Mastering git

Lesson 2

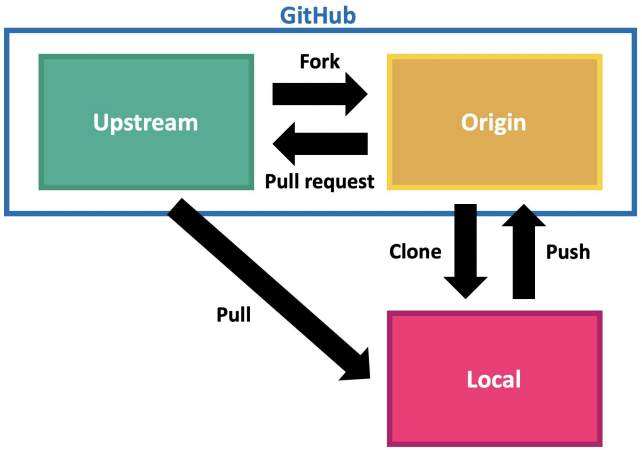
Irina Gulina

Tomas Tomecek

Red Hat

Recap Lab 1

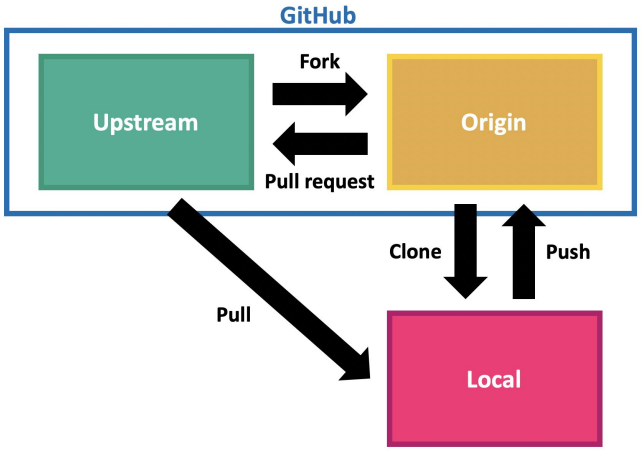
Lab 1: Local => Origin



- Slides 41 and 52 from Lecture #1
  - Create a local repository
  - Make it a version controlled
  - Create a file with some content. What will show `git status`?
  - Notify git about that new file
  - Save your changes in a repository
- Share your local repository publicly
  - Create an origin
  - How to connect your origin with a local? What protocol? Why?
  - Sync origin and local. How?

Lab 1: Origin => Local

- Create an origin repository in UI
- Create a local version of it. How?
- Difference between init and clone



Any questions or suggestions?

We have questions!

Questions

- What is a staging area?
- In what 4 states can a file be when running `git status`?
- What does `git commit` do?

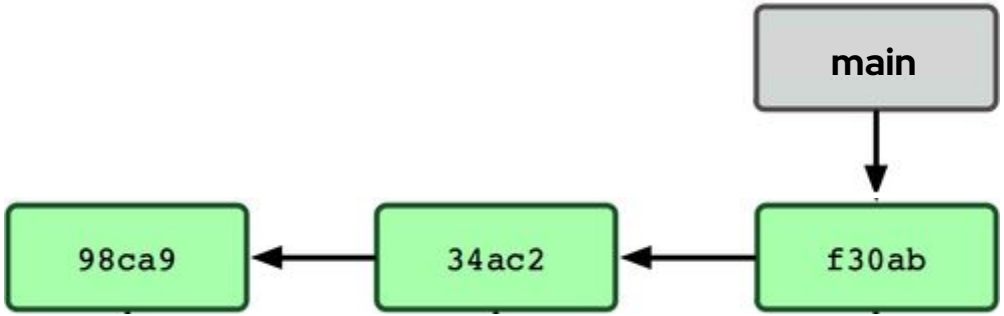
Today’s class

- **How does branching work in git**
- Best practices for branching
- Git tags: how to use them and what’s their use case?
- Stash: put your current work on a shelf and restore it later.

- Labs: branching!!
- Homework 1 assignment

Git Branching

What branch is?



More info: https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

Why do we need branches?

