
Intro to Linux & Command Line

Based on slides from CSE 391
Edited by Andrew Hu

slides created by Marty Stepp, modified by Jessica Miller & Ruth Anderson

<http://www.cs.washington.edu/391/>

Lecture summary

- Unix and Linux operating system
 - You will not be tested
 - Just to help you understand what the environment is
- Introduction to Bash shell (command line)

Operating systems

- What is an OS? Why have one?
- Ever heard any of these words? Linux, Unix, GNU

Linux

- **Linux:** An OS framework
 - Almost every server on the internet
 - Every Android phone
- **GNU:** A bunch of small tools
- **Distribution (distro):** Different Linux types
 - e.g. Ubuntu, Debian, **Raspbian**
- Key features of Linux:
 - **open source software:** source can be downloaded
 - free to use
 - constantly being improved/updated by the community

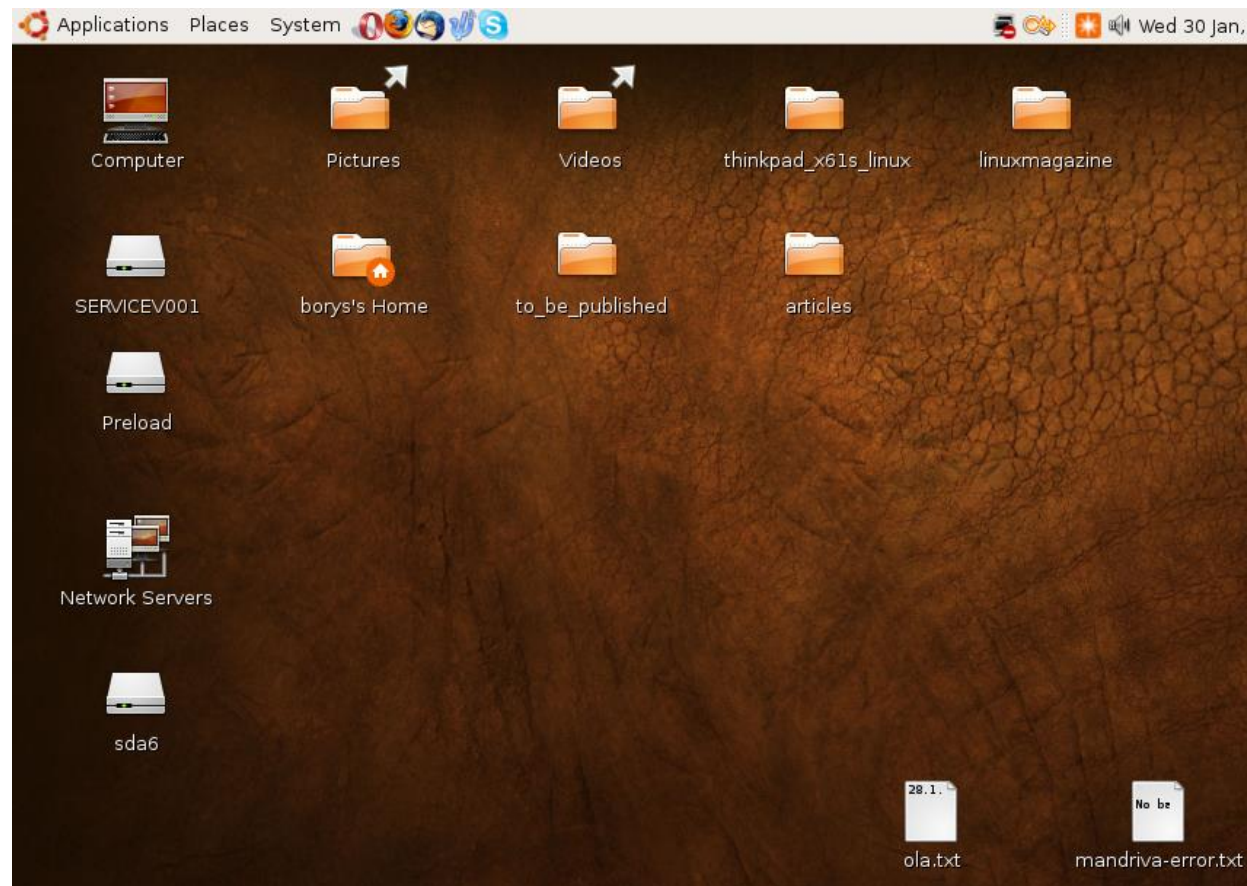


Linux vs Unix

- What the heck is a Unix?
 - Early OS, the progenitor of Linux
 - Nobody really uses Unix anymore
- Many people use them interchangeably
 - <http://unix.stackexchange.com>
- Don't worry about it

