HANDS-ON TUTORIAL

GIT IN PRACTICE

GIGA DOCTORAL SCHOOL 2022



WHO ARE WE?



- Phd student at the GIGA CRC in vivo imaging
 - Freshly graduated!
 - "Electromagnetic modelling of a head"





- High energy physicist, PhD&Ir
- Post-doctoral researcher at the GIGA CRC in vivo imaging

gregory.hammad@uliege.be

mar.grignard@uliege.be

GIT MODEL

- Distributed system
 - Each user has a full copy of the project, its own history and structure.
 - That's why Git works offline too!
- Snapshots
 - Instead of keeping tracks of each file individually, Git creates a "commit snapshot" (commit) of your working copy of the project
 - Allows users to go back to a certain state of the project if necessary...
- Branch:
 - A series of commit snapshots from a given working directory.





gregory.hammad@uliege.be

mar.grignard@uliege.be

GIT'S STRUCTURE

- Working directory:
 - List modified files
 - Modifications are not tracked (i.e not added to the history of modifications)
- Staging area:
 - List of "candidate" files whose modifications are meant to be tracked.
- Local repository:
 - Contains the whole history of changes (ie. commits)
- Remote repository:
 - Copy of the local repository, located on a distant server

mar.grignard@uliege.be



<u>gregory.hammad@uliege.be</u>

LET'S PRACTICE!

Exercise 0

- Initialise a git repository:
 - git init
- Add a file
 - git add "myfile.csv"
- Commit the file:
 - git commit

mar.grignard@uliege.be

WARM-UP

- Add a `ssh key` to your Gitlab account:
 - SSH keys:
 - Private/public key pair
 - Identify yourself to a SSH server (like gitlab)
 - Create a new key or use an existing one:
 - https://gitlab.uliege.be/help/ssh/README

WARM-UP

Add a `ssh key` to your Gitlab account:

🦊 GitLab Projects 🗸 Groups	s 🗸 Activity Milestones Snippets	t → Search Q ()	🔹 ກ 🔁 🎯 -
User Settings	User Settings > SSH Keys		Grégory Hammad @Gregory.Hammad
Profile Account	SSH Keys SSH keys allow you to establish a secure connection between your computer and	Add an SSH key To add an SSH key you need to generate one or use an existing key.	Profile Settings
D Chat	GitLab.	Paste your public SSH key, which is usually contained in the file '~/.ssh/id_rsa.pub' and bec with 'ssh-rsa'. Don't use your private SSH key.	Help Sign out
Access TokensEmails		3. Typically starts with "ssh-rsa"	
A Password			
 A Notifications SSH Keys 		Title	
🔑 GPG Keys		e.g. My MacBook key	
PreferencesActive Sessions		Name your individual key via a title Add key	

mar.grignard@uliege.be

Create a Git project/repository:

🤌 GitLa	Projects 🗸	Groups ~ Activity	Milestones	Snippets		+ ~	Search	۹ () <u>۹</u> (†)	r 🋞
	Projects								
	Your project	s Starred projects	Explore pr	rojects		Filter by name	Last updated	✓ New project	t
	All Persona	 I							
	G GIGA-C Test pro	RC In Vivo Imaging / L	ocalResourc	es / Tutorials / Git-tutorial	Owner			★ 0 updated 3 minutes a	U Igo
	M GIGA-C	RC In Vivo Imaging / N	<i>l</i> anagement	Owner				★ 0 updated 1 hour a	a Igo
	P GIGA-C Package	RC In Vivo Imaging / S to analyse actigraphy	s tudies / Cog data	Nap / Actigraphy / pyActig	raphy Owner			★ 1 updated 2 days a	a Igo

mar.grignard@uliege.be

From an existing "local" folder:

"I want to put my existing code on Git"

<u>`</u> C	jit	ir	nit`

cd existing_folder	
git init	
git remote add origin git@gitlab.uliege.be:CyclotronResearchCent	re/LocalResources/Tutorials/Git-tutorial.git
git add .	L
git commit -m "Initial commit"	
git push -u origin master	



mar.grignard@uliege.be

- From an existing project:
 - "I want to use and/or collaborate to an existing piece of code."
 - `git clone`

git clone git@gitlab.uliege.be:CyclotronResearchCentre/LocalResources/Tutorials/Git-tutorial.git cd Git-tutorial touch README.md git add README.md git commit -m "add README" git push -u origin master