- Work in parallel
- Keep main branch free from questionable code
- Experiment easily

What is the first branch?

```
[Irina@localhost test_git]$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
```

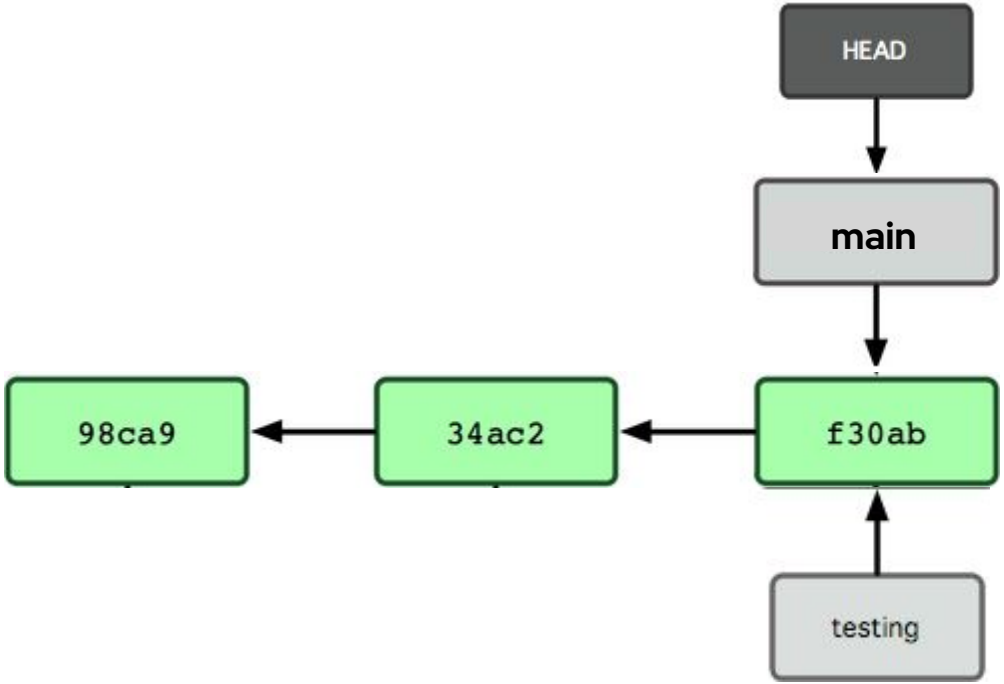
Git version 2.41.0

Branching operations

- Default branch
- Create a new branch
- List branches
- Switch branches
- Work in parallel on different branches
- Merge branches
- Delete a branch
- Rename a branch
- *Stash changes and tags
- Push and pull a branch to a remote server

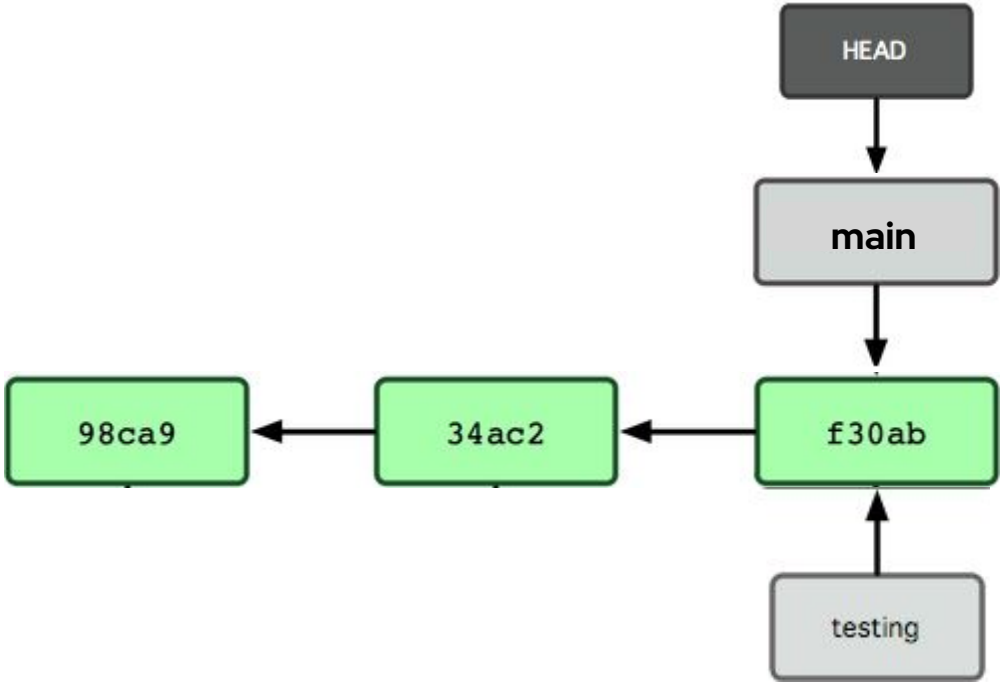
Create a branch

```
$ git branch testing
```



