ENSSAT

LANNION

NUMÉRIQUE
& PHOTONIQUE

UNIVERSITÉ DE RENNES 1

École affiliée IMT

ENSSAT
LANNION
NUMÉRIQUE
& PHOTONIQUE

Software Development: Introduction to GIT

SysNum • 1A

robin.gerzaguet@enssat.fr

Outline

Outline

Objectives

- A small introduction to GIT
  - What is a versioning tool ?
  - Why it is useful
  - And why GIT is a must

- But also more than that
  - Why versioning can help to be more efficient in workflow (continuous integration)
  - and a way to document your contributions with markdown

Enhanced lecture (?)

- Klaxoon for Quizz and feedback

Partitioning

- 1 lecture of 2 hours (very dense !)
- 6 practice lab under Linux with Github classes
  - Introduction to GIT (`https://lab.github.com/githubtraining/introduction-to-github`)
  - Managing merge conflicts (`https://lab.github.com/githubtraining/managing-merge-conflicts`)
  - Github actions (`https://lab.github.com/githubtraining/github-actions:-hello-world`)
  - And why markdown is good and you should use it (`https://lab.github.com/githubtraining/communicating-using-markdown`)

Evaluation

English report on open source code, one page to be pushed in a common repo !

- More information here: `https://gitlab.enssat.fr/rgerzagu/gitandmarkdown`)

Gratitude

These slides and the lecture structure have been inherited from the work of several (awesome) persons:

- Anthony Baire: `https://people.irisa.fr/Anthony.Baire/`
- Karol Desnos:`https://kdesnos.fr`
  - Checkout his videos on youtube: `https://www.youtube.com/watch?v=4AdO4VfbsVw`

A very classic cheat sheet from Atlassian

- `https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet`

Version Control Systems

A Version Control System

How to efficiently manage different code versions ?

- Code (or any material) may vary from time to time with different objectives
- Code can be collaborative with several persons working on same pool of files
- Manual numbering is not the solution !

Figure: Which is the first ? The last ? Different number formatting ? And disk space management ?

A Version Control System

This is where *Version Control System (VCS)* comes into play

- Manage your code and its evolutions,
- Keep an history to go backwards.

For what ?

- Mostly for code, but also for latex and any text file!
- Not adapted to big data, media files... [there are specific tools for that]

Key features

- Saving code and its history
- Possibility to restore at a specific time/state
- Merge works from different persons or different workflows

Centralized versus decentralized VCS

Centralized VCS

The code base is stored only at one place

- CVS (Concurrent Versioning System, old and not very used right now)
- SVN (Subversion)

Decentralized VCS

- GIT
- Mercurial (Hg)
- Bazaar (bzr)

▶ Lot of different solution and GIT is the most popular at the moment

GIT

GIT is a decentralized open source VCS

- Available on all platform (i386 // ARM) and all Operating Systems
- Creating in 2005 by Linus Torvald to host Linux code source

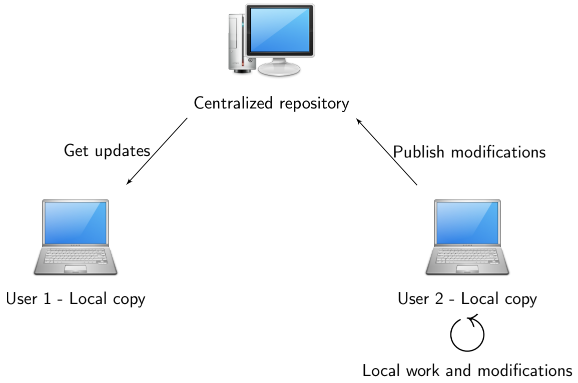
Git history

The development of Git began on 3 April 2005. Torvalds announced the project on 6 April and became self-hosting the next day. The first merge of multiple branches took place on 18 April. Torvalds achieved his performance goals; on 29 April, the nascent Git was benchmarked recording patches to the Linux kernel tree at the rate of 6.7 patches per second. On 16 June, Git managed the kernel 2.6.12 release.

⚠ Do not confuse GIT (the powerful tool) with Github or Gitlab (the host // environment based on GIT + extras)

Repository overview

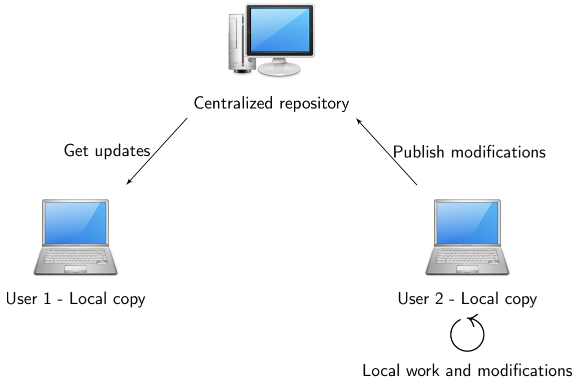
- Source code is placed in a *centralized repository*
- All users have local copy of the code
- Users get modifications from the repo and works locally
- After doing all modifications, repository is updated through a publication



Repository overview

- Repo can be duplicated and source code is copied at every user place: decentralized system !
- Asynchronous works are permitted (not lock // unlocked system)

▶ Need to monitor **conflicts** (code modified at several places)



GIT with local repository

File repository

You should store all files that are not generated by a tool:

- source files (.c .cpp .java .y .l .tex .jl...)
- build scripts / project files (Makefile, CMakefile.txt ...)
- documentation files (.txt README ...)
- resource files (images, audio, . . .)

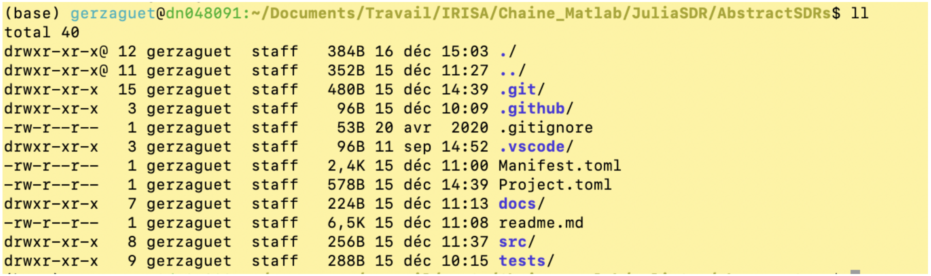
You should not store generated files (or you will experience many unneccessary conflicts)

- .o .a .so .dll .class .jar .exe .dvi .ps .pdf
- source files / build scripts when generated by a tool (like autoconf, cmake, lex, yacc)

Repository overview

Repository definition

It corresponds to a special folder (`.git`) working folder that will be used that will describe each commit (history) and branches (see later)

