Utilities – tmux & git

lctseng (2019-2020, CC BY-SA)
wnlee and others (1996-2018)

交大資工系資訊中心

Tmux

Terminal Multiplexer

交大資工系資訊中心

What is tmux?

- Terminal Multiplexer
  - Allow open multiple tabs (multiple terminals)

What is tmux?

SSH

Tmux Client

Tmux Server

What is tmux?

Why should I use tmux?

- Keep your working session alive
- By default, shell is terminated when connection is lost
  - Including any programs/editors opened
  - Any unsaved changes are discarded without warning
- tmux will not be terminated when connection is lost
  - Attach to previous sessions!
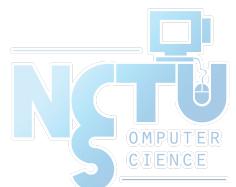
Example screenshot of tmux

https://github.com/lctseng/Unix-User-Config/blob/master/.tmux.conf

Advantages of tmux

- Multiple sessions, windows, panes
- Keep the sessions, attach/detach anytime
- Powerful window division (panes)
- Share screen by attaching to the same session

Start tmux

- tmux
- tmux attach [-t <number>]
- tmux detach
- tmux ls
- tmux kill-session [<number>]

tmux 101

交大資工系資訊中心

Computer Center of Department of Computer Science, NCTU

Session

- Create new session (open a new browser)
  - Execute outside of any tmux sessions
  - $ tmux
- Detach current session
  - When attached in a session
  - $ tmux detach
  - Or close the terminal directly
- Attach to previous session
  - $ tmux attach
  - Attach only if previous sessions exist

Multiple sessions (1)

● Open multiple browsers

```
(0) + 0: 4 windows
(1) + 5: 1 windows
(2) + 6: 1 windows (attached)




[6] 0:[tmux]*                                    "linux4.cs.nctu.edu.tw" 10:12 09-Sep-20
```

Multiple sessions (2)

- Open multiple browsers
- List opened sessions
  - $ tmux list-sessions

```
0: 1 windows (created Sun Jun 16 18:49:57 2019) [128x38]
1: 3 windows (created Sun Jun 16 18:50:03 2019) [128x38]
```

- Attach to previous session by id
  - $ tmux attach -t session-id

tmux - bindkey

- Operations start with a special key combination
- Default is C-b
  - Where C is Ctrl (control)
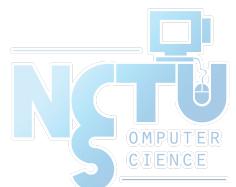
tmux - command

- Bindkey + :
- Open command prompt and execute tmux commands
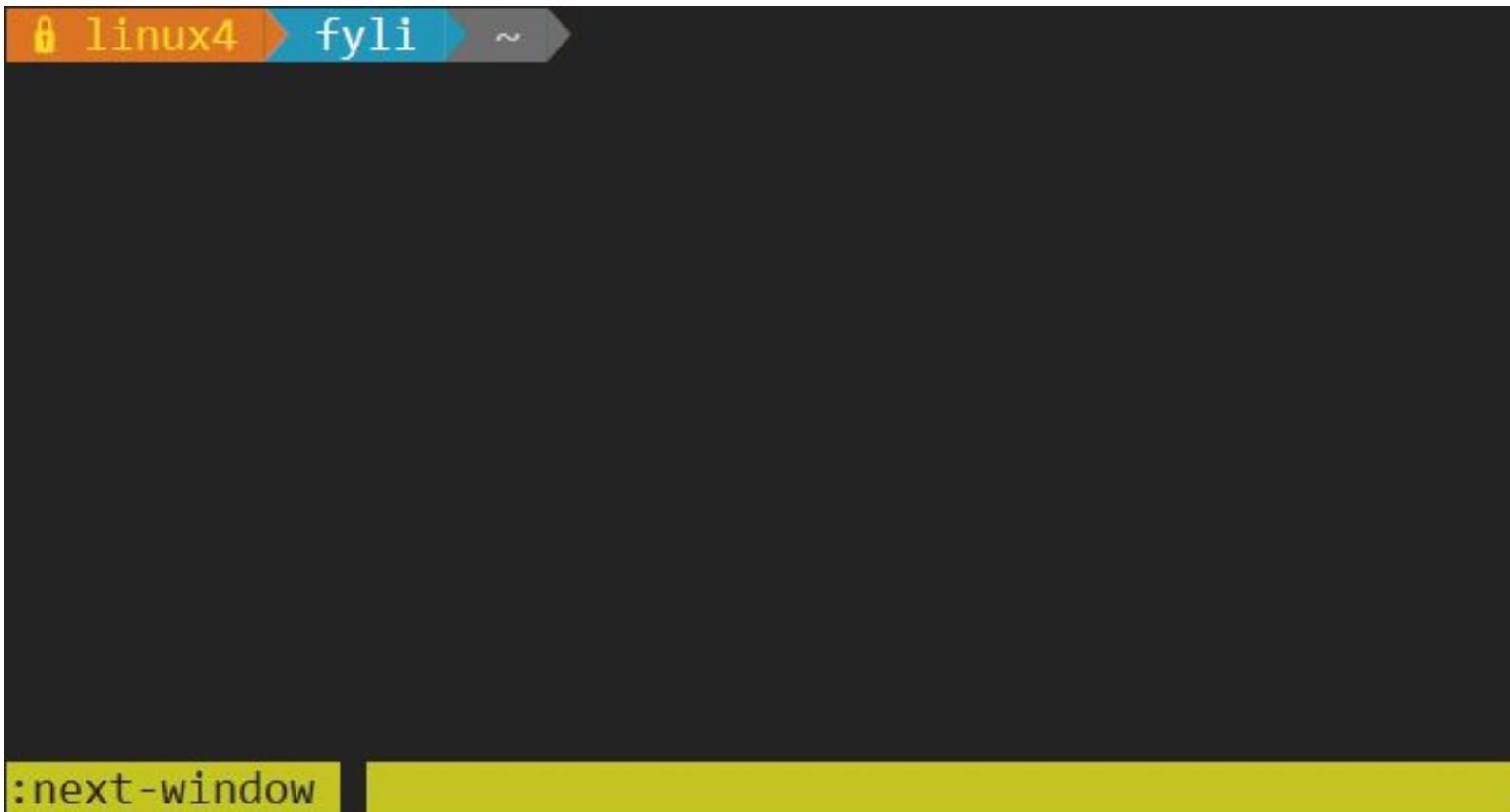
Bindkey - Window

