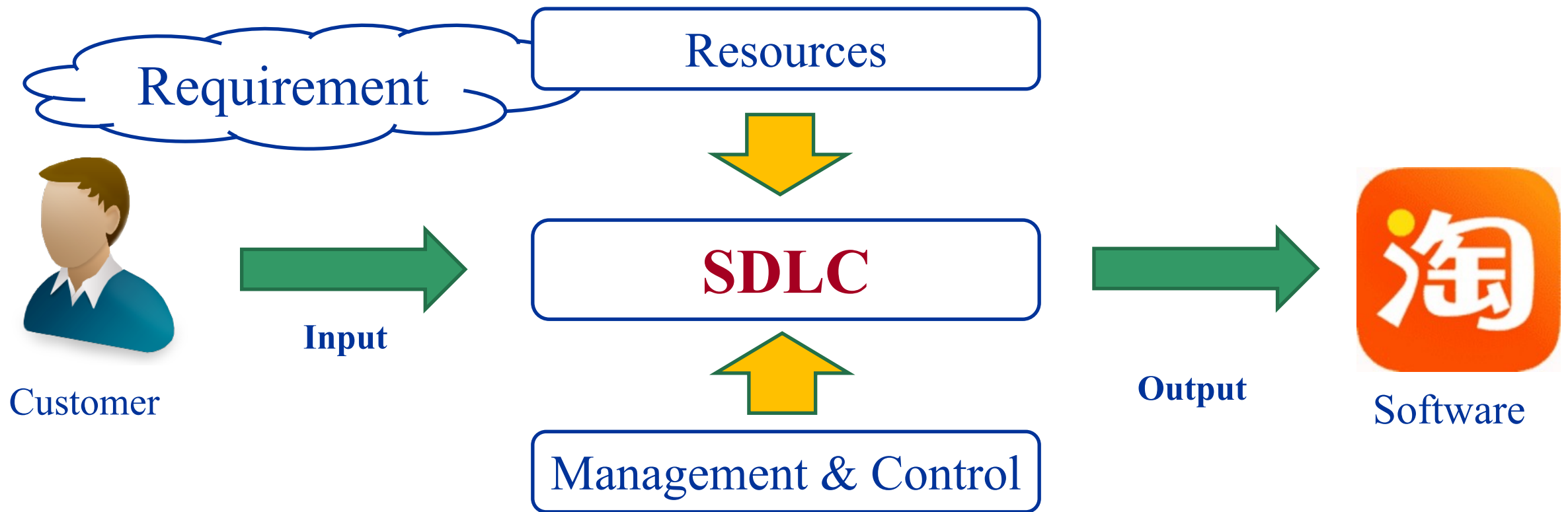
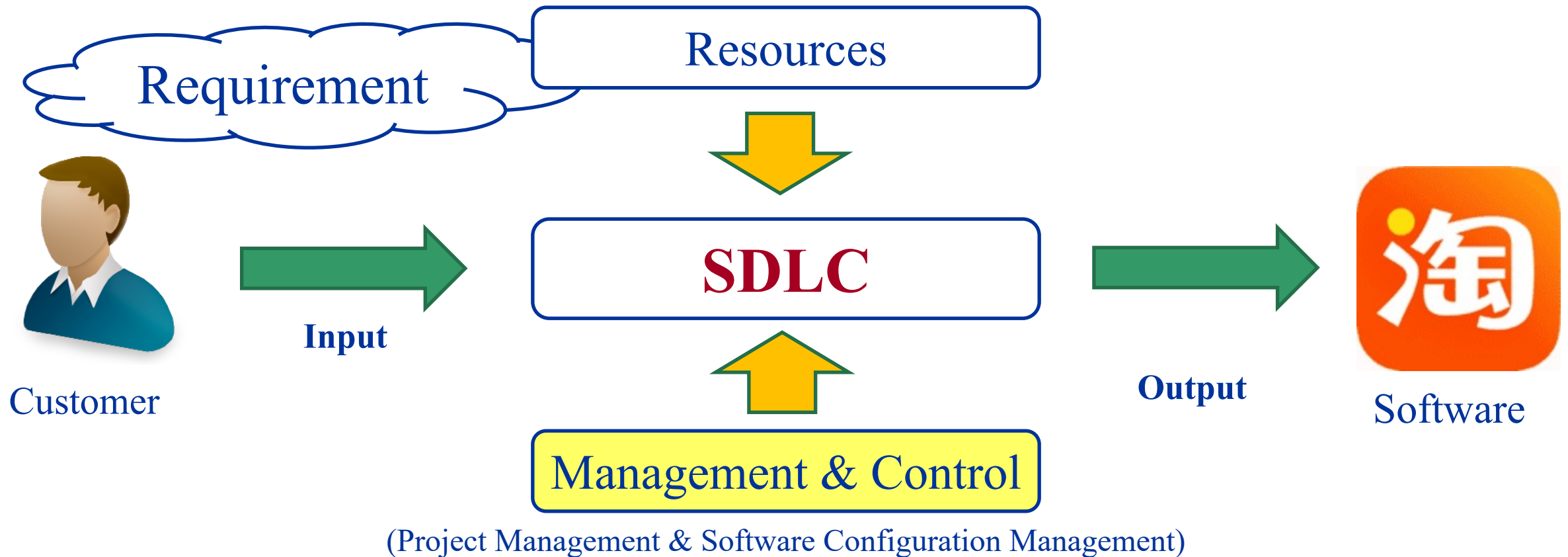


Software Development Life Cycle (SDLC)



Software Development Life Cycle (SDLC)



Lecture 3: Software Configuration Management

Yutian Tang
SIST@ShanghaiTech

Software Configuration Management (SCM)

- Software Configuration Management is the task of tracking and controlling changes in the software, including revision control and the establishment of baselines.
- Outline
 - Software Versioning
 - Version Control
 - Git & Github

Software Versioning

- A normal version number MUST take the form X.Y.Z where X, Y, and Z are non-negative integers, and MUST NOT contain leading zeroes.
- X is the major version, Y is the minor version, and Z is the patch version (Format: X. Y. Z).
- For instance: 1.9.0 \rightarrow 1.10.0 \rightarrow 1.11.0.
- Major version zero (0.y.z) is for **initial development**.
- Version 1.0.0 defines the **public API**.

Software Versioning (2)

- Patch version Z (x.y.Z) MUST be increased if only backwards compatible **bug fixes** are introduced.
- Minor version Y (x.Y.z) MUST be increased if **new**, backwards compatible **functionality** is introduced to the public API.
- Major version X (X.y.z) MUST be increased if **any backwards incompatible changes** are introduced to the public API.

Software Versioning-Pre-release Version

- A pre-release version MAY be denoted by appending a hyphen and a series of dot separated identifiers immediately following the patch version.
 - alpha/a: e.g., 1.0.0-alpha, 1.0.0-a, 1.0.0-a.1;
 - beta/b: e.g., 1.0.0-beta;
 - rc: release candidate;
- A pre-release version indicates that the version is unstable and might not satisfy the intended compatibility requirements as denoted by its associated normal version

Software Versioning-Pre-release Version (2)

- Pre-release Version:
- alpha: programs in an early stage;
- beta: programs become mature but are not yet ready for release;
- rc (release candidate): programs are soon to be released;
- Relation: $\text{alpha} < \text{beta} < \text{rc} < \text{release}$ (without pre-release version code)
- E.g.,: $1.0.0\text{-alpha} < 1.0.0\text{-beta} < 1.0.0\text{-rc} < 1.0.0$

Software Versioning - Summary

- Version: X. Y. Z
- Major. Minor. Patch
 - Major: break APIs;
 - Minor: not break APIs;
 - Patch: bug fixing;
- Pre-release Version: Major. Minor. Patch-Prerelease
 - alpha, beta, rc
 - E.g., 1.0.0-alpha, 1.0.0-beta
- Pre-release Version (not stable)

