Git 101

DINNGO Software Engineer
許哲維

Who am I?

- DINNGO Software Engineer
- DINNGO is a blockchain related company
  - DINNGO Exchange
  - DeFlast
  - FURUCOMBO

Agenda

- What is git and why we should use git
- Start using the git
- Log your version
- Work with others
- Git flow

What is git and why we should use git

Git is a software for version control. It will snapshot what you committed.

And why we should use git:

What is git and why we should use git

Git is a software for version control. It will snapshot what you committed.

And why we should use git:

- Make your version control more convenient

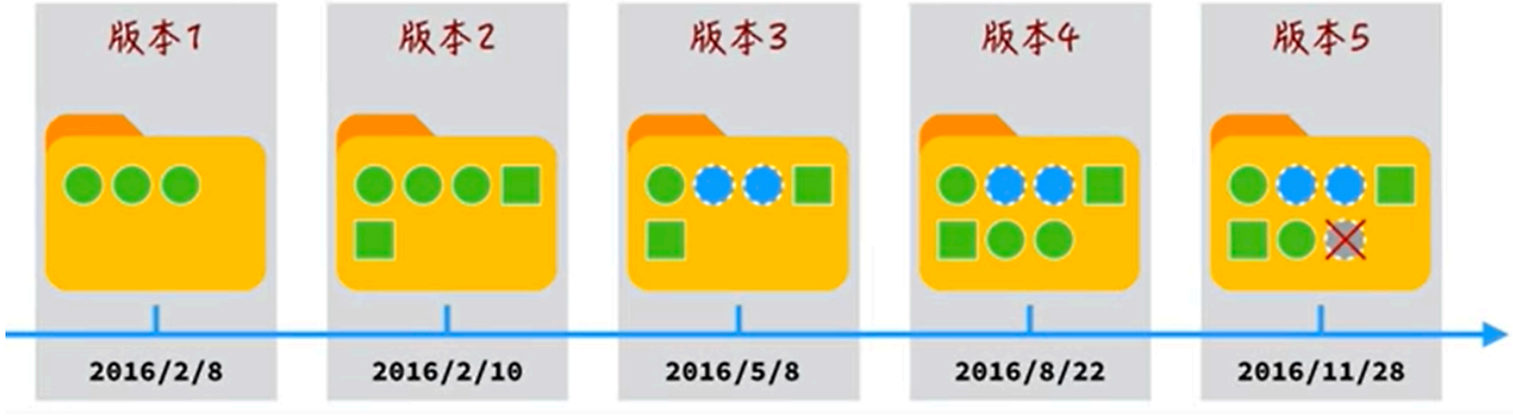
What is git and why we should use git

Git is a software for version control. It will snapshot what you committed.

And why we should use git:

- Make your version control more convenient

Without Git




```
ments/mcu_sharing_demo/without_git  ls -al

 weizard  staff  160 Sep  4 22:35 .
 weizard  staff  128 Sep  4 22:34 ..
 weizard  staff   64 Sep  4 22:35 demo_202209021
 weizard  staff   64 Sep  4 22:35 demo_202209022
 weizard  staff   64 Sep  4 22:35 demo_202209023
```

What is git and why we should use git

Git is a software for version control. It will snapshot what you committed.

And why we should use git:

- Make your version control more convenient

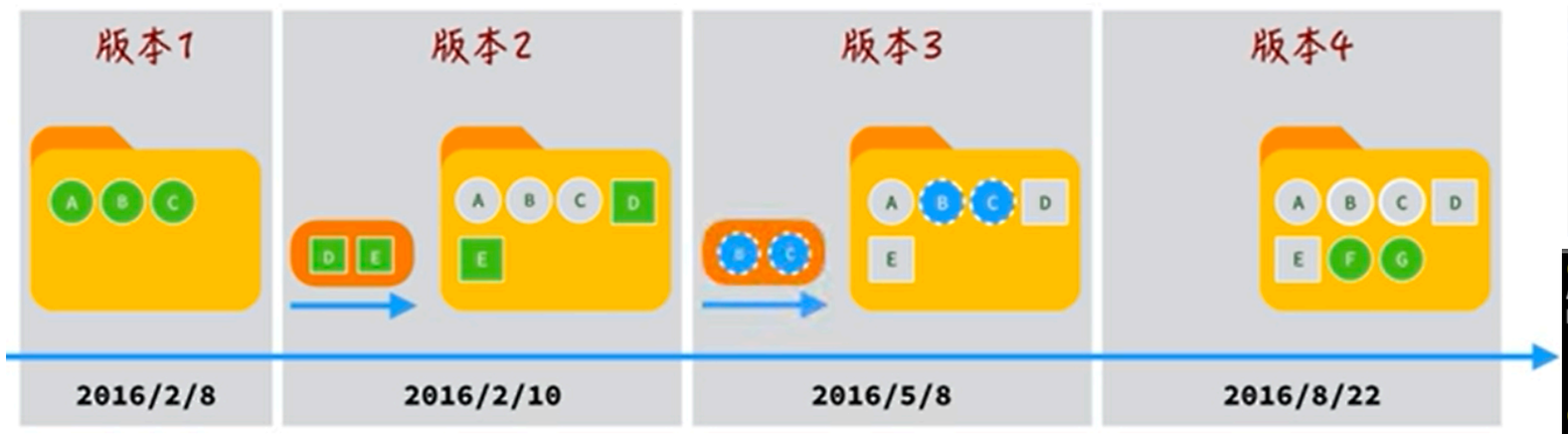
What is git and why we should use git

Git is a software for version control. It will snapshot what you committed.

And why we should use git:

- Make your version control more convenient

With Git




```
sharing_demo/with_git  git  master  ls -al

staff  128 Sep  4 22:38 .
staff  128 Sep  4 22:38 ..
staff  384 Sep  4 22:38 .git
staff   10 Sep  4 22:37 demo
```

```
commit 8db32853ba2049f6a2b88cb7ecfba1e8b766897d (HEAD -> master)
Author: weizard <purpledoor4921@gmail.com>
Date:   Sun Sep 4 22:38:00 2022 +0800

    feat: demo v3

commit 190f8aabb42e2242df03ef39a4a5e5f970a9027c
Author: weizard <purpledoor4921@gmail.com>
Date:   Sun Sep 4 22:37:41 2022 +0800

    feat: demo v2

commit 3aafd2169b964022bf025d1f0f5b38475a7988ea
Author: weizard <purpledoor4921@gmail.com>
Date:   Sun Sep 4 22:37:07 2022 +0800
```

What is git and why we should use git

Git is a software for version control. It will snapshot what you committed.

And why we should use git:

- Make your version control more convenient

What is git and why we should use git

Git is a software for version control. It will snapshot what you committed.

