Using CASToR Gitlab interface

October 22, 2024

Foreword

This document just describes the configuration of CASToR Gitlab platform, and some basic guidances for developpers regarding how to use it.

1 CASToR-collaboration

CASToR's Gitlab group is reacheable at this address https://gitlab.com/castor-collaboration. It currently contains the software distribution as the CASToR project, and another project (CASToR Tools) dedicated to regroup miscellaneous programs and scripts related to the use of CASToR.

2 CASToR project

The first step for a new member is to download the project, e.g using the following command with ssh: `git clone git@gitlab.com:castor-collaboration/castor.git`

The clone button at the right side provides different way to clone the project (fig. 1, blue square). The contents of the different branches are browsable using the button on the left (fig. 1, green square). The develop branch is the default branch. There are currently different categories of branch in the project:

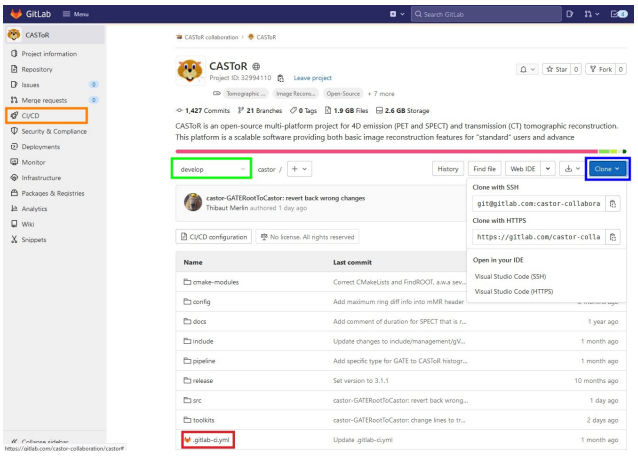

Figure 1: Gitlab CASToR project main interface

master: Branch dedicated to host the official CASToR versions. No one except main CASToR developers can push into this branch.

develop: The main CASToR develop branch, dedicated to gather all completed developments which will be released at some point in the master branch. The development of a new feature or the correction of a bug must not be performed directly in this branch, but in a specific branch

created from this branch. Once the work is done, the developer can ask for a merge request into the develop branch. In the current configuration, only the Maintainers (main CASToR developers) can validate the request. This branch also contains a pipeline (see section 3, red square) to check whether new modifications unpredictedly affect other parts of the code. For each push or merge request, the modifications will be checked by the pipeline. If it is expected that a new version alters some parts of the code, the pipeline must be modified accordingly.

feature/xxx: Branches dedicated to new features. Once the implementation is done, it must be merged into the develop branch.

hotfix/xxx: Branches dedicated to bug corrections. Once the implementation is done, it must be merged into the develop branch.

3 Pipeline

The pipeline aims at checking different parts of the code after a modification. It checks the compilation of CASToR in different configurations and perform several image reconstruction jobs using various methods and algorithms across different modalities. The pipeline is described in the *.gitlab-ci.yml* file (fig. 1, red square). The main image (docker container) for most tests is currently an ubuntu 20.04 system with root 6.22, from *rootproject*.

Compilation and reconstruction tests are performed in parallel. Reconstruction tests start once the main compilation test is performed. The status of the pipeline during and after testing can be monitored from the CI/CD tab (fig 1, orange square). If a pipeline fails, the log can be accessed by clicking on the related job (fig 2). The current pipeline takes about 20 minutes to complete. Current checks are listed below:

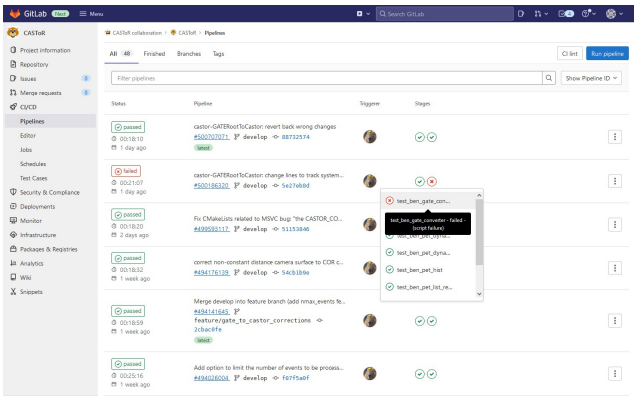

Figure 2: Gitlab CASToR project main interface

BUILD TESTS:

build_main_mk: Main compilation using makefile. The resulting job artifacts (*i.e.* compiled binaries) will be used for image reconstruction tests.

build_alt_mk: Makefile compilation with alternative data types (FLTNB/FLTNBDATA/FLTNBLUT: double, INTNB:int64_t).

build_cmk: Cmake compilation.

build_osx_mk: Makefile compilation on osx system.

