

Introduction to Git

The bare minimum any developer
MUST know about Git

`xavier.nodet@gmail.com`

Installation

- Windows:
 - <https://git-scm.com/download/win>
- Mac:
 - open a Terminal
 - enter the command `git` and follow the prompts
- Linux:
 - already there

If you will only remember these ones...

```
$ git init
```

```
$ git add [existing-file]
```

```
$ git commit -a -m "Summary of the changes"
```

```
$ git log
```

```
$ git show [commit-id]
```

```
$ git diff
```

Let's make this story a little bit longer

- Git is a version control system
- Used by a vast majority of companies and software projects
- It saves the state (existence, content, permissions) of a set of files, and allows to navigate the history of these files
- Primarily deals with text files, but binary files are accepted
- Git helps for backup, history, documentation, bug hunting...
- And facilitates collaboration between developers

git init

- Creates a hidden `.git` directory in the current directory
- This directory will hold the complete history of the whole project
- It is known as the *local repository*
- Don't touch its content!

```
git add myfile.txt
```

- This tells Git that you want it to store `myfile.txt`
- From now on, Git will track the existence and content of the file

```
git commit -a -m "Summary of  
the changes"
```

- Takes a snapshot of all the files known to Git
- Records that snapshot in the local repository, the `.git` directory
- Gives it an id, a 40 digits hexadecimal string

git log

- Lists all the commits in the current project
- Displays the id, author, date and commit message

git log

- Lists all the commits in the current project
- Displays the id, author, date and commit message

```
$ git log -2  
commit 69bdf1eec4c8c82f94bd4b864da3e5d734861cba (HEAD -> master, origin/master)  
Author: Xavier Nodet <xavier.nodet@gmail.com>  
Date: Mon May 3 17:17:03 2021 +0200
```

Comments

```
commit 93491245b9109b68b317020f2590926399714843  
Author: Xavier Nodet <xavier.nodet@gmail.com>  
Date: Mon May 3 16:47:21 2021 +0200
```

Don't restrict local network

git log

- Lists all the commits in the current project
- Displays the id, author, date and commit message

```
$ git log --color --pretty=format:"%C(yellow)%h%C(reset) %<(50,trunc)%s %C(green)%ad%C(reset) %C(blue)[%an]%C(reset)" --relative-date --decorate
```

69bdf1e	Comments	3 weeks ago	[Xavier Nodet]
9349124	Don't restrict local network	3 weeks ago	[Xavier Nodet]
b2f8b75	Add a README	3 weeks ago	[Xavier Nodet]
ab400de	Rename the speed variables	3 weeks ago	[Xavier Nodet]
929bf34	100 as SPEED3 is the max we can do without degra..	3 weeks ago	[Xavier Nodet]
8ea1bcf	Add a script to remove the limits	1 year, 5 months ago	[Xavier Nodet]
d0cb30e	Spare some outbound traffic from restrictions	1 year, 5 months ago	[Xavier Nodet]
b08aed8	Limit outbound as well	1 year, 5 months ago	[Xavier Nodet]
5af4877	And now, it works...	1 year, 5 months ago	[Xavier Nodet]
09258b4	Make the script executable	1 year, 5 months ago	[Xavier Nodet]
354231b	Direct copy from SuperUser	1 year, 5 months ago	[Xavier Nodet]

git show [commit-id]

```
git show --oneline ab400de
```

```
ab400de Rename the speed variables
```

```
diff --git a/throttle.sh b/throttle.sh
```

```
index 3218d04..266fcea 100755
```

```
--- a/throttle.sh
```

```
+++ b/throttle.sh
```

```
@@ -3,4 +3,4 @@
```

```
# Max speeds in KB/sec
```

```
-SPEED1=1000
```

```
-SPEED3=100
```

```
+INBOUND=1000
```

```
+OUTBOUND=100
```

```
@@ -39,5 +39,5 @@ EOF
```

```
# Create the dummynet queue
```

```
-dnctl pipe 1 config bw ${SPEED1}Kbyte/s queue 50
```

```
+dnctl pipe 1 config bw ${INBOUND}Kbyte/s queue 50
```

```
dnctl pipe 2 config queue 50
```

```
-dnctl pipe 3 config bw ${SPEED3}Kbyte/s queue 50
```

```
+dnctl pipe 3 config bw ${OUTBOUND}Kbyte/s queue 50
```

```
dnctl pipe 4 config queue 50
```

git diff

- Lists the changes since the last commit
- Similar output as `git show`

Other very useful commands

```
$ git status
```

- List modified files

```
$ git reset --hard HEAD
```

- Forget uncommitted changes

```
$ git checkout [commit-id]
```

- Restores the working copy to the state it had in the commit id.

```
$ git tag V1.0 69bdf1e
```

- Assigns a name to a commit

```
$ git blame myfile.txt
```

- List the file with date and commit id that last touched each line.

```
$ git checkout master
```

- Get back to the default branch

Collaboration

- Git facilitates collaboration in a team
- Typically, one repository serves as the collaboration hub
- For example <https://github.com>
- Each developer can add commits on top of the others'
- One gets other's commits with `git pull`
- And publishes his own commits with `git push`

Branching

- Trying out new ideas may be easier/safer if done in isolation
- The goal would be to push only once the whole thing is done
- *Branches* build a 'copy' of the current state
- Committing in a branch doesn't impact the other branches
- When ready, you *merge* back your changes

For more information

- The Git parable:
 - <https://tom.preston-werner.com/2009/05/19/the-git-parable.html>
 - <https://youtu.be/jm7Qsl-nNjk>
- Pro Git
 - HTML: <https://git-scm.com/book/en/v2>
 - Available as PDF or epub on the same page
- Tons of other sources...