CSCI 4830-009

Open Source Development

Spring, 2017

Ned McClain

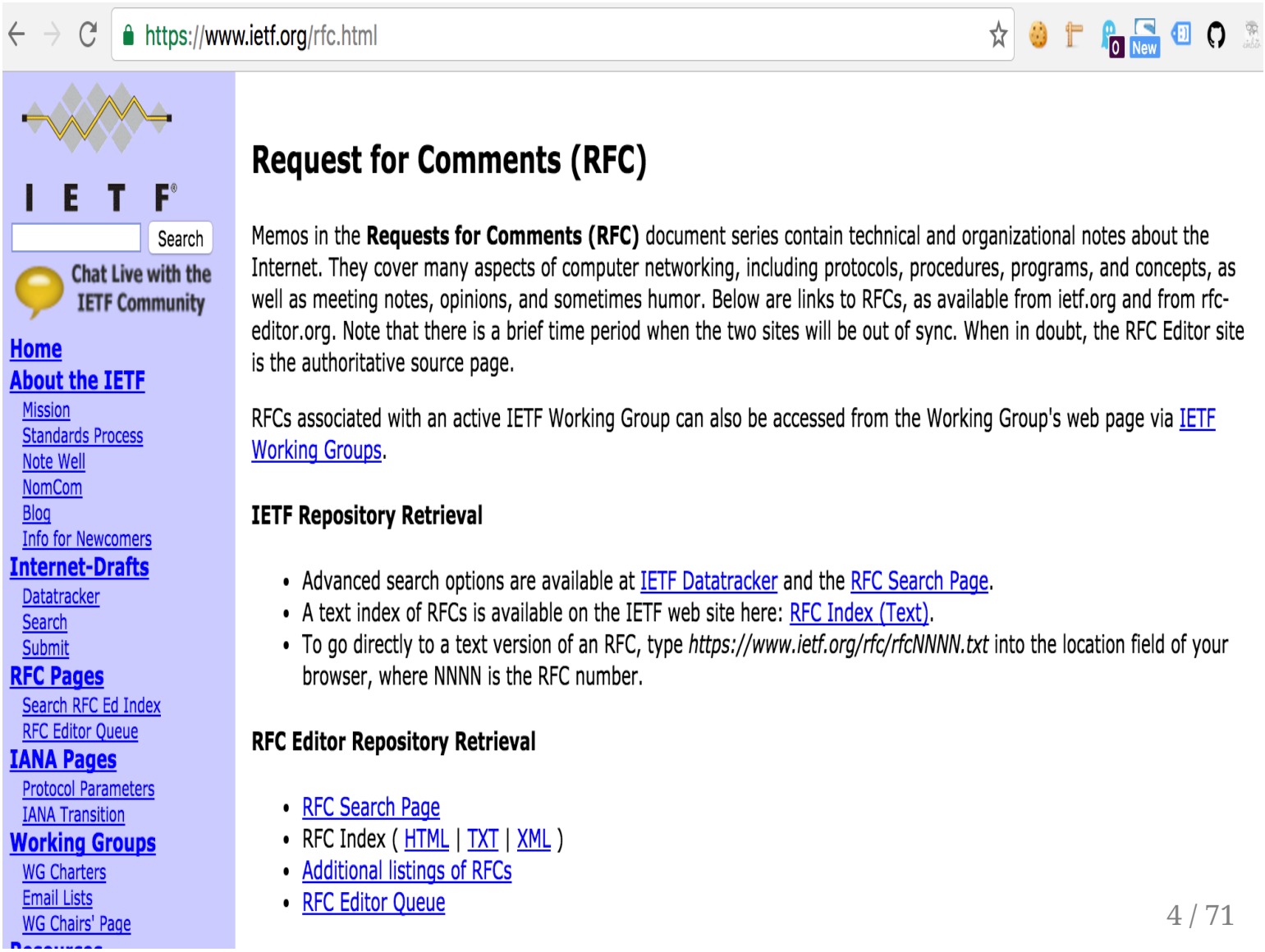
Welcome!

Today:

Hello Git!


RICHARD STALLMAN! YOUR VIRAL OPEN SOURCE LICENSES HAVE GROWN TOO POWERFUL.
THE GPL MUST BE STOPPED.
AT THE SOURCE.
YOU.
ZZZZ
HAH! MICROSOFT LACKEYS! SO IT HAS COME TO THIS.
SHING!
SHING!
A NIGHT OF BLOOD I'VE LONG AWAITED. BUT BE THIS MY DEATH OR YOURS, FREE SOFTWARE WILL CARRY ON! FOR A GNU DAWN! FOR FREEDOM!
...HEY, WHERE ARE YOU GOING?


MAN, YOU'RE RIGHT, THAT NEVER GETS OLD.
LET'S DO ERIC S. RAYMOND NEXT.
OR LINUS TORVALDS. I HEAR HE SLEEPS WITH NUNCHUCKS.
3 / 71


https://www.ietf.org/rfc.html
I E T F
Search
Chat Live with the IETF Community
Home
About the IETF
Mission
Standards Process
Note Well
NomCom
Blog
Info for Newcomers
Internet-Drafts
Datatracker
Search
Submit
RFC Pages
Search RFC Ed Index
RFC Editor Queue
IANA Pages
Protocol Parameters
IANA Transition
Working Groups
WG Charters
Email Lists
WG Chairs' Page
Request for Comments (RFC)
Memos in the Requests for Comments (RFC) document series contain technical and organizational notes about the Internet. They cover many aspects of computer networking, including protocols, procedures, programs, and concepts, as well as meeting notes, opinions, and sometimes humor. Below are links to RFCs, as available from ietf.org and from rfc-editor.org. Note that there is a brief time period when the two sites will be out of sync. When in doubt, the RFC Editor site is the authoritative source page.
RFCs associated with an active IETF Working Group can also be accessed from the Working Group's web page via IETF Working Groups.
IETF Repository Retrieval
Advanced search options are available at IETF Datatracker and the RFC Search Page.
A text index of RFCs is available on the IETF web site here: RFC Index (Text).
To go directly to a text version of an RFC, type https://www.ietf.org/rfc/rfcNNNN.txt into the location field of your browser, where NNNN is the RFC number.
RFC Editor Repository Retrieval
RFC Search Page
RFC Index (HTML | TXT | XML)
Additional listings of RFCs
RFC Editor Queue

https://tools.ietf.org/pdf/rfc821.pdf

RFC 821

SIMPLE MAIL TRANSFER PROTOCOL

Jonathan B. Postel

5 / 71

TABLE OF CONTENTS

The following are the SMTP commands:

HELO <SP> <domain> <CRLF>

MAIL <SP> FROM:<reverse-path> <CRLF>

RCPT <SP> TO:<forward-path> <CRLF>

DATA <CRLF>

RSET <CRLF>

SEND <SP> FROM:<reverse-path> <CRLF>

SOML <SP> FROM:<reverse-path> <CRLF>

SAML <SP> FROM:<reverse-path> <CRLF>

VRFY <SP> <string> <CRLF>

EXPN <SP> <string> <CRLF>

HELP [<SP> <string>] <CRLF>

NOOP <CRLF>

QUIT <CRLF>

TURN <CRLF>

https://tools.ietf.org/html/rfc1149

[Docs] [txt|pdf] [Errata]

Updated by: 2549, 6214 EXPERIMENTAL
Errata Exist

Network Working Group D. Waitzman
Request for Comments: 1149 BBN STC
1 April 1990

A Standard for the Transmission of IP Datagrams on Avian Carriers

Status of this Memo

This memo describes an experimental method for the encapsulation of IP datagrams in avian carriers. This specification is primarily useful in Metropolitan Area Networks. This is an experimental, not recommended standard. Distribution of this memo is unlimited.

Overview and Rational

Avian carriers can provide high delay, low throughput, and low altitude service. The connection topology is limited to a single point-to-point path for each carrier, used with standard carriers, but many carriers can be used without significant interference with each other, outside of early spring. This is because of the 3D ether

8 / 71

Why VCS?

1. We want people to be able to work simultaneously, not serially.

2. When people are working at the same time, we want their changes to not conflict with each other.

3. We want to archive every version of everything that has ever existed — ever.

VCS Flashback

Generation	Networking	Operations	Concurrency	Examples
First	None	One file at a time	Locks	RCS, SCCS
Second	Centralized	Multi-file	Merge before commit	CVS, SourceSafe, Subversion, Team Foundation Server
Third	Distributed	Changesets	Commit before merge	Bazaar, Git, Mercurial

VCS Flashback

Generation	Networking	Operations	Concurrency	Examples
First	None	One file at a time	Locks	RCS, SCCS
Second	Centralized	Multi-file	Merge before commit	CVS, SourceSafe, Subversion, Team Foundation Server
Third	Distributed	Changesets	Commit before merge	Bazaar, Git, Mercurial

- Revision Control System
- Version Control System
- Source Code Management

VCS Operations

- **Create**
- **Checkout**
- **Commit**
- **Update**
- Add
- Edit
- Delete
- Rename
- Move
- Status
- Diff
- Revert
- Log
- Tag
- Branch
- Merge
- Resolve
- Lock

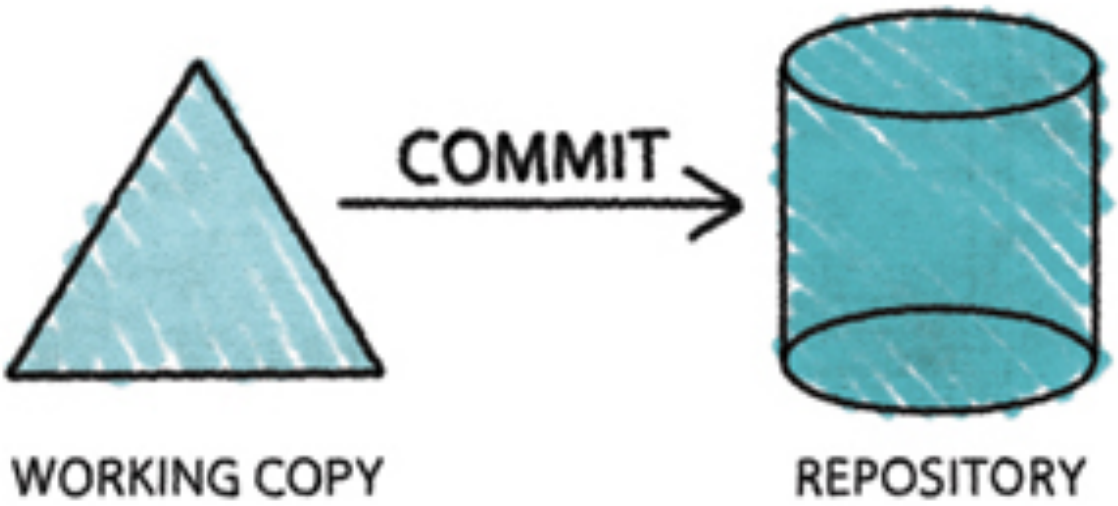
VCS Operations: Create (init)