bindkey (default is C-b) //C == control	C	new-window
	N	next-window
	P	previous-window
	L	last-window
	\<NUM\>	select-window -t := \<NUM\>
	&	confirm-before -p "kill-window #W? (y/n)" kill-window
	,	command-prompt -I "#W" "rename-window '%%'"
	.	command-prompt "move-window -t '%%'"

Bindkey - Pane

bindkey (default is C-b) //C == control	%	split-window -h
	"	split-window
	arrow-key	select-pane
	alt + arrow-kwy	resize-pane
	x	confirm-before -p "kill-pane #P? (y/n)" kill-pane
	{	swap-pane -U
	}	swap-pane -D

tmux - bindkey

- bindkey + ?

Configuration - tmux.conf

- ~/.tmux.conf
- design yourself style
- colorful

Configuration - tmux.conf

- ~/.tmux.conf

```
 1 set -g status-utf8 on
 2 setw -g utf8 on
 3 # GENERAL SETTING
 4 bind-key r source-file ~/.tmux.conf; display-message "~/.tmux.conf is reloaded"
 5 set-window-option -g automatic-rename off
 6 set-option -g default-terminal "xterm"
 7 set-option -g prefix C-a
 8
 9 # STATUSBAR STYLE
10 # main
11 set-option -g status-bg colour236
12 set-option -g status-fg colour166
13 setw -g window-status-current-format "#I:#W#F"
14 setw -g window-status-current-fg colour215
15
16 #left
17 set-option -g status-left ''
18 set-option -g status-left-length 0
19
20 #right
21 set-option -g status-right "#h [%Y-%m-%d %H:%M]"
22 # BIND KEY
23 bind -n F8 previous-window
24 bind -n F9 next-window
25 bind -n F10 last-window
26 bind -n M-Right next-window
27 bind -n M-Left previous-window
~
~
~
"~/.tmux.conf" 27L, 698C
0:tbsd21  1:csduty* 2:cshome  3:csschedule- 4:bsd3  5:jail  6:nasa  7:jsh
```

Configuration - bindkey

- ~/.tmux.conf
  - bind-key (alias: bind)
  - C (alias: <Ctrl>)
  - M (alias: <Alt>)
- bind-key <key> <command>
  - -T key-table (default table is prefix)
  - -n : alias for -T root   => Don't need to press C-b first
  - -r : repeat

```
22 # BIND KEY
23 bind -n F8 previous-window
24 bind -n F9 next-window
25 bind -n F10 last-window
26 bind -n M-Right next-window
27 bind -n M-Left previous-window
```

Configuration - set

- ~/.tmux.conf
  - set-window-option (alias: setw)

```
 1 set -g status-utf8 on
 2 setw -g utf8 on
 3 # GENERAL SETTING
 4 bind-key r source-file ~/.tmux.conf; display-message "~/.tmux.conf is reloaded"
 5 set-window-option -g automatic-rename off
 6 set-option -g default-terminal "xterm"
 7 set-option -g prefix C-a
 8
 9 # STATUSBAR STYLE
10 # main
11 set-option -g status-bg colour236
12 set-option -g status-fg colour166
13 setw -g window-status-current-format "#I:#W#F"
14 setw -g window-status-current-fg colour215
15
16 #left
17 set-option -g status-left ''
18 set-option -g status-left-length 0
19
20 #right
21 set-option -g status-right "#h [%Y-%m-%d %H:%M]"
```

