Git

Most useful commands

Content

[Introduction](#_k9nbf8vo1qwn)

[Purpose of the Document](#_a4di4jdzvt1i)

[Role and Responsibilities](#_bl84pz6nbdqb)

[Objectives](#_stzlnm41r59d)

[Commands](#_bw19sjr44ter)

[Project init](#_vfc7mhfvqp7p)

[First commit commands](#_n3qop7ux8com)

[Common commit commands](#_cmfdtr8puhzl)

[List](#_57y3y1wwf5cb)

[Setup and Config](#_rj1ibn3wr45k)

[Getting and Creating Projects](#_ah0nfnb6trh9)

[Basic Snapshotting](#_jenhu61d8r1a)

[Branching and Merging](#_ctjm7zv13nqe)

[Sharing and Updating Projects](#_tdt9p7d1vuc9)

[Patching](#_n9k6gjdvrenb)

[Remove settings and credentials](#_by40vo1dx4y5)

[Remove your user settings](#_93w3wipokiid)

[Remove credentials](#_10i2wu3b3zhs)

# Commands

## **Project init**

git init

git add .

git commit -m "first commit"

git remote add origin https://github.com/[your\_user]/[your\_reposiory].git

git push -u origin master

## **First commit commands**

git init

→ initialize a repository

git add .

→ adds all the files from the working directory into the staging area for the next commit

git commit -m “First commit”

→ add a comment to the new commit

git push origin master

→ push everything to master

## **Common commit commands**

git status

→ show you the different states of files in your working directory and staging area

git add **.**

→ adds all the files from the working directory into the staging area for the next commit

git commit -m “**<text\_comment>**”

→ add a **<text\_comment>** comment to the new commit

git push origin **<branch\_name>**

→ push every commited file to the git

## **List**

### Setup and Config

git config --global alias.co checkout

→ infer your command if you type it in partially, instead of typing git checkout, you just need to type git co

git config --global alias.st status

→ infer your command if you type it in partially, instead of typing git status, you just need to type git st

### Getting and Creating Projects

git init

→ initialize a repository

git clone **<url\_name>**

→ clone a repository from **<url\_name>** in the current folder

### Basic Snapshotting

git add **.**

→ adds all the files from the working directory into the staging area for the next commit

git add **<file\_name>**

→ adds **<file\_name>** from the working directory into the staging area for the next commit

git status

→ show you the different states of files in your working directory and staging area

git diff

→ used when you want to see differences between any two trees

git diff **<branch\_name>** ~/Desktop/**<file\_name>.path**

→ get differences between my current branch and <branch\_name> and store it on **<file\_name>.path**

git commit

→ to do the commit open **“vim”** and press **“i”** to write a comment, then **“Esc”** and

**“:wq”** to save and exit

git commit -m “**<text\_comment>**”

→ add **<text\_comment>** to the new commit

git reset HEAD~**<number>**

→ undo the last **<number>** commits from the current branch

git reset --hard origin/**<branch\_name>**

→ undo the **<branch\_name>** changes and set up like remote

git clean -fd

→ remove unwanted files from your working directory

### Branching and Merging

git branch **<branch\_name>**

→ create a new branch with **<branch\_name>** id

git branch -d **<branch\_name>**

→ delete a branch with **<branch\_name>** id

(you can’t delete a branch if you are on it)

git checkout .

→ discard all changes in the project

git checkout -- **<file\_name>**

→ discard all changes in **<file\_name>**

git checkout **<branch\_name>**

→ switch to **<branch\_name>** the branch you are currently on

git checkout -b **<branch\_name>**

→ if it not exists creates **<branch\_name>** and switch to it

git log

→ show the reachable recorded history of a project

git log -p

→ get an idea of what was introduced in each commit

git log --**<file\_name>**

→ show the reachable recorded history from **<file\_name>**

git stash -u

→ temporarily store uncommitted work in order to clean out your working directory without having to commit unfinished work on a branch

git stash pop

→ restored uncommited work previously stored

### Sharing and Updating Projects

git pull

→ fetch from the remote you specify and then immediately try to merge it into the branch you are currently on

git pull **<remote\_name>** **<branch\_name>**

→ download **<branch\_name>** from **<remote\_name>**

git push origin **<branch\_name>**

→ upload every commited file to **<branch\_name>** in the git

git push -u origin **<branch\_name>**

→ creates a branch on remote and copy it to origin

(need to be done before pull request or the first time to create a new local branch in remote)

git remote add origin **<url\_name>**

→ clone and link a repository from **<url\_name>** to the current workspace

git remote -v

→ show remotes configured on my repository

git remote set-url origin **<url\_name>**

→ change origin **<url\_name>** from the current repository

### Patching

git cherry-pick **<commit\_name>**

→ take the change introduced in **<commit\_name>** and try to re-introduce it as a new commit on the branch you are currently on

git rebase -i HEAD~**<number>**

→ show the commits from the current branch and allow to operate on the last **<number>** commits

git rebase -I s**<commit\_name>**

→ joins several commits in one

(before each **<commit\_name>** you want to join add **s)**

git rebase --abort

→ abort the rebase

Git rebase what it does is collect one by one the committed changes in one branch, and apply them on another. Using rebase can help us avoid conflicts as long as it is applied to commits that are local and have not been uploaded to any remote repository. If you are not careful with this and a work partner uses affected changes, you will surely have problems since these types of conflicts are usually difficult to repair.

# Remove settings and credentials

## **Remove your user settings**

git config --global --unset user.name

git config --global --unset user.email

Or all your global settings:

git config --global --unset-all

## **Remove credentials**

When github fed my credentials through windows credential manager instead of git bash.

You may have to delete windows credential manager and delete the github entry under: Control panel

→ User accounts

→ Credential manager

→ Windows credentials

→ Generic credentials