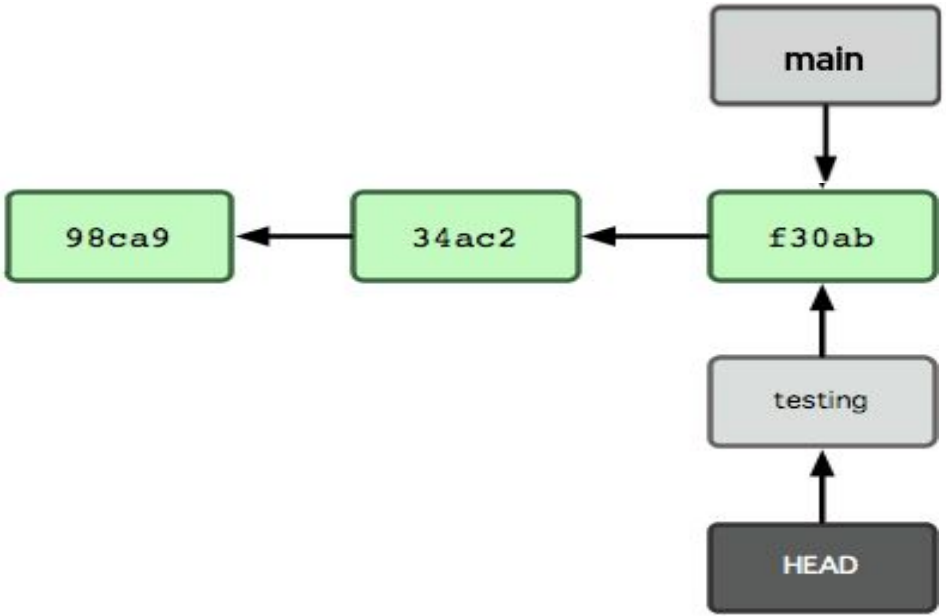
List branches

```
$ git branch <--list>
$ git branch -v
$ git branch -vv
$ git branch -a
```



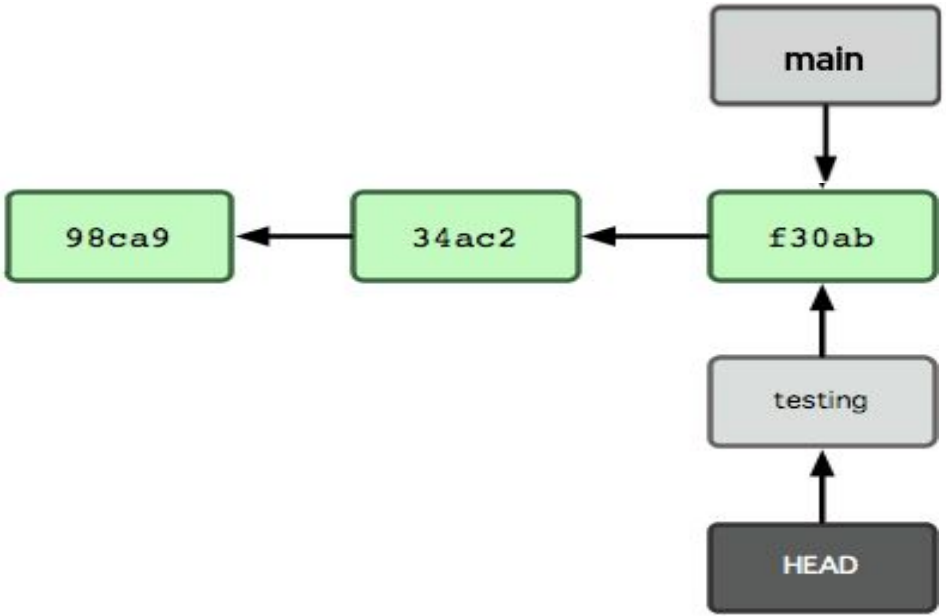
Switch branches

```
$ git switch testing
```



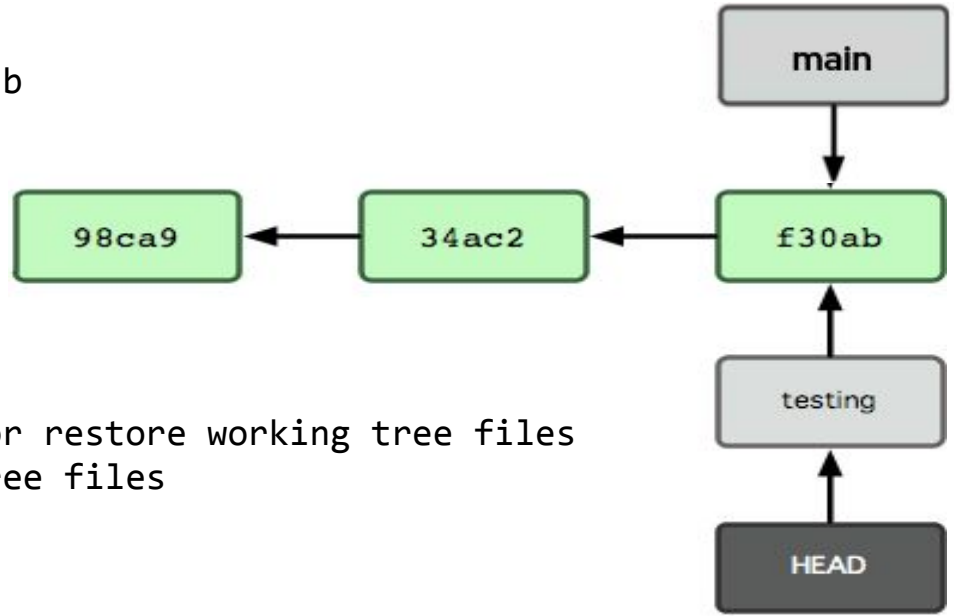
Create and switch

```
$ git switch -c testing
```



