

HERD SW WORKSHOP

INTRO TO GITLAB

BEFORE WE START...

It is common practice for a piece of software to be under a Version Control System, to allow different people to efficiently collaborate on the same codebase together.

The VCS of choice for HerdSoftware is *git* (<https://git-scm.com>) and the whole codebase is hosted on a *gitlab* instance at RECAS.

What this is not:

- A tutorial on how to use *git*.

We assume all of you have at least a basic working knowledge on how to work with *git*. In case you need a cheatsheet here are a couple of useful links:

- <https://docs.gitlab.com/ee/gitlab-basics/start-using-git.html>
- <https://docs.gitlab.com/ee/gitlab-basics/command-line-commands.html>

What this is:

- An overview of what *gitlab* offers on top of *git* to facilitate and improve the experience of working together on a big software project

WHAT IS GITLAB?

Gitlab is a web-application designed to improve and enhance collaborative development on a piece of software.

- It allows people to interact with one or more *git* repositories by using a common and friendly web interface
- It extends the functionality of the repository with dedicated tools (that actually make life a lot easier...)
- It is currently used and deployed by:
 - CERN (gitlab.cern.ch)
 - INFN (baltig.infn.it)
 - RECAS (git.recas.ba.infn.it)

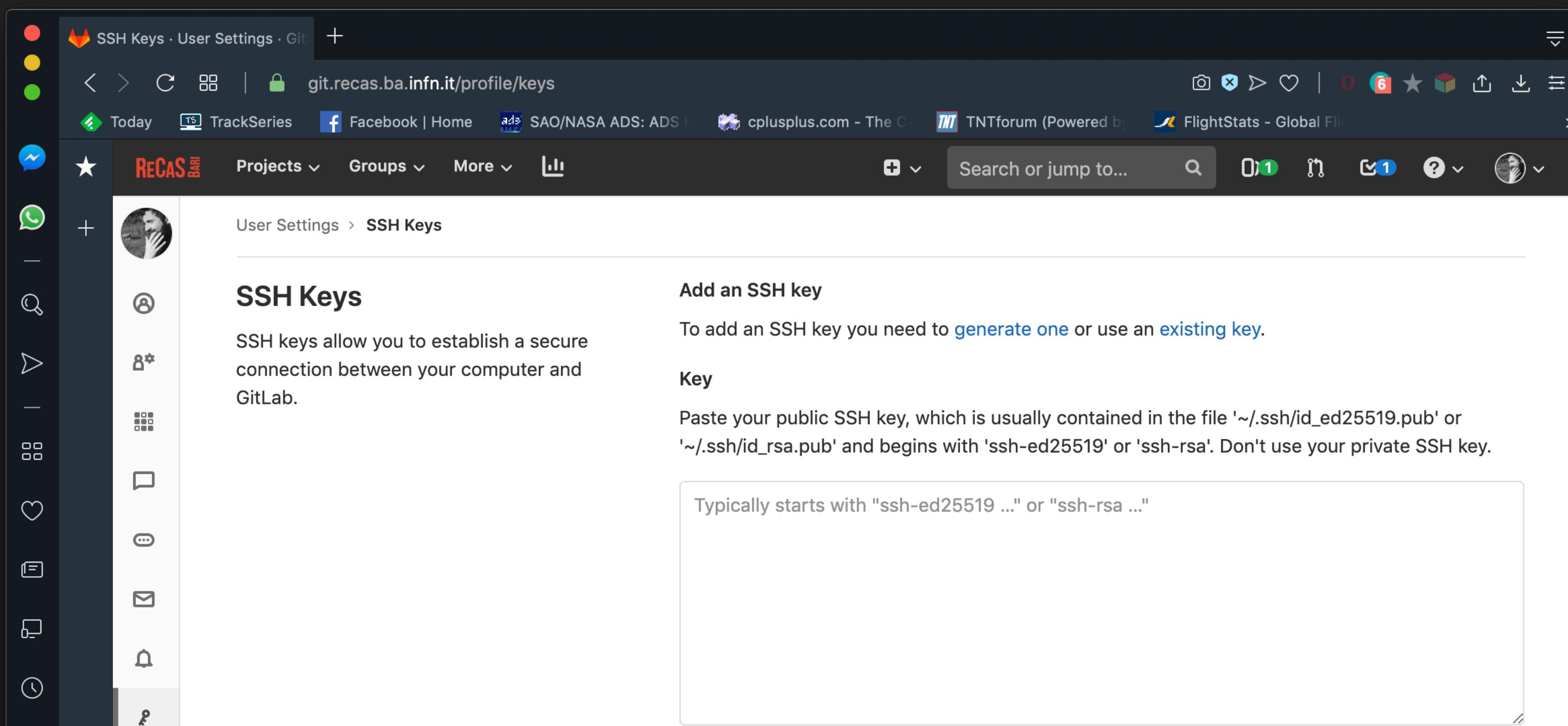
There are alternative projects that offer similar functionalities (*github*, *BitBucket*, etc...) but so far *gitlab* seems to be the most complete all-in-one solution.

