SSW345 Model-Based Software Engineering

Course Rebaselining & Version Control

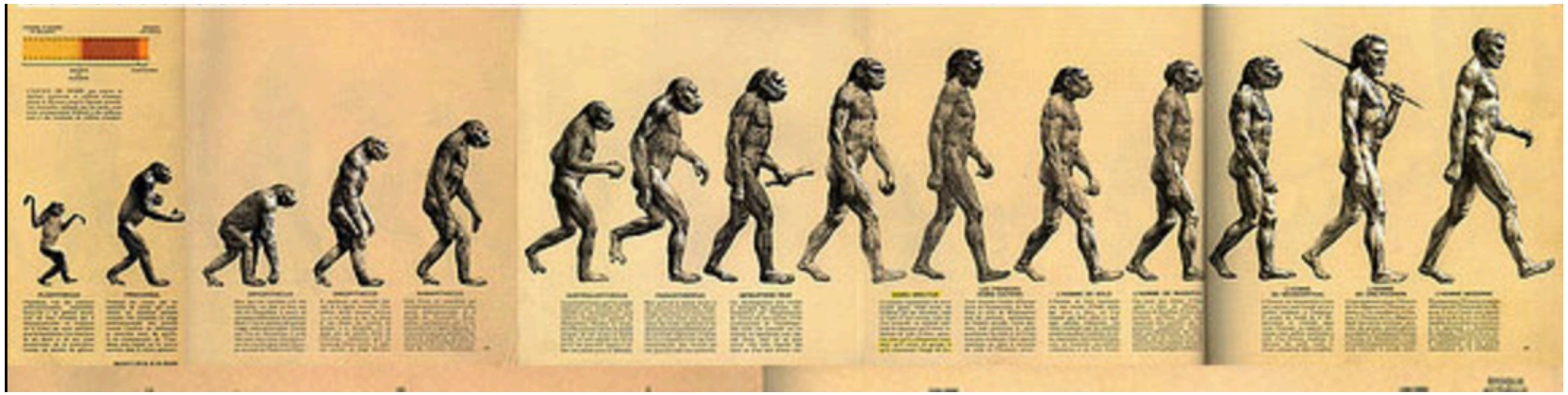
Prof. Ye Yang

This Week's Topic

- Course Rebaselining
- Version Control
- Intro to Git & Github
- Lab: Git

What Have Changed?

Evolution of Software Process Models



Waterfall Prototyping Agile/XP DevOps

So far and Plan forward…

Week	Concept	Problems Addressed	Tools
1-6	Model-based development, SPE modeling and analysis	Performance at design time	Eclipse, Papyrus, Excel
7-8	Version Control, Distributed coding platforms, RESTful	Versioning, Sharing, Archival	Git, GitHub
9-10	Code Review, Software Metrics, Testing, Web Basics	Software Quality, Refactoring	Gerrit, Github
11-12	Process, Virtualization/ Containers, Continuous Integration	Scalability, performance, portability	Docker
13-14	DevOps, Configuration Management, Interview	Scalable, Reliable Operation; time-to-market	Travis, Jenkins

This Week's Topic

- Course Rebaselining
- Version Control
- Intro to Git & Github
- Lab: Git

Version Control

Main Concepts

- Version Control
- Sharing/Workflow
- File System Management (tracks files/directories, new/move/delete)
- Centralized/Decentralized
- Parallel Maintenance
- Speed
- Intermittent Connectivity/Mobility
- Trust
- Integrity
- Transparency/Communication

Version Control (Simple)

- Here's Sue
- Sue is working by herself on a project.
- Sue works, tries things, makes mistakes, fixes, goes back, wants to know why she did something yesterday……
- …and finally finishes up (with a group of files, aka "release").
- *Sue Needs Version Control*

Sharing & Workflow (Simple)

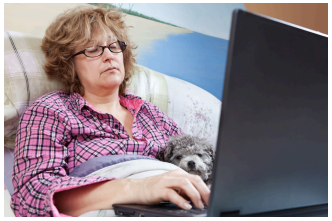
- Here's Sue and Jill, working on a (bigger) project
- Sue & Jill sometimes change different files
- Sometimes they change the same files. Avoid? Take turns? Make a mess & fix?
- *Sue & Jill need to agree on a workflow, and use a version control tool that supports it.*

Sharing & Workflow (Complicated)

- …here's a Facebook team. dozens? Hundreds? Thousands?
- Changing same or different files?
- Adding files? Moving/Renaming? Directories?
- Can they all meet & agree?
- *Big teams need to agree on "standard" workflows, and use the right tools*

Centralized .vs Distributed Version Control

- …where in the world is the team?
- Where should we keep the files?
- In one place? Problems?
- In many places? Problems?
- *Distributed teams need distributed version control.*




Parallel Maintenance

- Team successfully releases Version 1.0 of the software. People are using it. Great Joy.
- Team starts work on Version 2.0. Much Optimism.
- But if people are using Version 1.0, what happens?
- Where do we make the fix? 1.0? 2.0?
- Will there be more than 1 bug?
- What if a a new feature **HAS** to be released already in 1.0?
- *Team needs Branching & Merging*

Speed

- People need to make changes & share:

❖ Often

❖ Without Long Waits

❖ From may different locations/users

❖ Shouldn't depend on system size

- *Need efficient, fast, system for version control*

Connectivity

- People are often Mobile & Disconnected:

❖ Possible to work offline

❖ …and share/merge later

❖ Connections may be unreliable

❖ Bandwidth may be limited.

- *Need system that allows disconnected, remote operation with periodic "resynch"*

Trust

- Development is decentralized or open source:

❖ Is this a good or bad man?

❖ Should we accept his contribution?

❖ Who decides?

❖ How do we decide?.

- *Need a safe, friendly, efficient way to incorporate outsider contributions.*

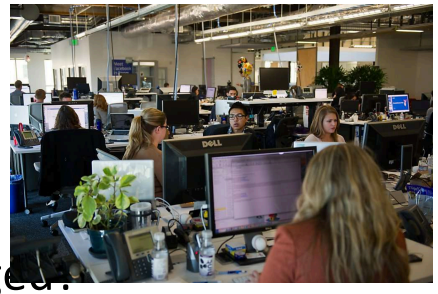
Integrity

- With multiple origins, multiple people, multiple versions…

❖ How do you know if a version is what it says it is?

❖ How do if nothing has really changed?

❖ How do you if something has changed for sure? And what changed?



❖ …and check quickly?

- *Need a way to quickly and reliably check if two versions are same or different.*

Transparency

- People work differently, lots of trial & error..

❖ How do I understand the changes you made?

❖ Can changes be reshaped, for clarity

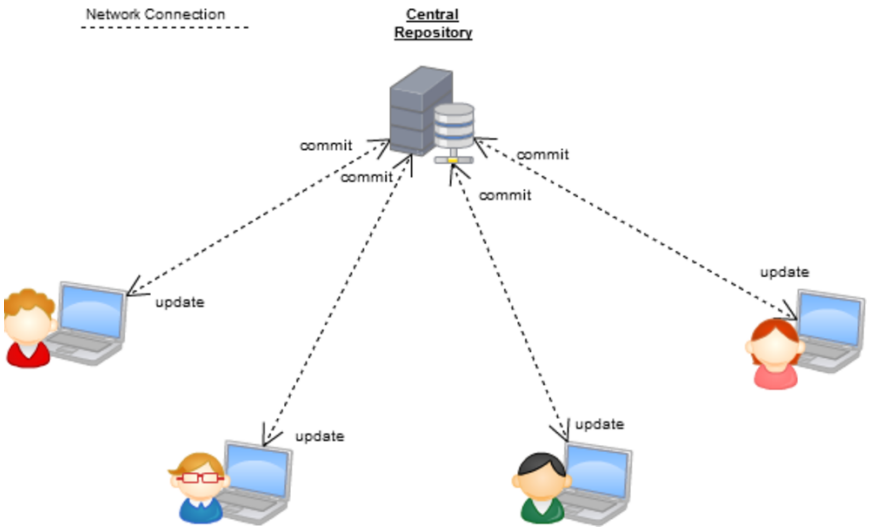
- *Need a way to reorder & regroup changes*

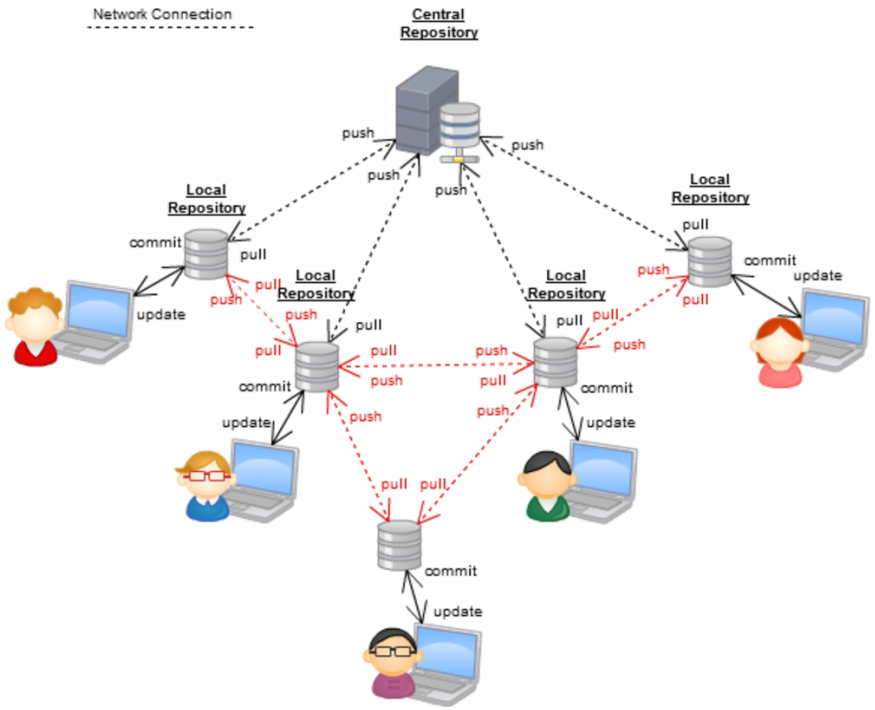
Types of Version Control Systems

- Make
- **Centralized VCS**
  - RCS, SCCS, ….
  - CVS, Subversion (SVN)
  - Clearcase, Perforce
- **Distributed VCS**
  - Git
  - Mercurial (Hg), Bazaar (Bzr)