Linux Desktop

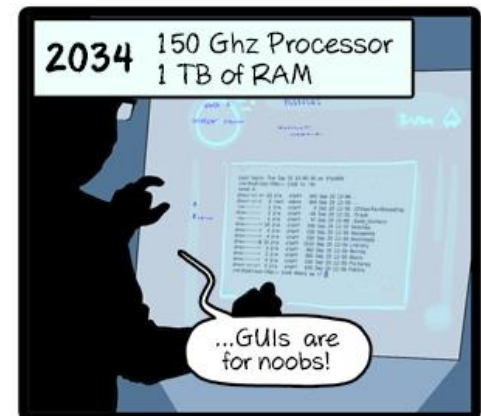
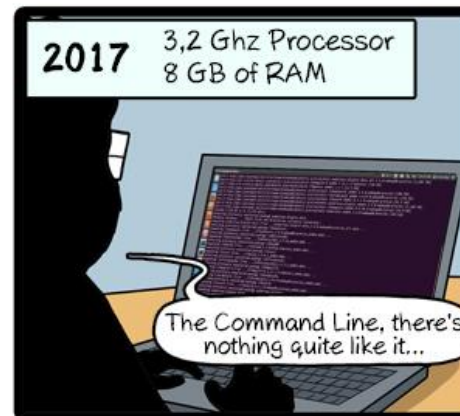
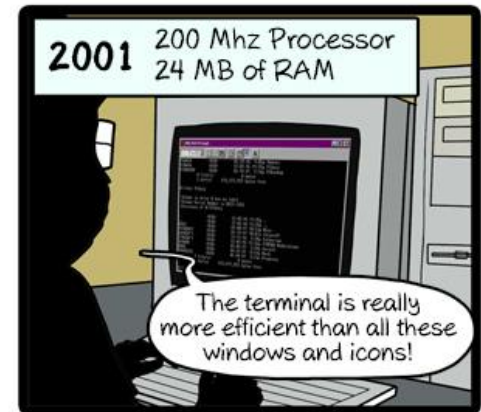
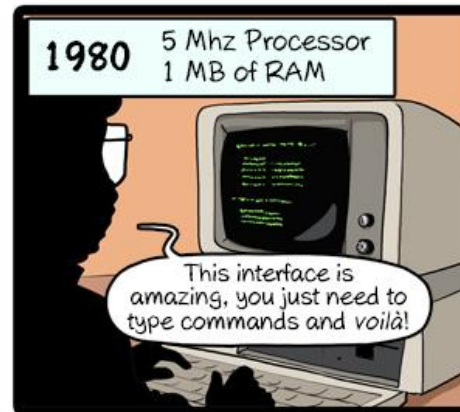
- Very similar to navigating Windows or OS X



Things you can do in Linux

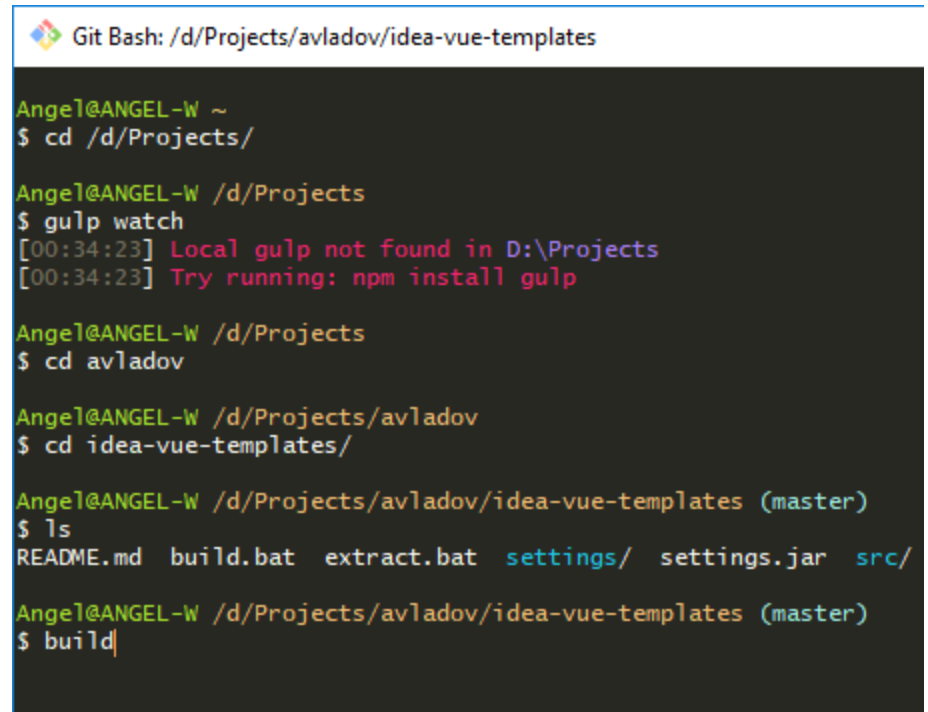
- Browse the internet
- Install and play games
- Play Music and Videos
- IM, Skype
- Basically: Everything

Command Line



Shell

- **shell**: uses user input to manage the execution of other programs.
 - Runs in a text window
 - User types commands, the shell runs the commands
- We will use Bash
 - Most commonly used shell on Linux
- Why should I learn to use a shell when GUIs exist?



```
Git Bash: /d/Projects/avladov/idea-vue-templates

Angel@ANGEL-W ~
$ cd /d/Projects/

Angel@ANGEL-W /d/Projects
$ gulp watch
[00:34:23] Local gulp not found in D:\Projects
[00:34:23] Try running: npm install gulp

Angel@ANGEL-W /d/Projects
$ cd avladov

Angel@ANGEL-W /d/Projects/avladov
$ cd idea-vue-templates/

Angel@ANGEL-W /d/Projects/avladov/idea-vue-templates (master)
$ ls
README.md  build.bat  extract.bat  settings/  settings.jar  src/

Angel@ANGEL-W /d/Projects/avladov/idea-vue-templates (master)
$ build
```

Why use a shell?

- Why should I learn to use a shell when GUIs exist?
 - faster
 - **work remotely**
 - programmable
 - **can handle repeated commands**

Navigating the file system

<code>ls</code>	lists files in a directory
<code>ls -l</code>	lists all files in directory with details
<code>cd [dir]</code>	<u>changes</u> the working <u>d</u> irectory

GUI (you don't see this)

Shell (what you see)



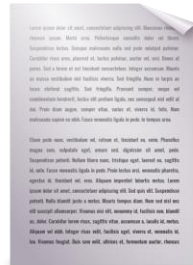
Me



CSE



Docs



Notes

\$ | `ls` |

Navigating the file system

<code>ls</code>	lists files in a directory
<code>ls -l</code>	lists all files in directory with details
<code>cd [dir]</code>	<u>changes</u> the working <u>d</u> irectory

GUI (you don't see this)

Shell (what you see)



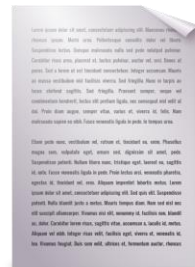
Me



CSE



Docs



Notes

```
$ ls
```

```
Me CSE Docs Notes
```

```
$ |
```

Navigating the file system

<code>ls</code>	lists files in a directory
<code>ls -l</code>	lists all files in directory with details
<code>cd [dir]</code>	<u>changes</u> the working <u>d</u> irectory

