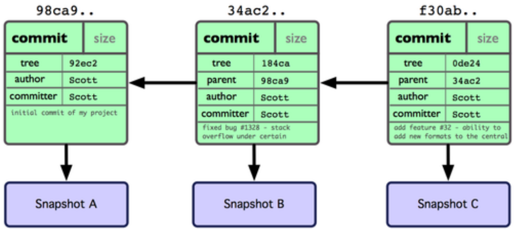
Intermediate Git

Day 3: Branching and Merging

Raman A. Shah

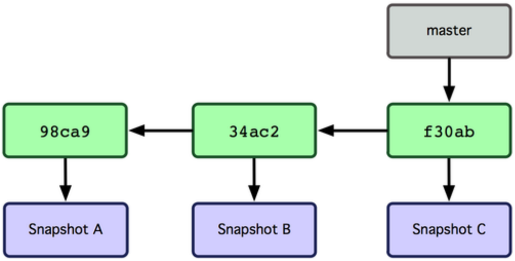
Commits point to commits



Each commit (but the first) has a parent.

Scott Chacon, *Pro Git*, Fig. 3-2. CC-BY-NC-SA.
https://progit.org/

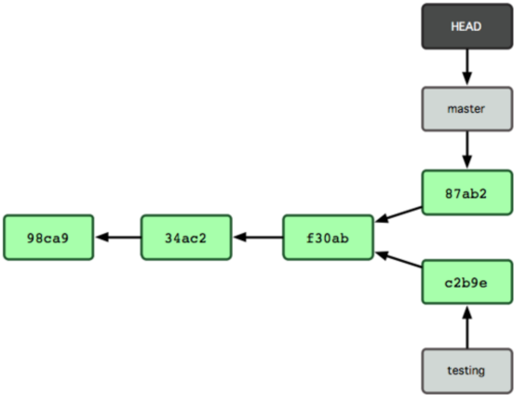
Branches point to commits



A branch is just a pointer to a commit.

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

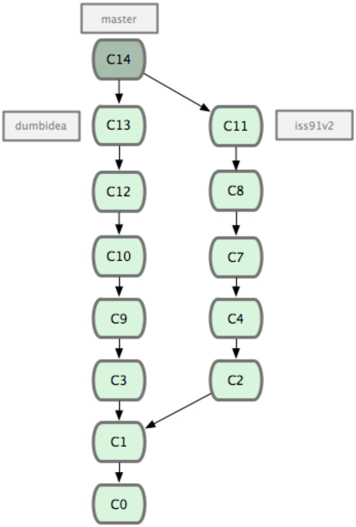
Branches can diverge



Branches enable you to develop different versions of the product concurrently.

Scott Chacon, *Pro Git*, Fig. 3-9. CC-BY-NC-SA.
https://progit.org/

Merging brings them back



Because of merging, you can work on different aspects of a final desired product concurrently.

Scott Chacon, *Pro Git*, Fig. 3-21. CC-BY-NC-SA. https://progit.org/

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/day3/sample.tgz .
tar xzvf sample.tgz
cd sample
git status
```

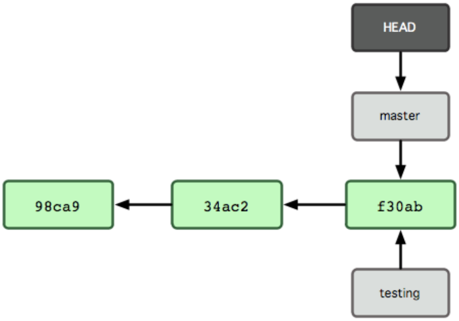
Your first branches

Between each of the following, monitor with:

```
git branch
```

```
git branch foo
git checkout foo
git checkout master
git branch -d foo
```

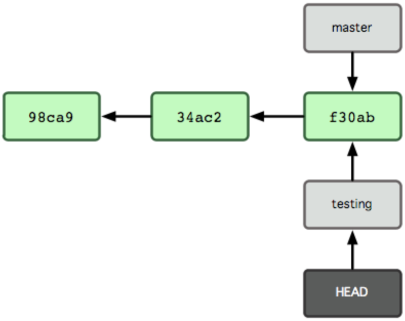
`git checkout` moves HEAD



`git checkout master`

Scott Chacon, *Pro Git*, Fig. 3-5. CC-BY-NC-SA.
https://progit.org/

git checkout moves HEAD



`git checkout testing`

Scott Chacon, *Pro Git*, Fig. 3-6. CC-BY-NC-SA.
https://progit.org/

Hacking on a branch

A shortcut for:

