Intermediate Git

Day 4: Collaborating on GitHub

Raman A. Shah

Back to the Day 2 repo

Do you have a clean sample repository from last time?

```
cd sample
git status
```

If not, either clone the course repo or refresh it with `git pull origin master`:

```
git clone https://github.com/\
ramanshah/intermediate_git.git
```

Then decompress an updated sample repository:

```
cp ./intermediate_git/day4/sample.tgz .
tar xzvf sample.tgz
cd sample
git status
```

Manage GitHub: git remote

Try these on your intermediate_git and sample repos. What do you get?

```
git remote
git remote -v
git remote show origin
```

To connect your repository to a server:

```
git remote add [remote_name] [url]
```

To delete the connection:

```
git remote rm [remote_name]
```

Remote branches

```
[remote]/[branch]
```

A read-only pointer to the state of `[branch]` on `[remote]` the last time you checked.

Tracking branches

A local branch is sometimes automatically set up to correspond to a remote branch:

- When you `git clone` a repo, `master` tracks `origin/master`.
- When you `git push` a new branch, that gets set up to track as well.

But to check out someone else's branch from a remote, you have to set up tracking yourself:

```
git checkout -b [branch] \
  [remote]/[branch]
```

Push, fetch, and pull

To put local commits on the remote server:

```
git push [remote] [branch]
```

To download remote objects:

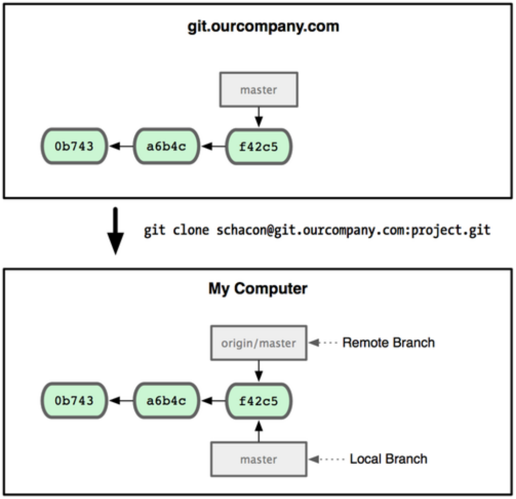
```
git fetch [remote]
```

To download remote objects *and* merge them into your current branch:

```
git pull [remote] [branch]
```

`git pull` = `git fetch` + `git merge`!

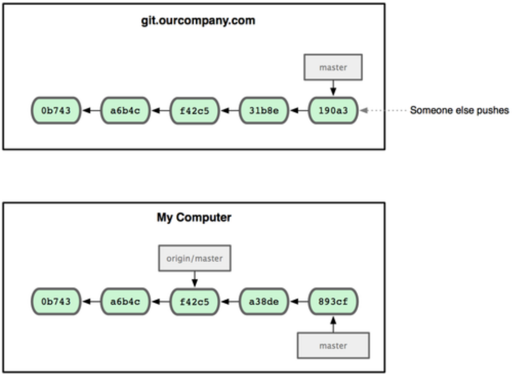
Everyone's on master



A standard `git clone`.

Scott Chacon, *Pro Git*, Fig. 3-22. CC-BY-NC-SA. https://progit.org/

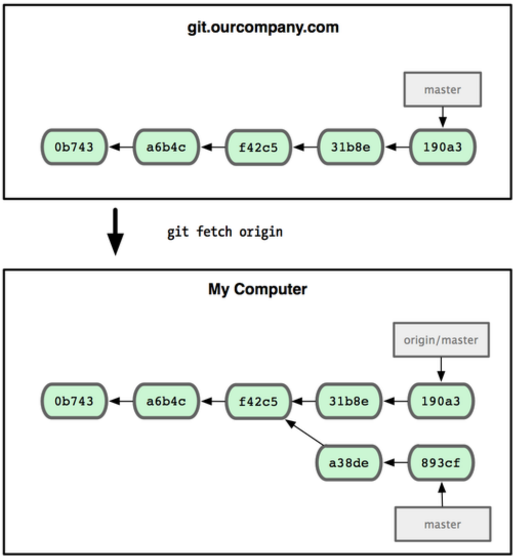
Everyone's on master



Local `master` has diverged from `origin/master`.