GUI (you don't see this)

Shell (what you see)



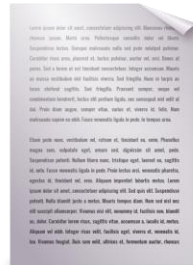
Me



CSE



Docs



Notes

```
$ ls
```

```
Me CSE Docs Notes
```

```
$ | cd CSE
```

```
$ |
```

Navigating the file system

<code>ls</code>	lists files in a directory
<code>ls -l</code>	lists all files in directory with details
<code>cd [dir]</code>	<u>c</u> hanges the working <u>d</u> irectory

GUI (you don't see this)

Shell (what you see)



142



143



391

```
$ ls
```

```
Me CSE Docs Notes
```

```
$ cd CSE
```

```
$ |
```

Directory commands

command	description
<code>ls</code>	list files in this directory
<code>ls -all</code>	list all files in this directory <i>with details</i>
<code>ls [dir]</code>	list files in a given directory
<code>pwd</code>	<u>p</u> rint the current <u>w</u> orking <u>d</u> irectory
<code>cd [dir]</code>	<u>c</u> hanges the working <u>d</u> irectory
<code>mkdir [dir]</code>	create a new directory
<code>rmdir [dir]</code>	delete a directory (must be empty)

- some commands (`cd`, `exit`) are part of the shell ("builtins")
- others (`ls`, `mkdir`) are separate programs the shell runs

Relative directories

directory	description
.	the directory you are in ("working directory")
..	the parent of the working directory (../.. is grandparent, etc.)
*	everything in a directory
[string1]*[string2]	everything in a directory that starts and ends with the specified strings
*.java	everything ending with the .java suffix
~	your <u>home</u> directory (on many systems, this is /home/ <i>username</i>)

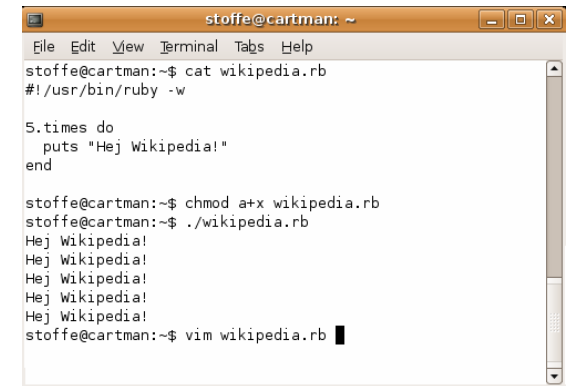
File commands

command	description
cat [file]	print out the contents of file (named for <u>con</u> catenate)
pico [file]	edit a file in the working directory
<i>(while running)</i> CTRL+C	stops the currently running program
wget [url]	download a file from the internet
alias short='long'	create a short version of a long command

- CTRL is often written as ^
 - CTRL+C would be ^C

Shell commands

- many accept **arguments** or **parameters**
 - example: cp (copy) accepts a source and destination file path
- a program uses 3 streams of information
 - **input**: comes from user's keyboard
 - **output**: goes to console
 - **errors** can also be printed (by default, sent to console like output)
- parameters vs. input
 - *parameters*: before Enter is pressed; sent in by shell
 - *input*: after Enter is pressed; sent in by user



```
stoffe@cartman: ~  
File Edit View Terminal Tabs Help  
stoffe@cartman:~$ cat wikipedia.rb  
#!/usr/bin/ruby -w  
  
5.times do  
  puts "Hej Wikipedia!"  
end  
  
stoffe@cartman:~$ chmod a+x wikipedia.rb  
stoffe@cartman:~$ ./wikipedia.rb  
Hej Wikipedia!  
Hej Wikipedia!  
Hej Wikipedia!  
Hej Wikipedia!  
Hej Wikipedia!  
stoffe@cartman:~$ vim wikipedia.rb
```

Command-line arguments

- Options: hyphen '-' followed by a letter
 - `gcc program.c -o program.o`
 - `-o` = “output”
 - some are longer words preceded by two - signs, such as `--count`
- many programs accept a `--help` or `-help` option to give more information about that command (in addition to man pages)
 - or if you run the program with no arguments, it may print help info

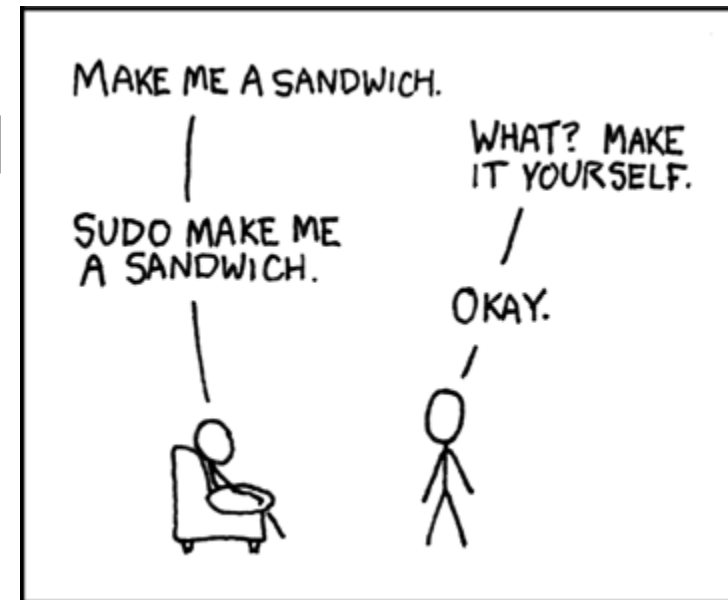
File commands

command	description
cp	copy a file
mv	move or rename a file
rm	delete a file
rm -r [DIR]	delete a directory and all of its contents
touch	create a new empty file, or update its last-modified time stamp

- Use “-i” (interactive) to force confirmation
 - `$ rm -r -i *`
remove file ‘bitcoinwallet.key’? no
- `touch prog.java -t 149204010000`
 - `-t [[CC]YY]MMDDhhmm`