Switch vs Checkout

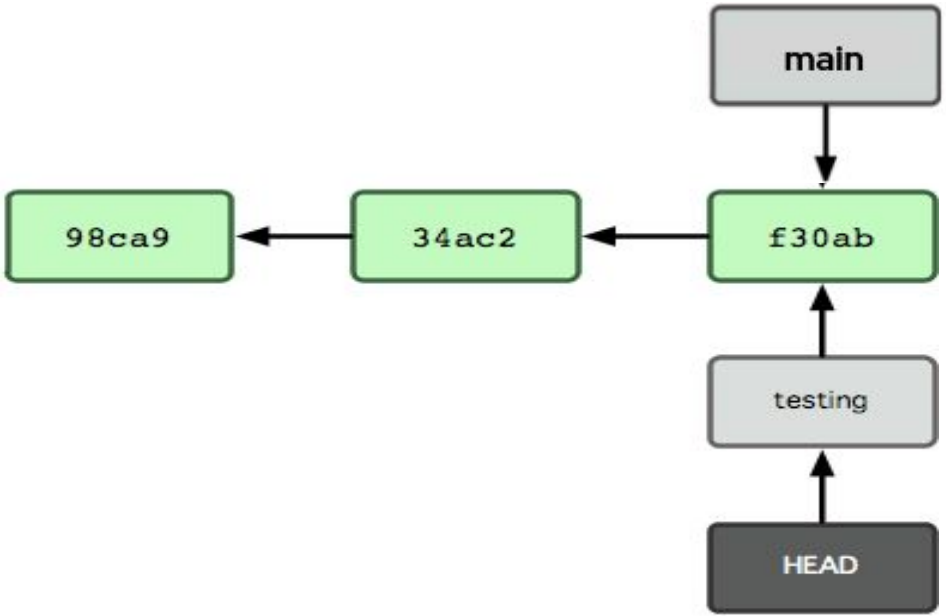
```
$ git switch = git checkout
$ git switch -c = git checkout -b
```




```
git-checkout - Switch branches or restore working tree files
git-restore - Restore working tree files
git-switch - Switch branches
```

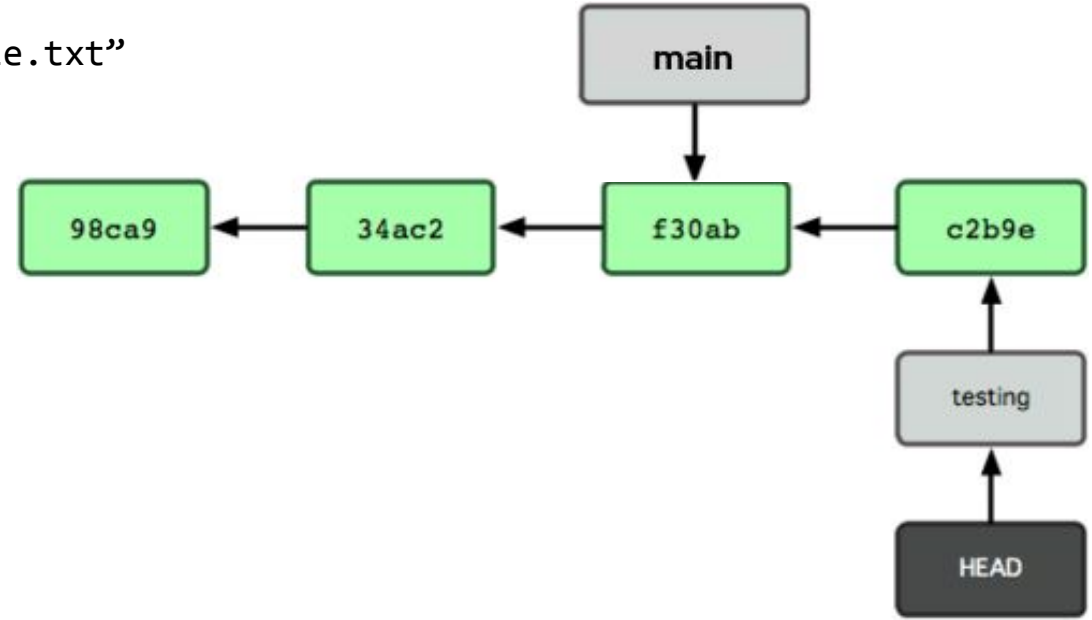
HEAD

```
.git/refs/heads/
.git/HEAD
```



