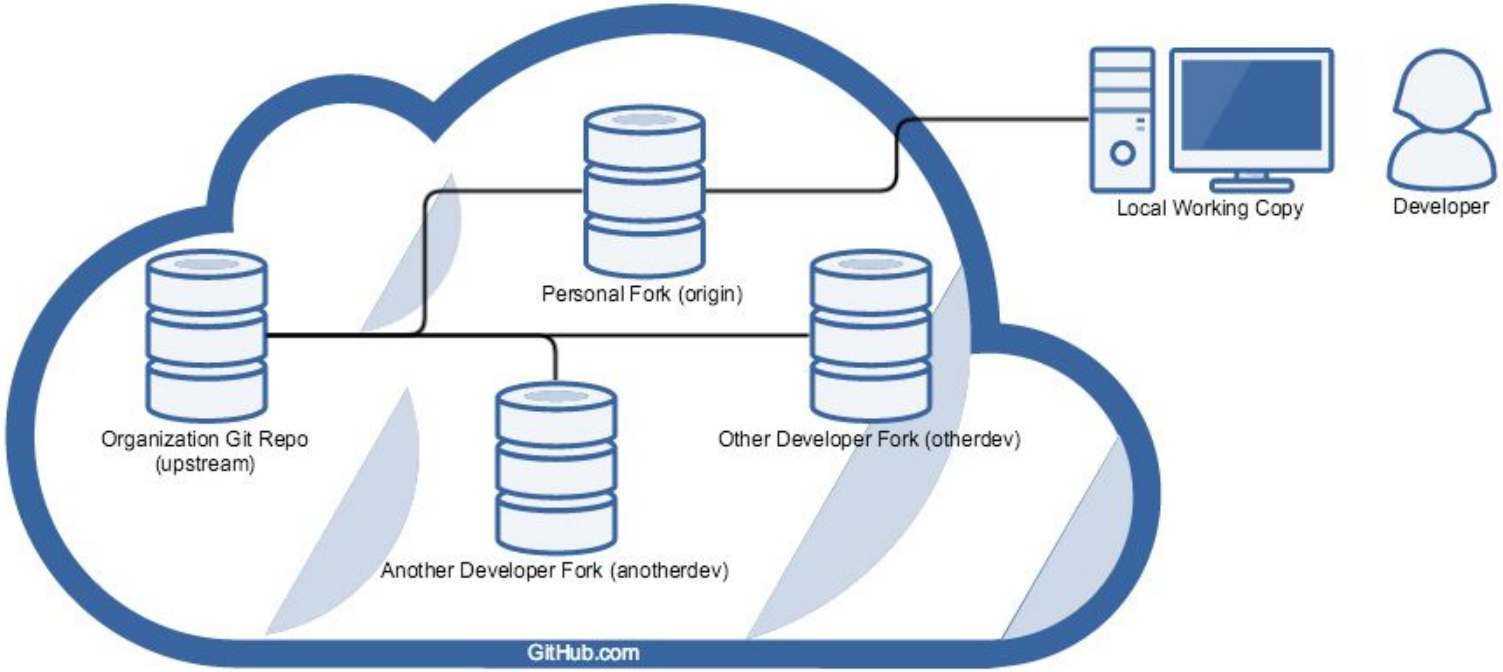
One repository per developer

Git fork workflow contains a project where you have one repository per collaborator

Provides a flexible way for large, organic teams (including untrusted third-parties) to collaborate securely