JOB TESTS:

test_ben_ct: CT recon benchmark.

test_ben_spect: SPECT recon benchmark.

test_ben_pet_hist: PET histogram recon benchmark.

test_ben_pet_list_reco: PET list-mode recon benchmark with pre-computed sensitivity image.

test_ben_pet_list_sens: PET list-mode sensitivity image computation.

test_ben_pet_dynamic: PET 4D reconstruction (kinetic model) example.

test_ben_pet_dynamic_rigid-mot: PET 4D reconstruction (rigid motion correction) example.

test_ben_gate_converter: Various GATE data conversion tests (geometry and root datafile conversions).

4 Members and branch configuration

Members of Gitlab projects can have different roles, i.e *Owner/Maintainer*, *Developer*, *Reporter* and *Guest* in decreasing permission order. *Developer* should be the standard role for new members, which allows to create new branches and merge request. Main CASToR developers have *Maintainer* status, which grants the possibility to directly edit files in develop and master branches. Details about each role's permissions are described in the following link: https://docs.gitlab.com/ee/user/permissions.html.

Every old branches from CASToR previous private git repository are restricted to *Maintainer* rights only. In other words, a *Developer* only has read access to these branches, contrary to the branches he/she created by him/herself.

5 Discourse forum

All CASToR users are strongly encouraged to share their experience with CASToR by posting questions/answers, suggestions or new developments to the CASToR discourse forum. Note that technical support can be exclusively provided through the forum. To subscribe or browse the forum to search through any previous discussions, go to the following link:

`https://castor-project.discourse.group/`

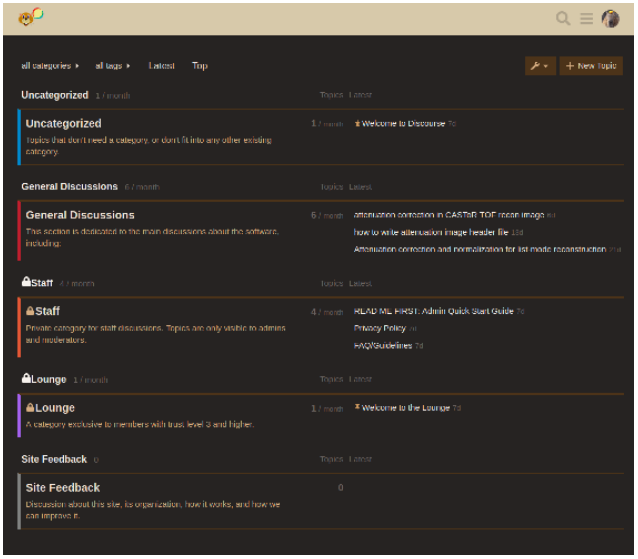

Figure 3: CASToR discourse forum

6 Mailing-List

The forum also acts as a mailing-list. Once you subscribed, head to your Preferences/Emails settings to enable the Enable mailing list mode parameter, and save changes.

Once it is done, you can directly create (or answer to) any topic by sending email to:

`castor_project+general-discussions-5@discoursemail.com`

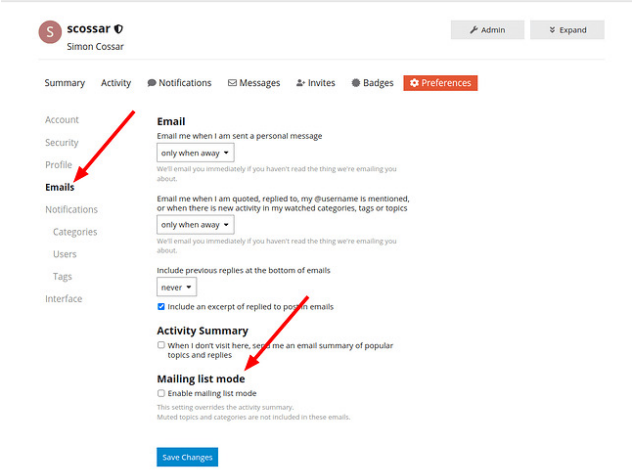

Figure 4: How to enable/disable mailing list

If you want to stop receiving email, you can then disable this feature from your profile settings.