Let's take a look at our instance... git.recas.ba.infn.it/herd

FIRST STEPS

My personal suggestion as a first step:

- Add your public SSH key to your *gitlab* profile. This will allow you to operate on the *git* repo from the command line without the need of constant authentication.
(if you don't have a *gitlab* account then obviously the first step is to create one)



EXAMPLE: ADD A NEW SSH KEY

GITLAB STRUCTURE

Every *git* repository in *gitlab* is associated to a **project** (<https://docs.gitlab.com/ee/user/project/>).

Projects encompass a *git* repo and offer additional tools around it:

- Issue tracker
- Merge requests
- Continuous integration
- Wiki documentation
- ...and many more

We'll touch briefly some of these tools, just for general knowledge, but in principle you won't have to deal directly with most of them.

The screenshot shows the GitLab web interface. The top navigation bar includes links for Today, TrackSeries, Facebook, SAO/NASA ADS, cplusplus.com, TNTforum, and FlightStats. The main content area displays the 'HerdSoftware' project details. The project summary shows 499 Commits, 5 Branches, 0 Tags, and 2.6 MB Files. A pipeline status is shown as 'running'. Below this, a commit history lists a recent update by Valerio Formato. At the bottom, there are buttons for CI/CD configuration, README, CHANGELOG, CONTRIBUTING, and Kubernetes cluster. A sidebar on the left contains various project management icons.

Name	Last commit	Last update
doc	doc: do not generate doxygen documentation...	8 months ago
dustbin	dustbin: add README.md and MCPrimaryPart...	1 day ago
examples	examples: Add missing files to Ex02	1 minute ago
include	examples: add some more functionality to Ex02	1 day ago
src	examples: add some more functionality to Ex02	1 day ago
testsuite	testsuite: fix and re-enable some tests.	1 day ago

GITLAB STRUCTURE

Several **projects** can be put together in the same **group** (<https://docs.gitlab.com/ee/user/group/>).