And why we should use git:

- Make your version control more convenient
- Work with others more effectively

What is git and why we should use git

Git is a software for version control. It will snapshot what you committed.

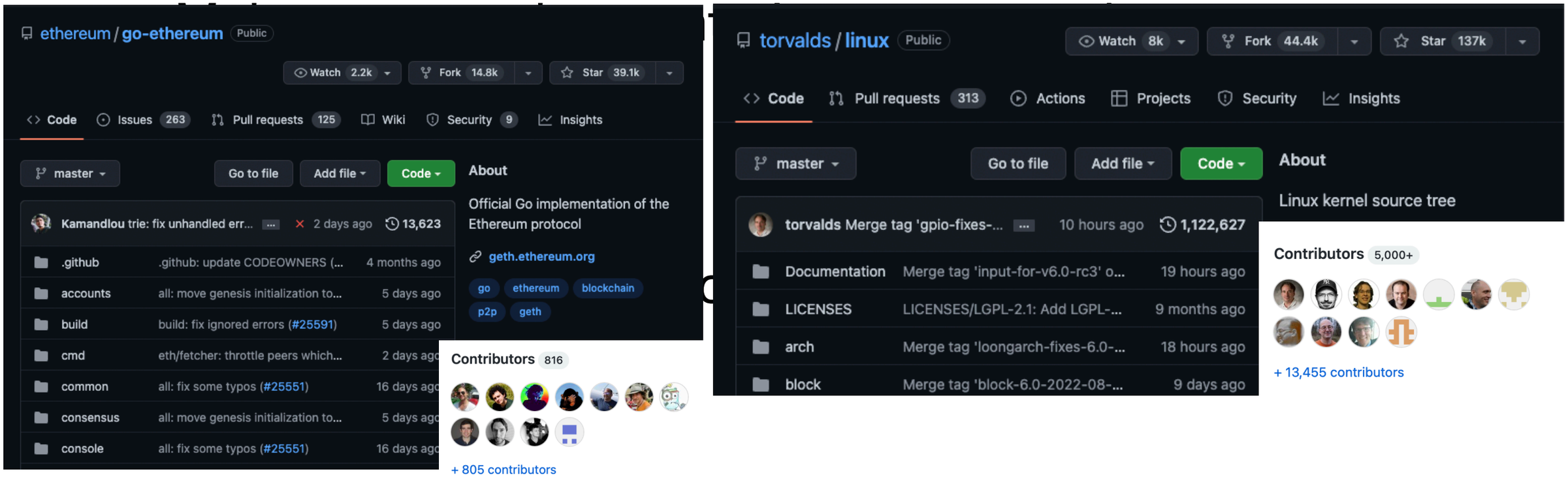
And why we should use git:

- Make your version control more convenient
- Work with others more effectively
- There is a lot of resource on GitHub/ GitLab …etc.

What is git and why we should use git

Git is a software for version control. It will snapshot what you committed.

And why we should use git:



Start using the git

Start using the git

- git init

Start using the git

- git init

```
~/Documents/mcu_sharing_demo ls -al
total 0
drwxr-xr-x@  5 weizard  staff  160 Sep 11 11:48 .
drwx------@ 27 weizard  staff  864 Sep 11 11:48 ..
drwxr-xr-x@ 19 weizard  staff  608 Sep  4 23:09 gotyour.pw
drwxr-xr-x@  4 weizard  staff  128 Sep 11 11:01 with_git
drwxr-xr-x@  5 weizard  staff  160 Sep  4 22:35 without_git
```

```
~/Doc/mcu_sharing_demo git master ?2 ls -al
total 0
drwxr-xr-x@  6 weizard  staff  192 Sep 11 11:49 .
drwx------@ 27 weizard  staff  864 Sep 11 11:48 ..
drwxr-xr-x@  9 weizard  staff  288 Sep 11 11:49 .git
drwxr-xr-x@ 19 weizard  staff  608 Sep  4 23:09 gotyour.pw
drwxr-xr-x@  4 weizard  staff  128 Sep 11 11:01 with_git
drwxr-xr-x@  5 weizard  staff  160 Sep  4 22:35 without_git
```

Start using the git

- git init

Start using the git

- git init

...or create a new repository on the command line

```
echo "# mcu_sharing" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:weizard/mcu_sharing.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:weizard/mcu_sharing.git
git branch -M main
git push -u origin main
```

