Version Control System - Git

lwhsu (2019, CC-BY)

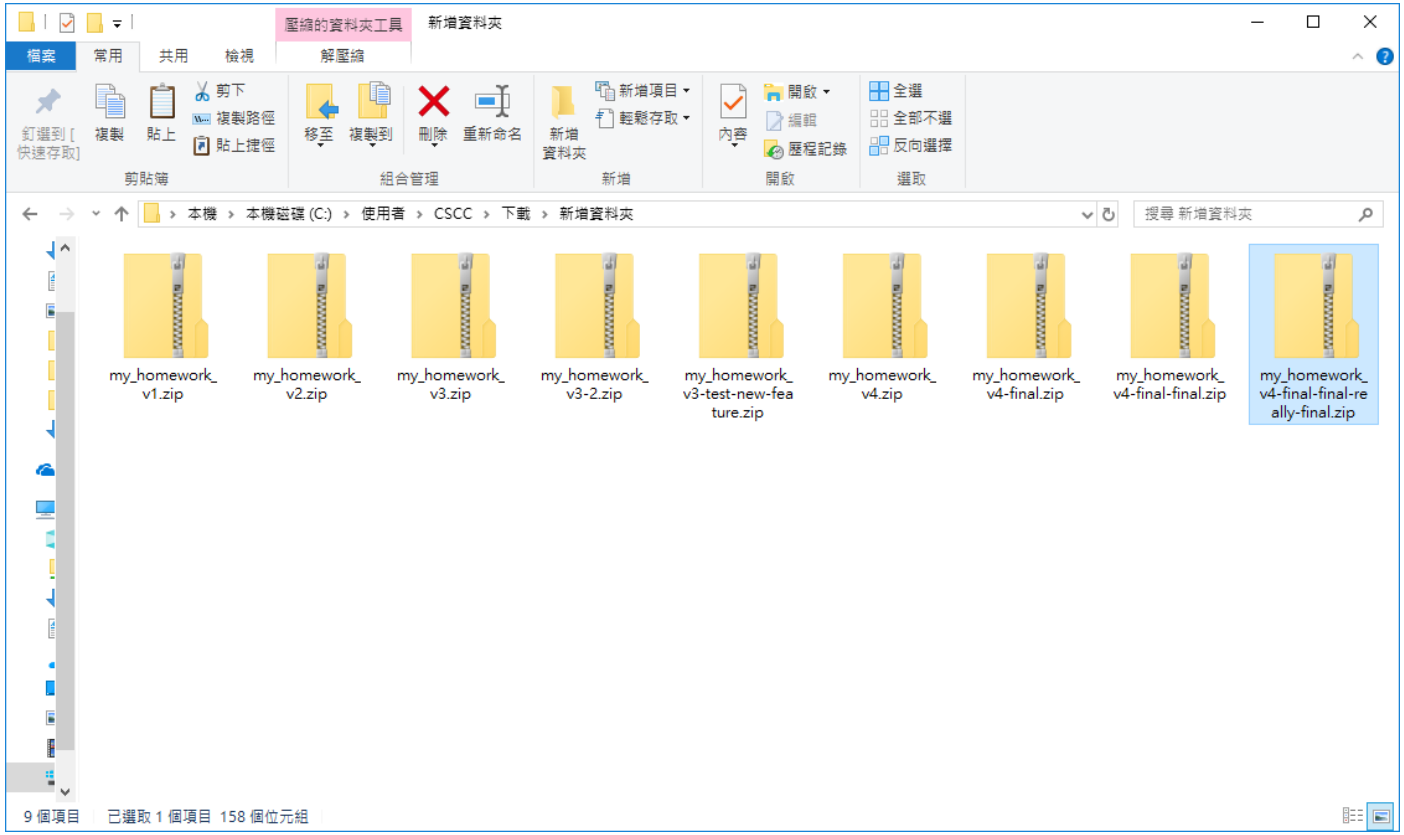
zswu (2018)

Outline

- Why VCS ?
- Why Git ?
- Using Git Personally
- Using Git with Others
- Etiquette Rules of Using Git
- Tools & Services
- Tips

Why VCS (1/3)

- How do you manage your homework ?



Why VCS (2/3)

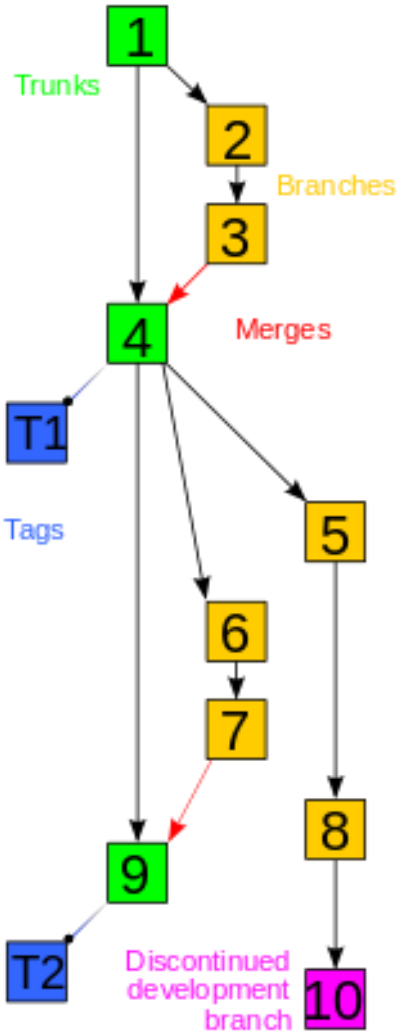
- How did people do version control ?
  - Share files or archivers through CD, USB or Network
  - Patch
- What's wrong with this?
  - Hard to maintain & debug
  - Hard for integrity checking
  - Hard for collaboration
  - Bad thing always happens

Why VCS (3/3)

- Version Control System
- A tool helps you keep tracking your project history
- A tool helps you collaboration with others
- Common VCS
  - RCS/CVS
  - Subversion (SVN)
  - Mercurial (hg)
  - Fossil
  - Bazaar (bzr)
  - Git
  - … https://en.wikipedia.org/wiki/Comparison_of_version-control_software