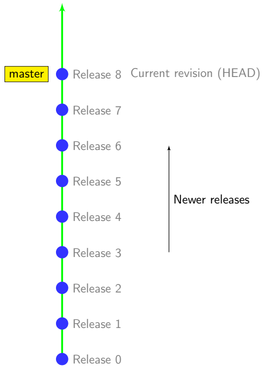


```
(base) gerzaguet@dn048091:~/Documents/Travail/IRISA/Chaine_Matlab/JuliaSDR/AbstractSDRs$ ll
total 40
drwxr-xr-x@ 12 gerzaguet  staff   384B 16 déc 15:03 ./
drwxr-xr-x@ 11 gerzaguet  staff   352B 15 déc 11:27 ../
drwxr-xr-x  15 gerzaguet  staff   480B 15 déc 14:39 .git/
drwxr-xr-x   3 gerzaguet  staff    96B 15 déc 10:09 .github/
-rw-r--r--   1 gerzaguet  staff    53B 20 avr  2020 .gitignore
drwxr-xr-x   3 gerzaguet  staff    96B 11 sep 14:52 .vscode/
-rw-r--r--   1 gerzaguet  staff   2,4K 15 déc 11:00 Manifest.toml
-rw-r--r--   1 gerzaguet  staff   578B 15 déc 14:39 Project.toml
drwxr-xr-x   7 gerzaguet  staff   224B 15 déc 11:13 docs/
-rw-r--r--   1 gerzaguet  staff   6,5K 15 déc 11:08 readme.md
drwxr-xr-x   8 gerzaguet  staff   256B 15 déc 11:37 src/
drwxr-xr-x   9 gerzaguet  staff   288B 15 déc 10:15 tests/
```

To transform a folder into a GIT repository run the command

```
gerzaguet@llamrei:~/ git init
```

Repository overview



- All history is stored (differential)
- Revision corresponds to code variations (minor or majors)
- Possibility to `tag` some of the releases

ENSSAT LANNION

GIT - Commit

Definition

A commit is a

- Set of modifications grouped into one global modification
- These modifications are identified by **Hash code** SHA-1

```
commit 4d3b51e0eae6e32da84ba030a706e1d4dee452d6
```

- The commit will be created jointly with a **message** (important for code monitoring)
- The commit will lead to a new revision of the code when published to the repository
- Possibility to see the history of the commit with a tree view

ENSSAT LANNION

GIT - Commit

Usual version control systems provide two spaces:

- the repository
  (the whole history of your project)
- the working tree (or local copy)
  (the files you are editing and that will be in the next commit)

Git introduces an intermediate space : the staging area (also called index)
The index stores the files scheduled for the next commit:

- git add files $\rightarrow$ copy files into the index
- git commit $\rightarrow$ commits the content of the index

GIT - Commit

- Working copy is the local folder where you work
- Index aggregates all the modifications
- `Master` is the name of the default branch (see later)

A very usefull glance here: `https://ndpsoftware.com/git-cheatsheet.html`

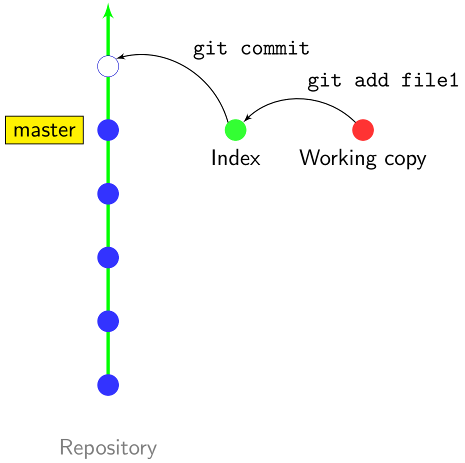


Figure: Difference between working copy, index and repository

GIT - Commit

How we do in practice [Terminal mode] ?

1. You have a local working repository and you do some modifications on one or several files
2. Modifications are OK, you add the files to be monitored

```
gerzaguet@llamrei:~/ git add myFile1 myFile2
```

3. and the commit can be registered in the index

```
gerzaguet@llamrei:~/ git commit -m "Adding myFile1 and myFile2"
```

⚠Message shall be clear for easy repo monitoring !

Possibility to use editor based IDE (VSCode, Atom . . .) for easier commits (part of files,...)

Navigating in Git Index

- Files are added with `git add`. But what files have been modified ?
- The **Git index** lists
  - all modified files
  - all untracked files (files that have never been added)

```
gerzaguet@llamrei : git status
On branch master
Your branch is up to date with 'github/master'.
Changes not staged for commit:
             modified:    myFile1
             modified:    myFile2
Untracked files:
               myFile3
```

Navigating in Git Index

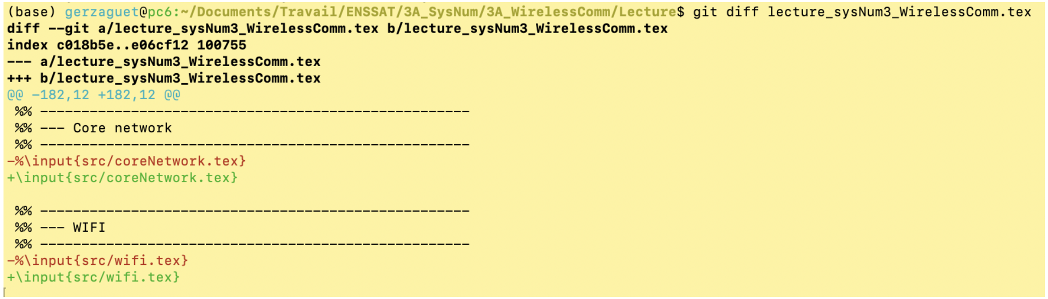
From this, easy to add or remove the desired files

```
gerzaguet@llamrei:~/ git add myFile1 myFile3
gerzaguet@llamrei:~/ git rm myFile2
```

Always have a look on the git status to monitor which file is in, which file is out !

Navigating in Git Index

- `git status` but what is modified in the file ?
- You can use `git diff myFile1` to see the difference between the staged file and the last modifications.



```
(base) gerzaguet@pc6:~/Documents/Travail/ENSSAT/3A_SysNum/3A_WirelessComm/Lecture$ git diff lecture_sysNum3_WirelessComm.tex
diff --git a/lecture_sysNum3_WirelessComm.tex b/lecture_sysNum3_WirelessComm.tex
index c018b5e..e06cf12 100755
--- a/lecture_sysNum3_WirelessComm.tex
+++ b/lecture_sysNum3_WirelessComm.tex
@@ -182,12 +182,12 @@
 %% ----------------------------------------------------
 %% --- Core network
 %% ----------------------------------------------------
-%\input{src/coreNetwork.tex}
+\input{src/coreNetwork.tex}

 %% ----------------------------------------------------
 %% --- WIFI
 %% ----------------------------------------------------
-%\input{src/wifi.tex}
+\input{src/wifi.tex}
```

Diff view shows the removed lines in red and added lines in green.