SUDO

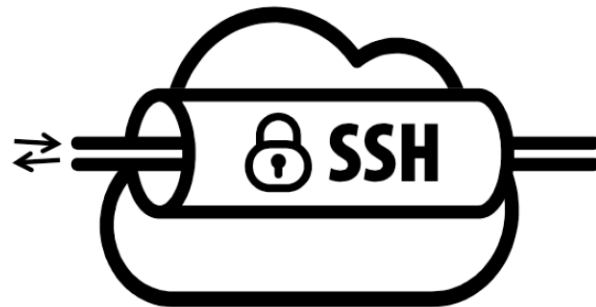
- super user do
 - Pronounced like “sue-dough” or “pseudo”
- similar to “execute as Admin” in Windows
- `sudo apt-get install [package]`
- `sudo apt-get update`
- `sudo rm [someone else's file]`



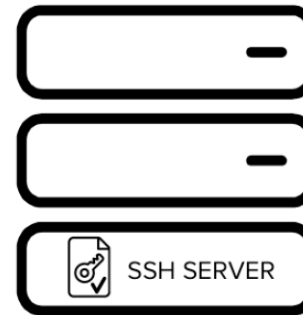
SSH

- Secure Shell
- Control a computer remotely, through a shell
- Do work from the comfort of your personal computer

Your PC/Laptop



Remote server
e.g. Vergil/Hyak

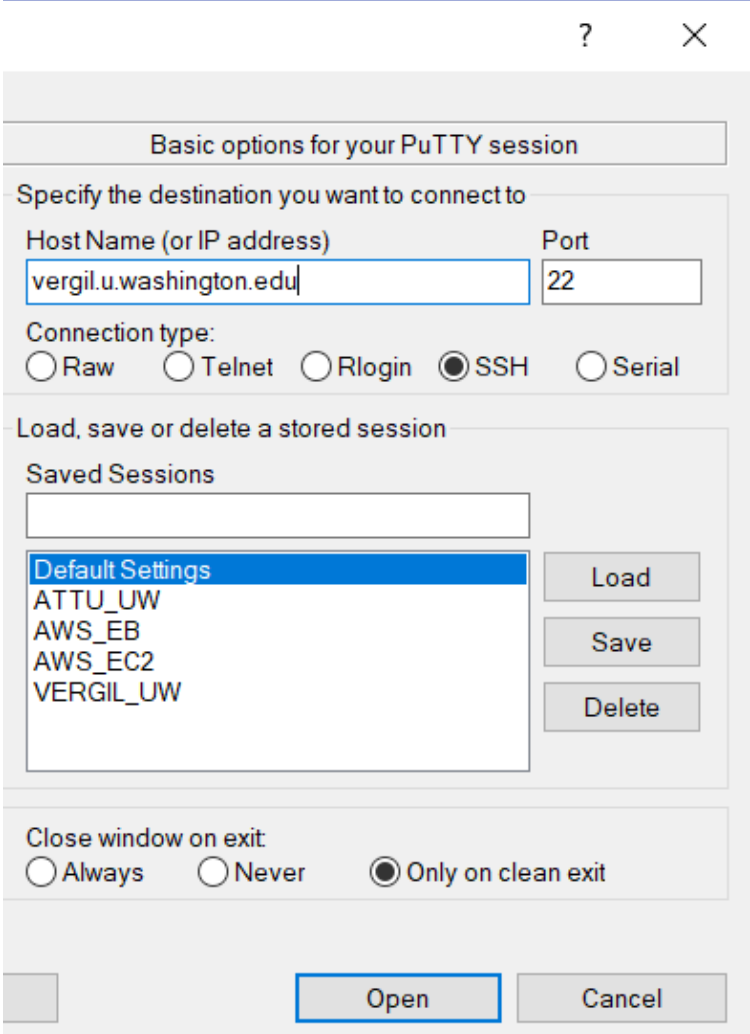


APPLICATION
SERVER



SSH Clients

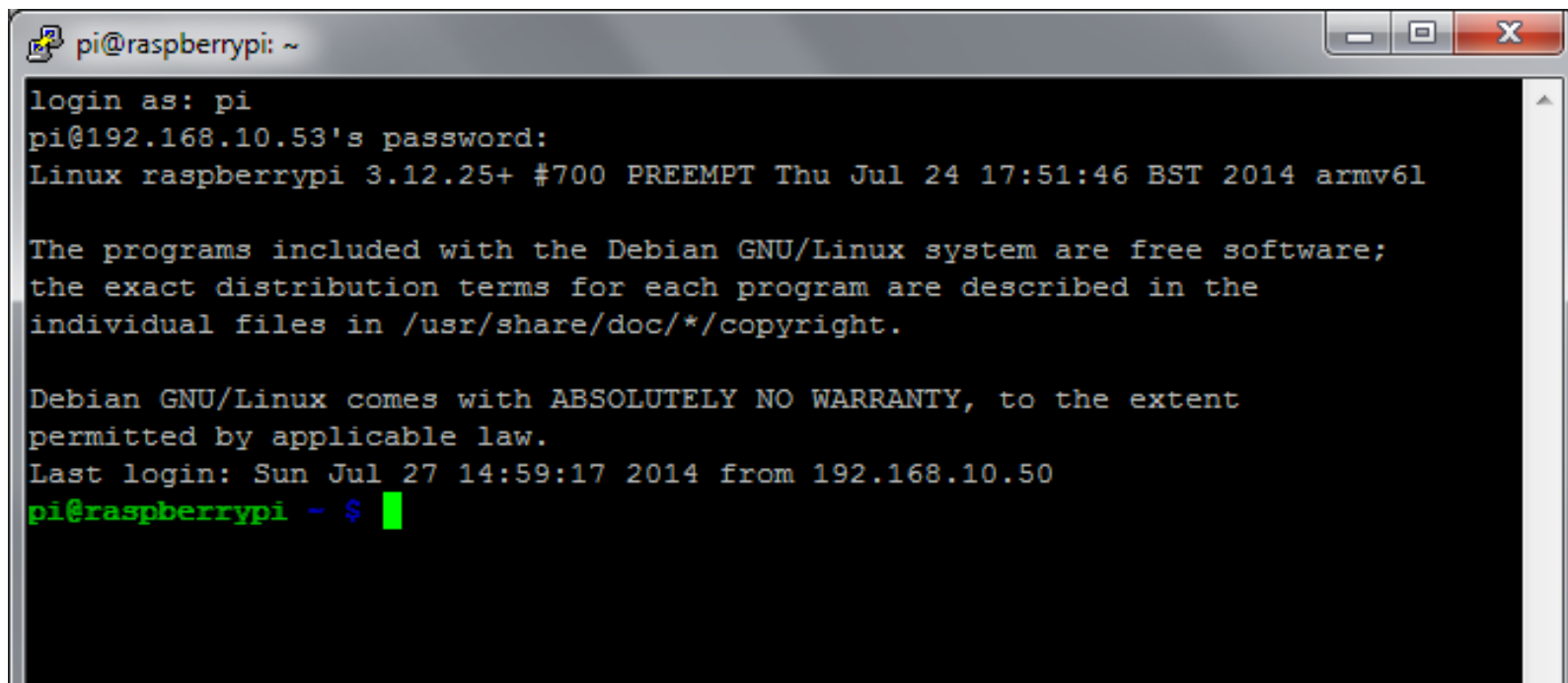
- Windows: PuTTY
 - type in server URL
- Mac: Built in shell client
 - just type ssh [user@example.com](#)
- Port should be 22