Version Control

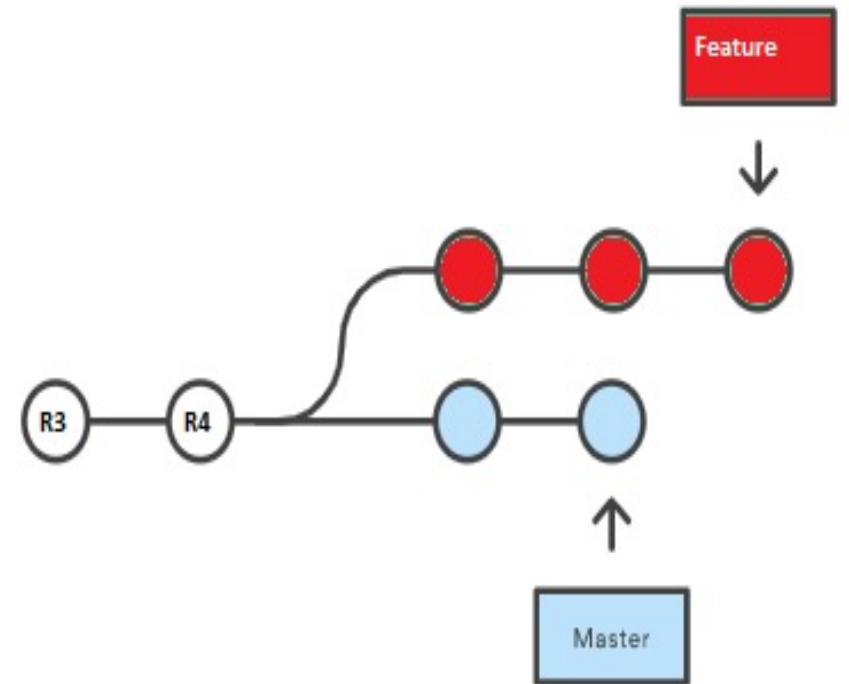
Return to Zero



EEWeb.com

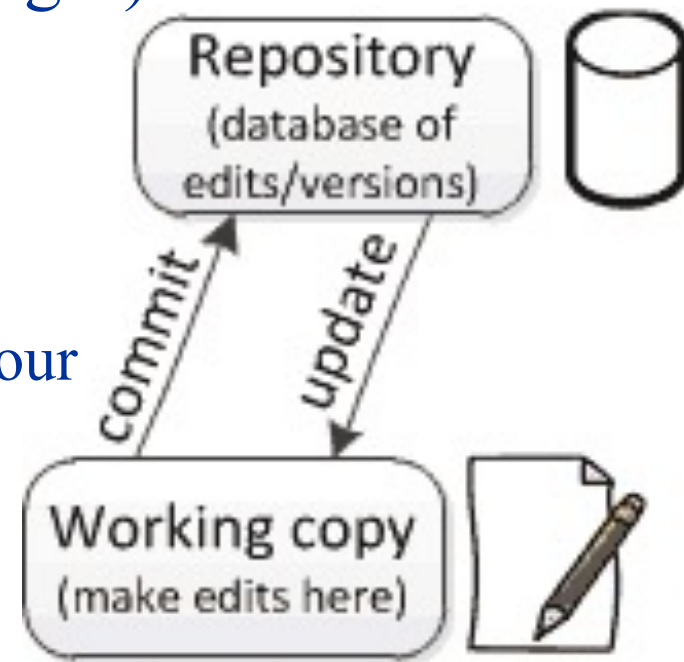
Version Control (2)

- Purpose
 - Manage change in time (history) and space (branches)
- Benefits
 - Ease collaboration
 - Allow branching and continuous integration
 - Track ownership and change history



Repositories and Working Copies

- Version control uses *a repository* (a database of changes) and *a working copy* where you do your work.
- Working copy
 - Your personal copy of all the files in the project.
 - You make arbitrary edits to this copy, without affecting your teammates.
 - You can **commit** your changes to a repository.
- Repository
 - Database of all the edits to, and historical versions of your project.
 - You can **update** your working copy to incorporate any new edits.



Git

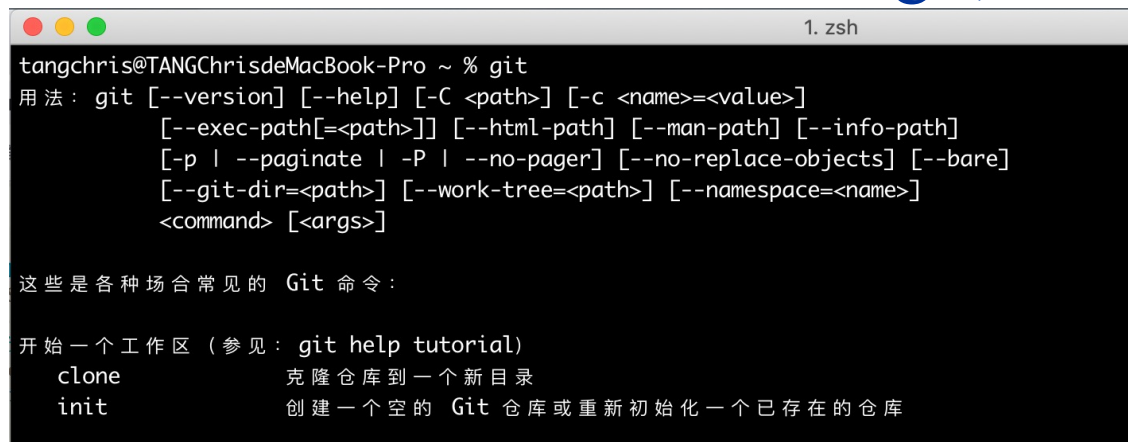


- <https://git-scm.com/>
- Free and open source distributed version control system
- Linus Torvalds (Linux)
- Linux Kernel (1991-2002)
 - changes to the software were passed to Linus as patches and archived files.
- Linux Kernel (2002-2005)
 - BitKeeper (for free of charge).
 - In 2005, the relationship between Linux and BitKeeper broken down
- Linux Kernel (2005+): Git



Installing Git

- Download Git from official website: <https://git-scm.com/>;
- Add Git to System PATH;
- Test Git from Terminal with command: git;



```
1. zsh
tangchris@TANGChrisdeMacBook-Pro ~ % git
用法: git [--version] [--help] [-C <path>] [-c <name>=<value>]
      [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
      [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
      [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
      <command> [<args>]

这些是各种场合常见的 Git 命令:

开始一个工作区 (参见: git help tutorial)
  clone          克隆仓库到一个新目录
  init           创建一个空的 Git 仓库或重新初始化一个已存在的仓库
```



- Reference: <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>.

Github

- What is Github?
 - <https://github.com/>
 - Based on Git;
 - Git with a web interface;
 - Git hosting;
 - Issue and request tracking system;
 - Documentation system;



Git vs Github

| |  |  |
|---|--|---|
| 1 | It is a software | It is a service |
| 2 | It is installed locally on the system | It is hosted on web |
| 3 | It is a command line tool | It provides a graphical interface |
| 4 | It is a version control system that lets you manage and keep track of your source code history | It is a cloud-based hosting service that let you manage Git repositories |
| 5 | It provides version control functions | It provides version control functions as well as its own features |

Create a Repository on Github

- <https://github.com/new>;

The name of the repository

The description of the repository

Public or private repository?