Scott Chacon, *Pro Git*, Fig. 3-23. CC-BY-NC-SA.
https://progit.org/

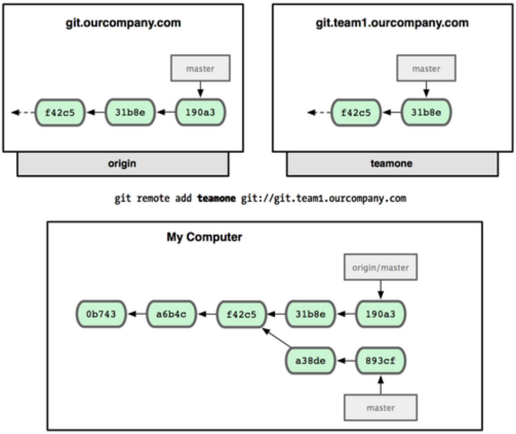
Everyone's on master



You have to `git fetch`, not `git pull`.

Scott Chacon, *Pro Git*, Fig. 3-24. CC-BY-NC-SA. https://progit.org/

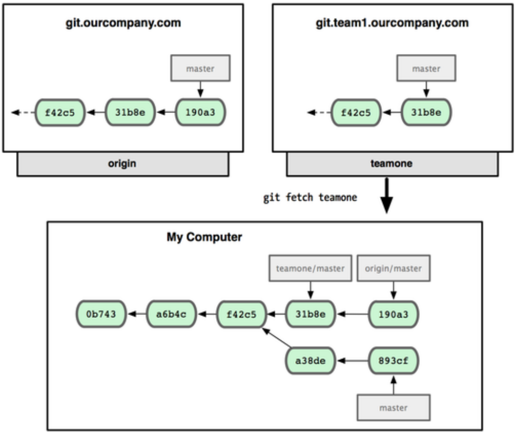
Everyone's on master



You can add a second remote.

Scott Chacon, *Pro Git*, Fig. 3-25. CC-BY-NC-SA. https://progit.org/

Everyone's on master



Again, `git fetch`, not `git pull`, is appropriate.

Scott Chacon, *Pro Git*, Fig. 3-26. CC-BY-NC-SA. https://progit.org/

Recommendation

When collaborating, don't work on `master`. If you do, avoid `git pull`.

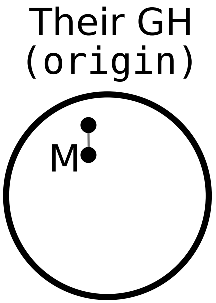
Pull requests—can push

Why wouldn't you just push to `master`?

- Pull requests give a mechanism for peer review of code.
- Others might expect `master` to be working or stable.
- The `master` branch might need to be ready to go, *e.g.*, deployable.

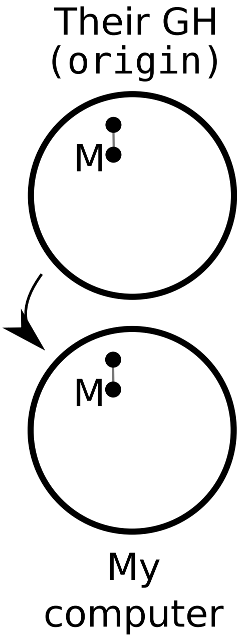
Even if you have push access, submit changes as pull requests as a matter of *safety* and *etiquette*.

Pull requests—can push



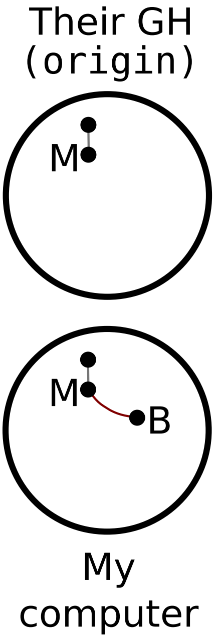
Someone has a project.
They give you push access.