Start using the git

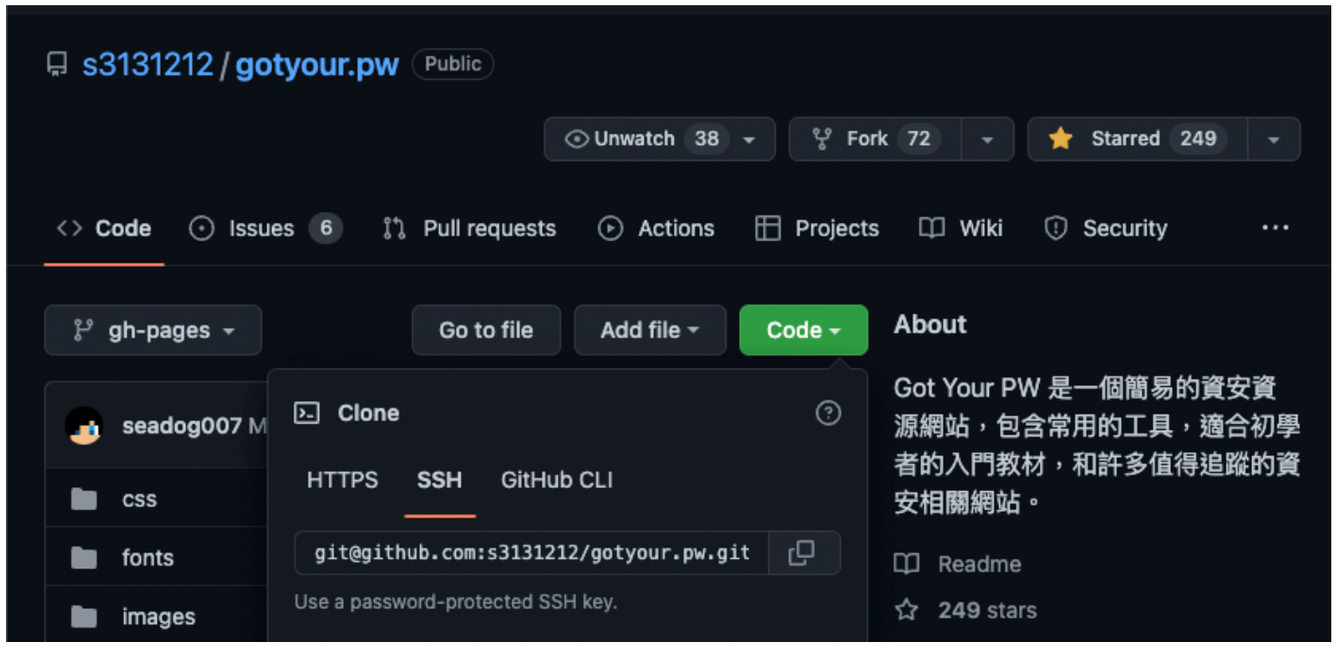
- git init

Start using the git

- git init
- git clone

Start using the git

- git init
- git clone

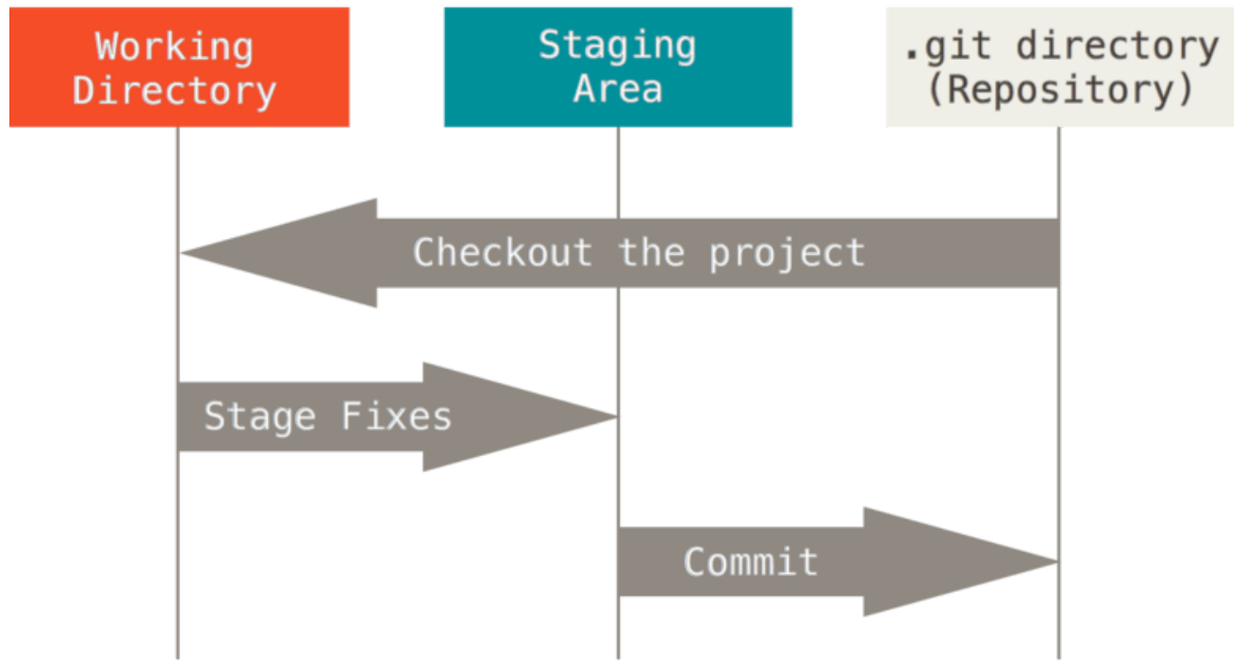



```
~/Documents/mcu_sharing_demo git clone git@github.com:s3131212/gotyour.pw.git
Cloning into 'gotyour.pw'...
remote: Enumerating objects: 596, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 596 (delta 11), reused 0 (delta 0), pack-reused 574
Receiving objects: 100% (596/596), 1.49 MiB | 1.38 MiB/s, done.
Resolving deltas: 100% (305/305), done.
~/Documents/mcu_sharing_demo ls
gotyour.pw  with_git    without_git
```

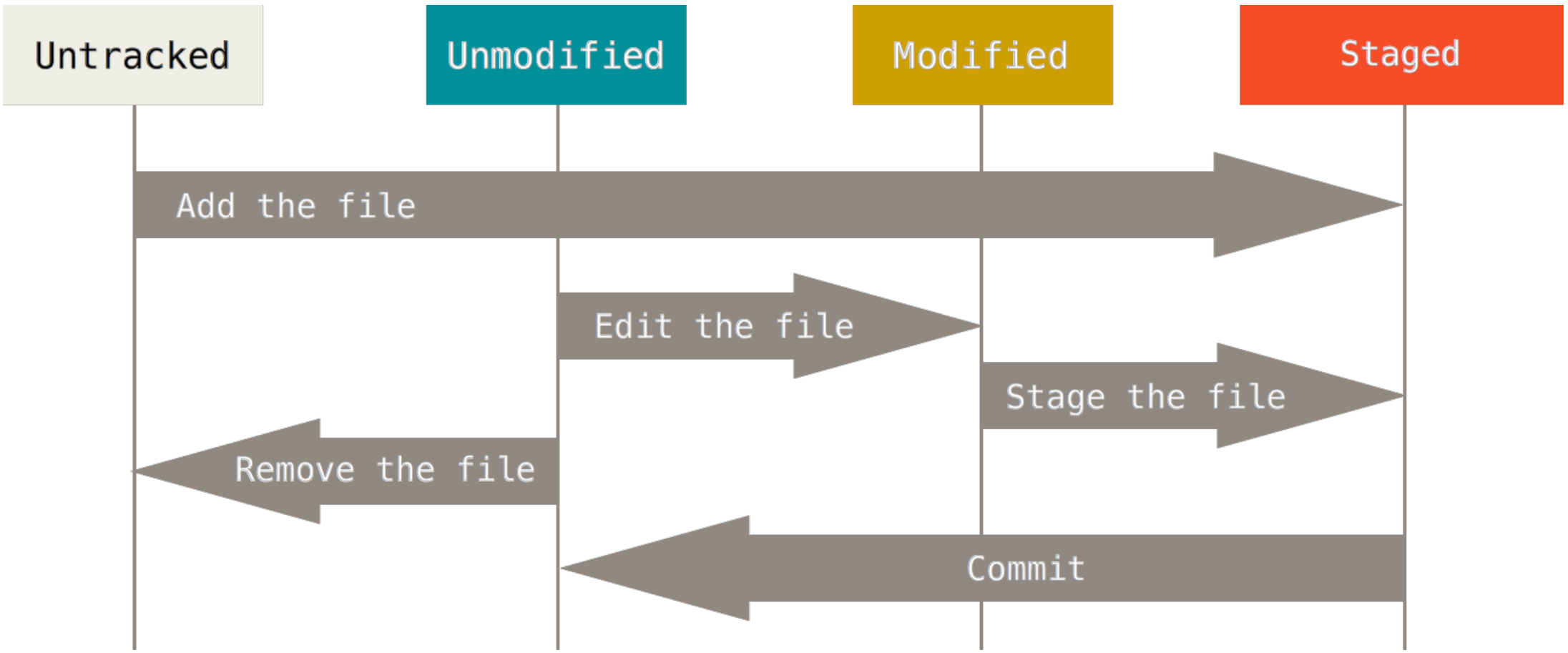
Log your version

The three state

- Working Directory (modified)
  - The files which you are editing.
- Staging Area (staged)
  - Stored the files which you prepare to take the snapshot.
- .git directory (committed)
  - All the snapshot are keeping at here.



