

Git and Github

greenwich.edu.vn



Alliance with  Education

About Git

- Created by Linus Torvalds, creator of Linux, in 2005
 - Came out of Linux development community
 - Designed to do version control on Linux kernel
- Goals of Git:
 - Speed
 - Support for non-linear development (thousands of parallel branches)
 - Fully distributed
 - Able to handle large projects efficiently
 - *(A "git" is a cranky old man. Linus meant himself.)*

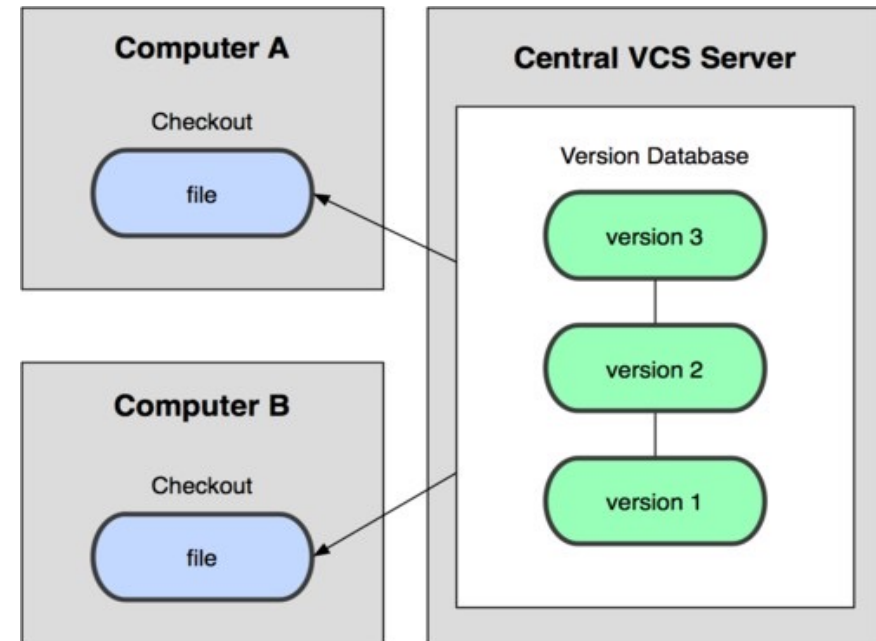


Installing/learning Git

- Git website: <http://git-scm.com/>
 - Free on-line book: <http://git-scm.com/book>
 - Reference page for Git: <http://gitref.org/index.html>
 - Git tutorial: <http://schacon.github.com/git/gittutorial.html>
 - Git for Computer Scientists:
 - <http://eagain.net/articles/git-for-computer-scientists/>
- At command line: (*where verb = config, add, commit, etc.*)
 - `git help verb`

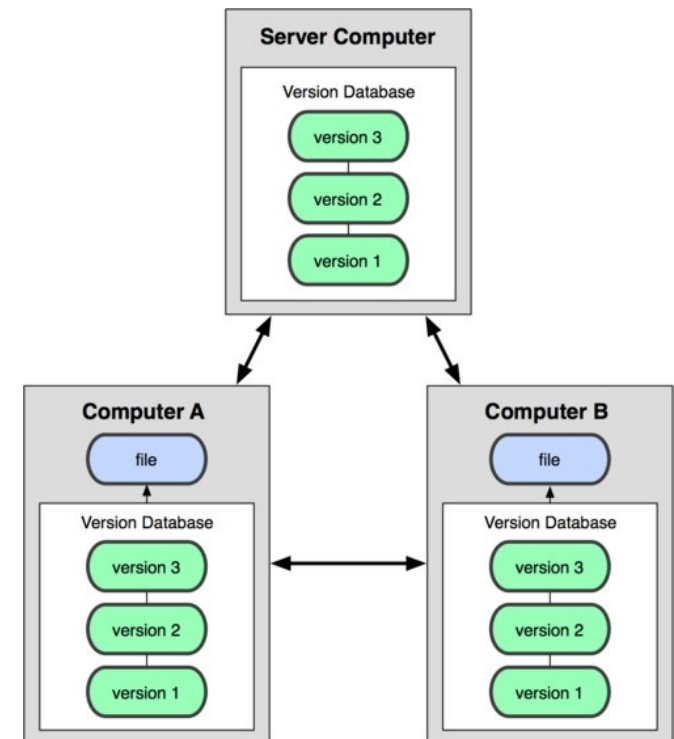
Centralized VCS

- In Subversion, CVS, Perforce, etc. A central server repository (repo) holds the "official copy" of the code
 - the server maintains the sole version history of the repo
- You make "checkouts" of it to your local copy
 - you make local modifications
 - your changes are not versioned
- When you're done, you "check in" back to the server
 - your checking increments the repo's version



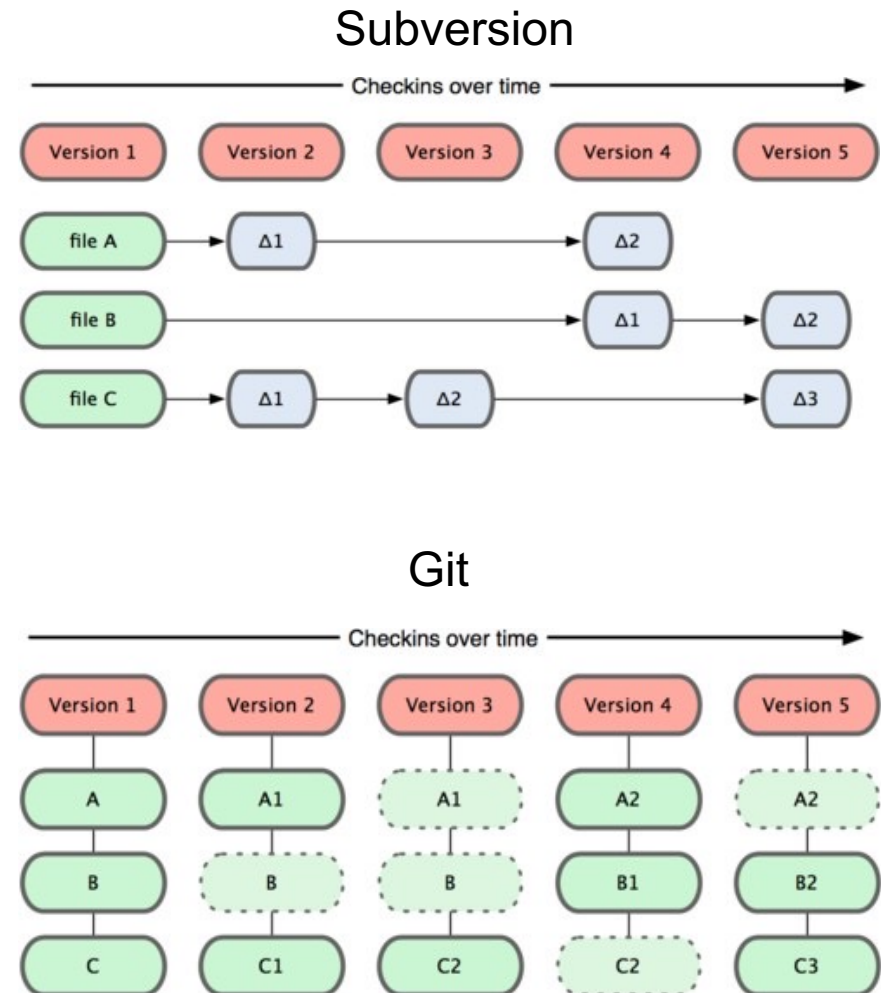
Distributed VCS (Git)

