{{lastupdated_at}} by {{lastupdated_by}}

Git workflow

We are using git as version control system for the ctools code and GitLab to manage the git repositories.

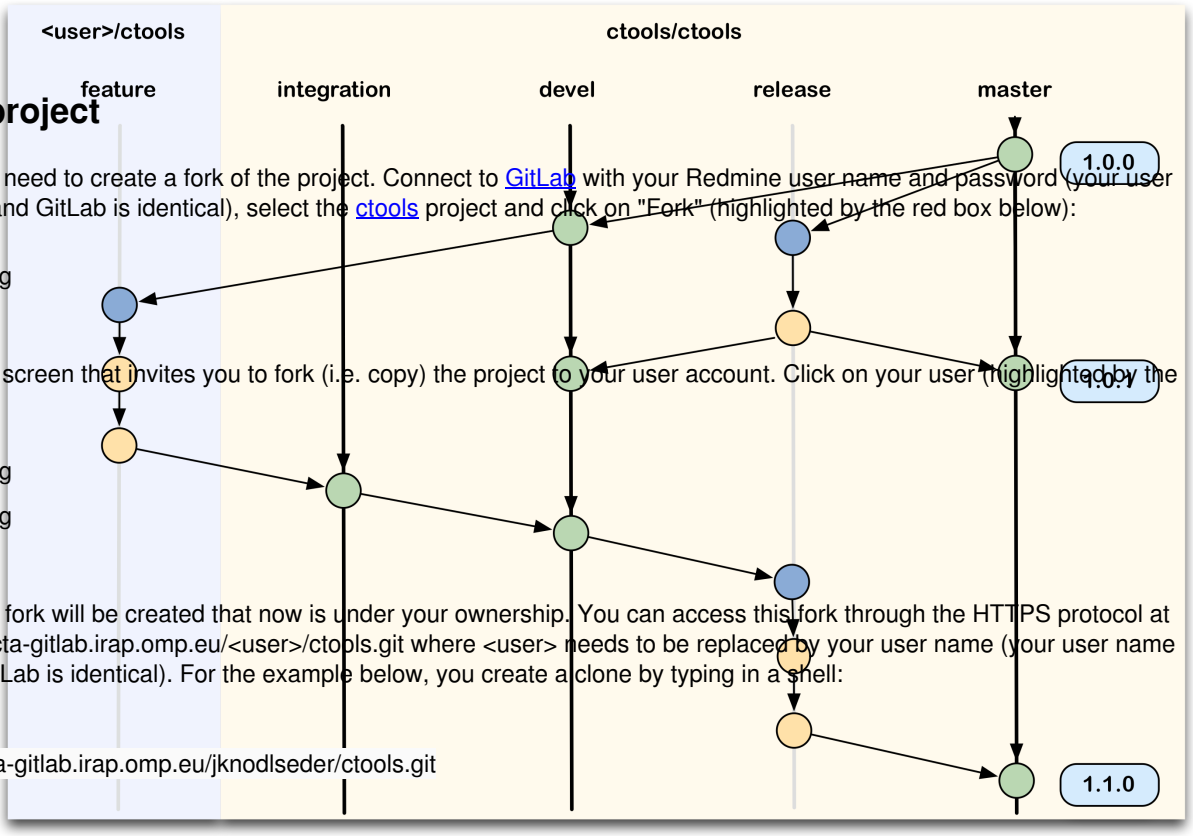
Every developer will have an own copy of the ctools code in the git repository, hence there is no need (and even no possibility) to push to the ctools repository. As a developer you will fork the ctools project, create a feature branch and add some new code (or correct a bug), and issue a pull request so that your change gets included in the trunk. This is identical to the workflow that you will follow when using GitHub.

Overview

The figure below illustrates the git workflow that we use for the ctools development. Permanent branches are shown as black lines, temporary branches as grey lines. There are three permanent branches in the central git repository:

- the master branch that holds the latest release
- the devel branch that is the trunk on which development progresses
- the integration branch that is used for code integration

A temporary release branch is used for hotfixes and generally for code testing prior to any release. As developer you will fork the ctools repository in your private GitLab area and work on temporary feature branches.



Forking the project

As the first step you need to create a fork of the project. Connect to GitLab with your Redmine user name and password (your user name on Redmine and GitLab is identical), select the ctools project and click on "Fork" (highlighted by the red box below):

gitlab-fork-step1.jpg

This brings you to a screen that invites you to fork (i.e. copy) the project to your user account. Click on your user (highlighted by the red box below):

gitlab-fork-step2.jpg
gitlab-fork-step3.jpg

After a short while a fork will be created that now is under your ownership. You can access this fork through the HTTPS protocol at the address https://cta-gitlab.irap.omp.eu/<user>/ctools.git where <user> needs to be replaced by your user name (your user name on Redmine and GitLab is identical). For the example below, you create a clone by typing in a shell:

```
$ git clone https://cta-gitlab.irap.omp.eu/jknodlseder/ctools.git
```

We recommend adding the main ctools repository as a remote repository named upstream so that you can fetch the latest code version before you start a new feature:

```
$ git remote add upstream https://cta-gitlab.irap.omp.eu/ctools/ctools.git
```

Modifying or adding code

To work on a feature or to correct a bug you step in the ctools directory that was created after cloning and you create a new feature branch using

```
$ git pull upstream devel
$ git checkout -b 9101-correct-nasty-bug
```

The first command pulled in the latest changes from the main ctools repository. Note that a branch name should always start with the number of the issue you work on and a short description what you actually are planning to do.

Suppose that there is a bug in the src/ctselect.cpp file. After correcting the bug you need to stage the change for commit using

```
$ git add src/ctselect.cpp
```

and then commit the change using

```
$ git commit -m "Fixed the nasty bug (#9101)"
```

where the message in hyphens should be as detailed as possible (the example here is definitely not very good). Please also attach the issue number in the commit message. This allows tracking which issues were in fact fixed by a given commit.

Now you can push your change into the git repository using

```
$ git push origin 9101-correct-nasty-bug
```

Note that the origin argument specifies where to push the change (origin means to the same repository where you have cloned from), and the 9101-correct-nasty-bug argument gives the name of the branch you want to push.

You can now verify on GitLab that a new branch exists in your project:

gitlab-push.jpg

Issuing a pull request

After having thoroughly checked your new code, and after having pushed the code into your git repository, you can now issue a pull request so that your change gets merged into the main code base (the so called "trunk"). For this you have to open the relevant issue in Redmine and put the status of the issue to "Pull request":

pull-request.jpg

In the notes field please indicate where your change is. This means you should provide at least your user name on GitLab and the name of the feature branch. You could also copy/paste the HTTPS URL of your repository.

You should also describe in the notes field the changes or addition you made to the code. Explain what you have done. Say if there is anything you'd like particular attention for - like a complicated change or some code you are not happy with.

If you don't think your request is ready to be merged, just say so in your pull request message. This is still a good way of getting some preliminary code review.

Managing your code

Delete a branch

To delete a branch on GitLab, select the "Branches" tab and click on the wastebasket behind the branch you want to delete:

gitlab-delete.jpg

Files

ctools-git-workflow.png	282 KB	12/05/2015	Knödlseder Jürgen