Version Control System

- Common Concepts
  - Repository
  - Working Copy (or directory/tree)
- Common Operations
  - Check in
  - Check out
  - Update
  - Merging / Resolving conflict
  - Locking
  - Labeling



https://zh.wikipedia.org/wiki/File:Revision_controlled_project_visualization-2010-24-02.svg

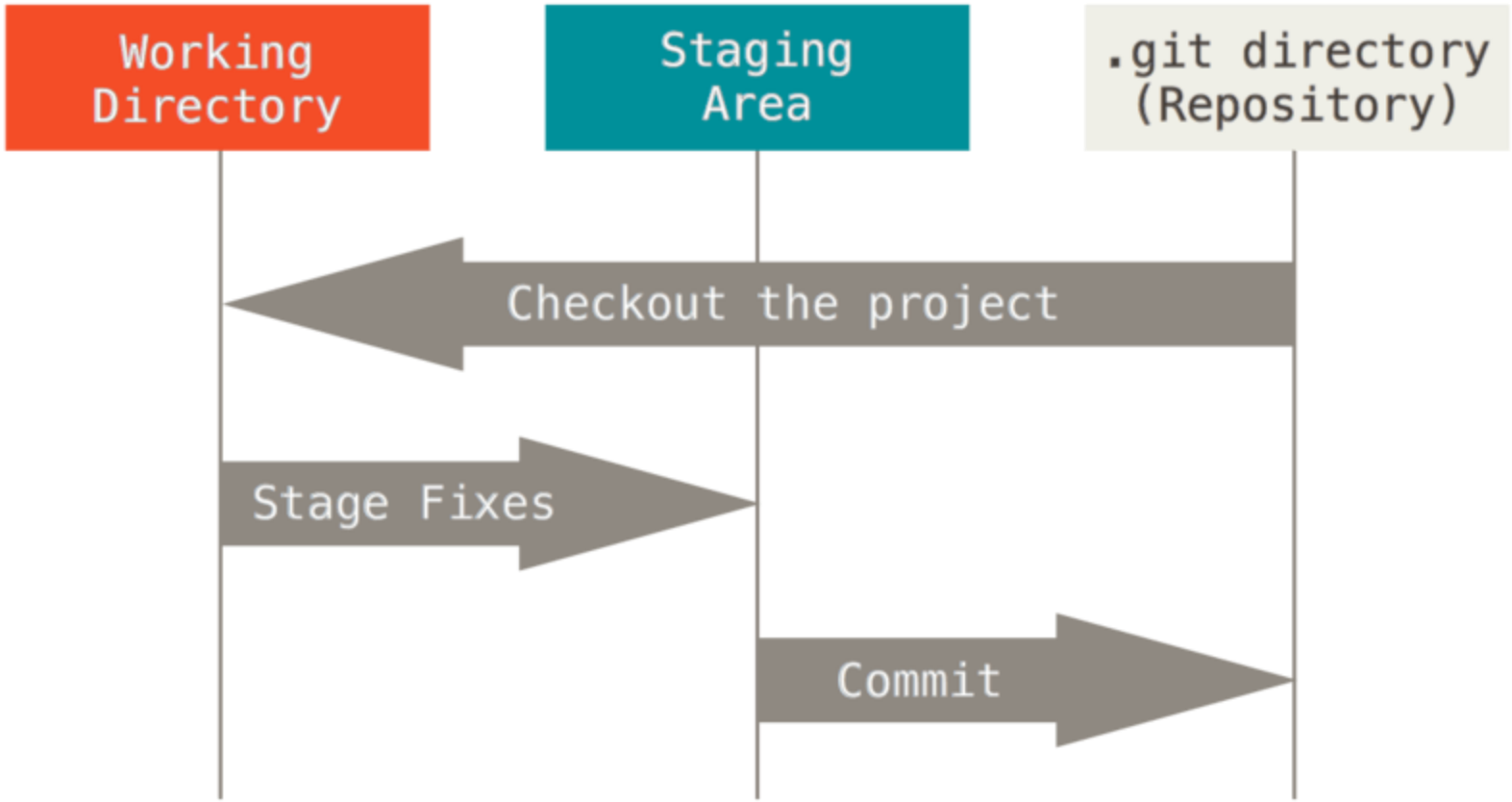
Git

- NOT a software only for you to download codes from Github
- Distributed version control system (DVCS)
  - P2P instead of Client-Server
  - Fast & convenient
    - Everything is at local and most of the operations don't need to connect to the remote server