Navigating in Git Index

Git diff

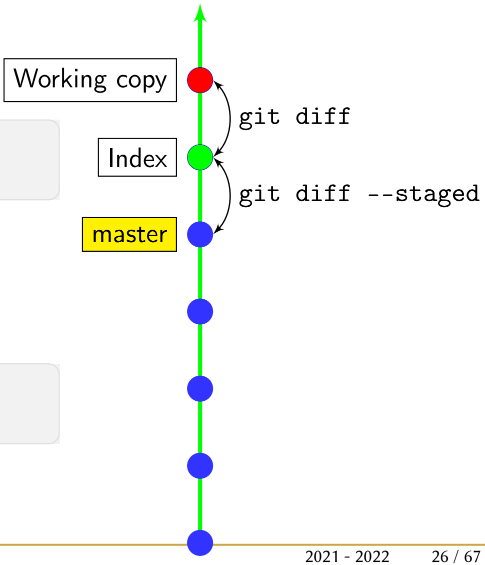
Complete syntax is

```
gerzaguet@llamrei:~/ git diff
[rev a [rev b]] [--path...]
```

- by default rev a is the index
- by default rev b is the working copy

```
gerzaguet@llamrei:~/ git diff
--staged [ rev a ] [ -- path ...]
```

- shows the differences between rev a and index



Navigating in git history

- Each time you commit, you will store the added file states
- Possibility to retrieve the state of the commit
  - All files at their states
  - Untracked files will not be shown

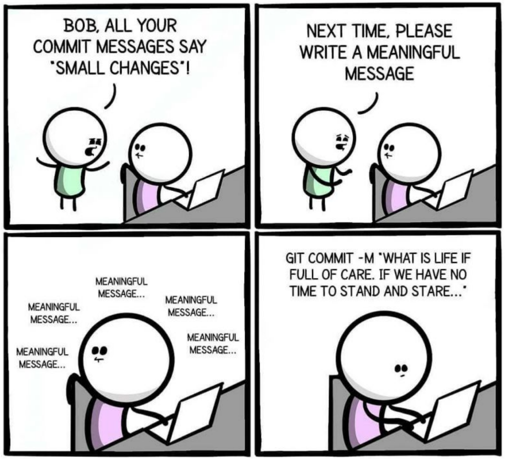
Possibility to switch to a different repository index with the commit identifier

```
gerzaguet@llamrei:~/ git checkout 4d3b51e0eae6e32da84ba030a706e1d4dee452d6
```

- Need to find the appropriate key based on commit index.
- Clearer commit messages helps to find the desired release.
- Alternatively tags can be used to mark some important releases.

Navigating in git history

- Commit messages is the best tool to find modifications in history
- Use short and clear commit messages.

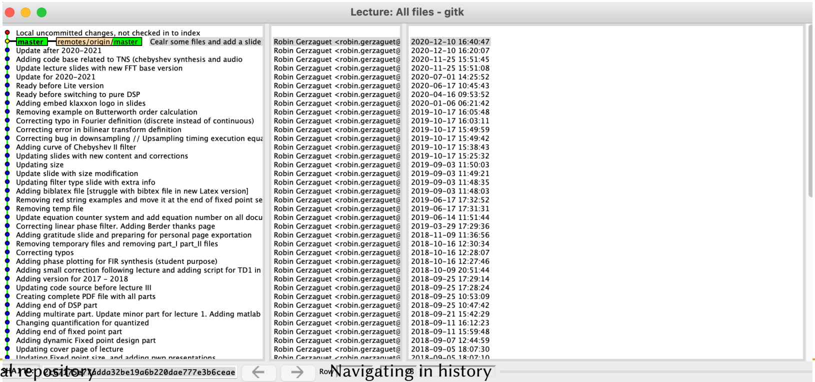

BOB, ALL YOUR COMMIT MESSAGES SAY "SMALL CHANGES"!
NEXT TIME, PLEASE WRITE A MEANINGFUL MESSAGE
MEANINGFUL MESSAGE...
MEANINGFUL MESSAGE...
MEANINGFUL MESSAGE...
MEANINGFUL MESSAGE...
MEANINGFUL MESSAGE...
GIT COMMIT -M "WHAT IS LIFE IF FULL OF CARE. IF WE HAVE NO TIME TO STAND AND STARE..."

Navigating in git history

Alternative possibility to navigate in history is to use `gitk`

```
gerzaguet@llamrei:~/ gitk
```