```
git branch experiment
git checkout experiment
```

is:

```
git checkout -b experiment
```

After making some commits, you can throw away the experimental commits with:

```
git checkout master
git branch -D experiment
```

Scenario #1: clean merge

In the push to complete a project, two things will happen concurrently:

- Debug an analysis code
- Flesh out a manuscript

Do work on two branches

Create a branch for debugging the analysis code:

```
git checkout -b fix_math
```

Fix the bug in `R/square.R` using R's `*` operator:

```
...
git commit ...
```

Do the work to flesh out the manuscript on `master`:

```
git checkout master
```

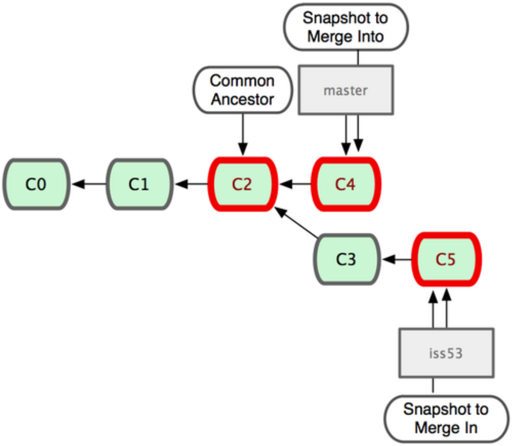
Explain in `paper/paper.tex` that $x^2 = x \times x$.

```
...
git commit ...
```

Branches are lightweight

```
cat .git/refs/heads/master
cat .git/refs/heads/fix_math
```

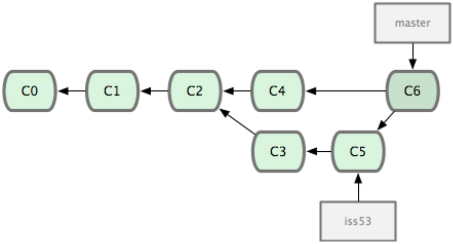
Git plans a three-way merge



Git figures out the best common ancestor on its own. . .

Scott Chacon, *Pro Git*, Fig. 3-16. CC-BY-NC-SA. https://progit.org/

Git plans a three-way merge



. . . and creates a "merge commit" from the three relevant snapshots.

Scott Chacon, *Pro Git*, Fig. 3-17. CC-BY-NC-SA. https://progit.org/

Merge our work

Merge the work *from* `fix_math` *into* `master`, updating `master`:

```
git checkout master
git merge fix_math
```

Behold your successful merge!

```
git log --oneline --graph
```

Scenario #2: merge conflict

Let's pretend that merge never happened:

```
git checkout master
git reset --hard HEAD~1
```

And say that on `master`, we fixed the math bug in a different way using R's `**` operator:

```
...
git commit ...
```

Then what?

```
git merge fix_math
```

Panic button

```
git merge --abort
```

Fixing a merge conflict

```
git merge fix_math
```

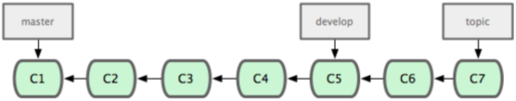
Have a look in the file that didn't merge cleanly. Make it look the way you want it to look. To tell Git that the file is "fixed," you stage it!

```
...
git add R/square.R
```

Then commit:

```
git commit
```

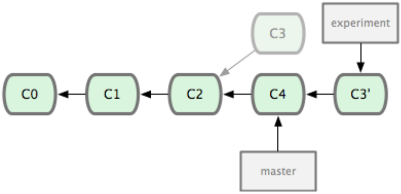
Fast-forward merges



`git checkout master` followed by `git merge develop` just moves a pointer!

Scott Chacon, *Pro Git*, Fig. 3-18. CC-BY-NC-SA.
https://progit.org/

Fast-forward merges



Rebasing can turn a branched history into a linear history.

Scott Chacon, *Pro Git*, Fig. 3-29. CC-BY-NC-SA. https://progit.org/

Rewrite history: git rebase

Never modify history that someone else has seen.

Scenario #3: clean rebase

We'll rewind to just before our merge in Scenario #1: let's get rid of that merge but also the conflicting commit that we'd introduced on `master`:

```
git reset --hard HEAD~2
```

Scenario #3: clean rebase

Replay the work *from* `fix_math` *onto* the tip of `master`, creating a similar (but different) `fix_math`:

```
git checkout fix_math
git rebase master
```

Now the merge is a fast-forward!

```
git checkout master
git merge fix_math
git log --oneline --graph
```

Throw away your successfully merged branch:

```
git branch -d fix_math
```