https://xkcd.com/1597/
https://explainxkcd.com/wiki/index.php/1597:_Git

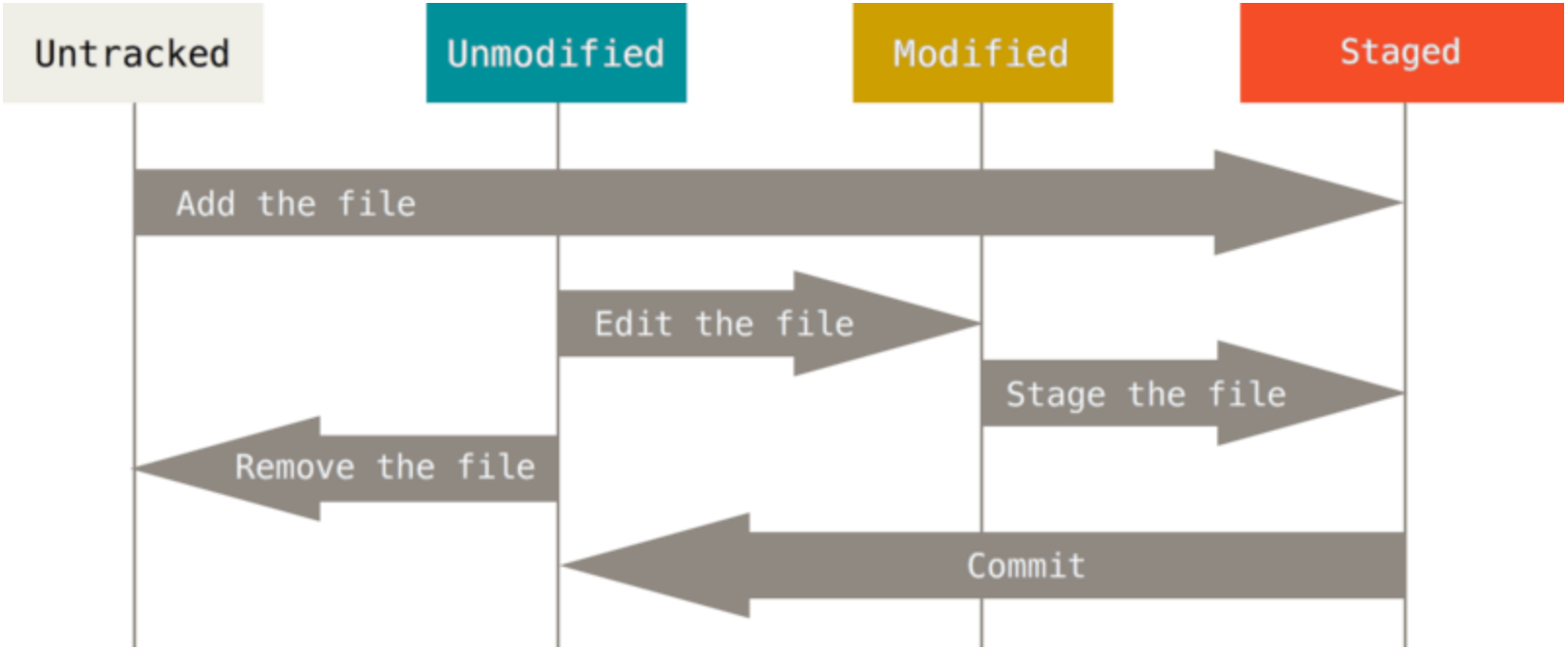
Git: the three sections & states

- Modified, Staged, Committed



https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F

Lifecycle of the status of the files



https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository

Using Git Personally

- Help
  - `` `git help` ``
- Base version control
  - init
  - status
  - add, commit
  - reset
- View history
  - log
  - blame
- Develop different feature
  - branch, tag, checkout
  - rebase, merge

Config

- Set basic config before using git
  - Position
    - System (/etc/gitconfig or /usr/local/etc/gitconfig)
    - Personal (~/.gitconfig or ~/.config/git/config)
    - Repository (project-root/.git/config)

    Try: `` `truss git 2>&1 | grep config` ``
  - Basic config
    - $ git config --global user.name "zswu"
    - $ git config --global user.email zswu@cs.nctu.edu.tw
  - Other useful config options
    - core.editor vim
    - merge.tool vimdiff
    - user.signkey

Init

- Initialize an empty repository (or reinitialize)
  - Create a .git directory to store metadata

```
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(09:55 PM)$ git init
Initialized empty Git repository in /net/cs/105/0516074/cscc/nasa/git/.git/
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(09:55 PM)$ ls -al
total 12
drwxr-xr-x 3 zswu cs 4096 Oct 29 21:55 .
drwxr-xr-x 5 zswu cs 4096 Oct 29 21:55 ..
drwxr-xr-x 7 zswu cs 4096 Oct 29 21:55 .git
```

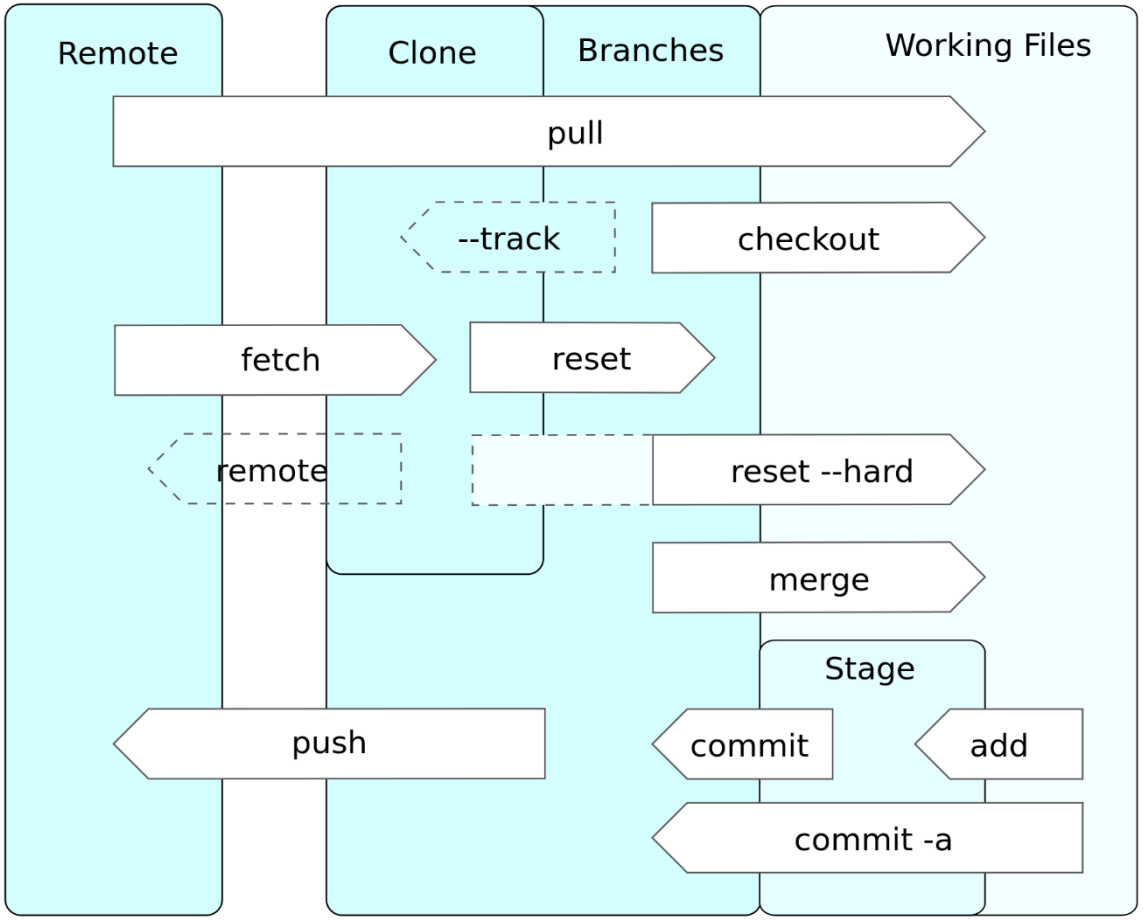
```
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(09:58 PM)$ ls .git
branches  config  description  HEAD  hooks  info  objects  refs
```