- In git, mercurial, etc., you don't "checkout" from a central repo
 - you "clone" it and "pull" changes from it
- Your local repo is a complete copy of everything on the remote server
 - yours is "just as good" as theirs
- Many operations are local:
 - check in/out from *local* repo
 - commit changes to *local* repo
 - local repo keeps version history
- When you're ready, you can "push" changes back to server



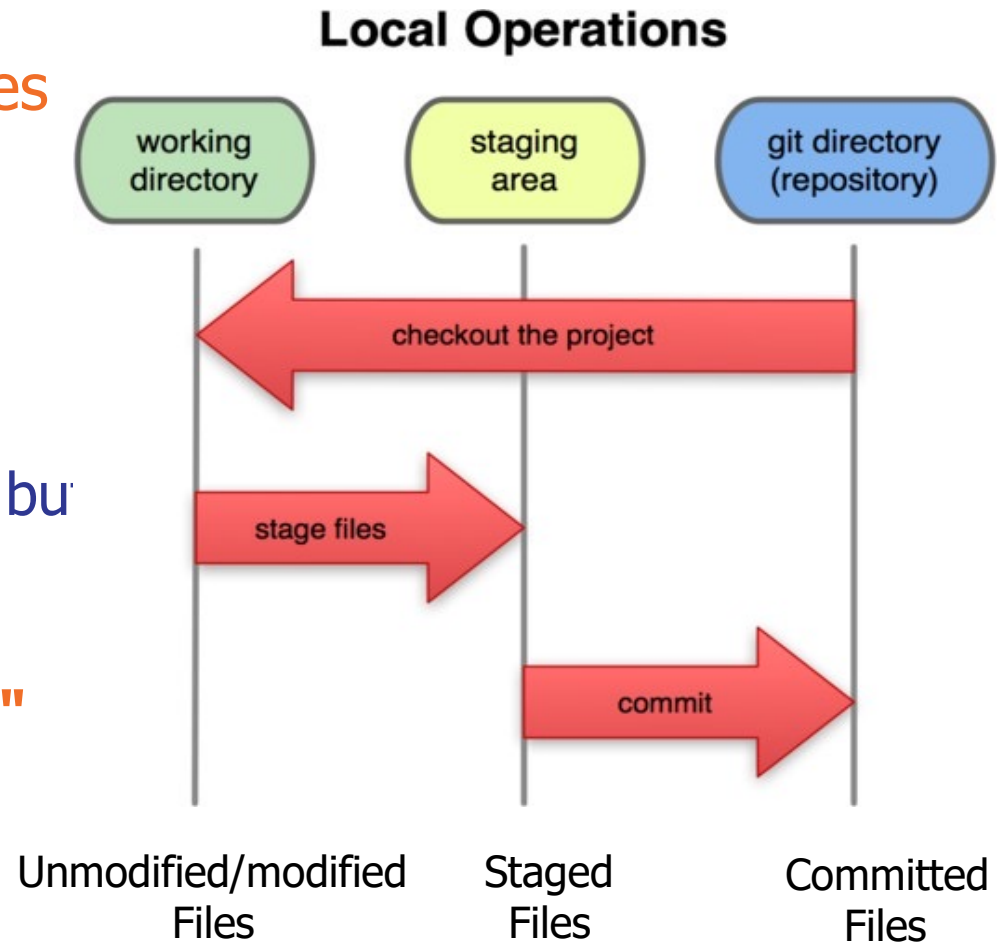
Git snapshots

- Centralized VCS like Subversion track version data on each individual file.
- Git keeps "snapshots" of the entire state of the project.
 - Each checkin version of the overall code has a copy of each file in it.
 - Some files change on a given checkin, some do not.
 - More redundancy, but faster.



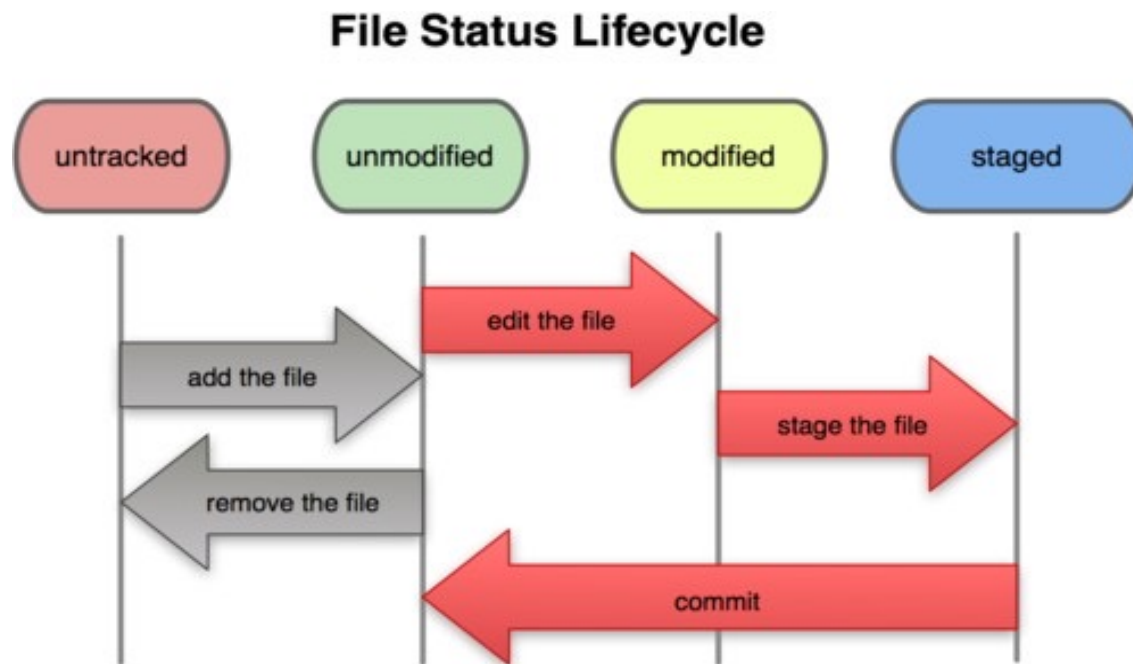
Local git areas

- In your local copy on git, files can be:
 - In your local repo
 - (committed)
 - Checked out and modified, but not yet committed
 - (working copy)
 - Or, in-between, in a **"staging" area**
 - Staged files are ready to be committed.
 - A commit saves a snapshot of all staged state.



Basic Git workflow

- **Modify** files in your working directory.
- **Stage** files, adding snapshots of them to your staging area.
- **Commit**, which takes the files in the staging area and stores that snapshot permanently to your Git directory.



