Git branching

Matthieu Haefele

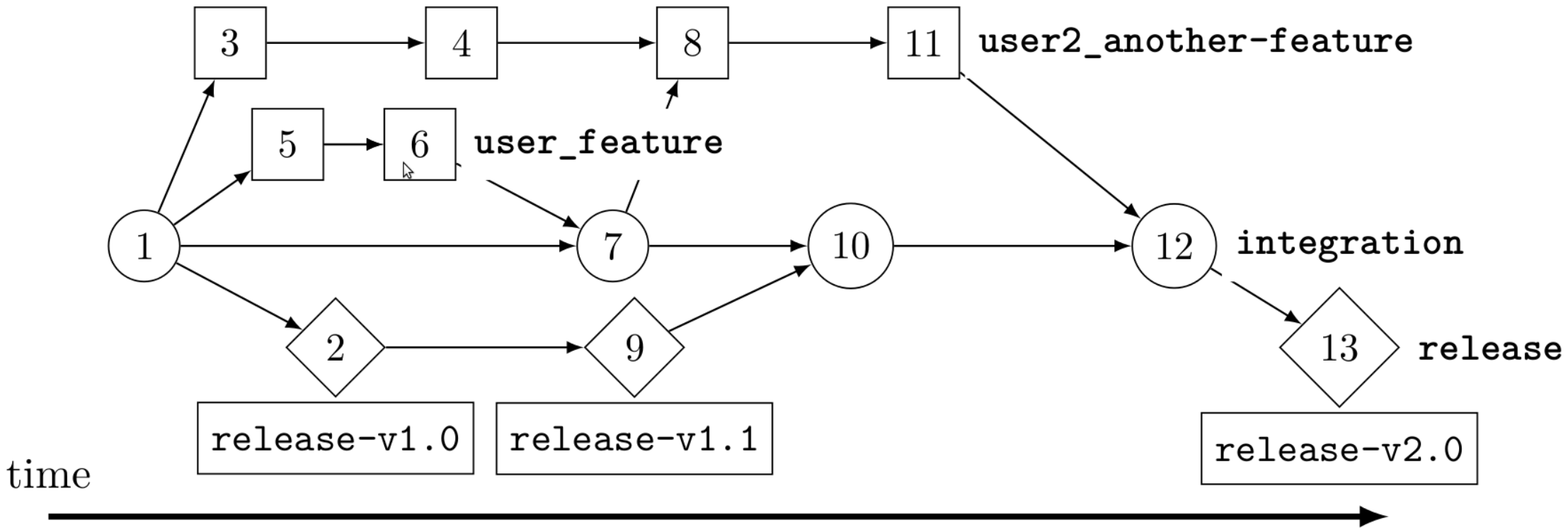
Pau, October 2023

Git internals

http://eagain.net/articles/git-for-computer-scientists/

Branching merging demo !

HPCSSS Gitflow



HPCSSS Gitflow

HPCSSS: High Performance Computing for Scientific Simulation Software

- Circle, square, diamond: commits
- Numbers: commits' ordering in time
- Arrow: starting point is the parent, arrival is the child
- Text in bold: branches with commits that were made in this branch in the same horizontal line
- Rectangles at the bottom: tags

Merge or Rebase, that is the question

Merge: the historian view

- The commit history of the project should not be changed
- Repository should contain what actually happened

Rebase: the project maintainer view

- The commit history is the story of how the project was made
- You would not publish the first draft of a book
- Telling a nice story is easier to read for future collaborators

What about binary files ?

- git not really helpful with binary files
- but git can store them for small ones
- `git annex` should be prefered for large files (or `git lfs` at least)

References

- https://git-scm.com/book/fr/v2
- http://git-scm.com/doc
- http://nvie.com/posts/a-successful-git-branching-model/

Hands-on: Shopping list

- Go on the shopping list repository at UPPA or the one at INRIA
- **Fork the repository on gitlab (one per group) and clone it on your laptop and follow instructions.**
- Once done, go ahead on git branching serious game

My god, a conflict !

1. Etienne and Markus clone their repository
2. Etienne modifies the value of Pi line 33
   Pi=3.1 $\Rightarrow$ Pi=3.14
3. Etienne commits
4. Etienne pushes
5. Markus modifies the value of Pi line 33
   Pi=3.1 $\Rightarrow$ Pi=3.1415
6. Markus commits
7. Markus pushes $\Rightarrow$
8. Markus pulls $\Rightarrow$

Now in Markus' file

```
<<<<<<< HEAD
Pi = 3.1415;
=======
Pi = 3.14;
>>>>>>> f6f2a6c975df9a06f353e6640997ca39c6d071e3
```

- Between the $<<<$ and the $===$ Markus's local version
- Between the $===$ and the $>>>$ Etienne's remote version
- Etienne and Markus have to agree on the value of Pi

How to fix a conflict

- Replace everything between the <<< and the >>> by the correct version
- Etienne and Markus agreed on adding yet another digit
- Check that your program works
- git add file
- git commit
- Automatic commit message indicating conflict resolution

  Pi = 3.14159;

Undoing with git

Un-stage staged files: `git restore --staged`

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes to be committed:
        modified:   list.txt

$ git restore --staged list.txt
```

NB: `git restore --staged list.txt` <=> `git reset HEAD list.txt`

Get back to last repo version: `git restore`

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes not staged for commit:
        modified:   list.txt

$ git restore list.txt
```

⚠️ modifications are not recoverable

⚠️ files should not be staged

Get back to a given repo version: `git checkout`

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes not staged for commit:
        modified:   list.txt

$ git checkout eb2b909 -- list.txt # bring list.txt at version eb2b909
$ git checkout supermarket -- list.txt # bring list.txt at version pointed by branch supermarket
```

NB: `git restore list.txt` <=> `git checkout HEAD -- list.txt`

<=> `git checkout -- list.txt`

⚠️ modifications are not recoverable (if any)

⚠️ files should not be staged

Reset the whole working dir to a given repo version: `git reset --hard`

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes not staged for commit:
        modified:   list.txt
        modified:   README.md

$ git reset --hard eb2b909
```

`git reset --hard` moves `master` to eb2b909 and updates the working dir

=> Discards all modifications

⚠️ modifications are not recoverable (if any)

Reset whole working dir: `git reset --hard`

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes not staged for commit:
        modified:   list.txt
        modified:   README.md

$ git reset --hard HEAD
```

`git reset --hard` moves the branch to the given commit and updates the working directory

=> As HEAD is pointing at master, master does not move.

=> Working dir reset to the master revision

=> Discards all modifications

⚠️ modifications are not recoverable (if any)

Undo one or several commits: `git revert`

`git revert HEAD` => Undo last commit

`git revert HEAD^` => Undo second to last commit

`git revert HEAD~3` => Undo fourth to last commit

`git revert 9e7185e` => Undo a specific commit

`git revert HEAD^3..HEAD` => Undo a specific range of commits

New commits are created to undo older commits