Git Training

Rachel Player
Jordy Gennissen
Nathan Rutherford
+ you!

Royal Holloway University of London
9 April 2021

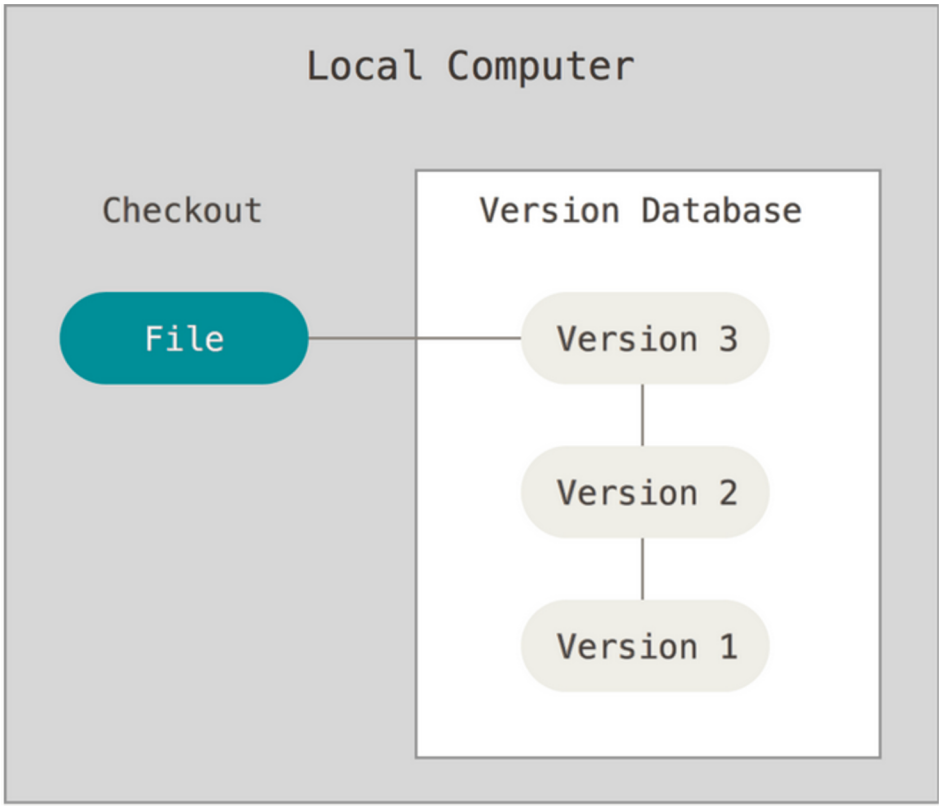
Motivation

- Imagine a bunch of researchers want to write a paper together
  - They are sitting in different offices (all over the world)
  - They want to work together on one/more documents
  - They want to edit the documents at the same time
  - Under pressure, mistakes happen
    - Also when not under pressure
- Solution: Office 365?
  - Disagreements fought over the editor
  - Who wrote that monstrosity / brilliant quote?
  - And who deleted my section, and why?

Git: The cloud before it was cool

About Version Control (1)

- Retain a database of all previous versions
- All changes have a record of who did what



About Version Control (2)

Free backups for everyone!

Disclaimer

This workshop does not tell you:

- Good coding collaborative practice
- How to be a “git master”

But does tell you:

- How to use git well when collaborating on a paper
- How to use it **practically**

Terminal Interrupt

~$

How do we navigate in a black box?

- Black box is called a shell/terminal
  - Open it now!
- You're always in a directory (folder)
  - Which one? Type `pwd` [enter]
- **C**hanging **D**irectory: `cd [path]`
  - Reset directory to your home: `cd` without a path

More terminal

- Show what files are in the directory
  - **ls** (stands for list)
- Make a new folder / directory
  - **mkdir** (make directory)
- Edit a file
  - nano [filename]
- From last slide
  - cd; cd [path]; pwd

Change Directory

- Need to specify a path
- Usually done as a *relative* path

```
ISG-CDT-00310:test VDAI002$ mkdir new
ISG-CDT-00310:test VDAI002$ ls
new
ISG-CDT-00310:test VDAI002$
ISG-CDT-00310:test VDAI002$ cd new
ISG-CDT-00310:new VDAI002$ nano notes.txt
ISG-CDT-00310:new VDAI002$ ls
notes.txt
ISG-CDT-00310:new VDAI002$ pwd
/tmp/test/new
ISG-CDT-00310:new VDAI002$ cd ..
ISG-CDT-00310:test VDAI002$ pwd
/tmp/test
ISG-CDT-00310:test VDAI002$
```

Installing git

- Linux:

```
$ sudo apt install git
```

- Mac OS:

Download from: https://git-scm.com/download/mac
Or use brew: brew install git

- Windows:

https://git-scm.com/download/windows

First-Time Git Setup

- Configure name

```
$ git config --global user.name “John Doe”
```

- Configure email address

```
$ git config --global user.email “johndoe@example.com”
```

- Configure editor for commit messages

```
$ git config --global core.editor nano
```

- Check settings

```
$ git config --list
user.name=John Doe
user.email=johndoe@example.com
...
```

Basic Commands

git clone

- Get a copy of an existing remote repository on your local machine
- The typical way to start any project

git status

- Lists the files which have been modified since the last commit
- Lists the untracked files in your local directory

git add

- Puts a file in the ‘staging area’ ready for a commit
- You can add several files ready for one commit

```
$ git add test.txt
```

Adds a new file called test.txt to the staging area (which can then be uploaded to the server)

git commit

- Commits the files in the staging area (that have been added with the previous command)
- Add a meaningful commit message so you/other people understand the change
- Commits are labelled by a hash value (SHA-1)

```
$ git commit –m “refer to [XYZ17] in introduction”
```

This means ‘Commit the file[s] that have been added to the local repository, with the message given after the symbol -m’.