Git commit checksums

- In Subversion each modification to the central repo increments the version # of the overall repo.
 - In Git, each user has their own copy of the repo, and commits changes to their local copy of the repo before pushing to the central server.
 - So Git generates a unique **SHA-1 hash** (40 character string of hex digits) for every commit.
 - Refers to commits by this ID rather than a version number.
- Often we only see the first 7 characters:
 - 1677b2d Edited first line of readme
 - 258efa7 Added line to readme
 - 0e52da7 Initial commit

Initial Git configuration

- Set the name and email for Git to use when you commit:
 - `git config --global user.name "Bugs Bunny"`
 - `git config --global user.email bugs@gmail.com`
- – You can call `git config -list` to verify these are set.
- Set the editor that is used for writing commit messages:
 - `git config --global core.editor nano`
 - (it is vim by default)

Creating a Git repo

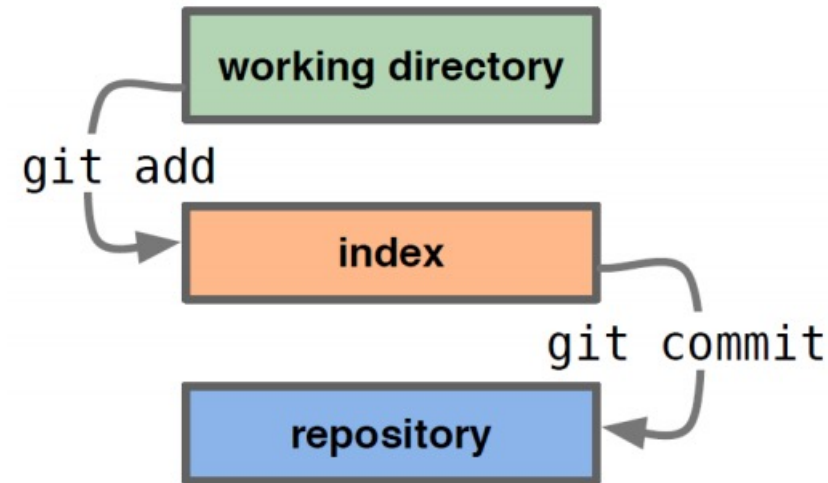
Two common scenarios: (only do one of these)

- To create a new **local Git repo** in your current directory:

- `git init`
 - This will create a `.git` directory in your current directory.
 - Then you can commit files in that directory into the repo.
- `git add filename`
- `git commit -m "commit message"`

- To **clone a remote repo** to your current directory:

- `git clone url localDirectoryName`
 - This will create the given local directory, containing a working copy of the files from the repo, and a `.git` directory (used to hold the staging area and your actual local repo)



Git commands

command	description
<code>git clone <i>url</i> [<i>dir</i>]</code>	copy a Git repository so you can add to it
<code>git add <i>file</i></code>	adds file contents to the staging area
<code>git commit</code>	records a snapshot of the staging area
<code>git status</code>	view the status of your files in the working directory and staging area
<code>git diff</code>	shows diff of what is staged and what is modified but unstaged
<code>git help [<i>command</i>]</code>	get help info about a particular command
<code>git pull</code>	fetch from a remote repo and try to merge into the current branch
<code>git push</code>	push your new branches and data to a remote repository
others: <code>init</code> , <code>reset</code> , <code>branch</code> , <code>checkout</code> , <code>merge</code> , <code>log</code> , <code>tag</code>	

Add and commit a file

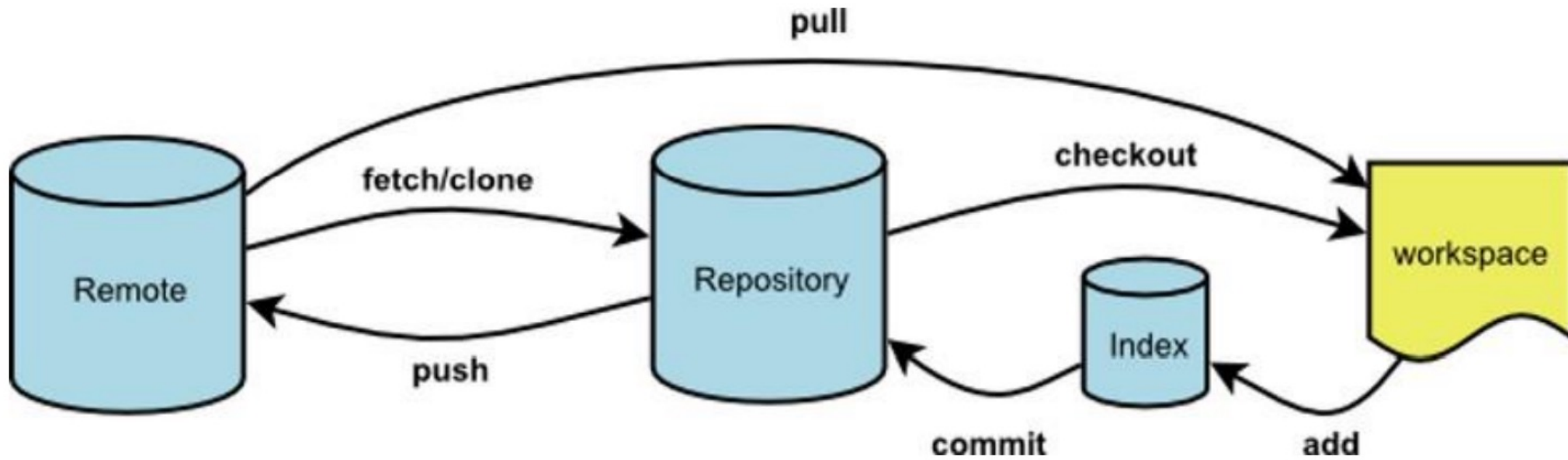
- The first time we ask a file to be tracked, *and every time before we commit a file*, we must add it to the staging area:
 - `git add Hello.java Goodbye.java`
 - Takes a snapshot of these files, adds them to the staging area.
 - In older VCS, "add" means "start tracking this file."
 - In Git, "add" means "add to staging area" so it will be part of the next commit.
- To move staged changes into the repo, we commit:
 - `git commit -m "Fixing bug #22"`
- To undo changes on a file before you have committed it:
 - `git reset HEAD -- filename` (unstages the file)
 - `git checkout -- filename` (undoes your changes)
 - All these commands are acting on your local version of repo.

Viewing/undoing changes

- To view status of files in working directory and staging area:
 - `git status` **or** `git status -s` (short version)
- To see what is modified but unstaged:
 - `git diff`
- To see a list of staged changes:
 - `git diff --cached`
- To see a log of all changes in your local repo:
 - `git log` **or** `git log --oneline` (shorter version)
 - 1677b2d Edited first line of readme
 - 258efa7 Added line to readme 0e52da7
 - Initial commit
 - `git log -5` (to show only the 5 most recent updates), etc.

An example workflow

```
[rea@attu1 superstar]$ emacs rea.txt
[rea@attu1 superstar]$ git status
    no changes added to commit
    (use "git add" and/or "git commit -a")
[rea@attu1 superstar]$ git status -s
    M rea.txt
[rea@attu1 superstar]$ git diff
    diff --git a/rea.txt b/rea.txt
[rea@attu1 superstar]$ git add rea.txt
[rea@attu1 superstar]$ git status
    #       modified:   rea.txt
[rea@attu1 superstar]$ git diff --cached
    diff --git a/rea.txt b/rea.txt
[rea@attu1 superstar]$ git commit -m "Created new text file"
```



Branching and merging

Git uses branching heavily to switch between multiple tasks.