Work in parallel

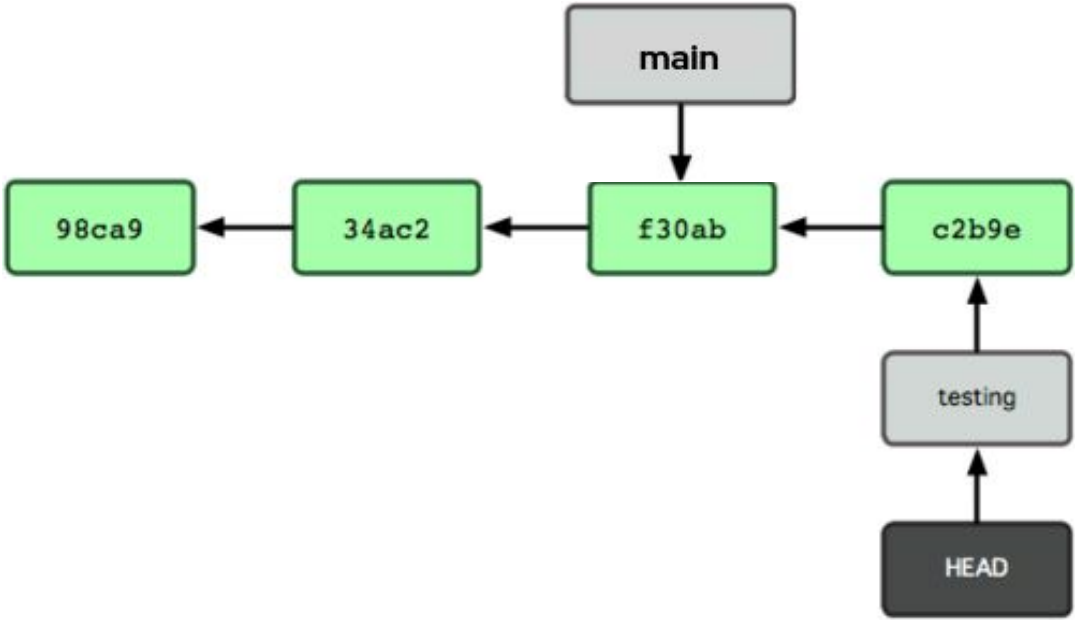
```
$ touch file.txt
$ git commit -a -m “add file.txt”
```