The screenshot shows the PuTTY configuration window titled "Basic options for your PuTTY session". It includes fields for "Host Name (or IP address)" with the value "vergil.u.washington.edu" and "Port" with the value "22". The "Connection type" section has radio buttons for "Raw", "Telnet", "Rlogin", "SSH" (which is selected), and "Serial". Below this is a section for "Load, save or delete a stored session" with a list of "Saved Sessions" including "Default Settings", "ATTU_UW", "AWS_EB", "AWS_EC2", and "VERGIL_UW". To the right of the list are "Load", "Save", and "Delete" buttons. At the bottom, there is a "Close window on exit" section with radio buttons for "Always", "Never", and "Only on clean exit" (which is selected). The "Open" button is highlighted at the bottom right.

Using SSH

- Shell interface
- This is why it's important to know how to use command line!



A screenshot of a terminal window titled "pi@raspberrypi: ~". The window shows the output of an SSH login. The text displayed is as follows:

```
login as: pi
pi@192.168.10.53's password:
Linux raspberrypi 3.12.25+ #700 PREEMPT Thu Jul 24 17:51:46 BST 2014 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Jul 27 14:59:17 2014 from 192.168.10.50
pi@raspberrypi ~ $
```


How to collaborate on code?

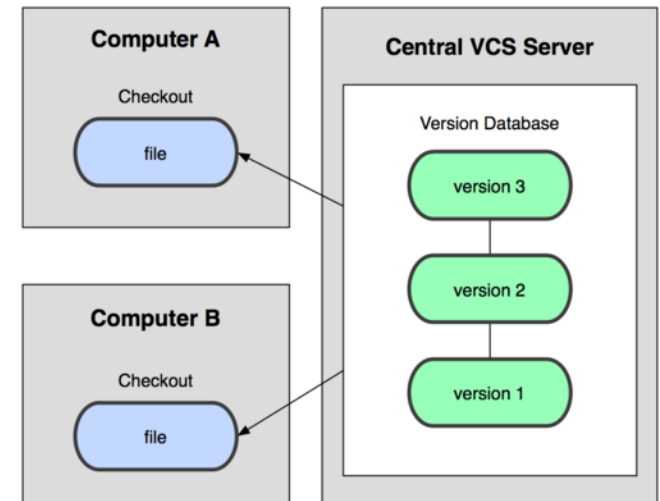
What have you tried in the past?

Version Control

- When you work on a group project, how do you share files?
 - Google Docs, OneDrive, Dropbox
 - These are examples of real-time centralized version control
- What happens when two people try to edit the same line?
- What happens when somebody deletes everything?
- How do you know who made what changes, and when?

Real-Time Centralized

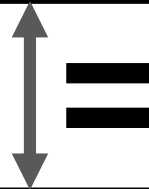
- You're probably very familiar with using a real-time centralized Version Control System
 - E.g. Google Docs, OneDrive, Dropbox
- Pros
 - Easy to set up, use, and share
 - Instant feedback from other editors
- Cons
 - Can't really resolve conflicts
 - Other editors "stepping on your toes"
 - You don't know who, when, or what each change was



What is a conflict?



```
print("The meaning of life is...");
```



```
print("The meaning of life is...");
```

```
print("The meaning of life is 42");
```



```
print("The meaning of life is...");
```

```
print("The meaning of life is that it ends");
```

What is a conflict?



```
print("The meaning of life is 42");
```



CONFLICT



```
print("The meaning of life is...");
```

```
print("The meaning of life is that it ends");
```

What is a conflict?



```
print("The meaning of life is 42");
```

OR

```
print("The meaning of life is that it ends");
```

What is a conflict?



```
print("The meaning of life is 42");
```

```
print("The meaning of life is that it ends");
```

What is a conflict?

