Table of Contents

Neutrino Platform Software Repository

git repositories

git is a distributed revision control and source code management system. It gains more popularity over SVN and CSV.

Documentation

If you have not used git before, you will need to customize your *git* environment. See here.

- _git_ Service main webpage
- CERN Knowledge base - *git* (internal only)

Web Tools

- CERN GITLAB - here you can create and manage *git* repositories

Access

The access to *git* repository is here

git commands

The basic commands are similar to SVN:

git	_svn_
git commit	svn commit

Follow this tutorial to learn about differences between *git* and *svn*.

HOWTO

How to fetch offline Core code and Applications from GIT repository ?

The entire project is available on gitlab - https://gitlab.cern.ch/NeutrinoPlatform.

Clone the desired repository (i.e Applications), using preferred auth method:

* https (password required on each operation):

```
git clone https://gitlab.cern.ch/NeutrinoPlatform/Applications.git
```

* ssh (need to have .ssh/id_rsa private key generated and public key uploaded to https://gitlab.cern.ch/profile/keys)

```
git clone ssh://git@gitlab.cern.ch:7999/NeutrinoPlatform/Applications.git
```

* kerberos

```
git clone https://:@gitlab.cern.ch:8443/NeutrinoPlatform/Applications.git
```

Checkout the desired branch:

```
cd Applications
```

```
git checkout xxxxx
```

Branch structure

Branches starting from release/ are production branches. At the moment we do not have such a structure

Each of them contains README.md file (presented on listed websites) describing its contents, environment setup and working configurations.

Users may create private branches that should start with prefix 'feature/' or 'bugfix/', depending on the type of changes. Their names should be descriptive, and once the development is finish, private branch may be merged into release and deleted.

Sample names:

- feature/add-new-reco-configuration-files
- bugfix/fix-wrong-alignment

Basic workflow

For an usual development we should checkout the release branch and synchronize with remote:

```
i.e. git checkout release/8.0.X
git pull
```

Do the changes:

```
# modify files
# check the changes with 'git status' or 'git diff'
git add <modified_file_path> # stage modified file for commit, alternatively stage all files with
```

Commit the changes:

```
git commit # this commands asks for commit message, the files get committed to local branch
# you may check 'git log' now
```

Once the changes are satisfactory, we may decide to make our commits public by pushing them to remote branch

```
git push # push all the commits to remote branch
git push -u origin master
```

From now on they will be visible to everyone and will appear in others' people local branches after executing "git pull".

Bigger changes

When working on bigger task, which can't be done during one day and needs to be integrated into code as a whole, it's recommended to create a dedicated branch. Therefore instead of woking on release branch directly, we create a bugfix or feature branch (depending on the nature of changes) using commands:

```
git branch bugfix/fix-wrong-alignment
git checkout bugfix/fix-wrong-alignment
git push --set-upstream origin bugfix/fix-wrong-alignment
```

The development process is similar, we can commit and push changes to the branch. If we have verified that the code is working in our branch, we may open a merge request to the appropriate release, so the code would be included into it.

All can be done on: https://gitlab.cern.ch/NeutrinoPlatform/Core/merge_requests. After creation, the changes should be reviewed (either by creator or someone else). When they got accepted, the code will be merged automatically if no conflicts occurred (if they weren't any other modifications of the same files in the meantime). Otherwise, conflicts must be resolved manually as described on the merge request website.

Such approach has an advantage that we can see exactly what has been developed and what is going into release, so it is possible to spot missing or excess elements and do the necessary cleaning.

Useful commands

Reset current repository to remote branch, deleting all local and unpushed changes:

```
git fetch # get the repository up to date
git reset --hard origin/release/8.0.X # reset to remote branch
git checkout release/8.0.X # checkout local branch
```

Back to CENF-Computing Main Page

Major updates:
-- NectarB - 21-Sept-2016

This topic: CENF > NeutPlatformSoftwareRepository
Topic revision: r4 - 2016-09-21 - NectarB