Pull requests—can push



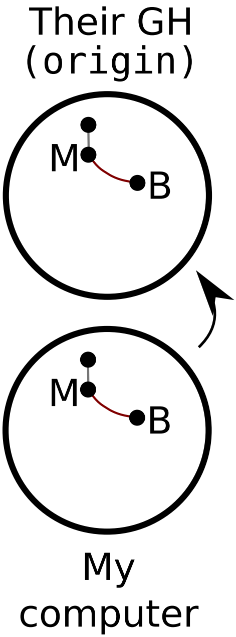
```
git clone [url]
```

Pull requests—can push



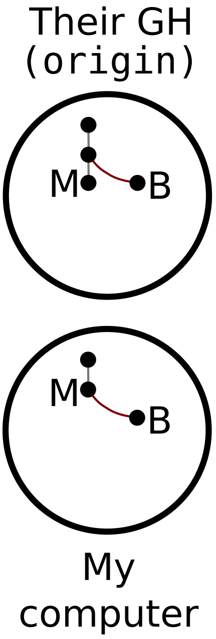
```
git checkout -b my_branch
git commit [...]
```

Pull requests—can push



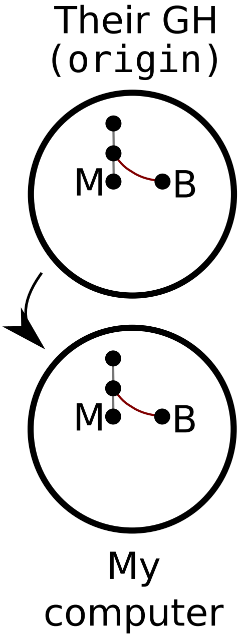
`git push origin my_branch`

Pull requests—can push



They put some new work on master.

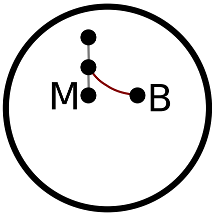
Pull requests—can push

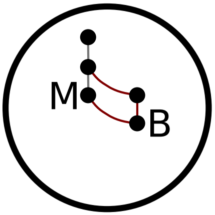


```
git checkout master
git pull origin master
```

Pull requests—can push

Their GH
(origin)

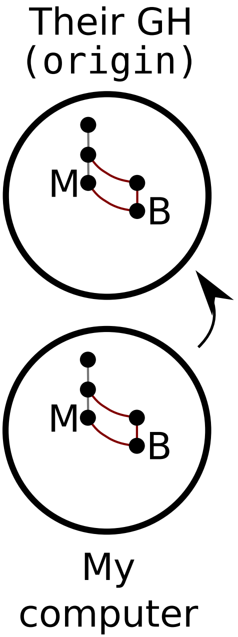




My
computer

```
git checkout my_branch
git merge master
```

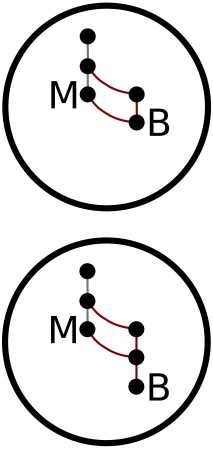
Pull requests—can push



```
git push origin my_branch
```

