Git tips

Some tips to use `Git`.

Summary

I. Git tips

1. Add remote repository on a project and how to use it

It's common to fork a project to contribute on it. However, it can be problematic if you want your master branch (or any other) to be up-to-date since your fork won't update automatically. That's where remote urls get useful.

Add a remote url

Nothing fancy or complicated in here. Once you have your remote url, just add it like this:

```
git remote add the_remote_name your_url
```

Of course, you need to replace "the_remote_name" by the name you want to give to the remote and "your_url" by the url of the remote repository. For example, I want to add a remote url on rust repository:

```
git remote add upstream https://github.com/rust-lang/rust
```

How to use it

Now if I want to update my `master` branch with the original rust repository:

```
git checkout master # to be sure we're on my local master branch
git fetch upstream # we download the lastest changes on the original rust repository
git rebase -i upstream/master # I update my local branch with "upstream" remote url and with its
```

And we're all good. If you want to push changes to a remote repository:

```
git push some_remote_repository HEAD:the_branch_I_want_to_push_on
```

Please note that this is **ALWAYS** better to provide all information to the `git push` command to avoid erroneous pushes.

2. Squash commits

In case you're contributing to a project or just want to do some cleanup in your project commits' history, knowing how to squash is really important.

First step

First, you need to know how many commits you want to squash and where they're located in the commit history. In here, I'll show you the case where you want to squash the lastest commits of your history.

For example:

```
> git log
commit 04fa3a3d24e5dc35dc6ab518dac413e9398bc998
Author: Guillaume Gomez <guillaume1.gomez@gmail.com>
Date:   Sun Jul 31 02:17:50 2016 +0200

    Fixup for pk-config#3

commit 04fa3a3d24e78935dc622518dac413e9398bc768
Author: Guillaume Gomez <guillaume1.gomez@gmail.com>
Date:   Sun Jul 31 02:16:59 2016 +0200

    Fixup for pk-config#2

commit 04fa3a3d24e5dc35dc622518dac413e9398bc989
Author: Guillaume Gomez <guillaume1.gomez@gmail.com>
Date:   Sun Jul 31 02:16:27 2016 +0200

    Fixup for pk-config

commit 0cc88a7d344c22e6a61dc2b67be6c63bdc6cb73a
Author: Guillaume Gomez <guillaume1.gomez@gmail.com>
Date:   Sun Jul 31 01:38:34 2016 +0200

    Add pkg-config to help looking for libs
```

In here, the 3 last commits (the fixup ones) are too much. We don't want to keep them. So let's squash them into "Add pkg-config to help looking for libs".

git rebase

To remove them from the history, we'll use the `git rebase` command:

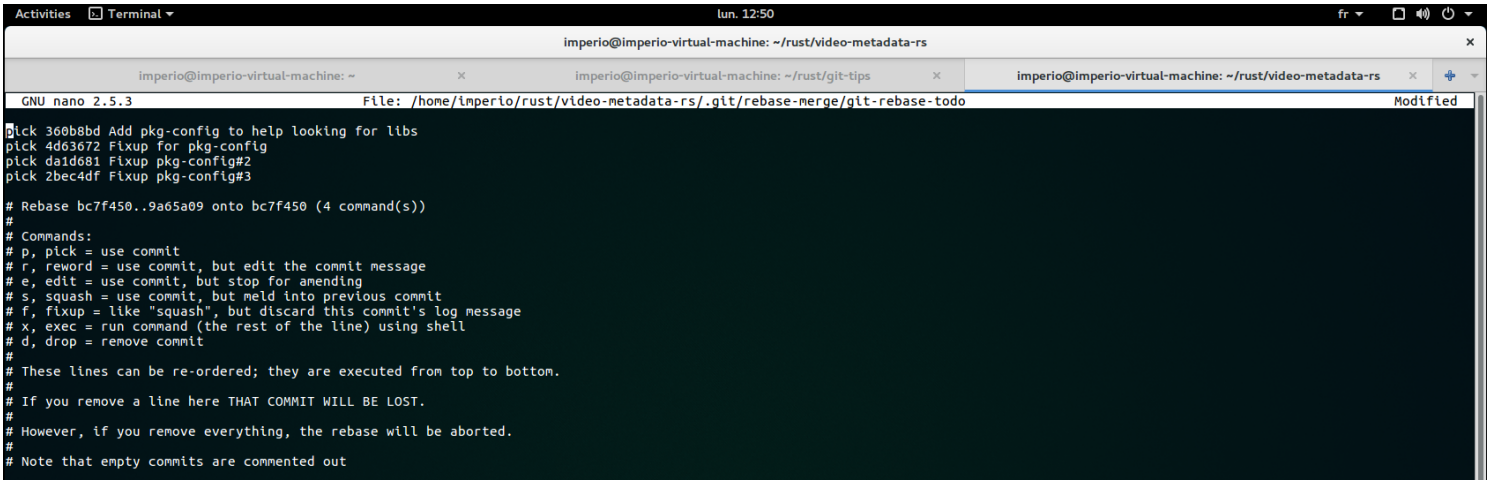
```
git rebase -i HEAD~4
```

Let me explain what I did first:

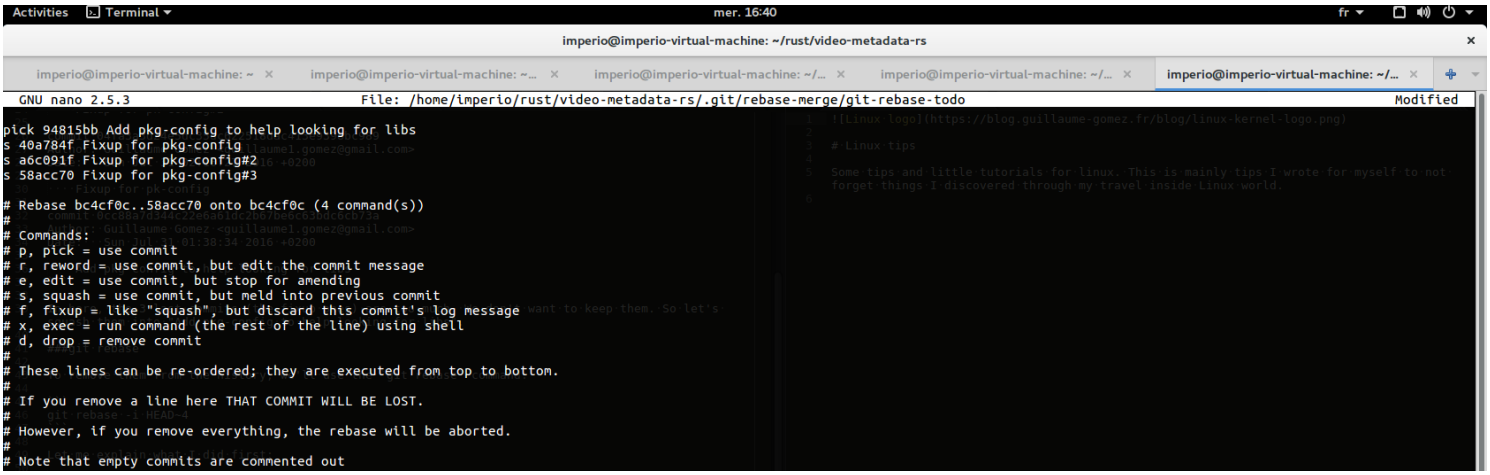
- `-i` is the interactive option.
- `HEAD~4` means (to put it simply) "I want to edit the last four commits starting from the HEAD (which is the last commit of the current branch)".

If you wonder "why 4?", it's because we need to squash all the commits into another one. So `3 we want to remove + 1 = 4`.

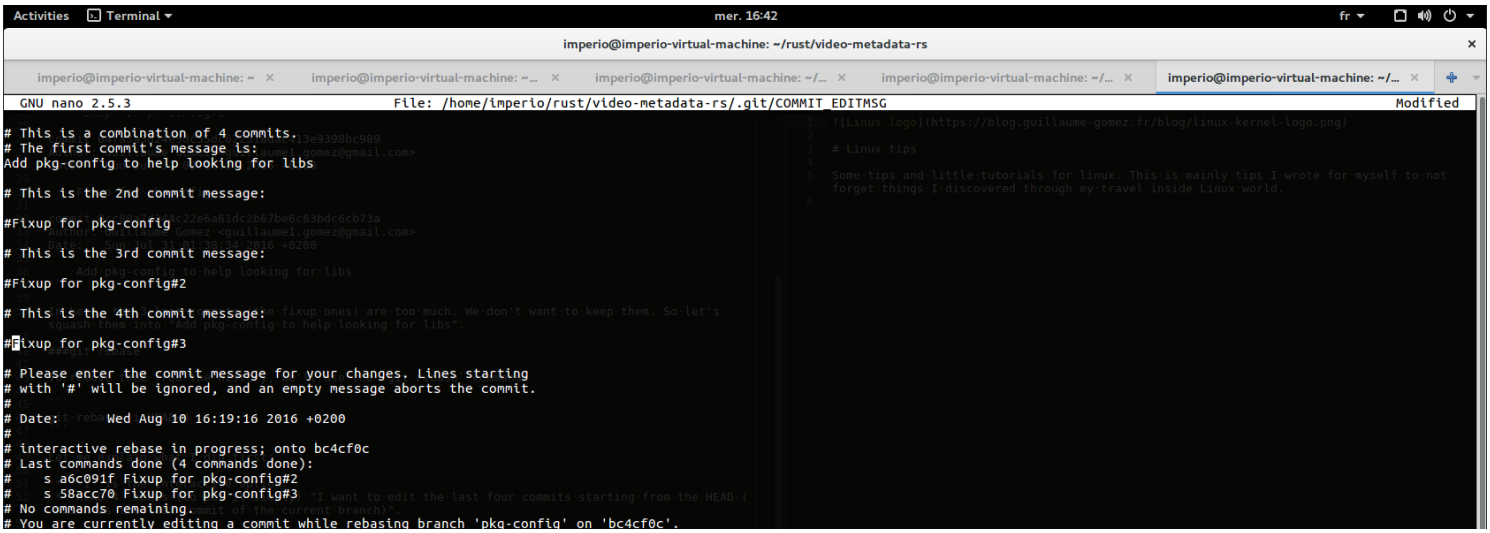
You should now see a screen like the following (if your default git editor is `nano`, otherwise it's just the same):



Since we need to squash the 3 last ones, replace their "pick" by "squash" (or "s"):



Now save and close. `git` will now open a last window which contains the commits' message. Add a "#" at the beginning of every line you don't want:



3. Rewrite a project history

Before going any further, I need to warn you: All changes are **definitive**! Once again, be very careful when using this.