repository = filesystem * time

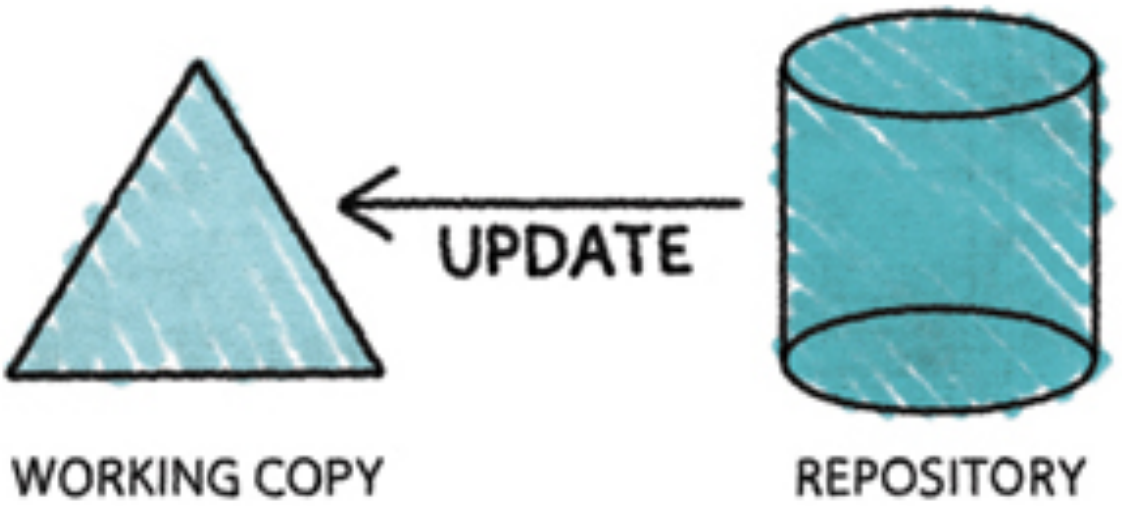
VCS Operations: Checkout

WORKING COPY

VCS Operations: Commit



VCS Operations: Update



"We were in this bad spot where we had thousands of people who wanted to participate, but in many ways I was the break point. I could not really scale to the point where I could work with thousands of people. So Git is my second big project, which was only created for me to maintain my first big project. This is literally how I work. I don't code for — well, I do code for fun — but I want to code for something meaningful, so every single project I've ever done has been something I needed and different." - Linus Torvalds

Advantages of Git

- Easy Branching
- Fast
- Offline
- Geography
- Private Workspace
- Flexible Workflows
- Easier Merging
- Implicit Backup
- Scale out, not just up
- Free and Open Source

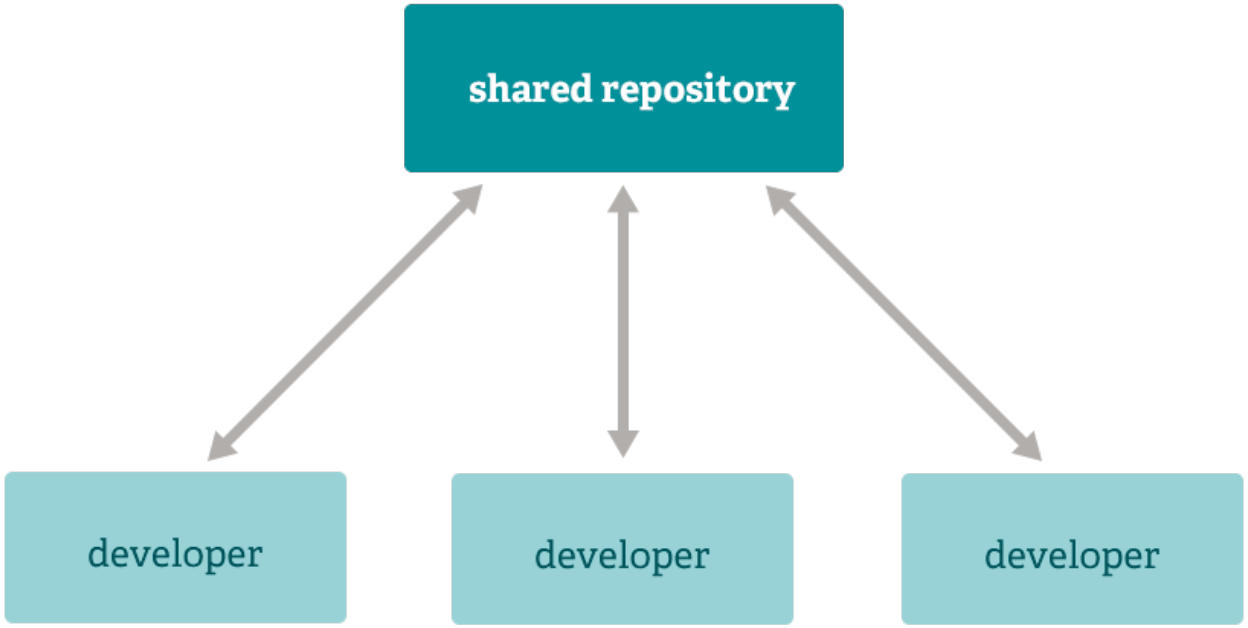
Git: Branching and Merging

- Frictionless Context Switching
- Role-Based Codelines
- Feature Based Workflow
- Disposable Experimentation

Git: Small and Fast

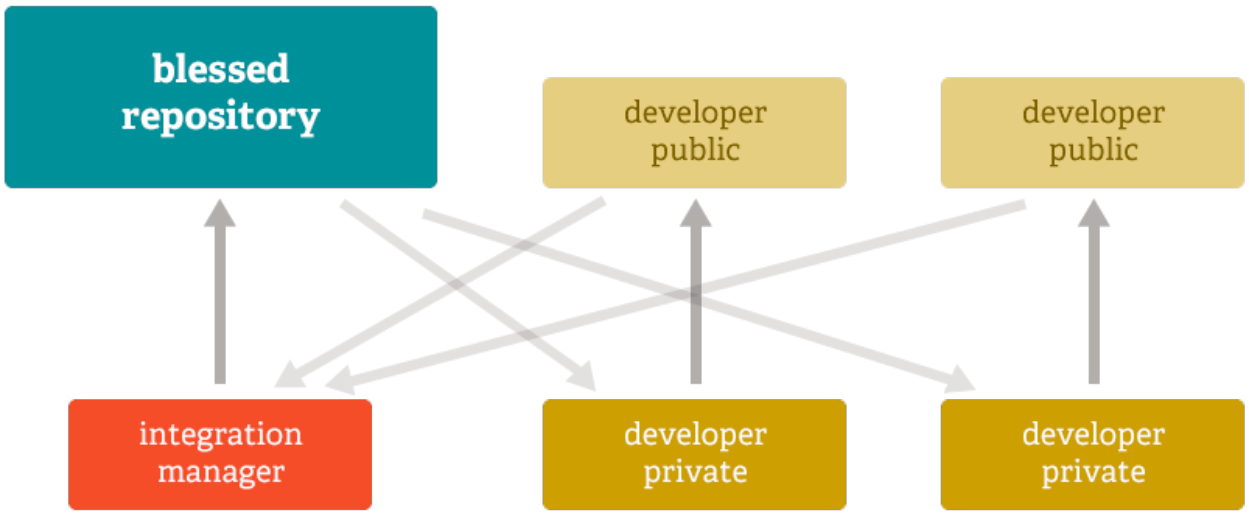
Git: Flexible, Distributed Workflows

Subversion-Style Workflow



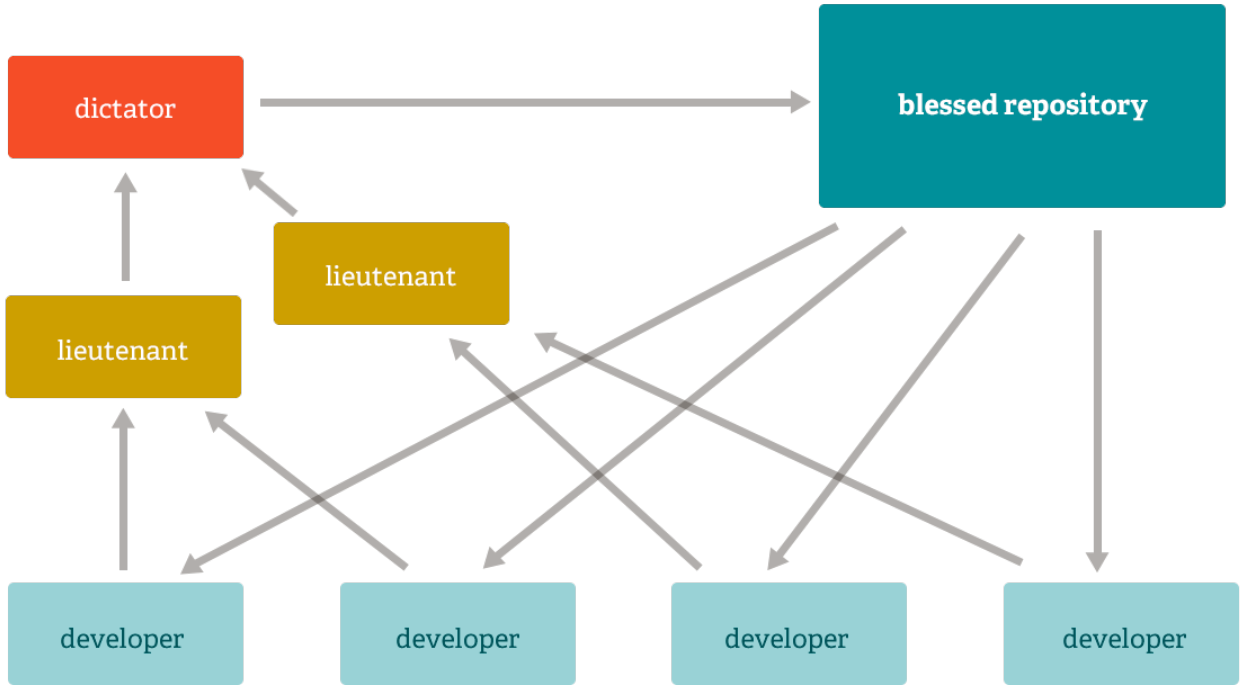
Git: Flexible, Distributed Workflows

Integration Manager Workflow



Git: Flexible, Distributed Workflows

Dictator and Lieutenants Workflow



```
ned@knife:/$ 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

work on the current change (see also: git help everyday)
   add        Add file contents to the index
   mv         Move or rename a file, a directory, or a symlink
   reset      Reset current HEAD to the specified state
   rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect     Use binary search to find the commit that introduced a bug
   grep       Print lines matching a pattern
   log        Show commit logs
   show       Show various types of objects
   status     Show the working tree status
```

