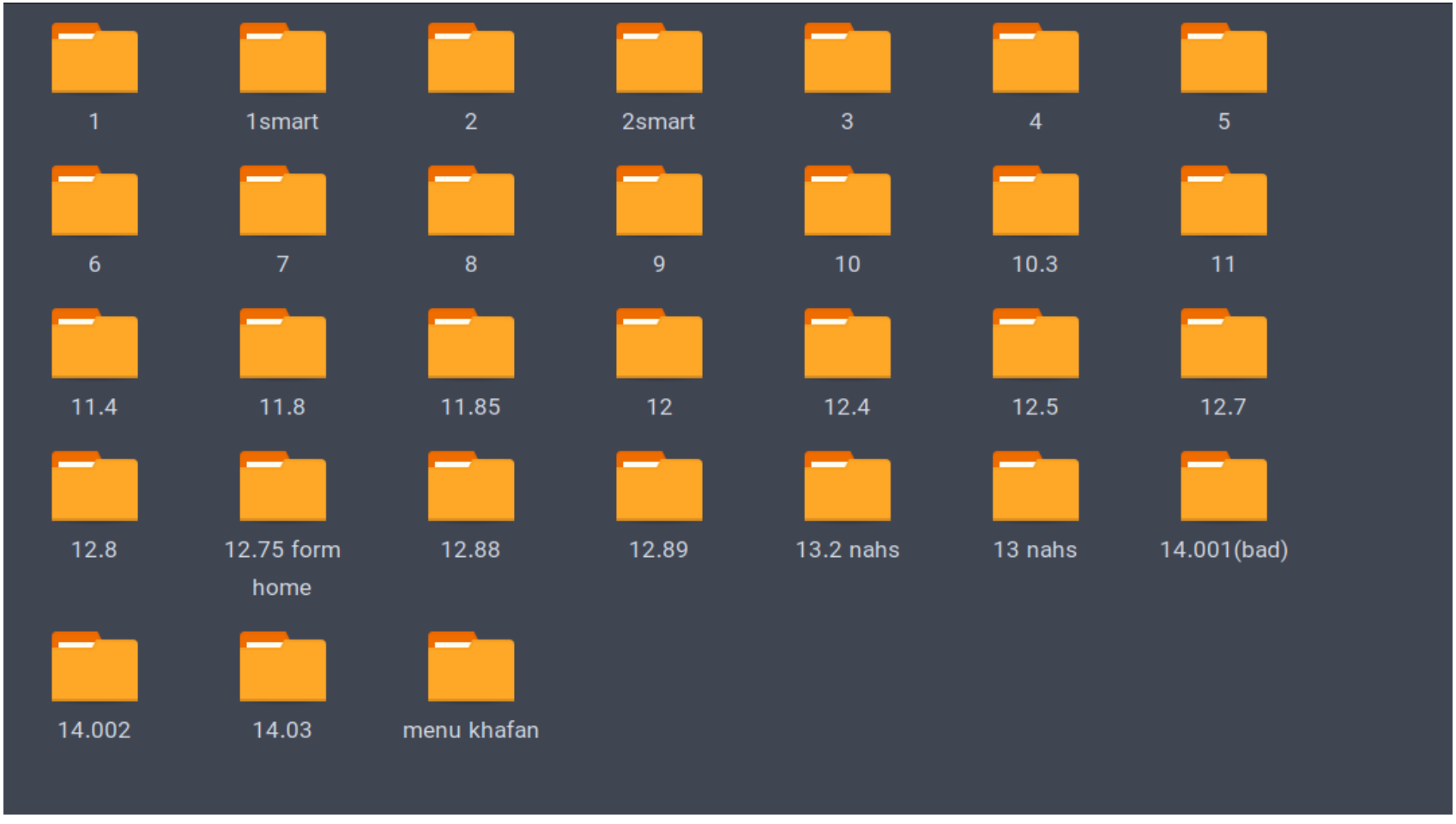
git!

Git (/git/)[7] is a distributed version-control system for tracking changes in source code during software development.

Table Of Contents

what do we do now? manual version controlling by programmer



problems

- copy-paste/save-as whole project after every stable build
- what if more than 1 developer work at the same time?
- which version was stable?
- all files are unnecessary while saving as?