- It opens a GUI with the history, the branches, the commit names and dates



Lecture: All files - gitk

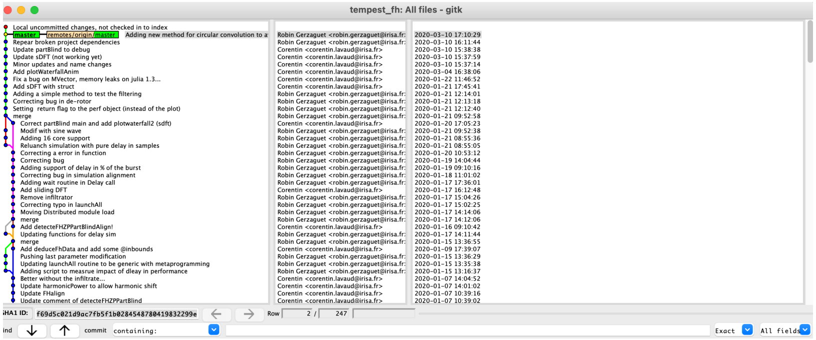
```
Local uncommitted changes, not checked in to index
master  remotes/origin/master  Cealr some files and add a slide    Robin Gerzaguet <robin.gerzaguet@   2020-12-10 16:40:47
Update after 2020-2021                                             Robin Gerzaguet <robin.gerzaguet@   2020-12-10 16:20:07
Adding code base related to TNS (chebyshev synthesis and audio     Robin Gerzaguet <robin.gerzaguet@   2020-11-25 15:51:45
Update lecture slides with new FFT base version                    Robin Gerzaguet <robin.gerzaguet@   2020-11-25 15:51:08
Update for 2020-2021                                               Robin Gerzaguet <robin.gerzaguet@   2020-07-01 14:25:52
Ready before Lite version                                          Robin Gerzaguet <robin.gerzaguet@   2020-06-17 10:45:43
Ready before switching to pure DSP                                 Robin Gerzaguet <robin.gerzaguet@   2020-04-16 09:53:52
Adding embed klaxxon logo in slides                                Robin Gerzaguet <robin.gerzaguet@   2020-01-06 06:21:42
Removing example on Butterworth order calculation                  Robin Gerzaguet <robin.gerzaguet@   2019-10-17 16:05:48
Correcting typo in Fourier definition (discrete instead of continuous)  Robin Gerzaguet <robin.gerzaguet@   2019-10-17 16:03:11
Correcting error in bilinear transform definition                  Robin Gerzaguet <robin.gerzaguet@   2019-10-17 15:49:59
Correcting bug in downsampling // Upsampling timing execution equa Robin Gerzaguet <robin.gerzaguet@   2019-10-17 15:49:42
Adding curve of Chebyshev II filter                                Robin Gerzaguet <robin.gerzaguet@   2019-10-17 15:38:43
Updating slides with new content and corrections                   Robin Gerzaguet <robin.gerzaguet@   2019-10-17 15:25:32
Updating size                                                      Robin Gerzaguet <robin.gerzaguet@   2019-09-03 11:50:03
Update slide with size modification                                Robin Gerzaguet <robin.gerzaguet@   2019-09-03 11:49:21
Updating filter type slide with extra info                         Robin Gerzaguet <robin.gerzaguet@   2019-09-03 11:48:35
Adding biblatex file [struggle with bibtex file in new Latex version]  Robin Gerzaguet <robin.gerzaguet@   2019-09-03 11:48:03
Removing red string examples and move it at the end of fixed point se  Robin Gerzaguet <robin.gerzaguet@   2019-06-17 17:32:52
Removing temp file                                                 Robin Gerzaguet <robin.gerzaguet@   2019-06-17 17:31:31
Update equation counter system and add equation number on all docu Robin Gerzaguet <robin.gerzaguet@   2019-06-14 11:51:44
Correcting linear phase filter. Adding Berder thanks page          Robin Gerzaguet <robin.gerzaguet@   2019-03-29 17:29:36
Adding gratitude slide and preparing for personal page exportation Robin Gerzaguet <robin.gerzaguet@   2018-11-09 11:36:56
Removing temporary files and removing part_I part_II files         Robin Gerzaguet <robin.gerzaguet@   2018-10-16 12:30:34
Correcting typos                                                   Robin Gerzaguet <robin.gerzaguet@   2018-10-16 12:28:07
Adding phase plotting for FIR synthesis (student purpose)          Robin Gerzaguet <robin.gerzaguet@   2018-10-16 12:27:46
Adding small correction following lecture and adding script for TD1 in  Robin Gerzaguet <robin.gerzaguet@   2018-10-09 20:51:44
Adding version for 2017 - 2018                                     Robin Gerzaguet <robin.gerzaguet@   2018-09-25 17:29:14
Updating code source before lecture III                            Robin Gerzaguet <robin.gerzaguet@   2018-09-25 17:28:24
Creating complete PDF file with all parts                          Robin Gerzaguet <robin.gerzaguet@   2018-09-25 10:53:09
Adding end of DSP part                                             Robin Gerzaguet <robin.gerzaguet@   2018-09-25 10:47:42
Adding multirate part. Update minor part for lecture 1. Adding matlab  Robin Gerzaguet <robin.gerzaguet@   2018-09-21 15:42:29
Changing quantification for quantized                              Robin Gerzaguet <robin.gerzaguet@   2018-09-11 16:12:23
Adding end of fixed point part                                     Robin Gerzaguet <robin.gerzaguet@   2018-09-11 15:59:48
Adding dynamic Fixed point design part                             Robin Gerzaguet <robin.gerzaguet@   2018-09-07 12:44:59
Updating cover page of lecture                                     Robin Gerzaguet <robin.gerzaguet@   2018-09-05 18:07:30
Updating Fixed point size, and adding pwn presentations            Robin Gerzaguet <robin.gerzaguet@   2018-09-05 18:07:10
SHA1 ID: [illegible]adda32be19a6b220dae777e3b6ceae    Row
```

Navigating in git history

Alternative possibility to navigate in history is to use `gitk`

```
gerzaguet@llamrei:~/ gitk
```

- It opens a GUI with the history, the branches, the commit names and dates



```
tempest_fh: All files - gitk

Local uncommitted changes, not checked in to index
master  remotes/origin/master  Adding new method for circular convolution to a   Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-03-10 17:10:29
Repear broken project dependencies                                  Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-03-10 16:11:44
Update partBlind to debug                                           Corentin <corentin.lavaud@irisa.fr>         2020-03-10 15:38:38
Update sDFT (not working yet)                                       Corentin <corentin.lavaud@irisa.fr>         2020-03-10 15:37:59
Minor updates and name changes                                      Corentin <corentin.lavaud@irisa.fr>         2020-03-10 15:37:14
Add plotWaterfallAnim                                               Corentin <corentin.lavaud@irisa.fr>         2020-03-04 16:38:06
Fix a bug on MVector, memory leaks on julia 1.3...                  Corentin <corentin.lavaud@irisa.fr>         2020-01-22 11:46:52
Add sDFT with struct                                                Corentin <corentin.lavaud@irisa.fr>         2020-01-21 17:45:41
Adding a simple method to test the filtering                        Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-21 12:14:01
Correcting bug in de-rotor                                          Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-21 12:13:18
Setting  return flag to the perf object (instead of the plot)       Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-21 12:12:40
merge                                                               Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-21 09:52:58
  Correct partBlind main and add plotwaterfall2 (sdft)              Corentin <corentin.lavaud@irisa.fr>         2020-01-20 17:05:23
  Modif with sine wave                                              Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-21 09:52:38
  Adding 16 core support                                            Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-21 08:55:36
  Reluanch simulation with pure delay in samples                    Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-21 08:55:05
  Correcting a error in function                                    Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-20 10:53:12
  Correcting bug                                                    Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-19 14:04:44
  Adding support of delay in % of the burst                         Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-19 09:10:16
  Correcting bug in simulation alignment                            Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-18 11:01:02
  Adding wait routine in Delay call                                 Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-17 17:36:01
  Add sliding DFT                                                   Corentin <corentin.lavaud@irisa.fr>         2020-01-17 16:12:48
  Remove infiltrator                                                Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-17 15:04:26
  Correcting typo in launchAll                                      Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-17 15:02:25
  Moving Distributed module load                                    Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-17 14:14:06
  merge                                                             Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-17 14:12:06
  Add detecteFHZPPartBlindAlign!                                    Corentin <corentin.lavaud@irisa.fr>         2020-01-16 09:10:42
  Updating functions for delay sim                                  Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-17 14:11:44
  merge                                                             Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-15 13:36:55
  Add deduceFhData and add some @inbounds                           Corentin <corentin.lavaud@irisa.fr>         2020-01-09 17:39:07
  Pushing last parameter modification                               Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-15 13:36:29
  Updating launchAll routine to be generic with metaprogramming     Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-15 13:35:38
  Adding script to measrue impact of dleay in performance           Robin Gerzaguet <robin.gerzaguet@irisa.fr   2020-01-15 13:16:37
  Better without the infiltrate...                                  Corentin <corentin.lavaud@irisa.fr>         2020-01-07 14:04:52
  Update harmonicPower to allow harmonic shift                      Corentin <corentin.lavaud@irisa.fr>         2020-01-07 14:01:02
  Update FHalign                                                    Corentin <corentin.lavaud@irisa.fr>         2020-01-07 10:39:16
  Update comment of detecteFHZPPartBlind                            Corentin <corentin.lavaud@irisa.fr>         2020-01-07 10:39:02

SHA1 ID: f69d5c021d9ac7fb5f1b0284548780419832299e      Row 2 / 247
ind   commit containing:      Exact   All fields
```