<u>gregory.hammad@uliege.be</u>

From an existing versioned `local` folder :

"I want to migrate from github.com/gitlab.giga to gitlab.uliege.be"

`git remote add`

cd existing_repo
git remote rename origin old-origin
git remote add origin git@gitlab.uliege.be:CyclotronResearchCentre/LocalResources/Tutorials/Git-tutorial.git
git push -u origin --all
git push -u origin --tags

git remote set-url origin git@gitlab.uliege.be :CyclotronResearchCentre/Studies/CogNap/Actigraphy/pyActigraphy-Tutorial.git

Or migrate the project via the interface...

mar.grignard@uliege.be

settings.

To only use CI/CD features for an external repository, choose **CI/CD for external repo**.

Tip: You can also create a project from the

command line. Show command

Migrating existing projects:

Projects									
New project									
A project is where you house your files	Blank proje	Blank project Create from template Z. Import project				CI/C	CI/CD for external repo		
(repository), plan your work (issues), and publish your documentation (wiki), among	Import project from	n							
other things.	😽 GitLab export	C GitHub	Bitbucket	♦ GitLab.com	G Google Coo	le 🕷 Fogbugz	i୍ଦି Gitea	git Repo by URL	
All features are enabled for blank projects,									
can disable them afterward in the project									

mar.grignard@uliege.be

- Save your modifications:
 - `git push`
 - update the remote repository with the local changes (i.e. commits).



Remote server

mar.grignard@uliege.be

- Update your working copy:
 - `git fetch; git merge` or `git pull`
 - apply changes recorded in the remote repository to your local copy



Remote server

14

mar.grignard@uliege.be

LET'S PRACTICE!

Exercise 1: `git push`

- Connect your local folder to an existing git repository:
 - git remote add origin git@gitlab.uliege.be:mygitrepo.git
- Add a file
- Commit the file
- Push the file to the remote server

LET'S PRACTICE!

Exercise 2: `git pull`

- Modify a file in the remote repository
- Fetch the corresponding commit
- Inspect the modifications
- Apply the changes to your local copy of the file
- Congratulate yourself!

mar.grignard@uliege.be

LET'S BRANCH! (DAVID BOWIE, 1983)

Branch:

- A series of commit snapshots from a given working directory.
- Create a branch: `git branch < branch-name > `
- Create a branch and check it out (ie. point that branch to your working dir): `git checkout -b <branch-name>`
- Push your branch to the remote server:
 - `git branch –set-upstream-to=origin/<branch-name>` + `git push`
 - or simply `git push -u origin <branch-name>
- Delete branch: `git branch -d <branch-name>



mar.grignard@uliege.be

LET'S BRANCH! (DAVID BOWIE, 1983)

Merge:

- Put the commit histories of two (or more) branches back together.
- `git checkout master`
- `git merge <branch-name>`
- Types of merging:
 - Fast-forward: the 2 branches did not diverge.
 - 3-way merging: there are commits that do appear in only one branch.
 - ▶ Good news; git choose what to do for you! Unless...



<u>gregory.hammad@uliege.be</u>

18

mar.grignard@uliege.be

LET'S PRACTICE!

Exercise 3

- Create your own branch and check it out:
 - git branch "NameOfYourBranch"
 - git checkout "NameOfYourBranch"
- Modify a local file and commit the modifications (cf previous Exo) to the new branch
- Switch back (i.e check out) to the main branch
- Merge the modifications from "NameOfYourBranch" onto the main branch
- Submit the resulting modifications to the remote server

mar.grignard@uliege.be

GIGA DS 2022: HANDS-ON TUTORIAL ON GIT

STRUCTURING YOUR PROJECT

- Git flow
 - https://nvie.com/posts/asuccessful-git-branching-model/
 - https://danielkummer.github.io/ git-flow-cheatsheet/
- Other ways...
 - https://www.atlassian.com/git/ tutorials/comparing-workflows



WHAT GITLAB CAN DO FOR YOU...