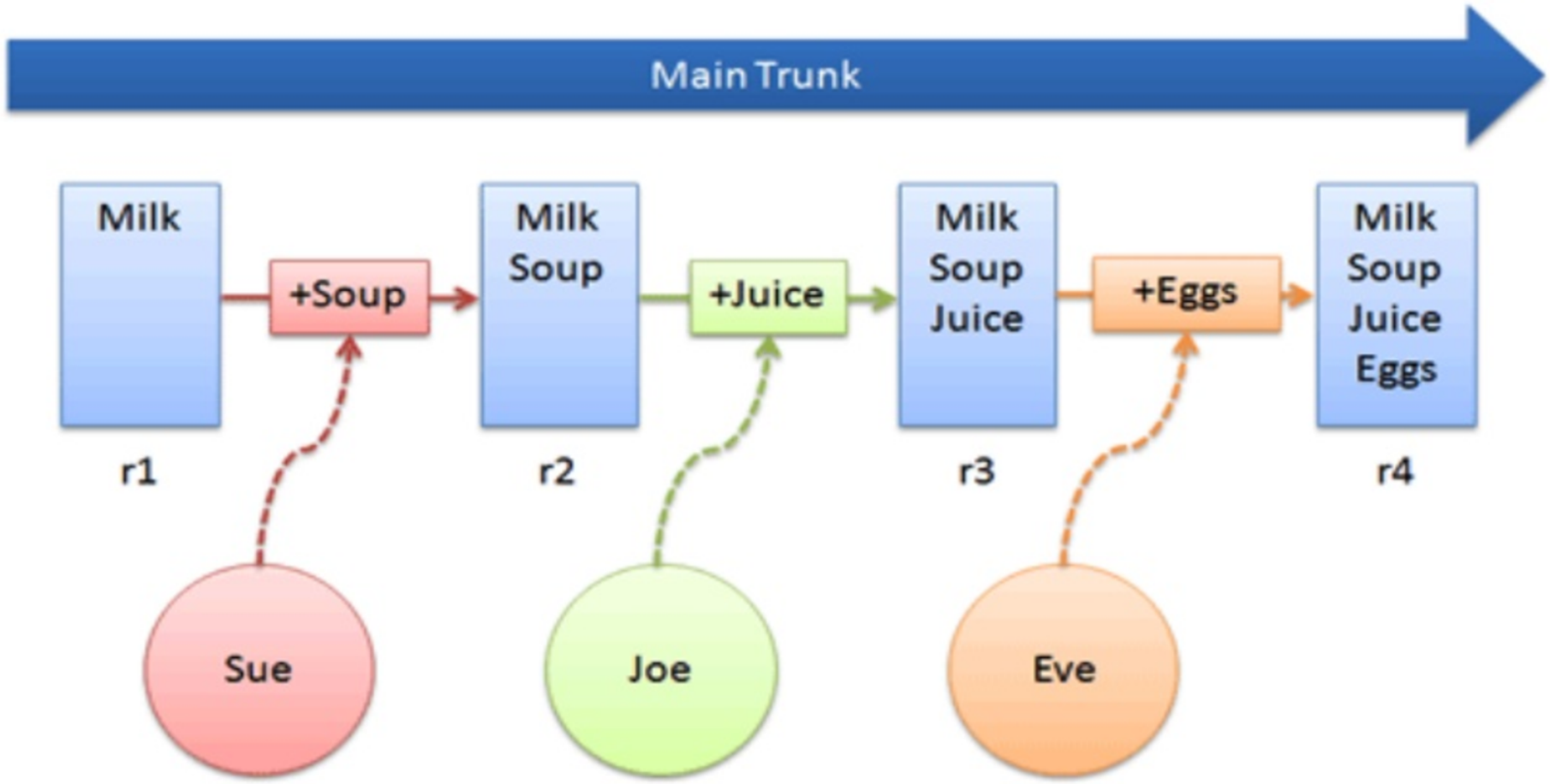
Centralized vs. Decentralized VCS

- Centralized
  - Internet,
  - Server up & running to commit
- Decentralized
  - No requirements

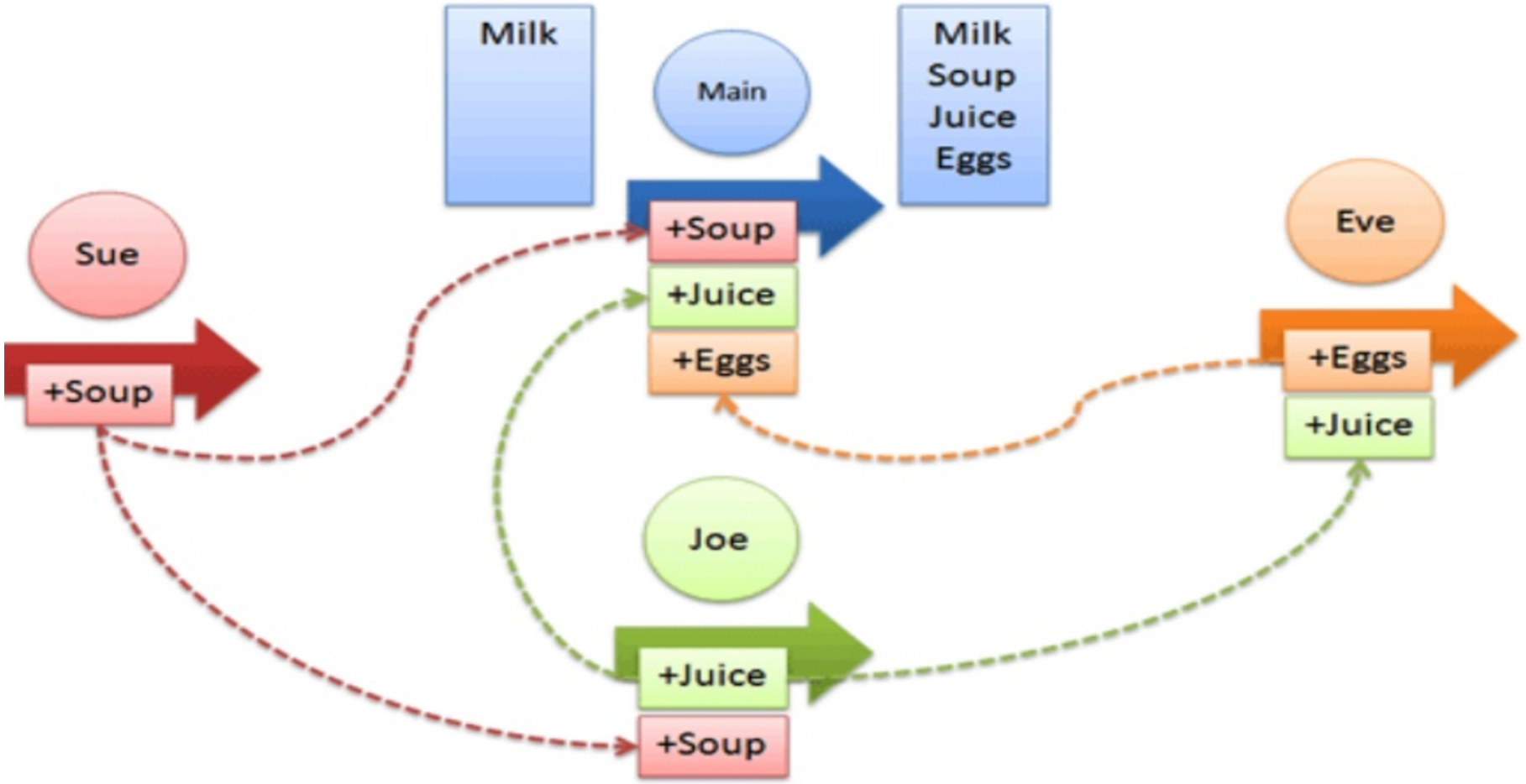




Centralized VCS



Decentralized VCS



This Week's Topic

- Course Rebaselining
- Version Control
- Intro to Git & Github
- Lab: Git

- A distributed VCS
- Started in 2005
- Created by Linus Torvald to aid in Linux kernel development

Vs.

- A web-based git repository hosting site
- Started in 2007, released in 2008
- Calls itself social coding
- As of March 2020: 40M+ users, 100M+ repositories

“I'm an egotistical bastard, and I name all my projects after myself. First Linux, now git[1].”

Linus Torvalds

“I have an ego the size of a small planet” - Linus

[1] git (n): British slang for a stupid or unpleasant person

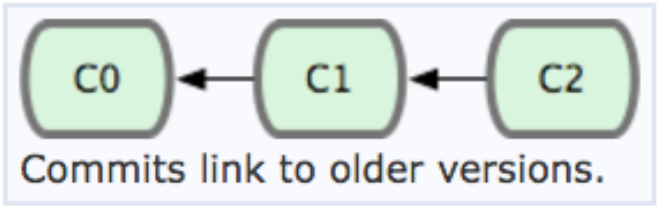
Key Concepts: Snapshots

- The way git keeps track of your code history
- Essentially records what all your files look like at a given point in time
- You decide when to take a snapshot, and of what files
- Have the ability to go back to visit any snapshot
- Your snapshots from later on will stay around, too

Key Concepts: Commit

- The act of creating a snapshot
- Essentially, a project is made up of a bunch of commits

- Commits contain three pieces of information:
  1. Information about how the files changed from previously
  2. A reference to the commit that came before it - called the “parent commit”
  3. A hash code name. Something like: f2d2ec5069fc6776c80b3ad6b7cbde3cade4e



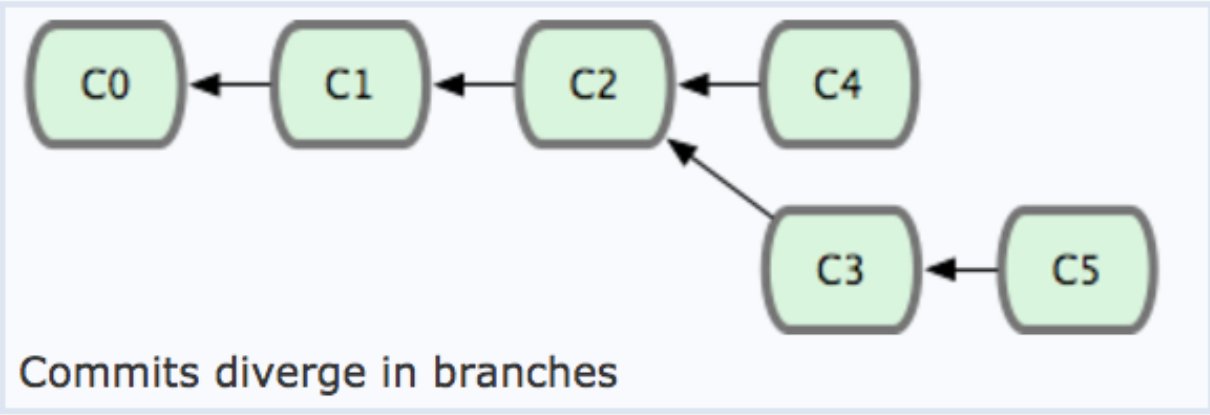
Key Concepts: Repositories

- Often shortened to 'repo'
  - A collection of all the files and the history of those files
  - Consists of all your commits
  - Can live on a local machine or on a remote server (GitHub!)