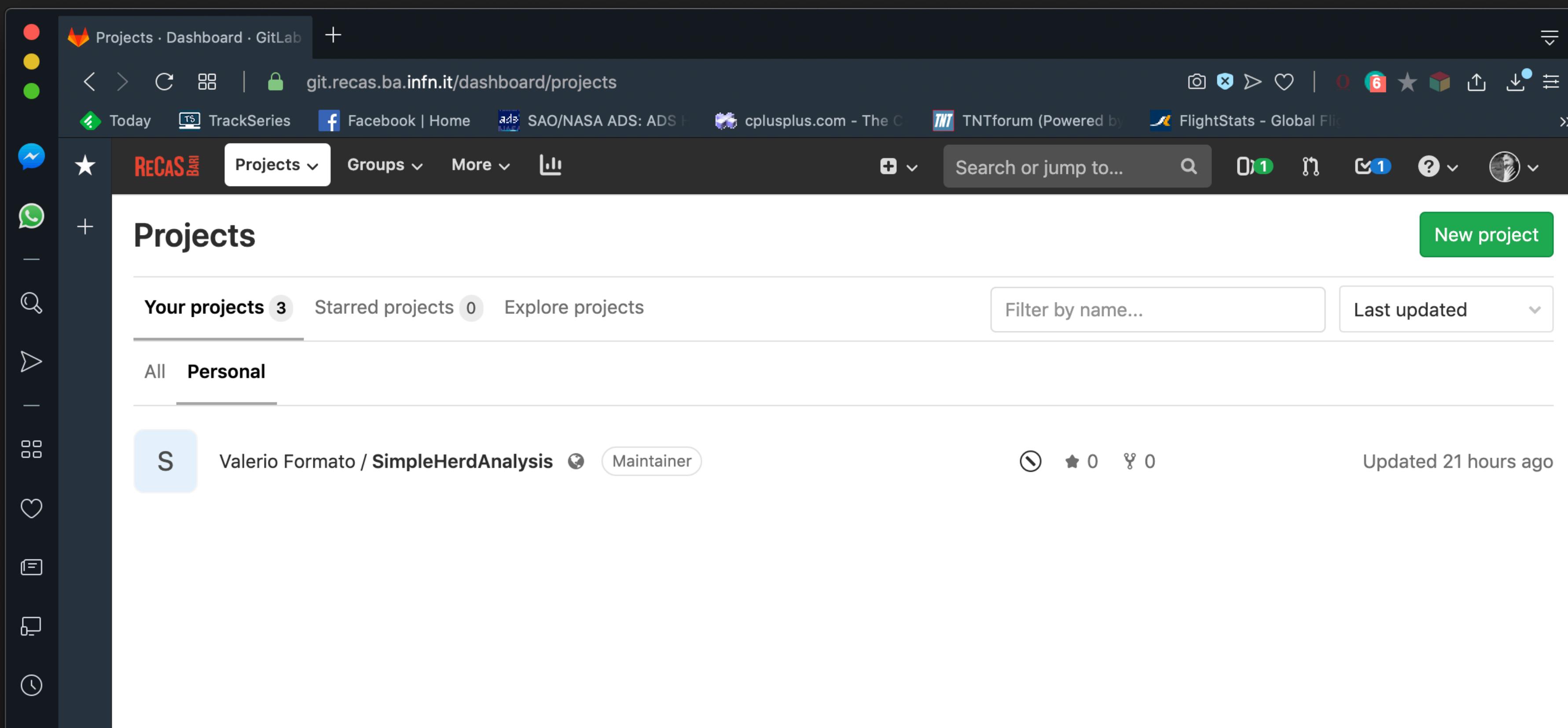
Groups are used for several different reasons, among which:

- Grant access to multiple projects and multiple team members in fewer steps by organizing related projects under the same namespace and adding members to the top-level group.
- Make it easier to mention all of your team at once in issues and merge requests by creating a group and including the appropriate members.

The screenshot shows the GitLab web interface. The top navigation bar includes links for Today, TrackSeries, Facebook, SAO/NASA ADS, cplusplus.com, TNTforum, and FlightStats. The main header shows the group name 'herd'. The left sidebar has a dark theme with various icons for navigation. The main content area displays the 'herd' group details, including its ID (17) and a 'Leave group' button. Below this, there's a section for 'herd experiment'. A tab bar at the bottom allows switching between Subgroups and projects, Shared projects, and Archived projects. Two projects are listed: 'herd-docker' (Repo for HERD docker images) and 'HerdSoftware'. Both projects have a lock icon and zero stars. The 'herd-docker' project was created 9 months ago, and 'HerdSoftware' was created 18 minutes ago. There are also search and filter options for the project list.