Red Hat

Work in parallel

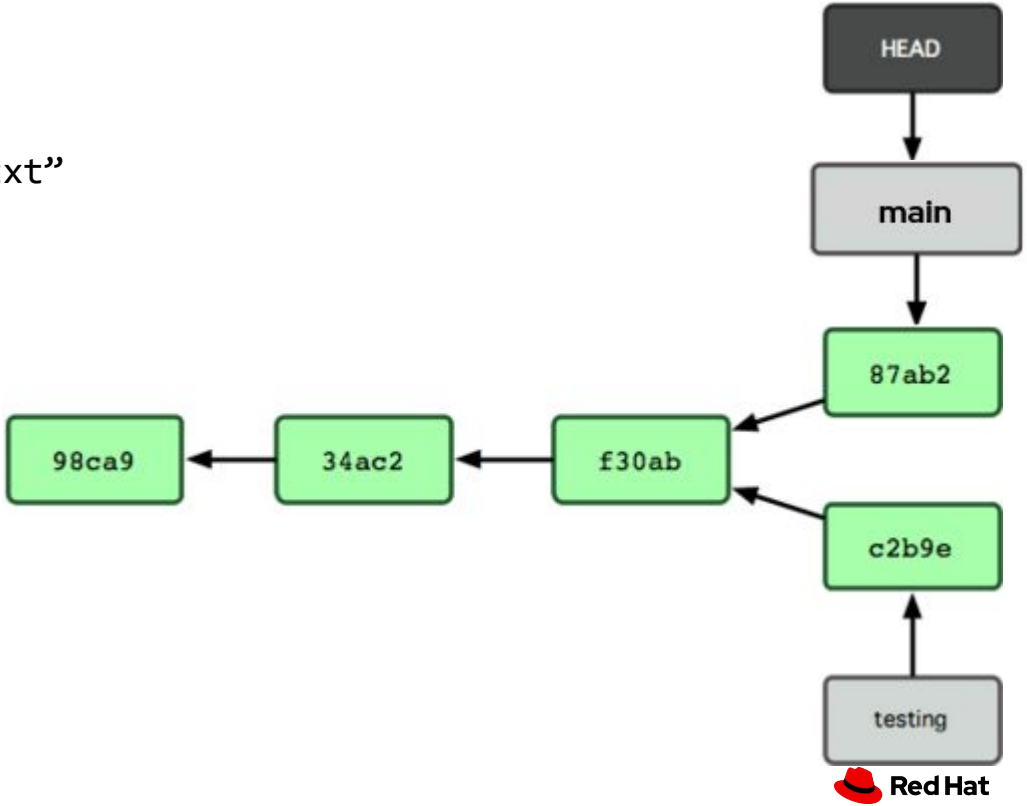
```
$ git switch main
```



Red Hat

Work in parallel

```
$ touch file2.txt
$ git commit -a -m “add file2.txt”
```

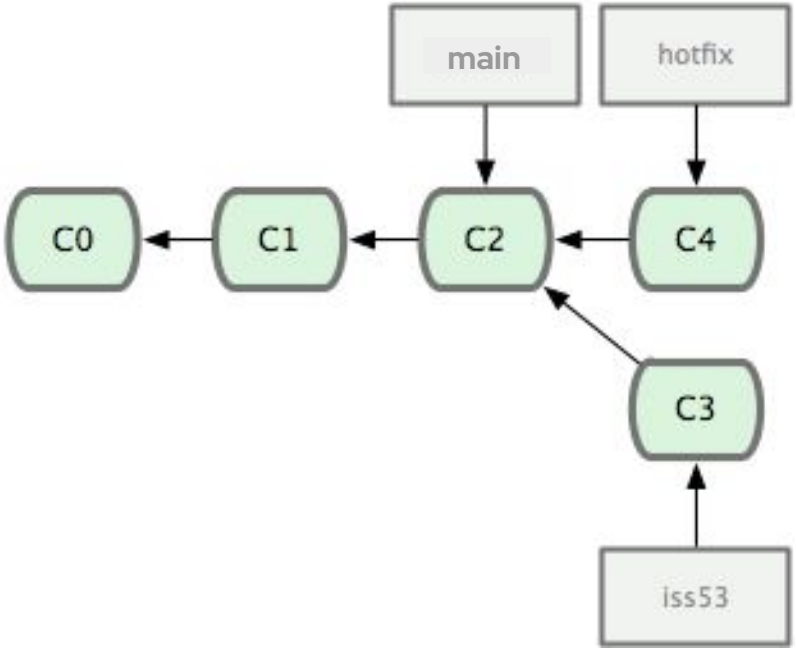


Rename a branch

```
$ git branch -m <new_name>
```

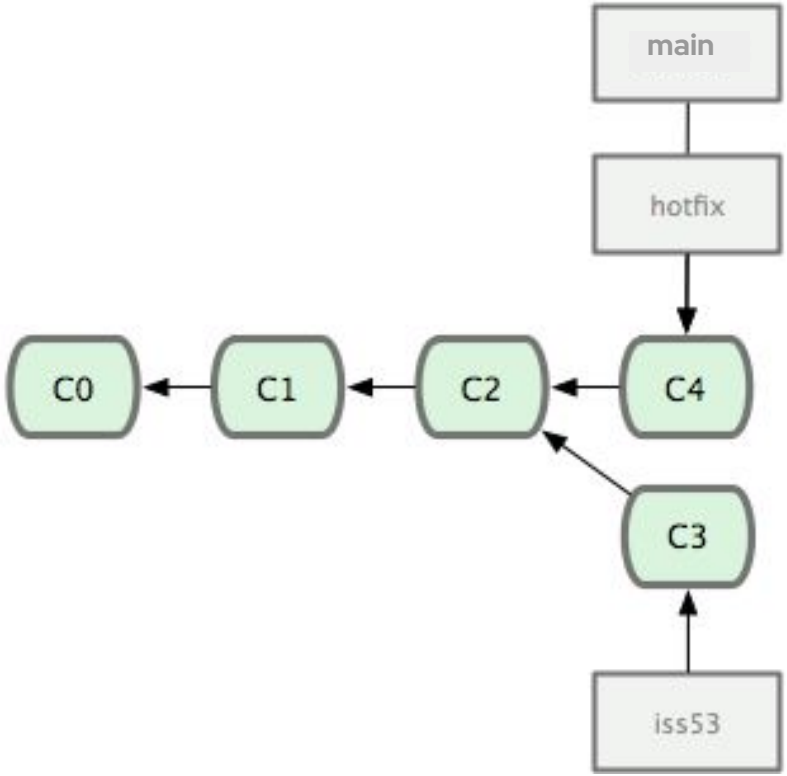
Merge branches

```
$ git switch main
$ git status
$ git fetch
$ git pull
$ git merge hotfix
```