Configuration - set

tmux - share session

- Both side can edit and execute commands
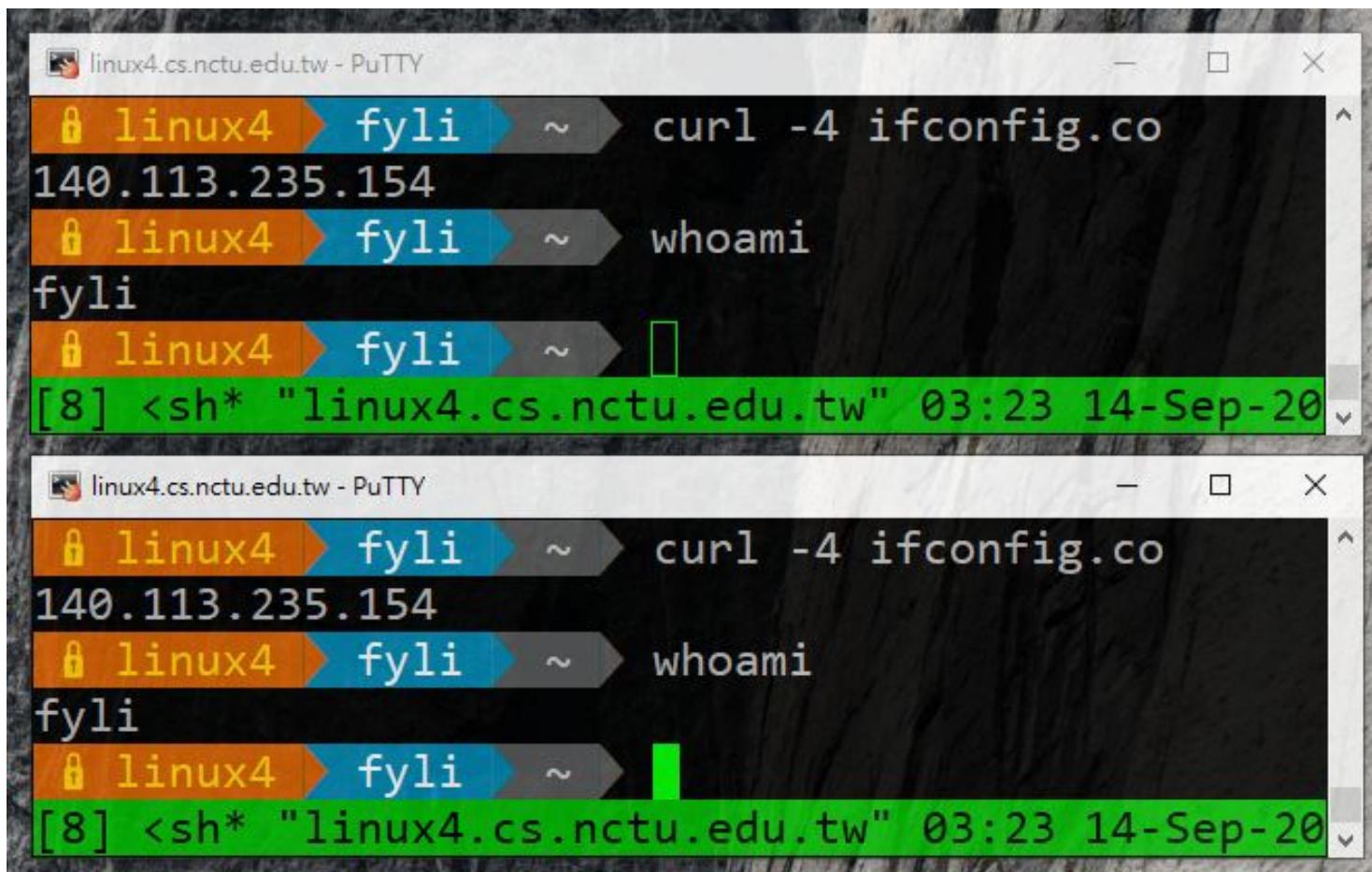
Reference

- [tmux (1)](#)
- tmux shortcuts & cheatsheet
  - https://gist.github.com/MohamedAlaa/2961058
- tmux brief introduction (chinese)
  - https://5xruby.tw/posts/tmux/

Appendix: tmux v.s. screen

		tmux	screen
Window Split		top-down & left-right	top-down only (default)
		different sessions can have different schemes	scheme must shared by all sessions
Session		switch between sessions without detach	detach first then re-attach another session
		multiple clients can attach to same session	one session for one client only (force detach)
Profile		.tmux	.screenrc
		highly customizable	less than tmux

Git

One of the most popular Version Control Systems

交大資工系資訊中心

Version Control Systems (VCS)

- Also known as Source Code Management (SCM)
- Records changes to a set of files over time so that you can recall specific versions later.
- Easy for developing, finding bug, blame someone else, ...
- Popular tools
  - Git, Subversion (svn), Mercurial (hg)

Version Control Systems (VCS)

- Web hosting
  - Backup projects
  - Collaborating
  - Commercial providers
    - Github, Gitlab, Bitbucket
  - Self-hosted
    - git.cs.nctu.edu.tw

In case of fire 🔥
- 1. git commit
- 2. git push
- 3. leave building
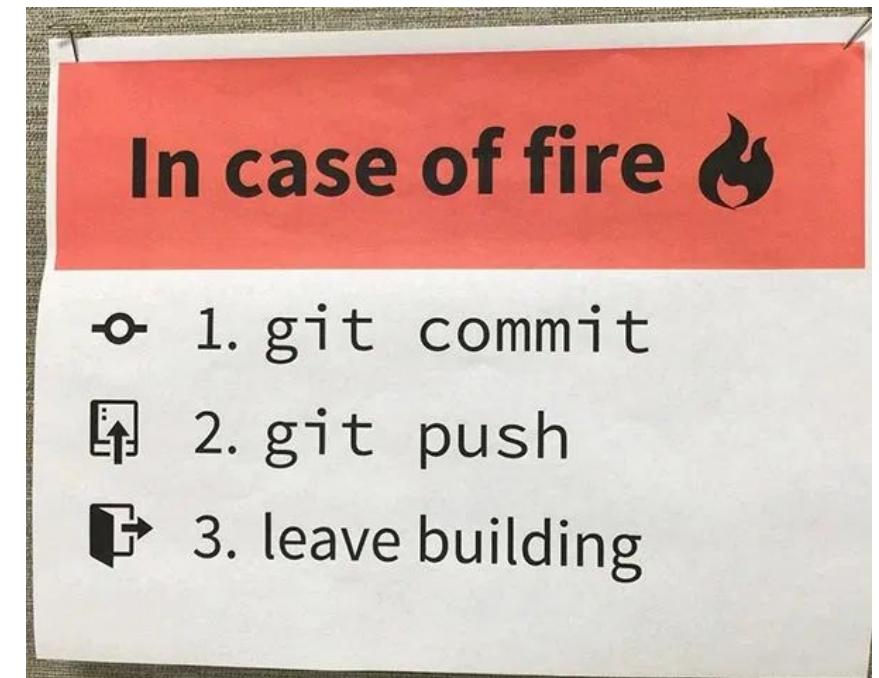
Without VCS