GITLAB STRUCTURE

You can also create **personal projects**, outside of any group scope. This is particularly useful if you want to work on your own personal analysis code and you don't need to share it with the whole group: just create a **project!**



EXAMPLE: CREATE A NEW PROJECT

EXAMPLE: CLONING HERD SOFTWARE

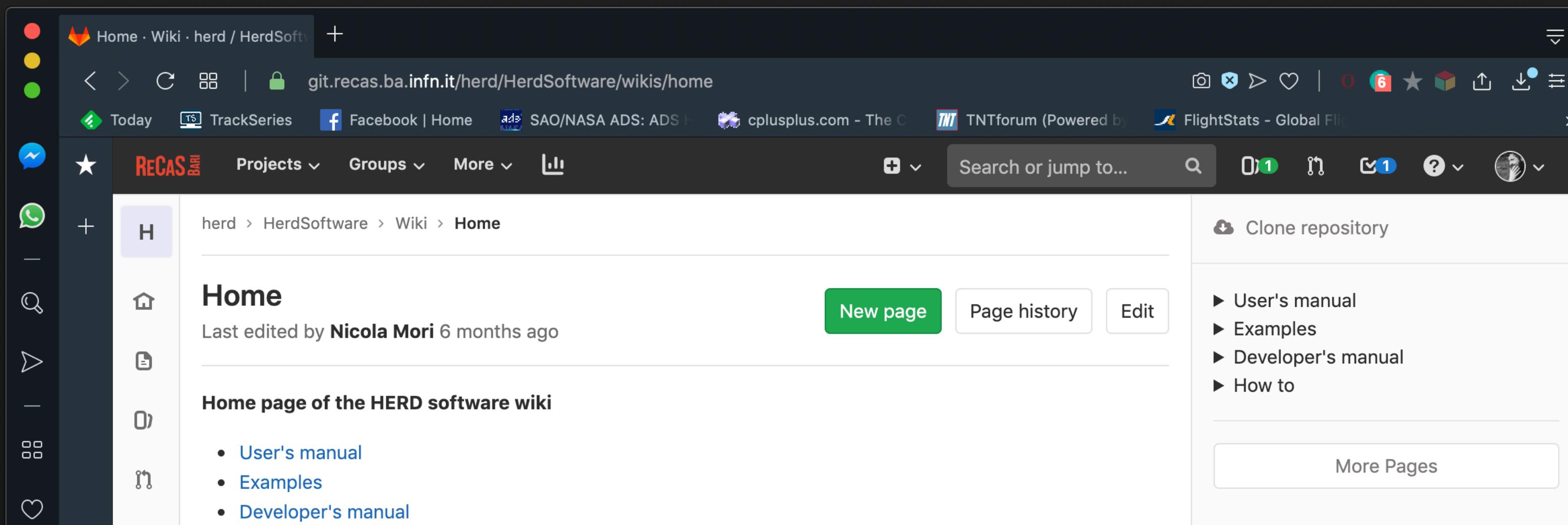
WIKI

A separate system for documentation called **Wiki**, is built right into each GitLab project. It is enabled by default on all new projects and you can find it under **Wiki** in your project.

Wikis are very convenient if you don't want to keep your documentation in your repository, but you do want to keep it in the same project where your code resides.

You can create **Wiki** pages in the web interface or locally using *git* since every **Wiki** is a separate *git* repository.

HerdSoftware comes with its own **Wiki** documentation that we try to keep always updated. You are encouraged to consult it whenever you have doubts about the software. If you think that some section is not described in enough detail, don't hesitate to write a mail or open an issue (see later...)