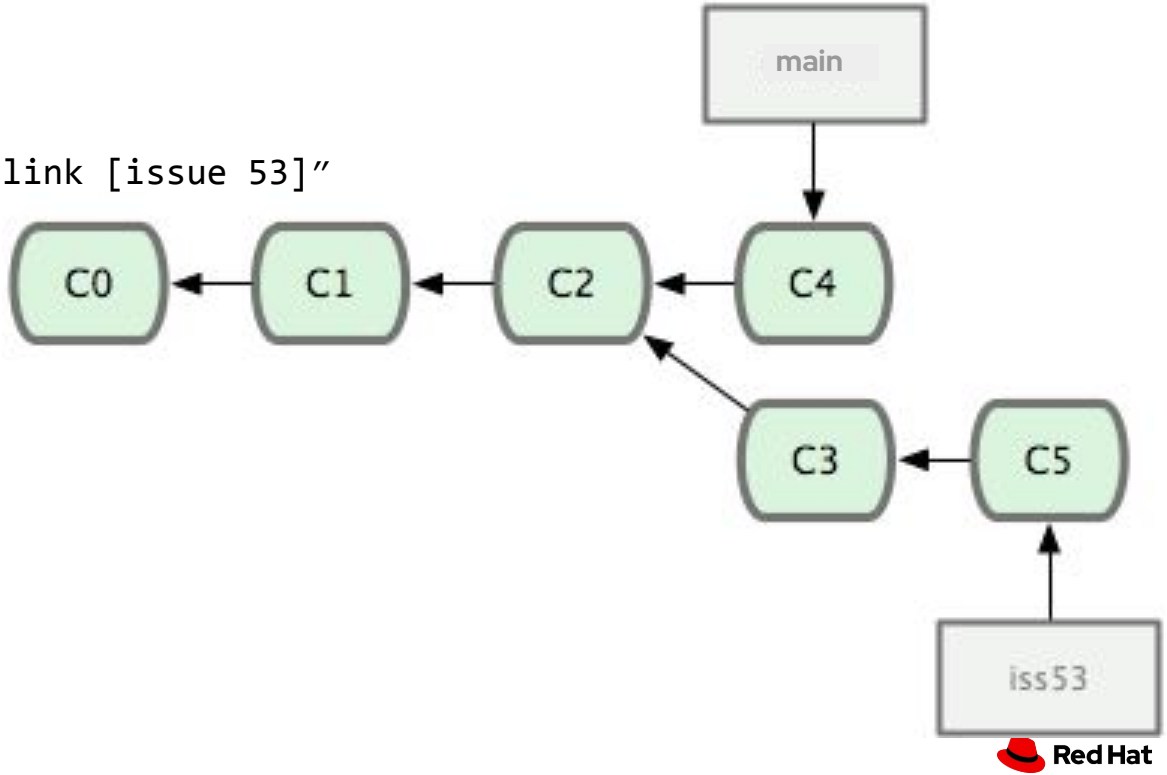
Merge branches (ff)

```
$ git switch main
$ git status
$ git fetch
$ git pull
$ git merge hotfix
```



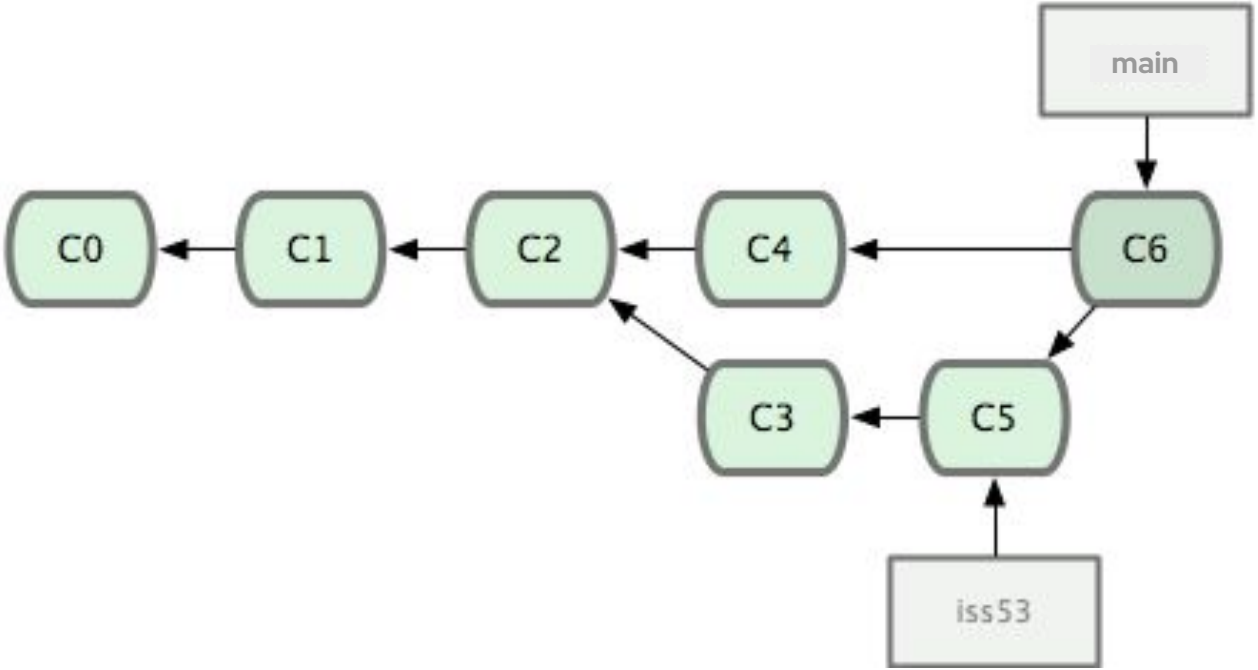
Merge branches

```
$ git switch iss53
$ vi index.html
$ git commit a -m “fix link [issue 53]”
```