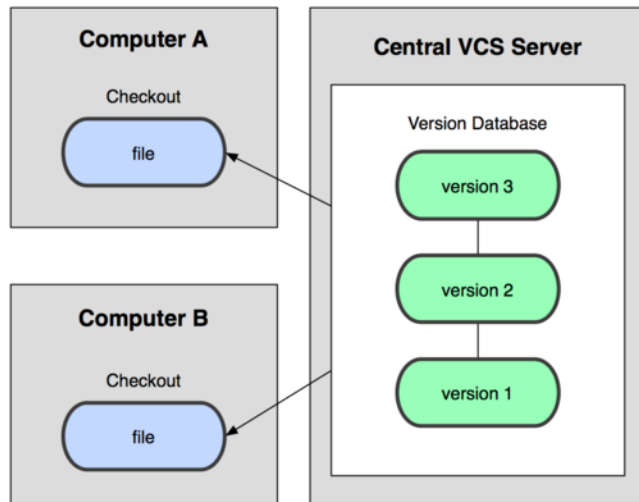


```
print("The meaning of life is that it ends");
```

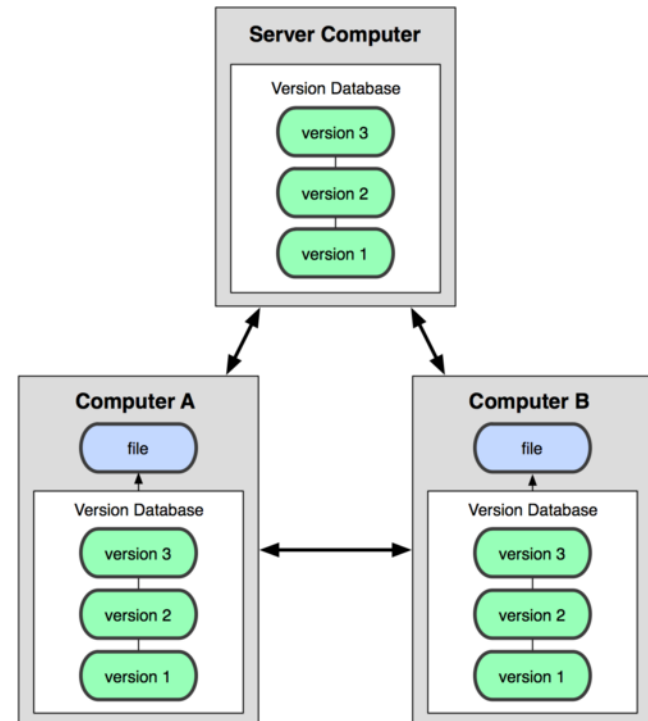


Distributed VCS

Centralized Model



Distributed Model



Distributed VCS

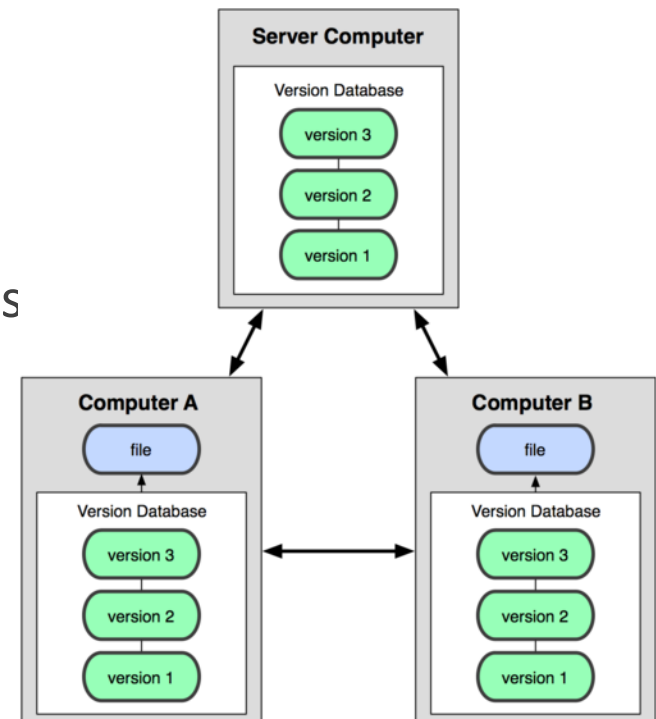
- A distributed VCS lets each user have their own copy of the database (called a repository or “repo”)
- Users can send their versions to each other
 - If there is a conflict, the sender decides which version to use

- Pros

- Each user’s repo is equally valid
- Users can choose how to resolve conflicts
- Detailed log of every change ever