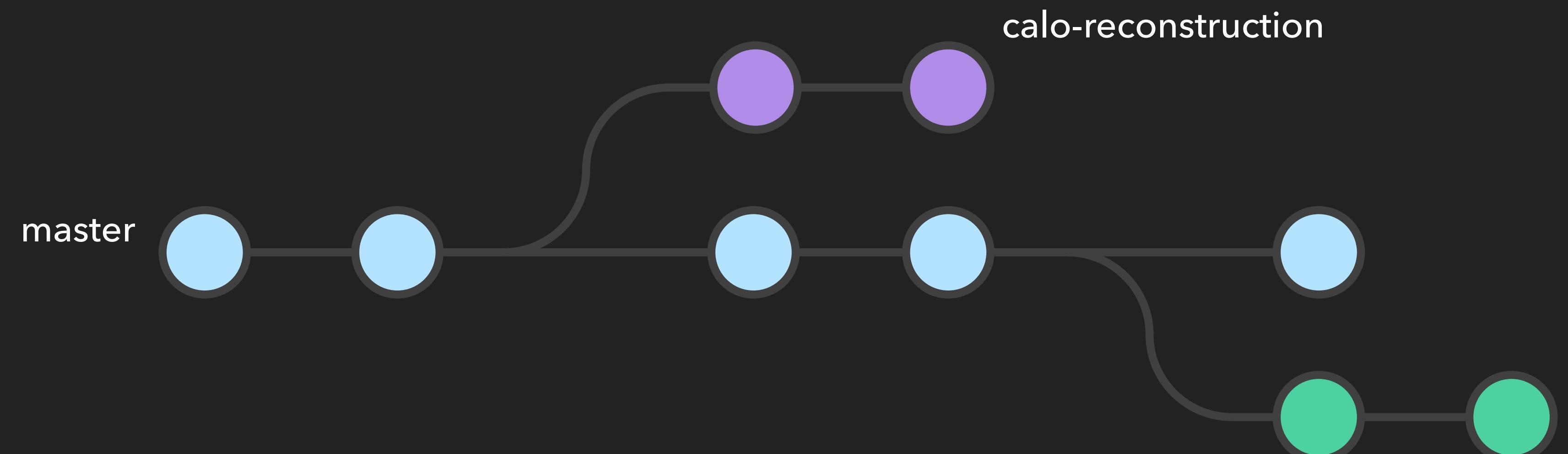
EXAMPLE: TAKING A LOOK AT THE WIKI

WORKING WITH BRANCHES

We expect that at the beginning the work will consist of both data analysis from the simulated files (which can be done in personal projects), as well as improvement and development of new reconstruction code in HerdSoftware.

In HerdSoftware the default branch is the *master* branch.

In order to allow everyone to work freely and keep the code in a nice clean and working state it is preferable to develop new features or fixes on dedicated branches and merge them into *master* only when the work is done.



Quick guide on how to work with branches:

<https://www.atlassian.com/zh/git/tutorials/using-branches>

PSF-study

CONTINUOUS INTEGRATION

Continuous Integration (CI - <https://about.gitlab.com/product/continuous-integration/>) is the practice of integrating code into a shared repository and building/testing each change automatically, as early as possible - usually several times a day.

Practically this translates in a set of checks and tests that will be automatically performed at each push and will alert the developer if any of these checks fails.

In HerdSoftware both the build process and the unit-tests are checked using the CI, over 4 possible environments (centos7, ubuntu18, ubuntu16 and OSX High Sierra)

The screenshot shows the GitLab interface for a project named 'herd / HerdSoftware'. The top navigation bar includes links for RECAS, Projects, Groups, More, and a search bar. On the left, there's a sidebar with various icons for navigation. The main content area displays a table of CI pipelines. The columns are: Status, Pipeline, Triggerer, Commit, Stages, and two timestamp columns. The table lists 10 recent pipeline runs, all of which have passed. Each row shows the pipeline ID, whether it's the latest, the triggerer (a bot icon), the commit hash and message, and the stages that passed (indicated by green checkmarks). The timestamps show the duration and time since the pipeline was run.