- The act of copying a repository from a remote server is called cloning
  - Cloning from a remote server allows teams to work together
- The process of downloading commits that don't exist on your machine from a remote repository is called pulling changes
- The process of adding your local changes to the remote repository is called pushing changes

Key Concepts: Branches

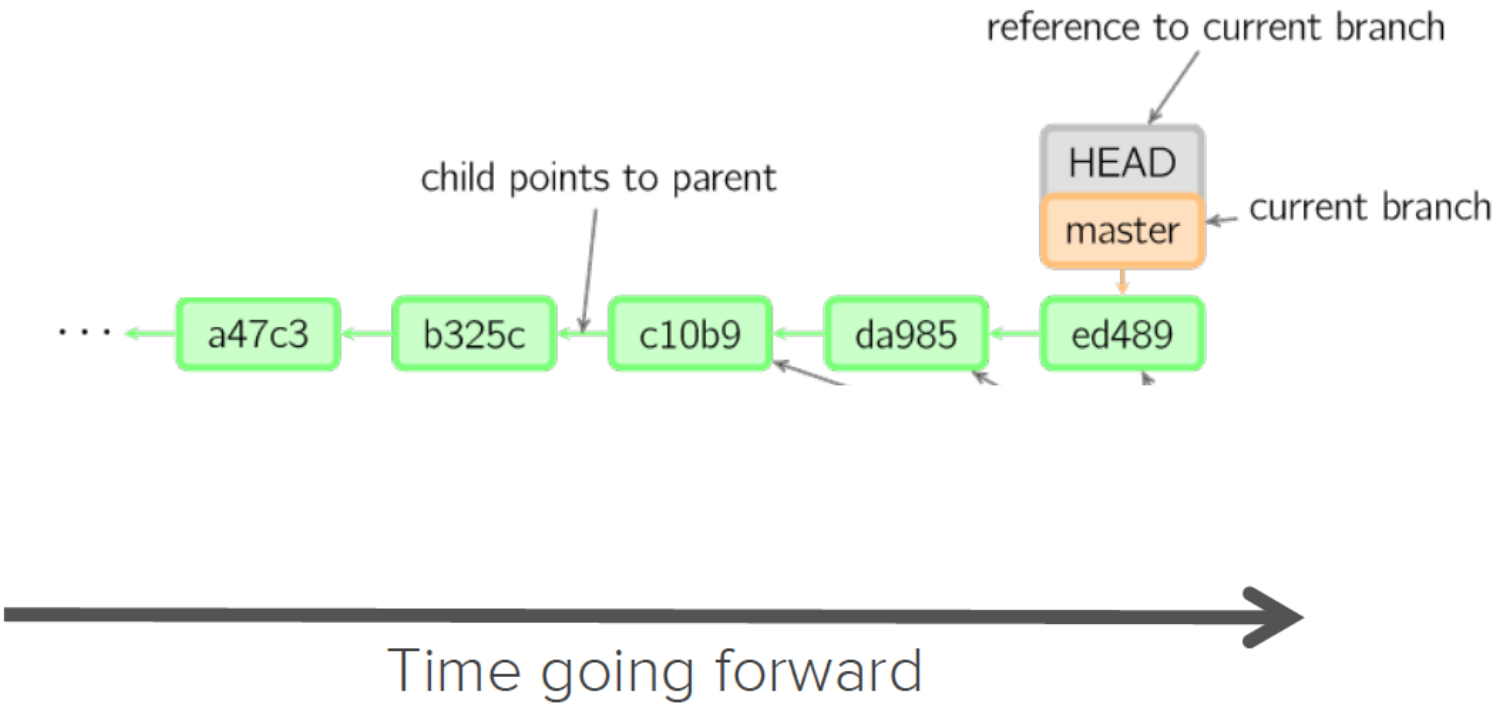
- Allows you to work in parallel
- All commits in git live on some branch
- But there can be many, many branches
- The main branch in a project is called the master branch



So, what does a typical project look like?

- A bunch of commits linked together that live on some branch, contained in a repository
- Following images taken and modified from:
  - http://marklodato.github.io/visual-gitguide/index-en.html
  - Also a good tutorial!

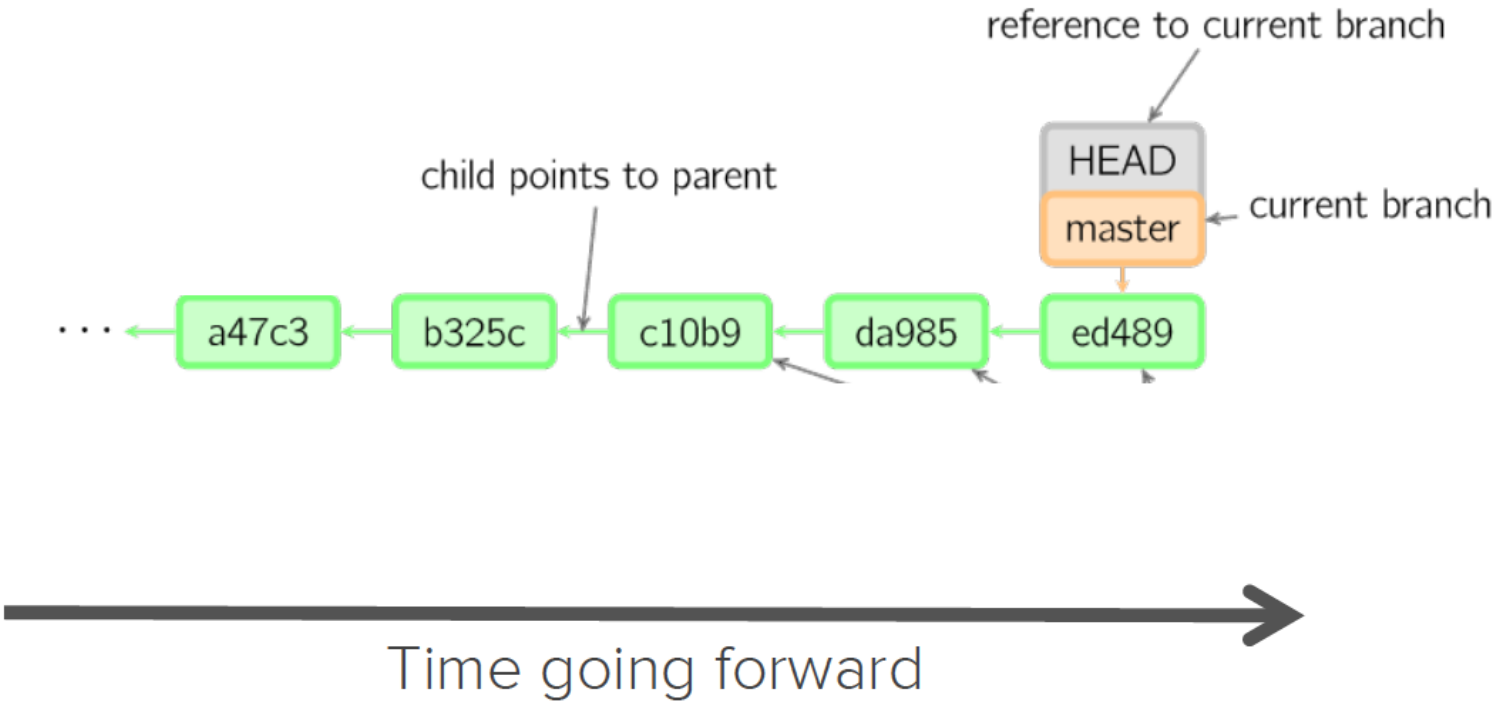
So, what does a typical project look like?



So, what is MASTER?

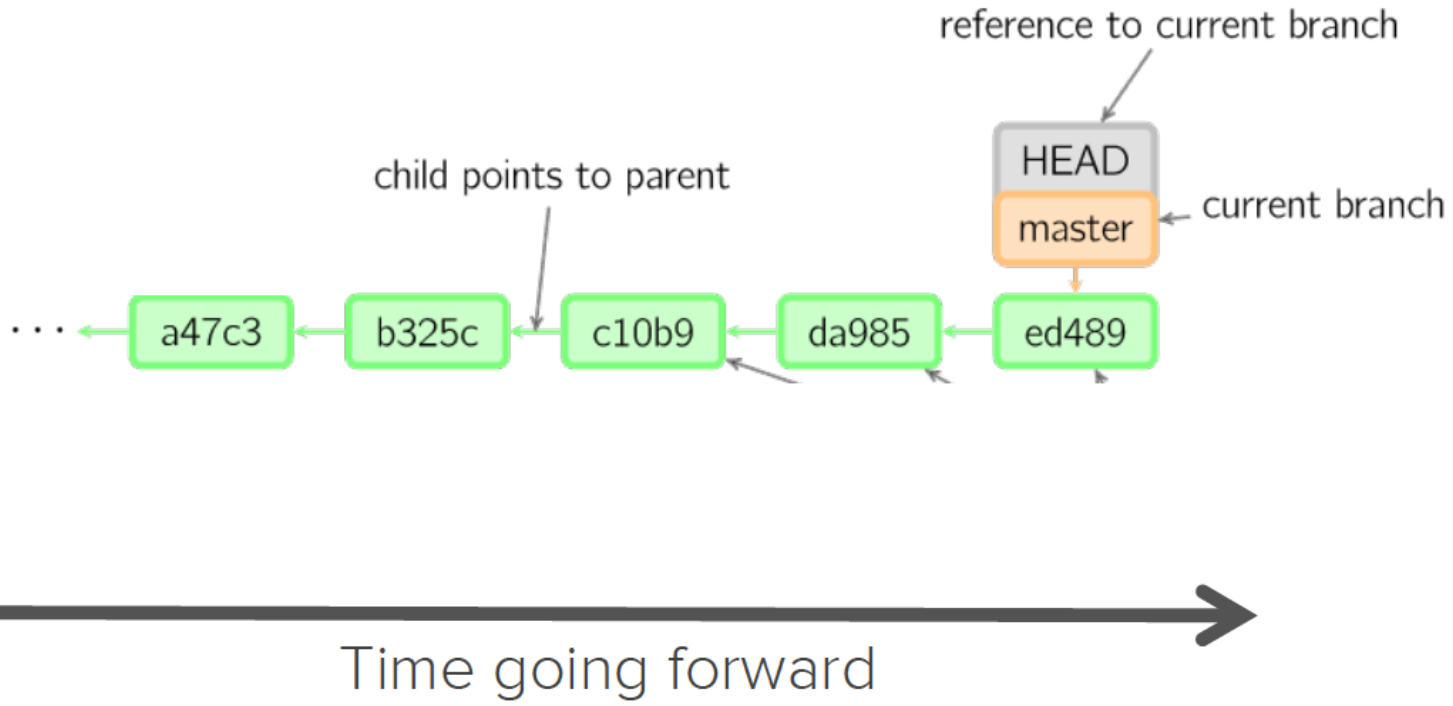
The main branch in your project

Doesn't have to be called master, but almost always is!



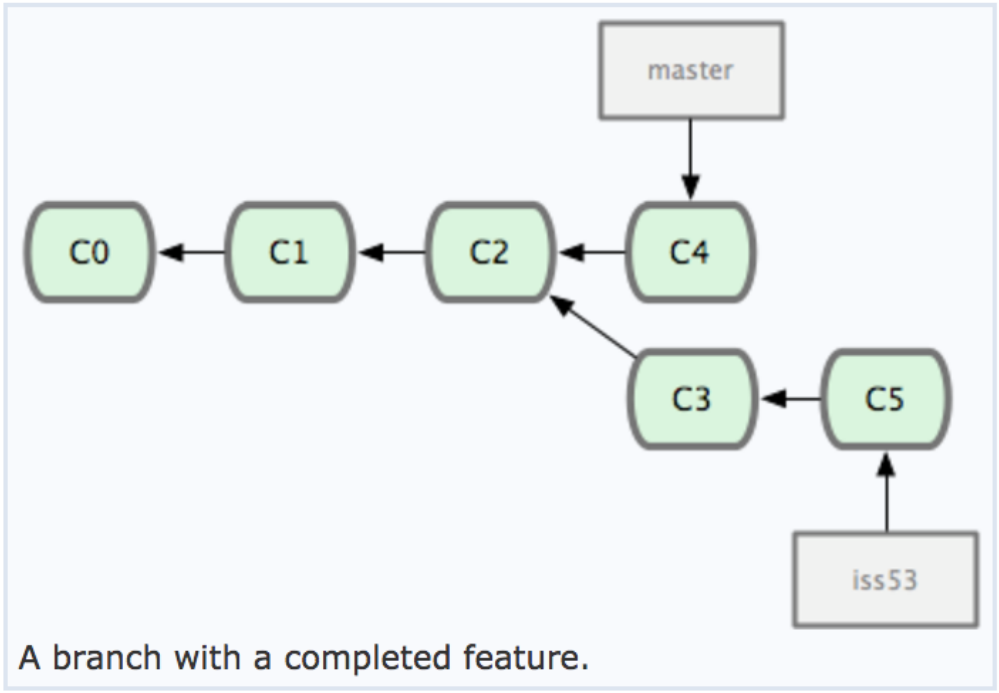
So, what is HEAD?