- To create a new local branch:
 - `git branch name`
- To list all local branches: (* = current branch)
 - `git branch`
- To switch to a given local branch:
 - `git checkout branchname`
- To merge changes from a branch into the local master:
 - `git checkout master`
 - `git merge branchname`

Merge conflicts

- The conflicting file will contain <<< and >>> sections to indicate where Git was unable to resolve a conflict:

```
<<<<<<< HEAD:index.html
<div id="footer">todo: message here</div>
=====
<div id="footer">
  thanks for visiting our site
</div>
>>>>>> SpecialBranch:index.html
```

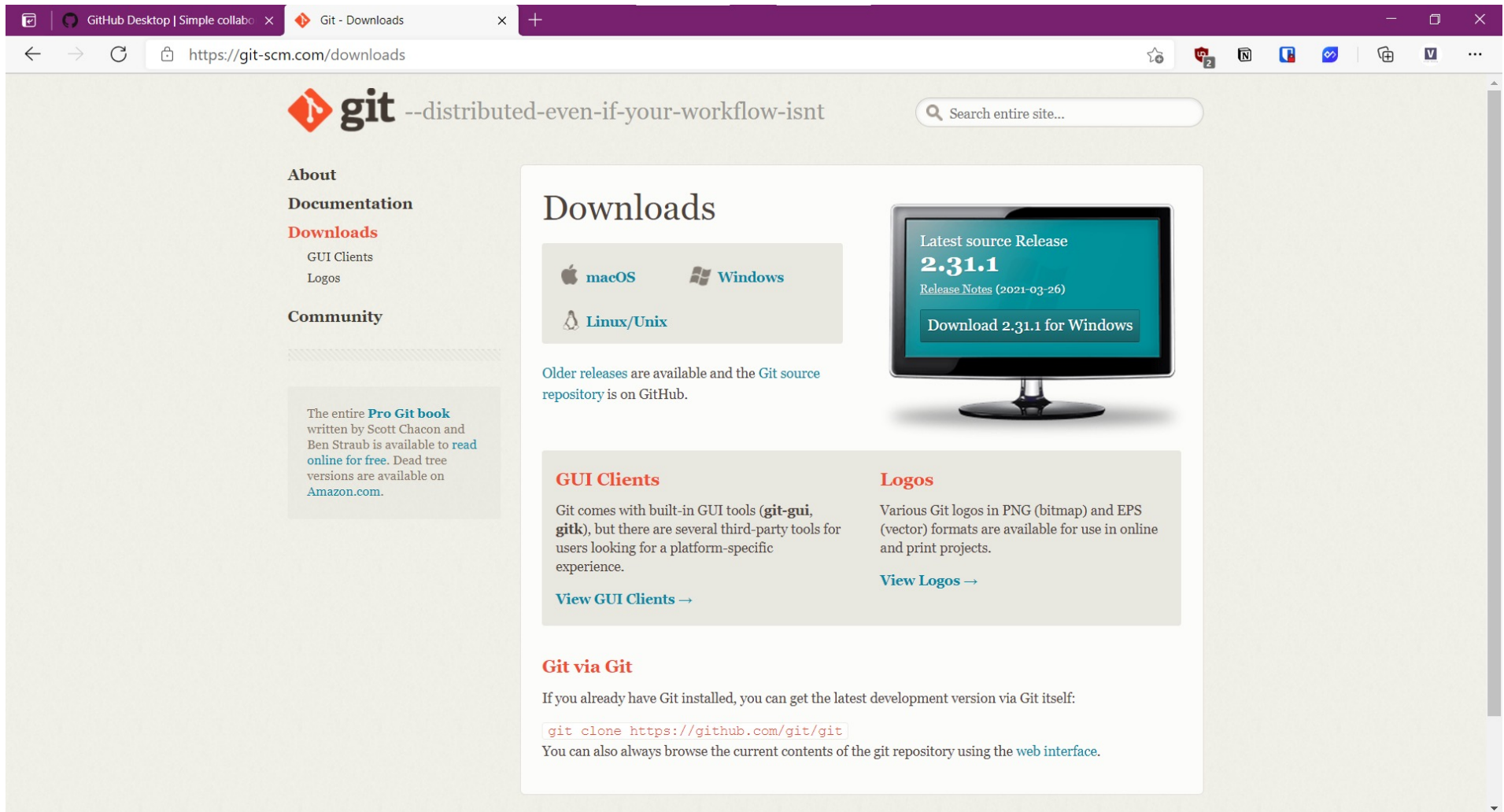
} branch 1's version

} branch 2's version

- Find all such sections, and edit them to the proper state (whichever of the two versions is newer / better / more correct).

Interaction with remote repo

- **Push** your local changes to the remote repo.
- **Pull** from remote repo to get most recent changes.
 - (fix conflicts if necessary, add/commit them to your local repo)
- To fetch the most recent updates from the remote repo into your local repo, and put them into your working directory:
 - `git pull origin master`
- To put your changes from your local repo in the remote repo:
 - `git push origin master`



The screenshot shows a web browser window with the URL `https://git-scm.com/downloads`. The page features the Git logo and the tagline "--distributed-even-if-your-workflow-isnt". A left sidebar contains links for "About", "Documentation", "Downloads" (highlighted), and "Community". The "Downloads" section includes links for "GUI Clients" and "Logos". The main content area is titled "Downloads" and includes a search bar. It features a section for the "Latest source Release 2.31.1" with a "Download 2.31.1 for Windows" button. Below this, there are sections for "GUI Clients" and "Logos". The "GUI Clients" section mentions built-in tools like `git-gui` and `gitk`, and provides a link to "View GUI Clients". The "Logos" section mentions various Git logos in PNG and EPS formats and provides a link to "View Logos". At the bottom, there is a section titled "Git via Git" which provides a command to clone the Git repository: `git clone https://github.com/git/git` and mentions the "web interface".

git --distributed-even-if-your-workflow-isnt

Search entire site...

About

Documentation

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](#).

Downloads

Older releases are available and the Git source repository is on GitHub.