- Cons

- More difficult to set up and use

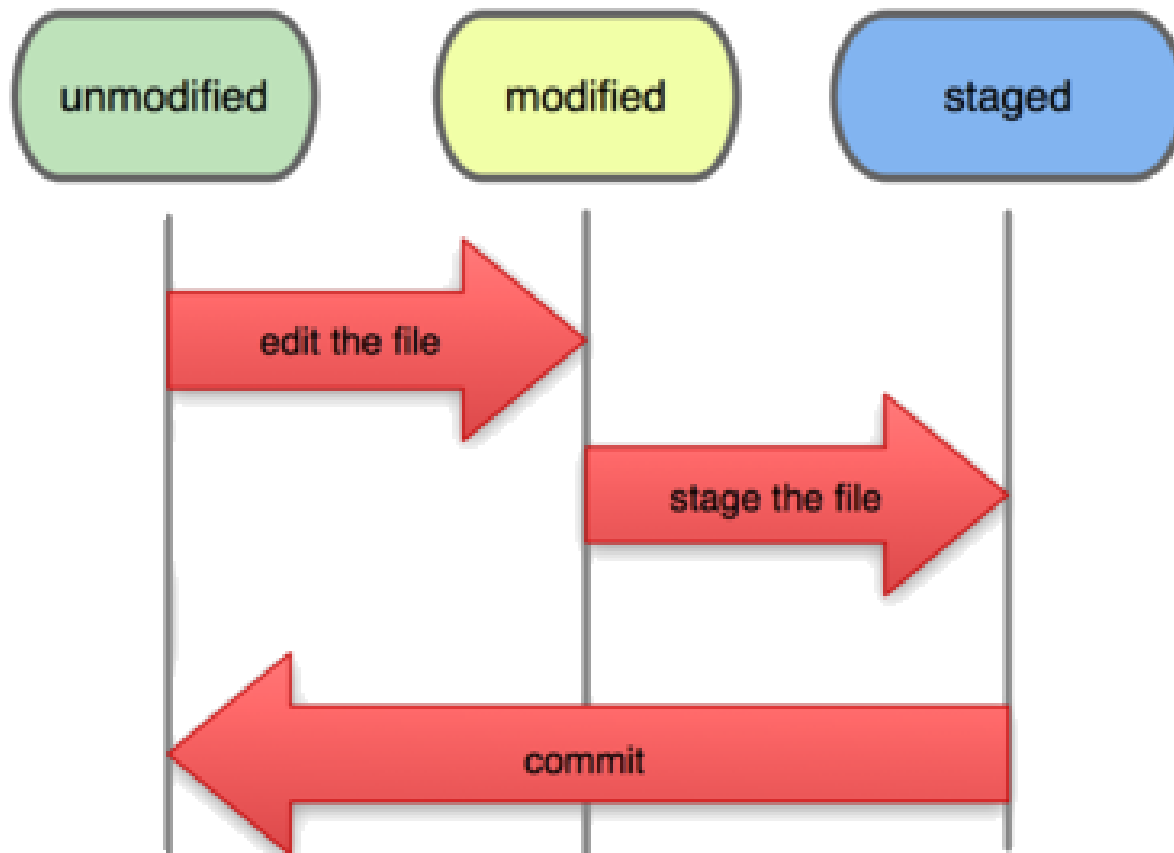


Git

- Git is a distributed VCS
- Most commonly used VCS
- Created by Linus Torvalds to support the development of Linux

Git on your local machine

File Status Lifecycle

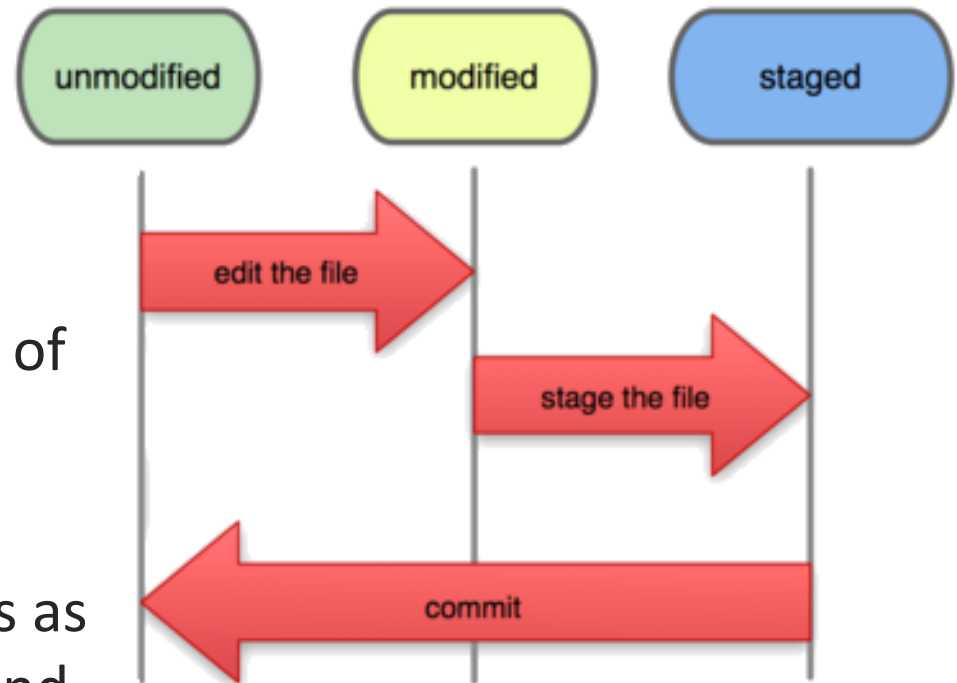


Basic Workflow

Basic Git workflow:

1. **Edit** files in your working directory
2. **Stage** files, adding snapshots of them to your staging area
3. **Commit**, which takes the files as they are in the staging area and stores that snapshot permanently

File Status Lifecycle



Use Good Commit Messages



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

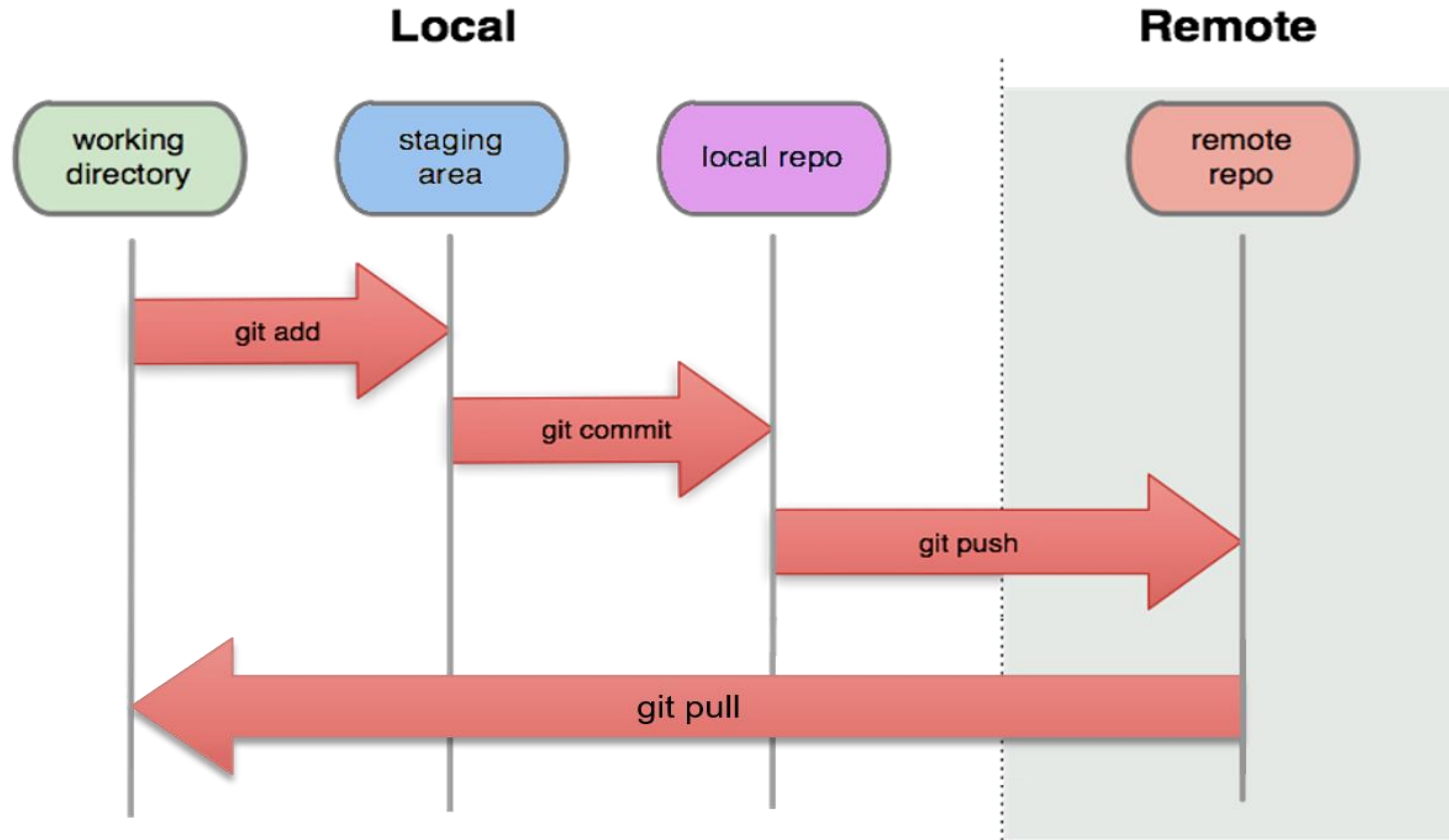
AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Commit message style