Installing Git on Linux

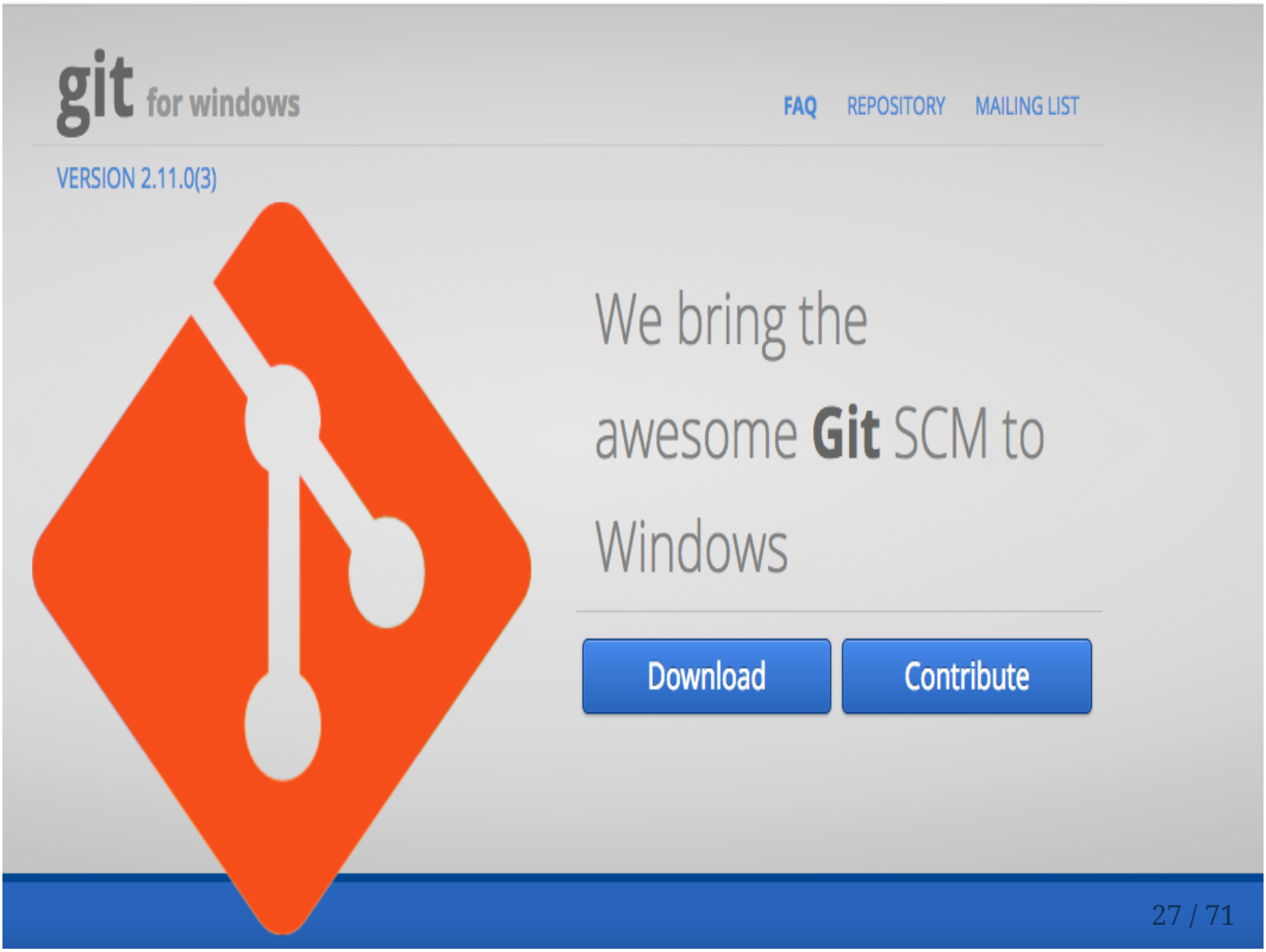
$ sudo yum install git-all

$ sudo apt-get install git-all

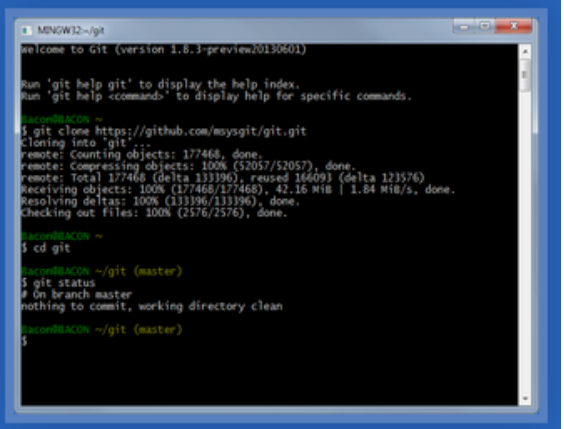
Installing Git on Mac

https://git-scm.com/download/mac

Or: `brew install git`

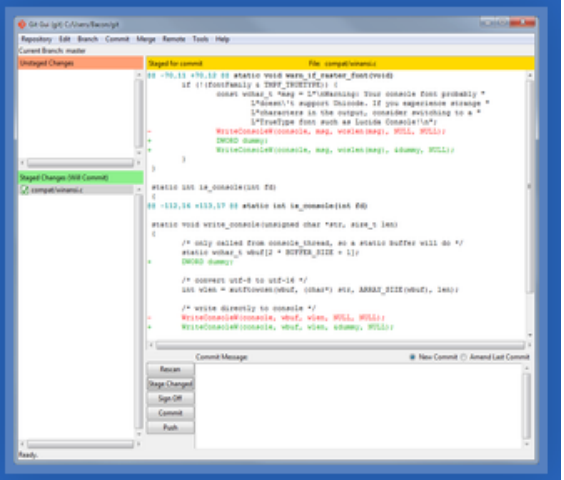

git for windows
FAQ
REPOSITORY
MAILING LIST
VERSION 2.11.0(3)
We bring the awesome Git SCM to Windows
Download
Contribute
27 / 71

Git for Windows focuses on offering a lightweight, native set of tools that bring the full feature set of the Git SCM to Windows while providing appropriate user interfaces for experienced Git users and novices alike.



Git BASH

Git for Windows provides a BASH emulation used to run Git from the command line. *NIX users should feel right at home, as the BASH emulation behaves just like the "git" command in LINUX and UNIX environments.



Git GUI

As Windows users commonly expect graphical user interfaces, Git for Windows also provides the Git GUI, a powerful alternative to Git BASH, offering a graphical version of just about every Git command line function, as well as comprehensive visual diff tools.

Shell Integration

Simply right-click on a folder in Windows Explorer to access the BASH or GUI.

https://git-scm.com/downloads/guis

git --local-branching-on-the-cheap

Search entire site...

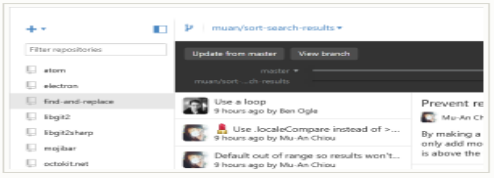
About
Documentation
Blog
Downloads
GUI Clients
Logos
Community

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

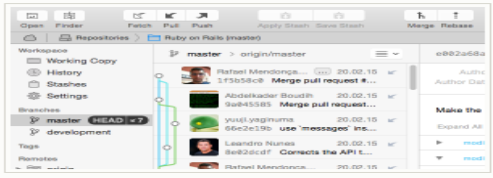
GUI Clients

Git comes with built-in GUI tools for committing (git-gui) and browsing (gitk), but there are several third-party tools for users looking for platform-specific experience.

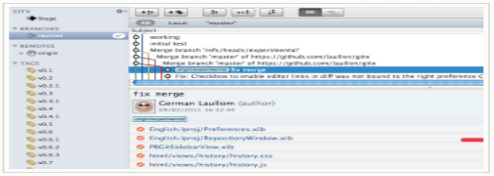
Only show GUIs for my OS (Mac)



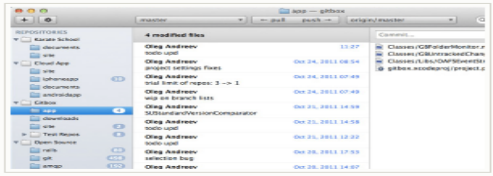
GitHub Desktop
Platforms: Windows, Mac
Price: Free



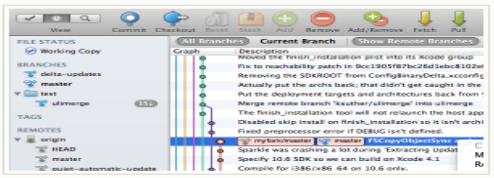
Tower
Platforms: Windows, Mac
Price: $79/user (Free 30 day trial)



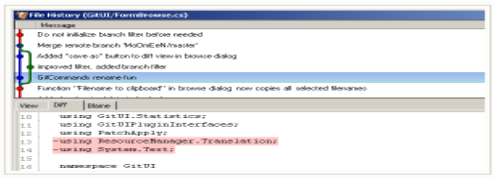
GitX-dev
Platforms: Mac
Price: Free



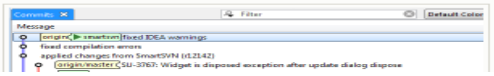
Gitbox
Platforms: Mac
Price: $14.99



SourceTree
Platforms: Mac, Windows
Price: Free



Git Extensions
Platforms: Windows
Price: Free



29 / 71

GitHub Desktop

Overview Release Notes | Help

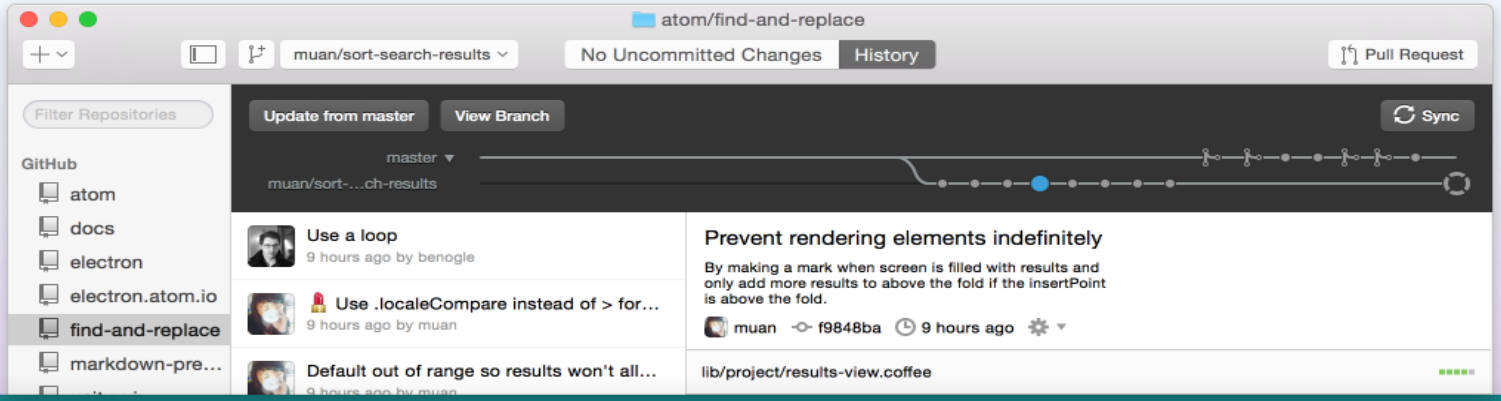
Simple collaboration from your desktop

GitHub Desktop is a seamless way to contribute to projects on **GitHub** and **GitHub Enterprise**.