A reference to the most recent
commit of the branch checked out



head: refers to any one of the named heads in the repository
HEAD: refers exclusively to the currently active head

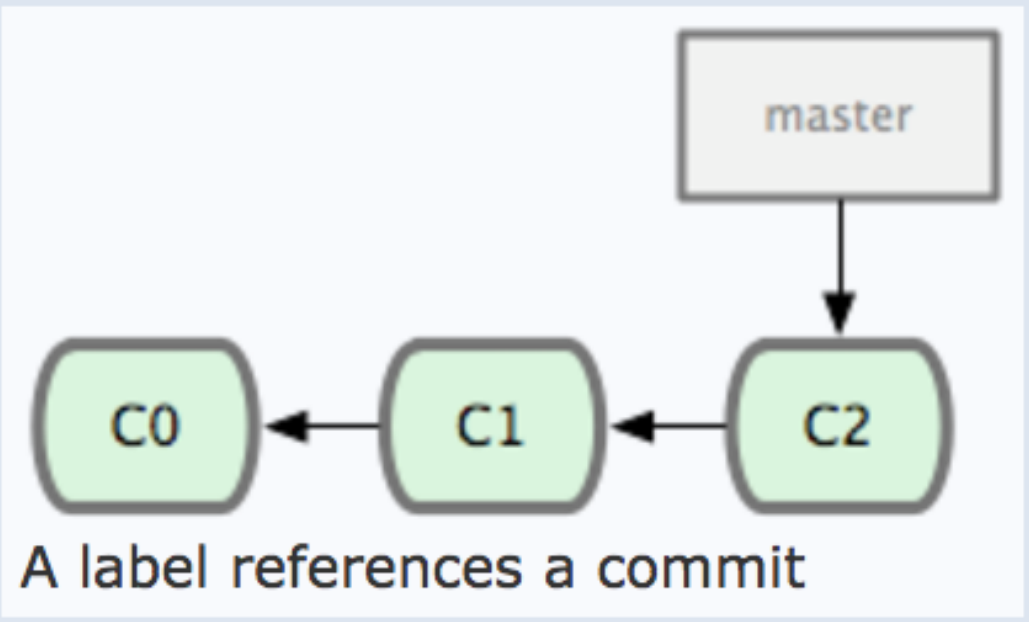
Key Concepts: Branching off of the master branch



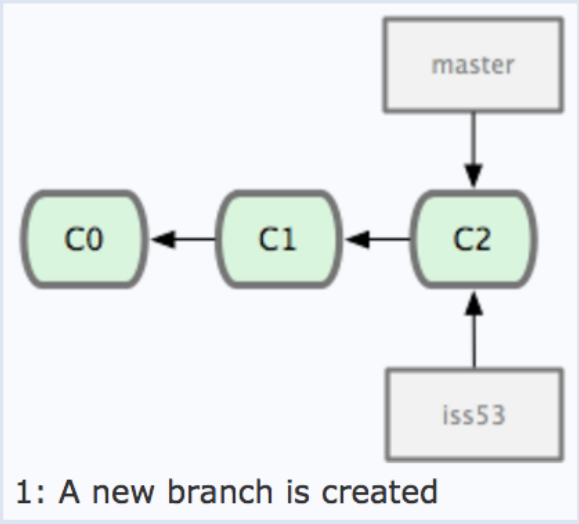
A branch with a completed feature.

Time going forward

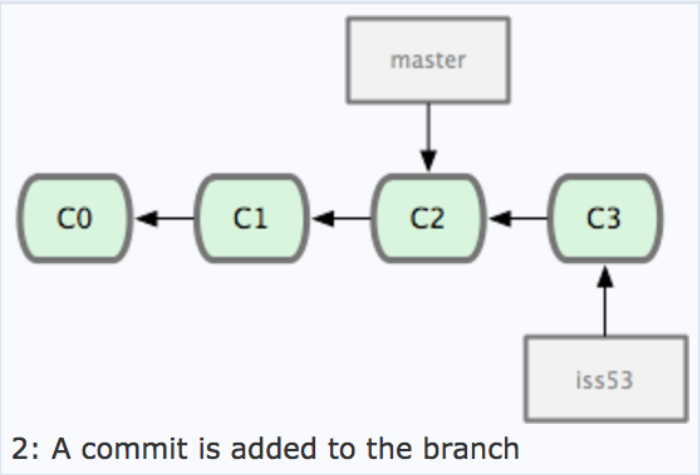
Unpacking Branching: current state



Unpacking Branching: creating a branch, switching to it



1: A new branch is created



2: A commit is added to the branch

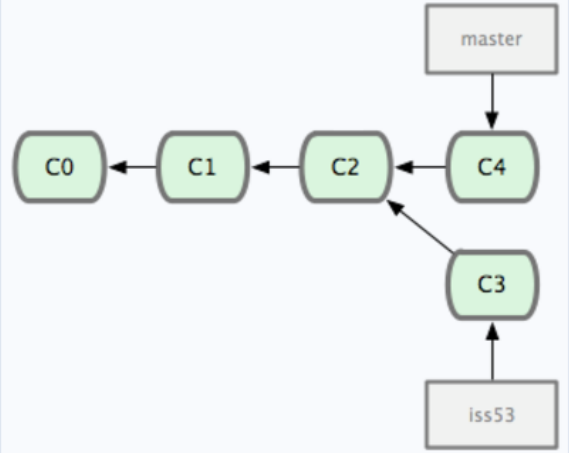
Making a branch, and committing it in:

```
git branch iss53        # create the branch (see figure 1)
git checkout iss53      # switch to it

...                     # edit a file

git commit -a           # commit the change in the new branch
```

Unpacking Branching: switching back to MASTER



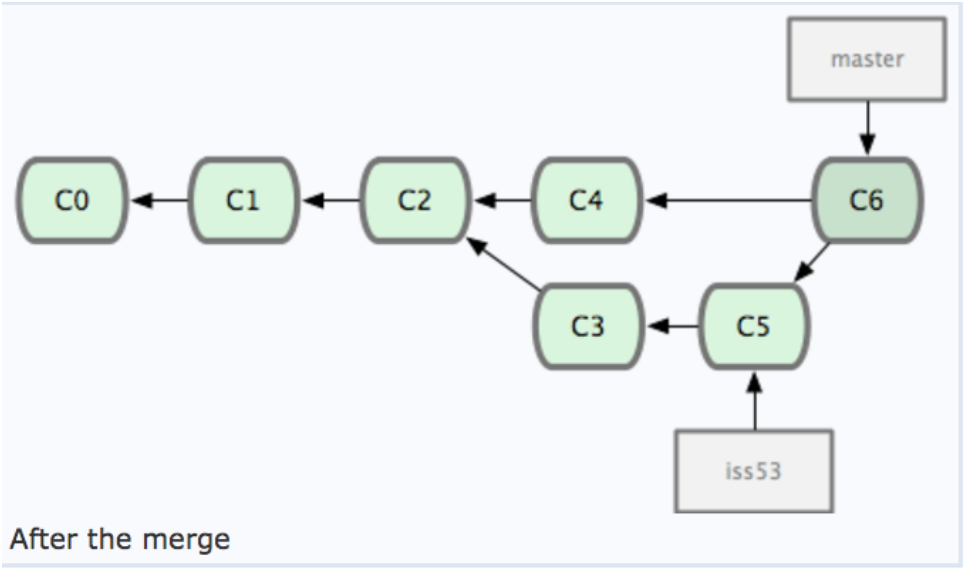
Result after making a new commit in the master branch

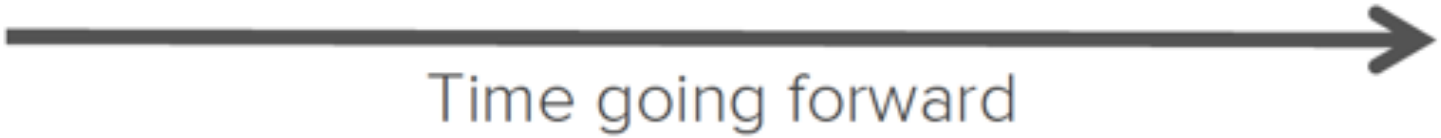
Switching back to master:

```
git checkout master
```

Key Concepts: Merging

Once you're done with your feature,
you merge it back into master



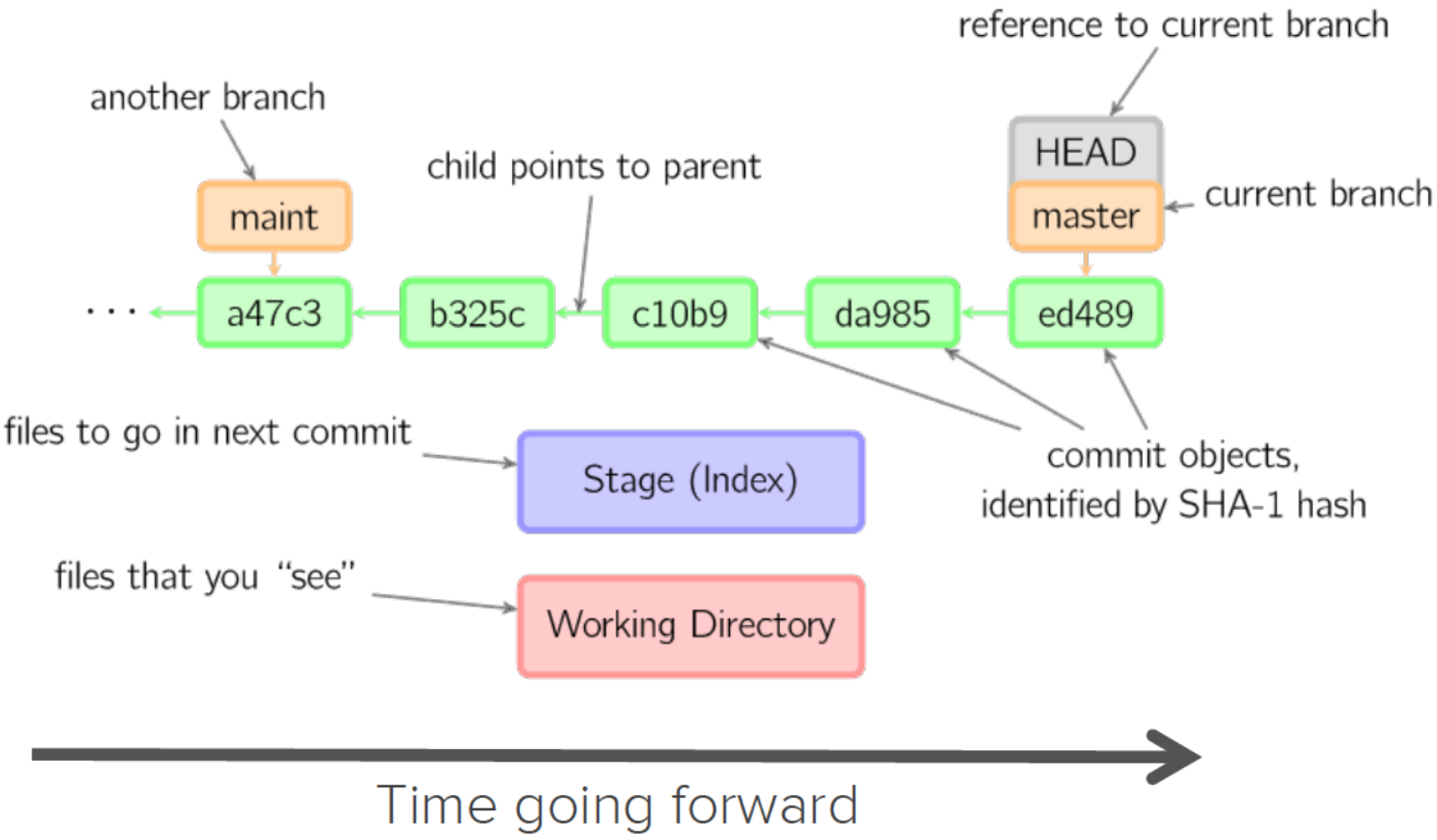


Key Concepts: How do you make a commit anyway?

