How to be a 1337 h4ck3r: Git

An introduction to Git, version control, and command line stuff

by Samsara Counts and Pat Cody (RIP)
Slides adapted from Neel Shah and Phil Lopreiato

DISCLAIMER: GW ACM does not promote illegal and/or unethical activity (hacking or otherwise), and that's not the focus of this workshop!

Most importantly, computer scientists come in ALL forms and that's why our field is so great!

To quote Melinda Gates:

"Not every good idea comes wrapped in a hoodie."

"It's time the world starts recognizing that the next Bill Gates might not look anything like the last one."

1337 h4ck3rz??

- Why 1337 h4ck1ng?

...well, you've seen the *hacker* stereotype (that's not any sort of true, smh)

The cool thing about hackers is that **they're powerful af**. And you know what?

So are you, when you learn how to use Git and the command line!

1337 h4ck3rz??, cont.

git

- Why Git?
- Why the command line?

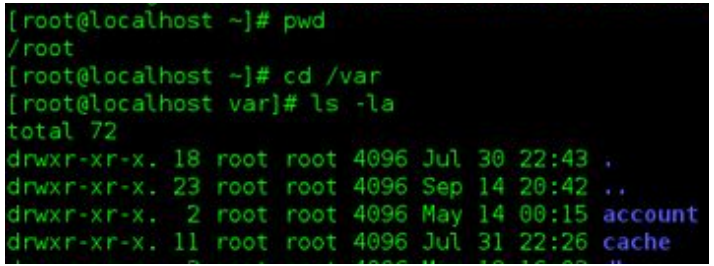
git and the command line are the essential tools of good programming: streamlining your workflow and keep track of different versions of your code

BASH

THE BOURNE-AGAIN SHELL

Goals of this workshop:

1. you know the **basic tools** to be a good software developer:
    a. command line interface
    b. git
    c. vim
2. you know some basic good practices for using those tools
3. You have a jumping off point to **build up a digital portfolio** and dive into software projects



This workshop

PART I: command line stuff

A. the command line + commands
B. Command line hacking: make a directory!
C. More tools!
D. More command line hacking: edit a file in vim!

PART II: Git and version control

A. git background
B. get git; try out github
C. github background
D. Clone our git repo + add a file; make your first pull request

Your Workshop TODO:

1. Put away your laptop (until we say so) and turn off your phone

2. Follow along!

3. Ask lots of questions

4. When it's time to try things out, READ OUR DOCS. *consider it practice for being a good programmer!*

Resources + Documents for this workshop

1. Git & command line Workflow Cheatsheet
2. Git Cheatsheet
3. Workshop Repository + README
4. Presentation Slides

PART I: command line stuff

shells and UNIX-like environments

"Any sufficiently advanced technology is indistinguishable from magic."

- Arthur C Clarke

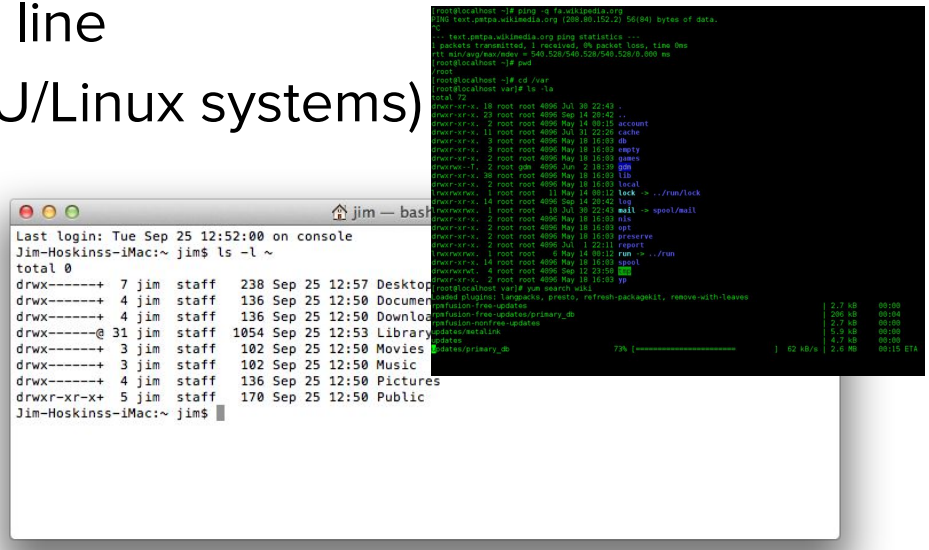
Wat is a shell

- a user interface for access to an operating system's services
- Can be a command line interface or a graphical user interface (GUI)