Pull requests—can push

Their GH
`(origin)`

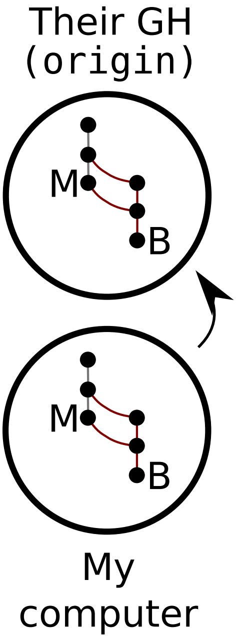


My
computer

Click "New Pull Request;" code review

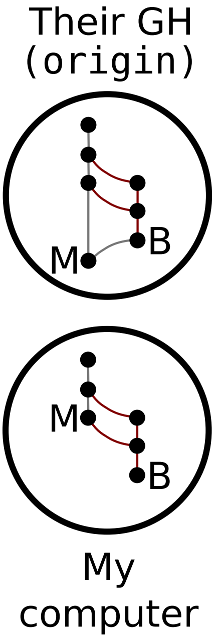
`git commit [...]`

Pull requests—can push



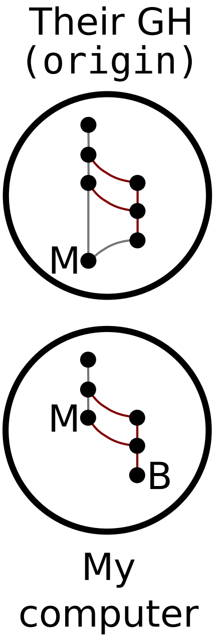
```
git push origin my_branch
```

Pull requests—can push



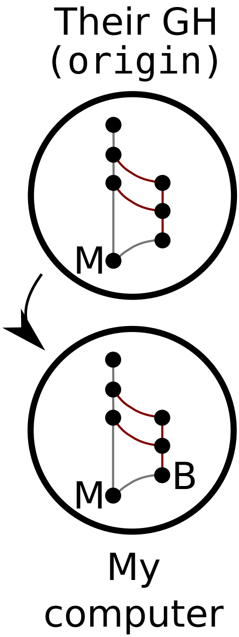
"LGTM"—"Looks Good To Me."
You click "Merge Pull Request."

Pull requests—can push



You click “Delete Branch.”

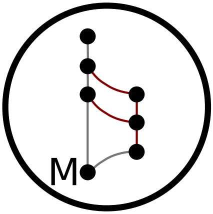
Pull requests—can push

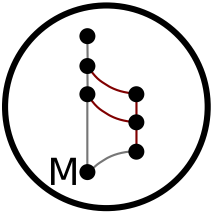


```
git checkout master
git pull origin master
```

Pull requests—can push

Their GH
`(origin)`





My computer

```
git branch -d my_branch
```

Done!

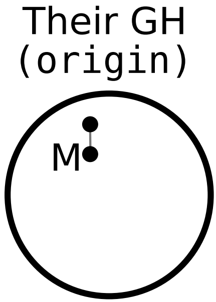
Pull requests—can't push

If you don't have push access, you have to:

- *Fork* the repo on GitHub.
- Curate your fork so that the changes in the fork can be applied to the original repo in a single pull without merge conflicts.
- In some projects, rebase your work so that the commits are "nice" and result in a fast-forward merge.

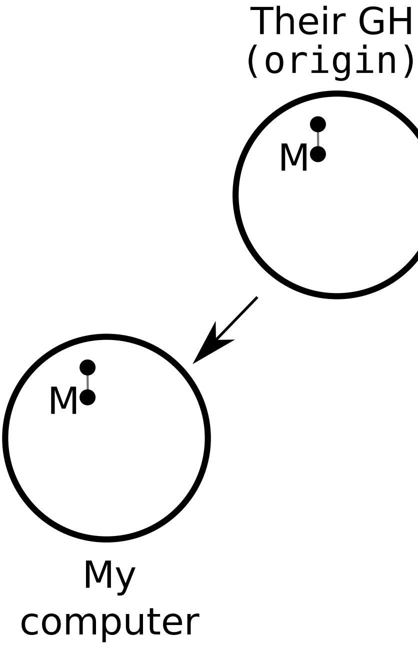
This is the default case in open source work.

Pull requests—can't push



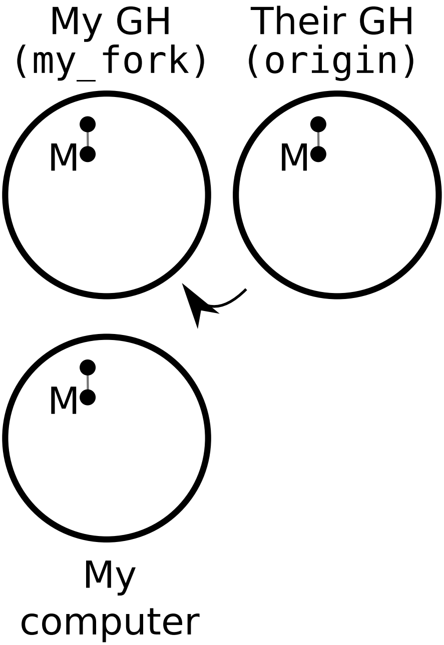
Someone has a project.