- There are a lot of 'states' and 'places' a file can be
- Local on your computer: the 'working directory'
- When a file is ready to be put in a commit you add it onto the 'index' or '**staging**'
- The process:
  - Make some changes to a file
  - Use the 'git add' command to put the file onto the staging environment
  - Use the command to create a new commit'

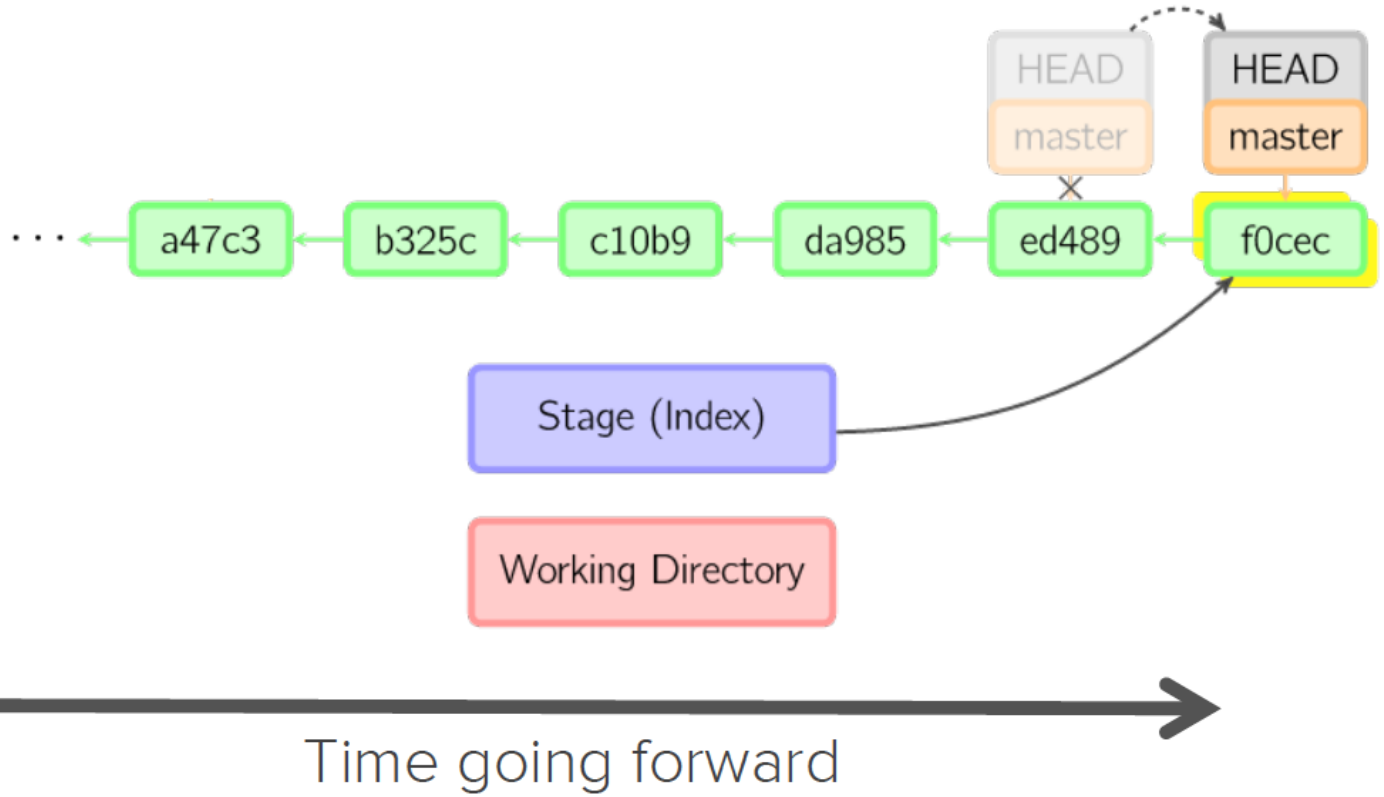
Key Concepts: How do you make a commit anyway?

git add



Key Concepts: How do you make a commit anyway?

git commit



Other concepts

- Conflict
- Backport
- Cherrypick

How people build software · G

Premkumar

GitHub, Inc. [US] | https://github.com

Apps | Yahoo! | Politics | Scholar | Apple | News | ACM/DL | Apple | literature | parnin course | bayesian | Inbox (289) | distractions | FB | (தமிழ்) | R info | smartsite

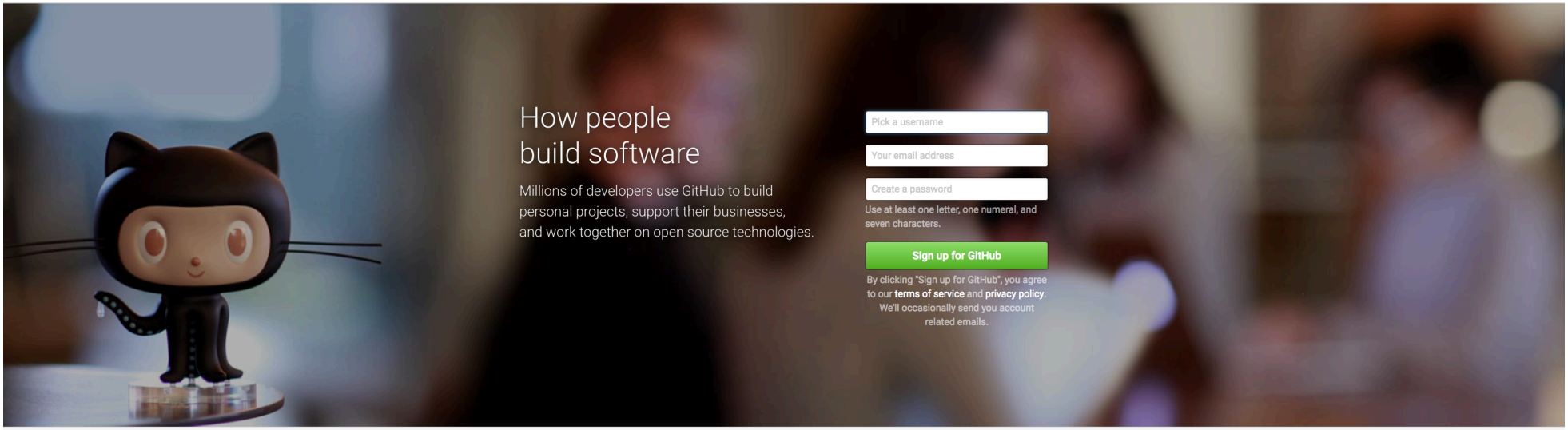
Scholar
http://scholar.google.com/

Personal Open source Business Explore

Pricing Blog Support

Sign in Sign up

Search GitHub



How people build software

Millions of developers use GitHub to build personal projects, support their businesses, and work together on open source technologies.

Pick a username

Your email address

Create a password

Use at least one letter, one numeral, and seven characters.

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our terms of service and privacy policy. We'll occasionally send you account related emails.

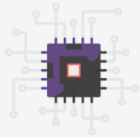
Welcome home, developers

GitHub fosters a fast, flexible, and collaborative development process that lets you work on your own or with others.

For everything you build
Host and manage your code on GitHub. You can keep your work private or share it with the world.

A better way to work
From hobbyists to professionals, GitHub helps developers simplify the way they build software.

Millions of projects
GitHub is home to millions of open source projects. Try one out or get inspired to create your own.



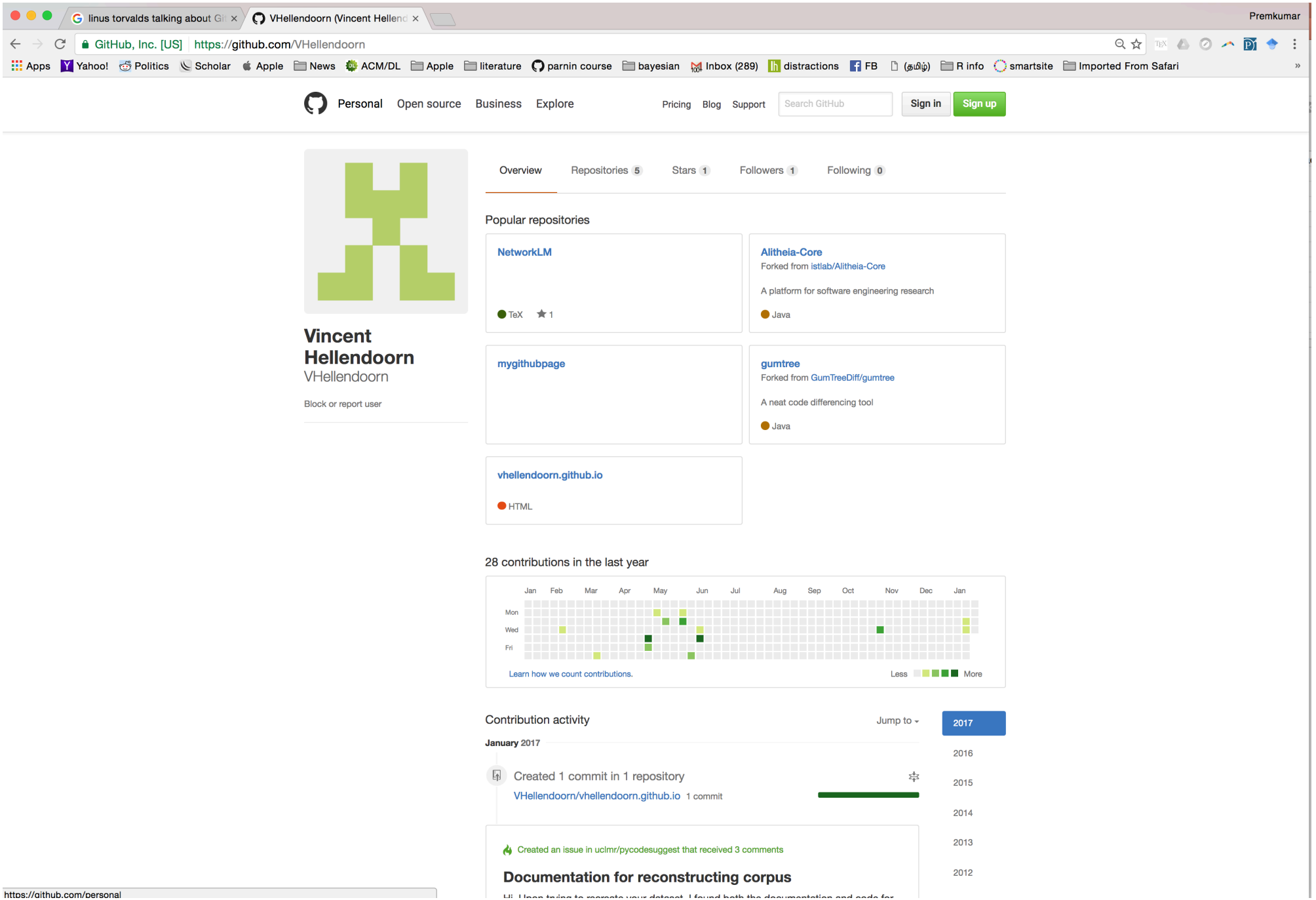
One platform, from start to finish
With hundreds of integrations, GitHub is flexible enough to be at the center of your development process.