The lifecycle of the status of your files



How to check your files status

- git status
  - It will tell your current branch
  - Provide each files status
  - Give you some hint

```
rueian@RueianMBPR:gui [master !?+]$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   topology_normal.temp
        new file:   topology_small.temp

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   index.html
        modified:   script/app/neutron.js
        modified:   script/app/vos.js

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        api/
        script/.DS_Store
        script/directive/.DS_Store
        script/directive/topology.js
        view/admin/neutron/.DS_Store
        view/admin/neutron/networks/topology.html
        view/general/muses/topology/
```

Staged your files

Staged your files

- git add

Staged your files

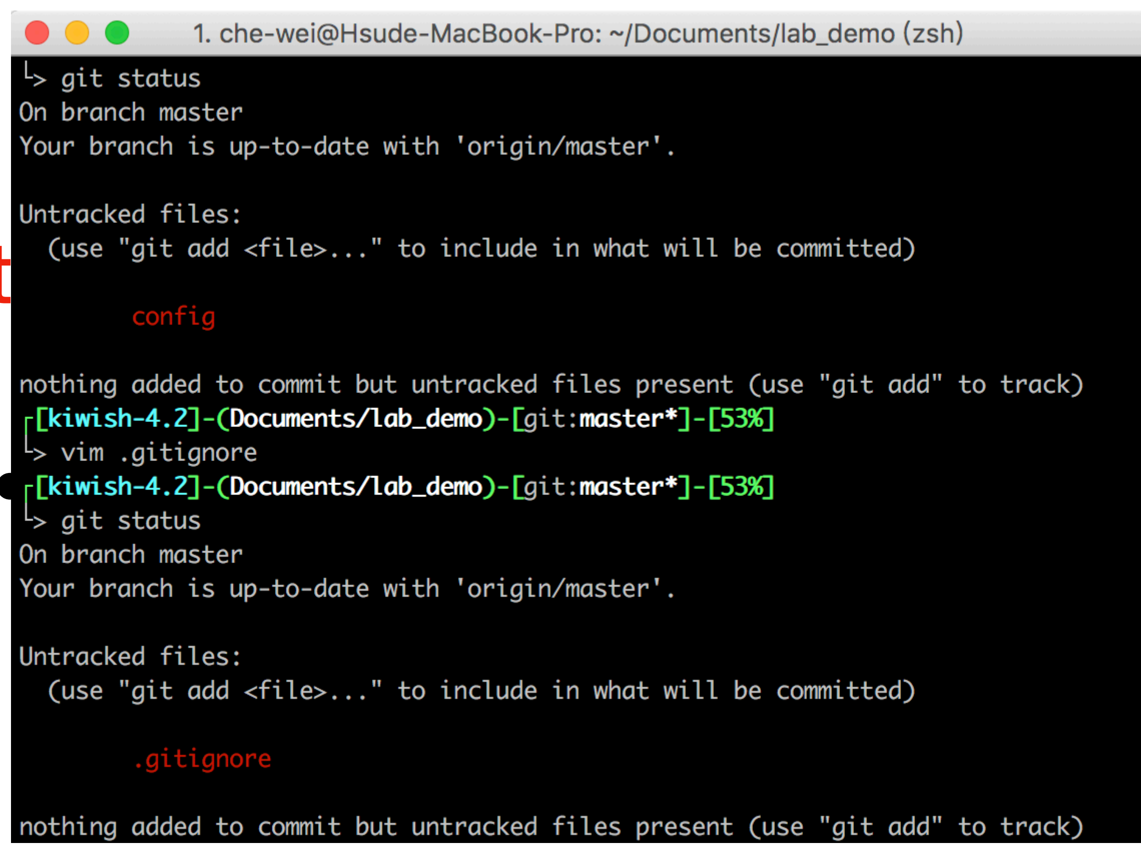
- git add
- .gitignore

Staged your files

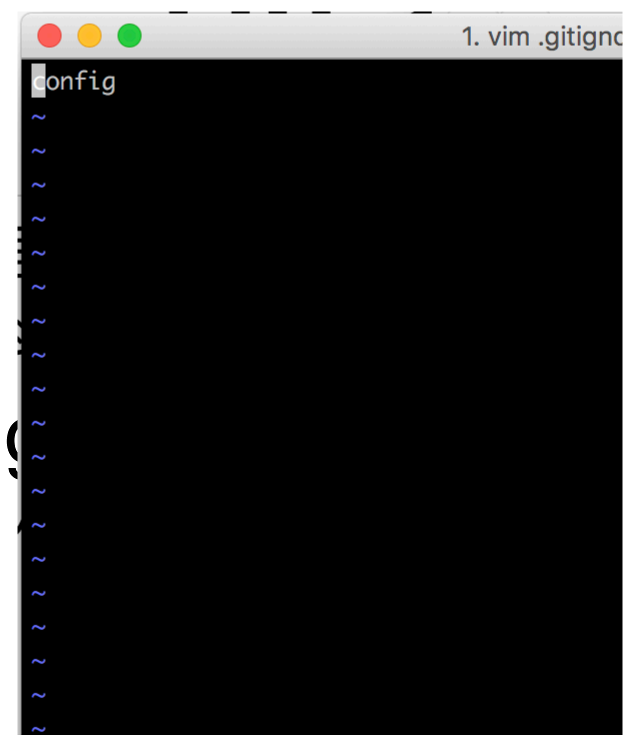
- git add
- .gitignore
    - Set your sensitive files into .gitignore to protect them.

Staged your files

- git
- .git



gitig



Staged your files

- git add
- .gitignore
    - Set your sensitive files into .gitignore to protect them.

Committed your files

- git commit
- git commit -m `set your commit msg`
  default will open your default editor let you type down your msg.