Pull requests—can't push



```
git clone [url]
```

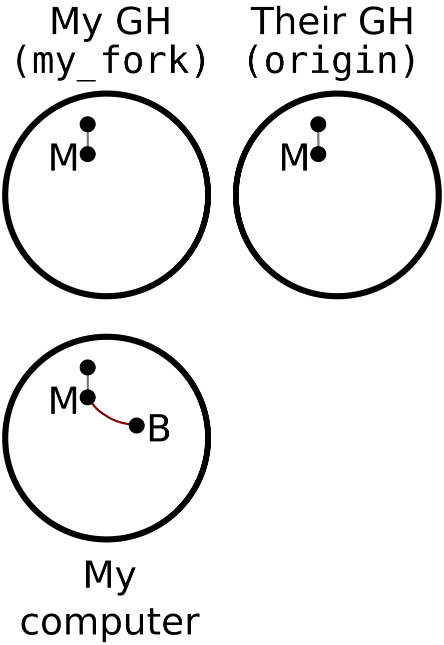
Pull requests—can't push



Click "Fork" on GitHub

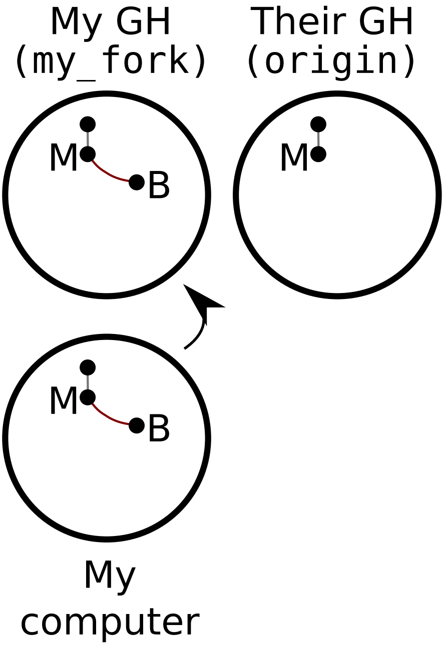
```
git remote add my_fork [url]
```

Pull requests—can't push



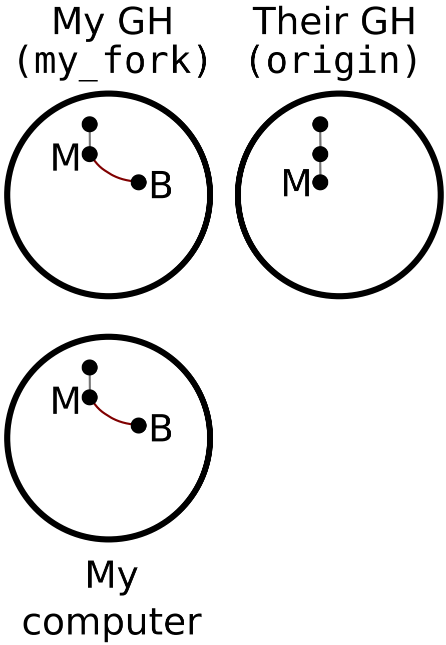
```
git checkout -b my_branch
git commit [...]
```