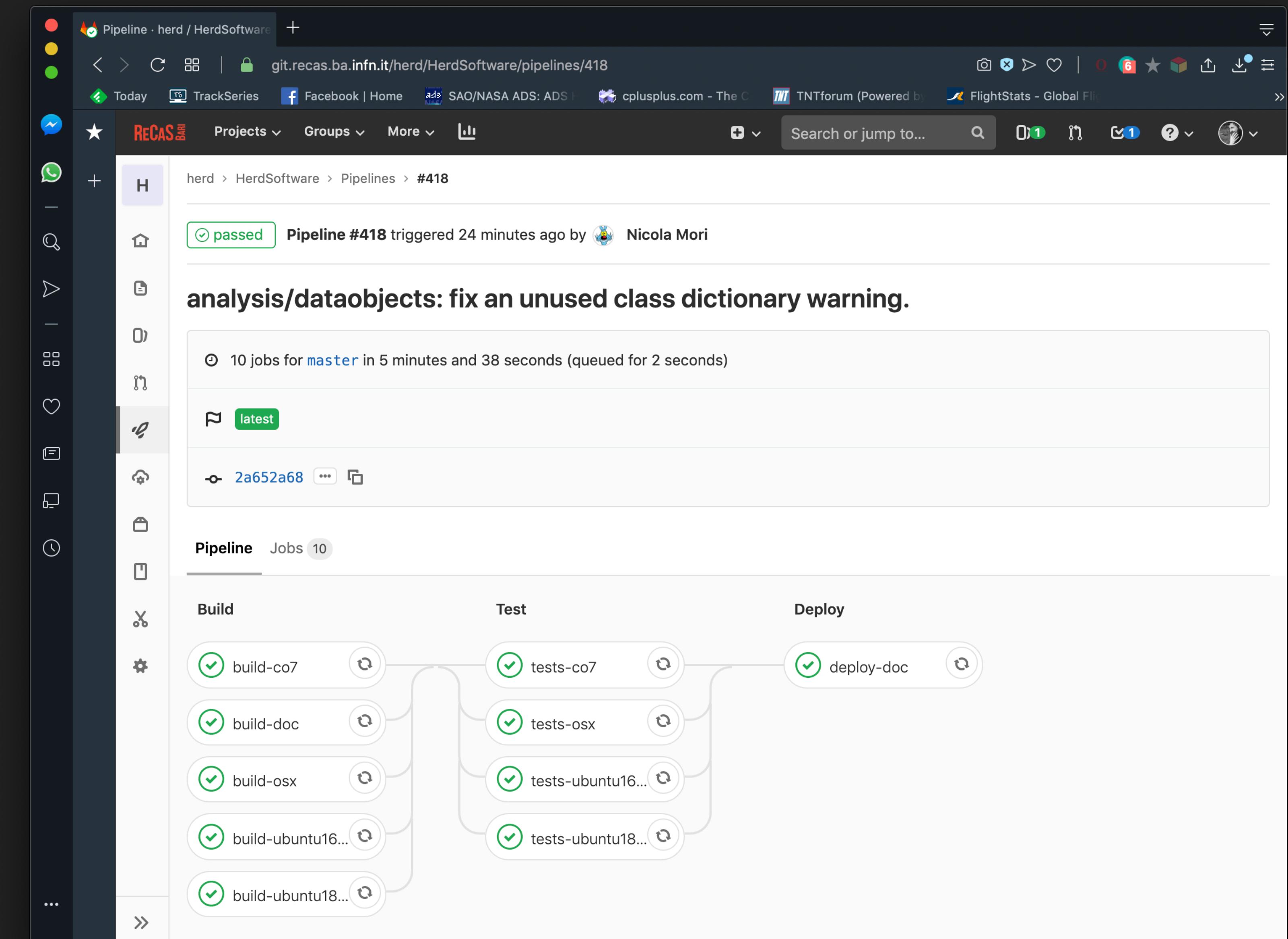
Status	Pipeline	Triggerer	Commit	Stages	Duration	Time Since
passed	#418 latest	bot	master → 2a652a68 analysis/dataobjects: fix ...	✓ ✓ ✓	00:05:38	19 minutes ago
passed	#416	bot	master → 802e436b Merge remote-tracking ...	✓ ✓ ✓	00:06:23	1 hour ago
passed	#415	bot	master → 1c126d11 examples: Add missing fi...	✓ ✓ ✓	00:05:44	1 hour ago
passed	#414	bot	master → 63eb0ded dustbin: add README.m...	✓ ✓ ✓	00:08:22	3 hours ago
passed	#412 latest	bot	81-check-an... → 63eb0ded dustbin: add README.m...	✓ ✓	00:09:44	3 hours ago
passed	#411	bot	81-check-an... → 8fcfd3899 testsuite: fix and re-ena...	✓ ✓	00:09:24	3 hours ago
passed	#410	bot	partial-mer... → c9fa9f53 examples: Fix Ex03	✓ ✓	00:07:57	3 hours ago
passed	#409	bot	84-fix-the... → 1a9772fb Merge branch 'Ex03' into ...	✓ ✓	00:18:23	3 hours ago

CONTINUOUS INTEGRATION

Continuous Integration (CI - <https://about.gitlab.com/product/continuous-integration/>) is the practice of integrating code into a shared repository and building/testing each change automatically, as early as possible - usually several times a day.