- git commit —amend
  modified your previous commit msg

Additional info about commit

- Give a prefix for your commit. git commit type include:
    - feat: 新增/修改功能 (feature)
    - fix: 修補 bug (bug fix)
    - docs: 文件 (documentation)
    - style: 格式 (不影響程式碼運行的變動 white-space, formatting, missing semi colons, etc)
    - refactor: 重構 (既不是新增功能，也不是修補 bug 的程式碼變動)
    - perf: 改善效能 (A code change that improves performance)
    - test: 增加測試 (when adding missing tests)
    - chore: 建構程序或輔助工具的變動 (maintain)
    - revert: 撤銷回覆先前的 commit 例如：revert: type(scope): subject (回覆版本：xxxx)

Additional info about commit
Cont.

- 該如何寫好 git commit message
    - 將標題與內容中間多一行空白
    - 標題限制 50 字元
    - 標題第一個字必須為大寫
    - 標題最後不要帶上句號
    - 標題內容可以使用強烈的語氣
    - 內容請用 72 字元來斷行
    - 內容可以解釋 what and why vs. how
- Ref:
    - https://blog.wu-boy.com/2015/09/how-to-write-git-commit-message/
    - https://wadehuanglearning.blogspot.com/2019/05/commit-commit-commit-why-what-commit.html

If you staged wrong file

- git reset
    - git reset --hard (danger!!)
- git restore (new cmd from 2.23.0)
    - More easy for use and understand, if you staged it will move back to work directory. And it your file at working directory will go back to last commit !! So you should do this more carefully! Or with ` --staged` opetion.

Check your commit history

- git log
  - git log --graph

```
commit 20f04738b9ba25a3fa4bff7c4da9476d0a0de596 (HEAD
Author: weizard <purpledoor4921@gmail.com>
Date:   Sun Sep 11 12:33:05 2022 +0800

    write down the commit msg

commit 8db32853ba2049f6a2b88cb7ecfba1e8b766897d
Author: weizard <purpledoor4921@gmail.com>
Date:   Sun Sep 4 22:38:00 2022 +0800

    feat: demo v3

commit 190f8aabb42e2242df03ef39a4a5e5f970a9027c
Author: weizard <purpledoor4921@gmail.com>
Date:   Sun Sep 4 22:37:41 2022 +0800

    feat: demo v2

commit 3aafd2169b964022bf025d1f0f5b38475a7988ea
Author: weizard <purpledoor4921@gmail.com>
Date:   Sun Sep 4 22:37:07 2022 +0800

    feat: demo v1
(END)
```

Check your commit history

- git log
  - git log --gr

```
* commit 7558c62288c72fcae4537498f41e268549f1aedb (HEAD -> feature/wallet_modal_add_total_val
ue_20220906, origin/feature/wallet_modal_add_total_value_20220906)
| Author: Jay Hsu <jay@dinngo.co>
| Date:   Tue Sep 6 12:19:23 2022 +0800
|
|     feat: wallet modal support total value display
|
*   commit 510147edb7fcae99fc14ee45494bd6b8fd41652d (develop)
|\  Merge: 4671ec910 2874d29ce
| | Author: Jay Hsu <jay@dinngo.co>
| | Date:   Tue Sep 6 06:31:54 2022 +0000
| |
| |     Merge branch 'feature/update_some_package_verison_20220905' into 'develop'
| |
| |     chore: update 3 packages
| |
| |     See merge request furucombo/furucombo-interface!363
| |
| * commit 2874d29ceef4509f877c8e463fad8fdcef077f54 (origin/feature/update_some_package_veris
on_20220905, feature/update_some_package_verison_20220905)
| | Author: Jay Hsu <jay@dinngo.co>
| | Date:   Mon Sep 5 16:50:32 2022 +0800
| |
| |     chore: update 3 packages
| |
* |   commit 4671ec910c40e118e99f07d01041ea1a8f2b8def
|\ \  Merge: 75481f754 a799314d1
| |/  Author: Jay Hsu <jay@dinngo.co>
|/|   Date:   Tue Sep 6 06:31:43 2022 +0000
| |
| |       Merge branch 'feature/fund_support_smart_wallet_20220802' into 'develop'
| |
| |       feat: init fund supoort smart wallet commit
| |
| |       See merge request furucombo/furucombo-interface!360
| |
| *   commit a799314d1e16a0c0aaccfa6aa8e087e1b7f08d7c (origin/feature/fund_support_smart_wall
et_20220802, feature/fund_support_smart_wallet_20220802)
| |\  Merge: f958db25f 75481f754
| |/  Author: Jay Hsu <jay@dinngo.co>
|/|   Date:   Tue Sep 6 12:22:26 2022 +0800
| |
| |       Merge branch 'develop' into feature/fund_support_smart_wallet_20220802
| |
* | commit 75481f754aff4ec670b02e6ddf6f3c66a9f05155
| | Author: Bob Lu <bob@dinngo.co>
| | Date:   Mon Sep 5 10:23:30 2022 +0800
| |
| |     fix: replace url of recruit fund manager
| |
* |   commit 12eb699a9a22e2963fa6fe5e15c673acd1f88a02
:
```

```
f7c4da9476d0a0de596 (HEAD
921@gmail.com>
2022 +0800

g

cb7ecfba1e8b766897d
921@gmail.com>
022 +0800

f39a4a5e5f970a9027c
921@gmail.com>
022 +0800

d1f0f5b38475a7988ea
921@gmail.com>
022 +0800
```

Check your commit history

- git log
  - git log --graph

```
commit 20f04738b9ba25a3fa4bff7c4da9476d0a0de596 (HEAD
Author: weizard <purpledoor4921@gmail.com>
Date:   Sun Sep 11 12:33:05 2022 +0800

    write down the commit msg

commit 8db32853ba2049f6a2b88cb7ecfba1e8b766897d
Author: weizard <purpledoor4921@gmail.com>
Date:   Sun Sep 4 22:38:00 2022 +0800

    feat: demo v3

commit 190f8aabb42e2242df03ef39a4a5e5f970a9027c
Author: weizard <purpledoor4921@gmail.com>
Date:   Sun Sep 4 22:37:41 2022 +0800

    feat: demo v2

commit 3aafd2169b964022bf025d1f0f5b38475a7988ea
Author: weizard <purpledoor4921@gmail.com>
Date:   Sun Sep 4 22:37:07 2022 +0800

    feat: demo v1
(END)
```