Who uses GitHub?

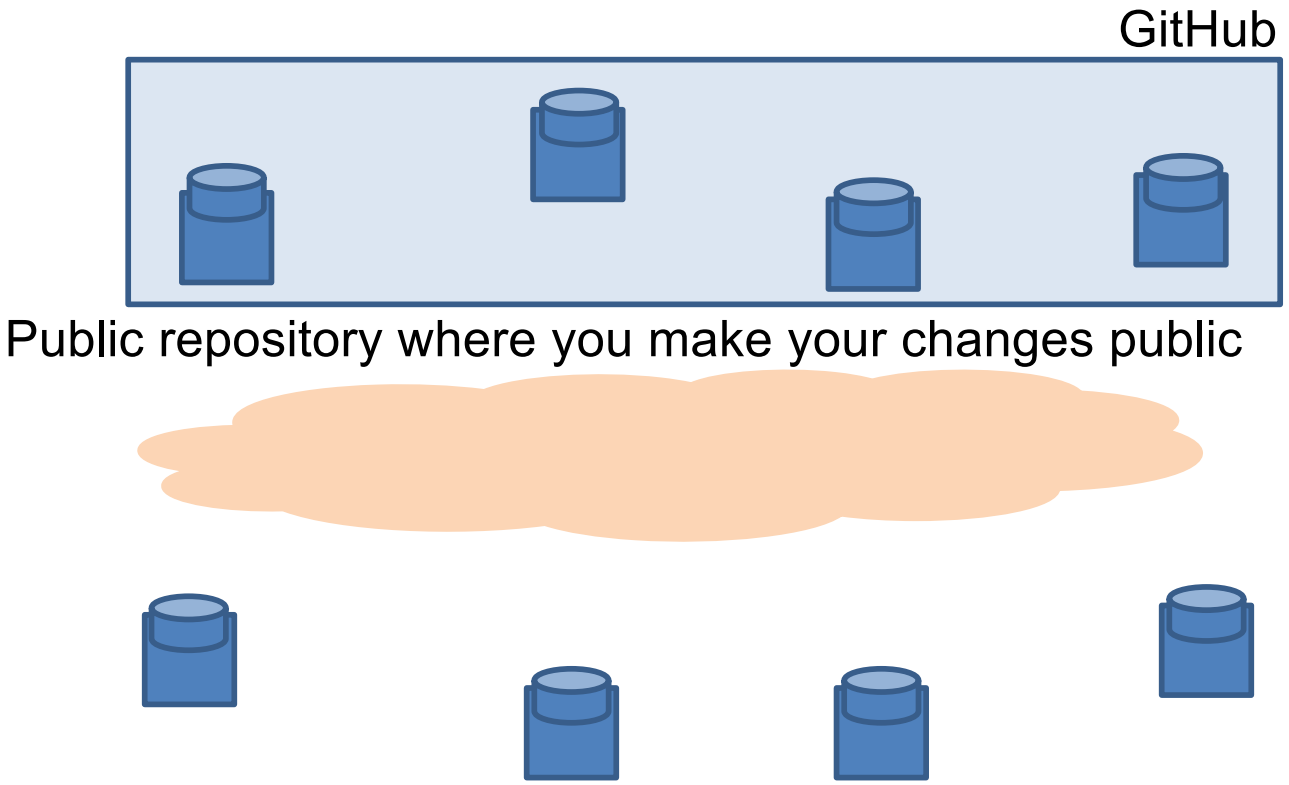
Individuals ›
Use GitHub to create a personal project, whether you want to experiment with a new programming language or host your life's work.

Communities ›
GitHub hosts one of the largest collections of open source software. Create, manage, and work on some of today's most influential technologies.

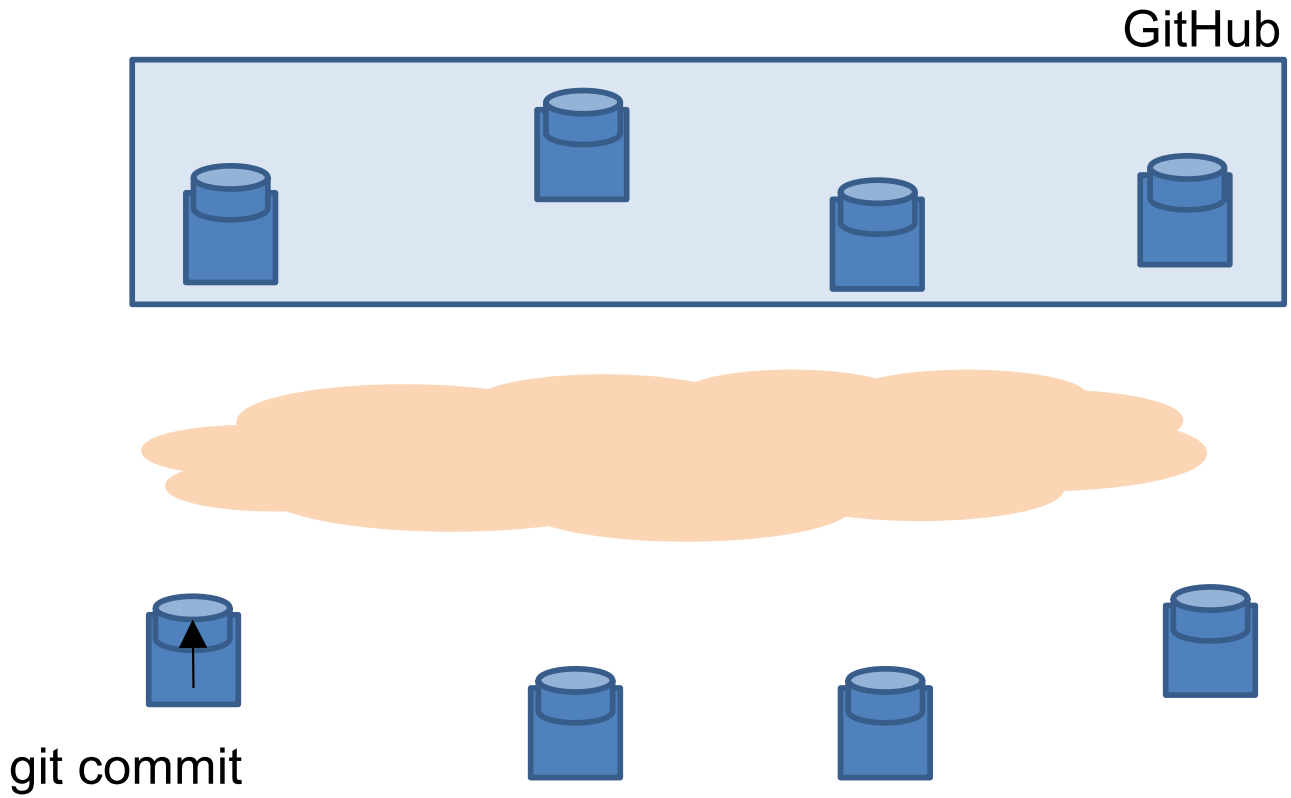
Businesses ›
Businesses of all sizes use GitHub to support their development process and securely build software.


linus torvalds talking about Gi
VHellendoorn (Vincent Hellend
Premkumar
GitHub, Inc. [US] https://github.com/VHellendoorn
Apps Yahoo! Politics Scholar Apple News ACM/DL Apple literature parnin course bayesian Inbox (289) distractions FB (தமிழ்) R info smartsite Imported From Safari
Personal Open source Business Explore
Pricing Blog Support
Search GitHub
Sign in
Sign up
Vincent Hellendoorn
VHellendoorn
Block or report user
Overview
Repositories 5
Stars 1
Followers 1
Following 0
Popular repositories
NetworkLM
TeX 1
Alitheia-Core
Forked from istlab/Alitheia-Core
A platform for software engineering research
Java
mygithubpage
gumtree
Forked from GumTreeDiff/gumtree
A neat code differencing tool
Java
vhellendoorn.github.io
HTML
28 contributions in the last year
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan
Mon
Wed
Fri
Learn how we count contributions.
Less More
Contribution activity
Jump to
2017
2016
2015
2014
2013
2012
January 2017
Created 1 commit in 1 repository
VHellendoorn/vhellendoorn.github.io 1 commit
Created an issue in uclmr/pycodesuggest that received 3 comments
Documentation for reconstructing corpus
https://github.com/personal

