git

GitHub

Adipta Basu

1. Initialising an empty GIT Repository

```
git init
```

2. How to check what files are changed, and not committed

```
git status
```

3. How to stage files to commit

`git add .a` → .a stages all the unstaged files
`git add file1 file2` → stages only the files, whose name is given.

4. How to commit the changes to GIT

`git commit -m "A proper message"` → '-m' basically message, followed by the message.

5. How to remove from staging

```
git restore --staged filename
```

6. How to check the change history of a project.

```
git log
```

7. Removing a commit

Commits in GIT are cascading in nature. Hence, to remove a specific commit means reverting back to the file before that specific commit.

Step 1:

git log → then select the commit hashcode of the commit that is just before the commit, that you desire to remove.

Step 2:

git remove hashcode

So what happens to the files, that were removed? ⇒ They were moved to the unstaged area. You can check them in status.

8. How to save the progrem in GIT, but not put it in a log.

First, the changes need to be staged, and then, they need to be stashed.

git stash

9. How to restore stash, and bring them to the unstaged area?

git stash pop

10. How to delete what is present in the stash?

git stash clear.

11. How to connect the online repo with your local repo?

`git remote add origin https://github.com/repopath`

12. How to push your changes to the online git url?

$\text{git push } \underbrace{\text{origin}}_{\text{the url that you push to}} \ \underbrace{\text{master}}_{} \rightarrow$ the branch that you push to.

13. How to make a new GIT Branch?

`git branch branchname`

14. How to checkout a specific GIT Branch?

`git checkout branchname` → when this command is used, the HEAD in the .git folder is changed to the mentioned branch.

15. How to merge branches in git?

Let us say, that we have a main branch, and a second branch, named branchB. So, to merge them, first the branch to which the other branch needs to be merged is checked out. In our case, check out the main branch, and then use the following command.

$\text{git merge } \underbrace{\text{branchname}}_{} \rightarrow$ for our case branch name will be branchB.

16. Working with existing projects, that donot belong to us.

The process of doing this is pretty simple. First we need to fork the project repository that we would like to work on, the clone it to our local, make a branch in the repo, and the make your changes. After they are made, and you are satisfied, just make the pull request.
So, after forking and cloning your desired repo, follow these steps:

1. Step 1

```
git remote add upstream https://github.com/mainrepoaccount/repo.git
```

2. Step 2

```
git branch newbranch
git checkout newbranch
```

→ So basically, we are making a new branch, and checking it out. And all the chan es will be done in this branch.

3. Step 3

```
git add.
git commit
git push origin branchname
```

→ This basically means that we will add the files to that specific branch. And then commit it to that branch and then push it.

After this is done, you can go to the web gui of GIT and then go to your branch. There, you will find the pop-up option for a new pull request.

After a pull request is made, make another pull request for another issue fix. So, for that you need to make a new branch. As, two pull requests cannot be created from the same branch.

4. Step 4

```
git fetch --all --prune  OR  git pull upstream main
git checkout main
git reset --hard upstream/main
```

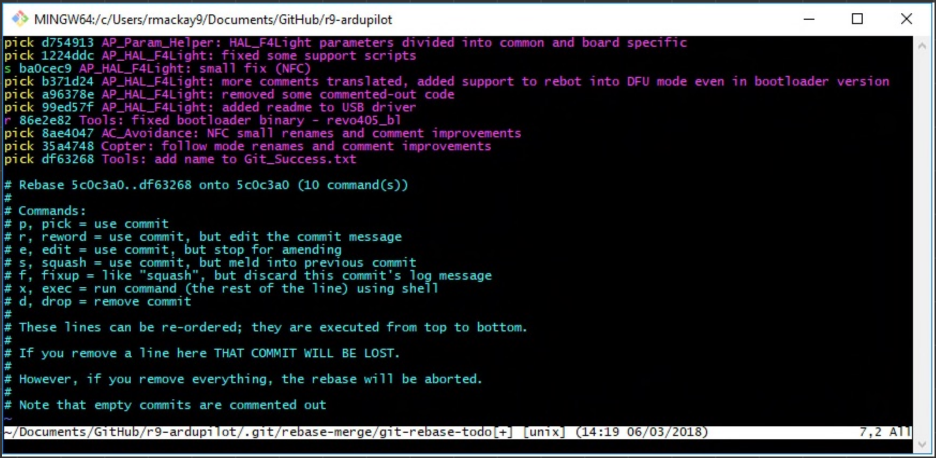
→ These are used, so that we can fetch the changes the changes in the upstream branch.

17. Picking and Squashing Commits (Basically means rationalising multiple commits to one)

Let us say that we have multiple commits, that we have not pushed. and now we need to push them. So for that, we need to find the hash of the last push that we had done. And then we use the rebase command.

`git rebase -i hashcode` $\longrightarrow$ we get this from log.

After this command is used, an interactive environment will open, where we will need to select the files that we need to pick or squash



MINGW64:/c/Users/rmackay9/Documents/GitHub/r9-ardupilot

```
pick d754913 AP_Param_Helper: HAL_F4Light parameters divided into common and board specific
pick 1224ddc AP_HAL_F4Light: fixed some support scripts
s ba0cec9 AP_HAL_F4Light: small fix (NFC)
pick b371d24 AP_HAL_F4Light: more comments translated, added support to rebot into DFU mode even in bootloader version
pick a96378e AP_HAL_F4Light: removed some commented-out code
pick 99ed57f AP_HAL_F4Light: added readme to USB driver
r 86e2e82 Tools: fixed bootloader binary - revo405_bl
pick 8ae4047 AC_Avoidance: NFC small renames and comment improvements
pick 35a4748 Copter: follow mode renames and comment improvements
pick df63268 Tools: add name to Git_Success.txt

# Rebase 5c0c3a0..df63268 onto 5c0c3a0 (10 command(s))
#
# Commands:
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
# d, drop = remove commit
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
# Note that empty commits are commented out
~
~/Documents/GitHub/r9-ardupilot/.git/rebase-merge/git-rebase-todo[+] [unix] (14:19 06/03/2018)          7,2 All
```

Selecting pick means that it will be considered as a commit, and picking squash means merging the files to the previous picked file.