git-demo

Nicolas Borboën nicolas.borboen@epfl.ch

2015-09-14

Contents

Chapter 1

Git-demo project

The aim of this repo is to provide a simple tutorial about git usage. You will need **git**[1] and a **merge tool**[2].

Define your git public informations

1. Set you username

   ```
   $ git config --global user.name "Your Name Comes Here"
   ```

2. Set you email

   ```
   $ git config --global user.email you@yourdomain.example.com
   ```

Basics commands

This section present all the “every day” git commands. How to clone a git repository locally, add files, commit changes, push to repository and pull from repository.

1. Create a projet on gitlab.epfl.ch / github.com or initialize a local project with the command

   ```
   $ git init
   ```

[1]Information about installing git on your system can be found here: https://git-scm.com/download/

[2]Many merge tool are available but Meld is a good multi-plateforme candidate

2. Clone the project on your computer

   ```
   $ git clone username@host:/path/to/repository
   ```

   ```
   $ git clone https://gitlab.epfl.ch/sti-it/git-demo.git
   ```

3. Edit the file foo.md
4. Commit your modification

   ```
   $ git commit -m "My modifications details" foo.md
   ```

5. Create a new file

   ```
   $ vim myTestFile.txt
   ```

6. Add the new file to git

   ```
   $ git add myTestFile.txt
   ```

   and commit it.
7. Check your modification

   ```
   $ git status
   ```

8. Push your modification

   ```
   $ git push
   ```

9. Browse to the repository page to see the modification
10. Others collaborators of the repository can now update their files to see your changes with

    ```
    $ git pull
    ```

Read the log

1. View all the logs

   ```
   git log
   ```

2. View the log of a specified author

   ```
   git log --author=bob
   ```

3. Test some logs option

   ```
   git log --pretty=oneline
   ```

   ```
   git log --graph --oneline --decorate --all
   ```

   ```
   git log --name-status
   ```

   ```
   git log --help
   ```

4. Note that you can use the following command to search for strings in any version of your project:

   ```
   $ git grep "hello"
   ```

Conflicts management

1. Reset changes you made to a file

   ```
   $ git checkout -- <filename>
   ```

2. Set the tool you want to use to resolve conflict

   ```
   $ git mergetool
   ```

3. Resets any changes to tracked files:

   ```
   $ git reset --hard origin/master
   ```

   Resets the index and working tree. Any changes to tracked files in the working tree since are discarded.

4. If you want to be more delicate than the latest command, you can use:

   ```
   $ git revert <commit>
   ```

   Reverting has important advantages over resetting as it doesn't change the project history, which makes it a "safe" operation for commits that have already been published to a shared repository.

Managing branches

1. Create a new branch

   ```
   $ git branch myTest
   ```

2. List the branches of your project

   ```
   $ git branch
   ```

   myTest master* (<– current branch)

3. Change the branch with the command

   ```
   $ git checkout myTest
   ```

4. Now edit the foo.md file and commit it, then change back on master branch

   ```
   $ git checkout master
   ```

   Check that the change you made is no longer visible, since it was made on the myText branch and you're back on the master branch.

5. Now you can edit files on the master branch, but at one point you may want to bring back the changes made on myTest to master branch:

   ```
   $ git merge myTest
   ```

   If the changes don't conflict, you're done. If there are conflicts, markers will be left in the problematic files showing the conflict.

6. Check the markers with the diff command:

   ```
   $ git diff
   ```

7. Now edit the file to resolve the conflicts and commit the changes made to the file. You can use the tool “gitk” to graphically see the resulting history.

   ```
   $ gitk
   ```

8. As the changes made to the myTest branch are now merged, you can delete this branch:

   ```
   $ git branch -d myTest
   ```

Cleaning up

1. Remove branches

   ```
   $ git branch
   ```

   and

   ```
   $ git branch -D <every branches but master>
   ```

2. Revert the changes made to foo.md and commit them

   ```
   $ git reset fd0dec5 foo.md
   ```

Documentation and links

RTFM

- `man git`
- `man 7 gittutorial`
- `man 7 gitworkflows`

On the www

Official

- Documentation
- Books
- Cheat Sheets: EN - FR

Good alternative documentation

- Roger Dudler
- Atlassian
- gitmagic

Courses

- GitHub
- CodeSchool
- Git Immersion

Documentation in French

- OpenClassRooms
- Polytechnique.fr
- gitmagic
- PutainDeCode
- Submodules vs Subtree

Facts

- Kernel SCM saga…
- Happiness is a warm SCM