git push

- Upload the committed local changes to the remote repository

```
$ git push
```

git pull

- Download the latest remote change to the local repository

```
$ git pull
```

git log

- Shows the history of commits (author/date/commit message)

```
$ git log
```

or

```
$ git log --graph
```

Exercise ☺

Exercises

- Setup
  - Create Github account (FYI: you already should've done this)
  - Tell us your username (FYI: you already should've done this)
  - Start Exercises (FYI: This is not something you should've done already)

Breakout rooms

Room 1: Dan, Stephanie, Sofia

Room 2: Kyra, James, Emma

Room 3: Giuseppe, Elle

Exercise 0

- Type ‘cd’ to be in the home directory
- Make a new directory called git
- Make a subdirectory called test
- Make a file called test.txt
- Change directory to the git directory ready for the next exercises

Exercise 1

- Create a new repository on Github
- Create a file named “test.txt” inside the repo
- Write your name in the text document
- Upload the text file to the repository

Exercise 2

- Clone the following repository:
  - https://github.com/rachelplayer/isg-playground.git
- Create a file “<your_firstname>.txt”
- Upload your file to the repository
- Download the files of the other people

Collaborating 2.0

- What if we edit the same file?
- git will automatically try to understand how to merge two updates
- If git doesn't know how: you get a ***merge conflict*** and will need to resolve it manually

When all goes wrong

- Two commands to reset everything

  ```
  git reset --hard
  ```

  Reset all git files to the latest commit

- Also reset the files that git doesn’t track:

  ```
  git clean --fd
  ```

Exercise 3

- Use the repository from previous exercise
- Write your name in the text document “names.txt”
- Upload the changes in names.txt

- Overall goal: Everyone’s name should be in the file names.txt

(slightly)

Advanced Commands

Git IDs

- Every git commit has a unique ID
- If you want to go back to a commit, use the ID!
- To find the ID, use the website or

```
$ git log
```

- Git log example:

```
commit 0cb46a492bd91e0b4389dfeacd83ed2701701222
Author: Rachel Player <rachelplayer@gmail.com>
Date:   Fri Jan 18 15:06:41 2019 +0000

    added the file rachel.txt
```

git checkout

- Revert a file to a version of the file from a previous commit

```
$ git checkout test.txt
```

This restores the file test.txt to the last uploaded version

```
$ git checkout 397344c2 test.txt
```

This restores the file test.txt to the version with commit id 397344c2

git diff

- Shows the differences between your version and the latest commit

```
$ git diff
```

.gitignore

- One can create a file and list all files that should be ignored by git

- For example, intermediate files from LaTeX including:
  *.bbl
  *.blg
  *.aux
  *.out
  *.log

git mv

- Move/Rename a file

```
$ git mv test.txt introduction.txt
```

This renames the file test.txt to introduction.txt

git rm

- Deletes a file from the git repository
  - If you delete the local file, but don’t commit the deletion, it still exists in the repo
  - To delete it in the repo, use

```
$ git rm test.txt
```

  - Note that you can still recover the file if necessary, even after deleting!

Forgot to pull?
And already made changes?

- Commit and merge, or:

```
$ git stash                 // save for later
$ git pull                  // get the latest version
$ git stash pop             // retrieve your changes
```

Exercise ☺

Exercise 4

- Go back to your own repo
- Add 2 files: “test2.txt, oops.txt”.
- Commit
- Revert the changes using *only git*
- Commit
- Recover these files using *only git*
- Remove oops.txt again

Pairs for Exercise 5

Pair 1: Sofia & Kyra
Pair 2: Dan & Elle
Pair 3: James & Giuseppe
Pair 4: Stephanie & Emma

(Each person is in a separate breakout room to the other person in their pair)

Exercise 5

- In pairs, invite someone to join your repo
- Add a file “review.txt” and push
- Both collaboratively write a review about this workshop. Push regularly, and resolve conflicts.
  Hint:
  - Git diff
  - Git log (also verify their useful commit messages!)

Exercise 6

- Copy the review into the shared repo
  - isg-playground
- add it, commit and push!

Useful Stuff for Paper Writing

Github / Gitlab / Bitbucket

- Web-based git/version control repositories
- Distributed version control
- Source code management
- Millions of users
- Offers public and private repositories
- Free repositories (with an academic email address) on all three

CryptoBib

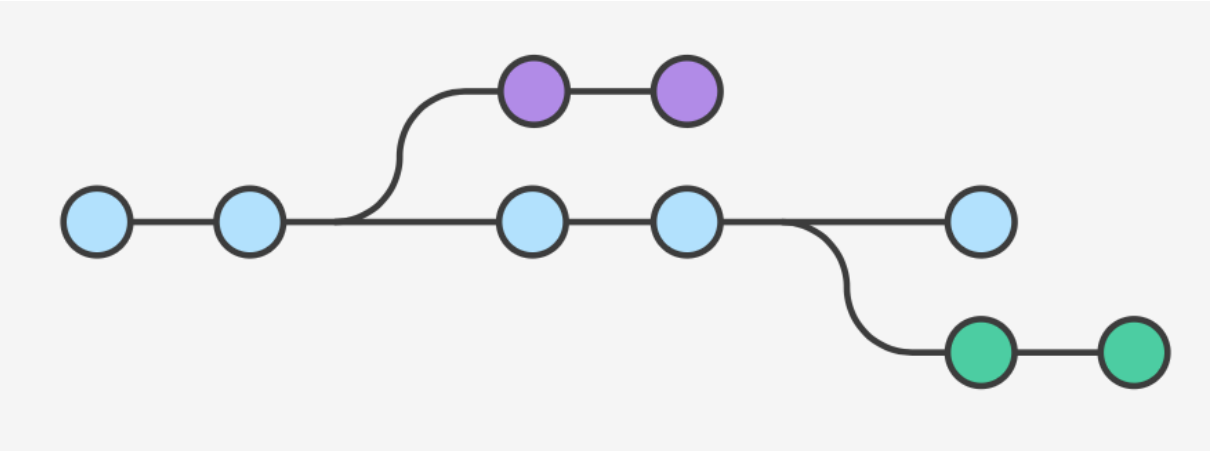
CryptoBib is a BibTeX database containing papers related to Cryptography, with manually checked entries and uniform BibTeX data.