benefits of version control

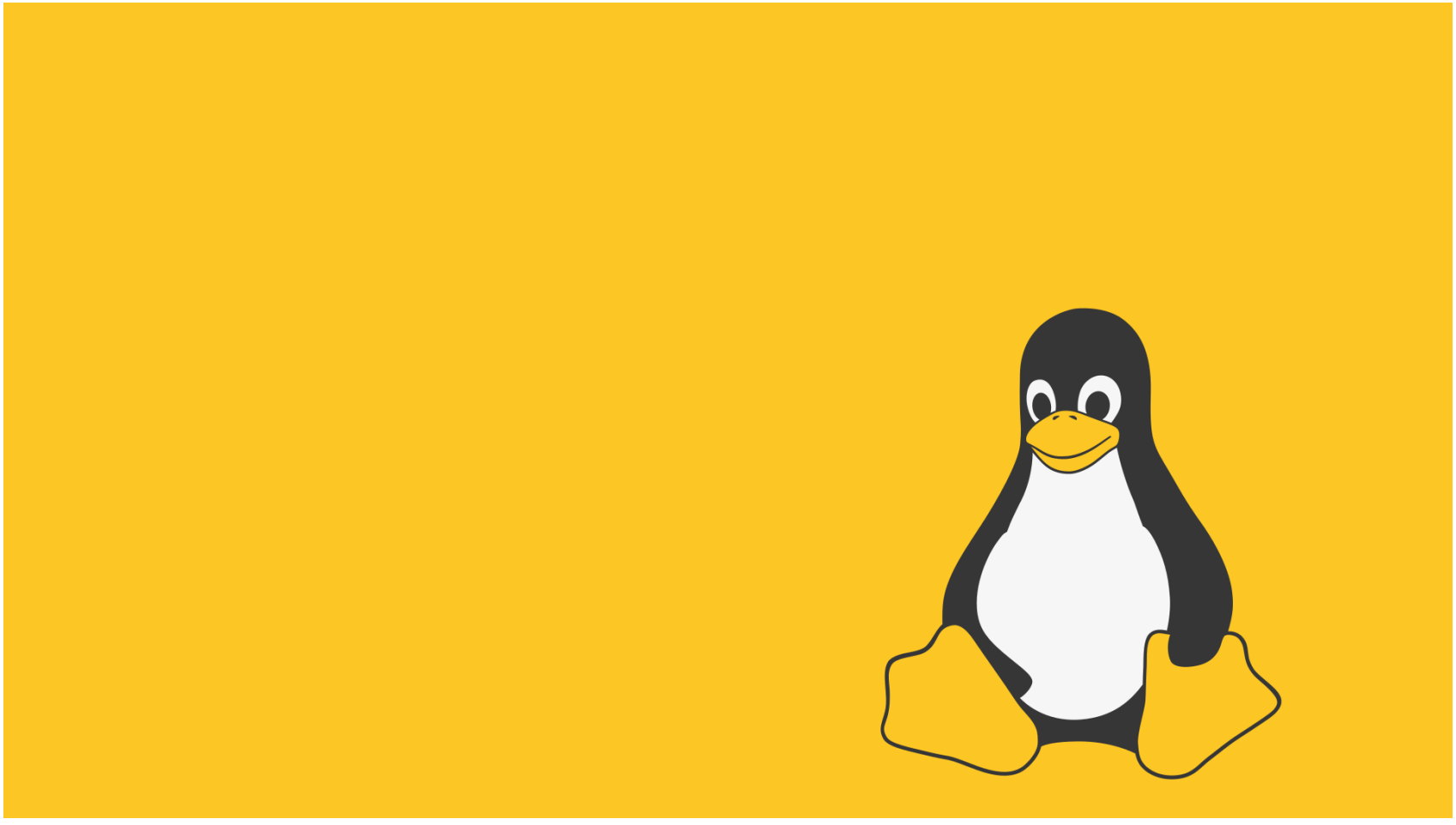
- easily management collaboration on a project
- ability to have unlimited number of developers
- easily revert back your files if something went wrong

SVN (by Apache)

Visual Studio Team Services code (by Microsoft)

git (by Linus Torvalds)

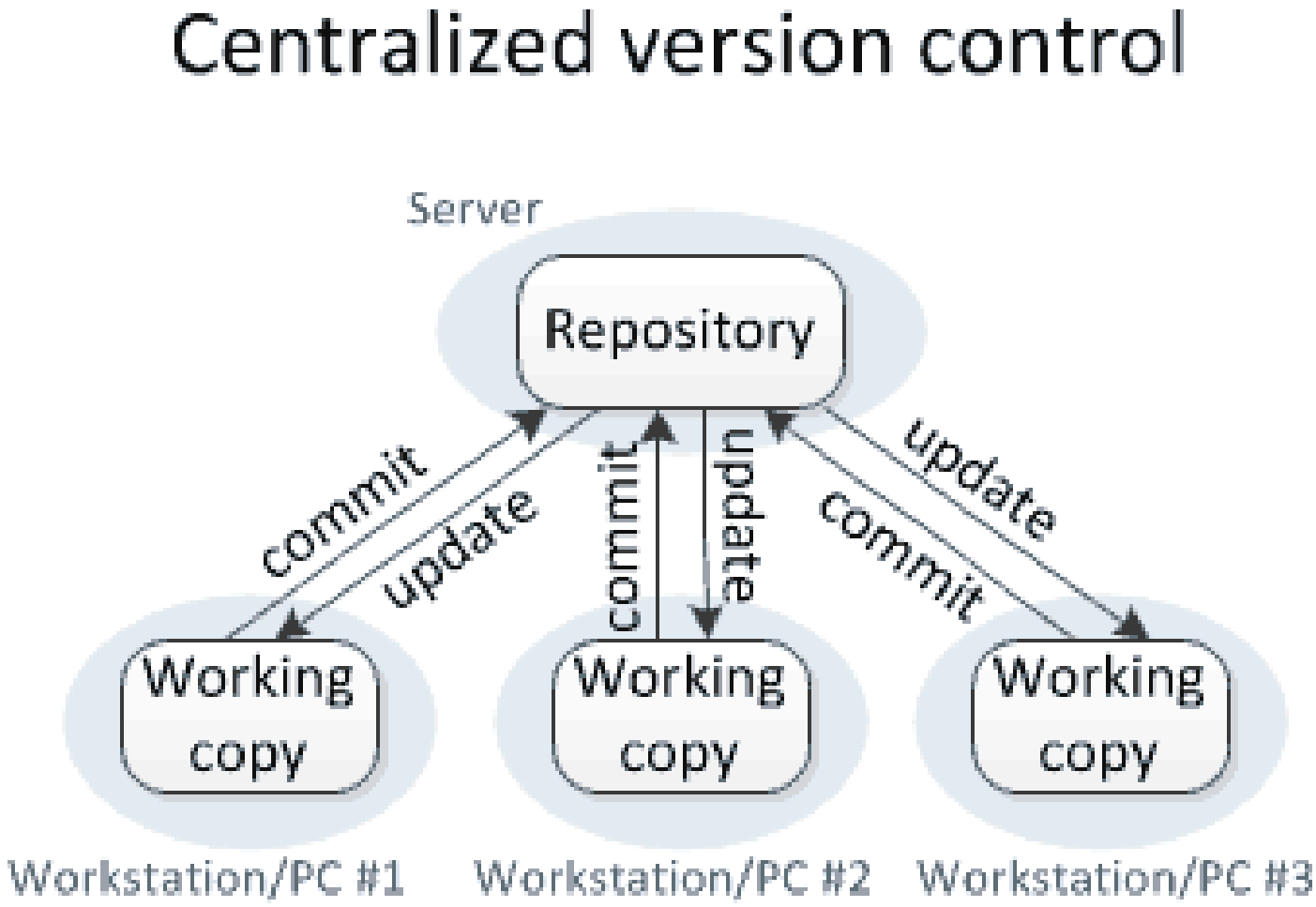
As of 2020, the 5.6 release of the Linux kernel had around 33 million lines of code.