Available for Mac and **Windows**

Download GitHub Desktop
OS X 10.9 or later

By clicking the Download button you agree to the **End-User License Agreement**



Your GitHub workflow in one native app

Clone repositories → Create branches → Commit changes → Share code

30 / 71

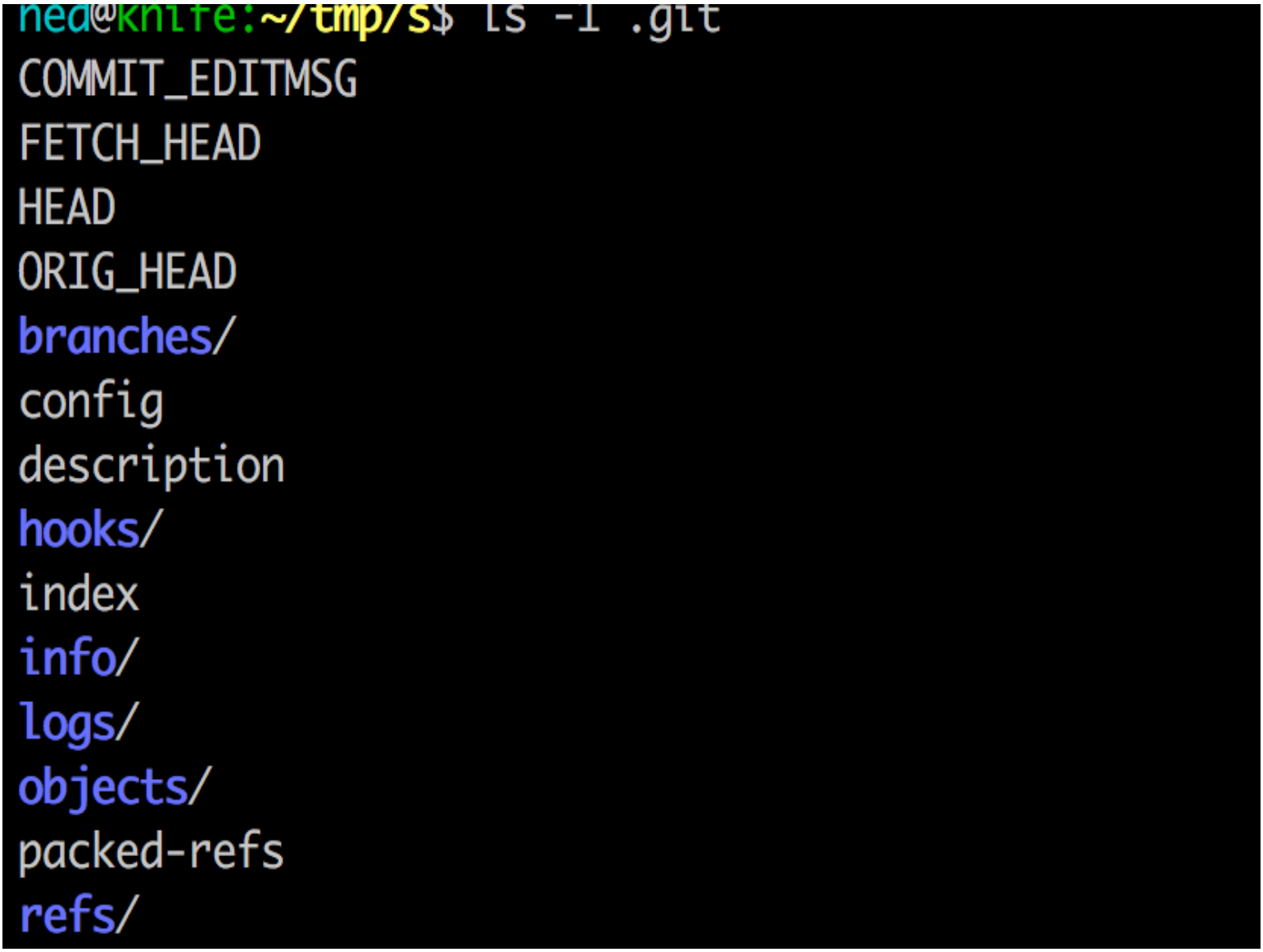
Git Essentials

create a new repository

create a new directory, open it and perform a

```
git init
```

to create a new git repository.


ned@knife:~/tmp/s$ ls -l .git
COMMIT_EDITMSG
FETCH_HEAD
HEAD
ORIG_HEAD
branches/
config
description
hooks/
index
info/
logs/
objects/
packed-refs
refs/

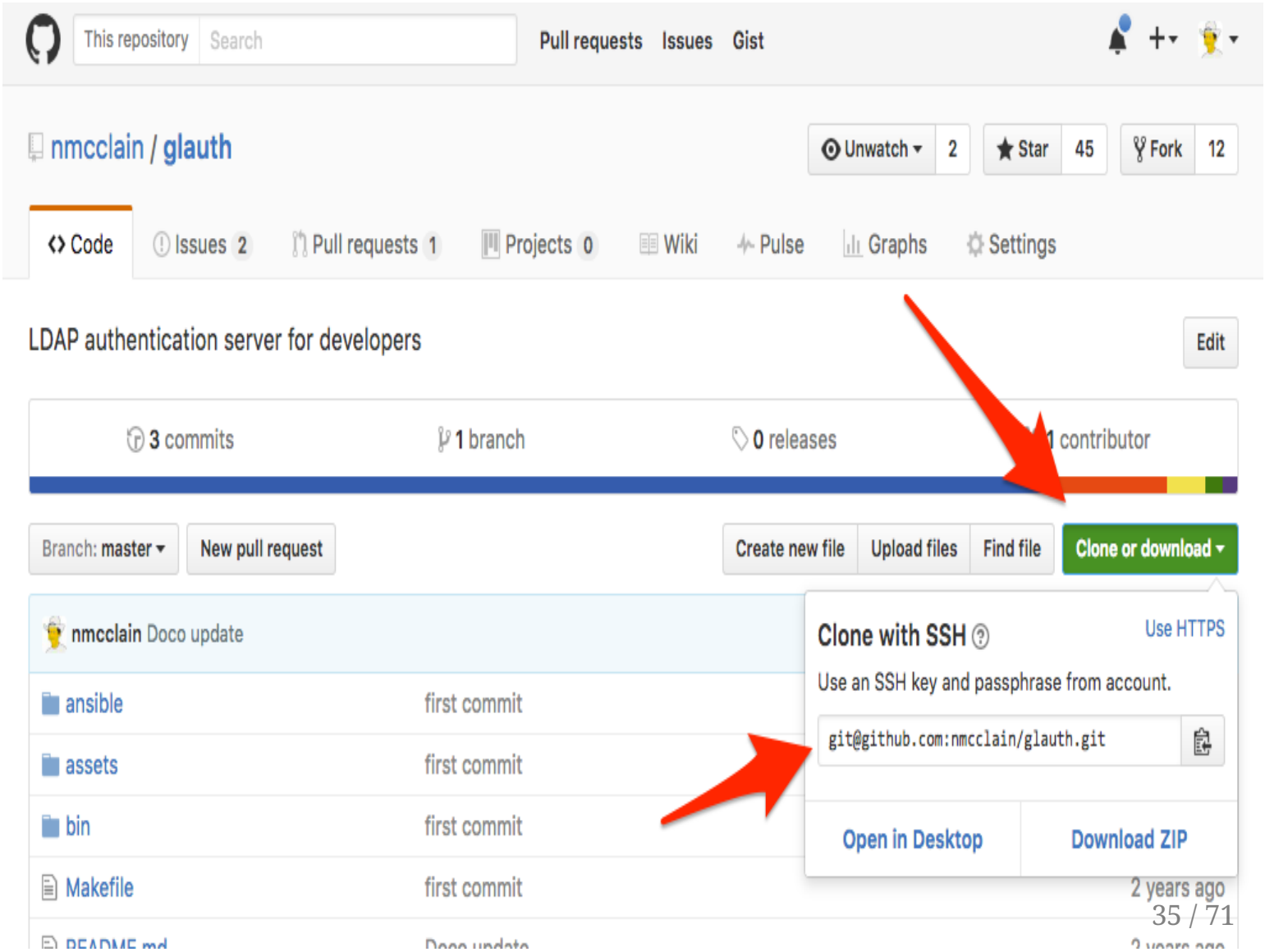
checkout a repository

create a working copy of a local repository by running the command

```
git clone /path/to/repository
```

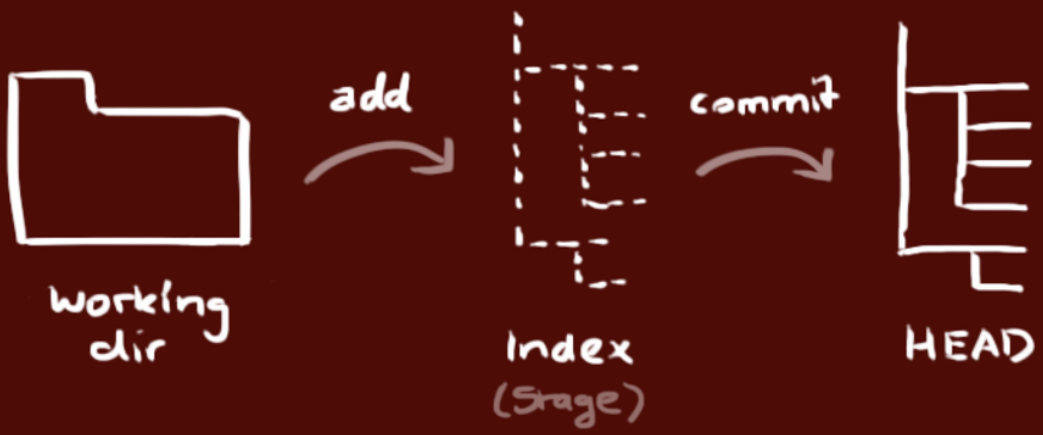
when using a remote server, your command will be

```
git clone username@host:/path/to/repository
```


This repository
Search
Pull requests
Issues
Gist
nmcclain / glauth
Unwatch 2
Star 45
Fork 12
Code
Issues 2
Pull requests 1
Projects 0
Wiki
Pulse
Graphs
Settings
LDAP authentication server for developers
Edit
3 commits
1 branch
0 releases
1 contributor
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
nmcclain Doco update
ansible first commit
assets first commit
bin first commit
Makefile first commit
2 years ago
Clone with SSH
Use HTTPS
Use an SSH key and passphrase from account.
git@github.com:nmcclain/glauth.git
Open in Desktop
Download ZIP

workflow

your local repository consists of three "trees" maintained by git. the first one is your `Working Directory` which holds the actual files. the second one is the `Index` which acts as a staging area and finally the `HEAD` which points to the last commit you've made.