https://cryptobib.di.ens.fr

Note from fire safety: don’t actually do this

Advanced Commands

Git Branches



- A branch represents a independent line of development
- There are local and remote branches

Git Branches

- List all branches in your repository:

```
$ git branch
```

- Create a new branch:

```
$ git branch <branch>
```

- Delete a branch:

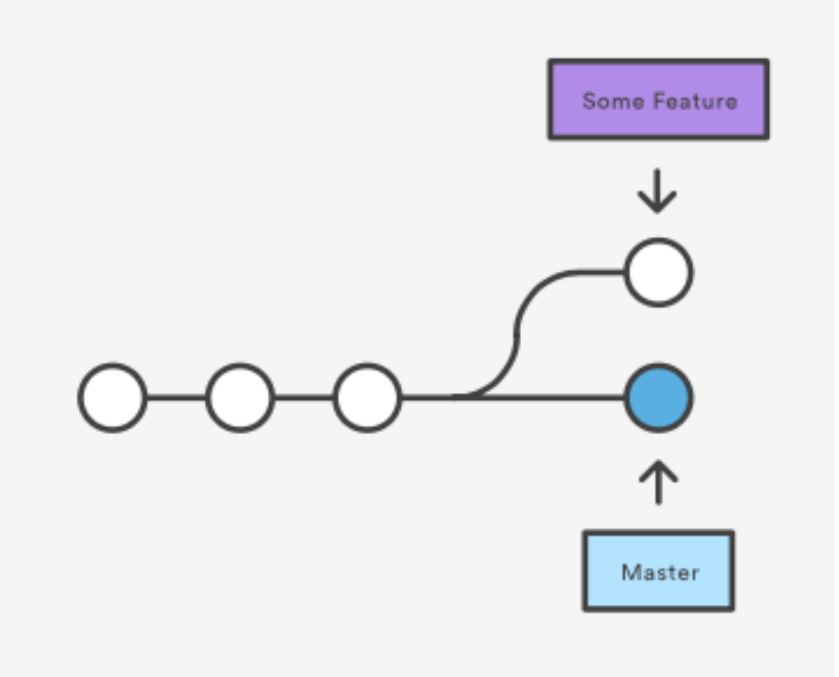
```
$ git branch -d <branch>
```

- Switch to /checkout a branch:

```
$ git checkout <branch>
```

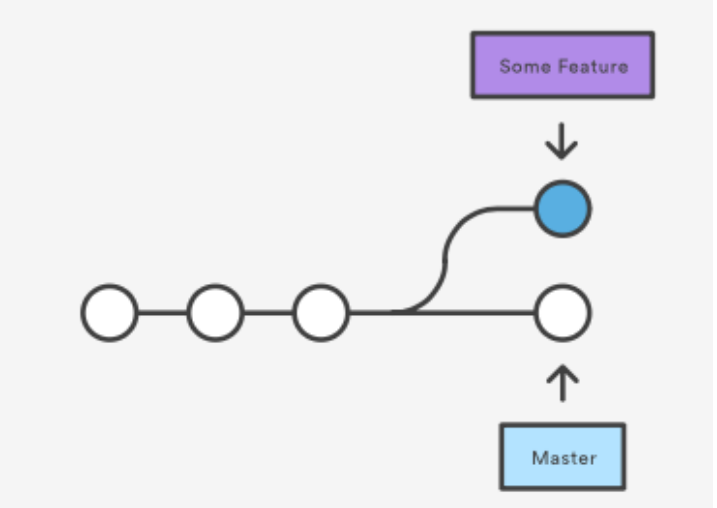
Git Branches - Example


```
$ git branch <some feature>
```

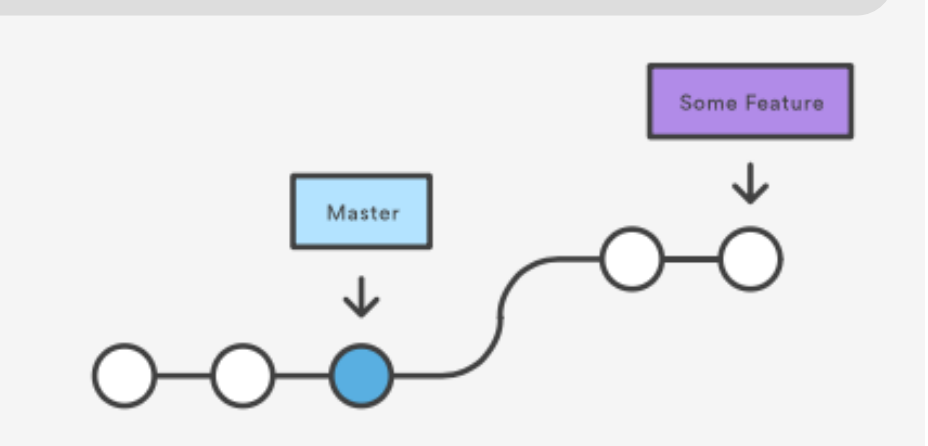


Git Branches - Example

```
$ git checkout <some feature>
```




```
$ touch test.txt
$ git add test.txt
$ git commit test.txt –m “add test.txt”
```



Git Branches - Merge

- Merge branch back to current branch:

```
$ git merge <branch>
```

- Merge branch (but always create a merge commit):