Status

- Show the working tree status
  - Untracked files
  - Modified files
  - Deleted files
- See "Short Format" section of `git help status`

```
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(10:00 PM)$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       A
nothing added to commit but untracked files present (use "git add" to track)
```

Operations



https://zh.wikipedia.org/wiki/Git#/media/File:Git_operations.svg

Add

- Add files to the staging area (to be committed)
  - You don't need to commit all the thing at the same time
  - Use a `.gitignore` file to prevent some files to be added

```
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(10:00 PM)$ git add .
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(10:06 PM)$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   A
#
```

Commit

- Save your current state
  - Has a hash ID
  - Parent hash ID
  - What you modified
  - Write some log message to help people know what and why have you done

```
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(10:09 PM)$ git commit -m 'Add file A'
[master (root-commit) 8e5f04c] Add file A
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 A
```

Reset

- Reset a commit or reset temporary area
  - If you regret, you can remove all the changes from temporary area, or remove a commit
  - HEAD^ / HEAD~1, the previous commit of HEAD
  - https://gitbook.tw/chapters/using-git/reset-commit.html

```
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(10:26 PM)$ git reset --soft HEAD^
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(10:27 PM)$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   B
#
```

Log

- List history of the repository
  - Commit message
  - Commit hash ID
  - Commit author

```
commit 347bf8fe3284f90c127abc2032901948908317ab
Author: zswu <zswu@cs.nctu.edu.tw>
Date:   Mon Oct 29 22:45:48 2018 +0800

    Add GG to A and B

commit b4235a938f911b30a9d15110a48431f986d295c5
Author: zswu <zswu@cs.nctu.edu.tw>
Date:   Mon Oct 29 22:30:32 2018 +0800

    Add file B

commit 8e5f04c36b3733249d2cc538738f564253560f5e
Author: zswu <zswu@cs.nctu.edu.tw>
Date:   Mon Oct 29 22:11:08 2018 +0800

    Add file A
(END)
```

Blame

- Detailed history of a file
  - To show the latest commit of each line
  - To find the last one edited this line

```
347bf8fe (zswu 2018-10-29 22:45:48 +0800 1) GG
41ddb36e (zswu 2018-10-29 22:51:29 +0800 2) YY
(END)
```

Branch

- Work on multiple thing at the same time
  - Add feature or debug individually
  - Keep master branch stable

```
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(10:30 PM)$ git branch feature1
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(10:30 PM)$ git branch debug1
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(10:30 PM)$ git branch
  debug1
  feature1
* master
```

Tag

- Tag specific commit
  - Release commit, e.g., v1.0.1, v1.2.3rc, v2.5b3
  - Tag point to only one commit
- Ref: Semantic Versioning
  - https://semver.org/lang/zh-TW/

```
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(10:30 PM)$ git tag Release-V1
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(10:35 PM)$ git tag
Release-V1
```

Checkout

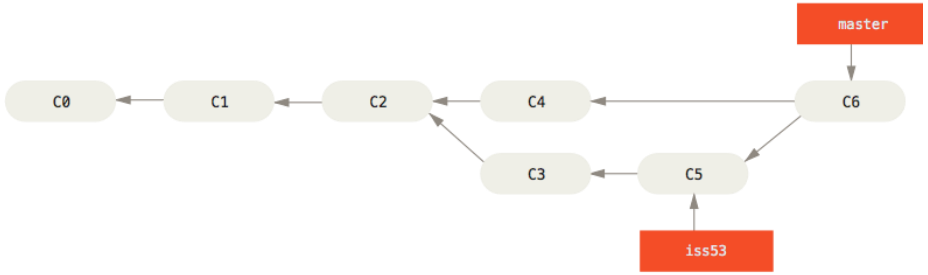
- Checkout something
  - Branch, Tag
  - Commit (by Hash ID)
  - A file / directory (to revert the local modification)

```
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(10:35 PM)$ git checkout debug1
Switched to branch 'debug1'
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(10:39 PM)$ git checkout master
Switched to branch 'master'
```

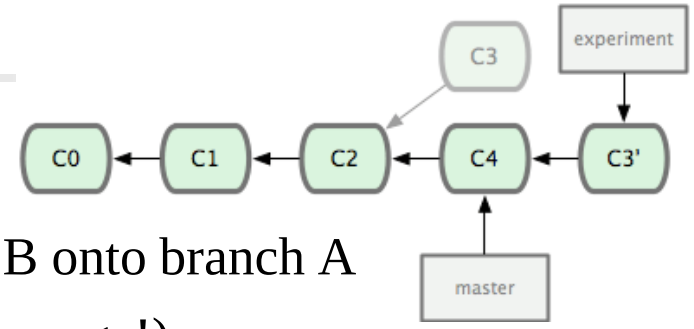
Merge

- Merge branch B to branch A
  - If B is based on A, Fast-forward is applied
    - by default or can be turned off by `--no-ff`
  - If both A and B are changed, a merge commit is created
  - If auto merge failed, you need to resolve the conflict by yourself

```
commit 58986100947aebd7df53f489eeadb5f058f74227
Merge: 30f91f7 7b57ca1
Author: zswu <zswu@cs.nctu.edu.tw>
Date:   Mon Oct 29 22:56:11 2018 +0800

    Merge branch 'debug1'

commit 7b57ca14cbed5f83686c04ca39678e8e4e7df0dd
Author: zswu <zswu@cs.nctu.edu.tw>
Date:   Mon Oct 29 22:55:58 2018 +0800

    Add file debug

commit 30f91f7599ba920a547aa9f76becfa3dd4fe6d80
Author: zswu <zswu@cs.nctu.edu.tw>
Date:   Mon Oct 29 22:55:15 2018 +0800

    Add file C

commit 41ddb36e7cedb79ba566ab5529814003e193308b
Author: zswu <zswu@cs.nctu.edu.tw>
```