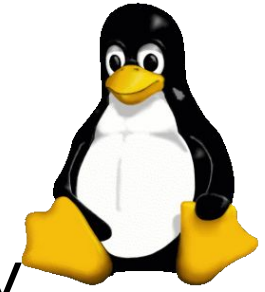
```
Last login: Thu Sep 20 21:51:46 on ttys001
echo $SHELL

samsaracounts ~ echo $SHELL
/bin/zsh
samsaracounts ~
samsaracounts ~ ls
Applications                    Sites
CAAR                            Voices
DCWiT                           acm
```

Wat is a command line interface

- a text-based application for viewing, handling, and manipulating files directly on your computer
- the GNU project's shell (Bourne Again SHell) is the most common command line interface (built into Macs & GNU/Linux systems)
- What 1337 h4ck3rz/Comp Scis use



basic Bash commands

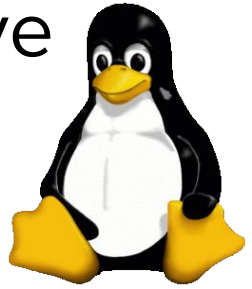


ls [dirpath]	-	list files in directory
cd [dirpath]	-	change directory to **dir**
mv [file] [dirpath]	-	move **file** to **dir**
mkdir [dirname]	-	make directory **dirname**
rm [file]	-	delete a file/directory*

*** - CAN'T BE UNDONE, rm'ing a directory needs -r recursive option**

basic Bash commands, cont.

> [**filename**]	-	make a new file called **filename**
~	-	reference to the home directory
.	-	reference to this (current) directory
..	-	reference to directory above current directory



It’s hacking time! (part I)

Do now (on your laptop):

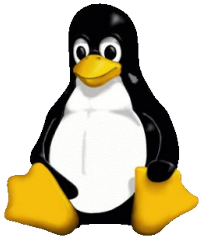
1. open go.gwu.edu/git for git & command line cheat sheets, basic instructions, + links
2. open go.gwu.edu/acmws
3. **Then**
   - **Mac/Unix: open** Terminal (command line)
   - **Windows:** download **git bash** and **open it**

Do now, cont: (on the command line):

1. list visible directories from current position
2. Make a directory
3. Navigate into that directory
4. make a file called ‘new.txt’
5. Copy a file from another location and put it in this folder (.)
6. Remove that file

DEMO I!

More Bash commands



pwd - print working directory

head [file] - show first 10 lines of file

tail [file] - show last 10 lines of file

clear - clear screen

grep “expression” [dir] - search for expression in **dir**

Intro to vim

- **vim** is a multipurpose text editor for the command line
- The name vim comes from “vi improved,” where vi is an older, less cool command line text editor

vim editing modes

- **normal mode** allows you to highlight segments of text, jump to line numbers, and enter commands like **write** and **quit**
- **insert mode** lets you enter and delete text
- **visual mode** lets you copy and paste

Vim commands

vim [filename]	-	open [filename] with vim
esc	-	go back to normal mode
i	-	go to insert
:wq	-	write (save) and quit
:w	-	write
:q!	-	force quit (doesn’t save)

It’s hacking time! (part II)

Do now (on the command line):

1. enter ‘**cd**’
2. enter **vim** and edit ‘**new.txt**’
3. In vim, edit that file
4. Write and exit vim
5. Remove that file

DEMO II!