36 / 71

add & commit

You can propose changes (add it to the **Index**) using

```
git add <filename>
```

```
git add *
```

This is the first step in the basic git workflow. To actually commit these changes use

```
git commit -m "Commit message"
```

Now the file is committed to the **HEAD**, but not in your remote repository yet.

37 / 71

pushing changes

Your changes are now in the **HEAD** of your local working copy. To send those changes to your remote repository, execute

```
git push origin master
```

Change *master* to whatever branch you want to push your changes to.

If you have not cloned an existing repository and want to connect your repository to a remote server, you need to add it with

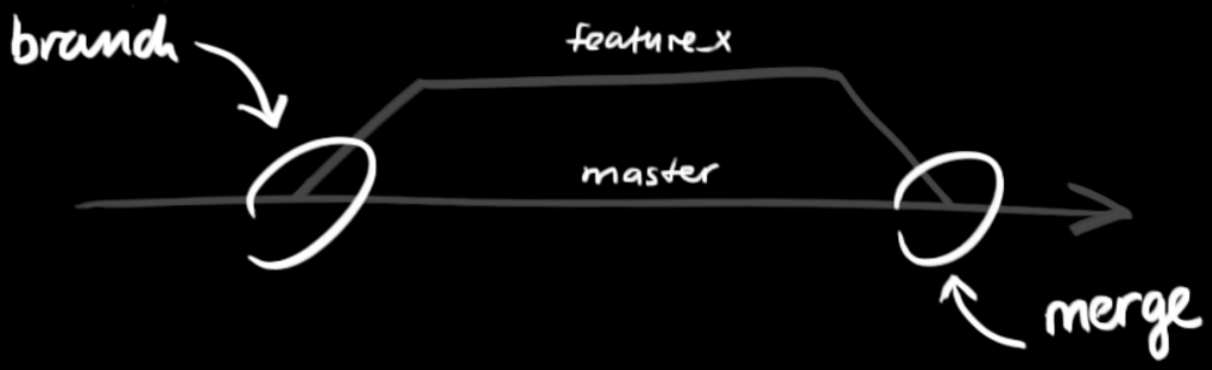
```
git remote add origin <server>
```

Now you are able to push your changes to the selected remote server

38 / 71

branching

Branches are used to develop features isolated from each other. The *master* branch is the "default" branch when you create a repository. Use other branches for development and merge them back to the master branch upon completion.



create a new branch named "feature_x" and switch to it using

```
git checkout -b feature_x
```

switch back to master

```
git checkout master
```

and delete the branch again

```
git branch -d feature_x
```

a branch is *not available to others* unless you push the branch to your remote repository

```
git push origin <branch>
```

update & merge

to update your local repository to the newest commit, execute

```
git pull
```

in your working directory to *fetch* and *merge* remote changes.

to merge another branch into your active branch (e.g. master), use

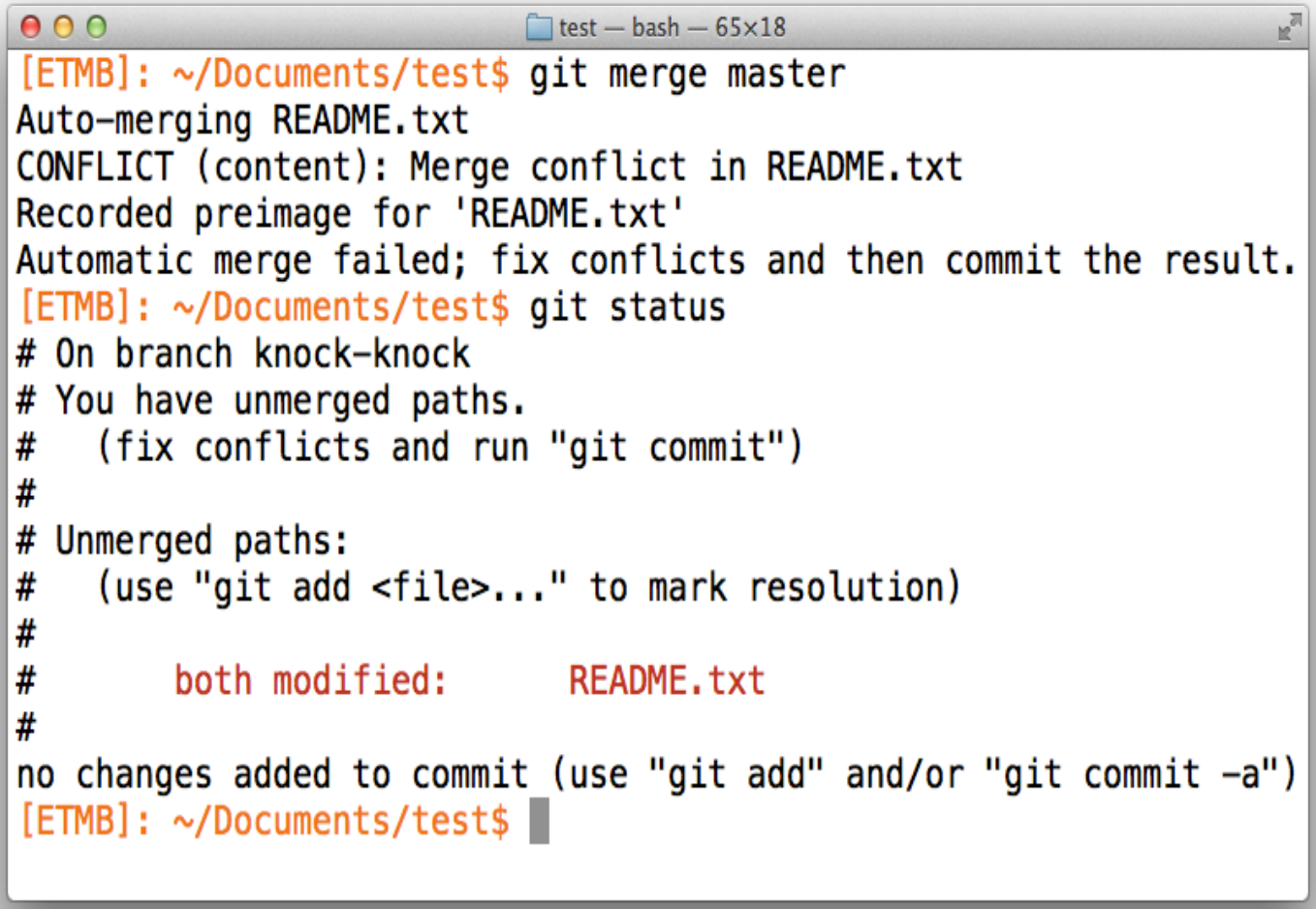
```
git merge <branch>
```

in both cases git tries to auto-merge changes. Unfortunately, this is not always possible and results in *conflicts*. You are responsible to merge those *conflicts* manually by editing the files shown by git. After changing, you need to mark them as merged with

```
git add <filename>
```

before merging changes, you can also preview them by using

```
git diff <source_branch> <target_branch>
```


test — bash — 65×18
[ETMB]: ~/Documents/test$ git merge master
Auto-merging README.txt
CONFLICT (content): Merge conflict in README.txt
Recorded preimage for 'README.txt'
Automatic merge failed; fix conflicts and then commit the result.
[ETMB]: ~/Documents/test$ git status
On branch knock-knock
You have unmerged paths.
(fix conflicts and run "git commit")
#
Unmerged paths:
(use "git add <file>..." to mark resolution)
#
both modified: README.txt
#
no changes added to commit (use "git add" and/or "git commit -a")
[ETMB]: ~/Documents/test$


<<<<<<< HEAD:config/environment.rb
# this is my new version
foo = Bar.new
=======
# it conflicts with this old one
foo = Baz.new
>>>>>>> Decided to use a softer route t
end
43 / 71

tagging

it's recommended to create tags for software releases. this is a known concept, which also exists in SVN. You can create a new tag named *1.0.0* by executing

```
git tag 1.0.0 1b2e1d63ff
```

the *1b2e1d63ff* stands for the first 10 characters of the commit id you want to reference with your tag. You can get the commit id by looking at the...

44 / 71

log

in its simplest form, you can study repository history using.. `git log`

You can add a lot of parameters to make the log look like what you want.

To see only the commits of a certain author:

```
git log --author=bob
```

To see a very compressed log where each commit is one line:

```
git log --pretty=oneline
```

Or maybe you want to see an ASCII art tree of all the branches,

decorated with the names of tags and branches:

```
git log --graph --oneline --decorate --all
```

See only which files have changed:

```
git log --name-status
```

These are just a few of the possible parameters you can use. For more,

see `git log --help`

replace local changes

In case you did something wrong, which for sure never happens ;), you can replace local changes using the command

```
git checkout -- <filename>
```

this replaces the changes in your working tree with the last content in HEAD. Changes already added to the index, as well as new files, will be kept.

If you instead want to drop all your local changes and commits, fetch the latest history from the server and point your local master branch at it like this

```
git fetch origin
```

```
git reset --hard origin/master
```