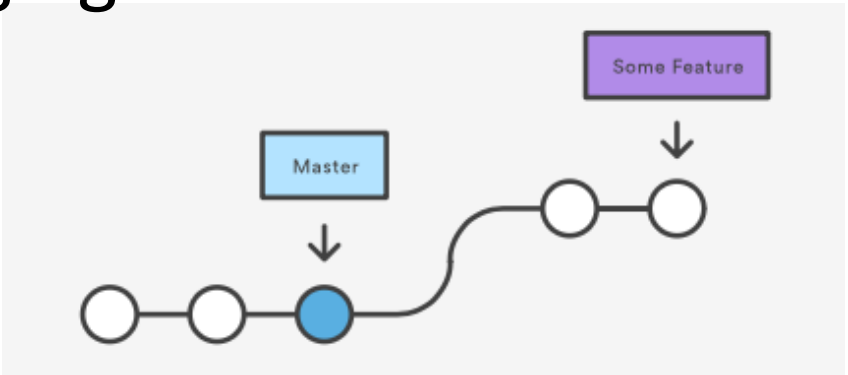
```
$ git merge --no-ff <branch>
```

- Several types of possible merges
  - Fast-forward merge
  - 3-way merge

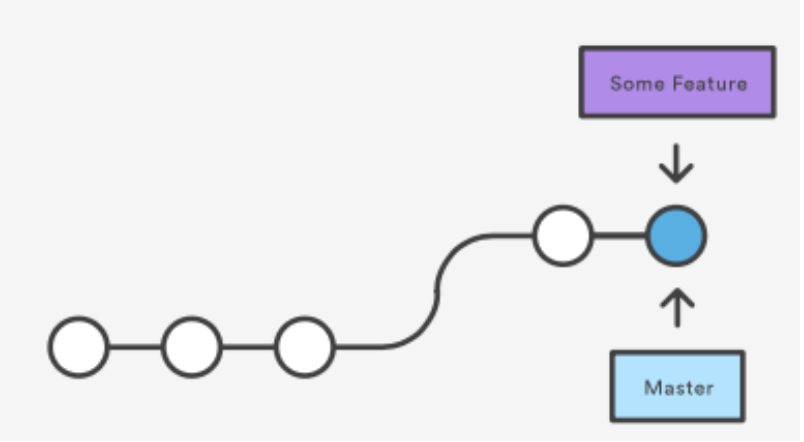
Git Branches – Fast-Forward Merge

```
$ git checkout master
$ git merge <some feature>
```

- Before merging:



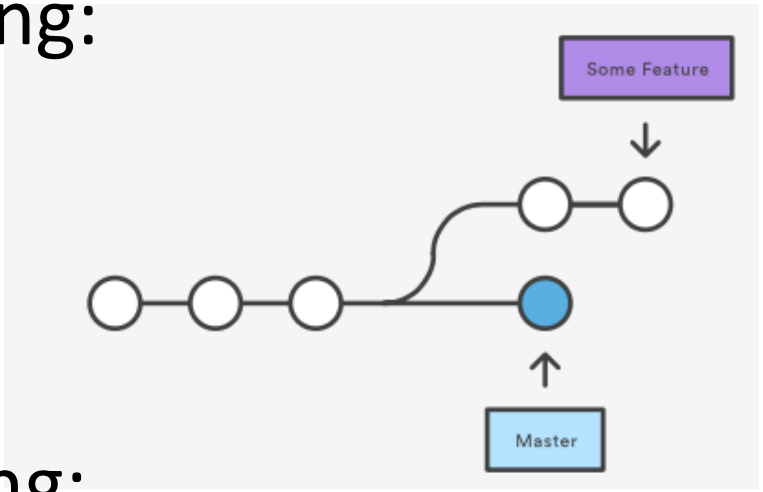
- After merging:



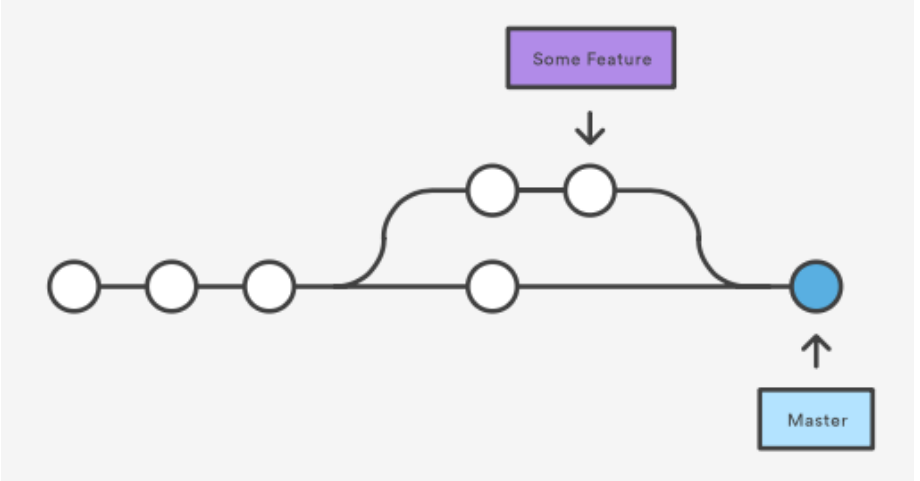
Git Branches – 3-way Merge

```
$ git checkout master
$ git merge <some feature>
```

- Before merging:



- After merging:



Git Branches – Merge conflicts

- If two branches change the same part of the same file, git can't handle the conflict

```
# On branch master
# Unmerged paths:
# (use "git add/rm ..." as appropriate to mark resolution)
#
# both modified: hello.py
#
```

- Resolve conflict manually
- Commit resolved conflict

Git Branches – Remote branches

- Publish/Push a local branch:

```
$ git push origin <branch>
```

- Pull a remote branch:

```
$ git checkout –b <localbranch>
origin/<remotebranch>
```

- List all branches (local and remote):