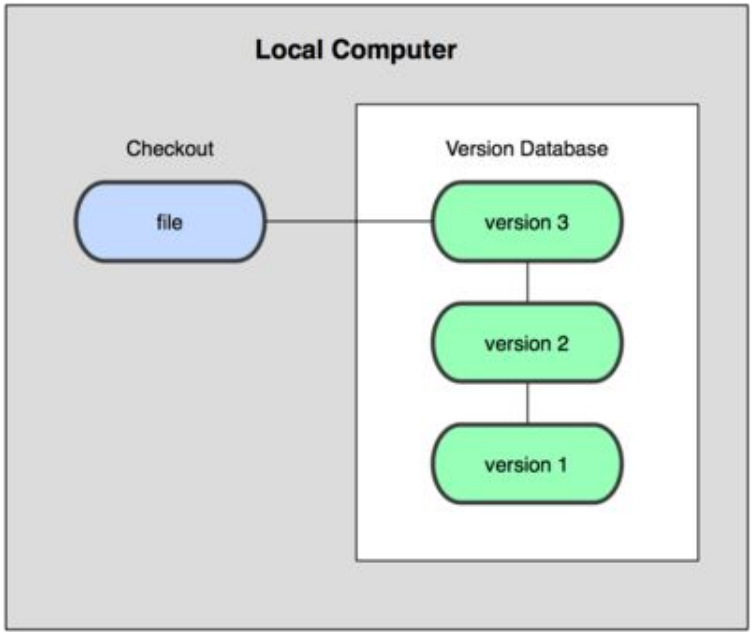
PART II: Git and version control

Git your life together

Slides adapted from Phil Lopreiato and Neel Shah

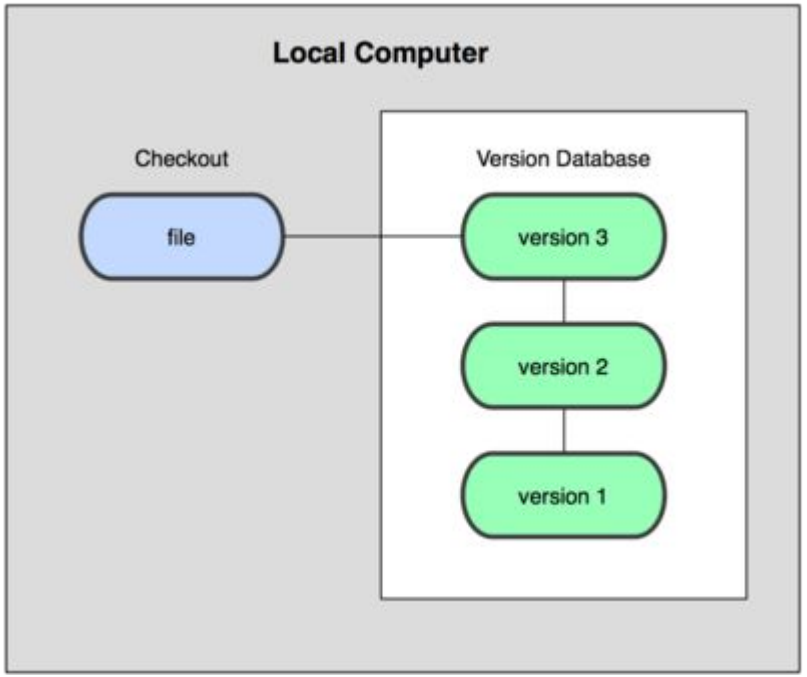
What is git?

- Git is a type of *version control software*
  - It's a way to track changes to files
- How would you do that?
  - Many zip files
  - Timestamped directories
  - A specially ordered collection of stones
  - Magic?
- Somebody smart decided to synchronize their files with a local database - this is **local version control**



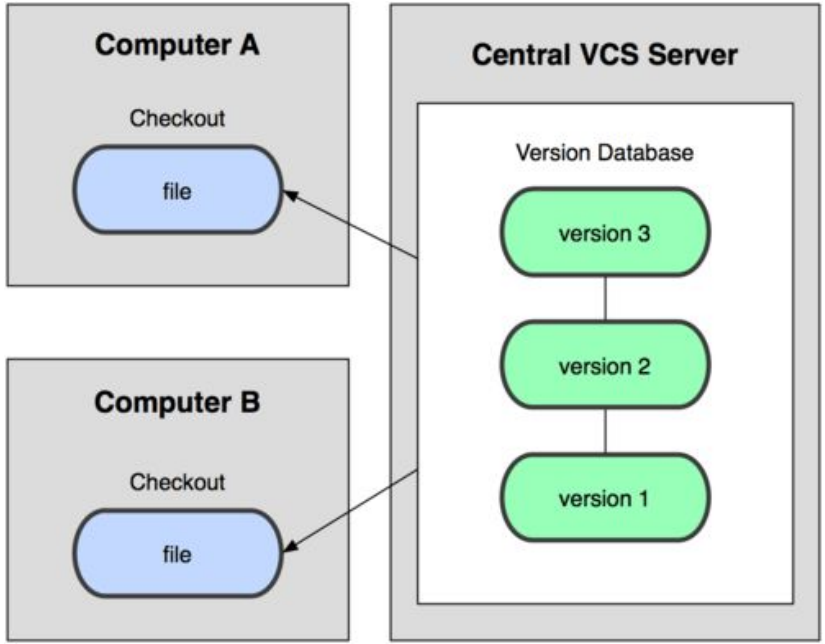
How is git structured?