Other configurations

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 *

 csytang-se / cs132-demo 

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

Description (optional)

This is a demo



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:

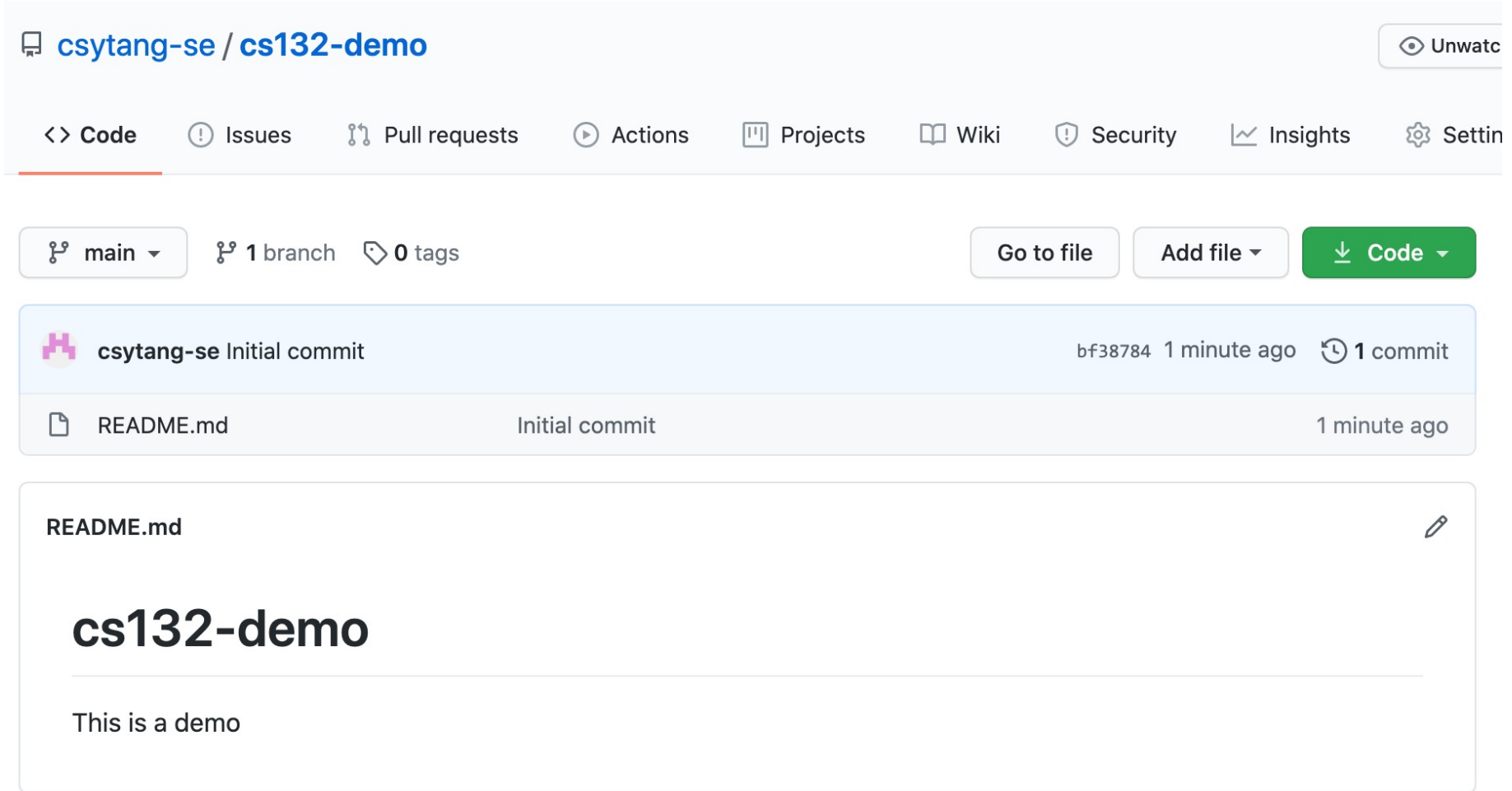
Skip this step if you're importing an existing repository.



Add a README file

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

Create a Repository on Github (2)



The screenshot shows the GitHub interface for a repository named 'csytang-se / cs132-demo'. At the top, there's a navigation bar with links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below this, a summary bar indicates the current branch is 'main', there is 1 branch and 0 tags, and buttons for 'Go to file', 'Add file', and 'Code'. The main content area shows a commit history with one commit: 'csytang-se Initial commit' by user 'csytang-se', committed 1 minute ago. Below the commit, a file named 'README.md' is listed as part of the initial commit. The 'README.md' content is displayed in a large box, showing the title 'cs132-demo' and the text 'This is a demo'.

Clone the Repository (1)

- To clone a Git repository, you need:
 - `git clone <url>`
- eg.,:
 - `git clone https://github.com/csytang-se/cs132-demo.git` (demo repo)
 - `git clone https://github.com/torvalds/linux.git` (linux)
 - `git clone https://github.com/tensorflow/tensorflow.git` (Tensorflow framework)

Clone the Repository (2)

The screenshot shows the GitHub interface for the repository `csytang-se / cs132-demo`. The repository is on the `main` branch, has 1 branch and 0 tags. The `Code` button is highlighted in green. The dropdown menu is open, showing options to clone the repository using HTTPS, SSH, or GitHub CLI (marked as 'New'). The HTTPS URL `https://github.com/csytang-se/cs132-de` is highlighted in a blue box. Below the URL, it says 'Use Git or checkout with SVN using the web URL.' There are also options to 'Open with GitHub Desktop' and 'Download ZIP'.

csytang-se / **cs132-demo** Unwatch

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

[main](#) [1 branch](#) [0 tags](#) [Go to file](#) [Add file](#) [Code](#)

csytang-se Initial commit

[README.md](#) Initial commit

README.md

cs132-demo

This is a demo

Clone ?

[HTTPS](#) [SSH](#) [GitHub CLI](#) [New](#)

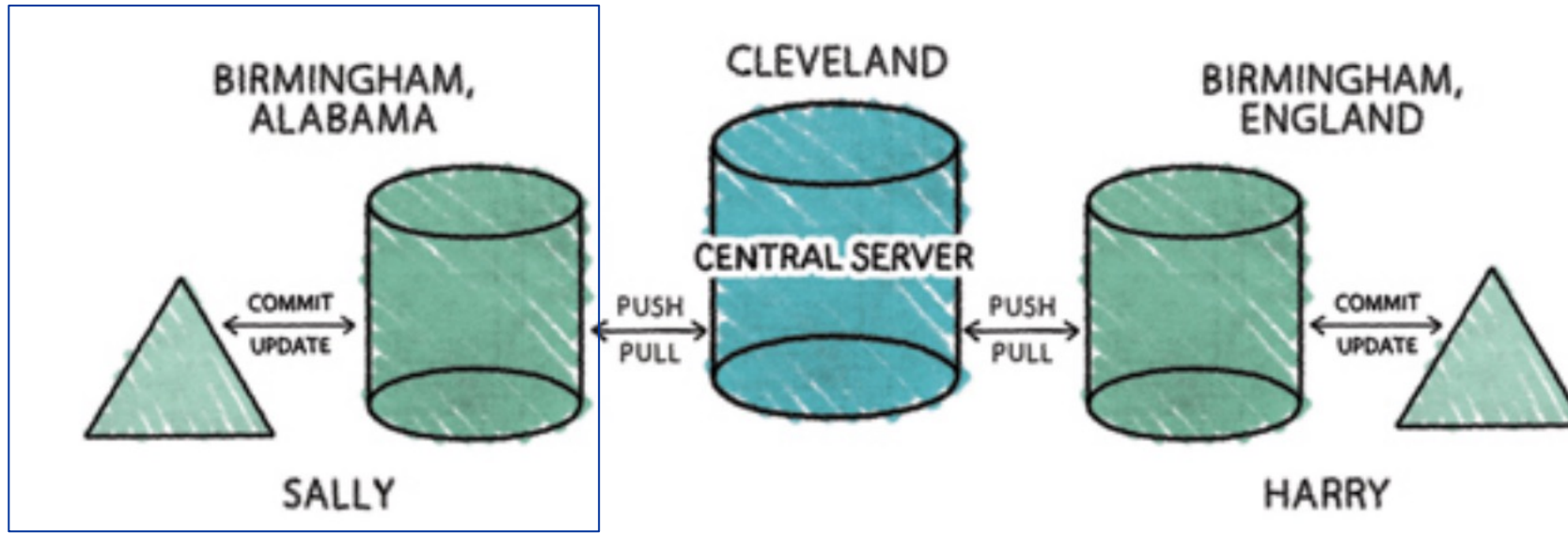
`https://github.com/csytang-se/cs132-de` [Copy](#)

Use Git or checkout with SVN using the web URL.

