Git Workshop

Turning y'all into mini gizzards

Git Overview

File System	Staging Area	Local Repo	Server Repo
Any changes	Changes you want to save	Changes are done (or at a save point)	Everyone can see changes

`git add file.c`

`git commit`

`git push`

`git fetch`
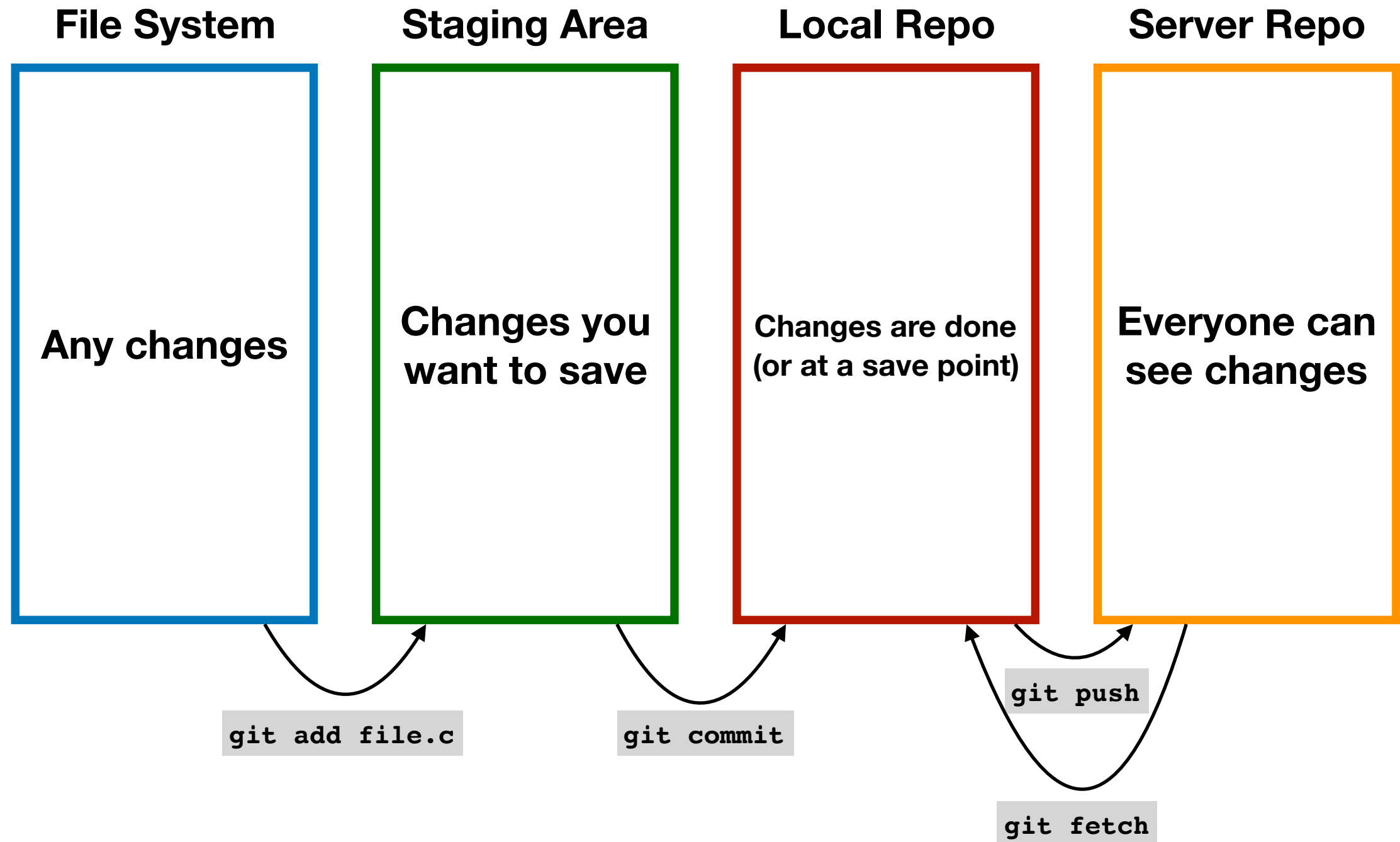
Branching