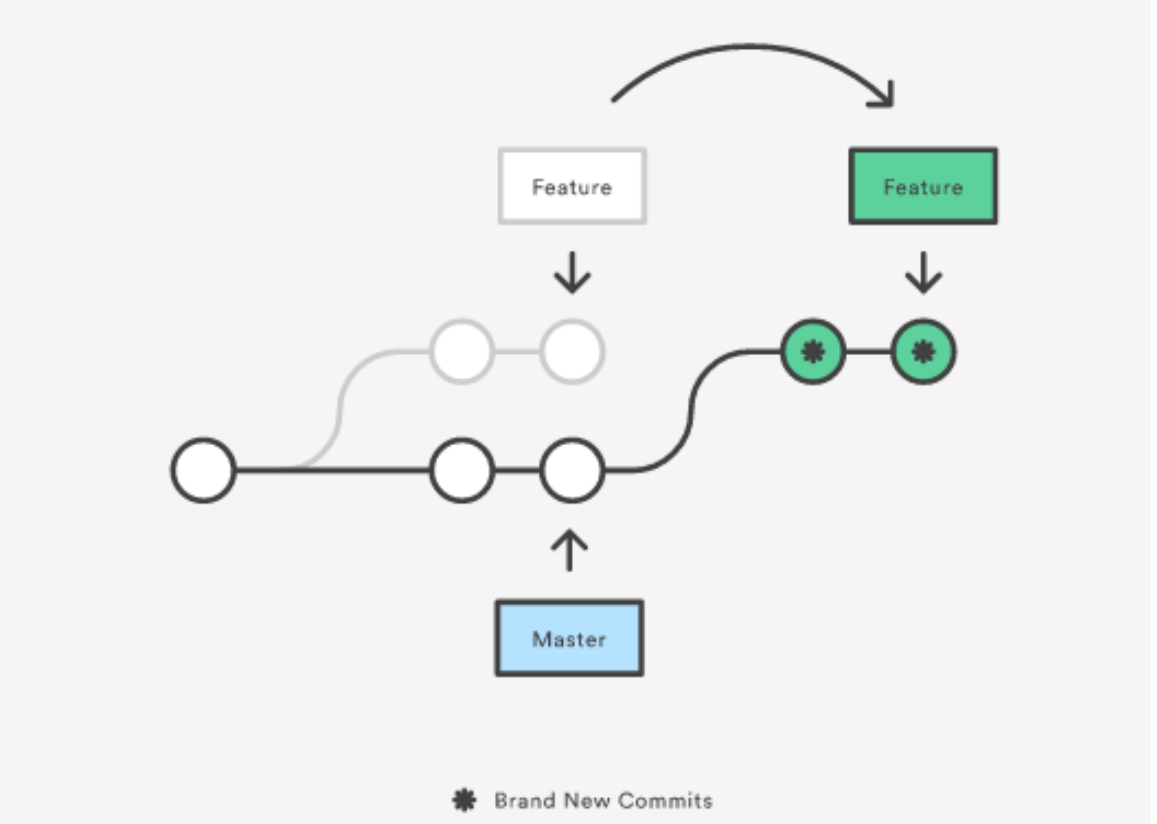
```
$ git branch -a
```

- Delete remote branch

```
$ git push origin --delete <remotebranch>
```

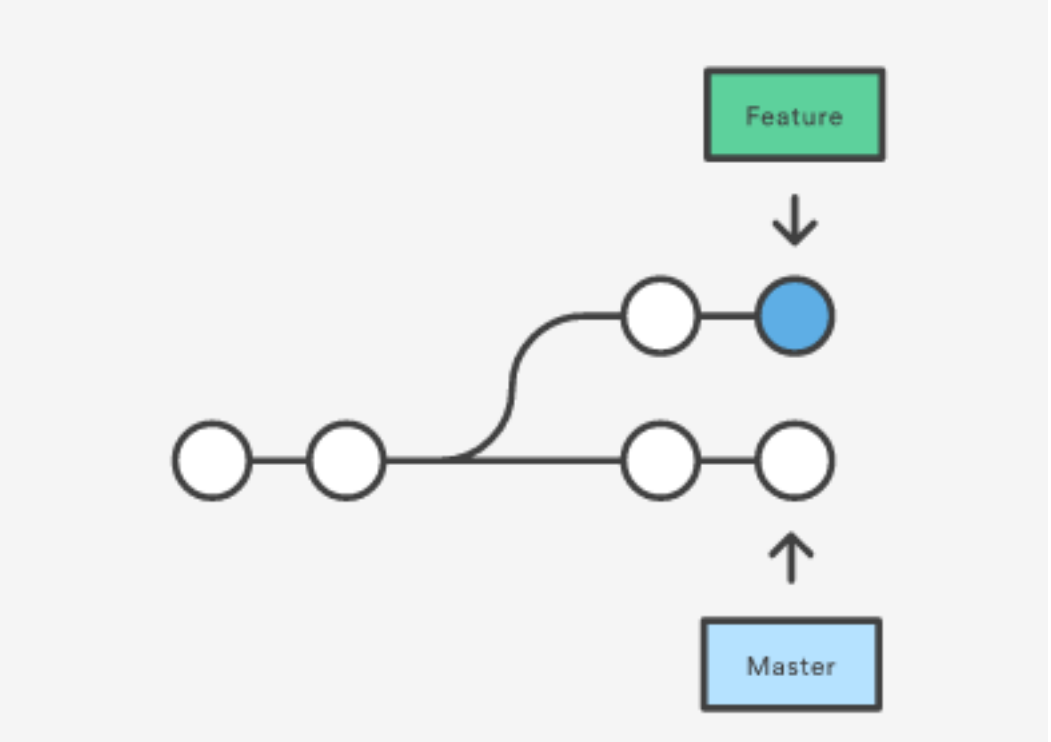
Git Branches – Rebasing

- Move a branch to a new base commit
- Maintain linear project history
- Don’t loose history from a branch