[Open with GitHub Desktop](#)

[Download ZIP](#)

Clone the Repository (3)



Working Copy 1 Local Repository 1

Remote Repository (on Github)

gitignore

- A **gitignore** file specifies intentionally untracked files that Git should ignore.
- Each line in a **gitignore** file specifies a pattern.
- STEP 1. Create a new file named ``.gitignore`` in your local repository;
- STEP 2. Specify the file patterns in `.gitignore` file
 - `*.class` → ignore all class files
 - `build/` → ignore all files in build folder

gitignore

```
# Eclipse settings  
.project  
.classpath  
.settings/  
  
# Maven output folder  
target/  
  
# Vim swap files  
*.swp
```

← Comments

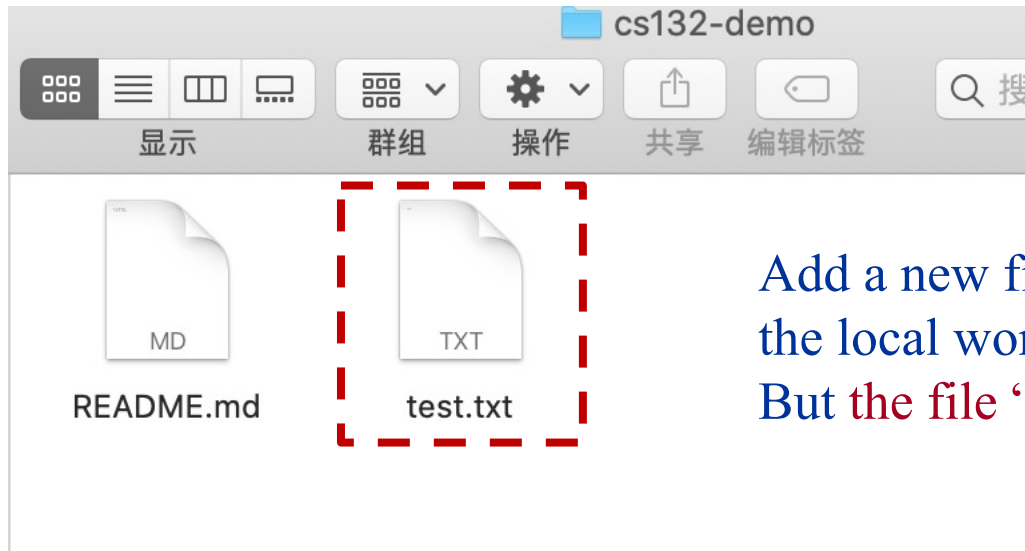
← Match by name (both file and folder)

← Match only folder

← Match by pattern

Add Files (1)

- STEP 1: Create a file to the working directory;



Add a new file named “test.txt” to the local working directory.

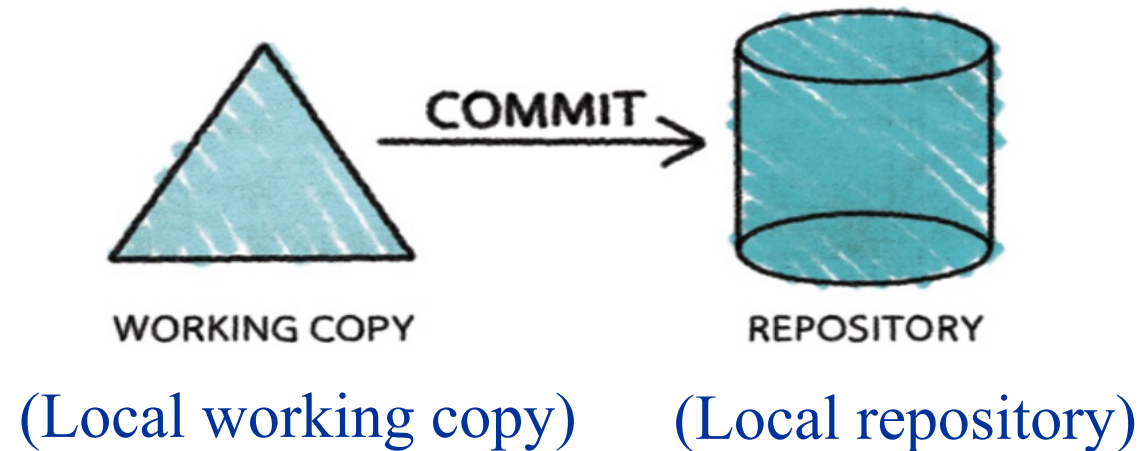
But the file “test.txt” is not under version control by git.

Add Files (2)

- STEP 2: Add a file to version control with **add** command;
 - `git add <file>`
- eg.,
 - `cd cs132-demo`
 - `git add test.txt`
- For multiple files, use ``git add --all``

Commit

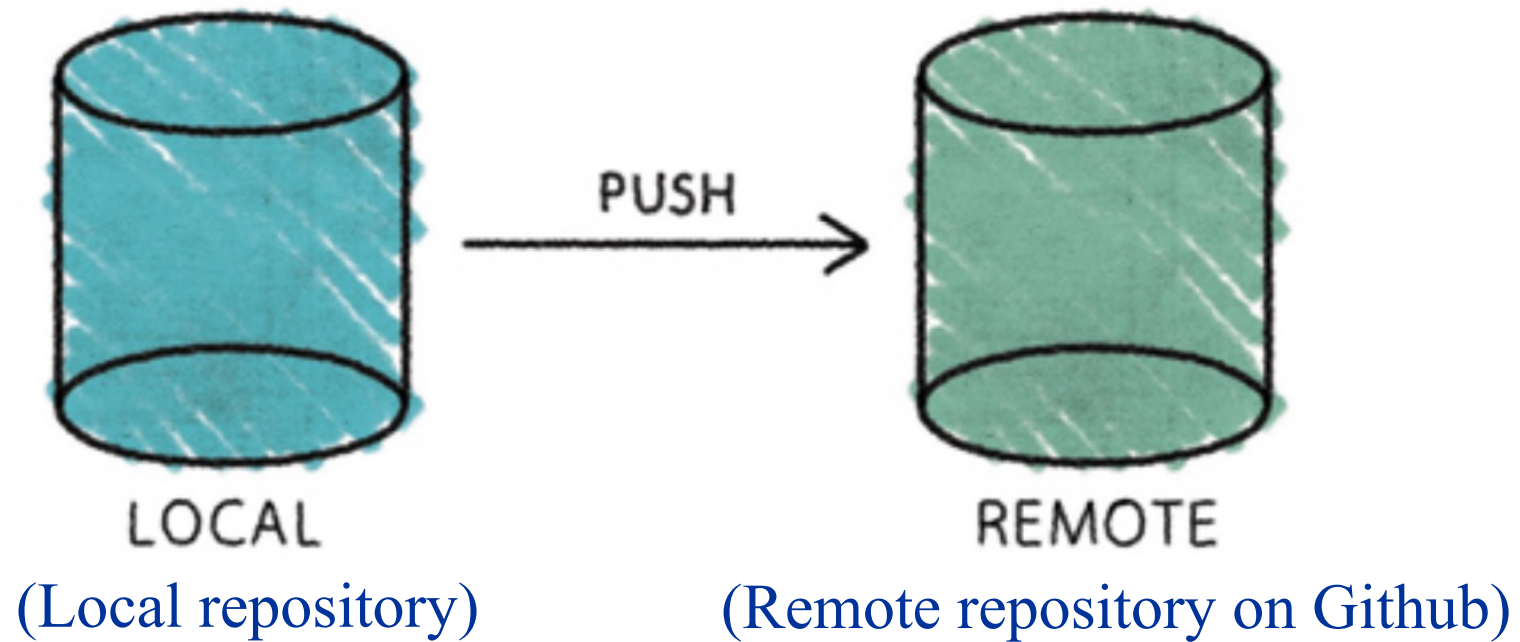
- STEP 3: Apply the modifications in the working copy to the repository as a new changeset.