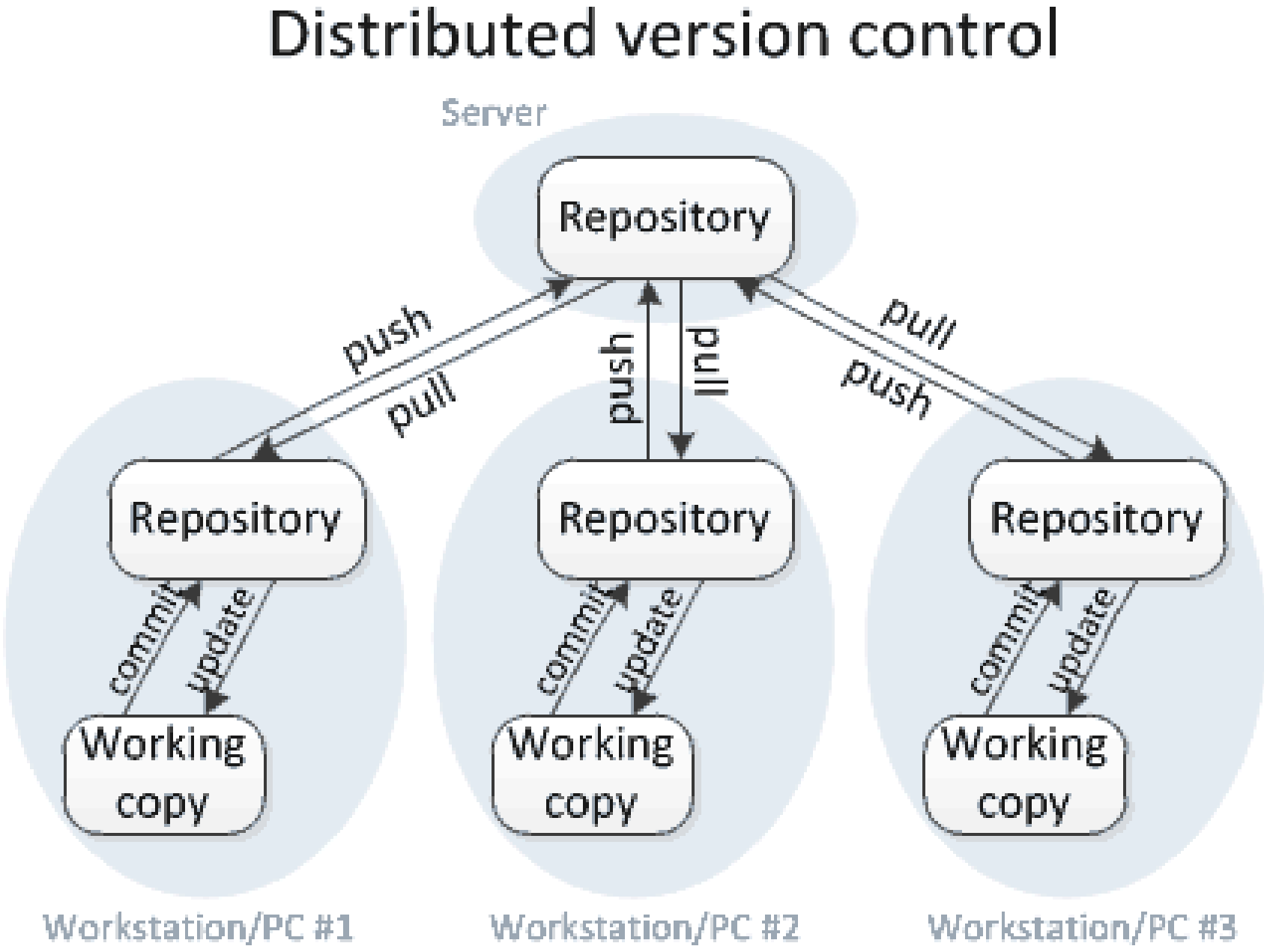
git features

- free and open source
- distributed
- non-linear (branches)
- handle large projects efficiently

