Molecular Biomedical Informatics

分 子 生 醫 資 訊 實 驗 室

Web Programming

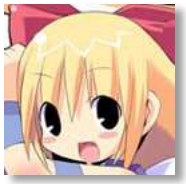
網 際 網 路 程 式 設 計

Git

Version Control - Episode 1 - Introduction

提升程式設計師生產力最好的工具是什麼？

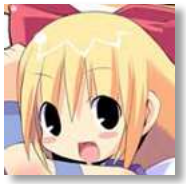
Git and Github



Why

version control

version control system



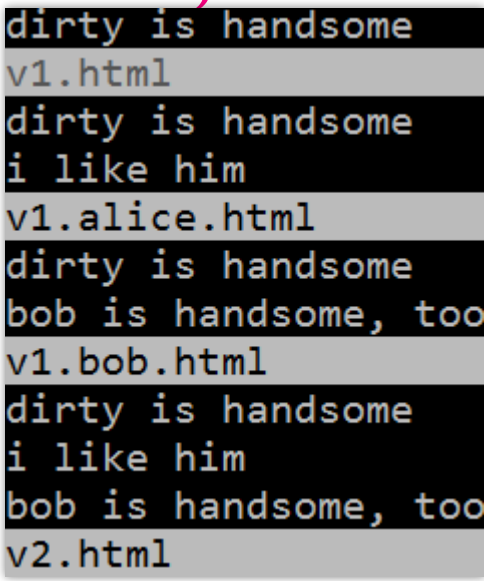
Suppose that

Alice and Bob are co-working on a html file,

how do you do version control

Diff

- Take turns?
- One solution is that both work on local copies and sync (merge) later (filename is used for version control)
  - in this case there is a conflict
  - someone need to solve it
- diff is a tool to help sync
  - `diff v1.alice.html v1.bob.html`
- Or you can use vi's diff mode
  - `vi -d v1.alice.html v1.bob.html`



Diff

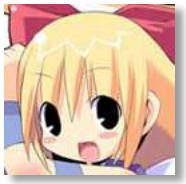
- Take turns?
- One solution is that both work on local copies and sync (merge) later (filename is used for version control)
  - in this case there is a conflict
  - someone need to solve it
- diff is a tool to help sync
  - `diff v1.alice.html v1.bob.html`
- Or you can use vi's diff mode
  - `vi -d v1.alice.html v1.bob.html`

```
2c2
< i like him
---
> bob is handsome, too
```

Diff

- Take turns?
- One solution is that both work on local copies and sync (merge) later (filename is used for version control)
  - in this case there is a conflict
  - someone need to solve it
- diff is a tool to help sync
  - `diff v1.a`

- Or you can use vi's diff mode
  - `vi -d v1.alice.html v1.bob.html`

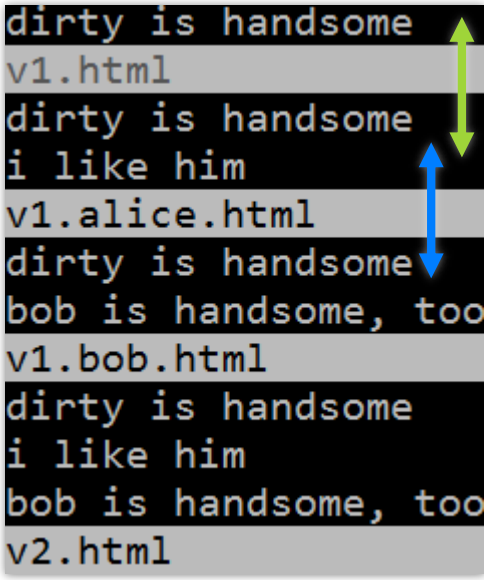


How

Bob should do if he finds v1.alice.html while editing v1.bob.html

Patch

- Bob can merge v1.alice to his local copy and keep editing
  - but if there is 1, 2… *n* more new versions during the editing
  - changes in v1.alice ↔ v1.bob is hard, since both changes
  - changes in v1.alice ↔ v1 is easy
- Apply v1 → v1.alice on Bob's local copy
  - `diff -u v1.html v1.alice.html > a`
  - edit the filename in alice.patch if required
  - `patch alice.patch`