- Copy-paste manually

```
# 2015-11-10
cp -r project project.bak
# 2015-11-11
cp -r project project.bak1
# 2015-11-12
cp -r project project.bak2
# 2015-11-15
cp -r project project.bak3
```


project

project - 複製

project - 複製 - 複製

project - 複製 - 複製 (2)

project - 複製 - 複製 (3)
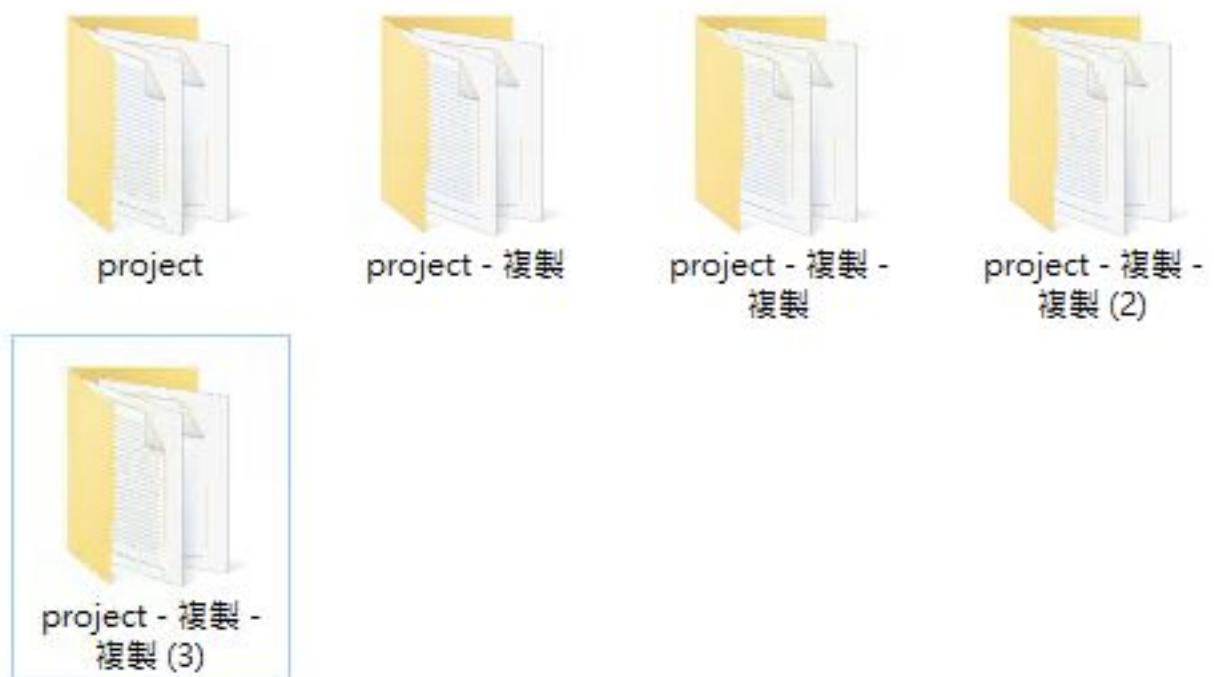
How VCS works

- In addition to your files, VCS stores extra information under your project folder
  - Hidden folders
    - .git for Git
    - .hg for Mercurial
  - Previous versions of files (compressed)
  - Remote repository information

Types of VCS (1)

- Local VCS
  - All versions are in local
  - Cannot share with others
  - No remote backup
- Example
  - Manually copy-paste
  - Git/Hg without setting remote upstream