Resetting changes

Want to remove something done in a specific file ?

```
gerzaguet@llamrei:~/     git reset [ --hard ] [ -- path ...]
```

git reset cancels the changes in the index (and possibly in the working copy)

- `git reset` drops the changes staged into the index, but the working copy is left intact
- `git reset –hard` drops all the changes in the index and in the working copy

Other useful local commands

```
gerzaguet@llamrei:~/ git show
```

- Information of the last commit (or arbitrary commit with associate ID)

```
gerzaguet@llamrei:~/ git log
```

- Show the history

```
gerzaguet@llamrei:~/ git mv myFile myNewLoc
```

- Move file to other place. Equivalent to `cp myFile myNewLoc && git rm myFile && git add myNewLoc`

Branches

Each commit object has a list of parent commits:

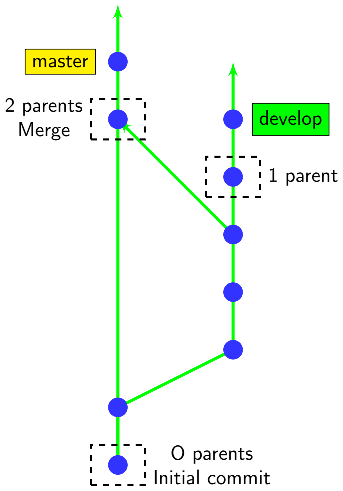
- 0 parents $\rightarrow$ initial commit
- 1 parent $\rightarrow$ ordinary commit
- 2+ parents $\rightarrow$ result of a merge

▶ This is a Direct Acyclic Graph

Branches

Develop new feature on a separate branch

- Develop without impact the master branch
- When it is ready, merge the develop branch into the master branch



Repository

Branches

Create a new branch

```
gerzaguet@llamrei:~/ git checkout -b new branch [
starting point ]
```

- starting point is the starting location of the branch (possibly a commit id, a tag, a branch, . . .). If not present, git will use the current location.

Switch between branches

```
gerzaguet@llamrei:~/ git checkout [-m] branch name
```

- With `-m` to request merging local changes into the destination branch.

Merge: where the fun begins

List all branches

```
gerzaguet@llamrei:~/ git branch [-a]
```

- With -a for listing remote branches (see after)

Merge: where the fun begins

Merging branches

- To merge specific branchA in current branch

```
gerzaguet@llamrei:~/ git merge branchA
```

- The merging automatically creates a commit (if no conflict)
- Conflicts may occur if the same file was independently modified in the two branches

Delete a branch

After merging branchA into branchB, one can remove branchA

```
gerzaguet@llamrei:~/ git branch -d branchA
```

Merge managing

Merge conflict

Happens where one file is modified on both branches.

- The operation will not succeed and additional operations are necessary depending on the file type

Text files

- lines changed in only one branch are automatically merged
- if a line was modified in the two branches, then conflict
- Materialized within <<<<<<< MERGE AREA>>>>>>>

Binary files

- Need a manual merge (remove, add)

Merge managing

Example of file after a merge conflict

```
gerzaguet@llamrei:~/ testDir git merge feature
Auto-merging hello.txt
CONFLICT (content): Merge conflict in hello.txt
Automatic merge failed; fix conflicts and then commit.
```

The content of the file is as follows

```
gerzaguet@llamrei:~/ vim hello.txt
<<<<<<< HEAD
This file has been modified in master branch
=======
This file has been modified in release branch a second time
>>>>>>> feature
```

Merge managing

To solve the merge

- Edit the file and remove the non necessary parts
- Or use a direct merge conflict manager (xxdiff, kdiff3, emerge, ...))
- then run

```
gerzaguet@llamrei:~/ git add mergedFiled
```

or

```
gerzaguet@llamrei:~/ git rm corruptedFile
```

GIT with remote repository

Remote repository

For the moment, GIT has been used in local repository

- Versioning code for one location
- Backup, tracking and branching for new features

But real power of GIT is for collaborative work

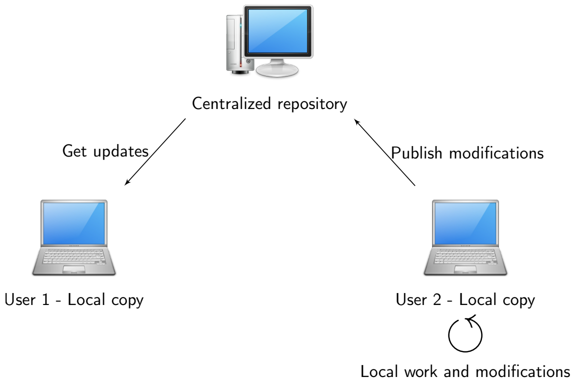
- Team that updates same code base with various features (branches)
- Necessity to manage merge conflicts (more frequents !) and keep a common code base

Remote repository

It requires

- A remote repository where code base will lies
- Possibility to synchronise each local code base with the one of the remote repository

Remote repository



- All local work is done the same way as with local repository
- Need to get data from centralized repository
- Need to push data to centralized repository

Get the image of the repo

First step is to get the code base exactly as it is.

```
gerzaguet@llamrei:~/ git clone https://git.remote.url.repo
```

- Get a clone of the centralized repository
- Get all the previous commit, branches, and so on

Example to get the Linux kernel which is on GitHub

```
gerzaguet@llamrei:~/ git clone https://github.com/torvalds/linux
```

Getting code from remote repository

updates the local mirror of the remote repository:

```
gerzaguet@llamrei:~/ git fetch
```

- It gets the change but it does not apply them !
- Application must be done afterwards with an explicit merge

```
gerzaguet@llamrei:~/ git merge
```

The modifications can be done in one command with *pull* command

```
gerzaguet@llamrei:~/ git pull
```

Getting code from remote repository

To get content of a remote branch

```
gerzaguet@llamrei:~/ git checkout branch
```

- If `branch` does not exist locally, then GIT looks for it in the remote repositories
- If it finds it, then it creates the local branch and configures it to track the remote branch.

To list all branches (both remotes and locals) use

```
gerzaguet@llamrei:~/ git branch -a
```

Sending code to centralized repository

After code modifications, code can be sent to remote repository using `push` command

- This requires remote to be initialized (i.e centralized repository address)
- A repository can have several remotes (push can be used for a specific remote)

```
gerzaguet@llamrei:~/ git push
```

- if the branch is tracking an upstream branch, then the local changes (commits) are propagated to the remote branch
- if on local branch, nothing happens (new local branches are private by default). In this case, do

```
gerzaguet@llamrei:~/ git push -u remote branch
```

- In case of conflict `git push` will fail and require to run `git pull` first

Manage remotes

A centralized repository is characterized by its remote

- Remote repositories are mirrored within the local repository
- It is possible to work with multiple remote repositories
- Each remote repository is identified with a local alias. When working with a unique remote repository, it is usually named `origin`

To add a remote repository

```
gerzaguet@llamrei:~/ git remote add name url
```

- name is a local alias identifying the remote repository
- url is the location of the remote repository

To list all remotes of the current repository

```
gerzaguet@llamrei:~/ git remote -v
```

Git stash

A very usefull Git command

1. Take your uncommitted changes (both staged and unstaged)
2. Save them away for later use in ths stash area
3. Revert them from your working copy

Advantages

When you are in the middle of something, being able to sync with the remote folder and then apply your modification on the top.

