Lecture 8
Integration Manager Workflow and Basic Rebasing

Sign in on the attendance sheet!
Last Time

- Remote tracking branches
- Introduced centralized Git workflow
- Made a pull request
Today

• What happens when we merge pull requests
• How we keep our fork up to date
• What is rebase and what does it have to do with any of this?
Merging a Pull Request on Github

- Pull request into `blessed-repo:master` from `remote-repo:my-feature`
- Clicking the merge pull request button is equivalent to executing the following commands in the blessed-repo repository itself:

  ```
  Note that we can’t actually execute commands in the remote repository but Github can
  $ git checkout master
  $ git remote add remote-repo https://github.com/user/remote-repo.git
  $ git fetch remote-repo
  $ git merge --no-ff remote-repo/my-feature
  ```

  Must merge cleanly! PR interface will not allow merge until conflicts are resolved
The integration manager can inspect and pull in your changes.
Merging a Pull Request Manually

- Pull request into `blessed-repo:master` from `remote-repo:my-feature`
- In the “integration manager’s” clone of `blessed-repo`:
  ```bash
  $ git checkout master
  $ git pull
  $ git remote add remote-repo https://github.com/user/remote-repo.git
  $ git fetch remote-repo
  $ git merge --no-ff remote-repo/my-feature
  $ git push origin master
  ```
You need to keep your fork up to date
You need to keep your fork up to date

In the private developer repo
$ git remote add upstream https://github.com/autolab/Autolab.git
$ git fetch upstream
$ git checkout master
$ git merge upstream/master
$ git push origin master
Rebasing

• Rebasing rewrites your git history, replaying the diffs of your commits
• Useful as an alternative to merging when you want to keep history neat
Merge vs. Rebase

```
git merge feature
```

```
git rebase master
```
Activity/Homework

Rebase the changes you made a PR for last week on top of the new upstream/master and push to your branch. Your PR should update automatically.