The openTELEMAC Git repository

Source code management for openTELEMAC

The openTELEMAC system source code is hosted on a GitLab server, available at the following address:**https://gitlab.pam-retd.fr/otm/telemac-mascaret**.

The Git repository allows users to stay connected to the updates and report issues, developers to submit and share their proposed changes and for the TELEMAC Consortium to ensure that the openTELEMAC system remains a professional and peer reviewed scientific code for real world environmental hydraulics applications.

This is the recommended method for downloading the source files.

About Git

Git is a tool used by many software developers to manage changes within their source code tree. Git provides the means to store not only the current version, but a record of all changes (and who made those changes) that have occurred to that source code. Use of Git is particularly common on projects with multiple developers, since Git ensures that changes made by one developer are not accidentally removed when another developer posts their changes to the source tree.

Required tools for access to the Git repository

First of all, to benefit from certain key features, you will need to install a recent version of Git (2.22.0 or over) as well as the Git LFS extension.

For Linux users

To install Git and Git LFS on Linux, you can do so through the package management tool that comes with your Linux distribution.

- If you are on Fedora (or any closely-related RPM-based distribution, such as RHEL or CentOS), you can use dnf:

```
$ sudo dnf install git-all
```

```
$ sudo dnf install git-lfs
```

- If you are on a Debian-based distribution, such as Ubuntu, try apt:

```
$ sudo apt install git-all
```

```
$ sudo apt install git-lfs
```

- For more options, there are instructions for installing on several different Linux distributions on the official Git website, at **https://git-scm.com/download/linux**.

For Windows users

You can download and install the latest “Git for Windows” from the official website at **https://gitforwindows.org/**. Make sure Git LFS is checked within the Components page, and override the default branch name to `main`.

Other settings include the following, from which you can choose:

- To use Git from the command line and also from 3rd party software (such as Git Extensions or other GUI client) and use the OpenSSL library;

- To check-out as-is (Linux) and commit Unix-style;

- To work directly within a DOS console windows;

- To please use the git pull --rebase option (see warning below for that command);
- To use the latest Git Credential Manager Core; and
- To enable file system caching.

GUI client

While it is better to learn Git through the command line, it is still recommended to use a GUI client, if only to have a better visibility of the commit history.

On Linux, we recommended **SmartGit**. It is free of charge when working with open source projects and can be installed in your HOME directory, if you don't have administrator rights.

On Windows, although SmartGit can be also be used, we recommend **Git Extensions**, which can be installed using a setup or as a portable application.

There are many other GUI clients, the most popular ones being **Sourcetree** (Windows only) and **GitKraken** (Windows, Linux and macOS), however, both require an online account to use them.

Details of the TELEMAC Git repository

As stated above, the repository is available at **https://gitlab.pam-retd.fr/otm/telemac-mascaret**. It provides a public access to the latest stable release of the openTELEMAC system as well as the current development version. As such, you do not need to be a registered developer with the TELEMAC Consortium to get access to the source code. Of course, we welcome all developments or suggestions and if you think you can make a difference, please contact us to be part of the registered developers.

Branches and tags

- The main development activity takes place in temporary branches. Such branches are a playground for major modifications that would otherwise render the principal version of the code unusable. After the goals of such isolated developments have been achieved, validated and peer reviewed, those temporary branches are being merged back with the main branch, through a GitLab **Merge Request** (MR).

- The main branch is the place where developments are centralized. Development activity in the openTELEMAC system varies depending on the time of year. During release periods, there are often many commits, sometimes just within a few days or hours. As a consequence, any snapshot of the main you may download, is a snapshot that is not regarded as being stable and usable. As such, it is not recommended to use the development version for any production use.

- Validated and peer reviewed releases are published officially as tags. They may be checked-out individually, mostly interesting to developers that like to compare official releases in their local development environment. The TELEMAC Consortium is responsible for approving validation of branches and for announcement of stable releases.

Cloning the TELEMAC repository

You have to “clone” a Git repository, and by default the entire repository is cloned. Options to limit the size of the cloned repository on your local system are available and provided below.