Centralized version control



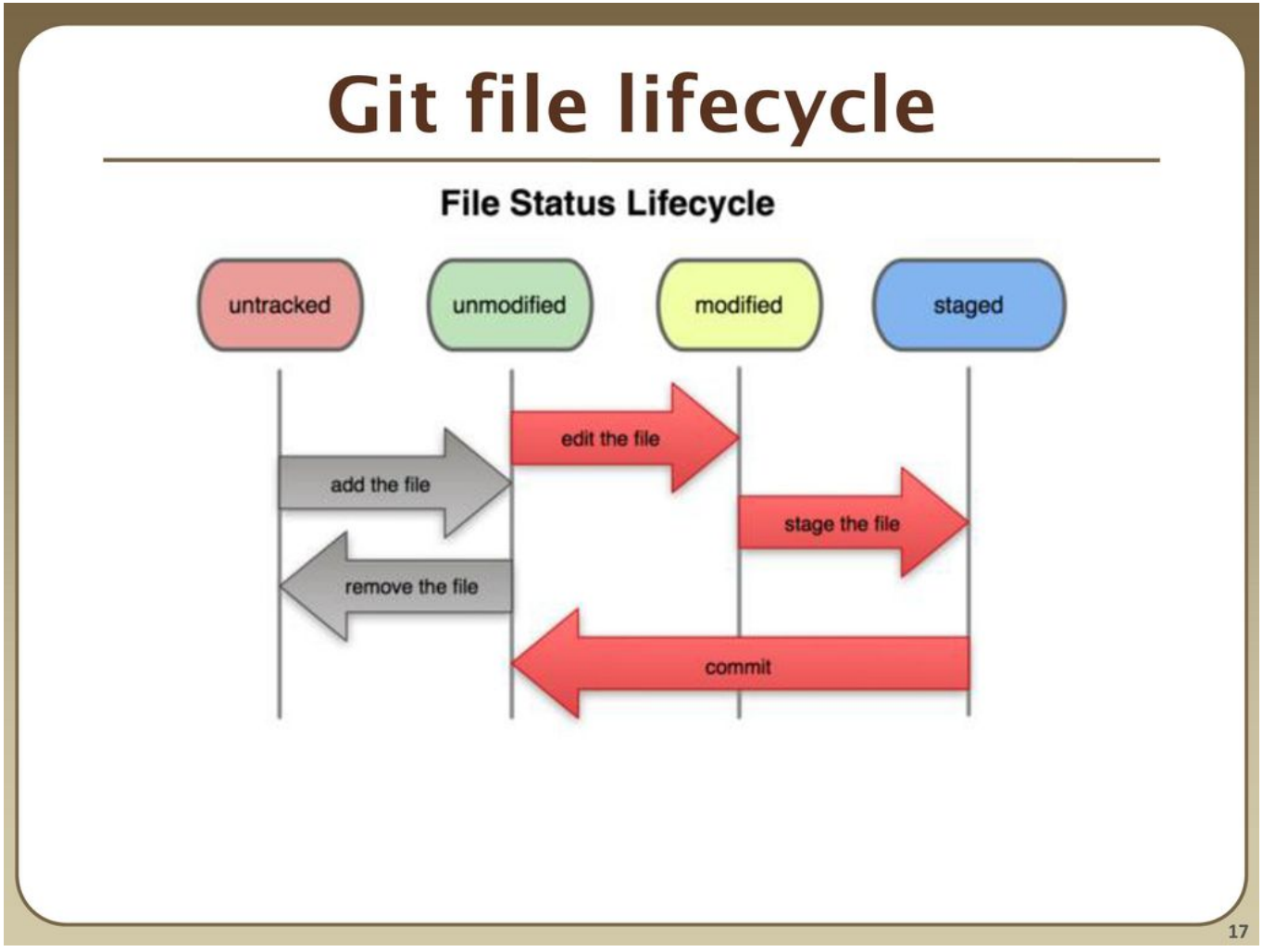
Distributed version control



how to use git

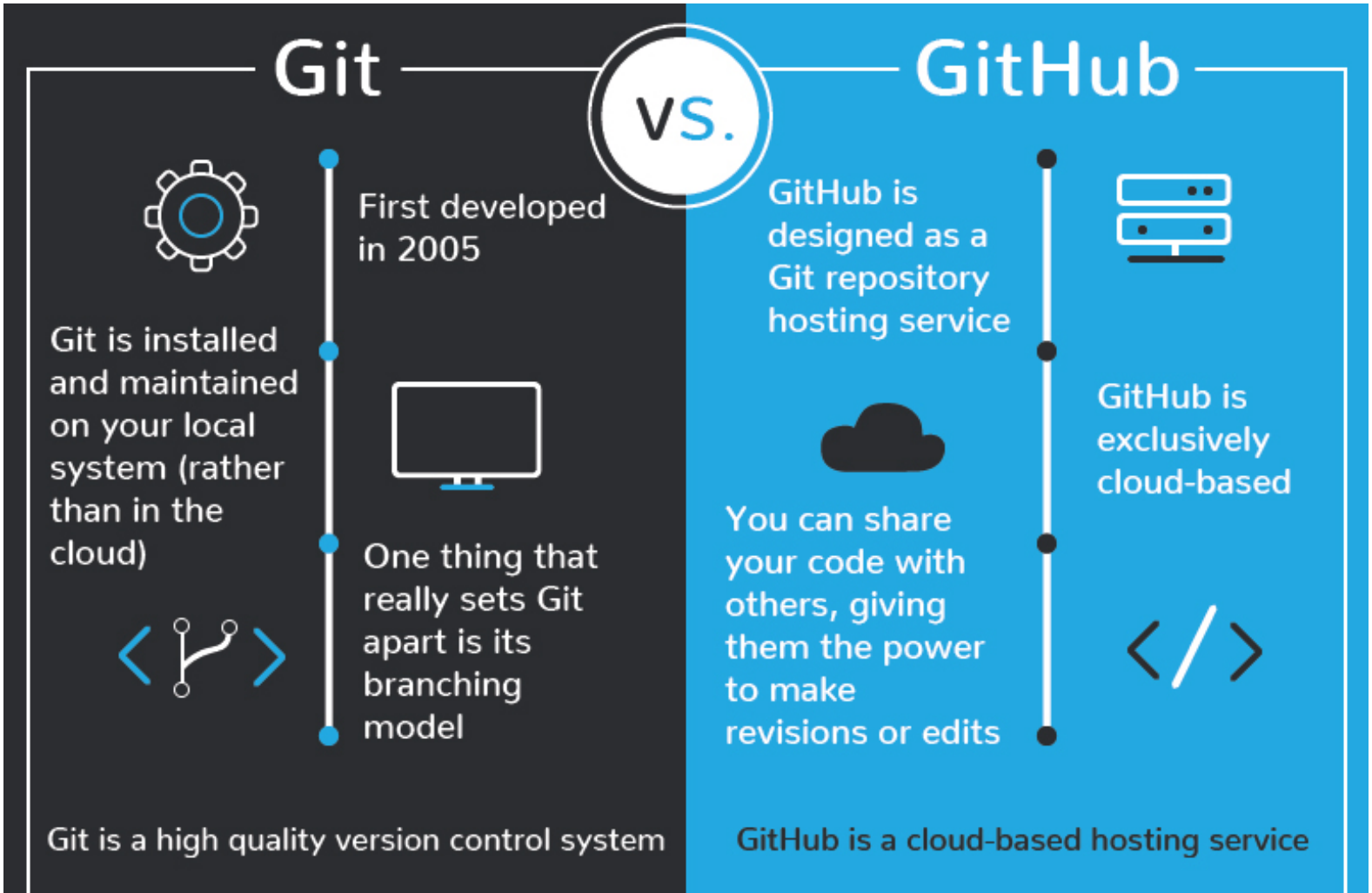
1. search!
2. I search too
3. everybody else does search too