- Commit changes with: `git commit -m "<message>"`
 - eg: `git commit -m "add test.txt to repo"`

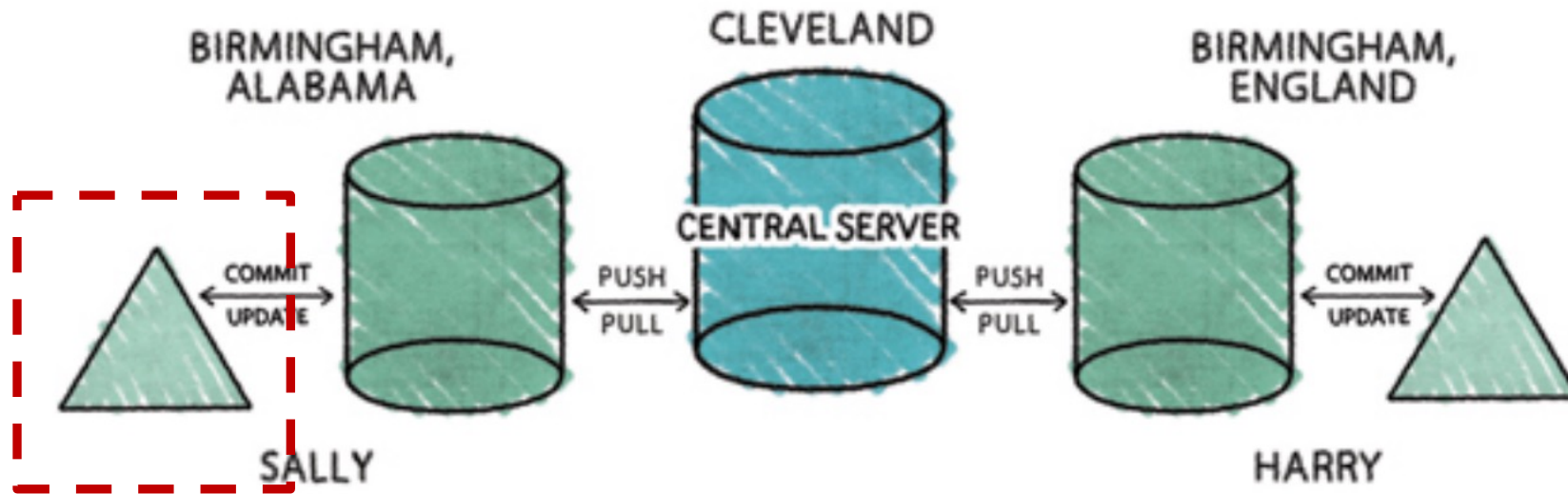
Push

- STEP 4: Copy changesets from a local repository instance to a remote one.
 - `git push`



Recap: Add File

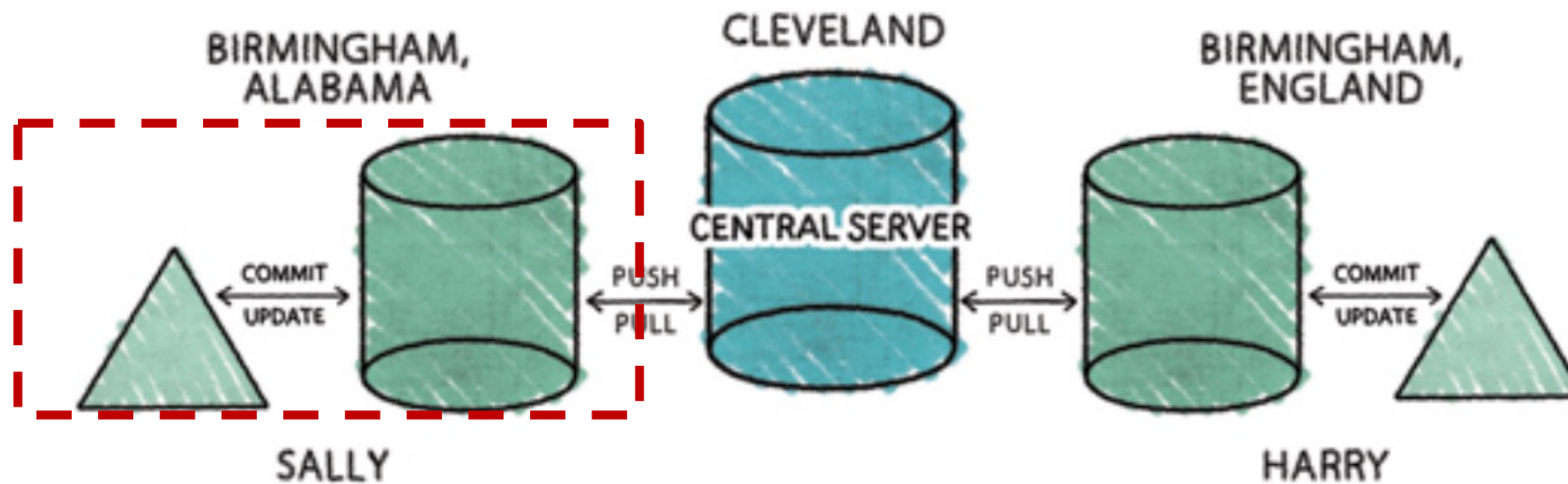
- STEP 1: Create a file to the working directory;
- STEP 2: Add the file to the version control system; (**git add ...**)



(Local working copy)

Recap: Add File (2)

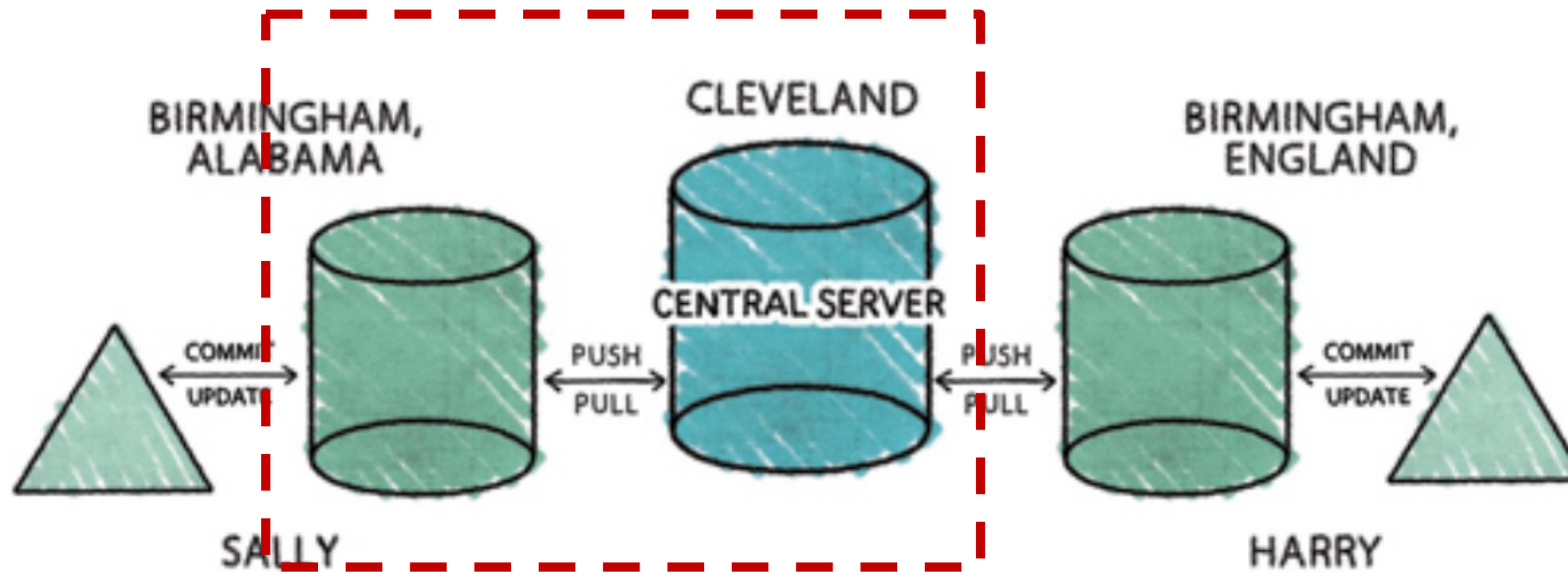
- STEP 3: Commit change to the local repository; (`git commit -m "..."`)