Pull requests—can't push



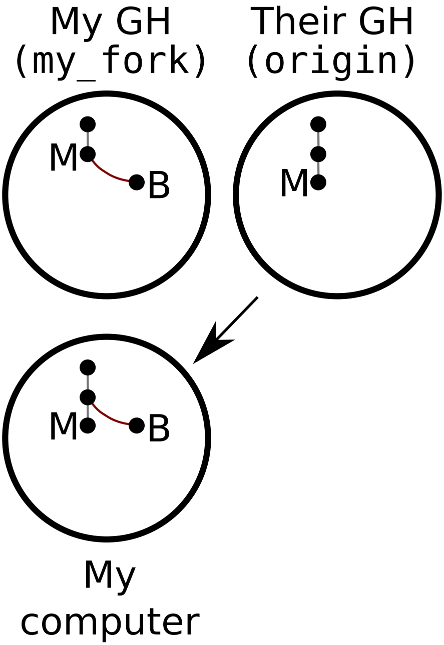
```
git push my_fork my_branch
```

Pull requests—can't push



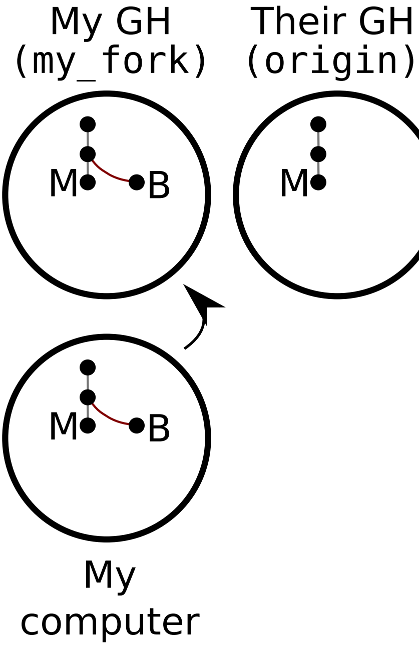
They put some new work on master.

Pull requests—can't push



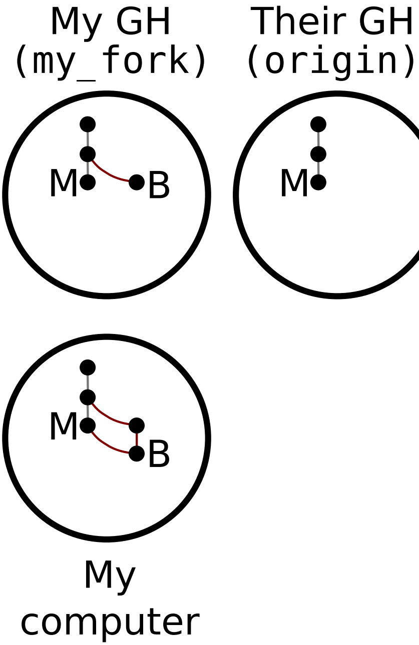
```
git checkout master
git pull origin master
```

Pull requests—can't push



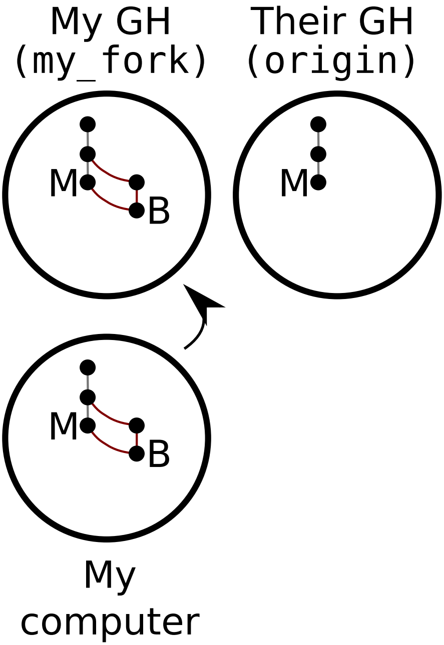
```
git push my_fork master
```

Pull requests—can't push



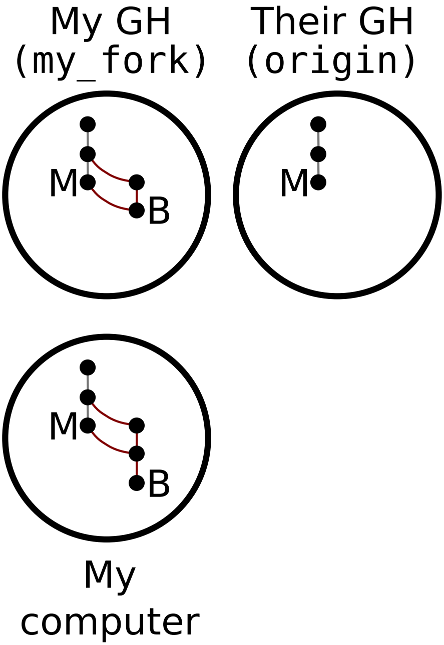
```
git checkout my_branch
git merge master
```