file status life cycle



github

- instagram for gits
- a place to keep gits, review them, fork them, star them.
- alternatives: gitlab, bitbucket, any other place



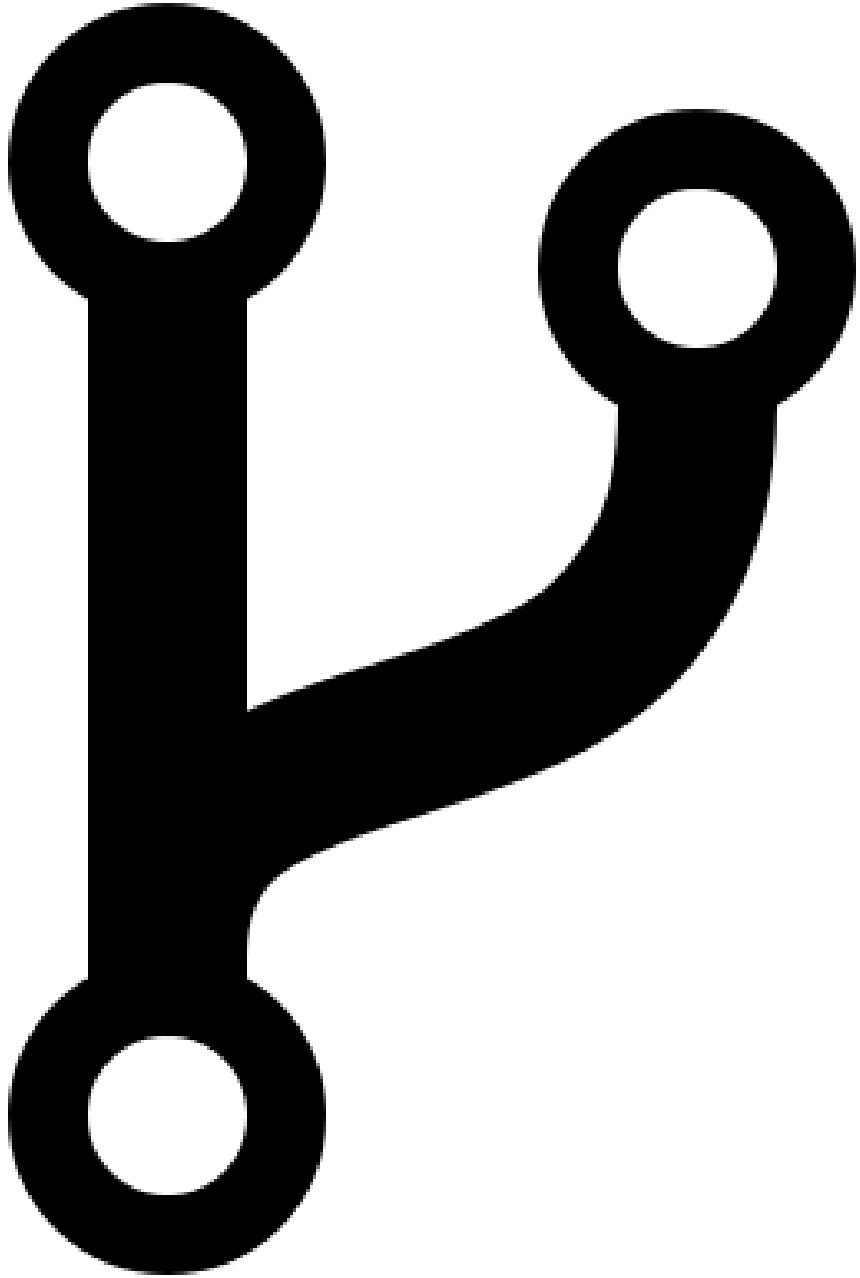
push? remote? clone?

- remote: where should i upload my gits
- push: act of uploading gits
- clone: download whole git
- pull: check for updates in the remote git