Patch

- Bob can merge v1.alice to his local copy and keep editing
  - but if there is 1, 2… *n* more new versions durin
  - changes in v1.alice ↔ v1.bob is hard, since bot
  - changes in v1.alice ↔ v1 is easy
- Apply v1 → v1.alice on Bob's local copy
  - `diff -u v1.html v1.alice.html > alice.patch`
  - edit the filename in alice.patch if required
  - `patch alice.patch`

```
1a2
> i like him
alice.diff
--- v1.html
+++ v1.alice.html
@@ -1 +1,2 @@
 dirty is handsome
+i like him
alice.patch
```

Patch

- Bob can merge v1.alice to his local copy and keep editing
  - but if there is 1, 2… *n* more new versions durin
  - changes in v1.alice ↔ v1.bob is hard, since bot
  - changes in v1.alice ↔ v1 is easy
- Apply v1 → v1.alice on Bob's local copy
  - `diff -u v1.html v1.alice.html > alice.patch`
  - edit the filename in alice.patch if required
  - `patch alice.patch`

```
1a2
> i like him
alice.diff
--- v1.html
+++ v1.alice.html
@@ -1 +1,2 @@
 dirty is handsome
+i like him
alice.patch
```

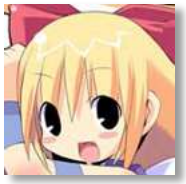
Tips of diff and patch

- diff –Naur [from] [to]
- patch –dry-run –p1

- Then google :p

Any Questions?

about diff and patch



How

Bob finds v1.alice.html

Server

- More common, repository
- So far, think it as a FTP server is fine
- No one can edit directly on the server copy
  - once the filename (version) fixes, the content fixes
  - any change induces a new file (version)
- So users have to change the filename (under some rules) of local copies and upload it to the server, namely commit
- Other users can download the new versions, diff and patch local copies, namely update

Version control system

- Provide a server for store/access versions
- Control versions instead of filenames
  - no more file renaming, keep it index.html rather than index.v1.html, index.v2.html…
  - a new version is automatically generated via commit
- Auto diff and patch
  - precisely, auto diff when committing, auto patch when updating
- Detect conflicts
  - if no conflict exists, auto merge/solve
  - the later committer is responsible to solve conflicts
  - a version might be overwritten, but never lost, which can be easily recovered

History of version control systems

- rcs, cvs, svn, (svk), git

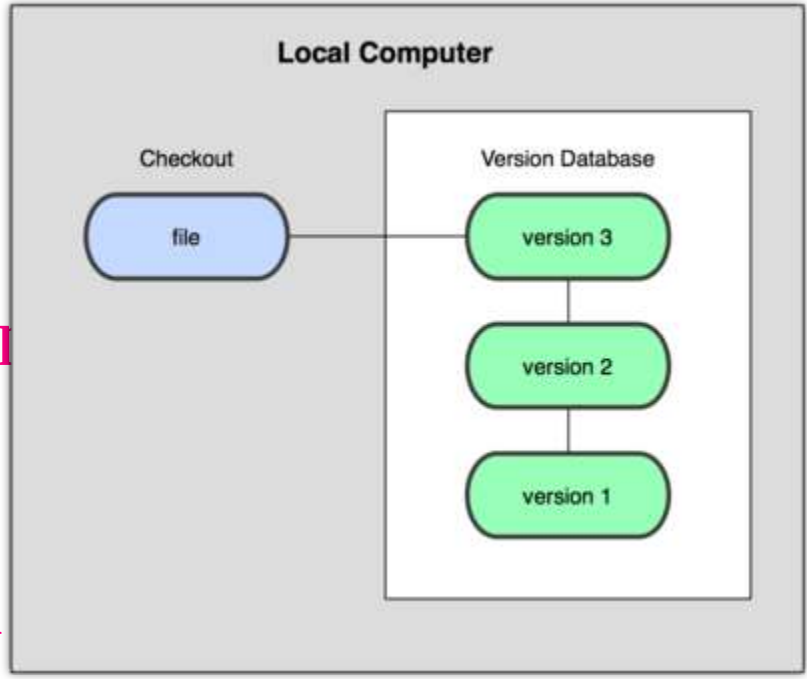
- Then google :p

History of version control systems

- rcs, cvs, svn, (svk), git

- Then google :p
- rcs only handles version control but not collaboration issues
- cvs and svn are similar, but svn was more popular (because of better GUI?)
- svk was developed by clkao (my classmate), which is a distributed system
- Doesn’t matter, git wins