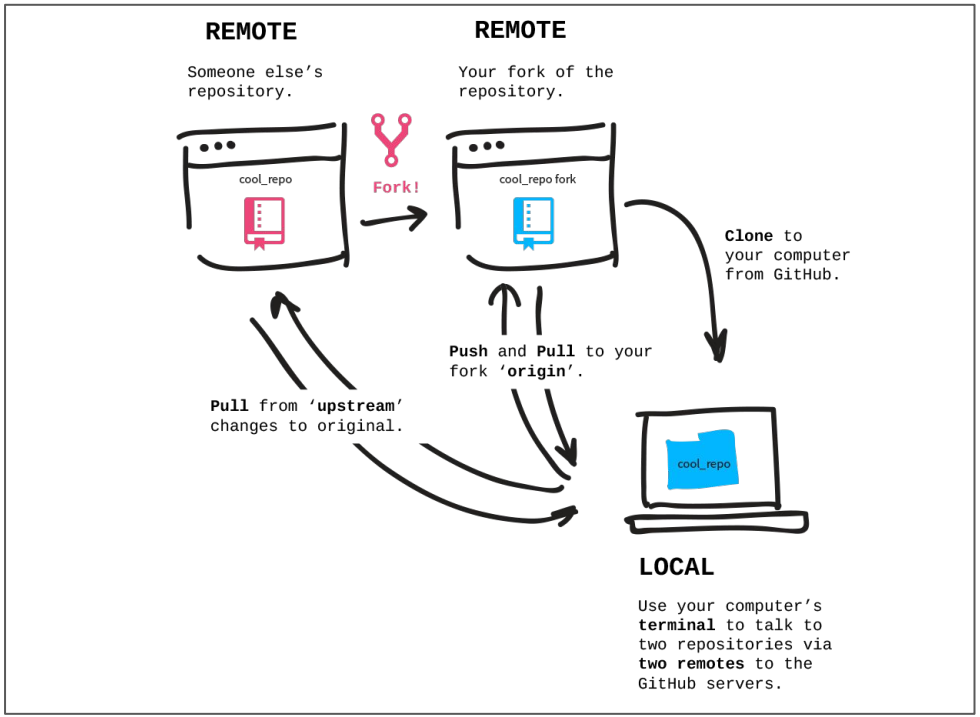


How this works between you and GitHub

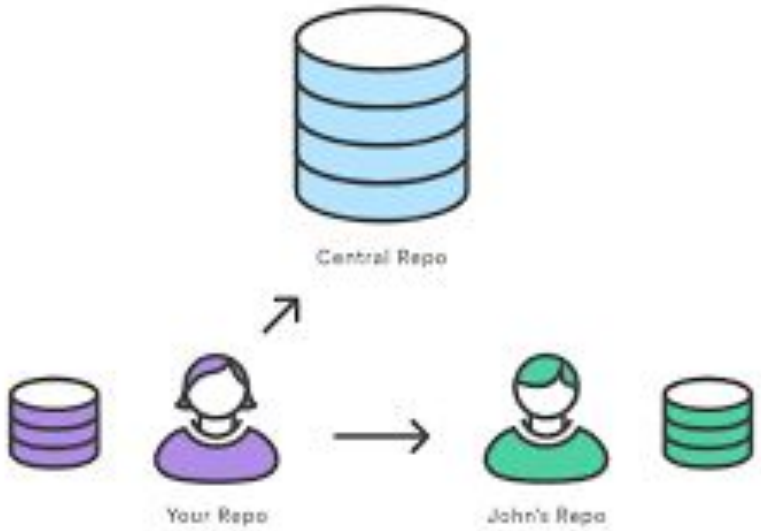


How this differs from the feature branch workflow:

Add other people's repositories in addition to other people's branches



`$ git remote add <name> <url>` creates a new connection to another person's repository



`$ git remote -v` lists all remote connections associated in your repository

`$ git branch -a` shows both local and remote branches

`$ git fetch <name>` downloads objects from another repository

`$ git checkout -b <name of new branch> <repo name/branch name>` will create a new branch from a particular branch in someone else's repository

`$ git merge <name>/<branch_name>` will merge someone else's branch from another repository into your currently checked out branch

The team needs to decide what the git workflow looks like so everyone is aware when they need to update their local version of `master`


Thank
you!
Pusheen.com