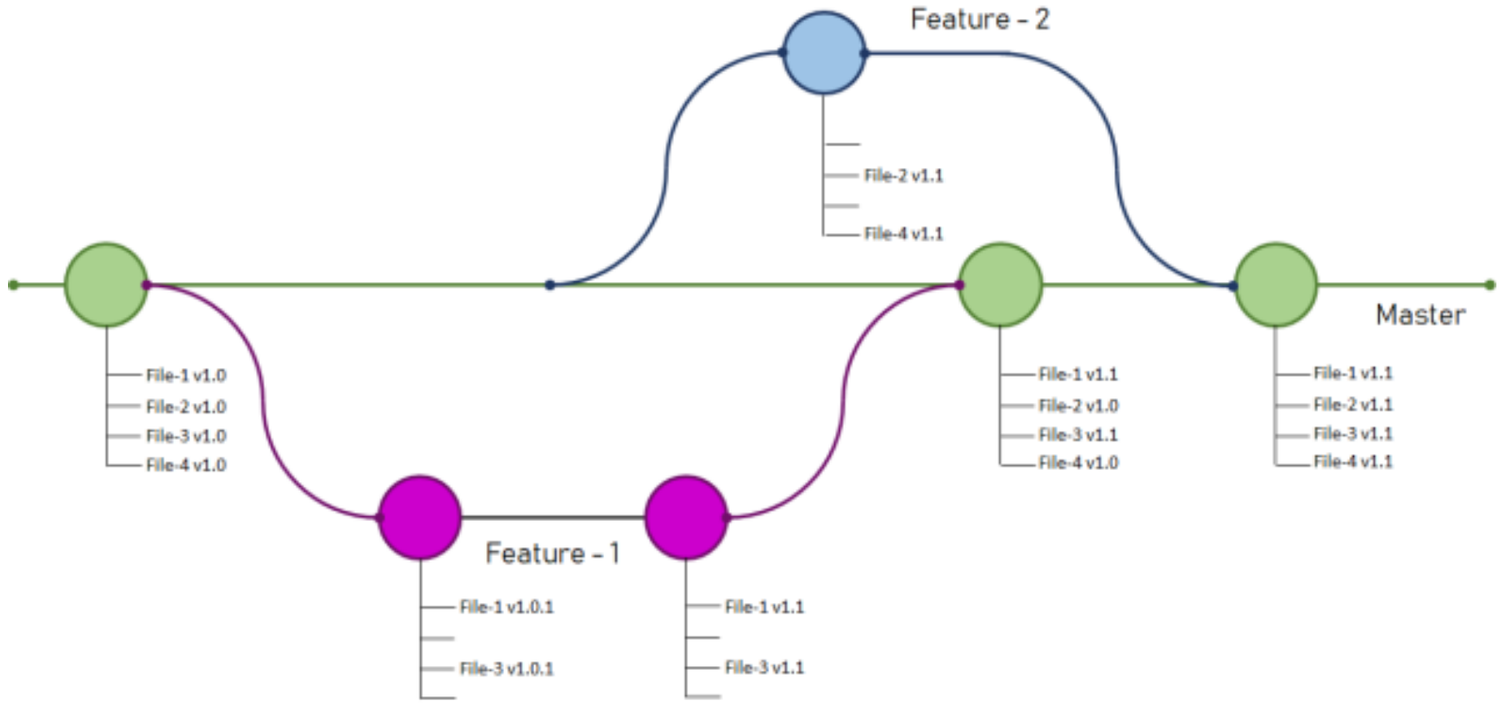
Work with others

Work with remote

- git remote -v
    - Remote means your git server. (i.g: GitHub,GitLab)
- git push
    - Push your commit to remote
- git pull
    - Pull the latest commit from remote

Brach

- git branch



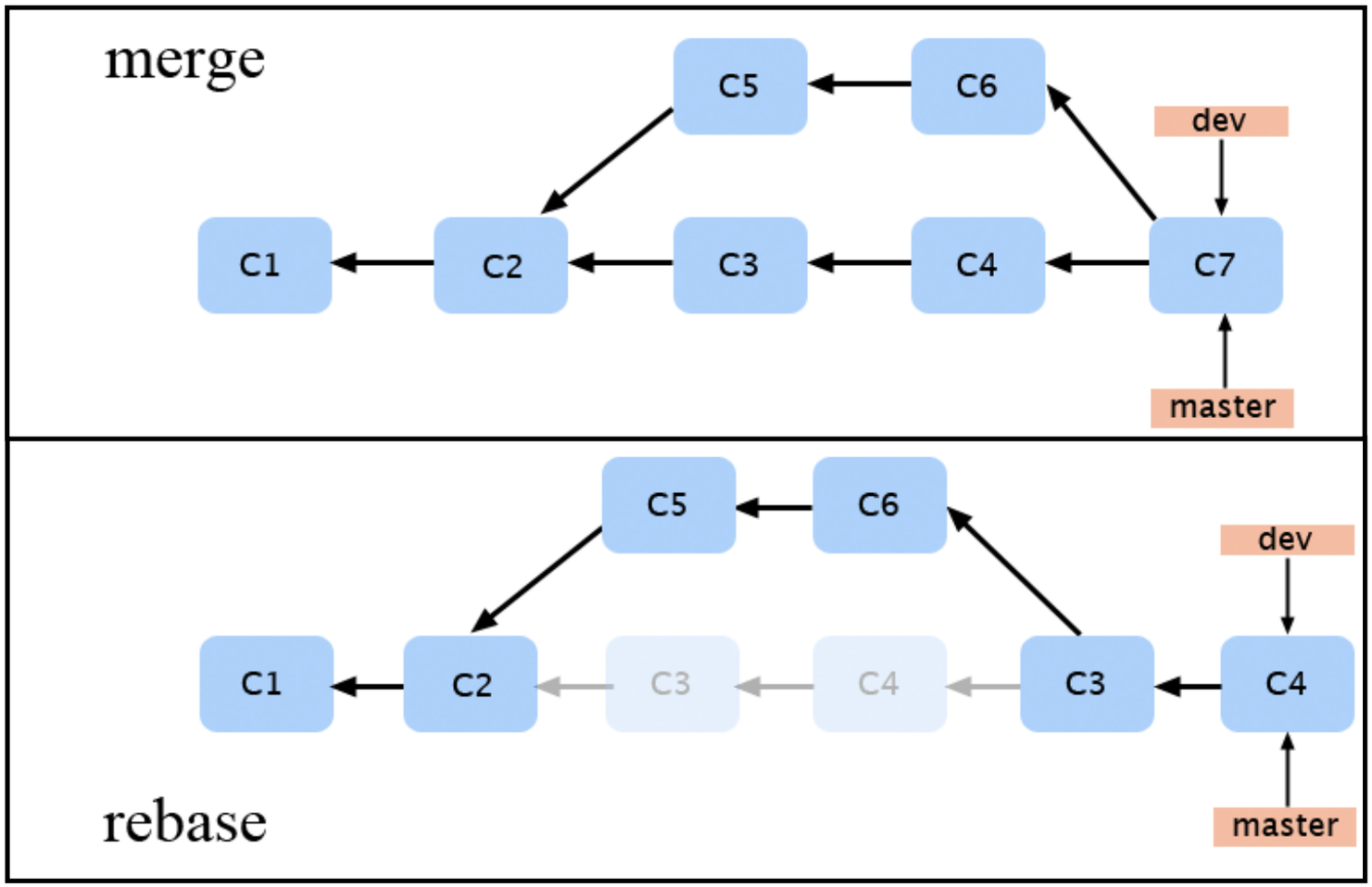
Switch to the branch you want

- git checkout <branch name>
    - git checkout -b <branch name>
      create a new branch and switch to this branch

- git checkout <file name>
    - Make file recover to the latest commit status.