Fear git rebase!

Yes, once again, this is `git rebase` which allows to do it. When you're using the interactive option ("-i" for those who didn't read the previous chapter) you can do a lot of things. For example, you can switch the order of a few commits or even delete some! Since there isn't much to tell, I'll provide a quick example:

We start a rebase:

```
git rebase -i HEAD~2
```

We now have the following screen:

We switch the two commits:

We save and close. Then we need to push the changes on the repository. Once again, I warn you (like I already did in the previous chapter): we'll use the force-push option so be very **careful**!

```
git push -f origin HEAD:the_branch_I_want_to_push_on
```

That's it!

4. Reset a branch on a precise commit

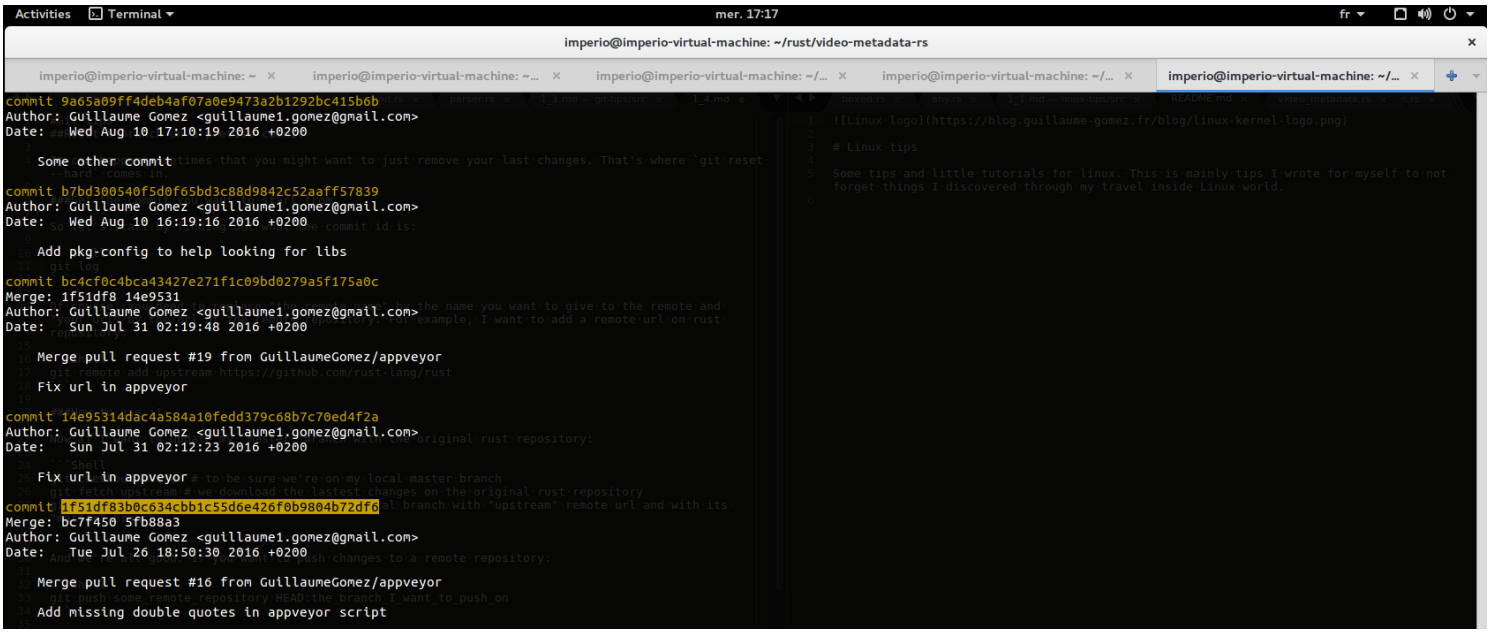
It can happen sometimes that you might want to just remove your last changes. That's where `git reset --hard` comes in.

Get the commit you want to start from

So let's start by finding out what the commit id is:

```
git log
```

Copy the commit id (the highlighted part below):



Then just reset your local branch on this commit:

```
git reset --hard 1f51df83b0c634cbb1c55d6e426f0b9804b72df6
```

Of course, you need to put your own commit id instead of mine.

If you want/need to push the changes on the repository. Once again, I warn you (like I already did in the previous chapters): we'll use the force-push option so be very **careful**!

```
git push -f origin HEAD:the_branch_I_want_to_push_on
```

That's it!