Practically this translates in a set of checks and tests that will be automatically performed at each push and will alert the developer if any of these checks fails.

In HerdSoftware both the build process and the unit-tests are checked using the CI, over 4 possible environments (centos7, ubuntu18, ubuntu16 and OSX High Sierra)



EXAMPLE: PUSH A NEW COMMIT AND TRIGGER THE PIPELINE

ADVANCED TOOLS: ISSUES

Issues are the fundamental medium for collaborating on ideas and planning work in GitLab.

In HerdSoftware they are used for:

- Report a bug
- Propose a new feature or work item
- Organise and split the work on big tasks between different developers

Each issue comes with a title and a description. Comments can be added as the work progresses. Gitlab can automatically create a dedicated branch (and a merge request) for a given issue. Issues can be labelled with custom with tags.

The screenshot shows a GitLab issue page for a project named 'herd'. The issue is titled 'Check and fix the test suite' and was opened 3 weeks ago by Nicola Mori. The description of the issue states: 'Fix any leftover breakage in the test suite.' There is one related branch labeled '81-check-and-fix-the-test-suite'. The issue has 0 likes and 0 dislikes. A comment from Nicola Mori 3 days ago discusses code testing for the initialization of STK geometry parameter objects. Another comment from Nicola Mori 3 days ago assigned her to the issue and added 'Doing' and 'Tests' labels. A final comment from Nicola Mori 3 days ago added the 'Doing' label and removed the 'To Do' label. The right side of the screen displays the issue's details: To Do (Add a To Do), Assignee (Nicola Mori), Milestone (Rework hits and geometry information), Time tracking (No estimate or time spent), Due date (None), Labels (Doing, Tests), Confidentiality (Not confidential), Lock issue (Unlocked), 1 participant, and Notifications.

ADVANCED TOOLS: ISSUES

In HerdSoftware they are used for:

- Report a bug
- Propose a new feature or work item
- Organise and split the work on big tasks between different developers

You can check/add new issues from the Issue list page.

The screenshot shows the RECAST web interface with the URL <git.recas.ba.infn.it/herd/HerdSoftware/issues>. The main content area displays the 'Issues' list for the 'HerdSoftware' project. The interface includes a sidebar with various icons for navigation and management. The main panel shows a list of issues with the following details:

Issue #	Title	Author	Status	Last Update
#90	Check the consistency between different geoParams and Hits	Lorenzo Pacini	Rework hits and geometry information Data objects To Do	updated 3 days ago
#89	Add tests for lateral STKs to StkGeoParams	Nicola Mori	Tests To Do	updated 4 days ago
#88	Define and set compilation flags for warnings	Nicola Mori	Build system To Do	updated 2 days ago
#87	Check PsdGeoParams test	Lorenzo Pacini	Tests To Do	updated 5 days ago
#86	Describe the software update procedure in the wiki	Nicola Mori	Documentation To Do	updated 6 days ago
#81	Check and fix the test suite	Nicola Mori	Rework hits and geometry information Doing Tests	updated 1 day ago
#80	Rework the detailed hits	Nicola Mori	Rework hits and geometry information Algorithms Data objects To Do	updated 3 weeks ago
#79	Rework the display algorithm for the new hits	Nicola Mori	Rework hits and geometry information Algorithms To Do	updated 3 hours ago
#75	Rework STK clustering and tracking for the new hits	Nicola Mori	Rework hits and geometry information Algorithms To Do	updated 3 hours ago

ADVANCED TOOLS: ISSUES

In HerdSoftware they are used for:

- Report a bug
- Propose a new feature or work item
- Organise and split the work on big tasks between different developers

You can check/add new issues from the Issue list page.

Or from the Issue board.