Local Computer

Checkout

Version Database

File

Version 3
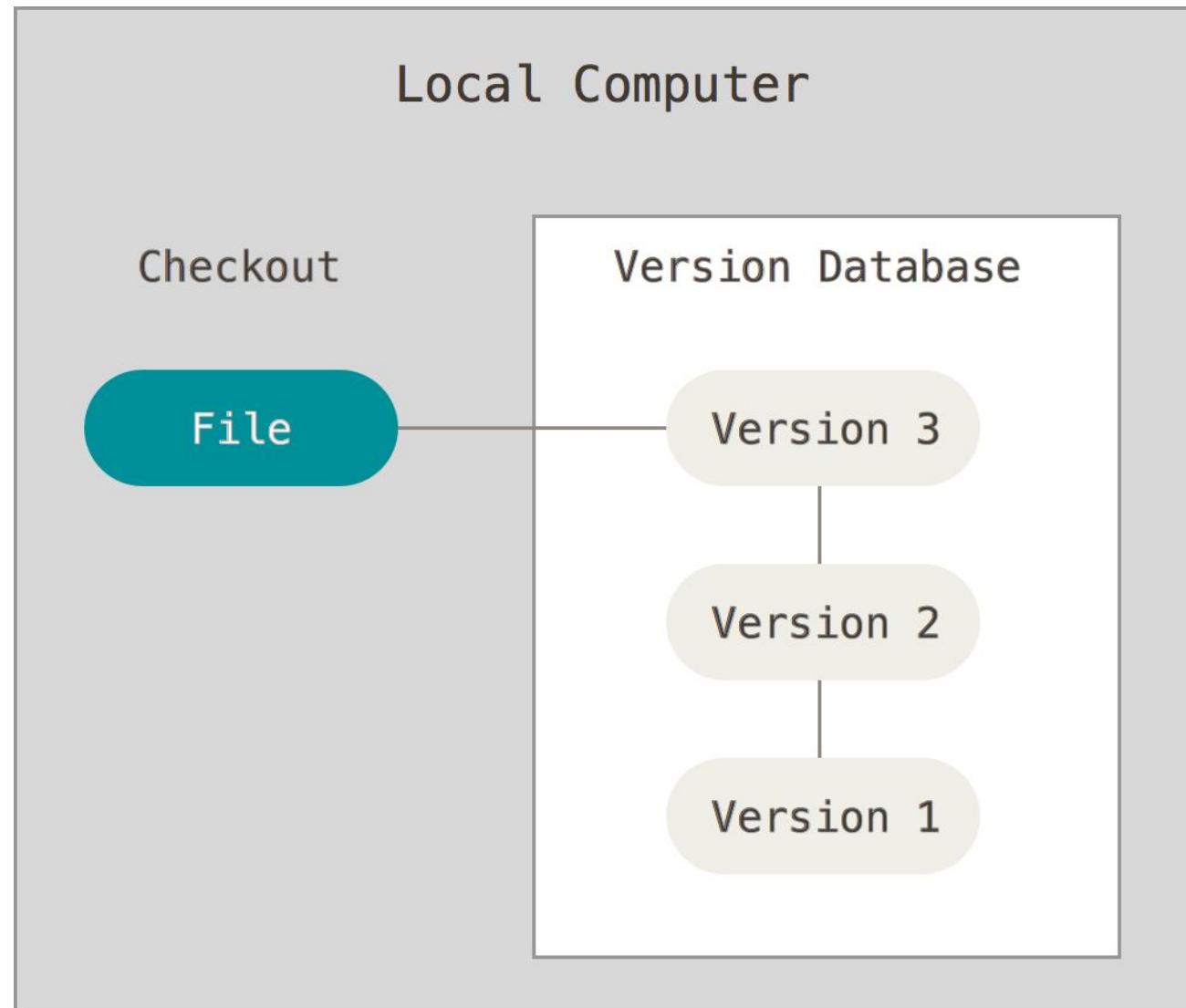
Version 2

Version 1

Types of VCS (2)

- Centralized VCS
  - User can checkout one specific version
  - Remote server has all versions
  - Lost access to other versions if network is down
  - Lost all versions if server is down
- Example
  - Subversion
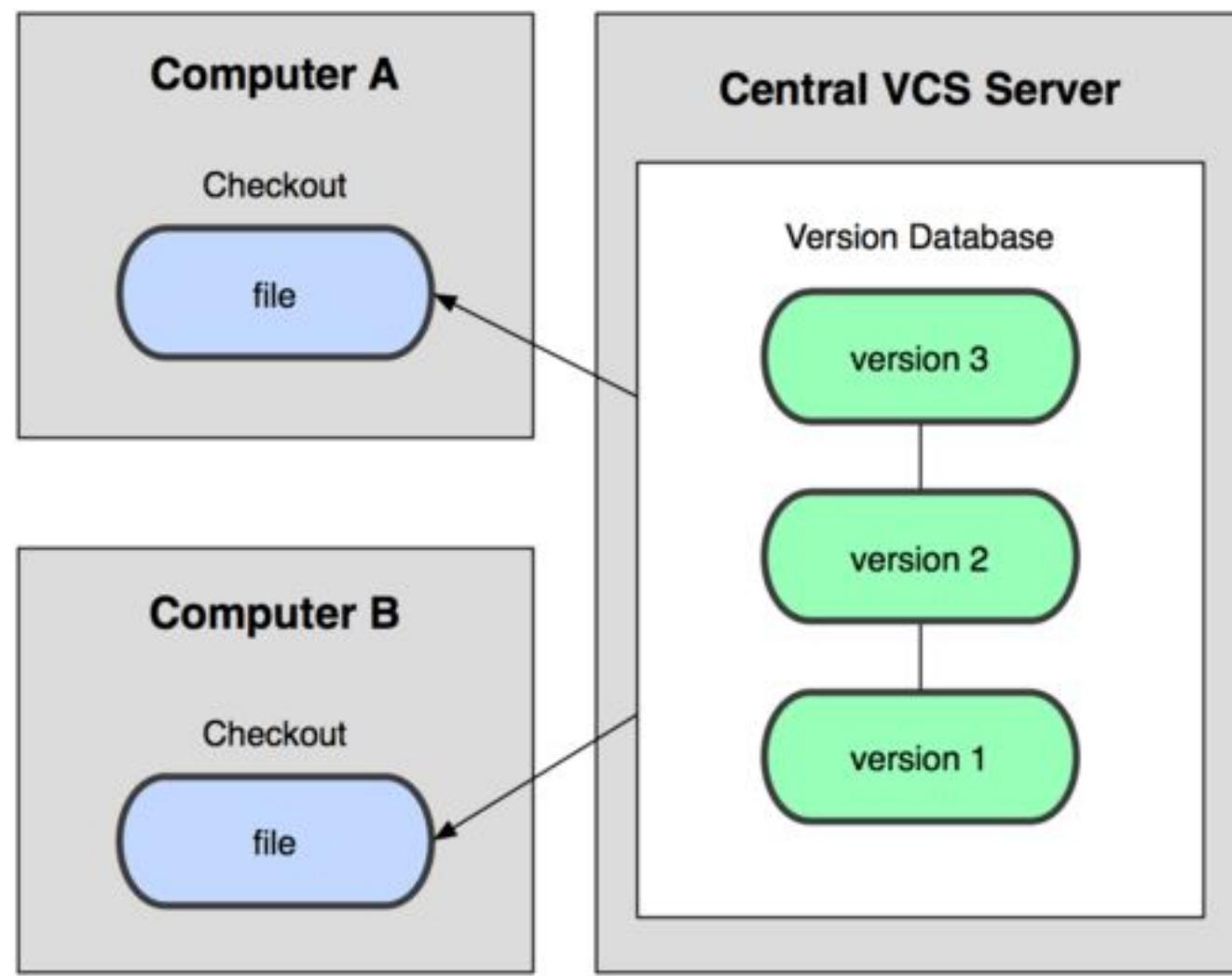
Types of VCS (3)