Merge your code with your partner

- git merge
- git rebase
  - Use this method carefully, if with wrong way you can’t find the commit anymore.
- git rebase -i combine commit



Conflict

- git mergetool

```
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   demo

no changes added to commit (use "git add" and/or "git commit -a")
```

Conflict

- git mergetool

```
1 <<<<<<< HEAD
2 become v3.2
3 =======
4 become v3.1
5 >>>>>>> feature/make_conflict_20220911
6
7 test
```

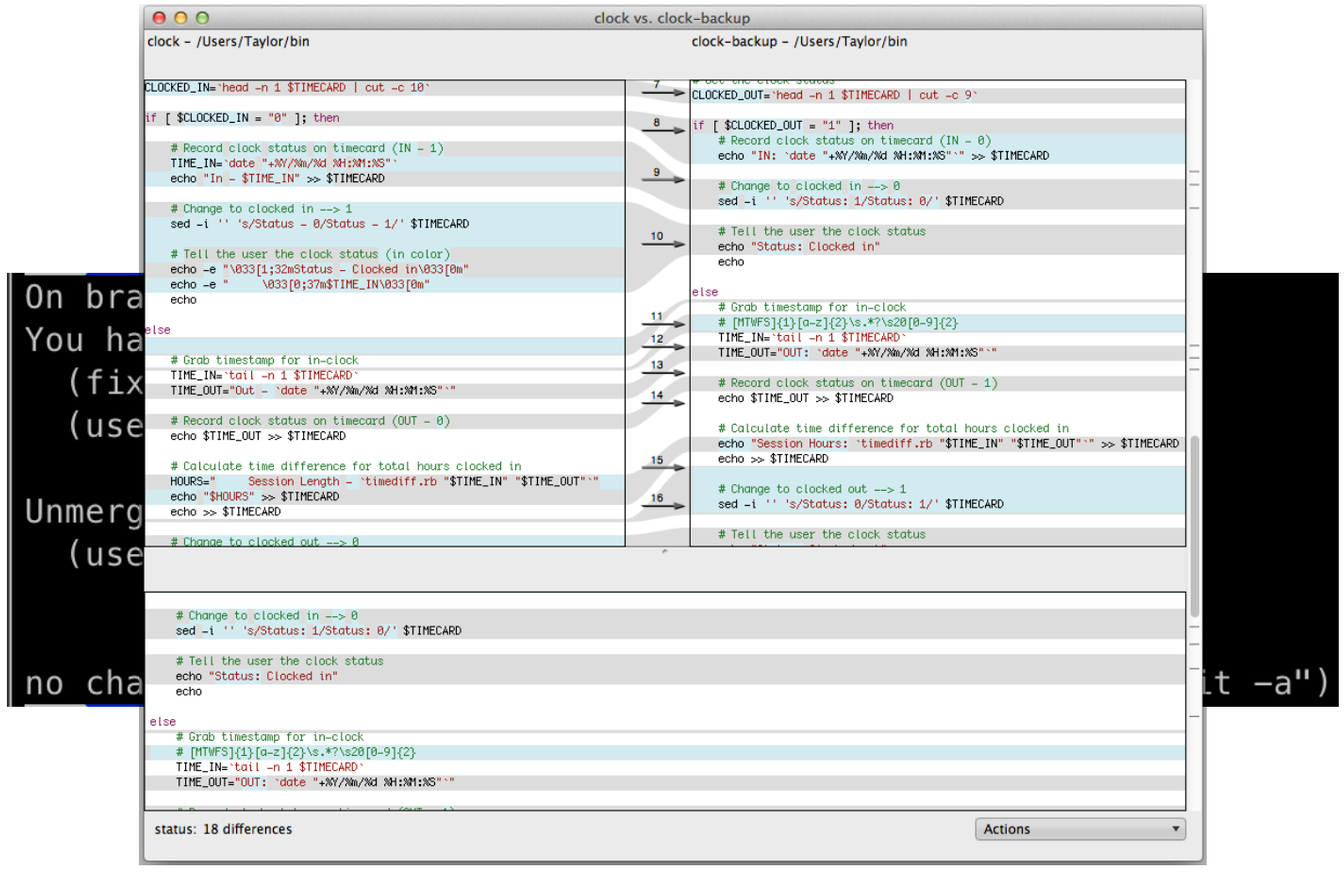
Conflict

- git mergetool

```
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   demo

no changes added to commit (use "git add" and/or "git commit -a")
```

Conflict

- git mergetool



Conflict

- git mergetool

```
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   demo

no changes added to commit (use "git add" and/or "git commit -a")
```

Conflict

- git mergetool

```
Merge branch 'feature/make_conflict_20220911'
#
# Conflicts:
#       demo
#
# It looks like you may be committing a merge.
# If this is not correct, please run
#       git update-ref -d MERGE_HEAD
# and try again.


# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# All conflicts fixed but you are still merging.
#
```

Conflict

- git mergetool

```
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   demo

no changes added to commit (use "git add" and/or "git commit -a")
```

Combine your commit

- git rebase -i

Combine your commit

- git rebase -i

```
commit 20f04738b9ba25a3fa4bff7c4da9476d0a0de596 (HEAD -> master)
Author: weizard <purpledoor4921@gmail.com>
Date:   Sun Sep 11 12:33:05 2022 +0800

    write down the commit msg

commit 8db32853ba2049f6a2b88cb7ecfba1e8b766897d
Author: weizard <purpledoor4921@gmail.com>
Date:   Sun Sep 4 22:38:00 2022 +0800

    feat: demo v3

commit 190f8aabb42e2242df03ef39a4a5e5f970a9027c
Author: weizard <purpledoor4921@gmail.com>
Date:   Sun Sep 4 22:37:41 2022 +0800

    feat: demo v2

commit 3aafd2169b964022bf025d1f0f5b38475a7988ea
Author: weizard <purpledoor4921@gmail.com>
Date:   Sun Sep 4 22:37:07 2022 +0800

    feat: demo v1
```