- Changes a classic `pull` order (your local change will be applied after the pull)
- The stash is stored *locally* (nothing is pushed to the remote)
- Untracked files will not be stashed by default (not an issue with pull anyway). Use *-a* to stash them also.
- Ignored files (with `.gitignore` will not be stashed by default. Use `-a` to stash them also.

Example

A simple folder with the following git index

```
gerzaguet@llamrei : ls
main.c         processing.c  readme.md
gerzaguet@llamrei : git status
On branch master
Changes not staged for commit:
 (use "git add <file>..." to update what will be committed)
 (use "git checkout -- <file>..." to discard changes
 in working directory)

   modified:   readme.md

Untracked files:
 (use "git add <file>..." to include in what will be committed)

   main.c

no changes added to commit (use "git add" and/or "git commit -a")
```

Example

```
gerzaguet@llamrei : git stash
Saved working directory and index state WIP on master:
0ffea1d first commit
```

```
gerzaguet@llamrei : git status
On branch master
Untracked files:
 (use "git add <file>..." to include in what will be committed)

   main.c
```

- We can no work (and pull) in peace

Reverting the stash

- Getting the stash list

```
gerzaguet@llamrei : git stash list
stash@{0}: WIP on master: 0ffea1d first commit
```

Apply the stash

- `git stash pop` to apply the stash (and remove the stash copy from the stash index)
- `git stash apply` to apply the stash *and* keep the stash copy

Reverting the stash

In case of multiple stashes

- You can revert one specific stash with its index

```
gerzaguet@llamrei : git stash pop stash@{2}
```

- And give stash more convenient names

```
gerzaguet@llamrei : git stash save "New hello world WIP"
```

GIT repository hosts

Several way to host a GIT Repository

- Local server
- External dedicated tools
  - Github is the most famous of them
  - Gitlab

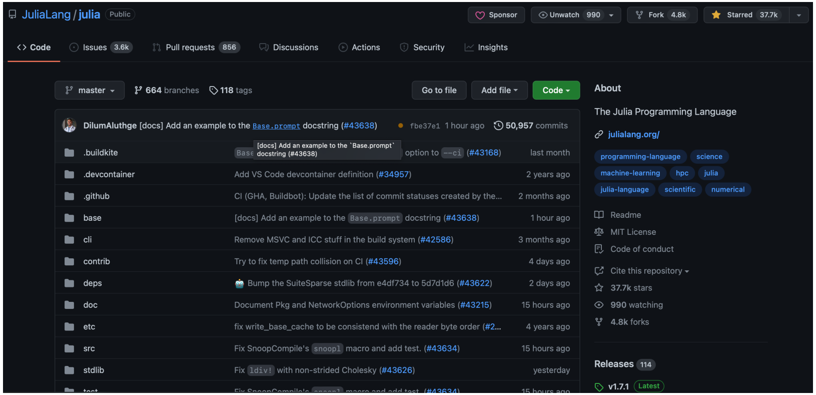
These solutions offers more than just GIT repo:

- Easy member management
- Code integration, merge conflict through graphical interface
- `Pull request (PR)` (external users wants to add something on a repo on which he has no rights)
- Github pages for documentation, markdown support
- Github actions (for Continuous integration, automatic testing, ...)

GIT in practice

Gitlab and Github

These are websites to host Git repositories and collaborate



JuliaLang / julia Public

Sponsor · Unwatch 990 · Fork 4.8k · Starred 37.7k

Code · Issues 3.6k · Pull requests 856 · Discussions · Actions · Security · Insights

master · 664 branches · 118 tags · Go to file · Add file · Code

DilumAluthge [docs] Add an example to the `Base.prompt` docstring (#43638) · fbe37e1 1 hour ago · 50,957 commits

[docs] Add an example to the `Base.prompt` docstring (#43638)

.buildkite	Base … option to `--ci` (#43168)	last month
.devcontainer	Add VS Code devcontainer definition (#34957)	2 years ago
.github	CI (GHA, Buildbot): Update the list of commit statuses created by the...	2 months ago
base	[docs] Add an example to the `Base.prompt` docstring (#43638)	1 hour ago
cli	Remove MSVC and ICC stuff in the build system (#42586)	3 months ago
contrib	Try to fix temp path collision on CI (#43596)	4 days ago
deps	Bump the SuiteSparse stdlib from e4df734 to 5d7d1d6 (#43622)	2 days ago
doc	Document Pkg and NetworkOptions environment variables (#43215)	15 hours ago
etc	fix write_base_cache to be consistend with the reader byte order (#2...	4 years ago
src	Fix SnoopCompile's `snoopl` macro and add test. (#43634)	15 hours ago
stdlib	Fix `ldiv!` with non-strided Cholesky (#43626)	yesterday
test	Fix SnoopCompile's `snoopl` macro and add test. (#43634)	15 hours ago

About

The Julia Programming Language

julialang.org/

programming-language · science · machine-learning · hpc · julia · julia-language · scientific · numerical

Readme

MIT License

Code of conduct

Cite this repository

37.7k stars

990 watching

4.8k forks

Releases 114

v1.7.1 Latest

Figure: Overwiew of GitHub project mainpage (here `https://github.com/JuliaLang/julia`)

Navigation in files

You can access to the code base with

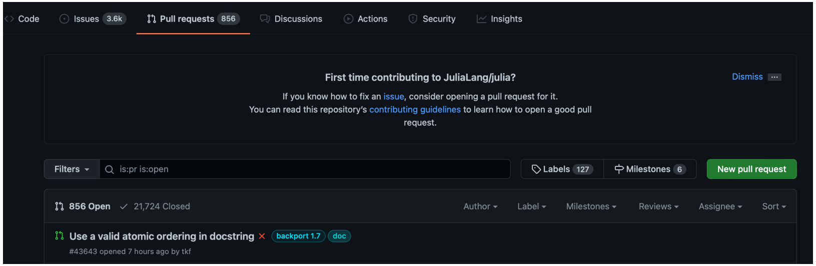
```
gerzaguet@llamrei : git glone github.com/myProject
```

But you can inspect and find lot of information directly on the GitHub webpage

- Documentation and readme with fancy printing
- Badges with actions and coverage (see later)
- Issues and PR

Graphical Pull Request

A pull request can be done with graphic interface



First time contributing to JuliaLang/julia?

If you know how to fix an issue, consider opening a pull request for it.
You can read this repository's contributing guidelines to learn how to open a good pull request.