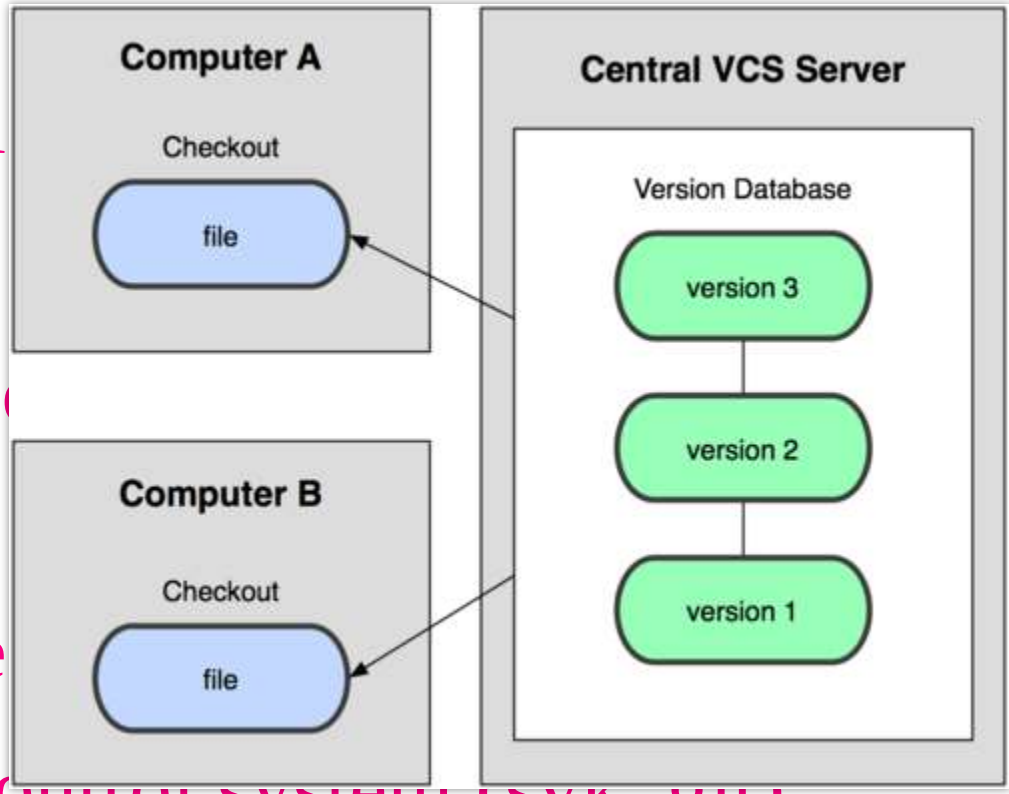
Distributed



- Local version control syste
  - no collaboration
- Centralized version control
  - slow/no network
  - single point of failure
- Distributed version control system (svk, git)
  - a complete/full-functional repository at local