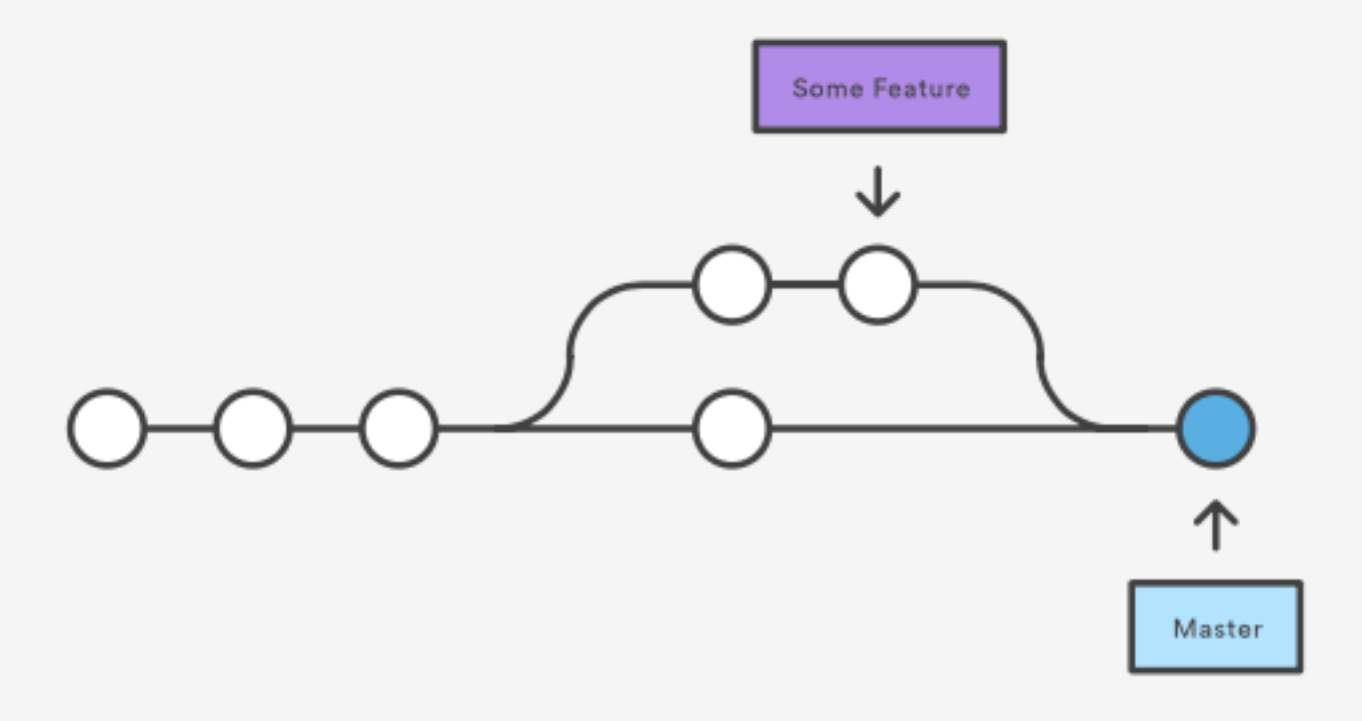
Git Branches – Rebasing

- Master branch has progressed since the start of a feature
- The feature depends on some commits of the master branch