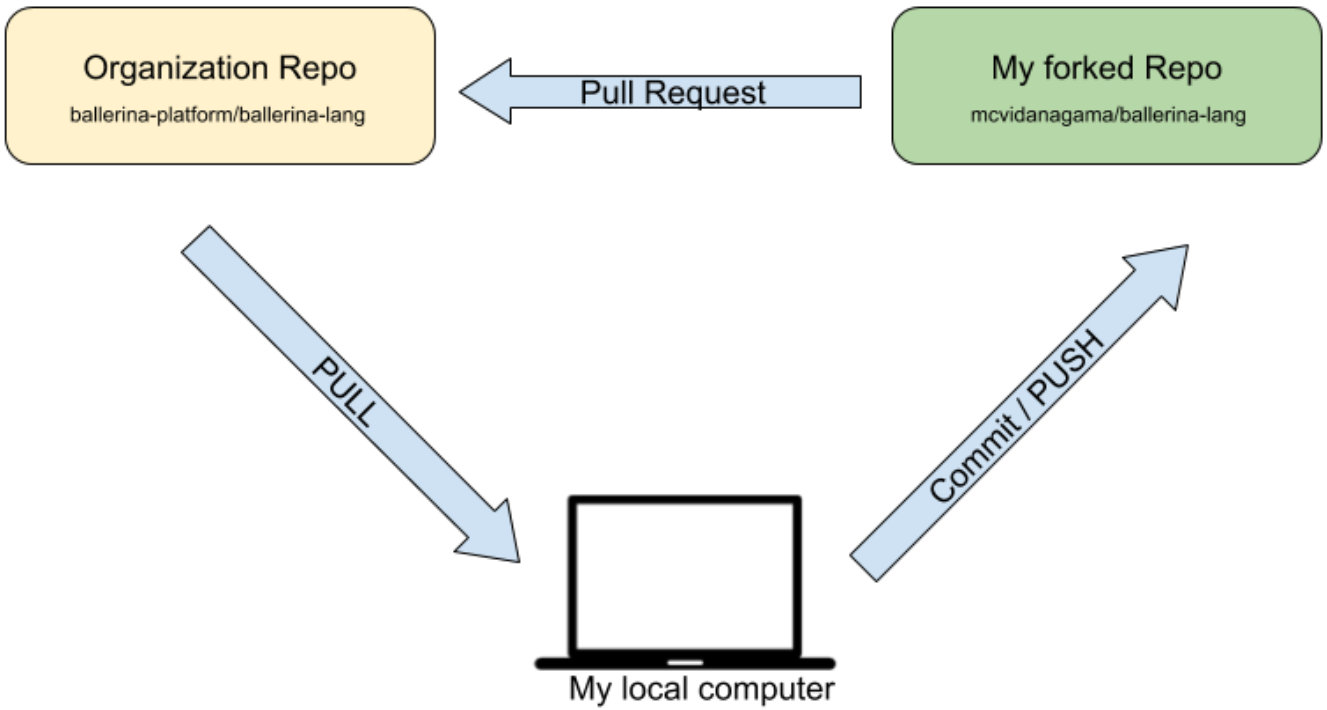

1. git commit


2. git push

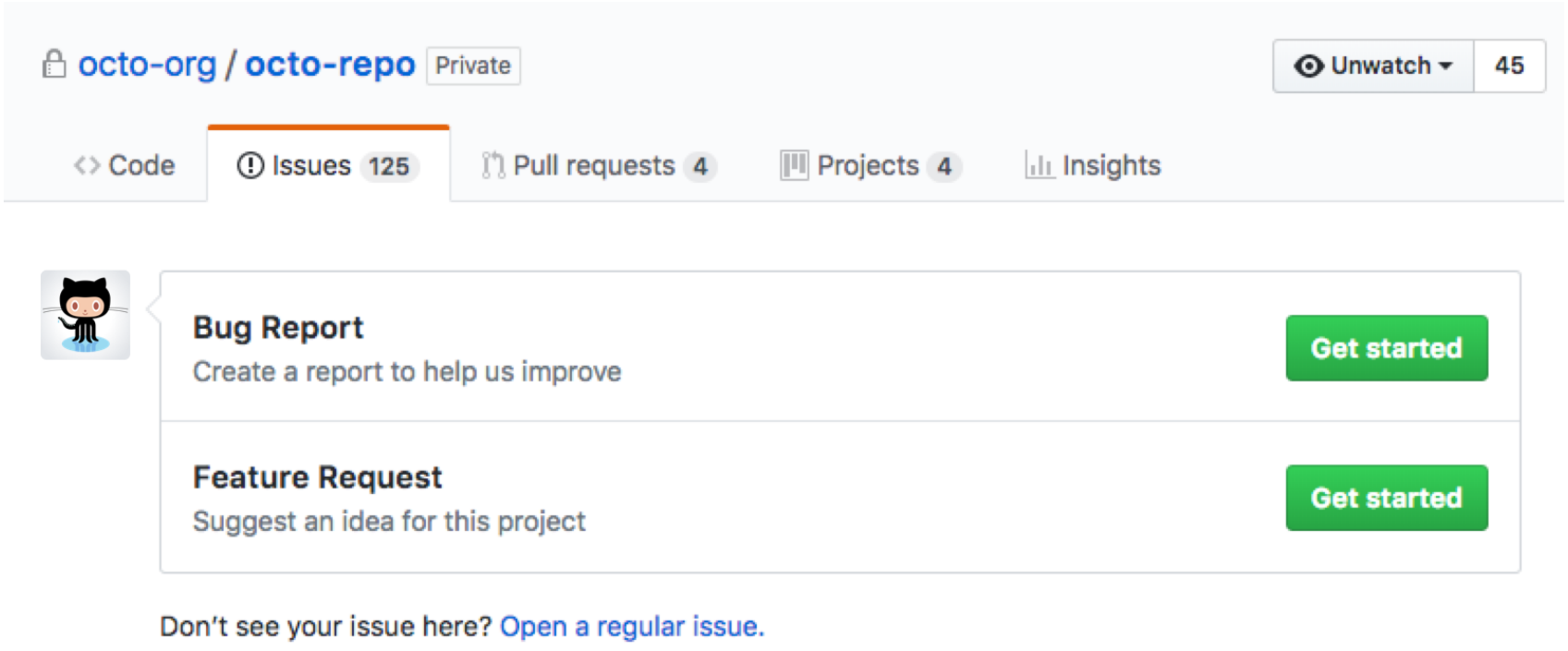
fork



PR


Organization Repo
ballerina-platform/ballerina-lang
Pull Request
My forked Repo
mcvidanagama/ballerina-lang
PULL
Commit / PUSH
My local computer