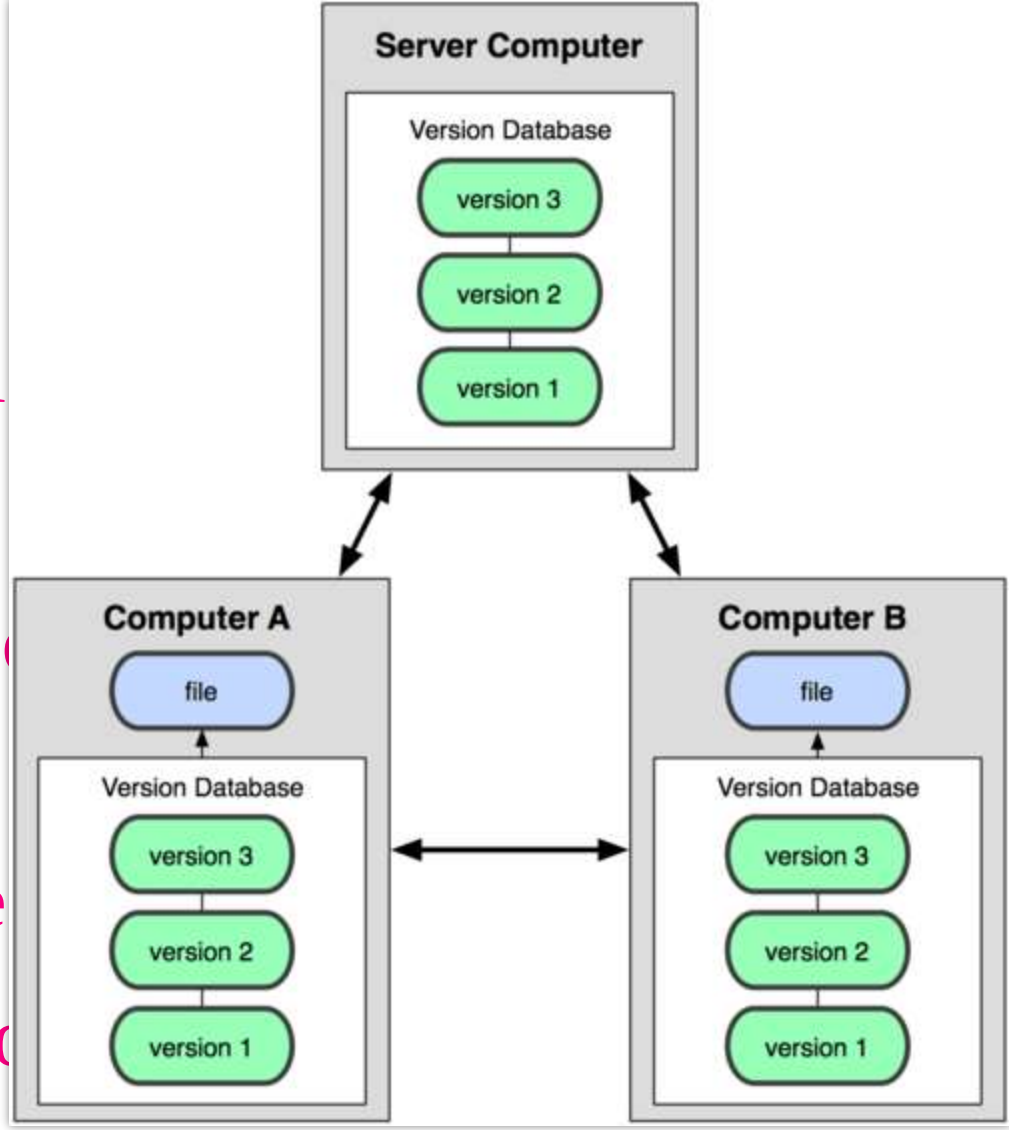
Distributed

- Local version control
  - no collaboration
- Centralized version c
  - slow/no network
  - single point of failure
- Distributed version control system (SVK, git)
  - a complete/full-functional repository at local



Distributed

- Local version control
  - no collaboration
- Centralized version c
  - slow/no network
  - single point of failure
- Distributed version c
  - a complete/full-functional repository at local