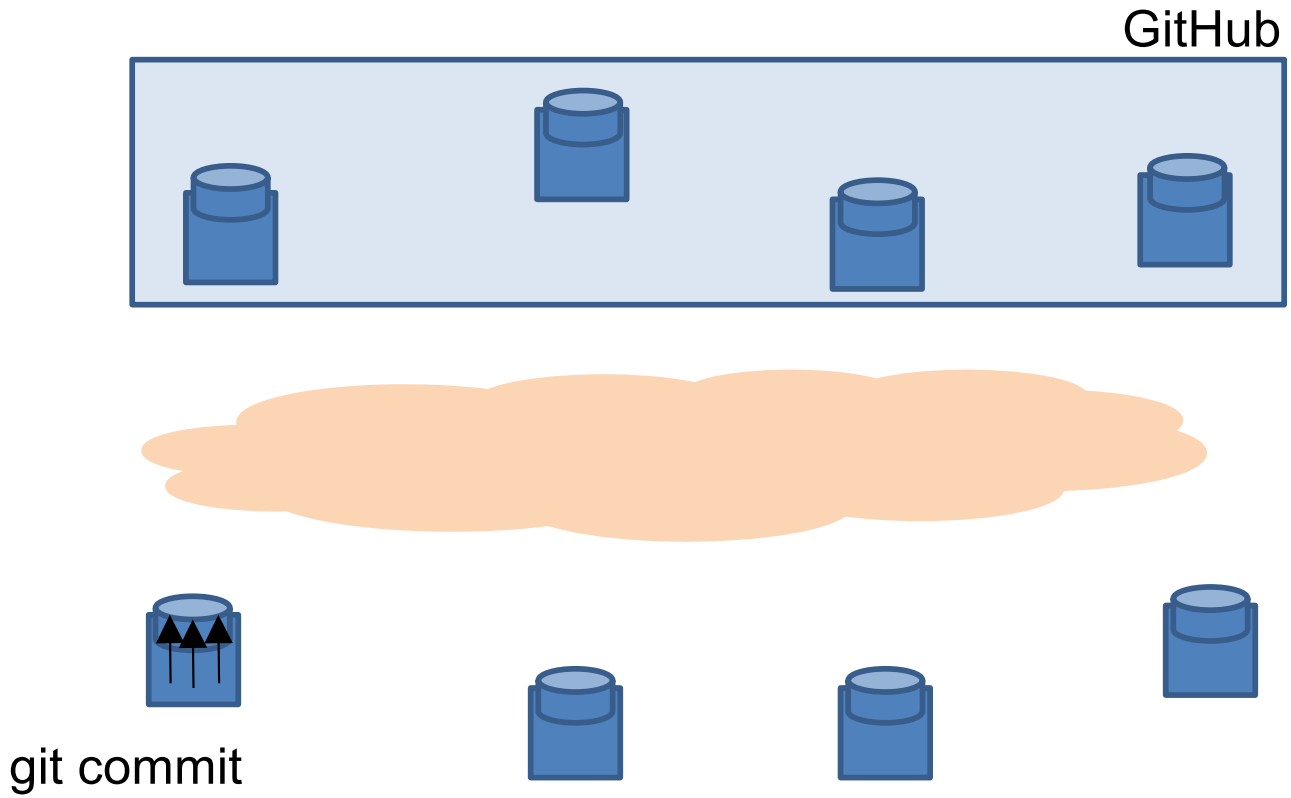
GitHub typical workflow



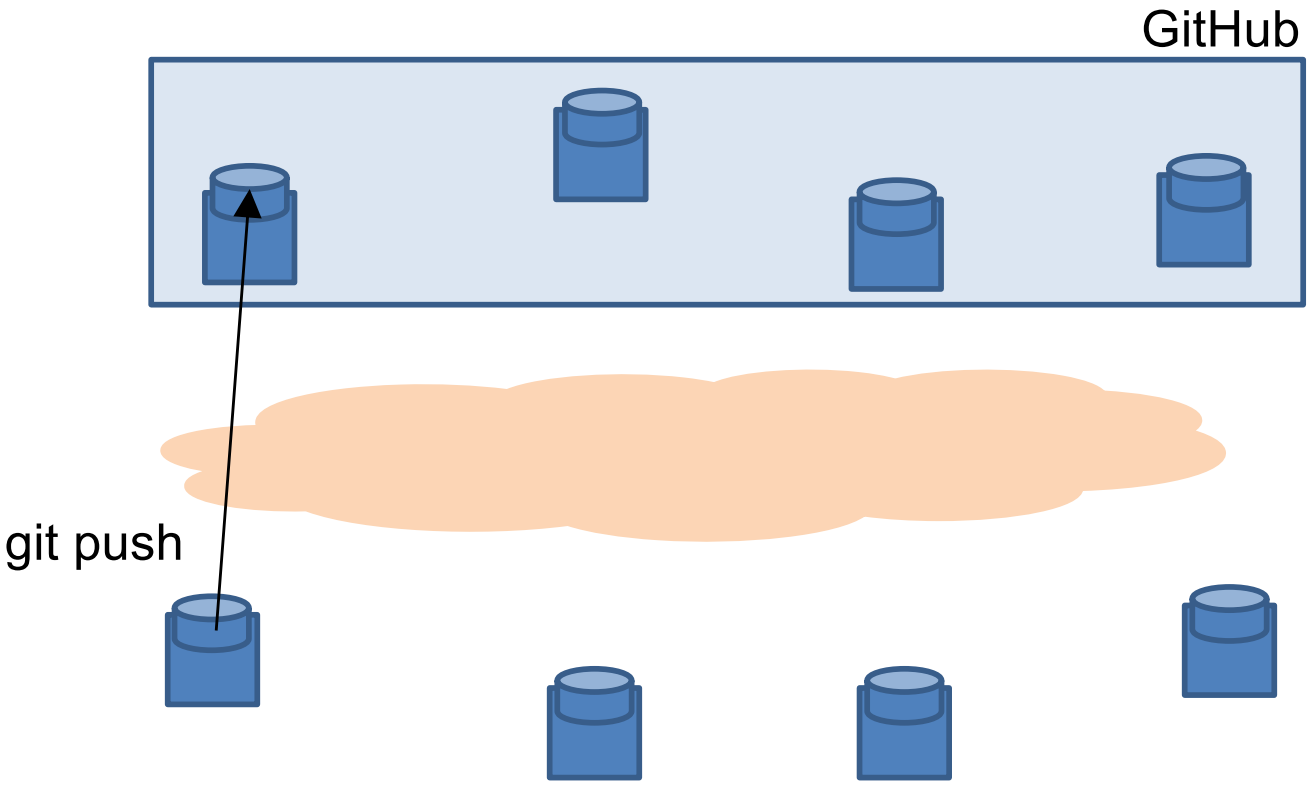
GitHub typical workflow



GitHub typical workflow

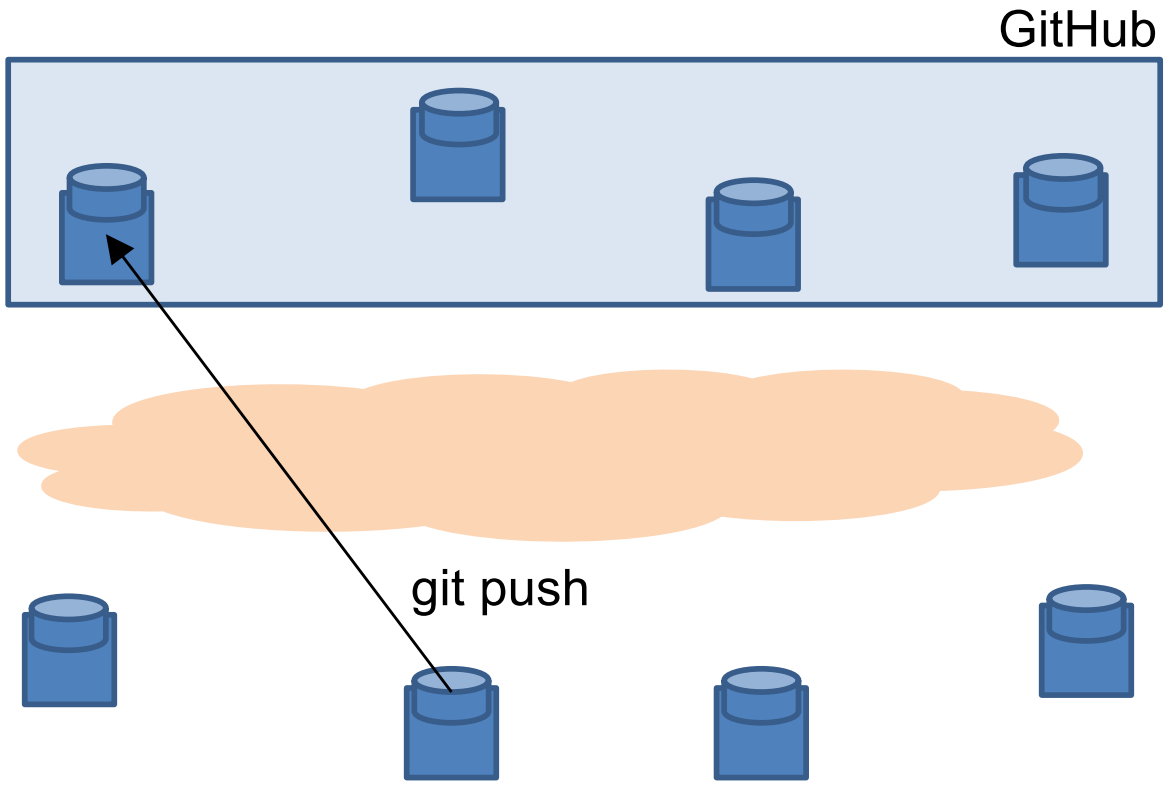


GitHub typical workflow



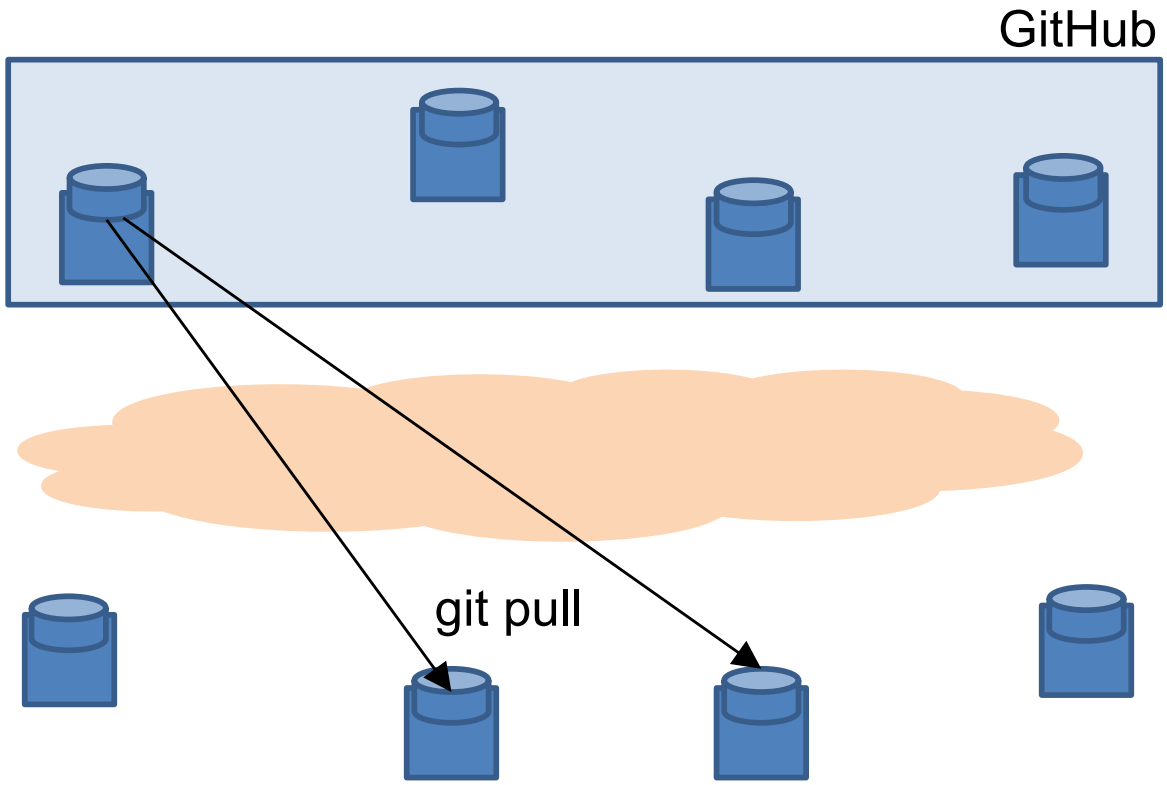
push your local changes into a remote repository.

GitHub typical workflow



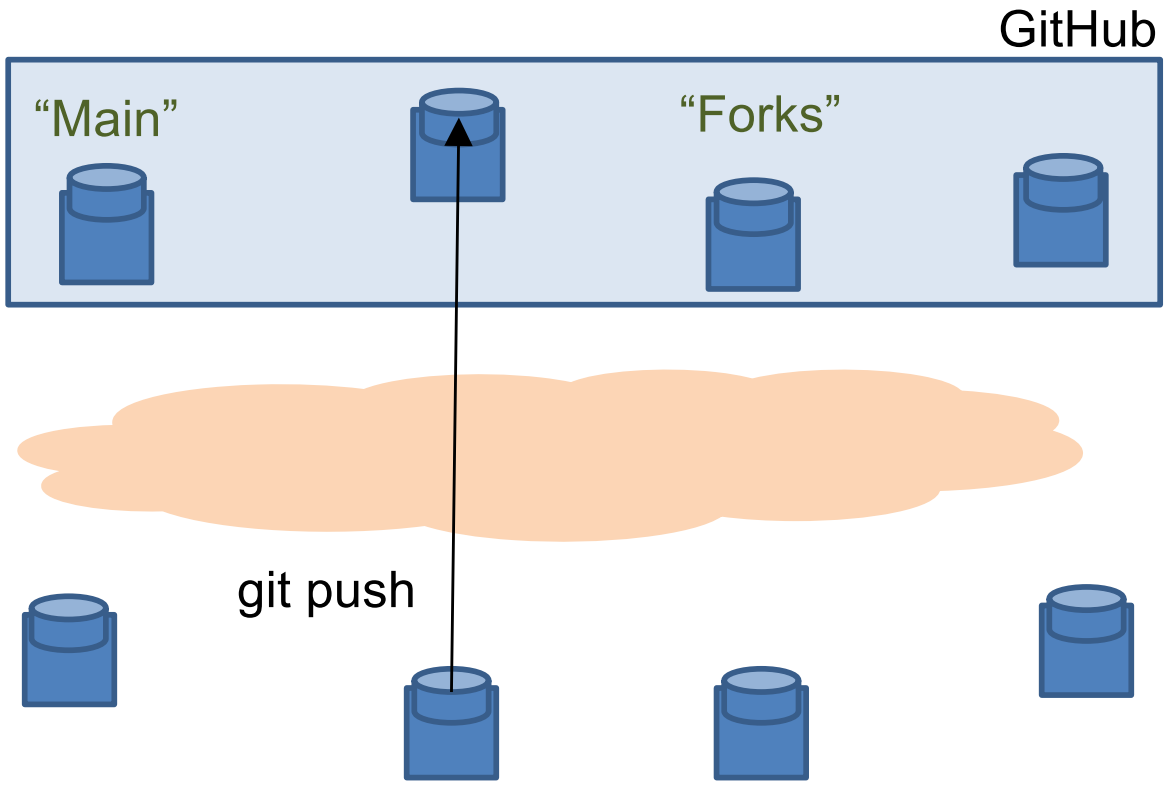
Collaborators can push too if they have access rights.