- Distributed VCS
  - Every node has complete copy of versions
  - Offline working
  - Synchronization
    - Usually using a server as the source of truth
- Example
  - Git, Mercurial
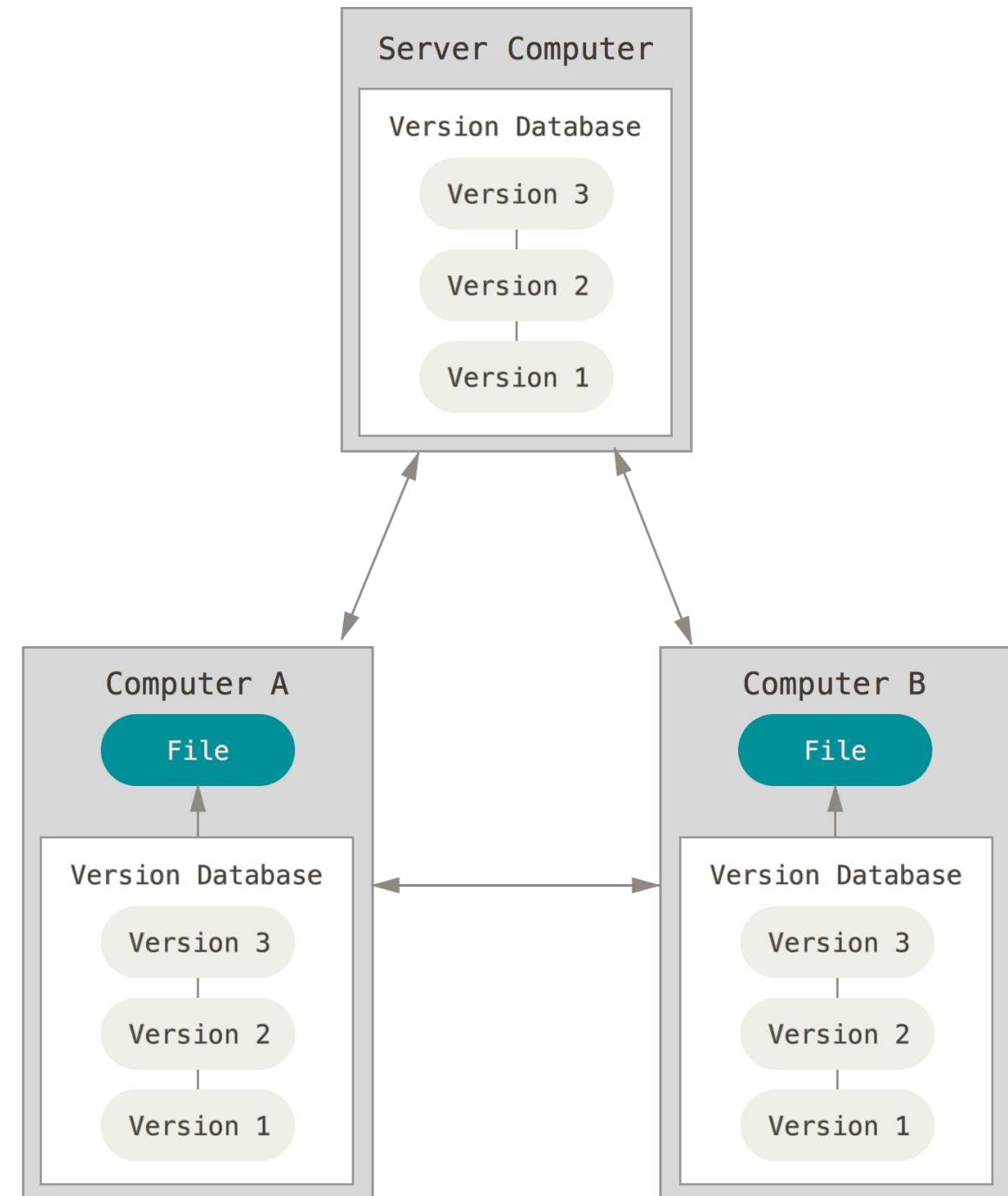
Git

- Distributed VCS
- History
  - Linus Torvalds (the creator of Linux)
  - Source Control Management for Linux Kernel
  - ~2005
    - Using BitKeeper (commercial software)
  - 2005~
    - Developed Git

Git

- Snapshots, not differences
  - All versions of all files are stored independently
  - Easy for checking out to any version
- Nearly every operation is local
- Git has integrity
  - SHA checking
- Git generally only adds data
  - Delete
    - Store the file in .git folder and hide from your workspace
  - Modify
    - Backup the original file to .git folder

Git

- Git is very powerful, with many features
- In this class, we only talk about the very simple one
- We will cover
  - How to install and create repos
  - How to add files/make changes
  - How to create commits   Homework 2
  - How to navigate between versions
  - How to push to remote
- Will NOT cover
  - branching (git branch, git merge, ...)
  - Anything else
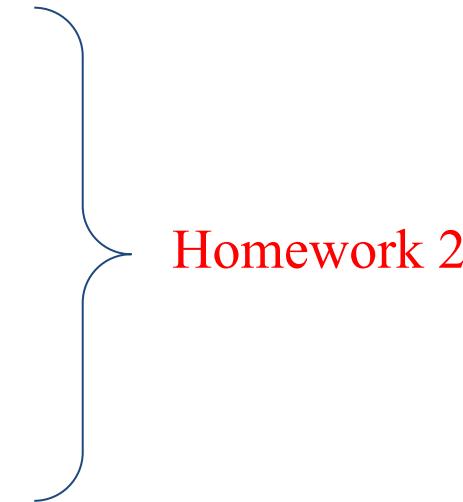
Git - installation