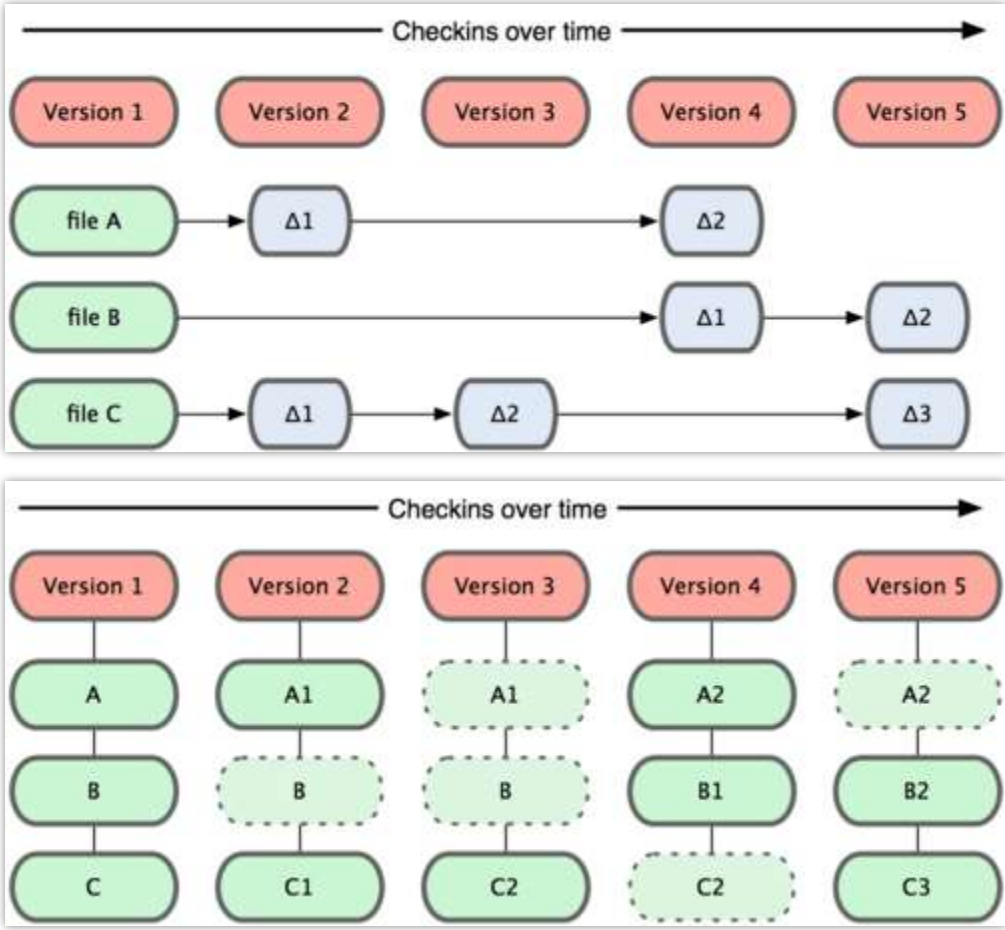

By Linus Torvalds
TRANSMETA
SOFTWARE
http://en.wikipedia.org/wiki/Linus_Torvalds

Linus Torvalds (born Dec 28, 1969)

- Linux creator
- Linus adopted BitKeeper, a distributed version control system
  - before that, Linux kernel was released by patch files
- He breached with BitKeeper at 2002, so he created a new one, git
- Git was released at 2005, now everybody uses it
  - never breach with masters
- The first project that uses git, Linux kernel
  - did you develop any project larger than Linux kernel?