Pull requests—can't push



```
git push my_fork my_branch
```

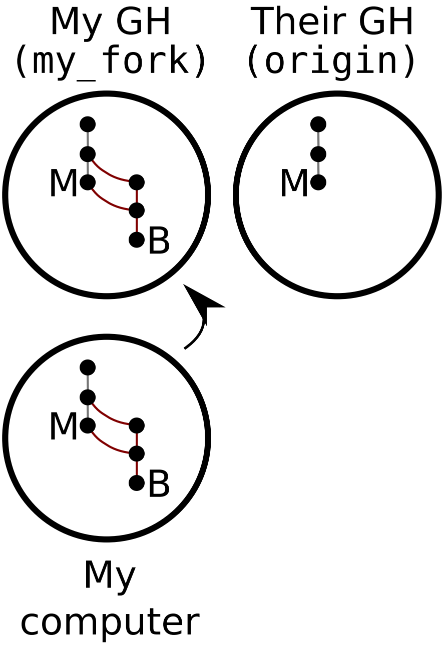
Pull requests—can't push



Click "New Pull Request;" code review

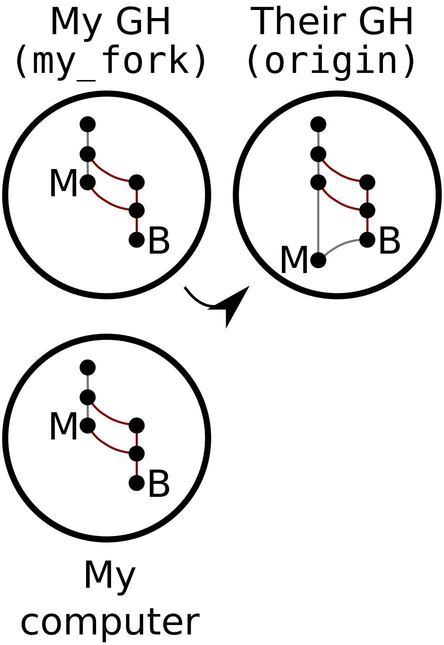
`git commit [...]`

Pull requests—can't push



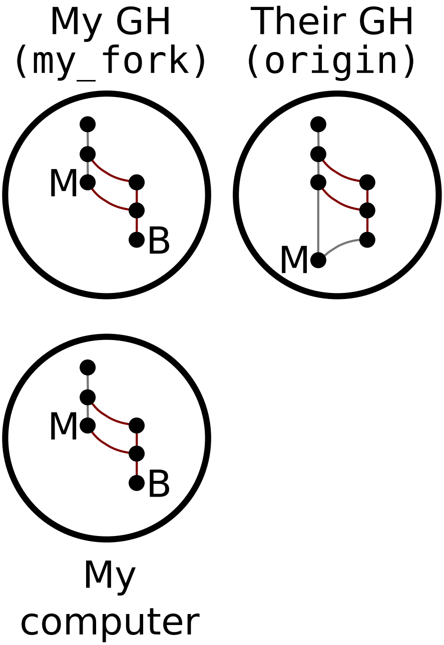
```
git push my_fork my_branch
```

Pull requests—can't push



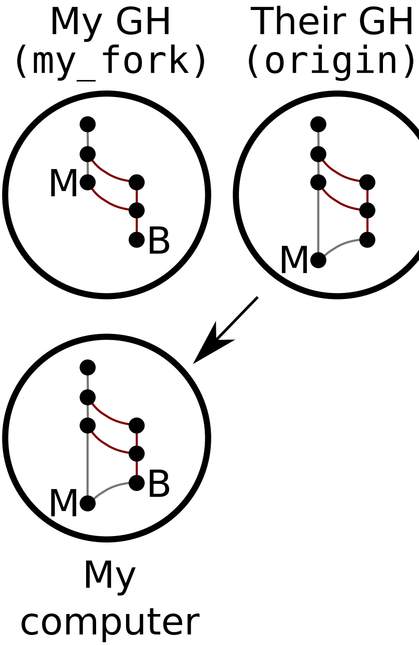
They're happy. They click "Merge Pull Request."