Latest source Release
2.31.1
[Release Notes \(2021-03-26\)](#)
[Download 2.31.1 for Windows](#)

GUI Clients

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

[View GUI Clients →](#)

Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

[View Logos →](#)

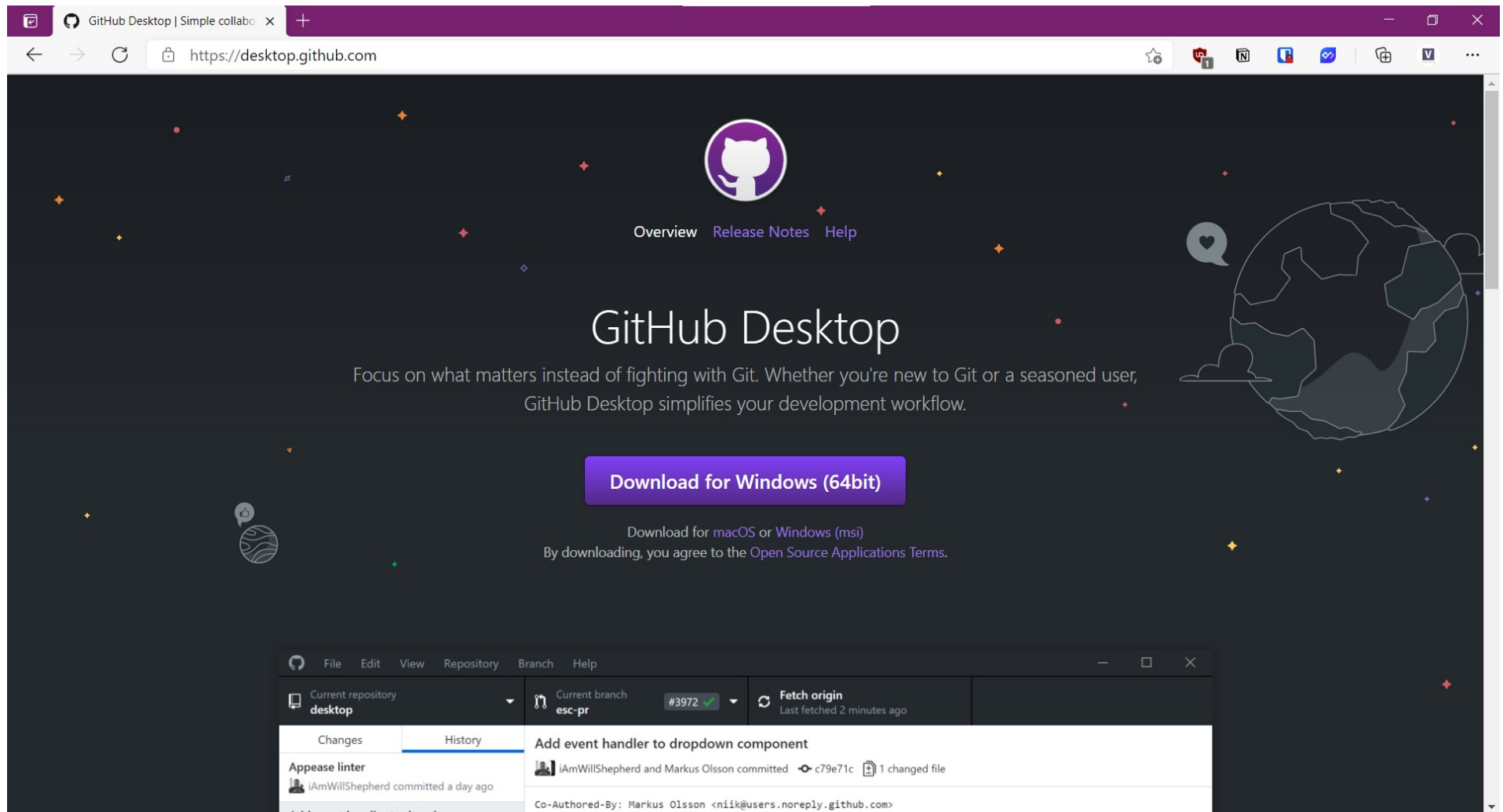
Git via Git

If you already have Git installed, you can get the latest development version via Git itself:

```
git clone https://github.com/git/git
```

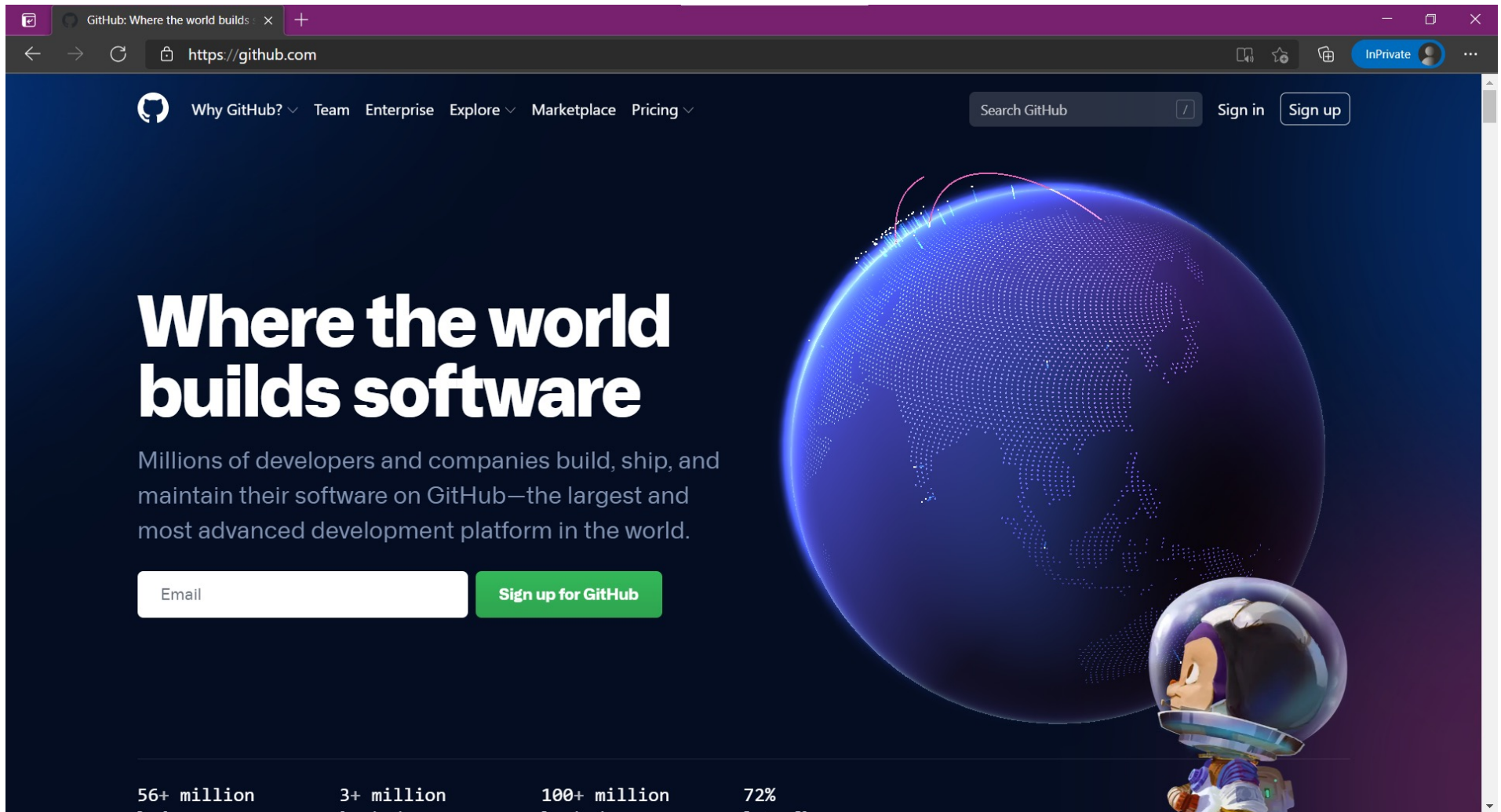
You can also always browse the current contents of the git repository using the [web interface](#).

