{{lastupdated_at}} by {{lastupdated_by}}

GitHub repository workflow for maintainers

This page describes the integration workflow for maintainers. In summary, the following steps are to be executed:

1. Clone central GammaLib repository
2. Connect to developer's GitHub repository
3. Integrate feature branch
4. Verify the integration
5. Merge feature into devel

Clone central GammaLib repository

As first step, a clone of the central GammaLib repository is needed:

```
$ git clone https://manager@cta-git.irap.omp.eu/gammalib
  Cloning into 'gammalib'...
  Password:
  remote: Counting objects: 22150, done.
  remote: Compressing objects: 100% (7596/7596), done.
  remote: Total 22150 (delta 17330), reused 18491 (delta 14497)
  Receiving objects: 100% (22150/22150), 80.12 MiB | 192 KiB/s, done.
  Resolving deltas: 100% (17330/17330), done.
```

where manager is the user name of the integration manager.

Connect to developer's GitHub repository

Now connect to the GitHub repository of the developer using

```
$ cd gammalib
$ git remote add developer git://github.com/developer/gammalib.git
$ git remote -v
  developer  git://github.com/developer/gammalib.git (fetch)
  developer  git://github.com/developer/gammalib.git (push)
  origin     https://mamanger@cta-git.irap.omp.eu/gammalib (fetch)
  origin     https://mamanger@cta-git.irap.omp.eu/gammalib (push)
```

developer here is the GitHub user name of the developer from which we want to integrate changes.

Integrate feature branch

Fetch developer's feature branch

Fetch now the developer's repo, create a feature branch from the developer's feature branch, and switch into that branch:

```
$ git fetch developer
  remote: Counting objects: 4, done.
  remote: Compressing objects: 100% (1/1), done.
  remote: Total 3 (delta 1), reused 3 (delta 1)
  Unpacking objects: 100% (3/3), done.
  From git://github.com/developer/gammalib
   * [new branch]      007-my-new-feature -> developer/007-my-new-feature
   * [new branch]      devel              -> developer/devel
   * [new branch]      release            -> developer/release
  * [new branch]      integration        -> developer/integration
$ git branch 007-my-new-feature --track developer/007-my-new-feature
  Branch 007-my-new-feature set up to track remote branch 007-my-new-feature from developer.
$ git checkout 007-my-new-feature
  Switched to branch '007-my-new-feature'
```

Rebase or merge

If there are only a few commits, consider rebasing to upstream:

```
$ git fetch origin
$ git rebase origin/devel
```

If there are a longer series of related commits, consider a merge instead:

```
$ git merge --no-ff origin/devel
```

Note the --no-ff above. This forces git to make a merge commit, rather than doing a fast-forward, so that these set of commits branch off devel then rejoin the main history with a merge, rather than appearing to have been made directly on top of devel.

Check the history

Now, in either case, you should check that the history is sensible and you have the right commits:

```
$ git log --oneline --graph
$ git log -p origin/devel..
```

The first line above just shows the history in a compact way, with a text representation of the history graph. The second line shows the log of commits excluding those that can be reached from devel (origin/devel), and including those that can be reached from current HEAD (implied with the .. at the end). So, it shows the commits unique to this branch compared to devel. The -p option shows the diff for these commits in patch form.

Merge into integration branch

Now it's time to merge the feature branch in the integration branch:

```
$ git checkout integration
$ git merge 007-my-new-feature
  Updating ccba491..562f236
  Fast-forward
   my_new_file |    1 +
   1 files changed, 1 insertions(+), 0 deletions(-)
   create mode 100644 my_new_file
$ git commit -am 'Merged 007-my-new-feature.'
$ git push origin
  Password:
  Counting objects: 4, done.
  Delta compression using up to 4 threads.
  Compressing objects: 100% (2/2), done.
  Writing objects: 100% (3/3), 306 bytes, done.
  Total 3 (delta 1), reused 0 (delta 0)
  remote: To https://github.com/gammalib/gammalib.git
  remote:    ccba491..562f236  integration -> integration
  remote:  * [new branch]      github/integration -> github/integration
  To https://manager@cta-git.irap.omp.eu/gammalib
     ccba491..562f236  integration -> integration
```

Verify the integration

The push will automatically launch the integration pipeline on Jenkins.

You should verify the all checks are passed with success.

Merge feature into devel

Once the new feature is validated, merge the feature in the devel branch:

```
$ git checkout devel
  Switched to branch 'devel'
$ git merge integration
  Updating ccba491..562f236
  Fast-forward
   my_new_file |    1 +
   1 files changed, 1 insertions(+), 0 deletions(-)
   create mode 100644 my_new_file
$ git push origin
```