Introduction to GIT

for PyLadies Vienna #3

lubojr

Lubomír Bucek <lubomir.bucek@eox.at>

18.01.2020

Overview

- install git, quick setup, create GitHub account
- what is git, why we use it
- what is GitHub, show alternatives
- how does git work, some terminology
- quick example using git and remote (GitHub)

Install GIT

MAC: https://git-scm.com/download/mac
Windows: https://git-scm.com/download/windows
Ubuntu/Fedora: **sudo apt-get install git**

```
Basic config after install -> new command line
git config --global user.name ‘‘Your Name and Surname‘‘
git config --global user.email youremail@example.com

Windows: git config --global core.editor notepad
         git config --global format.commitMessageColumns 80
         git config --global gui.encoding utf-8
Ubuntu: git config --global core.editor nano
```

What is git, why we have it?

- most widely used distributed version control system (initially developed by 2005 Linus Torvalds)
- security, flexibility, performance
- versioning files & collaboration standard
- git is a very well supported open source project with tons of tutorials, dedicated websites, external tools etc
- common criticism of Git “it is hard to learn”

What is GitHub?

- Global company providing hosting (hub) for projects using git + a lot of other own features
- free + paid tiers
- Alternatives: gitlab, bitbucket
- Internet archive, social networking platform
- UI, Bug tracking - Issues, Forking repositories, Pull requests, Changelogs, Versions

Key concepts - Repository (repo)

- all files and their history
- each “project” under git control needs to be in own directory
- can be located on your machine or also on a hub
- copying repository from remote server is called **cloning**

Key concepts - Snapshots

- the way git keeps track of your code history
- records of what all your files look like through time
- you can travel forth and back to history of repository
- you decide when you create snapshots

Key concepts - Commits

- Creating of a snapshot (I commited or made a commit)
- Contains information about:
  - how files changed (difference)
  - who did it and when
  - 40 character hash code (f4f78b319c308600eab015a5d6529add21660dc1)
  - message / description
  - which commit is a parent commit

Key concepts - Branches

- Basically just pointers to commits - logically dividing work
- branch essentially says: I want to include the work of this commit and all parent commits.
- master branch (convention)
- *branch early, branch often* - git mantra (not 100% strict, depends on workflow of a team)

Lets get to ACTION!

- https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet
- tutorial to help with commands when needed

Interactive demo workflow

---

git init (invisible .git folder created)
add file with text
commit file

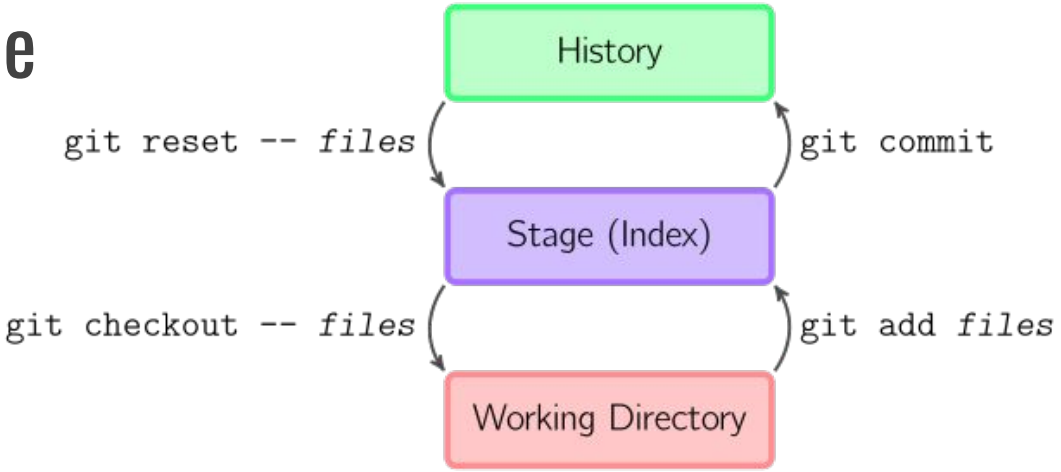
git show (last commit)

make a change to a file, git diff

add, commit (talk about commit messages, 70 chars)

Staging and inspecting tree

```
git log
git log --graph --oneline
```



Branching

---

until now one branch (master)
git branch <name> or git checkout -b <name>
HEAD
git commit
git checkout master, git merge hotfix
delete old branch

Branching with merge commit and conflict

---

commit on branch, commit on master
merge branch to master
merge commit has two parents
two commits on two branches with same line changed, merge
git status to see which files, edit manually, commit

Reverting bad commits

---

git reset
git reset --hard (most recent commit) or <commit>
~, ^ operators

git reset HEAD~3 (now be really careful about --hard) - deletes commits
whenever using --hard, --force, BE CAUTIOUS

GitHub and remote

- git clone <url.git> - download a remote repository to a local system, copy of branches on remote - origin
- git remote
- git push <origin> <branch> - puts your local commits to remote repository branch
- git fetch - just downloads all new commits from remote
- git pull - git fetch and git merge combined

Put your local repository to GitHub

go to github.com, log in, create new public repository
git remote add origin <url.git>
git remote -v (verify)
git push origin master (login needed)
until you push, most of operations are safe
when other people involved (remote), extra caution

working with remotes in more detail can be a quick topic for coding session in two weeks

git with ssh

---

removes necessity to put email & password on pushing

https://inchoo.net/dev-talk/how-to-generate-ssh-keys-for-git-authorization/ - for each platform

Some resources

---

bad commits:

http://justinhileman.info/article/git-pretty/git-pretty.png

awesome interactive tutorial on basic git and branching

https://learngitbranching.js.org/

Thank you!!

Have fun with git