Git client



- [GitHub.com](https://github.com) is a site for online storage of Git repositories.
 - You can create a **remote repo** there and push code to it.
 - Many open source projects use it, such as the Linux kernel.
 - You can get free space for open source projects, or you can pay for private projects.
- *Question:* Do I always have to use GitHub to use Git?
 - *Answer:* No! You can use Git locally for your own purposes.
 - Or you or someone else could set up a server to share files.
 - Or you could share a repo with users on the same file system, as long as everyone has the needed file permissions).

Create GitHub account



The screenshot shows the GitHub homepage in a web browser. The browser's address bar displays "https://github.com". The page features a dark blue background with a large, glowing blue globe on the right side. On the left, the text "Where the world builds software" is prominently displayed in white. Below this, a smaller line of text reads: "Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world." A white input field labeled "Email" and a green "Sign up for GitHub" button are positioned below the text. At the bottom of the page, four statistics are listed: "56+ million", "3+ million", "100+ million", and "72%". The GitHub logo is visible in the top left corner of the page, and navigation links for "Why GitHub?", "Team", "Enterprise", "Explore", "Marketplace", and "Pricing" are in the top center. A search bar and "Sign in" / "Sign up" buttons are in the top right.

GitHub: Where the world builds : x +

← → ↻ 🔒 https://github.com

🔍 InPrivate

Why GitHub? Team Enterprise Explore Marketplace Pricing

Search GitHub / Sign in Sign up

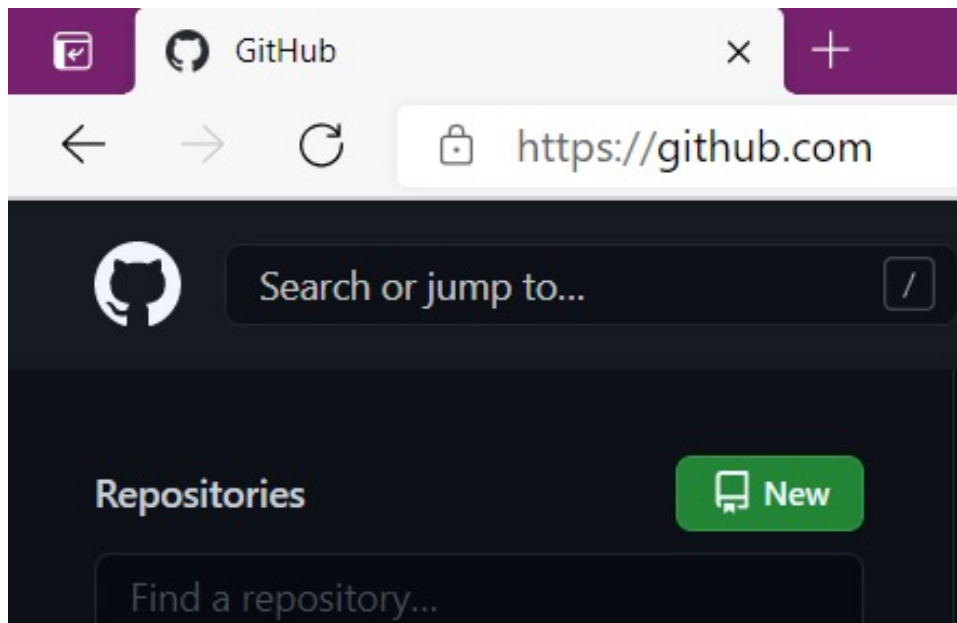
Where the world builds software

Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.

Email [Sign up for GitHub](#)

56+ million 3+ million 100+ million 72%

Create a new repository




Pull requests Issues Marketplace Explore

Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)


Owner * Repository name *

 vinhhn3 /

Great repository names are short and memorable. Need inspiration? How about [ideal-spoon](#)?

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

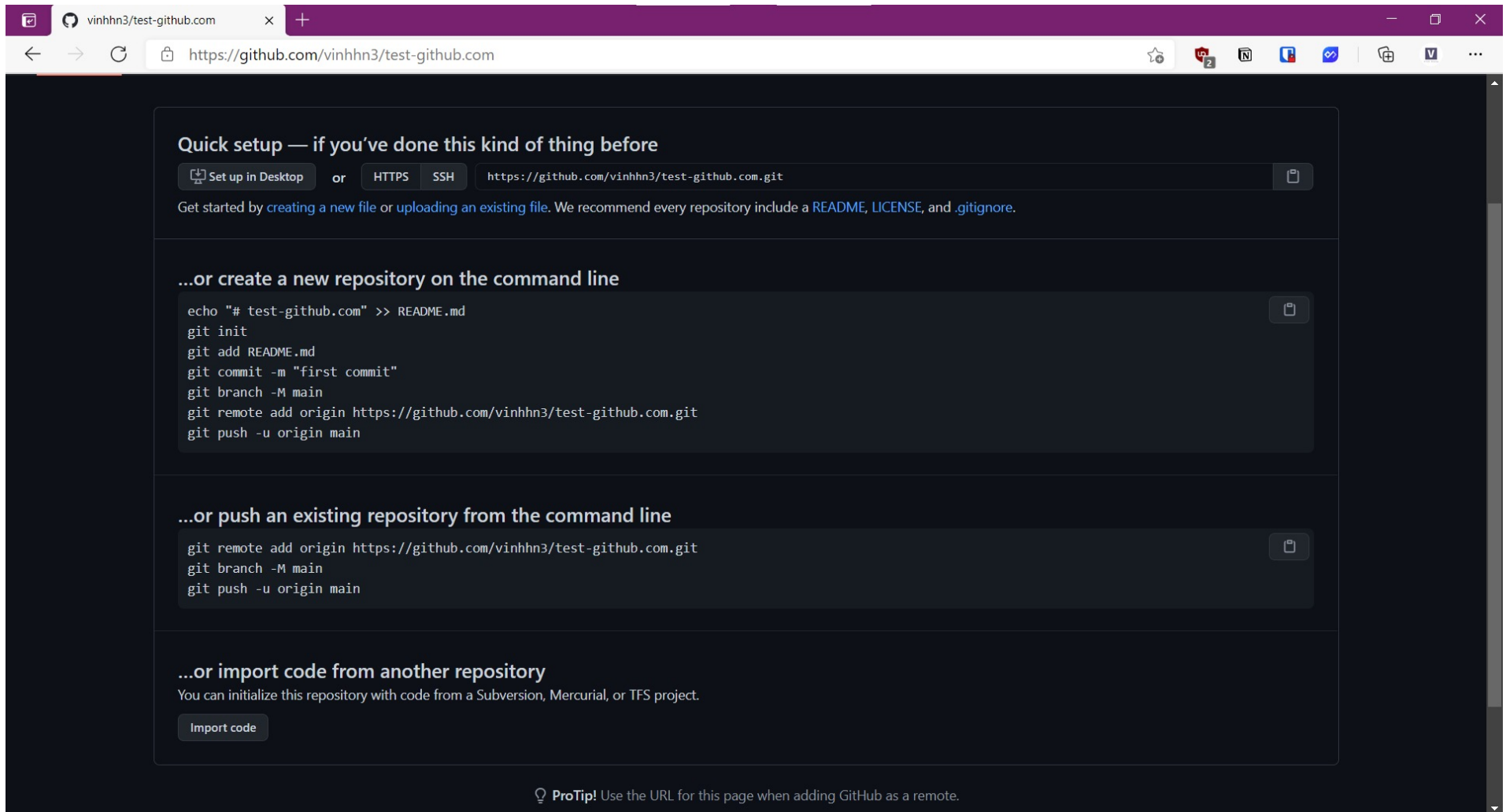
Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