(Local working copy) (Local repository)

Recap: Add File (3)

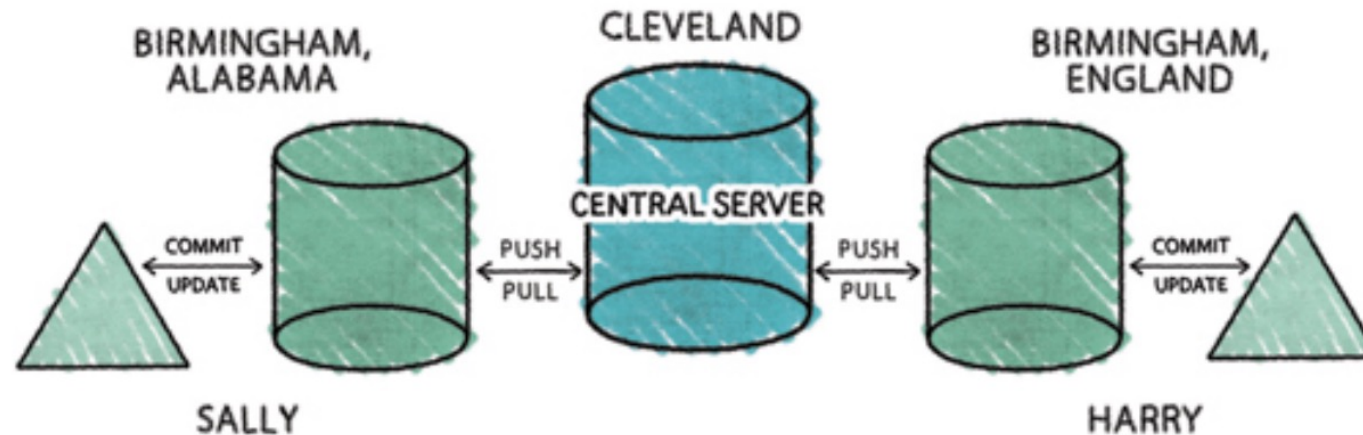
- STEP 4: Update the remote repository on Github (**git push**)



(Local repo) (Remote repo --- Github)

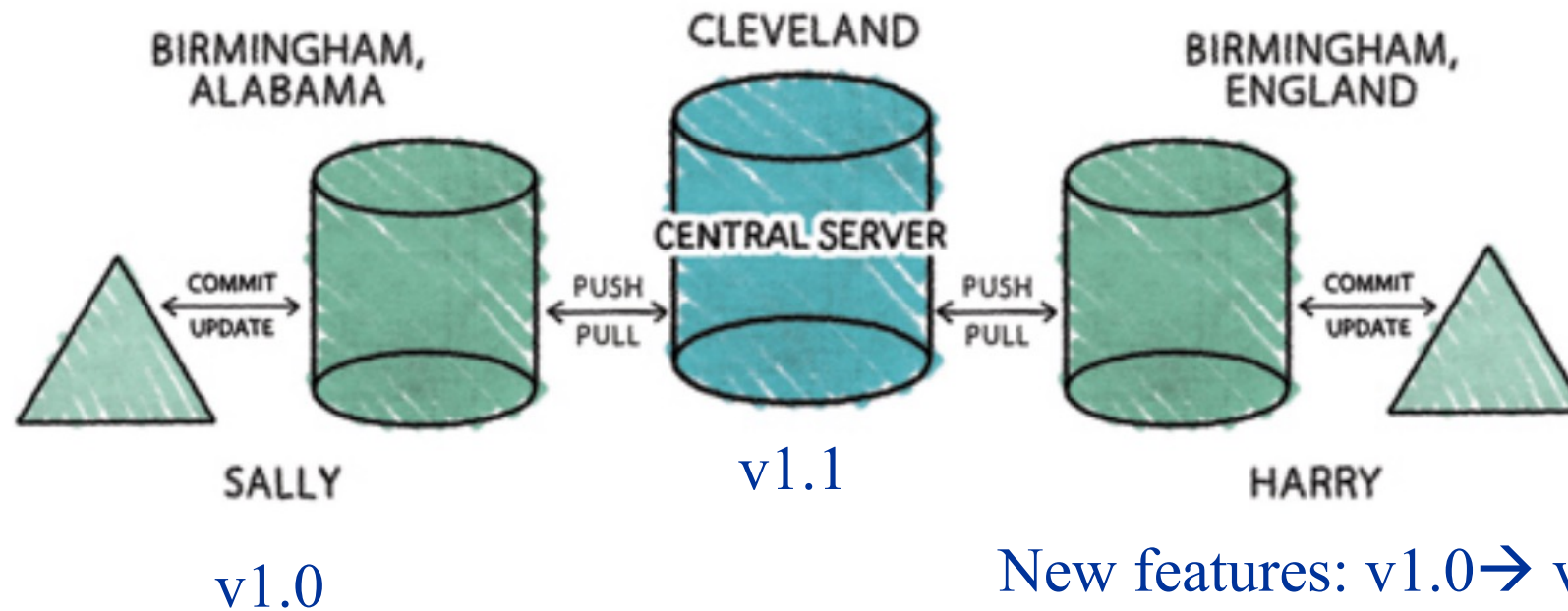
Recap: Add File (4)

- STEP 1: Create a file to the working directory;
- STEP 2: Add the file to version control system; (`git add ...`)
- STEP 3: Commit change to the repository; (`git commit -m “...”`)
- STEP 4: Update the remote repository on Github (`git push`)