- You keep projects in **repositories** (repos) that are backed up on a server
- you work in local copies of repos, then push changes to server



What is version control?

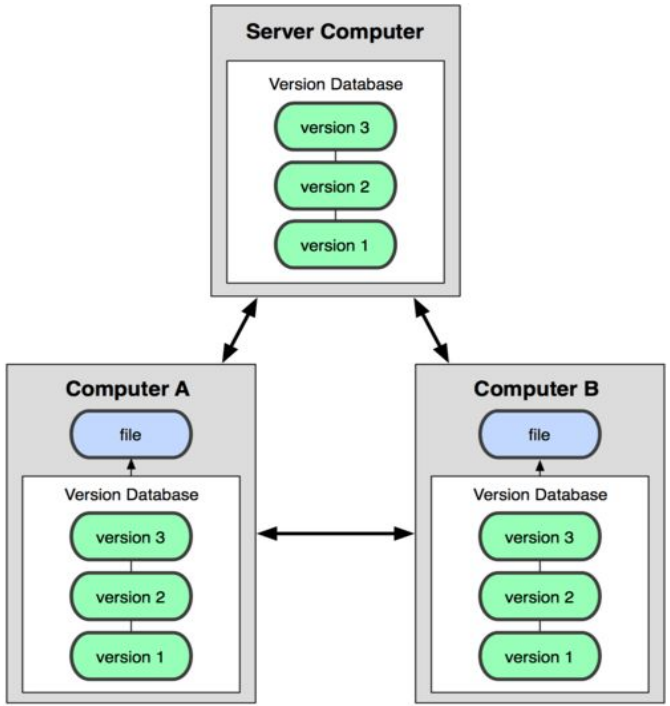
- But what if you want to share your changes with someone else?
  - Easy - just send the database to a server!
- This is called **centralized version control**
- But what happens if the central server goes down?



Optional deep dive!

Version control, cont.

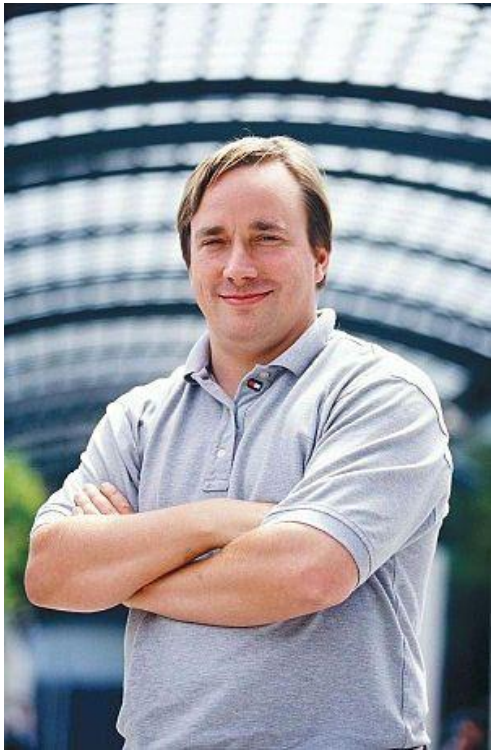
- Just make clients check out the full contents of the repository then!
  - Now, everybody who works on a project has a full copy of the history in case something happens
- This is called **distributed version control**
  - Many common *VCS* systems use this (git or Mercurial, for example)
- **GitHub** (github.com) is a popular web-based Git repository hosting service



Optional deep dive!

Git Ancient History

- In 2005, the Linux Kernel project needed a new source control system
- Linus Torvalds set out to write his own
  - popular version control software at the time was not “good enough” for him
  - needed to be distributed and protect against corruption
- Development began April 3, the project was announced April 6, became self-hosting on April 7, and used in the kernel by June



Optional deep dive!

How does it work?