- FreeBSD
  - pkg install git
- Other OS
  - https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

```
> git
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an existing one
```

Git - getting started

- Developer information
  - In terminal
    - `$ git config --global user.email "you@example.com"`
    - `$ git config --global user.name "Name"`
  - Or edit manually in ~/.gitconfig
- Create a new repository
  - `$ mkdir hw2`
  - `$ cd hw2`
  - `$ git init`
    - Initialize an empty project with .git directory

Git - the three stages
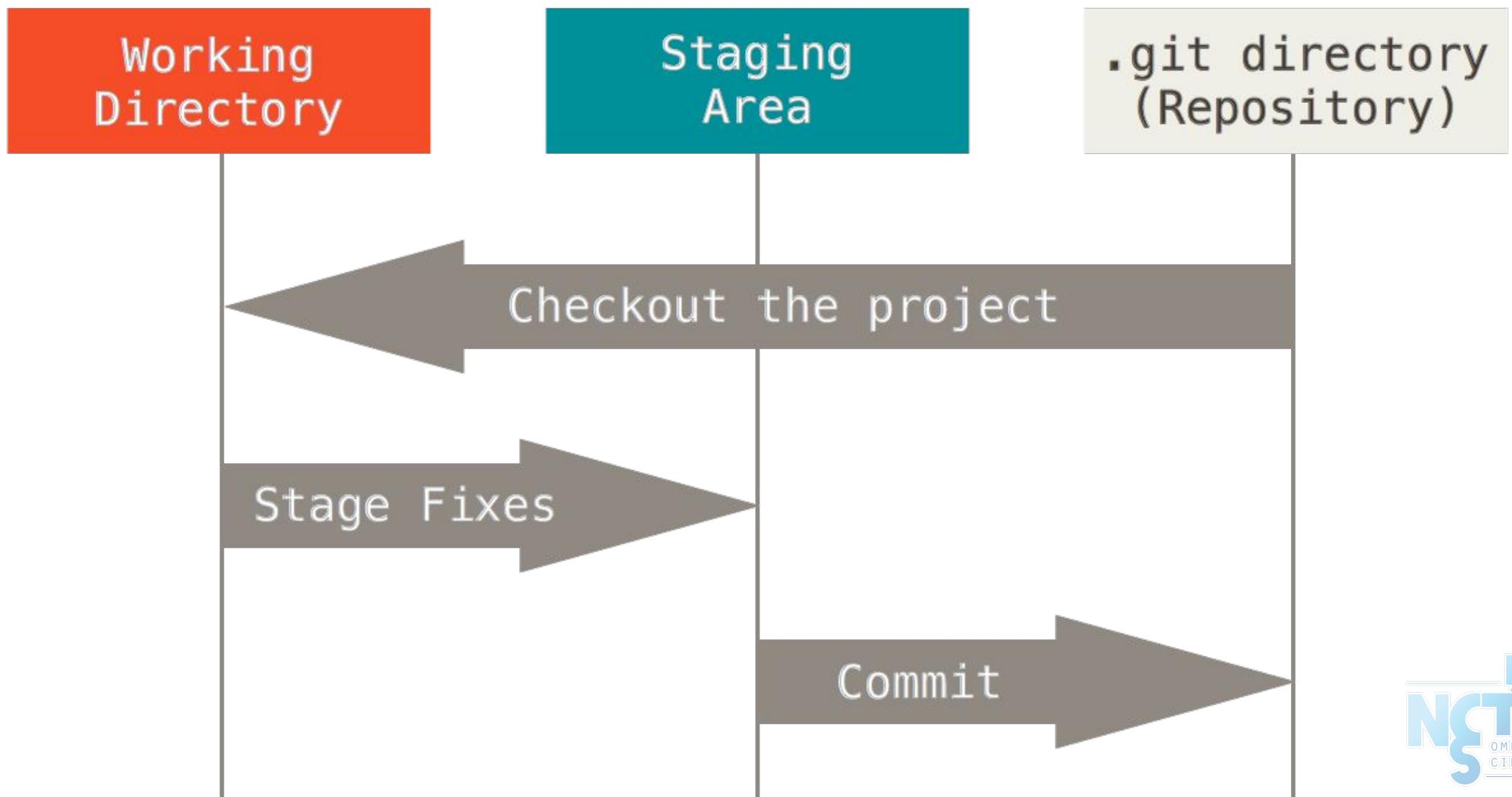
Git - file lifecycle