issue, issue template



.gitignore, .git

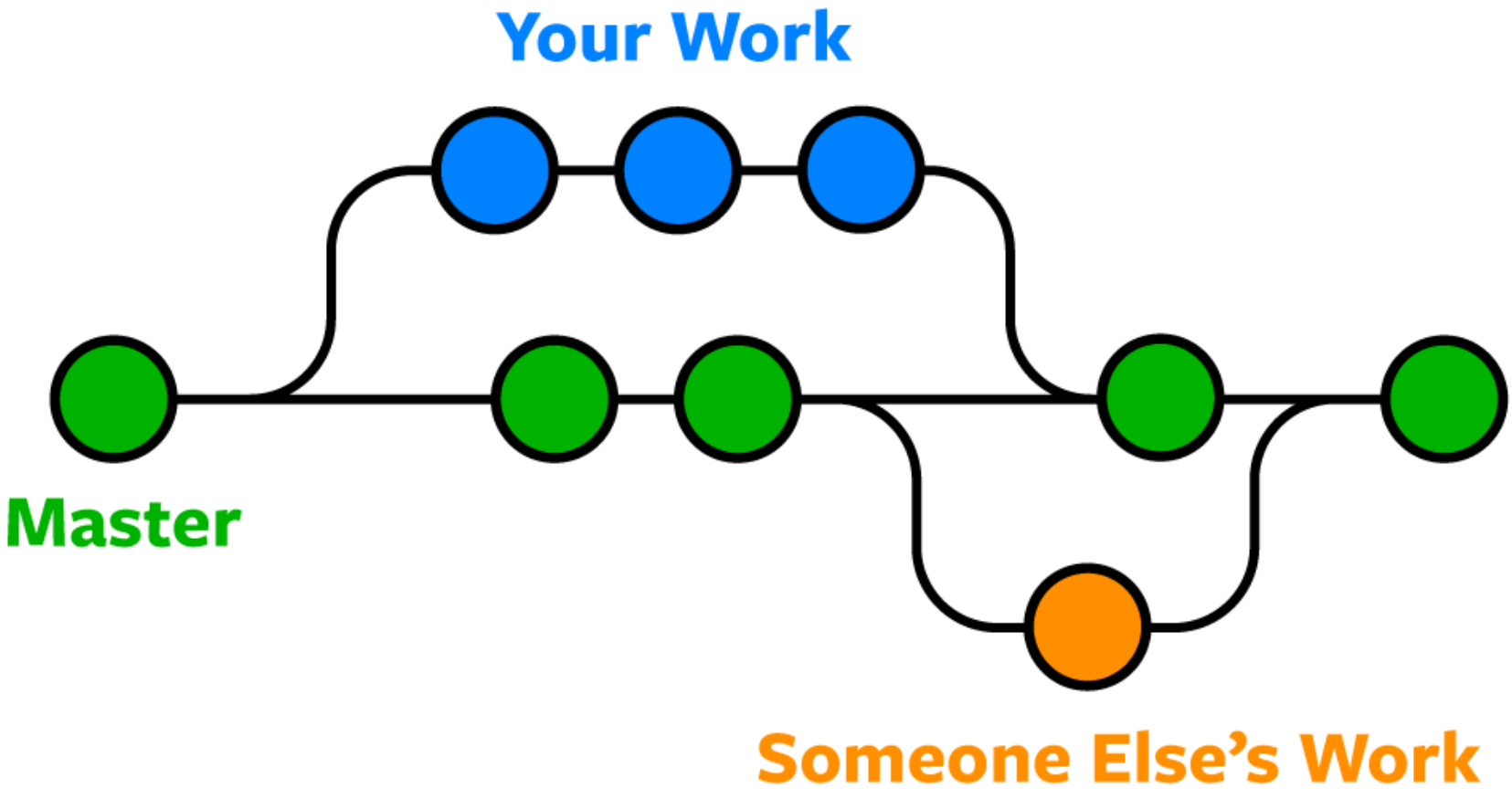
- .git: local and hidden folder that contains git internal files, don't open it!
- delete .git folder in case of removing git from project
- .gitignore: ignore these sort of files

```
*.class
.idea/
__pycache__/
```