- Git is a **Directed Acyclic Graph** of repository “snapshots”
- Every change is initially done locally
- Every change has verified integrity
  - The repository is “checksummed” after every change
  - That checksum is used to refer to each **commit**

“In many ways you can just see Git as a filesystem — it is content-addressable, and it has a notion of versioning, but I really really designed it coming at the problem from the viewpoint of a filesystem person (hey, kernels is what I do), and I actually have absolutely zero interest in creating a traditional SCM system.” - Linus

It’s (git) hacking time! (part III)

How can I “Git” Git?

- http://git-scm.com/downloads
- You can now use git in a terminal or “Git Bash” on Windows

Configuration...

```
$ git config --global user.name "YOUR NAME"
```

```
$ git config --global user.email "YOUR EMAIL ADDRESS"
```

Do now: make an account!

- Go to https://github.com/join

and create an account

(or log in)

- Get familiar with git!
  - https://try.github.io/

(Demo unnecessary)

What's this GitHub?

- GitHub makes coding "social" by providing a Git repository hosting service that maintains all of the "distributed" features of Git and adds a social aspect
  - Issue Trackers
  - Code Releases
  - Project Websites
  - Anyone can contribute!

basic git workflow commands

git clone url.of.repo - create local copy of repo on your computer

git add * - add all files to commit

git commit -m “init” - save all your edits with message “init”

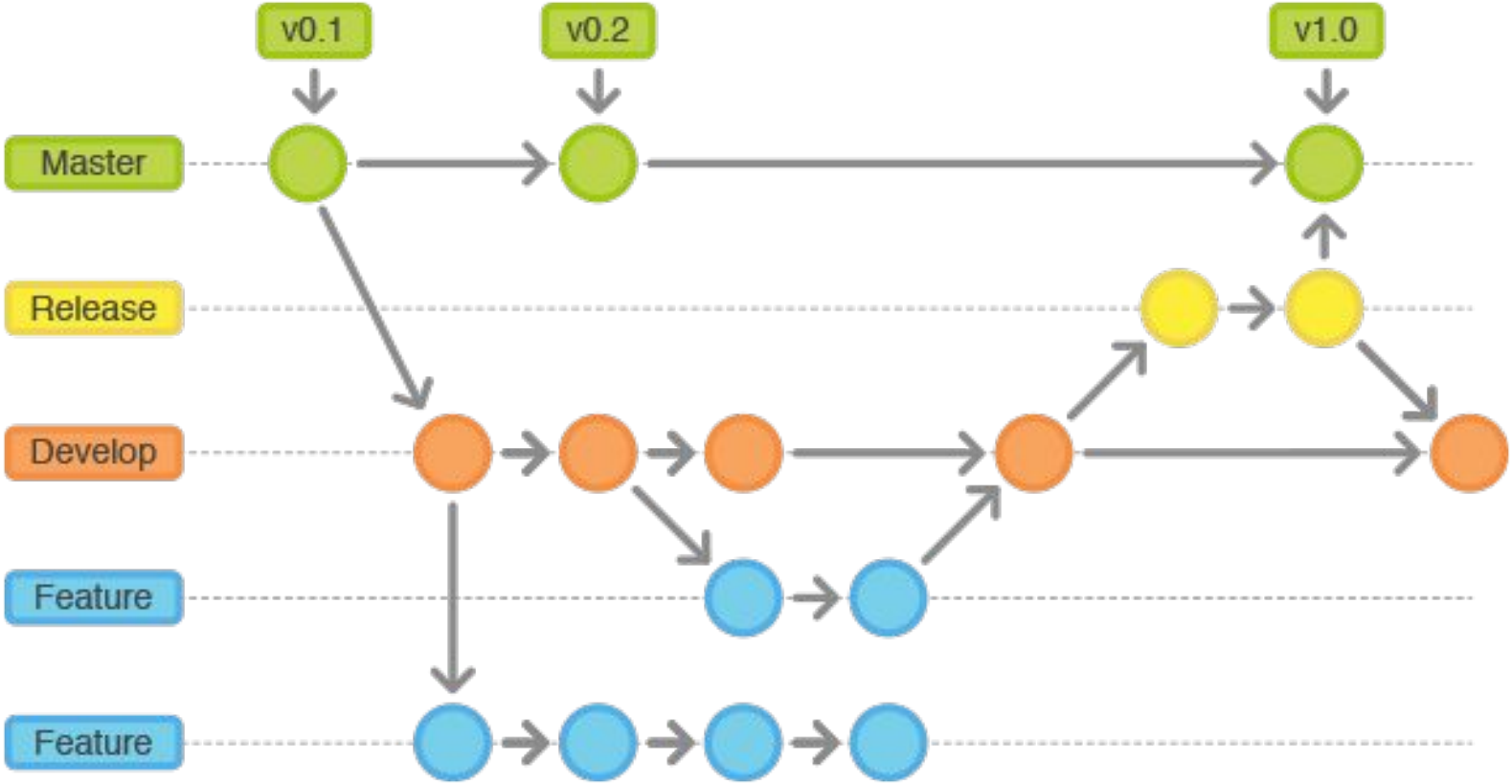
git push - push your changes to origin branch of repo you’re working on