Create a new repository



The screenshot shows the GitHub 'Create a new repository' page in a web browser. The browser's address bar displays 'https://github.com/vinhhn3/test-github.com'. The page content is organized into several sections with a dark background and light text. The first section, 'Quick setup — if you've done this kind of thing before', offers three options: 'Set up in Desktop', 'HTTPS', and 'SSH'. The 'SSH' option is selected, showing the URL 'https://github.com/vinhhn3/test-github.com.git'. Below this, a note states: 'Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).' The second section, '...or create a new repository on the command line', contains a code block with the following commands:



```
echo "# test-github.com" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/vinhhn3/test-github.com.git
git push -u origin main
```

 The third section, '...or push an existing repository from the command line', contains a code block with the following commands:

```
git remote add origin https://github.com/vinhhn3/test-github.com.git
git branch -M main
git push -u origin main
```

 The fourth section, '...or import code from another repository', includes a sub-header and a paragraph: 'You can initialize this repository with code from a Subversion, Mercurial, or TFS project.' Below this paragraph is a button labeled 'Import code'. At the bottom of the page, a 'ProTip!' icon is followed by the text: 'Use the URL for this page when adding GitHub as a remote.'

Quick setup — if you've done this kind of thing before

 Set up in Desktop or **HTTPS** **SSH** `https://github.com/vinhhn3/test-github.com.git` 

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# test-github.com" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/vinhhn3/test-github.com.git
git push -u origin main
```


...or push an existing repository from the command line

```
git remote add origin https://github.com/vinhhn3/test-github.com.git
git branch -M main
git push -u origin main
```

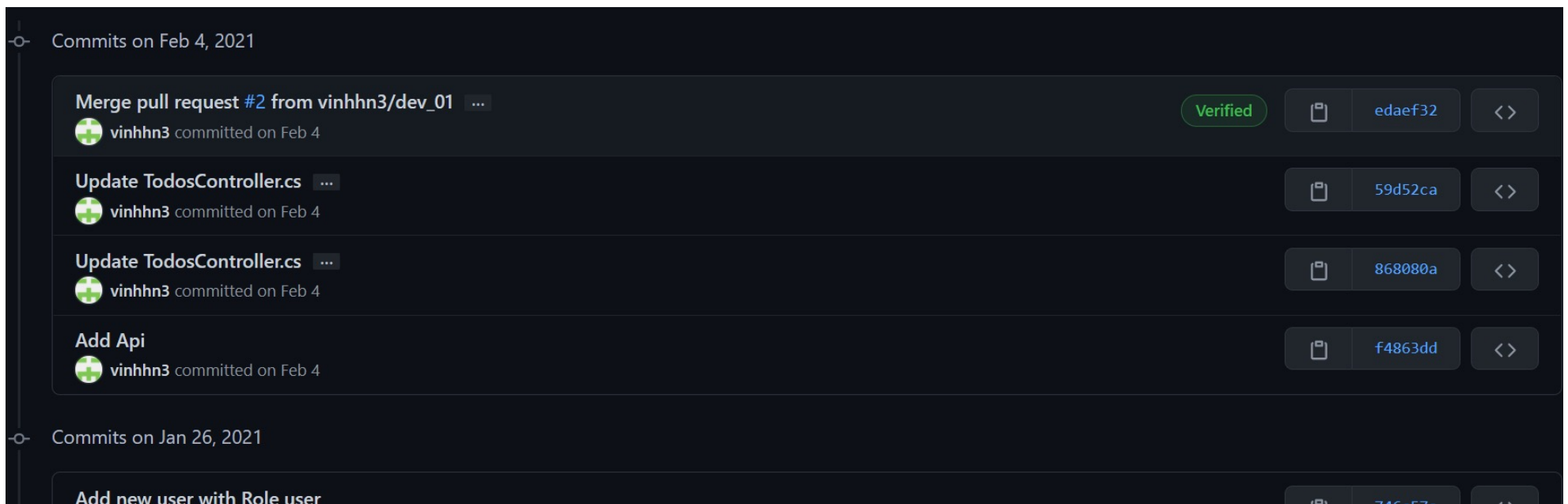
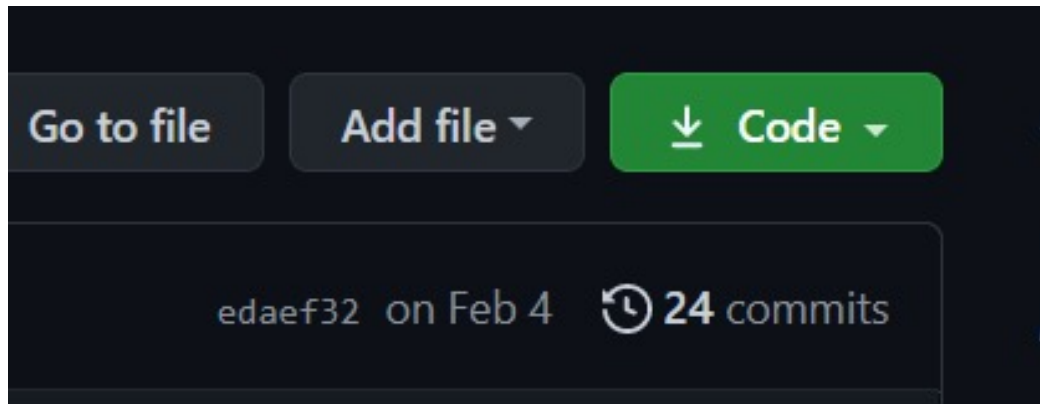
...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

 **ProTip!** Use the URL for this page when adding GitHub as a remote.

Commit



Showing 15 changed files with 608 additions and 107 deletions. Unified Split

6 GCD0704.AppDev/App_Start/RouteConfig.cs

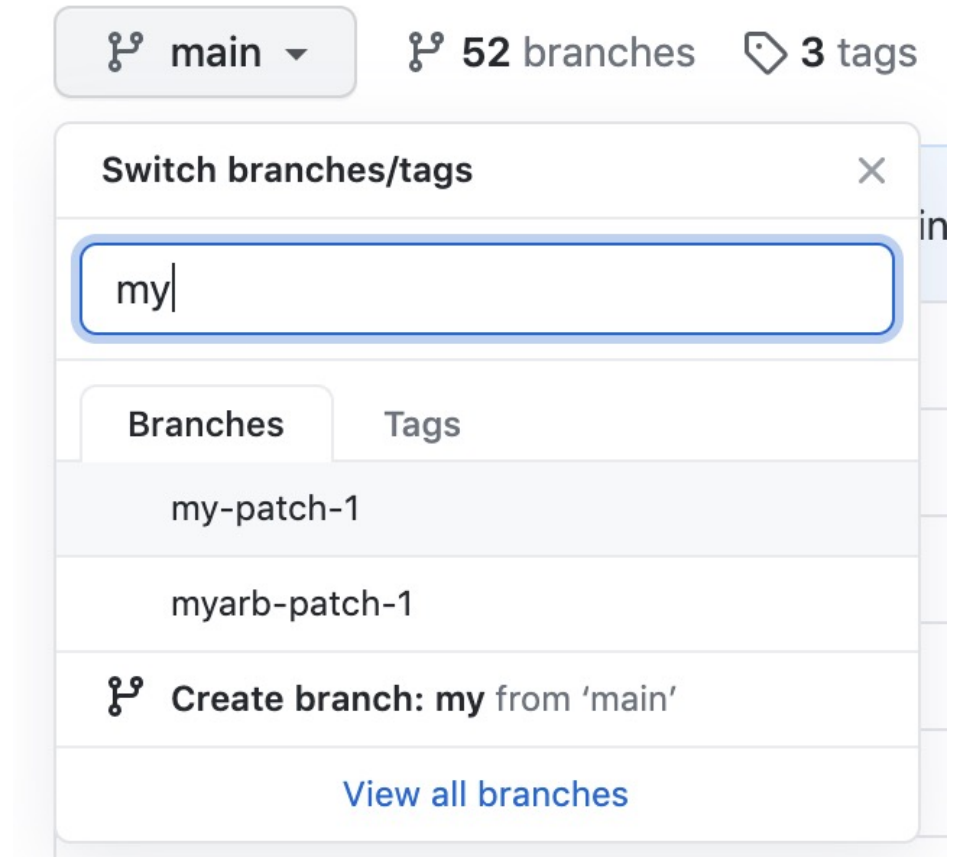
```
@@ -1,7 +1,4 @@
1 - using System;
2 - using System.Collections.Generic;
3 - using System.Linq;
4 - using System.Web;
5   using System.Web.Mvc;
6   using System.Web.Routing;
7
@@ -18,6 +15,7 @@ public static void RegisterRoutes(RouteCollection routes)
18         url: "{controller}/{action}/{id}",
19         defaults: new { controller = "Home",
20             action = "Index", id = UrlParameter.Optional }
21     );
22 }
23 }
```

21 GCD0704.AppDev/App_Start/WebApiConfig.cs