The screenshot shows the HerdSoftware project on the GitLab interface. The left sidebar includes icons for Issues (24), Boards, Labels, Milestones, Merge Requests (0), CI / CD, Operations, Packages, Wiki, Snippets, and Settings. The main area displays the 'Issue Boards' page for the 'Development' board. It features three columns: 'To Do' (23 issues), 'Bug' (2 issues), and 'Doing' (1 issue). The 'To Do' column contains tasks like 'Clean the horrible ArrayForwarder mess' and 'Document how to create dictionaries for data objects'. The 'Bug' column contains 'Fix display of digitized PSD hits (HERDward)'. The 'Doing' column contains 'Check and fix the test suite'. A search bar at the top right allows users to search or jump to specific issues.

ADVANCED TOOLS: ISSUES

In HerdSoftware they are used for:

- Report a bug
- Propose a new feature or work item
- Organise and split the work on big tasks between different developers

Issues can be grouped into Milestones, a collection of Issues all related to the same unit of work.

The screenshot shows the HerdSoftware project on GitLab. At the top, a browser window displays a milestone titled "Rework hits and geometry info" with a start date of Sep 19, 2019, and an end date of Oct 11, 2019. The milestone is 76% complete. Below the milestone, the "Rework hits and geometry information" page is shown, featuring a summary of 21 issues, 4 merge requests, 3 participants, and 10 labels. The issues are categorized into three groups: Unstarted Issues (open and unassigned), Ongoing Issues (open and assigned), and Completed Issues (closed). The completed issues include "Fix GGS bar volume ID" and "Rework the overall data model". The ongoing issue "Check and fix the test suite" is labeled with "Tests" and "Doing". The unstarted issue "Check the consistency between different geoParams and Hits" is labeled with "Data objects" and "To Do". Other issues listed include "Rework the detailed hits", "Rework the display algorithm for the new hits", "Update the wiki documentation", and "Rework STK clustering and tracking for the new hits". The sidebar on the left contains various project management icons.

ADVANCED TOOLS: MERGE REQUESTS

A Merge Request (MR) is the basis of GitLab as a code collaboration and version control platform. It is as simple as the name implies: a request to merge one branch into another.

- Compare the changes between two branches
- Review and discuss the proposed modifications inline
- Build, test, and deploy your code in a per-branch basis with built-in GitLab CI/CD
- Automatically close the issue(s) that originated the implementation proposed in the merge request
- Organize your issues and merge requests consistently throughout the project with labels
- Resolve merge conflicts from the UI
- Enable fast-forward merge requests

The screenshot shows a GitLab Merge Request page for a project named 'herd'. The merge request is titled 'Ex02 improve' and was opened 2 weeks ago by 'Matteo Duranti'. It is currently in a 'Merged' state. The description of the merge request states: 'Now the two sub-examples (RootScript and EventAnalysisJob) are coherent between themselves and have, each, a 'dedicated' directory. README (and wiki) improved accordingly'. Below the description, there is a section titled 'Request to merge Ex02_improve into master' which includes a pipeline status card. The pipeline card shows that Pipeline #356 passed for commit 6b616cce on the Ex02_improve branch, with a coverage of 89.50%. At the bottom of the merge request page, there is a note stating 'The changes were merged into master with 1a9772fb' and 'The source branch has been deleted'. On the right side of the page, there are various settings and status indicators for the merge request, such as 'To Do', 'Assignees', 'Milestone', 'Time tracking', 'Labels', 'Lock merge request', 'Participants', 'Notifications', and a reference link 'Reference: herd/HerdSoftware!6'.

ADVANCED TOOLS: MERGE REQUESTS

A Merge Request (MR) is the basis of GitLab as a code collaboration and version control platform. It is as simple as the name implies: a request to merge one branch into another.

- Compare the changes between two branches
- Review and discuss the proposed modifications inline
- Build, test, and deploy your code in a per-branch basis with built-in GitLab CI/CD
- Automatically close the issue(s) that originated the implementation proposed in the merge request
- Organize your issues and merge requests consistently throughout the project with labels
- Resolve merge conflicts from the UI
- Enable fast-forward merge requests