git pull - get changes from remote to local branch of repo

branching on git

- on a project, you're going to have a bunch of different features + ideas in progress at a time – some of which are ready to go, and others which aren’t
- **branch** - an environment where you can try out new ideas. Changes you make on a (named) branch don't affect the **master** branch.
- Branching exists to help you manage this workflow.

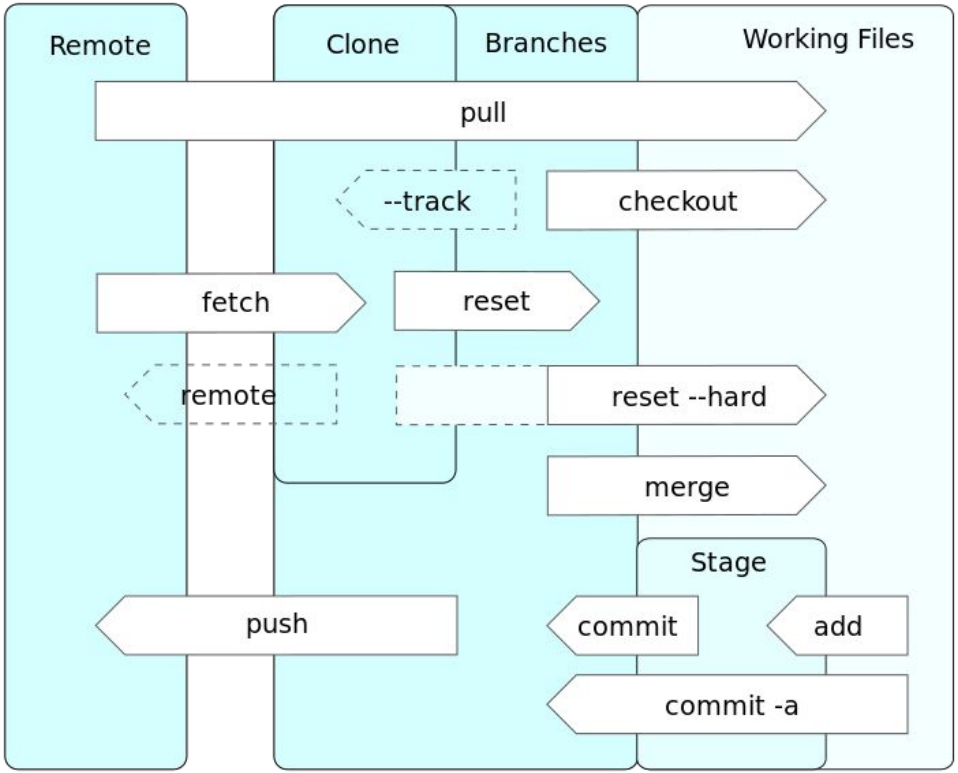
Cloning, forking, and branches



branching git workflow commands

git checkout -b new_branch	- switch to new branch
git checkout branch	- switch to existing branch
git push origin branch	- push changes to origin of branch you're working on