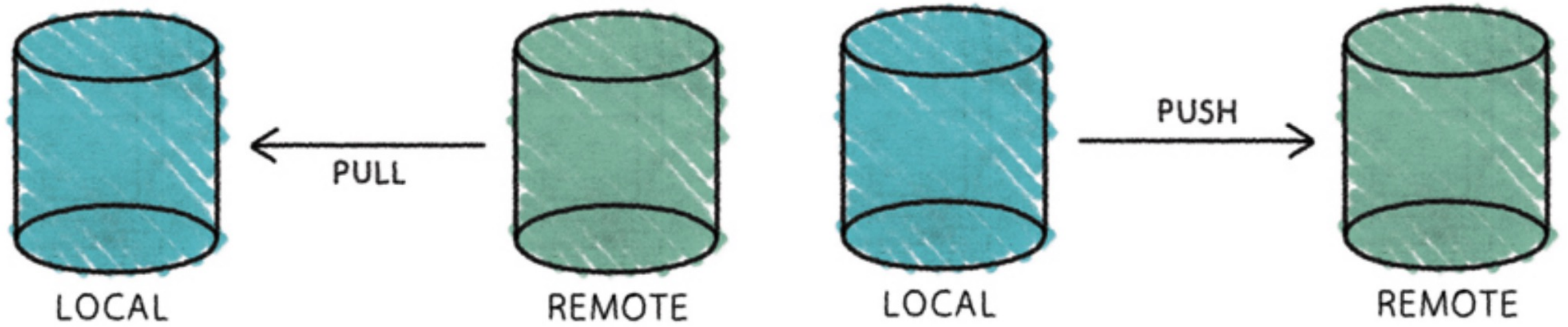


Pull

- Copy changesets from a remote repository instance to a local one.
- `git pull`



Push vs Pull



Delete

- Delete a file or directory;
- `git rm <file>`
- STEP 1: Delete a file or directory (`git rm <file>`)
 - eg., `git rm test.txt`
- STEP 2: Commit the change to the local repository (`git commit -m “<message>”`)
 - eg., `git commit -m “remove test.txt”`
- STEP 3: Update to the remote repository on Github (`git push`)

Edit

- Edit an existing file under version control;
- STEP 1: Edit an existing file;
- STEP 2: Record the changes with git (**git add <file>**);
- STEP 3: Commit to local repository repository (**git commit -m “<message>”**);
 - eg., git commit -m “edit test.txt”
- STEP 4: Update to the remote repository on Github (**git push**);

Move

- Move a file or directory;
- STEP 1: `git mv <file> <dest>`
 - eg. `git mv test.txt src`
- STEP 2: Commit to local repository repository (`git commit -m “<message>”`);
 - eg., `git commit -m “edit test.txt”`
- STEP 3: Update to the remote repository on Github (`git push`);

Rename

- Rename a file under version control;
- STEP 1: `git mv <old-name> <new-name>`
 - eg. `git mv test.txt newname.txt`
- STEP 2: Commit to local repository repository (`git commit -m “<message>”`);
 - eg., `git commit -m “edit test.txt”`
- STEP 3: Update to the remote repository on Github (`git push`);

Status

- List the modifications that have been made to the working copy.
- `git status`

```
tangchris@TANGChrisdeMacBook-Pro cs132-demo % git status
```

位于分支 main

您的分支与上游分支 'origin/main' 一致。

要提交的变更：

(使用 "`git restore --staged <文件>...`" 以取消暂存)

新文件: "edit\347\232\204\345\211\257\346\234\254.txt"

新文件: "edit\347\232\204\345\211\257\346\234\2542.txt"

新文件: "edit\347\232\204\345\211\257\346\234\2543.txt"

新文件: "edit\347\232\204\345\211\257\346\234\2544.txt"

Log

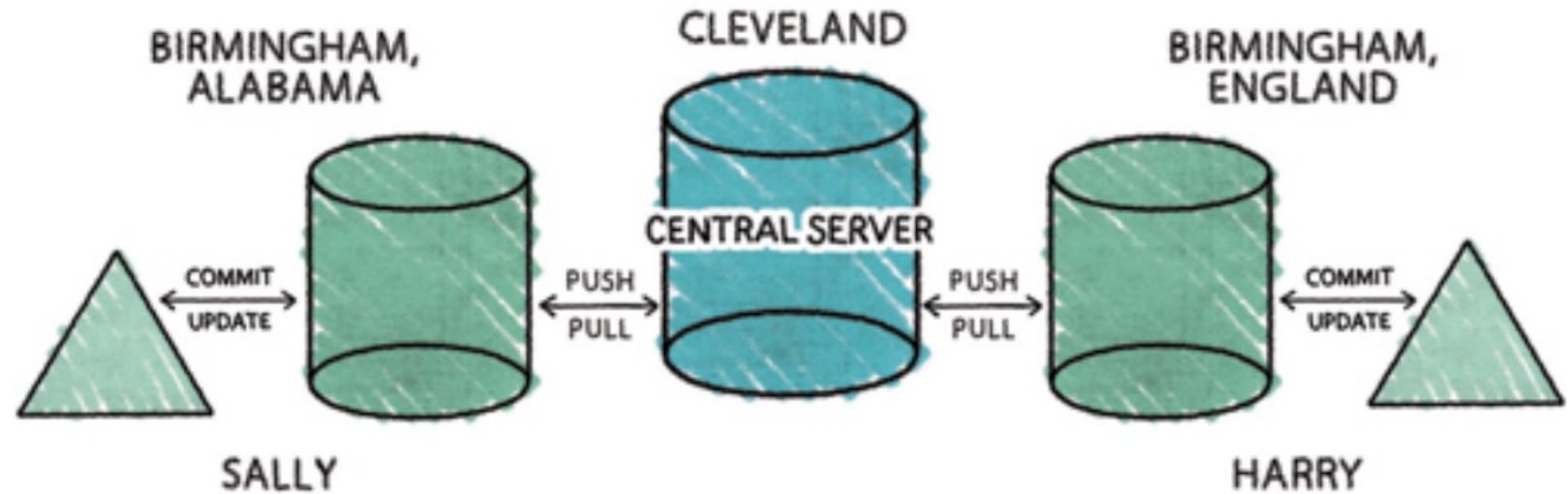
- Show commit logs
- `git log`
- To view recent `n` commit logs:
 - `git log -n`
 - eg., `git log -2` (recent 2 commits)

```
*   commit 9c8e9fd335381fe6a97708f7b3cd1d5acf670d2d
|\  Merge: 8aba87e... 6041ddd...
| | Author: Nick Quaranto <nick@quaran.to>
| | Date:   Sun Jan 25 13:22:03 2009 -0500
| |
| |     Fixing conflict!
| |
*   commit 6041dddac354fff0feec911e75a575082d8addb8
| | Author: Nick Quaranto <nick@quaran.to>
| | Date:   Sun Jan 25 13:10:23 2009 -0500
| |
| |     Changing cutoff default
| |
*   commit 8aba87e2e24744b7d1941e104b35033b9e2dbab5
|/  Author: Nick Quaranto <nick@quaran.to>
|   Date:   Sun Jan 25 13:16:04 2009 -0500
|   |
|   |     Causing a merge on purpose
|   |
*   commit 670e3538533554d0643ca128428997c98eb5d54e
|   Author: Nick Quaranto <nick@quaran.to>
|   Date:   Sun Jan 25 13:04:30 2009 -0500
|   |
|   |     Adding cutoff method to string
```

Diff

- Compare current with a specific commit
- `git diff <commit-id>`
- eg., `git diff`
92154f8dc38d273dc093d88ed98e21e757
5d66f7

