to **create** a new branch called release

```
$ git checkout  -b release
$ git push origin release
```

---

to **switch** to another existing branch

```
$ git checkout release
```

---

to **list** all branches

```
$ git branch -a
```

---

to **remove** the *release* local branch from your machine:

```
$ git branch -d release
```

---

to **remove** the *release* remote branch:

```
$ git push origin :release
```

To **revert** to previous git commit

```
$ git log
```

```
commit d9d3978fdbe6791577dc1e9407ac2587eac19d6a
Author: Ivano Calabrese <xxx@gmail.com>
Date:   Mon Mar 24 16:58:41 2014 +0100

    some changes to the readme.md file

commit e6c65b22d704dd9377434d6bbfd30f78070f971b
Author: Federico Favaro <xxx@gmail.com>
Date:   Mon Mar 24 16:18:32 2014 +0100

    first commit on private repo

commit 456ba4e6154bd661e946a8141ae7ce284a8e1733
Author: Underwater projects <uwsignet@users.noreply.github.com>
Date:   Mon Mar 24 07:37:56 2014 -0700

    Initial commit
```

and then if you want the Initial commit, you have to type the following command:

```
$ git checkout 456ba4e6154bd661e946a8141ae7ce284a8e1733

Note: checking out '456ba4e6154bd661e946a8141ae7ce284a8e1733'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:
```

```
  git checkout -b new_branch_name

HEAD is now at 456ba4e... Initial commit
```

To **clean** the working copy from uncommitted files (usually compilation files)

`$ git log -n -d <path>` let you see which files would be erased, without erasing them, to check that you are not erasing important files

`$ git log -f <path>` erase actually all the files not committed

To **see** local history of your modifications on git repo

`$ git reflog` let you see the local history of your modifications on your actual git repo locally, such as commits, merges, etc.

To delete a commit

`$ git reset --hard HEAD~n` let you to come back to the previous n-th commit. (for example, HEAD~2 let you come back of 2 commits. Pay attention that merge of 2 commits **is not counted as a commit**. Please note that also the SHA1 instead of HEAD~n is ok

---

To return to a previous commit

`$ git merge <SHA1>` let you go to a specific commit specified by its SHA1. The SHA1 can be easily retrieved using `git log`.

To select which file imports from a branch using a checkout

`$ git checkout -p master` let you see which files are changed between the current branch (for example release) and master and which files are completely new. For each one of these files git ask you what to do (add, modify, etc.). Nothing is done for unchanged files.

Tools for a new release

You can use the following guide to do a new release (thanks to Ivano)

1. Merge to the master all your modifications, commit and push them on master branch.
2. From the folder **releaseTools** copy *config* into your **.git** folder.

3. From the folder **releaseTools** copy *pre-commit* script into **.git/hooks/**
4. Check that *pre-commit* has the proper rights to be executed.
5. Check the file *desert.gitignore* in order to see if something private has to be released. In that case delete the correspondent line. On the contrary, if you want to something to be kept private and it is not listed on the file, please add it on the file.
6. Commit and push modifications on *desert.gitignore*
7. Go to the **release** branch using `$ git checkout release`
8. Import the modifications (new files, modifications on files, ….) using
    a. `$ git checkout master desert.gitignore`
    b. `$ git checkout master <file1>` (do it on the same folder where the new file should go)
9. Commit your changes giving a proper name to the commit (e.g. “release desert-x.y.z”)
    a. At this moment the *pre-commit* script should do its work, cleaning all the private files, you can check from the standard out if all the private files has been deleted.
10. Tag your new commit using `$ git tag <label> [usually desert_underwater-x.y.z]`
11. Push your commit using `$ git push origin release`
12. Push your tag using `$ git push origin tag <label>`
13. Push the last commit on the public repo using
    a. `$ git push github_pub_release release:master`
14. Push the last tag on the public repo using
    a. `$ git push github_pub_release tag <label>`

How to change name/mail address of the last commit

The commit I want to change is the latest in history, so I’ll use the --amend directive to make my changes. Putting all things together, our final command becomes

```
$ git commit --amend --author 'new_name <mail_name@provider.com>'
```

How to merge a branch into master branch

```
git checkout master
git merge <branch>
git commit -m “the merge has been done”
git push origin master
```