Complete clone

From a terminal:

```
$ git clone https://gitlab.pam-retd.fr/otm/telemac-mascaret.git
my_opentelemac

$ cd my_opentelemac
```

The clone will be made within the directory named “my_opentelemac”; you can name it as you wish.

Single branch clone

It is possible to clone a single branch as follows:

```
$ git clone --branch feature_branch --single-branch
```

```
https://gitlab.pam-retd.fr/otm/telemac-mascaret.git
my_opentelemac/feature_branch
```

The disadvantage of this process is that you can no longer recover another branch within "my_opentelemac" without doing a low-level manipulation on the local repository. Therefore, cloning a single branch is not recommended, unless you work in a separate directory for each branch.

Shallow clone

It is also possible to make a "shallow clone" which consists in limiting the history to a certain number of commits. For example, to retrieve the last 20 commits only:

```
$ git clone --depth 20 https://gitlab.pam-retd.fr/otm/telemac-mascaret.git
```

However, this method is not recommended either outside of a repository submodule or a continuous integration environment.

Partial clone

Finally, it is possible to make a "partial clone" by filtering the directories, which consists in recovering only part of the Git repository. This process requires a recent version of Git as well as the presence of a filter file on the system repository.

For TELEMAC, such file has been included on the repository that ignores the examples. Other filter files can be added upon request. The Git command is as follows:

```
$ git clone --sparse --filter=sparse:oid=main:.gitfilterspec
https://gitlab.pam-retd.fr/otm/telemac-mascaret.git my_opentelemac

$ cd my_opentelemac

$ git config --local core.sparsecheckout true

$ git show origin/main:.gitfilterspec » .git/info/sparse-checkout

$ git checkout main
```

The above list of commands is quite heavy as Git is not at all intended for partial clone, but it works.

Cloning the repository – GUI users

All GUI clients provide a clone command located under the main menu, for example in “Repository > Clone” for SmartGit, or “Start > Clone Repository…” for Git Extensions.

Most interfaces provide you with direct access to single branch and/or shallow clone. You can do a partial clone using the command line description above.

Building the openTELEMAC system after a clone

After the clone finished successfully you have a full-blown development version of the entire openTELEMAC system.
Before you can use it you must configure and build the system on your computer. Please see the corresponding article on how to do this.

Developer access to GitLab

People who requested a developer access should all have received an e-mail to access the GitLab server when their account was created.

If the identifiers are not necessary to clone the repository, since it is public, it is obviously not the same to modify it. Access rights are required through a two-factor authentication (2FA). Both authentication factors are required every time you log on to GitLab (https://gitlab.pam-retd.fr/), a web server maintained by EDF R&D and connected to its Git repositories, including the TELEMAC repository.

The first level authentication factor is your GitLab username and password. The second level factor is a code that is generated through an OTP (One-Time Password) application that you have to install on another device, such as your phone. Such applications include FreeOTP or Microsoft Authenticator, but many other are available for Android and iOS.

Fortunately, this OTP is only necessary when accessing the GitLab web page. But when working with Git, the access is stored locally on your system in the form of a personal “access token”.

To generate this token, you need first to connect to the GitLab web interface. On that occasion, you will need your GitLab username and password as well as the OTP code. Once logged on, the access token is available through “Edit profile”, selecting “Access Tokens”.

You must then generate the access token by filling in a name (of the application that uses Git, for instance) and making sure that you have checked “read_repository” and “write_repository” at least.

Then click on “Create personal access token” for the key to appear at the top of the page: note it down, copy/paste it, keep it, as you will lose it once you leave the web page.

For Linux users

In order to avoid entering the token each time you push to the repository, you need to enter the following command to store your credentials:

```
$ git config --global credential.helper store
```

Then, the token is to be written to `$HOME/.git-credentials`, in the following form:

`https://oauth2:<token>@gitlab.pam-retd.fr` (of course replacing <token> with your own token)

From here you can work.

For Windows users

Since Git Credential Manager Core should have been enabled through the Git setup, storing your credentials requires only to enter the token once and for all, for example when cloning the repository (or later on, when doing a push):

```
$ git clone
https://oauth2:<token>@gitlab.pam-retd.fr/otm/telemac-mascaret.git
my_opentelemac
```