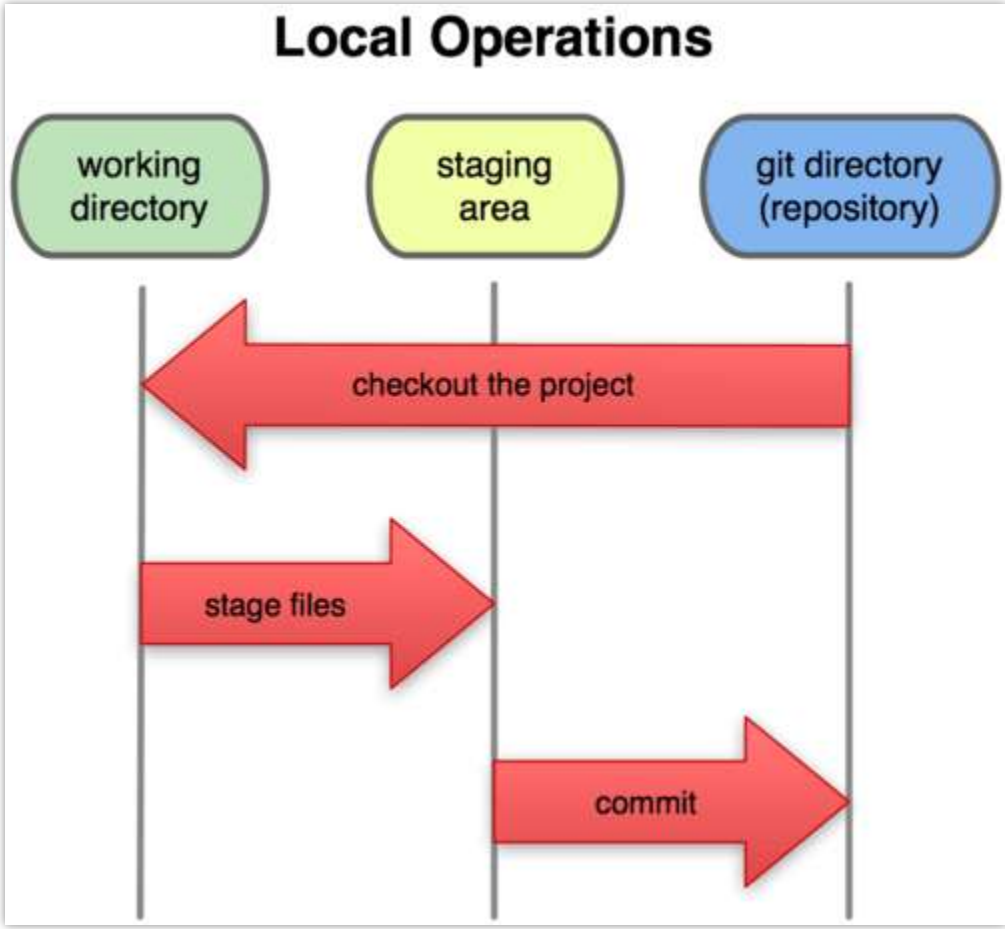
Snapshots, not differences

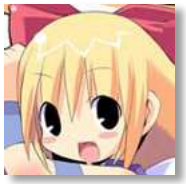
- Fast, and not stupid
- Capable as a mini file system



Staging

- Sometimes you changed many files but only some of them are ready to commit





How many you remember

commit, conflict, diff, merge, patch, repository (repo), staging, solve, update, …

Tips of git

- `git add` # add files to staging (then commit to repo)
- `git push` # push/sync the local repository to a remote repository
- `git checkout` # checkout/update files from repo to working
- `git pull` # pull/sync remote repository to the local repository and checkout
- `git diff` # diff two versions without checkout them first
- `git clone` # clone/copy an existing repo here
- `git help` # the most useful one

Demo

- A step-by-step for newbies?
  - Git - Book
  - Git and Github
  - 寫給大家的 Git 教學
  - A Visual Git Reference
- So I don’t want another one, and…

Use it now

or you never know git

Any Questions?

about git

GitHub?

a git server implementation

GitHub

- Okay, it should be the best git server implementation
- Do you know hub?
  - GitHub stores many projects
- It is a web service that uses git for project development
  - issue system, statistics, social elements…
- There are other git servers such as gitolite
  - useful when you want privacy and don't want to pay
- There are many step-by-step tutorials of GitHub online, so here let's not humiliate our intelligence

Today's assignment
今天的任務

Clone a GitHub repo

- You need to install git, register a GitHub account and clone an interesting repo. If you have no idea, 成電新手村 is a good choice. Try run nckuee-village on your server or your merry home. If you want to contribute (e.g. commit something), just let me know.
- Reference
  - Git
  - GitHub
- Your web site (http://merry.ee.ncku.edu.tw/~xxx/cur/, ex13) will be checked not before 23:59 1/14 (Tue). You may send a report (such as some important modifications) to me in case I did not notice your features. Or, even better, just explain your modifications in the homepage.

Appendix

Step-by-step of git environment

- Download git and install it (http://git-scm.com)
- Enter git console (many ways, in Windows 7, try [Start] → type git)
- Generate personal private/public keys
  - `ssh-keygen`
  - remember the location (in Windows 7, try C:\Users\xxx\.ssh)
  - id_rsa is your private key, keep it at local and NEVER spread it
  - id_rsa.pub is your public key (give it to the git server or project manager)
- Set your name and email
  - `git config --global user.name “xxx”`
  - `git config --global user.email “xxx”`

Clone a GitHub repo

- Enter git console
- Change to a folder that you store projects
  - `cd path/to/projects`
- Clone the repo
  - `git clone git@github.com:mbilab/nckuee-village.git`
  - or
  - `git clone https://github.com/mbilab/nckuee-village.git`
  - path/to/projects/nckuee-village/ will be your working directory

Create a GitHub repo

- Sign up at GitHub
- Setup your public key
  - [Account Setting] → [SSH Keys] → [Add SSH Key] → paste your public key in [Key]
- Create a GitHub repo: [Repositories] (repo in short) → [New]
- Enter git console
- Change to the working directory
  - `cd path/to/projects/project/`
- Initialize
  - `git init`
  - `git remote add origin [URL of the GitHub repo]`
- Create the first version, namely commit something
  - `git add xxx`
  - `git commit -m “comment of this commit”`
- Push it to the server (GitHub), and you can see it on GitHub
  - `git push origin master`