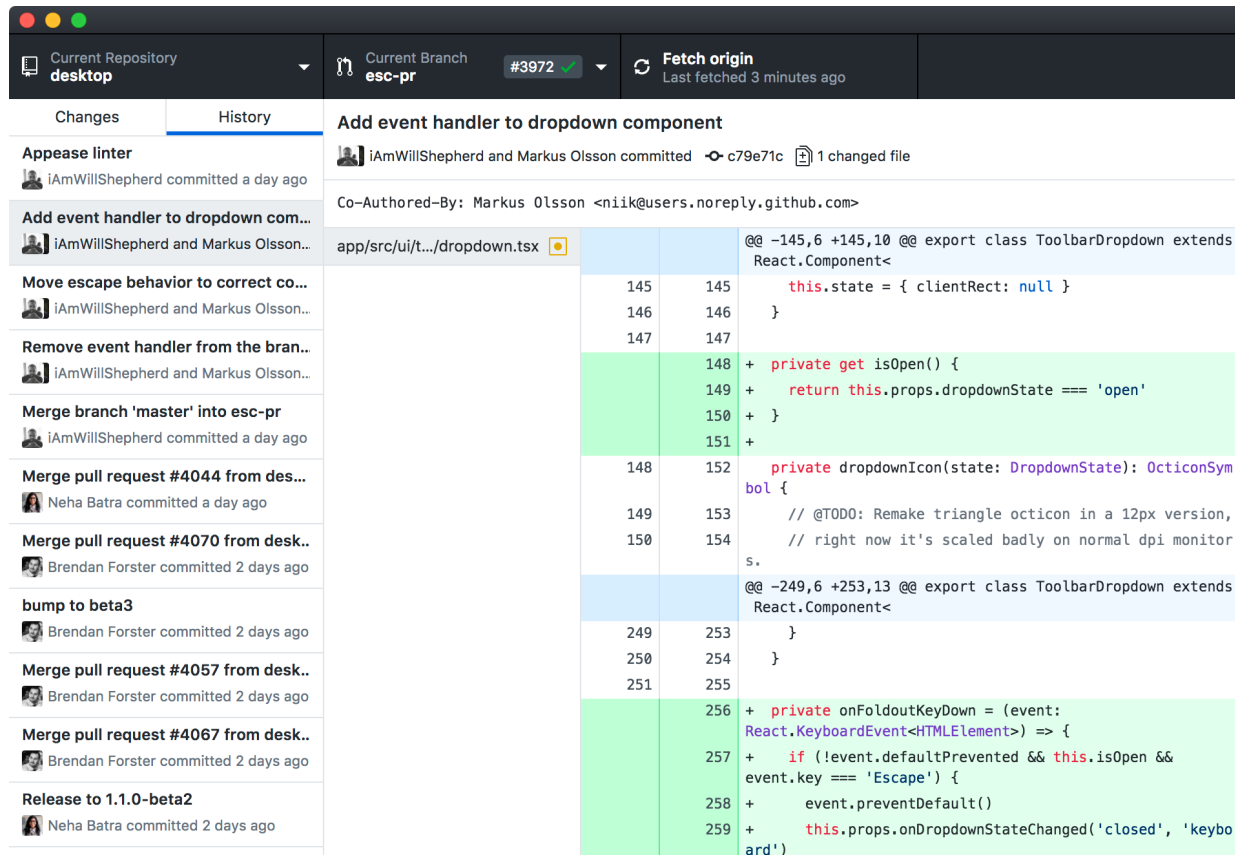
Recap



- Local commands: mv, rm, add, commit, status, log
- Local \leftrightarrow Remote commands: pull, push

Installing Github Desktop

- Download Github Desktop from <https://desktop.github.com/>;



Issue Tracking on Github

The screenshot shows the GitHub interface for the repository 'csytang-se / cs132-demo'. At the top, there are buttons for 'Unwatch' (1), 'Star' (0), and 'Fork' (0). Below these are tabs for 'Code', 'Issues' (highlighted with a blue box), 'Pull' (circled with a blue circle and the number '1'), 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. A notification banner reads 'Label issues and pull requests for new contributors' and 'Now, GitHub will help potential first-time contributors discover issues labeled with good first issue'. Below the banner is a search bar with 'Filters' and 'is:issue is:open'. To the right of the search bar is a 'Labels' dropdown (circled with a blue circle and the number '2') and a 'New issue' button (highlighted with a blue box). The main content area is empty, showing a large question mark icon.

Issue Tracking on Github (2)

<> Code

! Issues

Title & Description

Attributes



Demo issue

Write

Preview

H B I @

This is a demo issue : please fix the bug

Attach files by dragging & dropping, selecting or pasting them.



Styling with Markdown is supported

Submit new issue

Assignees

csytang-se

Labels

enhancement

Projects

None yet

Milestone

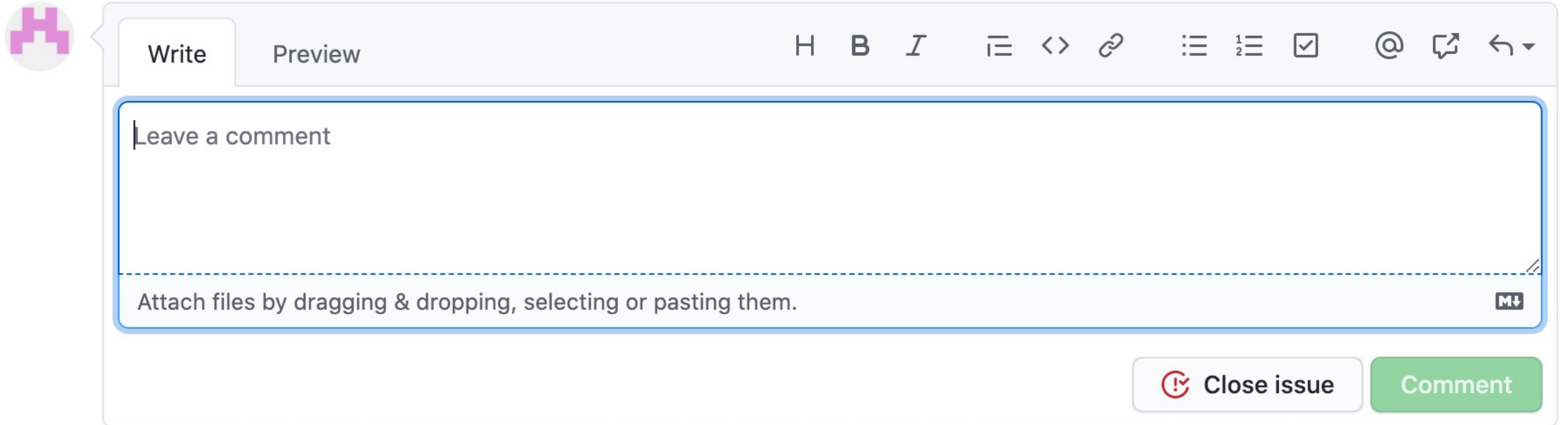
No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

Issue Tracking on Github (3)

- Close the issue when it is resolved.



The screenshot shows the GitHub interface for writing a comment on an issue. On the left is a circular profile picture of a pink pixelated 'H'. The main area has two tabs: 'Write' (active) and 'Preview'. Above the text area is a rich text editor toolbar with icons for heading (H), bold (B), italic (I), list (≡), code (<>), link (chain), table (≡), table of contents (1/2 ≡), checklist (checkbox), mention (@), share (share icon), and a dropdown arrow. The text area contains the placeholder text 'Leave a comment'. Below the text area is a dashed line and the text 'Attach files by dragging & dropping, selecting or pasting them.' with a file upload icon (M) on the right. At the bottom right are two buttons: 'Close issue' with a red circle and checkmark icon, and a green 'Comment' button.

Wiki page on Github

<> Code ⓘ Issues 1 🔗 Pull requests ▶ Actions 📁 Projects 📖 Wiki 🛡 Security 📈 Insights ⚙ Settings



Welcome to the cs132-demo wiki!

Wikis provide a place in your repository to lay out the roadmap of your project, show the current status, and document software better, together.








Create the first page

Wiki page on Github (2)

Create new page

Home

Write Preview

h1 h2 h3   **B** *i* `<>`      Edit mode: Markdown

```
Welcome to the cs132-demo wiki!  
# Introduction  
....  
  
# Environment  
....  
  
# HowTo  
....  
  
# Demo  
....
```

```
# Introduction  
# Environment  
# Installation  
# Howto  
# Bug fixing  
# Contact
```

Reference

- Version Control by Example. Eric Sink
https://ericsink.com/vcbe/vcbe_a4_lo.pdf (Free, online available);
- Pro Git: <https://git-scm.com/book/zh/v2>
- Official doc of Github: <https://docs.github.com/cn>

