Basic Tools for System Management

Frederic.Mallet@univ-cotedazur.fr

UNIVERSITÉ CÔTE D'AZUR

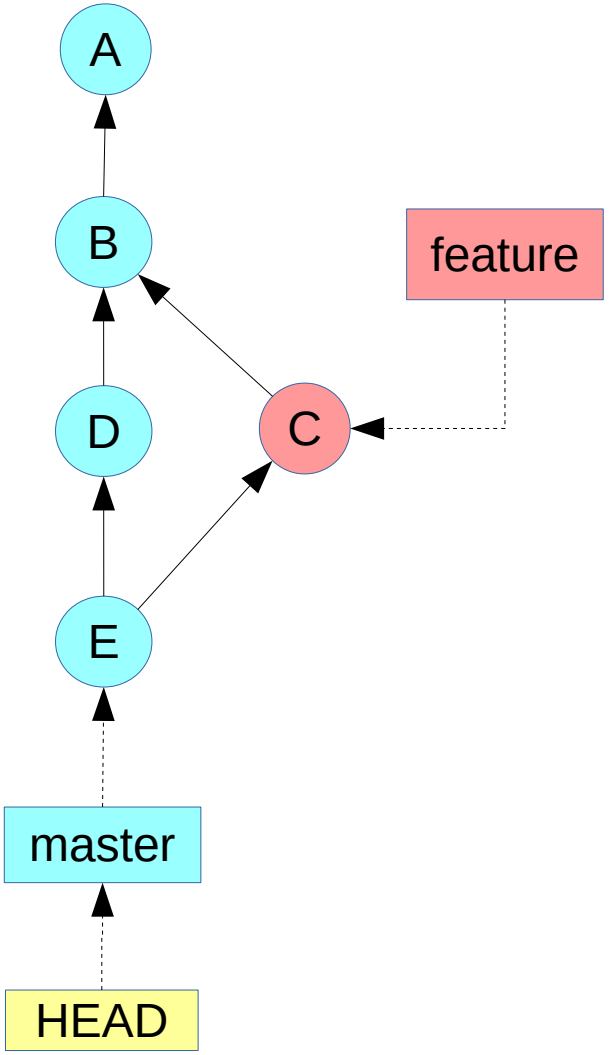
Objectives

- Learn how to manage your own computer
  - Using the shell
    - Linux / Windows
  - Version control and git
  - Installing a Virtual Machine
    - Linux Mint 18.4
  - Using bash
  - Eclipse: JDT and Modeling

Git graph

- git log --oneline --graph

```
fmallet@chevalerios ~/git/light-ccsl
File  Edit  View  Search  Terminal  Help
* 67df470 Fix bug in generating errors (add if) replace c = c + 1 by c++ add com
ment at the beginning
* ce6cf47 Fix initial State Fix problems with import due to change in package na
ming convention
* cbf4156 Add the explicit notion of state
* ebc2dd4 Fix another bug in comments Add the explicit notion of Region, Block,
Comment, Statement
* 9b8f7f9 Fix a bug in merge
*   d5cf68c Merge branch 'master' of git@gitlab.inria.fr:fmallet/light-ccsl.git
|\
| * 1a96f7b Make all kieler specific code into a separate plugin
* | 77602a5 Fix a bug in comments
* | 9c59e71 Change Package Name
* | ad4c52f Make all kieler specific code into a separate plugin
|/
* c299baf Propose to use SCTX format
* 4d9f93a core becomes safety
* 7b97b48 Remove deprecated plugins. Has been replaced by TimeSquare safety.core
.ui plugin
* eb16774 Add SOS executor with SAT4J
* 5b8e523 Move dependency to TimeSquare plugin
* 480de28 Move to TimeSquare plugin
* b7cb3f1 Move dependency to TimeSquare plugin
* 6f67480 move lightccsl.core to old, replaced by fr.kairos.timesquare.safety.co
re from TimeSquare git
*   0ebdda4 fix merge
|\
| * d1d292e Add Comments and authors to lighccsl.core. Update the copyright 2017
,2018 removing author
* | 8dab55e Implements an interactive solver Fix bug in intersection, minus, per
iodic and delayFor
fmallet@chevalerios ~/git/light-ccsl $
```



Git objects and IDs

- Git Objects
  - Commit object – small text file
  - Annotated tag – a reference to a specific commit
  - Tree – Directories and filenames in the project
  - Blob – The content of a file in the project
- Git IDs = name of Git object
  - SHA-1 values = 40 characters hexadecimal string
  - Short id = 4-7 characters

```
fmallet@chevalerios ~/git/pCCSL $ git log
commit 6289c8d7af3f5a6554823b3a12a712ada8a9ee9d
```


git hash-object <file>
git show 6289c

Git References

- User friendly name
  - Points to SHA-1 id
  - Points to another reference (symbolic reference, like HEAD)
- Example: HEAD, master, origin
  - git show HEAD
- Branch labels are git references
  - Stored in .git/refs/heads
- HEAD (.git/HEAD): points to the current commit
  - branch label on the current branch
  - One HEAD per repository

```
fmallet@chevalerios ~/git/light-ccsl $ cat .git/refs/heads/master
67df470538e135bf3717c2720f19c598ec29e0bf
```