Use a valid atomic ordering in docstring

#43643 opened 7 hours ago by tkf

Figure: Pull Request in Github

- Between 2 active branches (and often master/main)
- Require reviews from collaborators, and test coverage !
- Only maintainer can finally do the merge

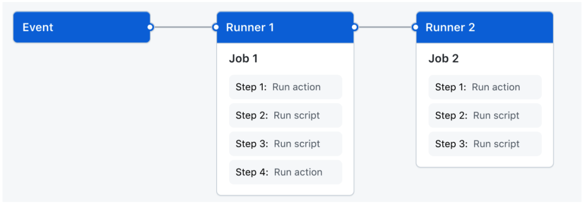
GitHub actions

A very powerful tool to deploy applications in different languages

- Task automatization (doc building)
- Test after commits
- Project management (automatic pull requests based on dependencies. . .)

GitHub action principle

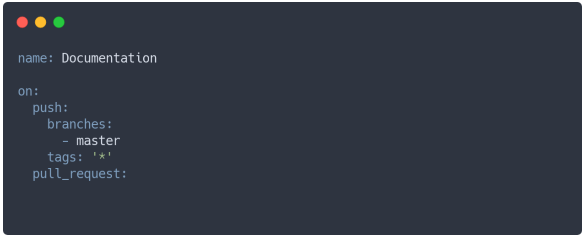
- A workflow (succession of operations) is triggered when a specific event occur (push, merge, new issue, . . .)
- A workflow has one or more sequential jobs that will be run in a virtual machine (*runner*)
- Each jobs can have one or more steps



GitHub actions

The workflow is defined in YAML (YAML Ain't Markup Language) language.
Example to automatize test with Python

- Top of file for the trigger
- Bottom for the workflow itself

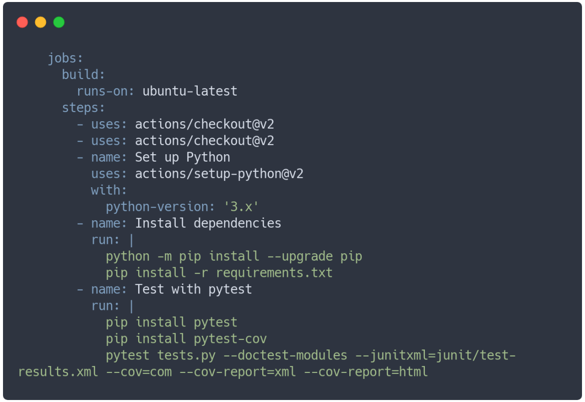


```yaml
name: Documentation

on:
  push:
    branches:
      - master
    tags: '*'
  pull_request:
```

GitHub actions

The workflow is defined in YAML (YAML Ain't Markup Language) language.
Example to automatize test with Python

- Top of file for the trigger
- Bottom for the workflow itself

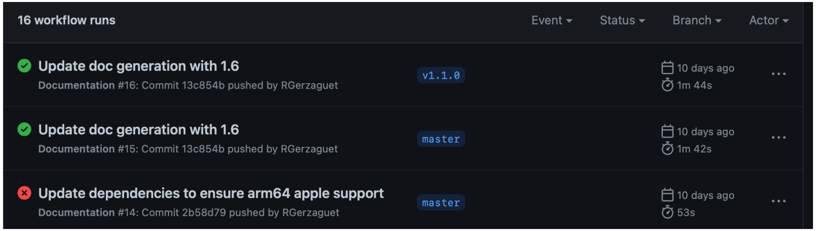


```
jobs:
  build:
    runs-on: ubuntu-latest
  steps:
    - uses: actions/checkout@v2
    - uses: actions/checkout@v2
    - name: Set up Python
      uses: actions/setup-python@v2
      with:
        python-version: '3.x'
    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        pip install -r requirements.txt
    - name: Test with pytest
      run: |
        pip install pytest
        pip install pytest-cov
        pytest tests.py --doctest-modules --junitxml=junit/test-
results.xml --cov=com --cov-report=xml --cov-report=html
```

GitHub actions

The actions can be tracked in the action pane in GitHub

- Check that tests are OK or action terminate successfully
- Tracks bugs and regressions



✓ Update doc generation with 1.6
Documentation #16: Commit 13c854b pushed by RGerzaguet
v1.1.0
10 days ago
1m 44s

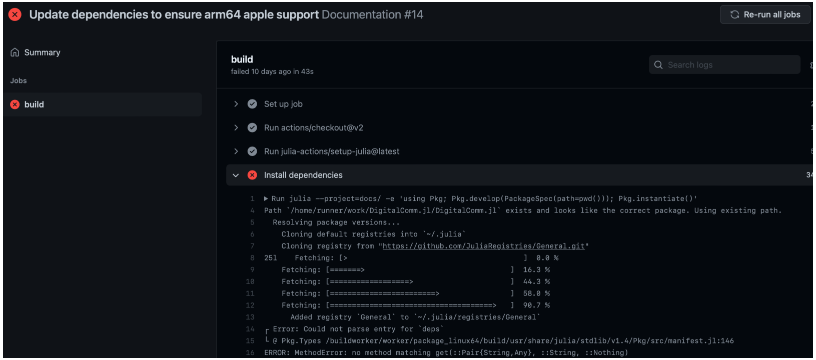
✓ Update doc generation with 1.6
Documentation #15: Commit 13c854b pushed by RGerzaguet
master
10 days ago
1m 42s

✗ Update dependencies to ensure arm64 apple support
Documentation #14: Commit 2b58d79 pushed by RGerzaguet
master
10 days ago
53s

Figure: Panel of successful and failed GitHub actions

GitHub actions

In case of fail, don't panic (yet) and have a look on where the runner fails


Update dependencies to ensure arm64 apple support Documentation #14
Re-run all jobs
Summary
Jobs
build
build
failed 10 days ago in 43s
Search logs
Set up job
Run actions/checkout@v2
Run julia-actions/setup-julia@latest
Install dependencies

```
▶ Run julia --project=docs/ -e 'using Pkg; Pkg.develop(PackageSpec(path=pwd())); Pkg.instantiate()'
Path `/home/runner/work/DigitalComm.jl/DigitalComm.jl` exists and looks like the correct package. Using existing path.
  Resolving package versions...
    Cloning default registries into `~/.julia`
    Cloning registry from "https://github.com/JuliaRegistries/General.git"
251    Fetching: [>                                        ]  0.0 %
    Fetching: [=======>                                 ]  16.3 %
    Fetching: [==================>                      ]  44.3 %
    Fetching: [========================>                ]  58.0 %
    Fetching: [=====================================>   ]  90.7 %
      Added registry `General` to `~/.julia/registries/General`
┌ Error: Could not parse entry for `deps`
└ @ Pkg.Types /buildworker/worker/package_linux64/build/usr/share/julia/stdlib/v1.4/Pkg/src/manifest.jl:146
ERROR: MethodError: no method matching get(::Pair{String,Any}, ::String, ::Nothing)
```