- The lifecycle of the status of your files
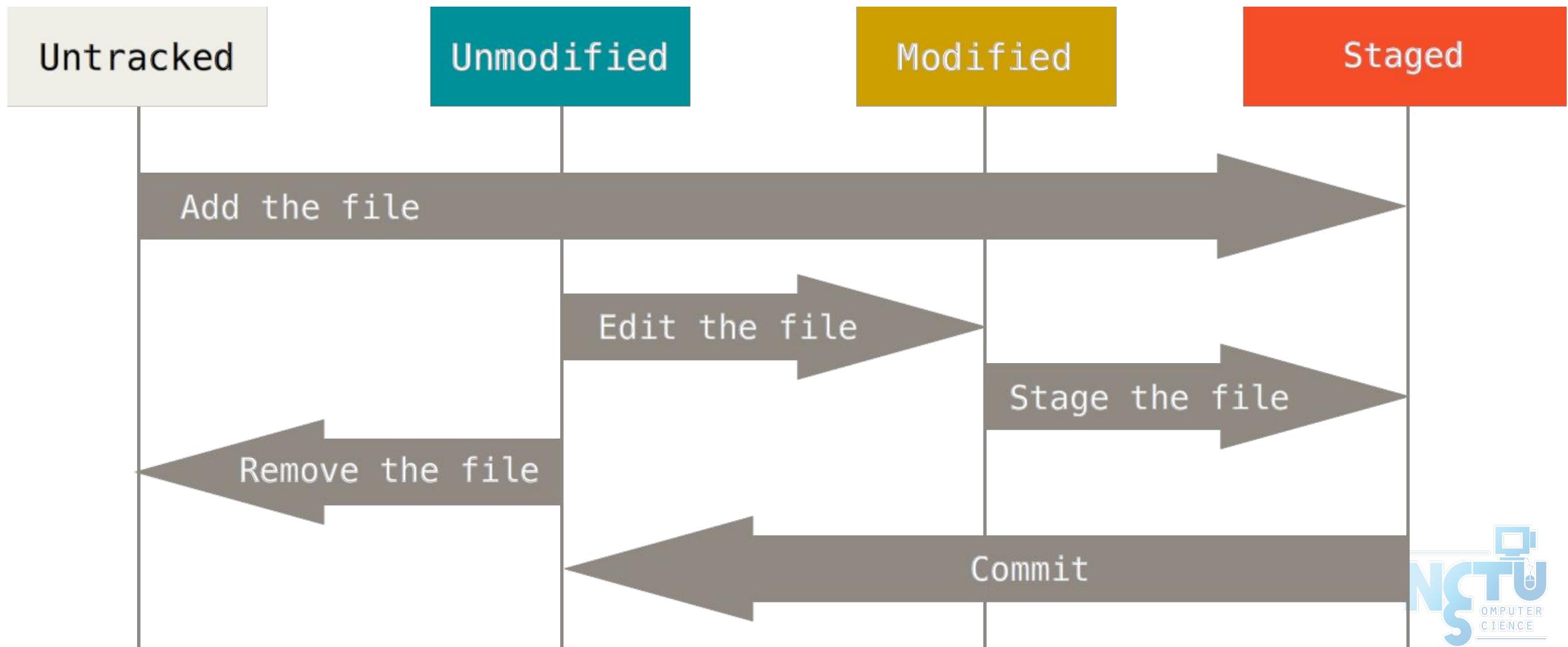
Git - basic operations (1)

- git add
  - Add file contents to the index (staged)
  - `$ git add file1 file2...`
  - `$ git add directory`
  - `$ git add .`

Git - basic operations (2)

- git commit
  - Confirm your staged change and create a commit (revision)
  - Will open default text editor for commit message
  - Useful options
    - -a: stage all modified and deleted path (git add)
    - -m MSG: use MSG as commit message without opening editors

Git - basic operations (3)

- git status
  - View the working directory status of current project

```
$ git status -s
M README.md           # updated in index
D run.sh              # deleted from index
R src/main.js         # renamed in index
A src/index.html      # added to index
?? src/README.md      # untracked
```
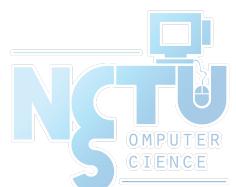
Git - basic operations (4)

- git diff
  - Compare change with the previous version
    - https://git-scm.com/docs/git-diff

```
$ git diff
diff --git a/README.md b/README.md
index 76f177f..f4986c2 100644
--- a/README.md
+++ b/README.md
@@ -1 +1 @@
-# Hi
+# Hello
```

Git - basic operations (5)

- git log
  - View the commit log (history)
  - Change the log format
    - http://gits-scm.com/docs/pretty-formats

```
$ git log

commit 2f2bd00051fbd4d4978b7e96508b97950d6e60b1
Author: lctseng lctseng@cs.nctu.edu.tw
Date:    Thu Oct 8 17:10:13 2020 +0800


    Initial commit
```

Git - basic operations (6)

- git checkout
  - Go to other revisions or branches
  - `$ git checkout <commit-id>`
  - Go back to latest change:
    - `$ git checkout master`

```
$ git status
On branch master

$ git checkout 2f2bd00
Note: checking out '2f2bd00'.

$ git status
HEAD detached at 2f2bd00
```
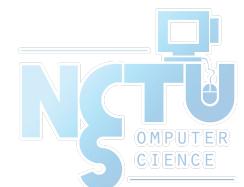
Git - basic operations (7)

- git bisect
  - Finding problematic commit
  - Flow (similar to binary search)
    - `$ git bisect start`
    - Define endpoints
      - `$ git bisect good <good-commit-id>`
      - `$ git bisect bad <bad-commit-id>`
    - After that, git will checkout to some commit
    - If that commit is good
      - `$ git bisect good`
    - If that commit is bad
      - `$ git bisect bad`
    - After several rounds, git will checkout to first bad commit

Git - push to remote

- git remote
  - Manage the remote repository
  - $ git remote add origin
    https://git.cs.nctu.edu.tw/lctseng/sa-demo.git
    - Set the remote server & repo for push/pull
- git push
  - $ git push -u origin master
    - Push to the remote and create the remote master branch
    - Later on, you can simply use "git push"
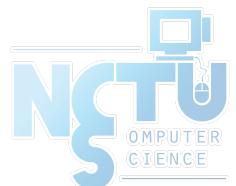
Git - pull from remote

- git clone
  - Download the full repository from remote server
  - `$ git clone https://github.com/curl/curl.git`
- git pull
  - Pull the latest revisions from remote in an existing repository