Prior Commits

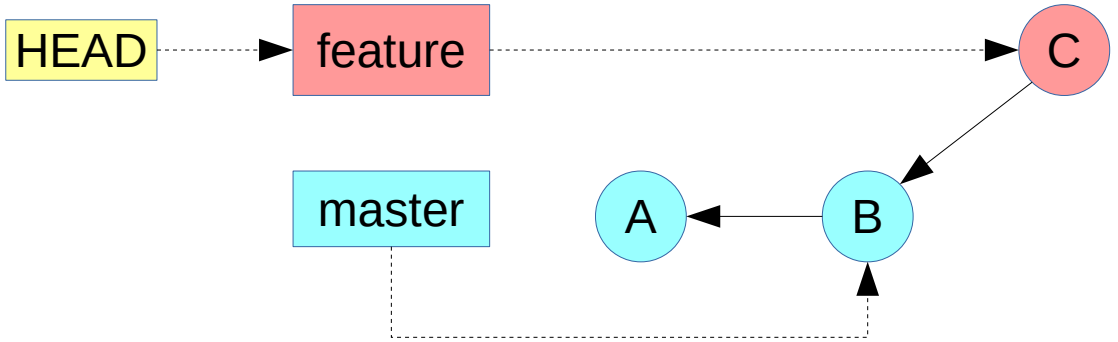
- Use tild (~)
  - ~ or ~1 = parent
    - git show HEAD~
  - ~2 or ~~ = parent's parent
- Use caret (^): parent in a merge commit
  - ^ or ^1: first parent of the commit
  - ^2 – second parent of the merge commit
  - ^^ - first parent's first parent
- Combine
  - HEAD ~^2: second parent of the first parent of HEAD

TAGS

- Reference/label attached to a commit
  - Lightweight tag = reference
  - Annotated tag
    - Includes tag author, tag date, tag message, commit ID
    - Can be signed and verified using GNU Privacy Guard (PGP)
- Commands:
  - git tag <tagname> [<commit>]
    - git tag v1,0
    - git tag v0.1 HEAD^
  - git tag → v1.0, v0.1
  - git tag -a [-m <msg> | -F <file>] <tagname> [<commit>]
  - git tag -d <tagname> to delete tags
- You need to push the tags to the remote repository
  - git push <remote> <tagname>
  - git push <remote> --tags to push all the tags

Branches

- Branch
  - set of commits that trace back to the first commit
  - In practice, this is just a reference
  - Topic vs. long-running branches
    - Bug fixes vs. long-lived branches
  - **git branch** to display the list of branches



Dealing with branches

- Creating a branch (just a new label)
  - **git branch <name>**
  - It does not change the current branch
- Checkout
  - **git checkout <branch_or_commit>**
  - Put the HEAD on the branch or commit
  - If checkout an older commit then HEAD is detached
    - You need to create a branch before committing anything else
- In one-single instruction
  - **git checkout -b <branchname> [<dangling_commit>]**
- Deleting a branch
  - Only if there is no dangling commits (-D to force) => garbage collection
  - **git branch -d feature**
  - **git reflog** => undo accidental branch deletion

Merging

- Combines a topic branch into a base branch
  - It creates a merge commit
  - Then merged commits belong to both branches
- 4 kinds of merge
  - Fast-forward merge
  - Merge commit
  - Squash merge
  - Rebase

Fast-forward Merge

- It moves the base branch label to the tip of the topic branch
  - If no commit has been done to the base branch
  - Only the base branch label is moved
- 3 Steps to fast-forward merge
  - **git checkout master** (base branch)
  - **git merge feature** (topic branch)
  - **git branch -d feature** (delete the topic branch)

Merge commit

- Combines the commits at the tips of the merged branches
- It places the result in the merge commit
- 3 steps (if no conflicts)
  - **git checkout master** (base branch)
  - **git merge feature** (topic branch)
    - Accept or modify the merge message
  - **git branch -d feature** (delete the topic branch)
- Use git merge --no-ff <featurename>
  - Force the merge commit even if fast-forwardable

Assessment

- In Git, what is modeled as a directed acyclic graph?
- How are Git commits connected?
- What is a Git ID?
- If a large file changes by one character, what would you expect to happen to its corresponding SHA-1 value?
- What do branch labels point to?
- How many HEAD references are in a local repository?
- What happens when a branch is created?
- What does a detached HEAD mean?
- What does "deleting a branch" immediately do?

Lab

- Answer the questions of the previous slide and commit your answers onto a branch with your name within a file called “**lab3-<name>-assess.md**”
- Create two branches within your git repository
  - One with your **name** and one **<name>-devel**
  - Do something that would create a fast-forward commit from **<name>-devel** onto **<name>**
  - Do something else to create a merge commit without conflicts from **<name>-devel** onto **<name>**
- Backup your repository into another “backup” directory using rsync
  - Write the command that you used to do the backup and that you should use to restore it in a file “**lab3-<name>-backup.md**” within your branch of your git repository.