46 / 71

useful hints

built-in git GUI

```
gitk
```

use colorful git output

```
git config color.ui true
```

show log on just one line per commit

```
git config format.pretty oneline
```

use interactive adding

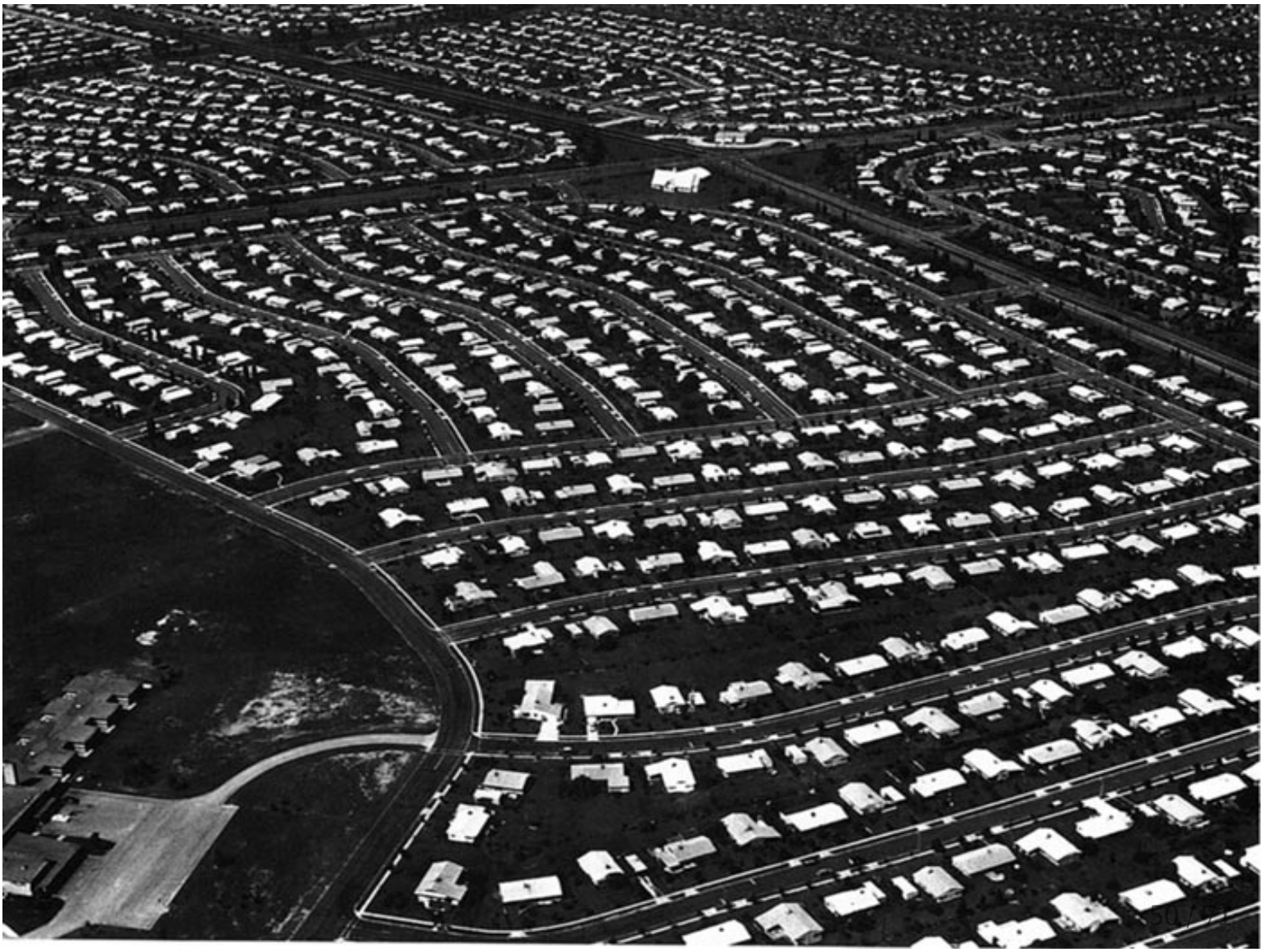
```
git add -i
```

47 / 71

Git Collaboration Tips

- Comment and format code per project/team standards
- Use reasonable project/file structure
- Good commit message
- Pull from remote often
- Push to remote often

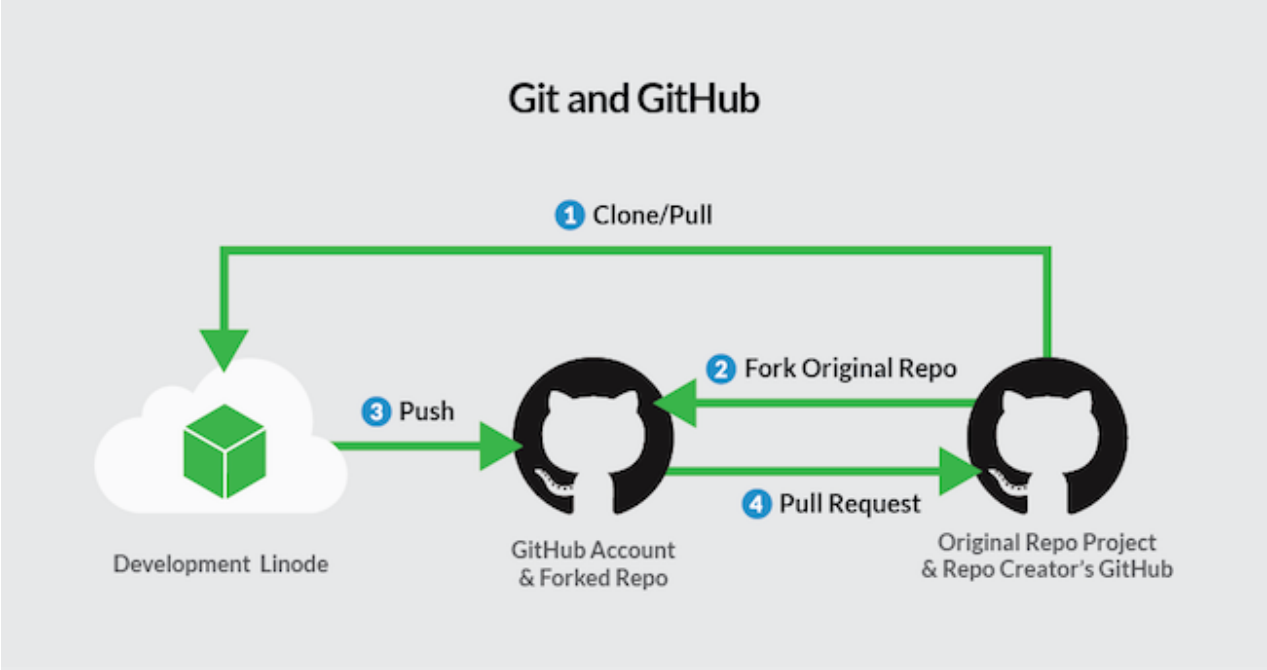
Innovation Often Causes Unexpected Change



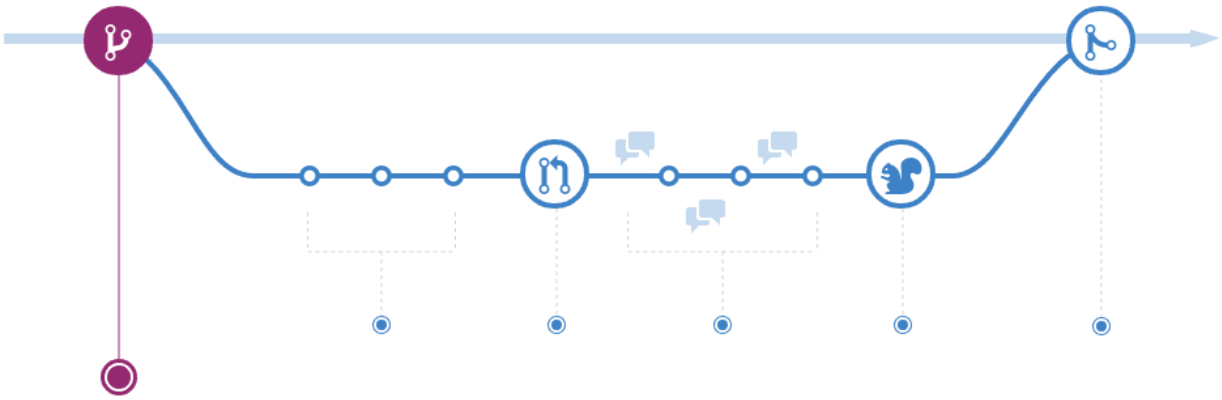

github
SOCIAL CODING


git
github
SOCIAL CODING

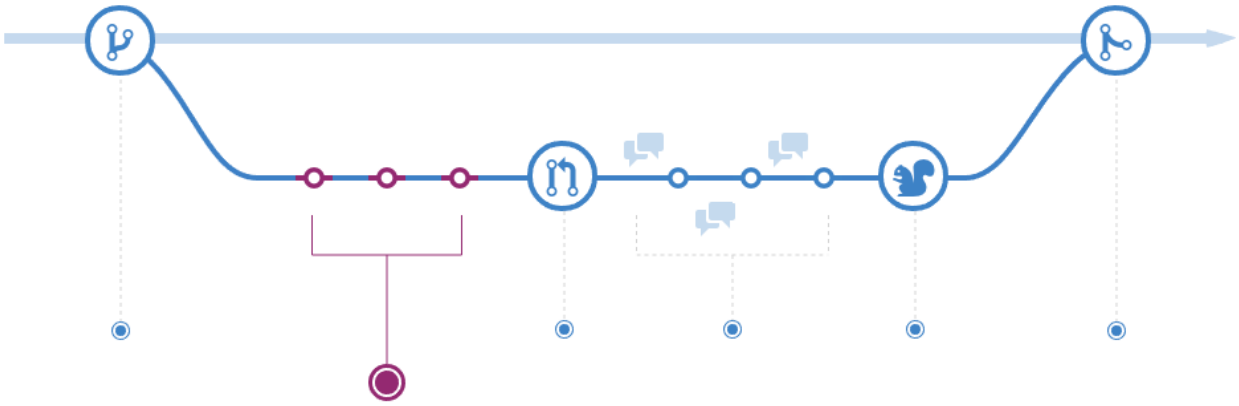
GitHub Flow


Git and GitHub
1 Clone/Pull
2 Fork Original Repo
3 Push
4 Pull Request
Development Linode
GitHub Account
& Forked Repo
Original Repo Project
& Repo Creator's GitHub