Rebase



- Rebase branch B onto branch A
  - Apply each commit (diff) of branch B onto branch A
  - No merge commit (because it is not merge!)
  - If your branch B is forked from branch A, you can use rebase to apply the latest branch A commits
  - History is modified!

```
commit 600e9e340a98ecf8577b1408e6159a7814480807
Author: zswu <zswu@cs.nctu.edu.tw>
Date:   Mon Oct 29 23:07:39 2018 +0800

    Add file feature1

commit b4235a938f911b30a9d15110a48431f986d295c5
Author: zswu <zswu@cs.nctu.edu.tw>
Date:   Mon Oct 29 22:30:32 2018 +0800

    Add file B

commit 8e5f04c36b3733249d2cc538738f564253560f5e
Author: zswu <zswu@cs.nctu.edu.tw>
Date:   Mon Oct 29 22:11:08 2018 +0800

    Add file A
(END)
```

```
commit d87d25b38fa74e770056e91dee06cc626e2c0893
Author: zswu <zswu@cs.nctu.edu.tw>
Date:   Mon Oct 29 23:07:39 2018 +0800

    Add file feature1

commit 58986100947aebd7df53f489eeadb5f058f74227
Merge: 30f91f7 7b57ca1
Author: zswu <zswu@cs.nctu.edu.tw>
Date:   Mon Oct 29 22:56:11 2018 +0800

    Merge branch 'debug1'

commit 7b57ca14cbed5f83686c04ca39678e8e4e7df0dd
Author: zswu <zswu@cs.nctu.edu.tw>
Date:   Mon Oct 29 22:55:58 2018 +0800

    Add file debug

commit 30f91f7599ba920a547aa9f76becfa3dd4fe6d80
```

Using Git with Others

- Remote repository
  - Clone
  - Remote
  - Fetch
  - Push
  - ~~Pull~~ => Fetch + Merge 、Fetch + Rebase
- Conflict
  - Rebase/Merge --about/--skip/--continue
  - Stash / Stash pop
  - Revert

Clone

- Clone a git repository
  - Copy all things including history, branches, tags

```
>_< :130(09:53 PM)$ git clone --bare --depth=10  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
Cloning into bare repository 'linux.git'...
remote: Counting objects: 99201, done.
remote: Compressing objects: 100% (77824/77824), done.
remote: Total 99201 (delta 35189), reused 53778 (delta 20180)
Receiving objects: 100% (99201/99201), 187.75 MiB | 52.00 KiB/s, done.
Resolving deltas: 100% (35189/35189), done.
zswu@ linux3 (/bin/bash): ~/alpha/kernel
^_^(10:55 PM)$
```

Remote

- Manage remote (tracked) repositories
  - You can add many remote repositories
  - Usually, origin is your default remote repository

```
zswu@ linux3 (/bin/bash): ~/alpha/kernel/linux.git
^_^(11:12 PM)$ git remote -v
origin  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git (fetch)
origin  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git (push)
```

Fetch

- Fetch new commits from remote
  - Remote branch will be placed at remote-name/branch-name
    - origin/master
    - origin/debug1
  - Usually, we will do this after fetch
    - $ git rebase origin/master

```
zswu@ linux3 (/bin/bash): ~/alpha/kernel/linux.git
^_^(11:12 PM)$ git fetch origin
From https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux
 * branch            HEAD       -> FETCH_HEAD
```

Push

- Push things to remote branch
  - You must have write permission on the remote server
  - Push a branch to remote repository
    - If exists, update it
    - Can also be used to delete a remote repository (push an empty branch)

```
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git.remote
^_^(11:23 PM)$ git init --bare
Initialized empty Git repository in /net/cs/105/0516074/cscc/nasa/git.remote/
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git.remote
^_^(11:23 PM)$ cd ..
zswu@ linux3 (/bin/bash): ~/cscc/nasa
^_^(11:23 PM)$ cd git
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(11:23 PM)$ git remote add origin ../git.remote
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(11:23 PM)$ git push origin master
Counting objects: 17, done.
Delta compression using up to 6 threads.
Compressing objects: 100% (13/13), done.
Writing objects: 100% (17/17), 1.33 KiB | 0 bytes/s, done.
Total 17 (delta 4), reused 0 (delta 0)
To ../git.remote
 * [new branch]      master -> master
```

Pull

- Pull is equal to fetch + merge, or fetch + rebase
  - Defaults to fetch + merge
  - Pull something adds a lots of merge message into your project
    - Use fetch + rebase instead
    - Still depends on your (team's) workflow

```
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(11:23 PM)$ git pull origin master
From ../git.remote
 * branch            master     -> FETCH_HEAD
Already up-to-date.
```

Conflict

- Why
  - Modify the same file
    - Auto merge failed
  - Squash commits
    - Squash will change commits history
  - $ git push -f
    - Try to not do this (to master branch)

```
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(11:33 PM)$ git merge conflict1
Auto-merging A
CONFLICT (content): Merge conflict in A
Automatic merge failed; fix conflicts and then commit the result.
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
>_< :1(11:33 PM)$
```

Merge/Rebase Conflict

```
1 GG
2 YY
3 <<<<<<< HEAD
4 AA
5 =======
6 BB
7 >>>>>>> conflict1
```