```
@@ -0,0 +1,21 @@
1 + using System;
2 + using System.Collections.Generic;
```

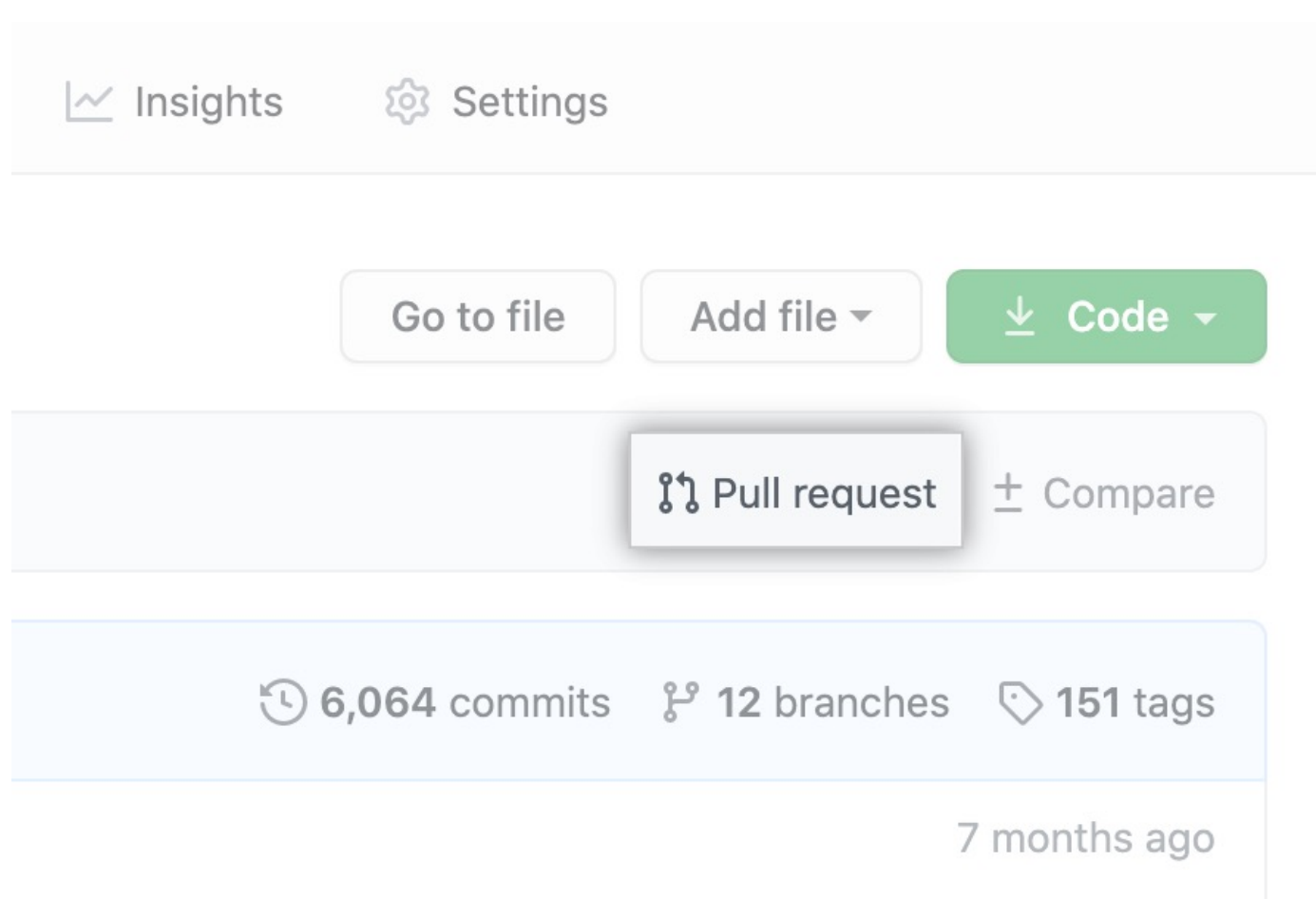
Create a Pull Request

- On GitHub, navigate to the main page of the repository.
- In the "Branch" menu, choose the branch that contains your commits.



Create a Pull Request

- Above the list of files, click Pull request.




Create a Pull Request

- Use the base branch dropdown menu to select the branch you'd like to merge your changes into, then use the compare branch drop-down menu to choose the topic branch you made your changes in.

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).



base: main ▼

 ←


compare: my-patch-1 ▼

✓ **Able to merge.** These branches can be automatically merged.

Create a Pull Request

- Type a title and description for your pull request.

Please review the [guidelines for contributing](#) to this repository.



Add CONTRIBUTING.md

Write

Preview

AA ▾ B i “ <> 🔗 ☰ ☷ ✓ ↶ @ 📌

Let's add a contributing file, so we can work better, together!

Attach files by dragging & dropping, [selecting them](#), or pasting from the clipboard.

Labels

None yet

Milestones

No milestones

Assignees

No one assigned

Create a Pull Request

- To create a pull request that is ready for review, click Create Pull Request. To create a draft pull request, use the drop-down and select Create Draft Pull Request, then click Draft Pull Request. For more information about draft pull requests, see "About pull requests."