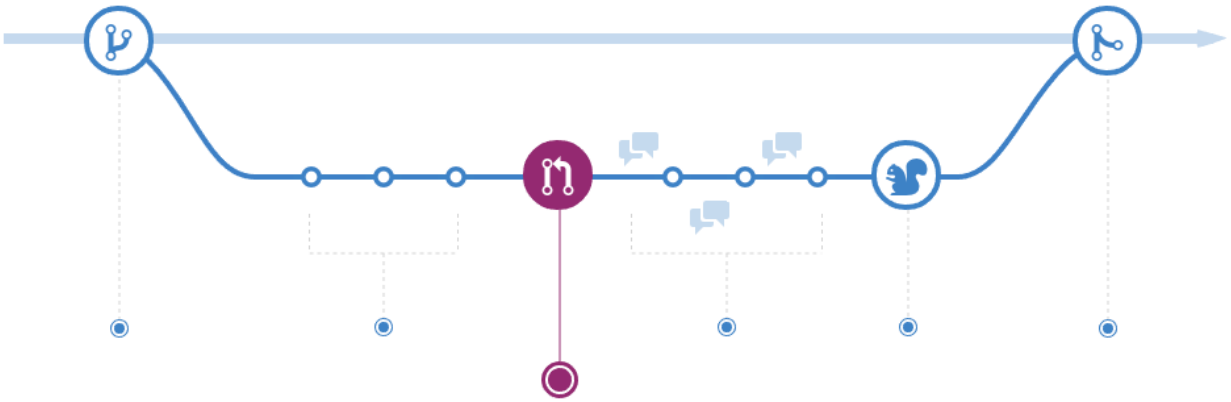
Create a branch



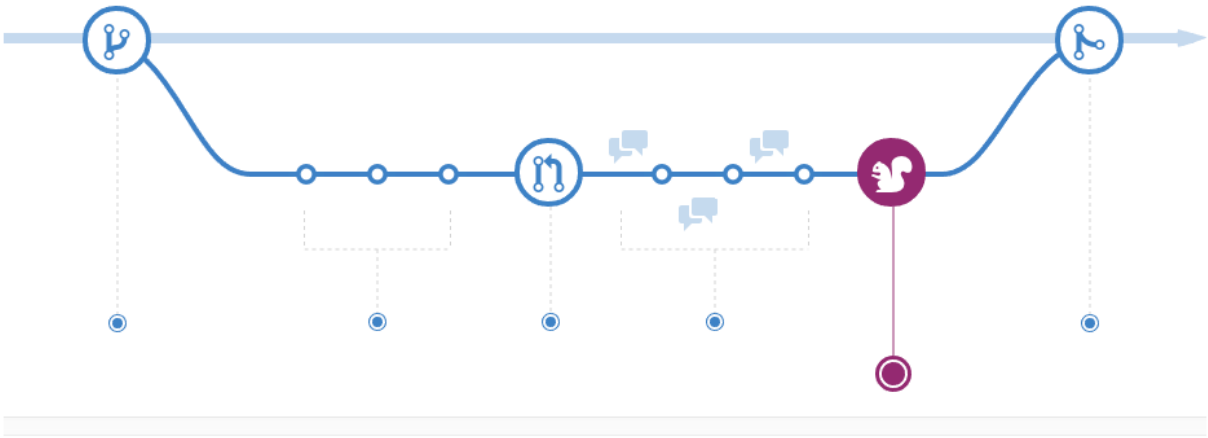
Add commits



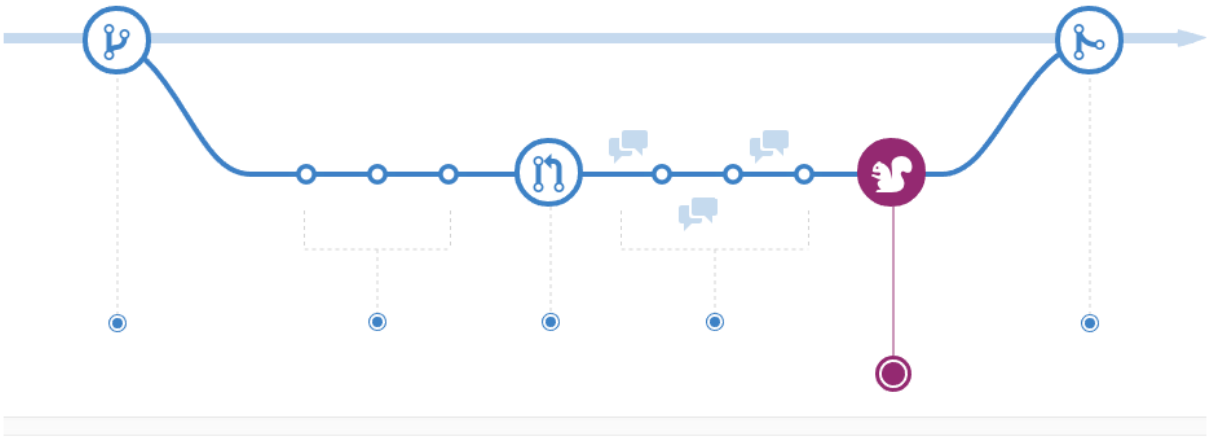
Open a Pull Request



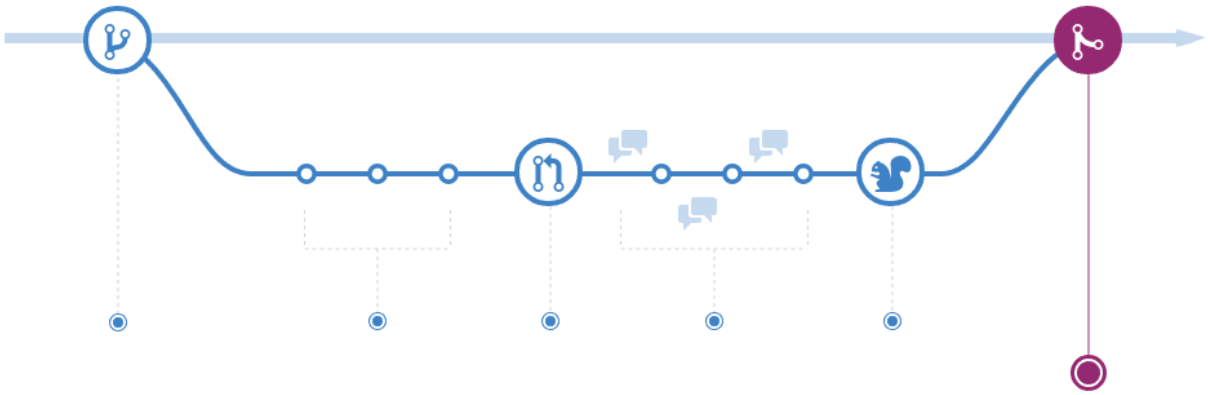
Discuss and review your code

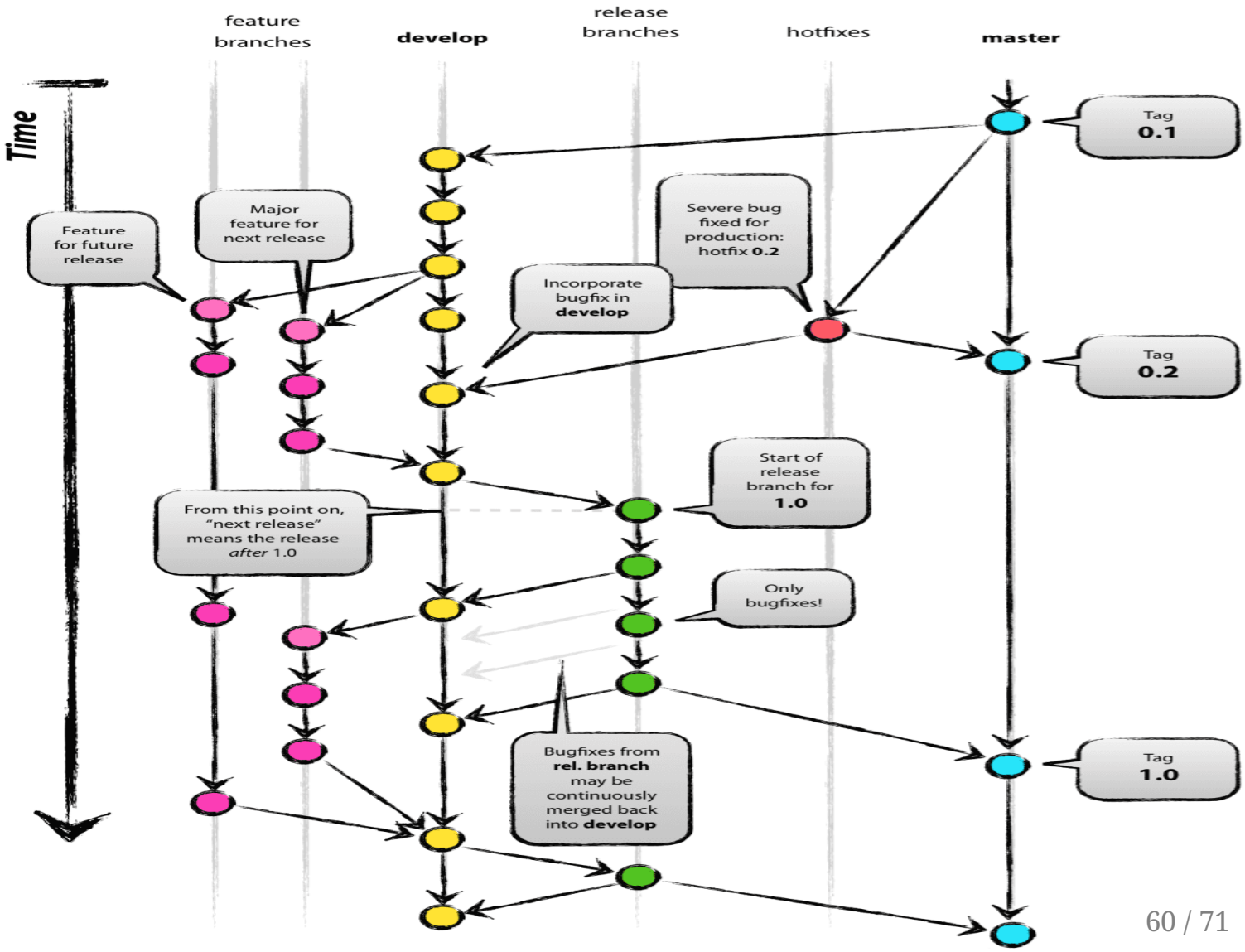


Deploy



Merge



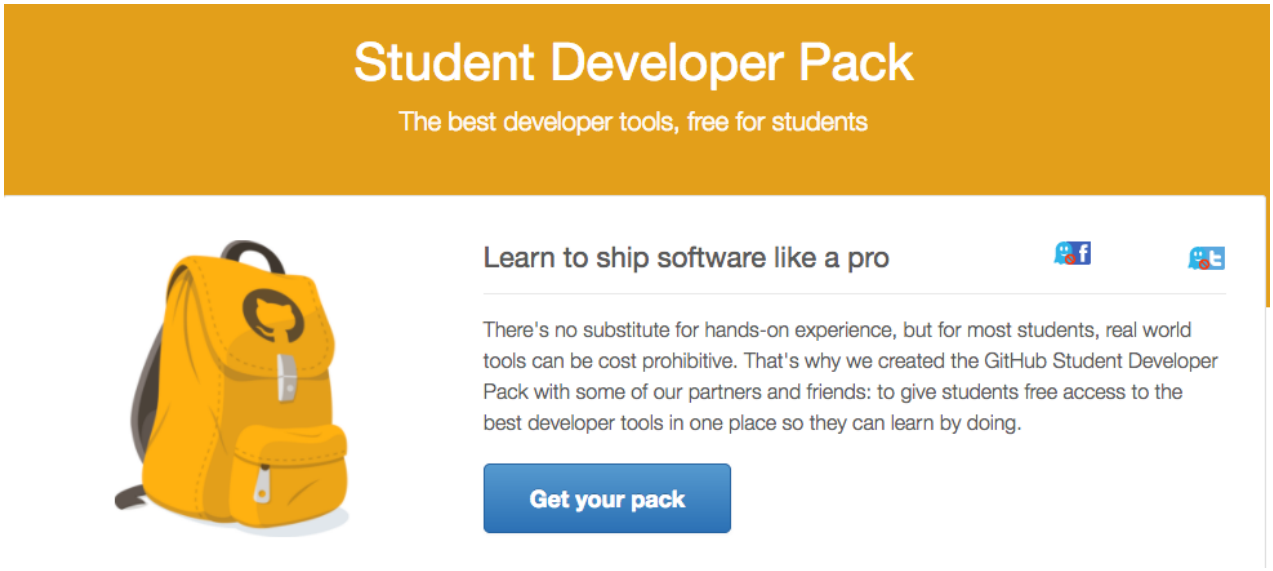

feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

60 / 71


JUST
SHIP
IT.
61 / 71

GitHub Student DevPack

https://education.github.com/pack



GitHub

Powerful collaboration, code review, and code management

DETAILS Unlimited private repositories (normally $7/month) while you are a student.

axosoft GitKraken

The downright luxurious Git client for Windows, Mac and Linux

DETAILS GitKraken Pro account for 1 user. Free for 1 year (normally $60/year).