- ❑ Continue
  - After you fixed the conflict, continue the action (Merge/Rebase)
- ❑ Skip
  - Skip this commit, it won't be merge into target branch
- ❑ Abort
  - Abort, nothing will change ☺

```
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(11:37 PM)$ git status
# On branch master
# You have unmerged paths.
#   (fix conflicts and run "git commit")
#
# Unmerged paths:
#   (use "git add <file>..." to mark resolution)
#
#       both modified:      A
#
no changes added to commit (use "git add" and/or "
```

```
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
>_< :1(11:39 PM)$ git rebase master
First, rewinding head to replay your work on top of it...
Applying: Add BB to A
Using index info to reconstruct a base tree...
M       A
Falling back to patching base and 3-way merge...
Auto-merging A
CONFLICT (content): Merge conflict in A
Failed to merge in the changes.
Patch failed at 0001 Add BB to A
The copy of the patch that failed is found in:
   /net/cs/105/0516074/cscc/nasa/git/.git/rebase-apply/patch

When you have resolved this problem, run "git rebase --continue".
If you prefer to skip this patch, run "git rebase --skip" instead.
To check out the original branch and stop rebasing, run "git rebase --abort".
```

Stash / Stash Pop

- Stash things not yet commit
  - Like a stack, first in last out
  - Convenient when rebasing/merging

```
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(11:43 PM)$ git rebase conflict1
Cannot rebase: You have unstaged changes.
Please commit or stash them.
```

```
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(11:43 PM)$ echo YY > B
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(11:43 PM)$ cat B
YY
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(11:43 PM)$ git stash
Saved working directory and index state WIP
HEAD is now at 2850c32 Add AA to A
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(11:43 PM)$ cat B
GG
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(11:43 PM)$ git stash pop
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what
#   (use "git checkout -- <file>..." to dis
#
#       modified:   B
#
no changes added to commit (use "git add" a
Dropped refs/stash@{0} (64e9da160b0372f8753
zswu@ linux3 (/bin/bash): ~/cscc/nasa/git
^_^(11:43 PM)$ cat B
YY
```

Revert

- Revert a commit
  - Revert a commit by adding a new commit
    - Won't break the history
  - The new commit is applying the negative version of the old commit

```
commit ea9e911d30bafd89ee5bd2f921656d58891f4b9a
Author: zswu <zswu@cs.nctu.edu.tw>
Date:   Mon Oct 29 23:50:55 2018 +0800

    Revert "Add AA to A"

    This reverts commit 2850c329fd4084e865c6fc7cab5a20c8e3c37f20.

commit 2850c329fd4084e865c6fc7cab5a20c8e3c37f20
Author: zswu <zswu@cs.nctu.edu.tw>
Date:   Mon Oct 29 23:31:08 2018 +0800

    Add AA to A

commit 58986100947aebd7df53f489eeadb5f058f74227
```

Etiquette Rules of Using Git (1/5)

- Commit Message
  - What you done
  - Why
  - How
  - Format is important
  - `` `git --amend` / `git rebase -i HEAD~x` ``
- Don't modify master directly (after pushing)
  - Master branch may become unstable
  - Others need to solve the conflict that cause by your temporary code
  - To keep the history untouched

Etiquette Rules of Using Git (2/5)

- Keep history clean
  - Don't change the history if you share the branch with others
    - Unnecessary conflicts
  - Try not to use `git push -f`
    - Unnecessary conflicts
  - Try not to use merge except you are on the master branch
    - Use `rebase` instead
    - Branch may become incapable when being merged
    - unnecessary merge commit(s)
  - Try not to do `git pull` (fetch + merge)
    - Might add an unnecessary merge commit
  - Use `rebase -i`
    - Shape your commits

Etiquette Rules of Using Git (3/5)

- Good commit message
  - Help others (including future you!)
  - Again, “what and why”
- Ref:
  - https://chris.beams.io/posts/git-commit/
    - https://blog.louie.lu/2017/03/21/如何寫一個-git-commit-message/
  - http://blog.fourdesire.com/2018/07/03/撰寫有效的-git-commit-message

Etiquette Rules of Using Git (4/5)

- Bad commit message
- Ref:
  - http://www.commitlogsfromlastnight.com/
  - http://whatthecommit.com/

https://xkcd.com/1296/

Etiquette Rules of Using Git (5/5)

- Use rebase -i to create clean history
  - $ git rebase -i HEAD~10

```
 1 pick 2850c32 Add AA to A
 2 pick ea9e911 Revert "Add AA to A"
 3 pick a56ac34 Add T
 4 pick 2895ff7 fix bug
 5 pick 95c09b5 fix bug
 6 pick 045b0cf fix bug
 7 pick bbced05 fix bug
 8 pick b298fa8 fix bug
 9 pick adf2b3b fix bug
10 pick 424a124 fix bug
11
12 # Rebase 5898610..424a124 onto 5898610
13 #
14 # Commands:
15 #  p, pick = use commit
16 #  r, reword = use commit, but edit the commit message
17 #  e, edit = use commit, but stop for amending
18 #  s, squash = use commit, but meld into previous commit
19 #  f, fixup = like "squash", but discard this commit's log message
20 #  x, exec = run command (the rest of the line) using shell
21 #
22 # These lines can be re-ordered; they are executed from top to bottom.
23 #
24 # If you remove a line here THAT COMMIT WILL BE LOST.
25 #
26 # However, if you remove everything, the rebase will be aborted.
27 #
28 # Note that empty commits are commented out
~
```

Tools & Services (1/2)

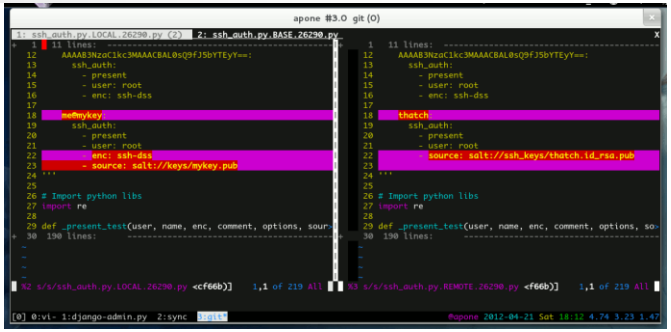
- Online Git Service
  - Web interface
  - Remote repository
  - Issue, Pull Requests (or Merge Requests… etc.)
  - Code Review
  - Community
  - CI/CD
  - Example
    - GitHub
    - GitLab
    - Bitbucket

Tools & Services (2/2)

❑ Tools

- Gerrit
  - Web-based code review tool
- Tig
  - ncurses-based text-mode interface for Git
  - https://www.jianshu.com/p/e4ca3030a9d5
- Git GUI
  - GUI-based interface for Git
- KDiff3 / vimdiff / meld
  - Help users to solve conflict
- etc.