- Pull requests
- Issues
- CI/CD
 - Tests
 - Docs with Sphinx
 - Code coverage*
- Private code, public computer!?

*Not covered here

mar.grignard@uliege.be

- Pull requests (PR):
 - "Please, use my modifications"
 - Request the merge of a branch into another one (usually master).
 - Advantages:
 - Code review
 - Protection about unwanted changes
 - Nice interplay with issues

mar.grignard@uliege.be

- Typical workflow:
 - Create a "feature" branch locally
 - Publish it to the remote server (i.e set an upstream branch)
 - Work on your "feature"
 - Commit and push the modifications.
 - Merge the "feature" branch into the "master" branch via PR

Exercise 3

Local merge -> Push to remote

Exercise 3

Local merge -> Push to remote

- Create your own branch and check it out:
 - git branch "NameOfYourBranch"
 - git checkout "NameOfYourBranch"
- Modify a local file and commit the modifications (cf previous Exo) to the new branch
- Switch back (i.e check out) to the main branch
- Merge the modifications from "NameOfYourBranch" onto the main branch
- Submit the resulting modifications to the remote server

mar.grignard@uliege.be

Exercise 3

Local merge -> Push to remote

- Create your own branch and check it out:
 - git branch "NameOfYourBranch"
 - git checkout "NameOfYourBranch"
- Modify a local file and commit the modifications (cf previous Exo) to the new branch
- Switch back (i.e check out) to the main branch
- Merge the modifications from "NameOfYourBranch" onto the main branch
- Submit the resulting modifications to the remote server

mar.grignard@uliege.be

Exercise 3

Local merge -> Push to remote

- Create your own branch and check it out:
 - git branch "NameOfYourBranch"
 - git checkout "NameOfYourBranch"
- Modify a local file and commit the modifications (cf previous Exo) to the new branch
- Switch back (i.e check out) to the main branch
- Merge the modifications from "NameOfYourBranch" onto the main branch
- Submit the resulting modifications to the remote server

mar.grignard@uliege.be

Exercise 3

Local merge -> Push to remote

- Create your own branch and check it out:
 - git branch "NameOfYourBranch"
 - git checkout "NameOfYourBranch"
- Modify a local file and commit the modifications (cf previous Exo) to the new branch
- Switch back (i.e check out) to the main branch
- Merge the modifications from "NameOfYourBranch" onto the main branch
- Submit the resulting modifications to the remote server

Exercise 4

mar.grignard@uliege.be

<u>gregory.hammad@uliege.be</u>

Exercise 3

Local merge -> Push to remote

- Create your own branch and check it out:
 - git branch "NameOfYourBranch"
 - git checkout "NameOfYourBranch"
- Modify a local file and commit the modifications (cf previous Exo) to the new branch
- Switch back (i.e check out) to the main branch
- Merge the modifications from "NameOfYourBranch" onto the main branch
- Submit the resulting modifications to the remote server

Exercise 4

Push to remote -> remote merge

Exercise 3

Local merge -> Push to remote

- Create your own branch and check it out:
 - git branch "NameOfYourBranch"
 - git checkout "NameOfYourBranch"
- Modify a local file and commit the modifications (cf previous Exo) to the new branch
- Switch back (i.e check out) to the main branch
- Merge the modifications from "NameOfYourBranch" onto the main branch
- Submit the resulting modifications to the remote server

Exercise 4

Push to remote -> remote merge

- Create your own branch, track its modifications on the remote server and check it out:
 - git branch -set-upstream-to=origin/
 "NameOfYourBranch" "NameOfYourBranch"
 - git checkout "NameOfYourBranch"
- Modify a local file and commit the modifications (cf previous Exo) to the new branch
- Submit the resulting modifications to the remote server

Merge the modifications from "NameOfYourBranch" onto the main branch **directly on the remote server via a PR**

mar.grignard@uliege.be

GUI TO THE RESCUE

Demo



https://gitahead.github.io/gitahead.com/

mar.grignard@uliege.be

USER'S CORNER

Issues:



Report bugs

- Suggest improvements
 - New functionalities
 - Documentation...