GitHub typical workflow



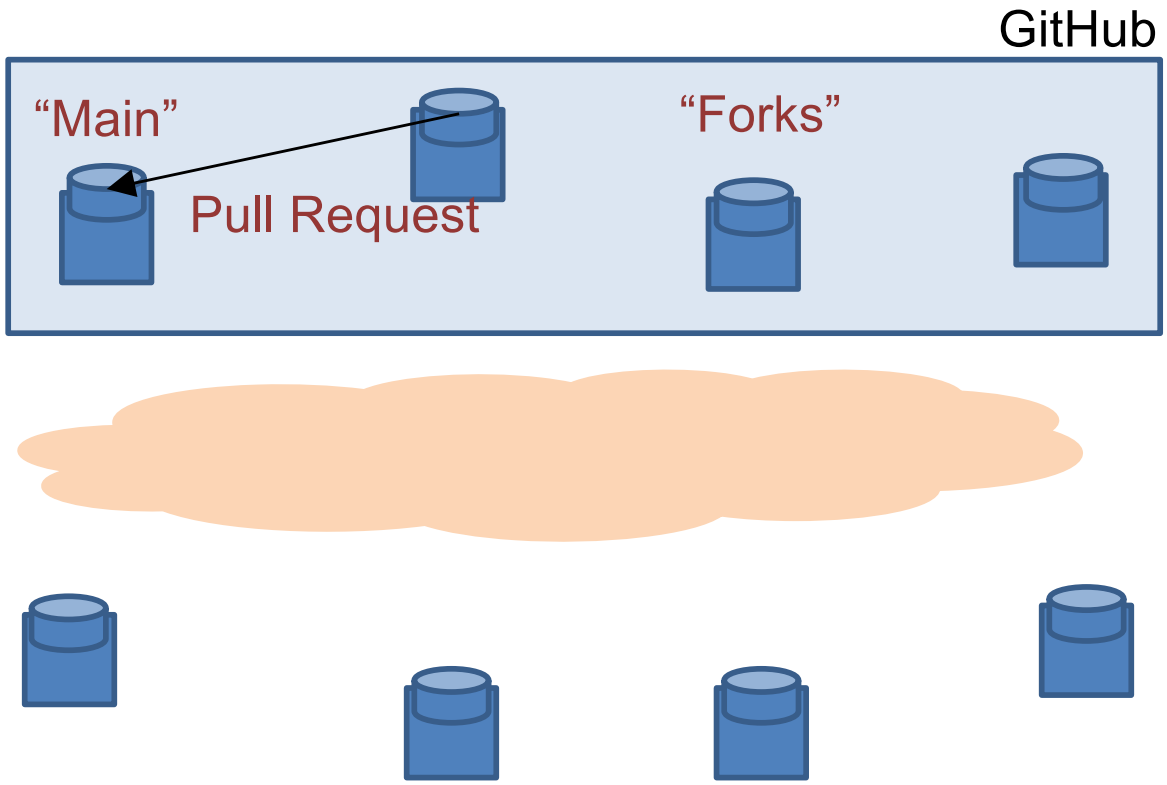
Without access rights, “don’t call us, we’ll call you” (*pull* from trusted sources) … But again requires host names / IP addresses.

GitHub typical workflow



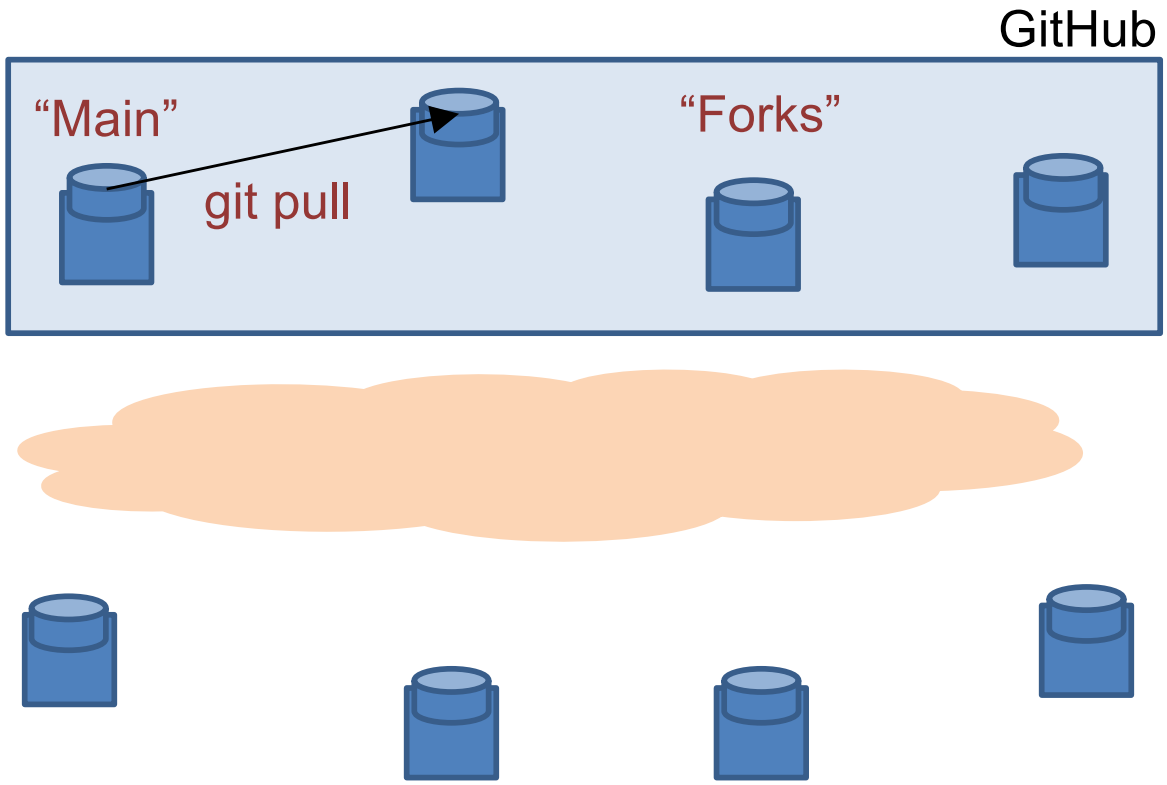
Instead, people maintain public remote “forks” of “main” repository on GitHub and push local changes.

GitHub typical workflow



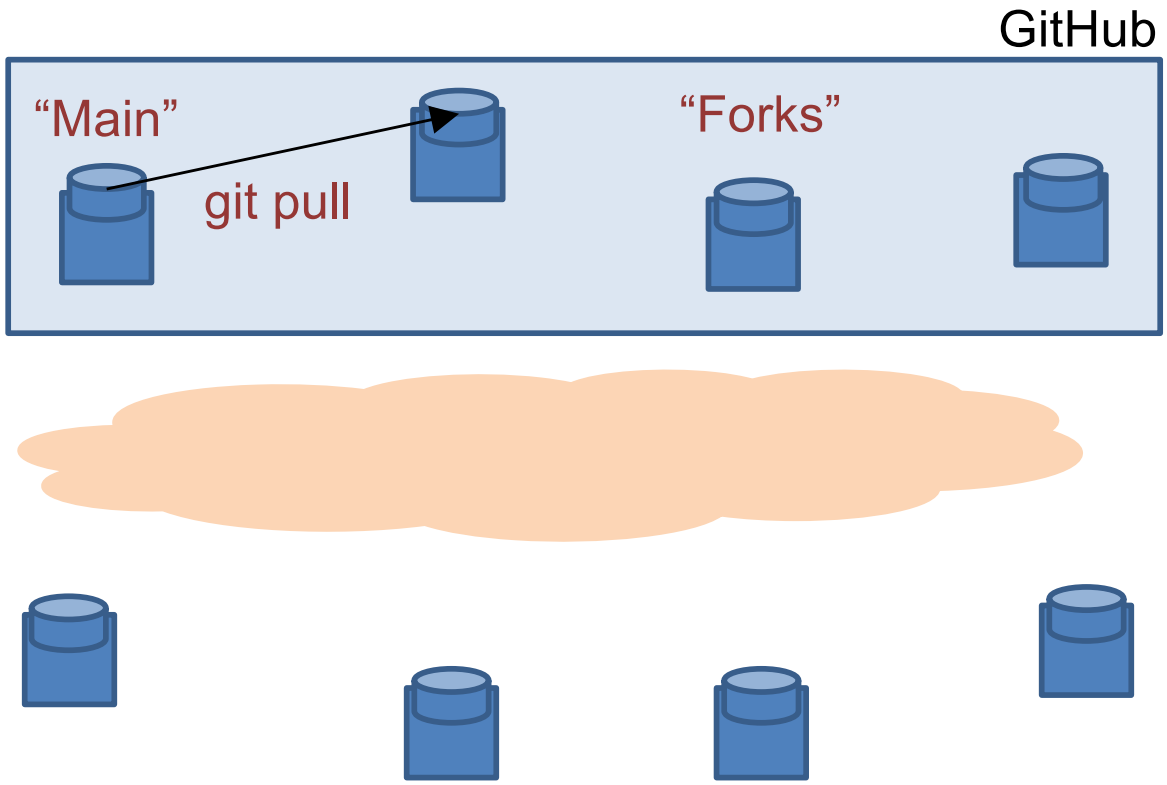
Availability of new changes is signaled via ”Pull Request”.

GitHub typical workflow



Changes are pulled into main if PR accepted.

GitHub typical workflow



Changes are pulled into main if PR accepted.

Additional Resources

- Official git site and tutorial:

  https://git-scm.com/
- GitHub guides:

  https://guides.github.com/
- Command cheatsheet:

  https://training.github.com/kit/downloads/github-git-cheat-sheet.pdf
- Interactive git tutorial:

  https://try.github.io/levels/1/challenges/1
- Visual/interactive cheatsheet:

  http://ndpsoftware.com/git-cheatsheet.html

This Week's Topic

- Course Rebaselining
- Version Control
- Intro to Git & Github
- Lab: Git

Lab: Git

- HW3
  - Individual assignment
  - Solve the first four levels in: http://pcottle.github.io/learnGitBranching/