Tips (1/3)



- Conflict Solving
  - Use merge tools
    - $ git config --global merge.tool kdiff3
  - Show common ancestor
    - $ git config --global merge.conflictstyle diff3
  - Change merge algorithm
    - $ git merge --strategy-option=patience

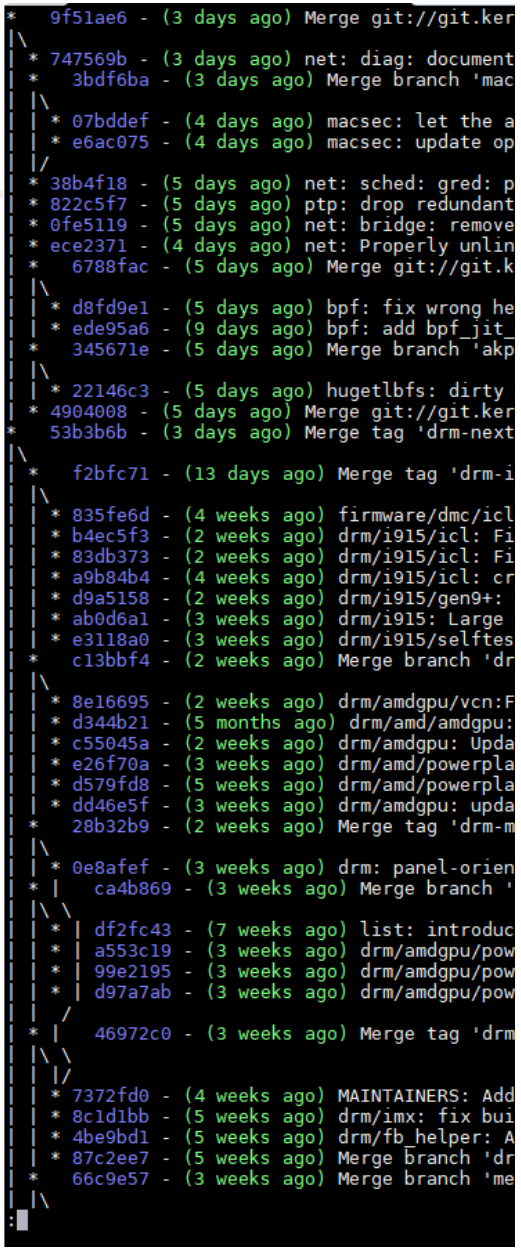
```
cauliflower
<<<<<<< HEAD
peas
potatoes
||||||| merged common ancestors
peas
=======
>>>>>>> topic
tomatoes
```

  - More...
    - https://developer.atlassian.com/blog/2015/12/tips-tools-to-solve-git-conflicts/

Tips (2/3)

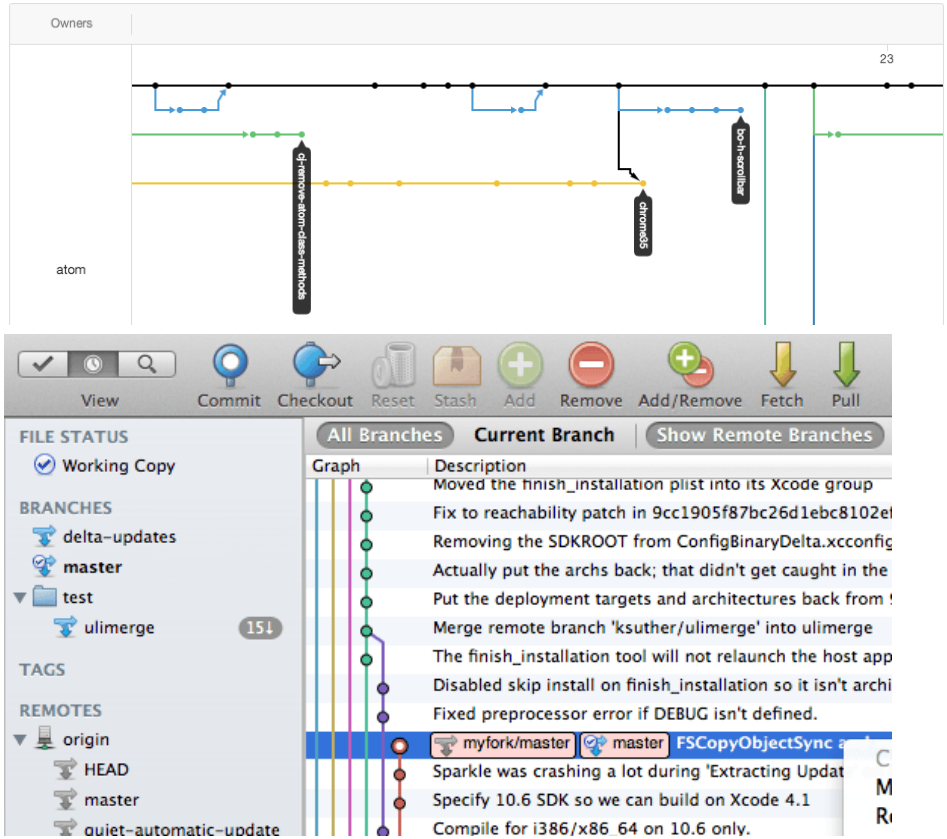
- History visualize
  - Use git log
    - $ git log --decorate --graph [--oneline]
    - $ git log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset) %C(white)%s%C(reset) %C(dim white)-%an%C(reset)%C(bold yellow)%d%C(reset)' --all
    - $ man git-log
  - ~/.gitconfig:

```
[alias]
  gl = log --graph --pretty=oneline --
```