{ } HACKHANDS

Live programming help available 24/7

DETAILS $25 in platform credit

Microsoft Imagine

A suite of Microsoft Azure cloud services and developer tools, including the Visual Studio IDE

DETAILS Microsoft Azure, Visual Studio Community 2015 and the rest of the Microsoft developer tools while you're a student

63 / 71

A hackable text editor for the 21st Century

DETAILS Open Source by GitHub, free for everyone

Access to the AWS cloud, free training, and collaboration resources

DETAILS Student Developer Pack members receive up to $110 in bonus AWS credits for a total of $75-$150

Install cloud applications in a single click

DETAILS Business 3 plan (normally $49/month) free for one year

Crowdsourcing and data enrichment platform

DETAILS Access to the Crowdflower platform

64 / 71

Affordable registration, hosting, and domain management

DETAILS One year SSL certificate (normally $9/year)

DETAILS One year domain name registration on the .me TLD (normally $18.99/year)

Email infrastructure as a service

DETAILS Student plan 15K free emails/month (normally limited to 200 free emails/day) while you're a student

Web and mobile payments, built for developers

DETAILS Waived transaction fees on first $1000 in revenue processed

Dynamic A/B testing, smart push notifications and custom analytics for native mobile apps

DETAILS Complete access to the suite of tools for native mobile apps. Unlimited access to the platform free for 6 months.

65 / 71

Cloud-based infrastructure monitoring

DETAILS Pro Account, including 10 servers. Free for 2 years.

Simple cloud hosting, built for developers

DETAILS $50 in platform credit for new users

Simple DNS management with one-click services and a robust API

DETAILS Personal hosted DNS plan (normally $5/month) for one year

Learn Web Development from the premier coding bootcamp for launching developers.

DETAILS Free one-month membership ($149 value) to Community-Powered Bootcamp: a proven online Web Development course with a curated community of learners.

66 / 71

transifex

Localization platform that easily integrates with your code base

DETAILS One year free of the Starter plan, a $99/month value. Get 50,000 hosted words, unlimited projects, and access to translation partners to bring your software to a global market from the start.

Continuous integration platform for open source and private projects

DETAILS Private builds (normally $69/month) while you're a student

Enroll in a Nanodegree program, and launch your career in Web and Mobile Development, Machine Learning, Data Science, and more.

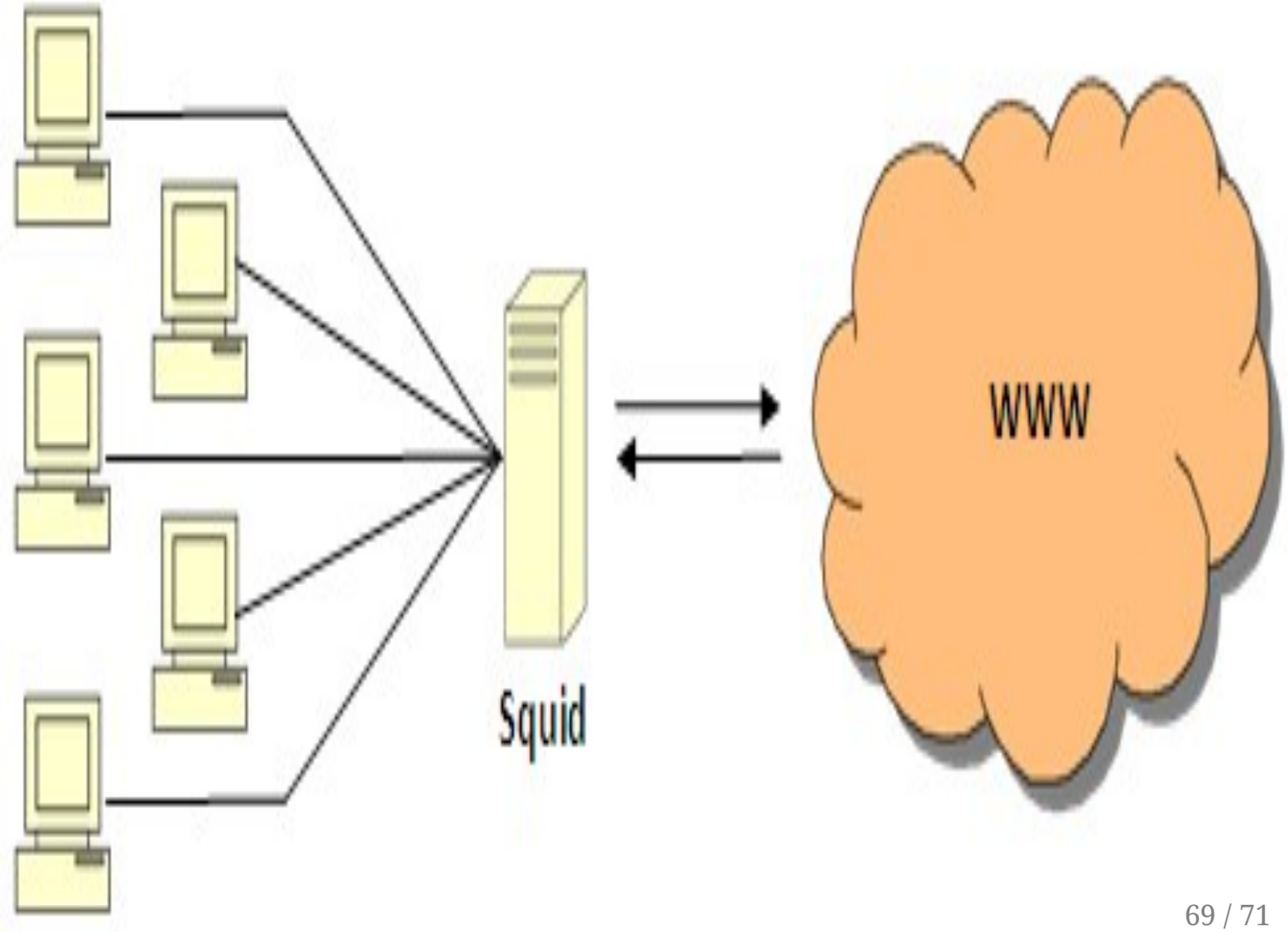
DETAILS Student Developer Pack members get one month free access to any Nanodegree program (normally $199)

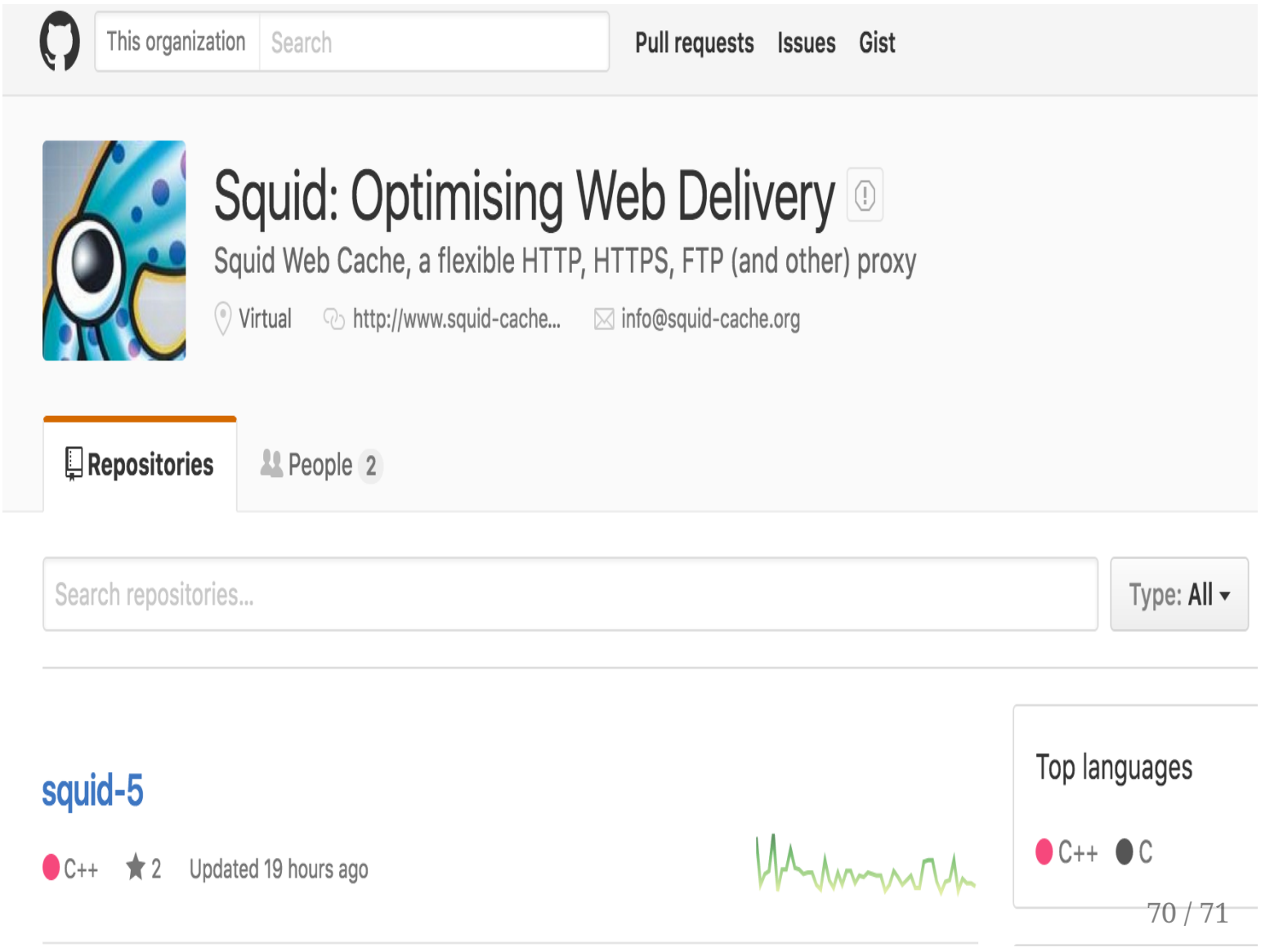
A complete suite of game development tools for PC, console, mobile, web and VR

DETAILS Unreal Engine while you're a student

67 / 71


Squid


Squid
www


This organization
Search
Pull requests
Issues
Gist
Squid: Optimising Web Delivery
Squid Web Cache, a flexible HTTP, HTTPS, FTP (and other) proxy
Virtual
http://www.squid-cache...
info@squid-cache.org
Repositories
People 2
Search repositories...
Type: All
squid-5
C++
2
Updated 19 hours ago
Top languages
C++
C

Next Week: More Git + Open source Business Models

Reading Assignment:

The Cathedral and the Bazaar
Chapters 5-8.

Have a great week!!!