Used to commit features currently being worked on / tested
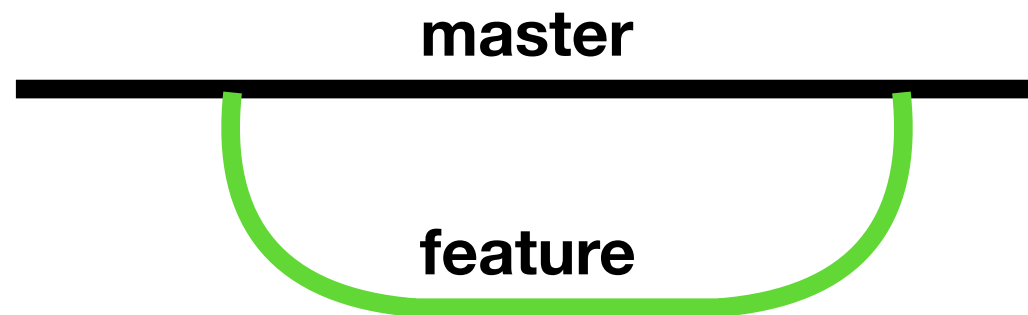
master

feature

feature_experimental
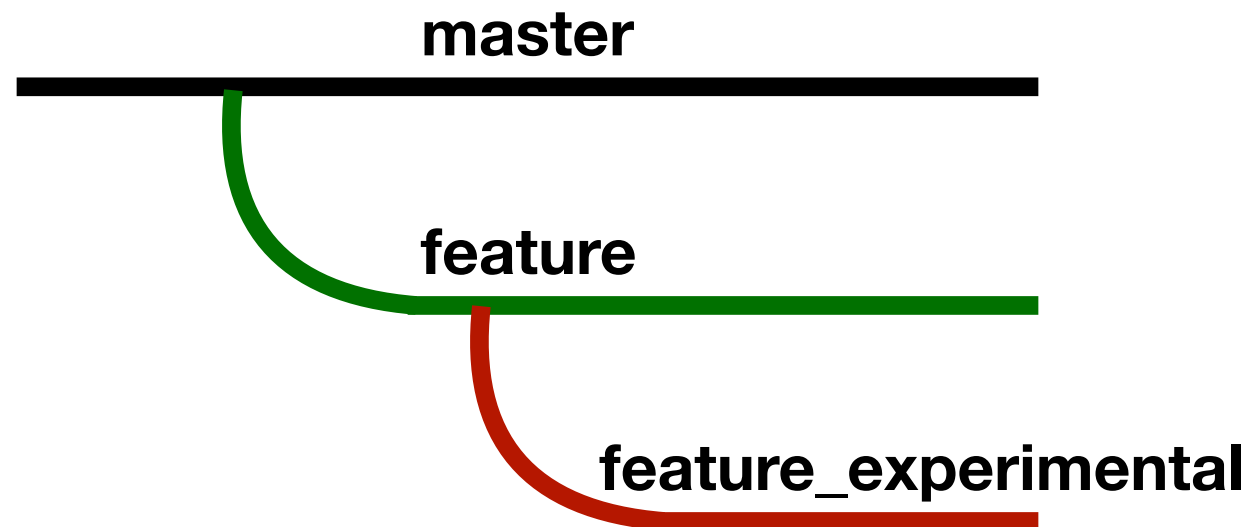
Create new branch: `git checkout -b branch_name`
Switch to branch: `git checkout branch_name`
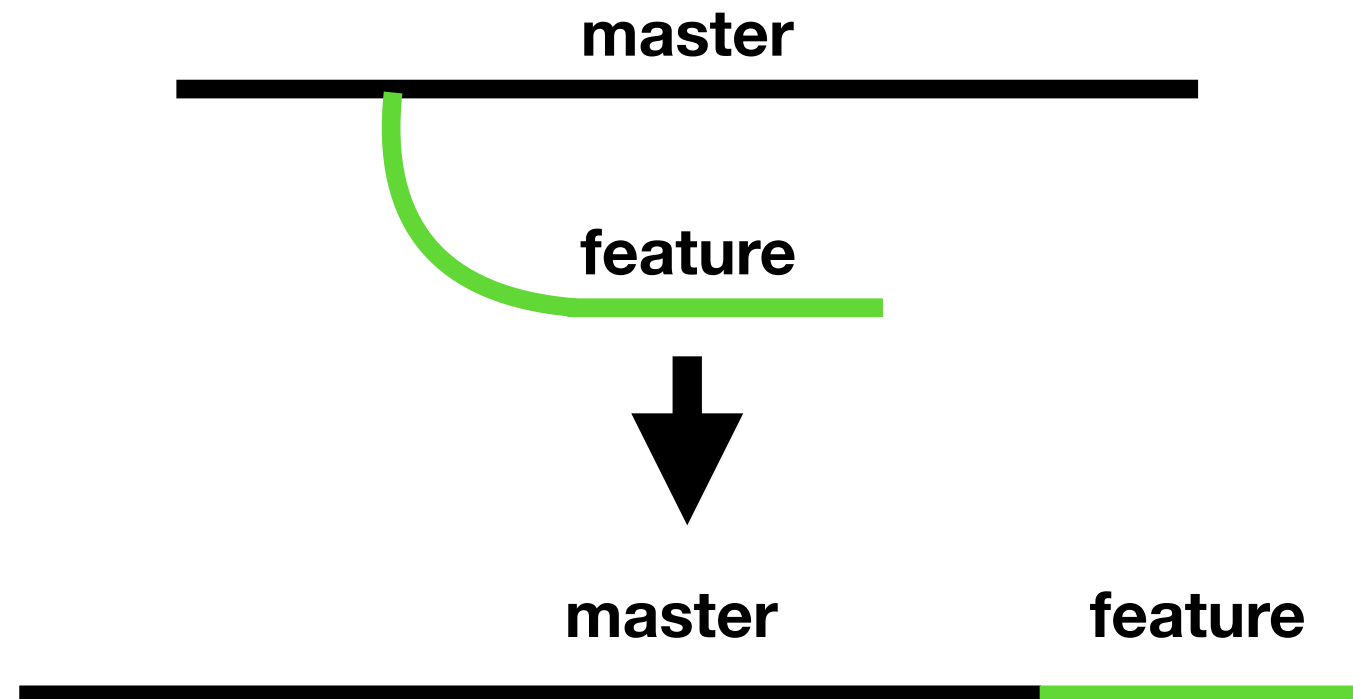
Rebasing / Merging