Pull requests—can't push



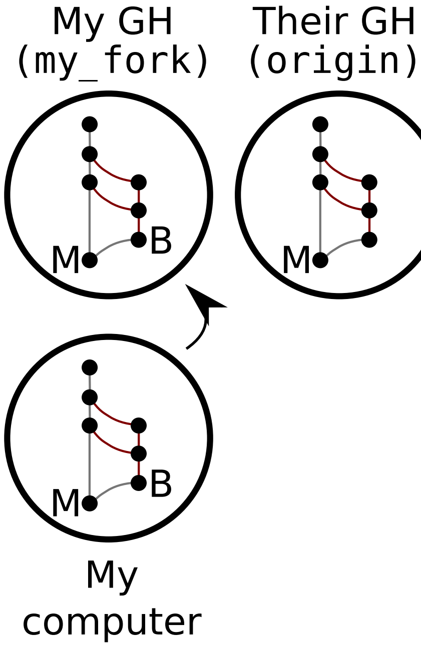
They click "Delete Branch."

Pull requests—can't push



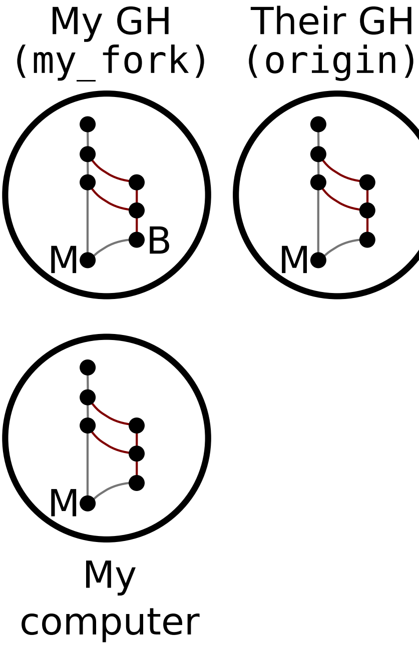
```
git checkout master
git pull origin master
```

Pull requests—can't push



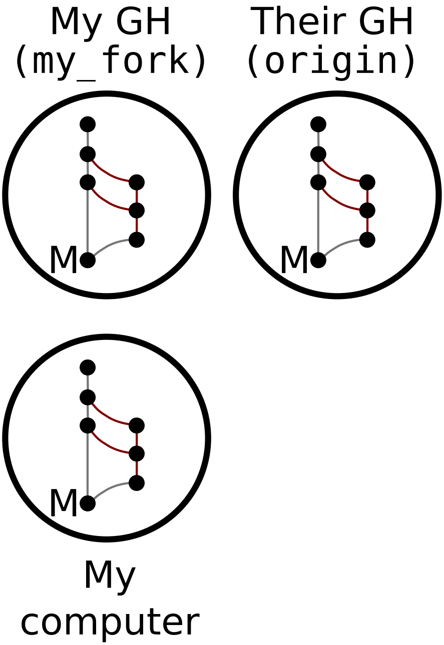
```
git push my_fork master
```

Pull requests—can't push



```
git branch -d my_branch
```

Pull requests—can't push



```
git push my_fork :my_branch
```

Done!

Keep pull requests small!

Shorter pull requests involve:

- Less *time* to diverge from `master`—hence fewer merge conflicts
- Less *cognitive load* for the developer and maintainer to comprehend and communicate the reasoning behind the changes

My personal rule of thumb is that a pull request is too big if you exceed:

- Roughly one week of active development time
- Roughly 150 lines of source code

Conclusion