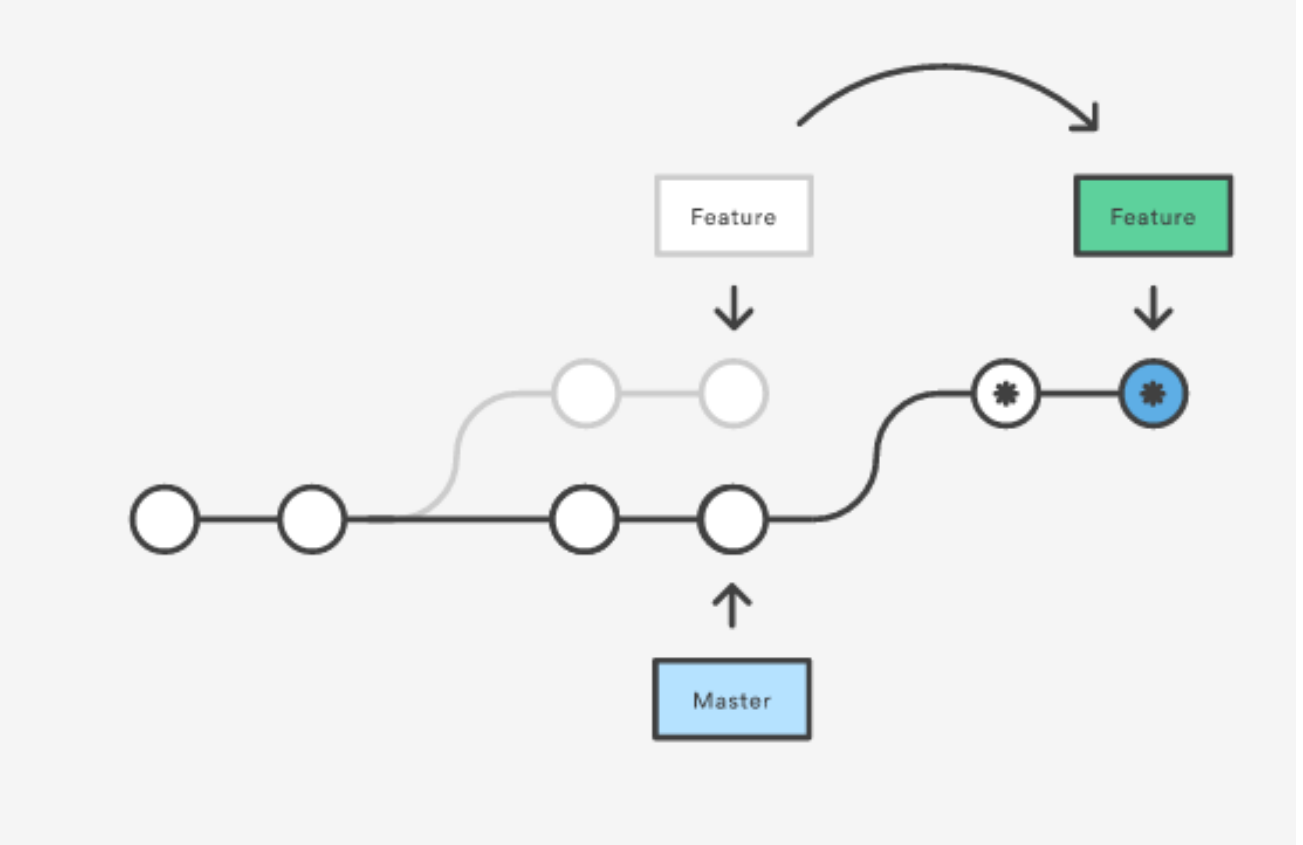
Git Branches – Rebasing

- Solution 1: Merge directly with a 3-way merge and a merge commit



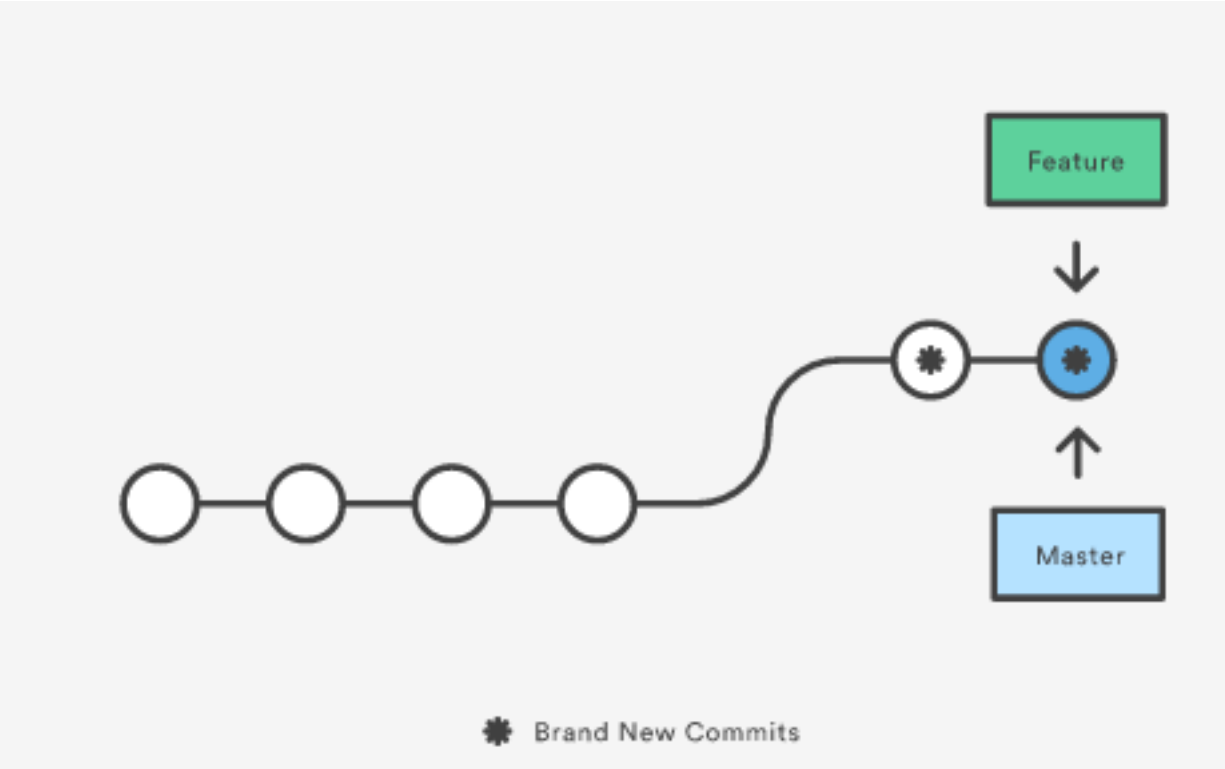
Git Branches – Rebasing

- Solution 2: Rebase



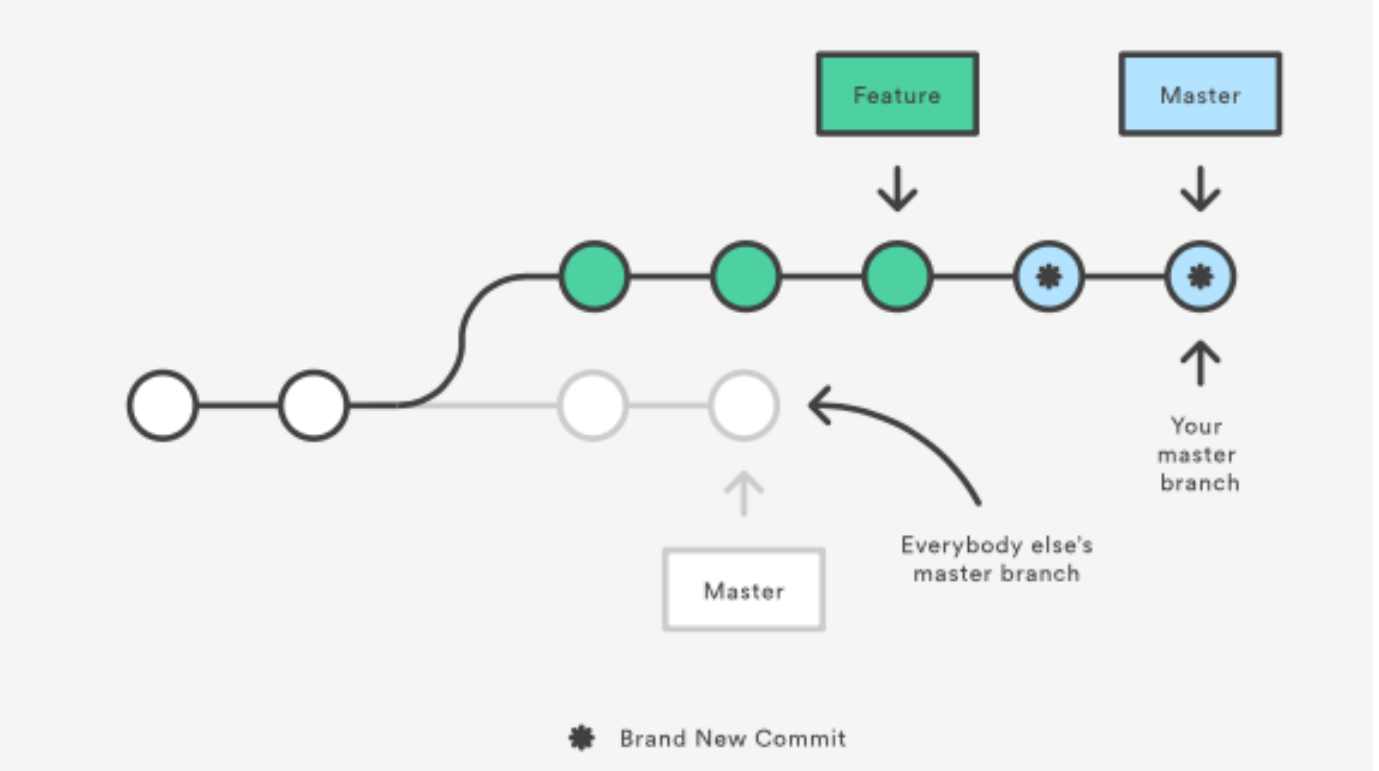
Git Branches – Rebasing

- Solution 2: fast-forward merge

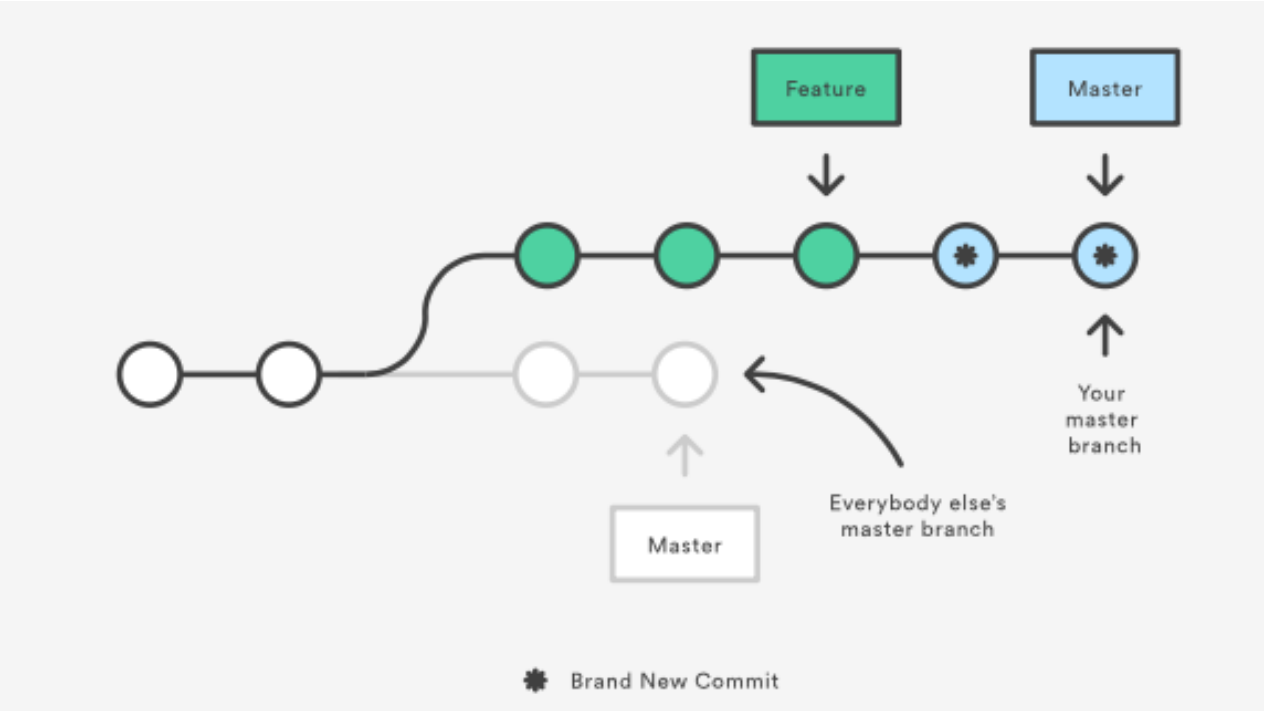


Git Branches – Rebasing

- Golden Rule of Rebasing: Don’t rebase public branches
  - Example: Rebase the master branch onto your feature branch



Git Branches – Rebasing



- This **only** happens in **your** repository
- Everyone else will work on the old master
- Rebase creates new commits – git thinks that your master branches diverge from the other master
- Merging them together will results in a merge commit with two different histories

Git Submodules

- Use other git repository in your git repository
- Use external libraries managed in a git repository
- Create a new submodule:

```
$ git submodule add <link to repository> <directory>
```

- Clone a git repository with submodules:

```
$ git clone -- recursive <link to repository>
```

- Update a submodule:

```
$ git submodule update --init
```

Exercise ☺

Exercises 7

- Go back to your own repo
- Create a branch with your name
- Edit the file ”test.txt” in your branch and write your name in it
- Upload your branch to the repository
- Checkout the master branch again

Exercise 8

Merge your branch to the master branch

Add CryptoBib as a submodule

Further Tutorials

- https://git-scm.com/book/en/v2
- https://www.atlassian.com/git/tutorials
- https://www.git-tower.com/blog/git-cheat-sheet/