Combine your commit

- git rebase -i

Combine your commit

- git rebase -i

```
1 pick 190f8aa feat: demo v2
2 pick 8db3285 feat: demo v3
3 squash 20f0473 write down the commit msg
4
5 # Rebase 3aafd21..20f0473 onto 3aafd21 (3 commands)
6 #
7 # Commands:
8 # p, pick <commit> = use commit
9 # r, reword <commit> = use commit, but edit the commit message
10 # e, edit <commit> = use commit, but stop for amending
11 # s, squash <commit> = use commit, but meld into previous commit
12 # f, fixup [-C | -c] <commit> = like "squash" but keep only the previous
13 #                    commit's log message, unless -C is used, in which case
14 #                    keep only this commit's message; -c is same as -C but
15 #                    opens the editor
16 # x, exec <command> = run command (the rest of the line) using shell
17 # b, break = stop here (continue rebase later with 'git rebase --continue')
18 # d, drop <commit> = remove commit
19 # l, label <label> = label current HEAD with a name
20 # t, reset <label> = reset HEAD to a label
21 # m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
22 # .       create a merge commit using the original merge commit's
23 # .       message (or the oneline, if no original merge commit was
24 # .       specified); use -c <commit> to reword the commit message
```

Combine your commit

- git rebase -i

Combine your commit

- git rebase -i

```
1 # This is a combination of 2 commits.
2 # This is the 1st commit message:
3
4 feat: demo v3
5
6 # This is the commit message #2:
7
8 write down the commit msg
9
```

Combine your commit

- git rebase -i

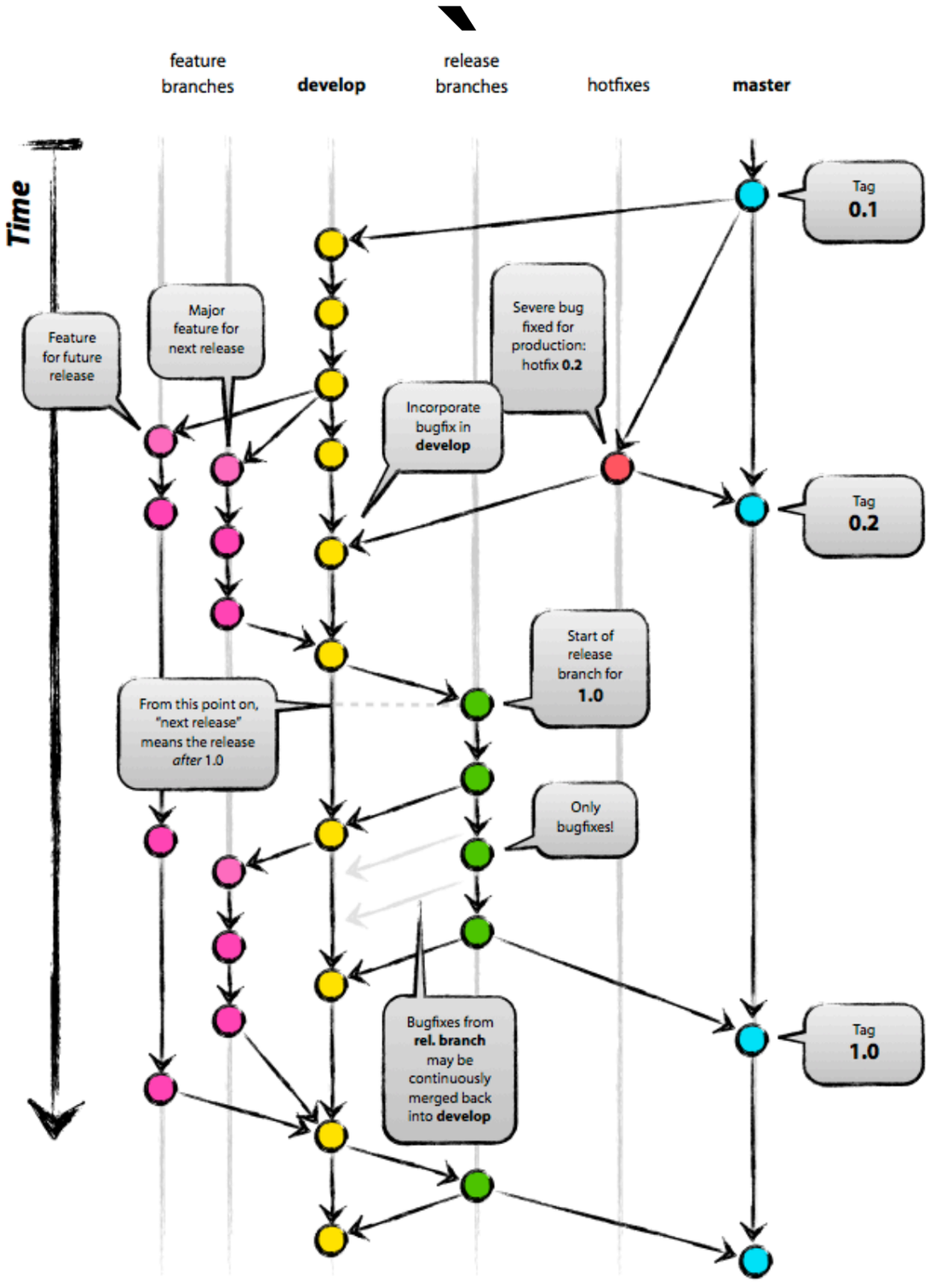
Combine your commit

- git rebase -i

```
commit 4395eda95d270c81498740d149ee29a35753abfb (HEAD -> master)
Author: weizard <purpledoor4921@gmail.com>
Date:   Sun Sep 4 22:38:00 2022 +0800

    feat: this commit is after squash

commit 190f8aabb42e2242df03ef39a4a5e5f970a9027c
Author: weizard <purpledoor4921@gmail.com>
Date:   Sun Sep 4 22:37:41 2022 +0800

    feat: demo v2

commit 3aafd2169b964022bf025d1f0f5b38475a7988ea
Author: weizard <purpledoor4921@gmail.com>
Date:   Sun Sep 4 22:37:07 2022 +0800

    feat: demo v1
(END)
```

git flow

- 主要分支
  - master : 永遠在 production-ready的狀態。
  - developer : 主要的開發線。
- 輔助分支
  - release : 開發完成後進行測試及修bug的版本，只會merge 回master 跟 developer。
  - hotfix : 緊急搶修用，將問題解決後也只會merge 回master 跟 developer。
  - feature : 用來開發新功能，完成後merge 至 developer 上。


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Additional

- Markdown
- github.io
- Gitbook
- CI
  - GitHub Action
  - GitLab CI
  - Travis

Q&A