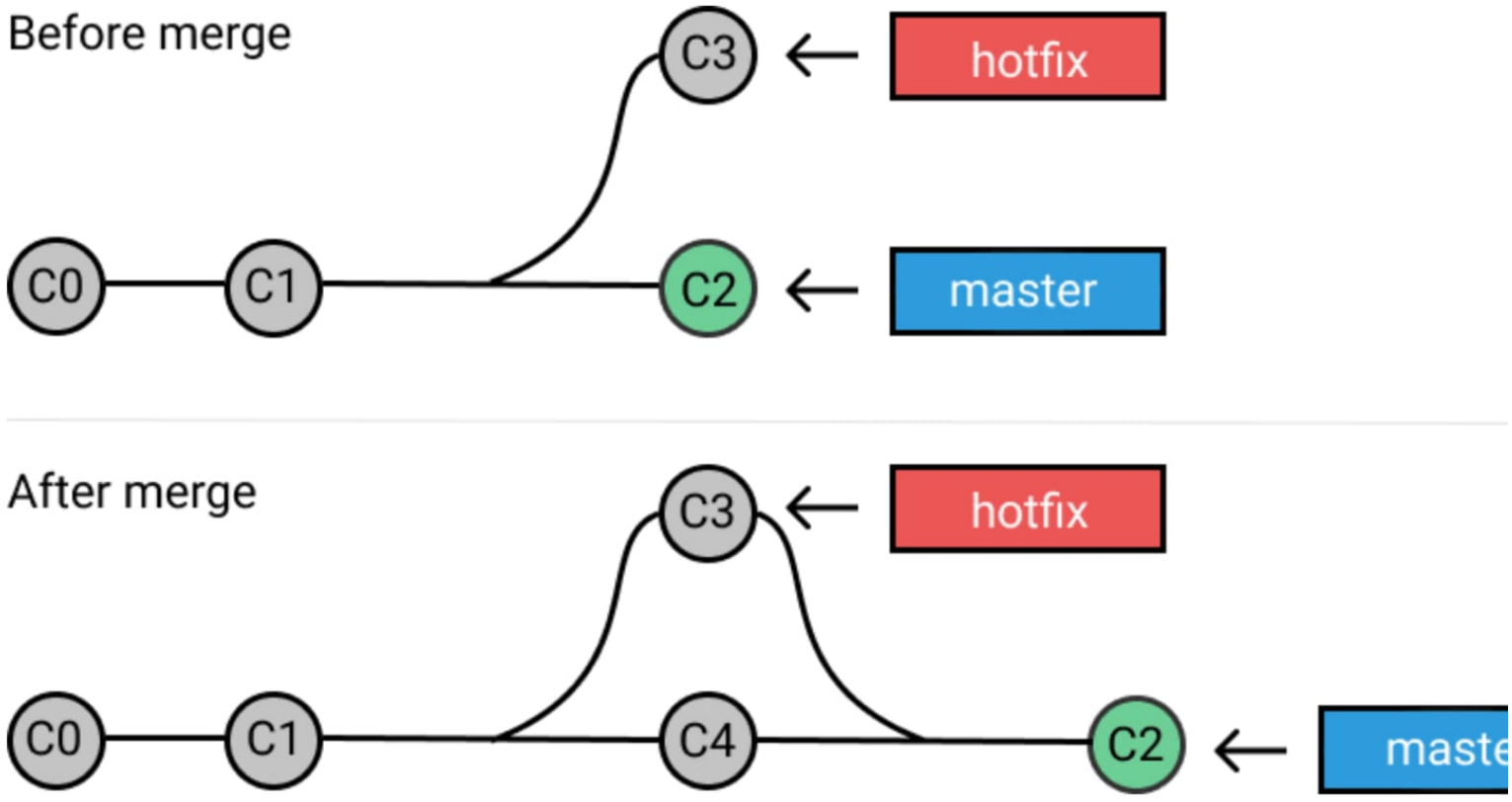
good site: gitignore.io

do not commit large and binary files!

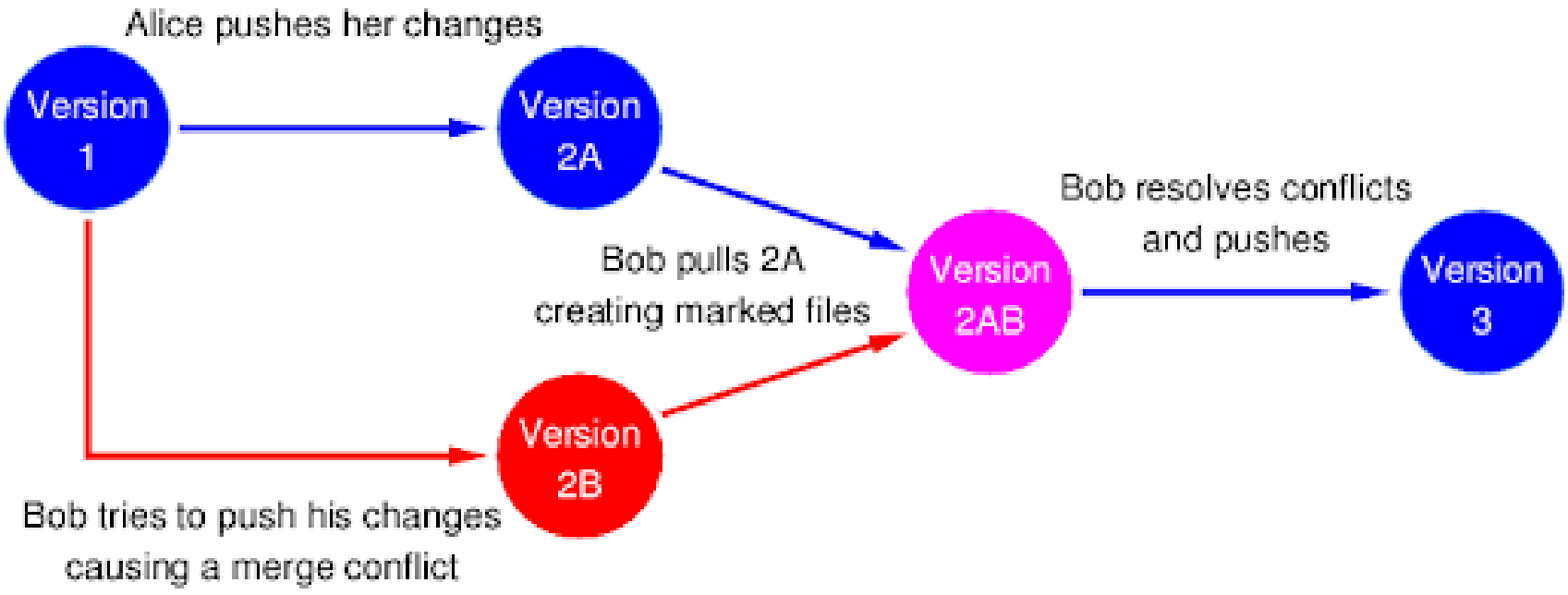
branch



merge


Before merge
C3
hotfix
C0
C1
C2
master
After merge
C3
hotfix
C0
C1
C4
C2
maste

merge conflict



common commands (1)

```
# first time initialize
git config --global user.name "Bugs Bunny"
git config --global user.email bugs@gmail.com
git init
```

common commands (2)

```
# regulary code and commit
git status
git add -A # or git add filename
git commit -m 'commit message'
```

common commands (3)

```
# work with remote
git remote add origin https://github.com/yc/yr.git
git push origin master # from master to origin remote
git pull
git clone https://github.com/sb-acc/some-repo.git
```

common commands (4)

```
# see old commits and other versions
git log
git log --abbrev-commit --pretty=oneline
git checkout # change HEAD
git diff # difference
```

common commands (5)

```
# eveything messed up
git reset --hard HEAD # revert to last commit
rm -rf .git # get rid of git!
```

further read

- this github io page
- command by command explain
- jadi's videos
- step by step
- this good slide
- tags