Figure: Description of failed action

▶ More on GitHub actions in the lab

Markdown language

Collaboration requires discussion and explanation

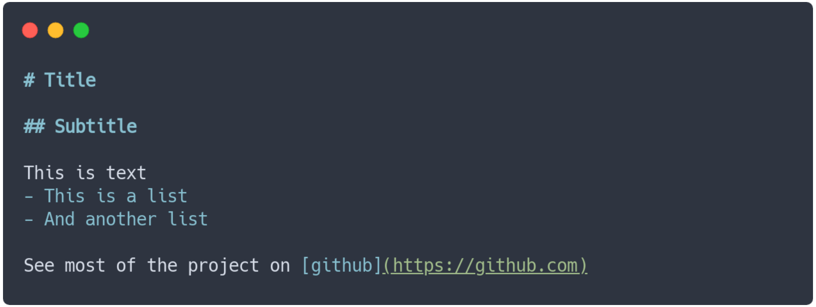
- Need to have text with rich format to ease readability
- But without strong verbosity (Latex) or heavy format (docx). Text should be text !

▶ Here comes Markdown

- Almost only text
- Can be read as it is
- Can be exported in various format (PDF, HTML)

Markdown syntax

This is only text with tags ...

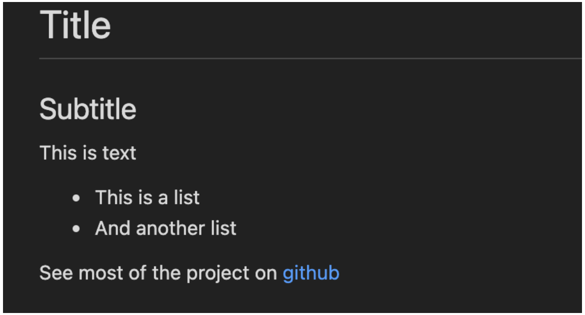


```markdown
# Title

## Subtitle

This is text
- This is a list
- And another list

See most of the project on [github](https://github.com)
```

Figure: Example of markdown syntax

Markdown syntax

With convenient rendering



Title

Subtitle

This is text

- This is a list
- And another list

See most of the project on github

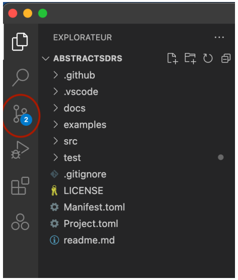
Figure: Example of markdown rendering

▶ Syntax is easy to learn, see for instance
`5https://www.markdownguide.org/cheat-sheet/`

VSCode extension

A simple way to use GIT is to use third party tools

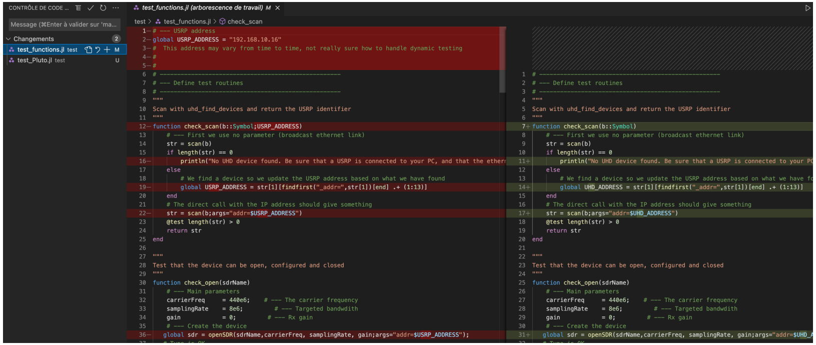
- Git Kraken (`https://www.gitkraken.com`) not open source but free
- VSCode plugin, open source, free, and seamless interaction with VSCode
- Vim and Emacs plugin for those who know



VSCode extension

In the github pane, possibility to see the changes in the file

- To do all GIT commands
- To commit only part of a file !



CONTRÔLE DE CODE ...

Message (⌘Enter à valider sur 'ma...

Changements 2

test_functions.jl test M

test_Pluto.jl test U

test_functions.jl (arborescence de travail) M

test > test_functions.jl > check_scan

```
# --- USRP address
global USRP_ADDRESS = "192.168.10.16"
#  This address may vary from time to time, not really sure how to handle dynamic testing
#
#
# ----------------------------------------------------
# --- Define test routines
# ----------------------------------------------------
"""
Scan with uhd_find_devices and return the USRP identifier
"""
function check_scan(b::Symbol;USRP_ADDRESS)
    # --- First we use no parameter (broadcast ethernet link)
    str = scan(b)
    if length(str) == 0
        println("No UHD device found. Be sure that a USRP is connected to your PC, and that the ethern
    else
        # We find a device so we update the USRP address based on what we have found
        global USRP_ADDRESS = str[1][findfirst("_addr=",str[1])[end] .+ (1:13)]
    end
    # The direct call with the IP address should give something
    str = scan(b;args="addr=$USRP_ADDRESS")
    @test length(str) > 0
    return str
end

"""
Test that the device can be open, configured and closed
"""
function check_open(sdrName)
    # --- Main parameters
    carrierFreq     = 440e6;    # --- The carrier frequency
    samplingRate    = 8e6;          # --- Targeted bandwdith
    gain            = 0;          # --- Rx gain
    # --- Create the device
   global sdr = openSDR(sdrName,carrierFreq, samplingRate, gain;args="addr=$USRP_ADDRESS");
```

```
# ----------------------------------------------------
# --- Define test routines
# ----------------------------------------------------
"""
Scan with uhd_find_devices and return the USRP identifier
"""
function check_scan(b::Symbol)
    # --- First we use no parameter (broadcast ethernet link)
    str = scan(b)
    if length(str) == 0
        println("No UHD device found. Be sure that a USRP is connected to your PC
    else
        # We find a device so we update the USRP address based on what we have fo
        global UHD_ADDRESS = str[1][findfirst("_addr=",str[1])[end] .+ (1:13)]
    end
    # The direct call with the IP address should give something
    str = scan(b;args="addr=$UHD_ADDRESS")
    @test length(str) > 0
    return str
end

"""
Test that the device can be open, configured and closed
"""
function check_open(sdrName)
    # --- Main parameters
    carrierFreq     = 440e6;    # --- The carrier frequency
    samplingRate    = 8e6;          # --- Targeted bandwdith
    gain            = 0;          # --- Rx gain
    # --- Create the device
   global sdr = openSDR(sdrName,carrierFreq, samplingRate, gain;args="addr=$UHD_A
```

Figure: Diff view in VSCode

Conclusion

Conclusion

GIT is a decentralized Version Control System

- Powerful, easy to use
- Handle local and remote repositories
- Manage different branches that can be merged
- Command that can be used in terminal or with third party tools

This is just a basic introduction and GIT is more complicated and powerful that you might think

- Combination of different commit
- `git stash` to save and restore the index
- Commit amending, code rebase, git submodules
- ...