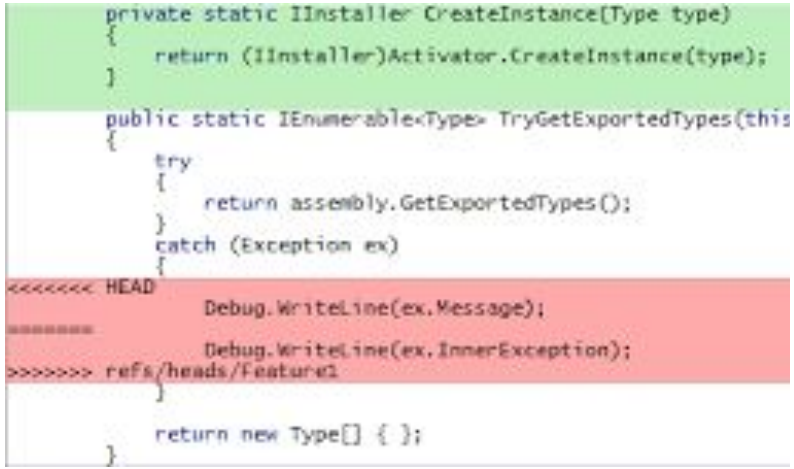
Distributed versions, visualized



Merge conflicts

merge conflict - when you merge branches that have competing commits.

- git needs your help to decide which changes to incorporate in the final merge
- can happen when people make different changes to the same line of the same file



How to fix merge conflicts

1. ***Don’t have them:***
    a. Use branches!! for new features/directions
    b. talk to your contributors about when you’re making changes to the same repo/branch!
2. If you ***do* have a merge conflict:**
    a. must manually edit conflicted file(s) to select changes to keep in the final merge.

merge/status git workflow commands

git status - shows the state of the working directory (local changes to files)

git checkout file.txt - revert file to last commit (delete local changes to file)

git rm --cached f.txt - delete file from git (not from your computer)

git log - lists commits made in repository in reverse chronological order

It’s (github) hacking time! (part IV)

Do now (on github + command line):

1. Full instructions in the README: go.gwu.edu/git
2. fork the above repository to your account
3. On the command line, navigate into the directory you made
4. Clone your fork of the repo on your computer
5. Navigate into that repo
6. Use touch to make a .txt file with your name on it (ex: samsara.txt)

Do now (on github + command line):

7. Tell git you made changes with add
8. Commit those changes with a nice Message
9. Check the status of your repo (git status)
10. Push them to your forked repo
11. On github, submit a pull request to our original repository

DEMO IV!

GitHub

- Example GitHub profiles:
  - https://github.com/aaroncoplan
  - https://github.com/samsaranc
  - https://github.com/pcodes
- Example Repositories:
  - Linux
  - The Blue Alliance
  - Composite OS
- Example Organization:
  - GWCloudLab
- Explore other repos and projects:
  - https://github.com/explore

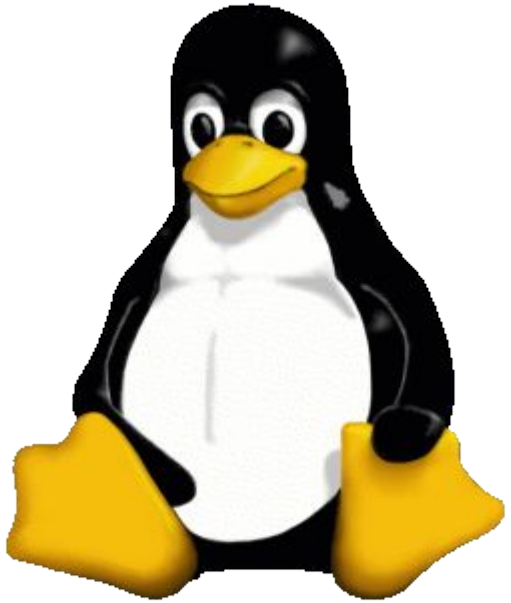
Links + Further reading

- GitHub workflow GUIDES
  - https://guides.github.com/activities/hello-world/
  - http://readwrite.com/2013/09/30/understanding-github-a-journey-for-beginners-part-1
  - http://blog.udacity.com/2015/06/a-beginners-git-github-tutorial.html
- Git branch guides
  - https://guides.github.com/introduction/flow/
- Git merge guide:
  - https://help.github.com/en/articles/resolving-a-merge-conflict-using-the-command-line
- Git documentation
  - https://git-scm.com/docs
  - http://wildlyinaccurate.com/a-hackers-guide-to-git/
- Some other Git hosts
  - Bitbucket
  - GitLab

Want to learn more about how this stuff works?

- Go to the Systems Hacking Club (SHC)
- Find out who this penguin is: ->
- Bonus points: add ssh keys to your computer and

  clone git repos with ssh



Fin.

Happy h4ck1ng!