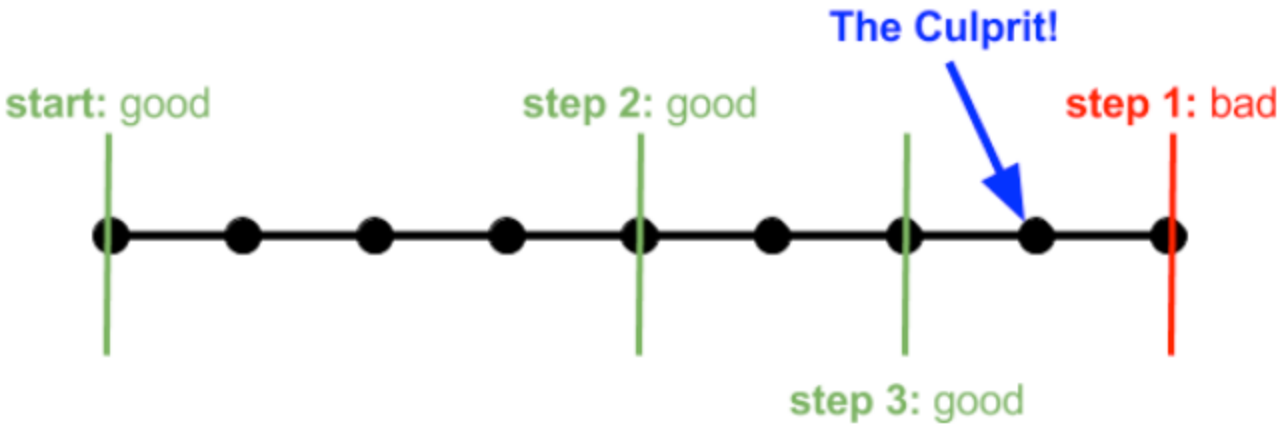
Tips (3/3)

- History visualize (cont.)
  - GitHub / Online tools
  - Git GUI



bisect

- Use binary search to find the commit that introduced a bug
  - git bisect start
  - git bisect good/bad



- Ref:
  - https://git-scm.com/book/en/v2/Git-Tools-Debugging-with-Git

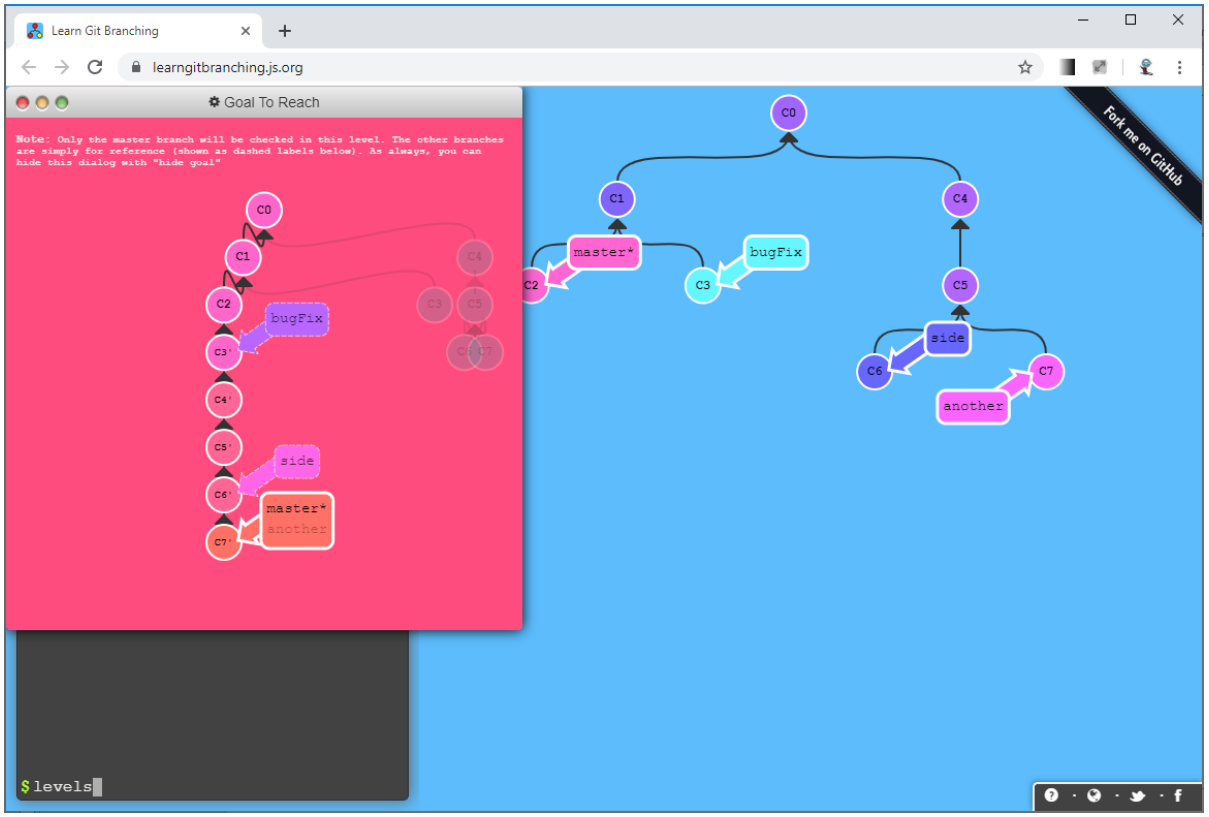
https://www.sumologic.com/blog/who-broke-my-test-a-git-bisect-tutorial/

Useful links

- Reference
  - 為你自己學 Git
    - https://gitbook.tw
  - Pro Git
  - https://git-scm.com/book/en/v2
  - A Visual Git Reference
  - https://marklodato.github.io/visual-git-guide/index-en.html
- Utilities
  - .gitignore generator
  - https://www.gitignore.io

Learn Git Branching

- https://learngitbranching.js.org/



Git Workflow(s)

- GitHub flow
  - https://guides.github.com/introduction/flow/
- Git workflow
  - https://nvie.com/posts/a-successful-git-branching-model/
  - https://gitbook.tw/chapters/gitflow/why-need-git-flow.html
- Which one?
  - https://blog.wu-boy.com/2017/12/github-flow-vs-git-flow/