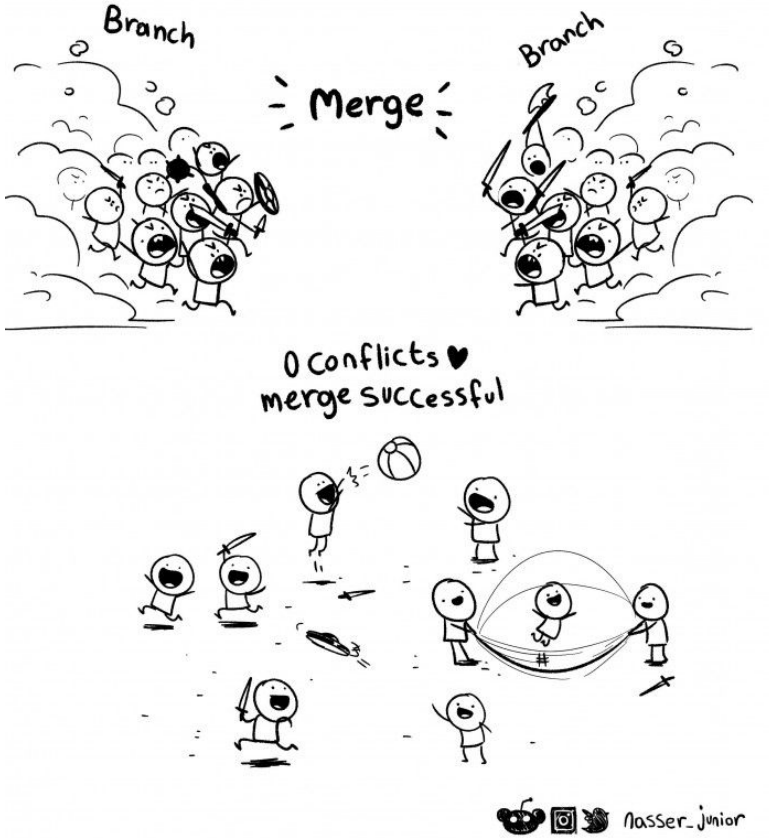


Merge branches (merge commit)

```
$ git switch main
$ git status
$ git merge iss53
```



Merge branches



Red Hat

Delete a branch

```
$ git branch -d testing
$ git branch -D testing
```

```
$ git push origin --delete testing
$ git push origin :testing
```

Stash

git stash

A special “branch”

git has high-level operations to work with it

Handy to put things on side

git stash (commands)

git stash list

git stash show

git stash [push]

git stash apply vs. git stash pop

stash@{2}

git log stash

Tags

git tag

Points to a commit and doesn’t change as you commit more

Used mainly to track releases and deployments

Lightweight vs. Annotated

Push/pull tags

Test your knowledge now!

Task

- Fork this repo and clone it: https://gitlab.com/redhat/research/mastering-git/
- Create a branch
- Switch to that branch
- List branches
- Create another branch
- Switch to it and create a commit
- Switch to the previous branch and merge the last branch into it
- Delete the previous branch
- Special task: create a new branch and merge it into the first branch in a way so it's not fast-forward: there is a merge commit

Questions?

Class 2 homework

https://gitlab.com/redhat/research/mastering-git#class-2-homework

All homework info will be in README.md

Bonus Task

Make a contribution to an Open Source project.

- Not an University Project
- Not owned by you
- MR/PR doesn't need to be merged by the Task/Course deadline
- A change can be of any content (not necessary code, it can be docs for example), but it must be meaningful, positive
- See details in "Mastering git" Readme

First Timers Only
10 C++ open source projects welcoming contributions
Contributions-welcome topics on GitHub
Hacktoberfest - 10th anniversary. Check on participation info

THANK YOU!