mar.grignard@uliege.be

- Issue board "Kanban":
 - Prioritise and organise your developments

Development v Search or filter results		Edit board	Add list V Add issues
- Backlog 5 +	To Do 1 +	Doing 0 +	▼ Closed
AWD file header with space separated names 🚸 #8 bug	Detection of AonT/AoffT of activity/rest periods #10 enhancement To Do		Speed improvements #5 enhancement
Analyse week days #9 enhancement			
Daylight Saving Time (DST) #3			
Move helper functions to a dedicated file #4 enhancement			

mar.grignard@uliege.be

- Continuous integration/deployment:
 - Run a set of jobs (=pipeline), each time a commit is pushed
 - Mostly "test" jobs but not only (build, doc, ...)
 - Ensure the quality* of the code does not degrade...
 - Typically;
 - Protect "master" (<u>https://docs.gitlab.com/ee/user/project/protected_branches.html</u>)
 - Configure CI/CD jobs to run on develop
 - Use output of CI/CD jobs to validate Merge request from "develop" to "master"

mar.grignard@uliege.be

<u>gregory.hammad@uliege.be</u>*broad sense

🦊 GitLab Projects 🗸 Gi	roups ~ Activity Milestones Snippets + ~	This project	Search	٩	0)4	n 6		
P pyActigraphy	GIGA-CRC In Vivo Imaging > > Actigraphy > pyActigraphy > Merge Requests > !12		Todo		Add todo) »		
🔂 Project	Open Opened 6 minutes ago by 🍪 Grégory Hammad Edit	uest	Assignee					
Repository	Feature/docs							
() Issues 5				Milestone			Edit	
1 Merge Requests	Image: The second se	pranch P	•	Time track	king		Ø	
🤗 CI / CD				No estimat	te or time	e spent		
🗘 Operations	Pipeline #4 passed for effe7a22a on feature/docs		Labels					
🔳 Wiki						\+	Edit	
🔏 Snippets	No Approval required							
🏟 Settings	Merge Remove source branch Squash commits ³ Modify commit message			1 participa	nt			
	You can merge this merge request manually using the command line			Notificatio	ns			
				Reference	: Cyclotro	onResearch.		
	Discussion 0 Commits 5 Pipelines 4 Changes 17							

mar.grignard@uliege.be

- Continuous integration/deployment:
 - Jobs are run on:
 - Iocal ressources
 - shared "runners" (external machines)
 - Not configured yet in <u>gitlab.uliege.be</u>
 - Ongoing discussion with the SEGI...
 - Target: 31/01/2019

mar.grignard@uliege.be

- Continuous integration/deployment:
 - How to set up a pipeline?
 - Simply add a .gitlab-ci.yml file to your repository
 - Configure (or choose if available) a runner (i.e. a machine to run the pipeline jobs)
 - Project's Gitlab page > Settings > CI / CD

GIGA DS 2022: HANDS-ON TUTORIAL ON GIT

DEVELOPER'S CORNER

- Continuous integration/deployment:
 - Example:
 - Using a docker image
 - Define 3 jobs:
 - "build": ~ compilation test
 - "test": run a test suite
 - "deploy": create doc and make it available
 - What about Matlab? Use Octave!

mar.grignard@uliege.be

```
image: python:latest
            variables:
             PIP_CACHE_DIR: "$CI_PROJECT_DIR/.cache"
            cache:
             paths:
               - .cache/pip
               - venv/
            stages:
             – build
              - test
             - deploy
            before_script:
              - python -V
              – pip install virtualenv
              - virtualenv venv
             - source venv/bin/activate
            run:
              stage: build
              script:
              - pip install -e .
            test:
              stage: test
              script:
              - pip install pytest
              - python -m pytest -vv --disable-pytest-warnings
            pages:
              stage: deploy
              script:
              - pip install sphinx sphinx-bootstrap-theme
              - cd docs ; make html
              - mv build/html/ ../public/
              artifacts:
                paths:
                - public
              only:
gre
              - feature/docs
```