- Start your commit message with either +,-,=
 - + when functionality was added
 - - when functionality was taken away
 - = for small edits that don't change the functionality
- Less than one sentence
 - You can usually omit language like “added” or “removed” since it's implied by the + or -
- Use multiple commits as needed
 - If you added two separate features, make two separate commits!

**When in doubt,
Commit more, rather than less**

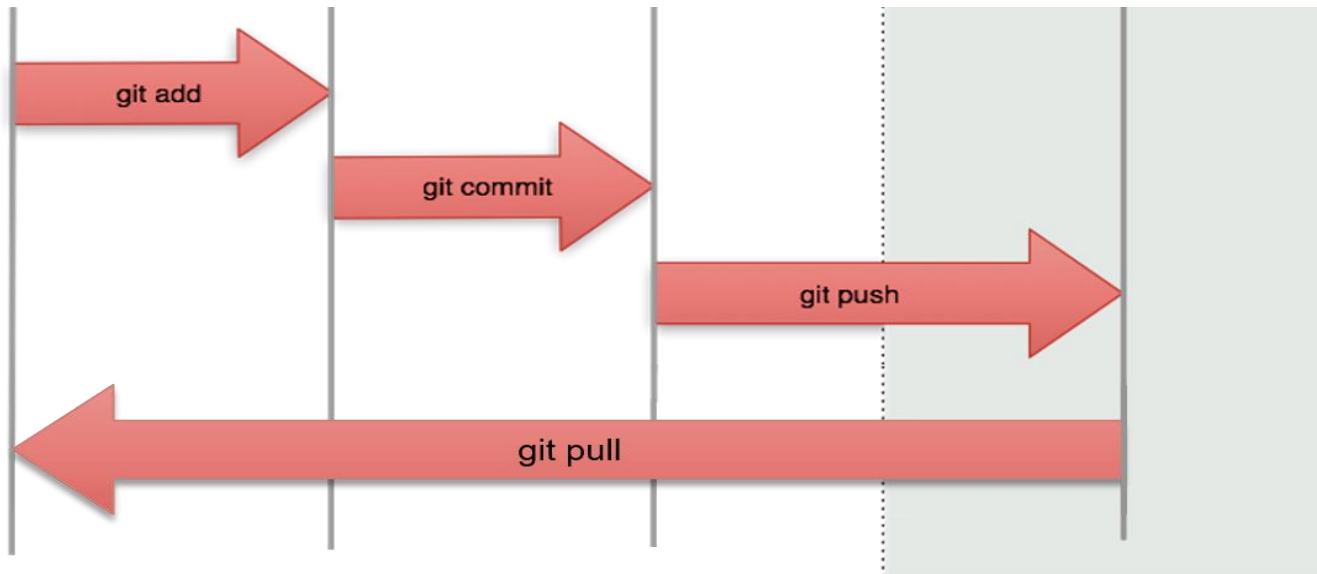
Git with a remote server



- **Push** commits to the server to publish your commits
- **Pull** from the server to get other users' commits

Git shell commands

command	description
<code>git status</code>	See which files are/need to be staged
<code>git add [-A] [file]</code>	Add [file] to the staging area
<code>git commit -m [mssg]</code>	Commit these changes and label [mssg]
<code>git push [origin master]</code>	Send your commits to the server
<code>git pull</code>	Download updates from the server



Vi / Vim

- Powerful command line text editor
- Notoriously infuriating
- [“how to exit vim” is one of the most commonly asked questions on Stack Overflow](#)



I Am Devloper
@iamdevloper



Following

I've been using Vim for about 2 years now, mostly because I can't figure out how to exit it.

↩ Reply ↻ Retweet ★ Favorite ⋮ More

RETWEETS
4,846

FAVORITES
2,105



4:56 AM - 18 Feb 2014

Just memorize these fourteen contextually dependant instructions



Exiting Vim

Eventually

ONLY?

@ThePracticalDev

Basic Vim Commands

command	description
:w	Write changes to the file
:x	Write and quit
:q!	Quit without saving
i or a	Enter Insert mode (AKA normal typing mode)
v	Enter Visual mode (AKA select mode)
ESC	Exit Insert or Visual mode