Used to combine two branches

Merge

master
feature

`git merge feature`

(While master is checked out)

Rebase

master
feature

master   feature

`git rebase master`

(While feature is checked out)

Rebasing is preferable because it creates a cleaner branch history

Tips

Copy leo.adberg.com/gitconfig into your ~/.gitconfig

Now, `git lg` gives nice looking git histories:

Tips

Someone else pushed changes on the branch you were working on?

```
 ! [rejected]           feature -> feature (non-fast-forward)
error: failed to push some refs to 'git@github.com'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Merge the remote changes (e.g. 'git pull')
hint: before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

FETCH AND REBASE!!!

```
git fetch
```

```
git rebase origin/feature
```

(While feature is checked out)

This keeps unnecessary merges from showing up

Tips

Forgot to add a file before committing?

```
git add file.c
```

```
git commit --amend
```

Already pushed the commit?

Just make a new commit with the new changes, overwriting server history is dangerous

Tips

Use an SSH key to push without logging in

```
ssh-keygen -t rsa
```

Then add the public key (id_rsa.pub) to your Github account in settings

(Also, this works for any computer you SSH into, which is really useful for lab computers)

Test Repo

Clone https://github.com/neelayjunnarkar/git-workflow-test/
and PM Neelay for contributor access to the repo

Try completing these, feel free to ask for help:

1. Create your own branch under your name
2. Add some files and commit and push to your branch
3. Make changes to someone else's branch and push those
4. Rebase your branch onto master2
   - Hard Mode: Squash multiple commits from your branch into one commit
     - Hint: rebase -i
   - We do this so master isn't cluttered with WIP commits, but one commit per major change
   - In this example, master2 is a substitute for a master that has been updated