- Automatic documentation
 - Sphinx (<u>http://www.sphinx-doc.org/en/master/index.html</u>)
 - Extract informations directly from the source code:

reader.pv def read_raw(input_path, reader_type, n_jobs=1, prefer=None, verbose=0): """Reader function for multiple raw files. Parameters input_path: str Path to the files. Accept wild cards. E.g. '/path/to/my/files/*.csv' reader_type: str Reader type. Supported types: AWD (ActiWatch), MTN (MotionWatch8) and RPX (Respironics) n_jobs: int Number of CPU to use for parallel reading prefer: str Soft hint to choose the default backendself. Supported option: 'processes', 'threads'. See joblib package documentation for more info. Default is None. verbose: int Display a progress meter if set to a value > 0. Default is 0. Returns raw : list A list of instances of RawAWD, RawMTN or RawRPX <u>man.gngnarueunege.ve</u>

pyActign	aphy 0.1.dev0	Quick Start	Documentation	API	Examples	Source		Search	
pyActigra	phy.io.r	ead_r	aw						
pyActigraphy.io. r	ead_raw (input_p	oath, reader_t	ype, n_jobs=1, pr	efer=No	one, verbose	=0)			[source]
Parameters: • input_p • reader, • n_jobs; • prefer • verbos Returns: raw - Alls Return type: list	ath (str) – Path to the fil type (str) – Reader type int) – Number of CPU to str) – Soft hint to choose ((nt) – Display a progre t of instances of RawAW	es. Accept wild ca Supported types use for parallel n the default back ss meter if set to a /D, RawMTN or Ra	rds. E.g. '/path/to/my/ :: AWD (ActiWatch), M eading endself. Supported or a value > 0. Default is awRPX	/files/*.cs TN (Motio ption:'pro 0.	v/ onWatch8) and cesses', 'thread	RPX (Respironi s'. See joblib pa	cs) ackage documentation for more i	info. Default is None.	
© Copyright 2018-201 Created using Sphinx	8, Grégory Hamma 1.7.8.	d.							Back to top



- Automatic documentation
 - Advantages:
 - Catchy and user-friendly online documentation
 - Supports multiple languages (including Matlab: <u>https://</u> <u>pypi.org/project/sphinxcontrib-matlabdomain/</u>)
 - Caveats:
 - Quite complex file structure
 - Use reStructuredText directives (not Markdow...)

mar.grignard@uliege.be

- Private code but public computers? How to deploy code?
 - A not-so fictional example:
 - "I want to maintain up-to-date the code I put on the MRI computer (aka: cogent). For this, I cloned the project to the MRI computer and 'git pull' each time I need to update the code".
 - For this to work, you just need to install your private key on the cogent computer. Easy...
 - Wait a minute! Cogent is a public computer. Everyone will have access to your private key!!!

- Private code but public computers? How to deploy code?
 - Deploy keys to the rescue
 - https://docs.gitlab.com/ee/ssh/#global-shared-deploy-keys
 - In brief:
 - Generate a dedicated ssh key pair
 - Put the private on the "public" computer
 - Link the key to the project and assign it as "read only"!

Private code but public computers? How to deploy code?

₩ GitLab Projects ~ Groups ~ Activity	Milestones Snippets	Q ()]4	n C	🎯 ~
P pyActigraphy	GIGA-CRC In Vivo Imaging > > Actigraphy > pyActigraphy > Repository Settings			
🔂 Project				
Repository	Push Rules Push Rules outline what is accepted for this project.	Expand		
() Issues 5				
1 Merge Requests	Protected Branches	Expand		
🤗 CI/CD	Keep stable branches secure and force developers to use merge requests.			
G Operations	Protected Tags	Expand		
🗂 Wiki	Limit access to creating and updating tags.			
🔏 Snippets	Deploy Keys	Collance	12	
🏘 Settings	Deploy keys allow read-only or read-write (if enabled) access to your repository. Deploy keys can be used for CI, staging or	Collapse] 2.	
General	production servers. You can create a deploy key or add an existing one.			
Members	Create a new deploy key for this project			
Badges	Title			
Integrations				
Repository	Кеу			
CI / CD				

<u>mar.grignard@uliege.be</u>

BACK-UP SLIDES

INSTALL GIT ON WINDOWS

https://gitforwindows.org/

mar.grignard@uliege.be

GLOSSARY

Terms	Github	Gitlab
Directory	Repository	Project
	Pull request	Merge request

Ressources

http://git-school.github.io/visualizing-git/

mar.grignard@uliege.be