

# INTRODUCTION TO GIT & GITHUB

# BEFORE GIT

Programmers want to restore a file to its previous state, they may do it in this way:

Copy the file before edit, rename it

Multi-version files will be generated:

Project\_1, Project\_1\_Revised, Project\_1\_Small Revised, Proejct\_1\_Final, Project\_1\_New Final...

**Disaster if changes affect multiple files !!**



# BEFORE GIT

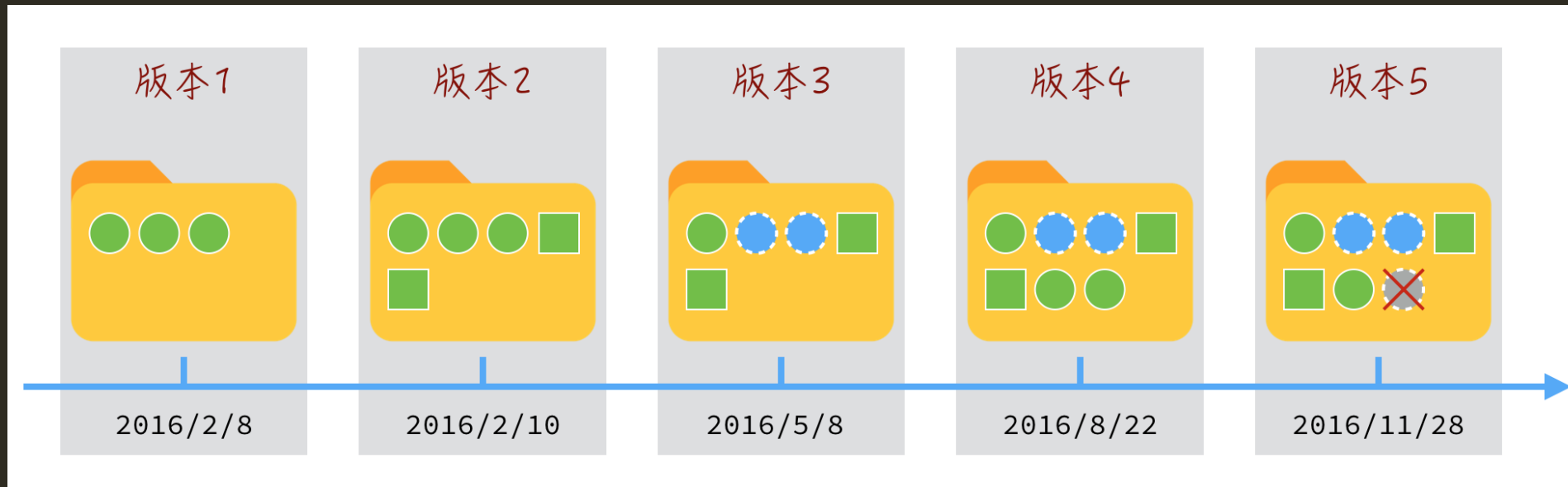
No version control:

- Difficult to collaborate
- Files will be overwritten
- Cannot keep track of the changes and the editors



# WHAT IS GIT?

## Distributed Version Control System



Suppose you are writing a website and store the content to a folder. What Git does is that it keeps track of all of these changes.

Simply speaking, Git is just like **computer game saving**. One can create or restore a backup whenever he wants.

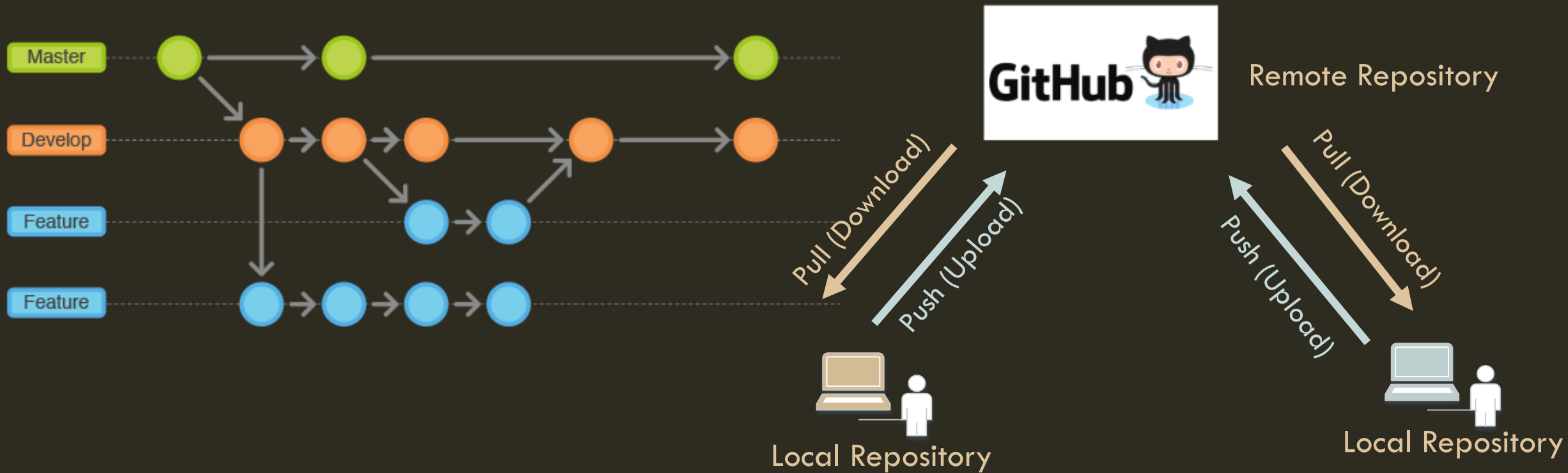
# WHAT IS GITHUB?

- A web-based **Git repository** hosting service
- Allow users to download and use the open source code
- Collaborate with software engineers all over the world and contribute your code to projects
- A software engineer version Facebook



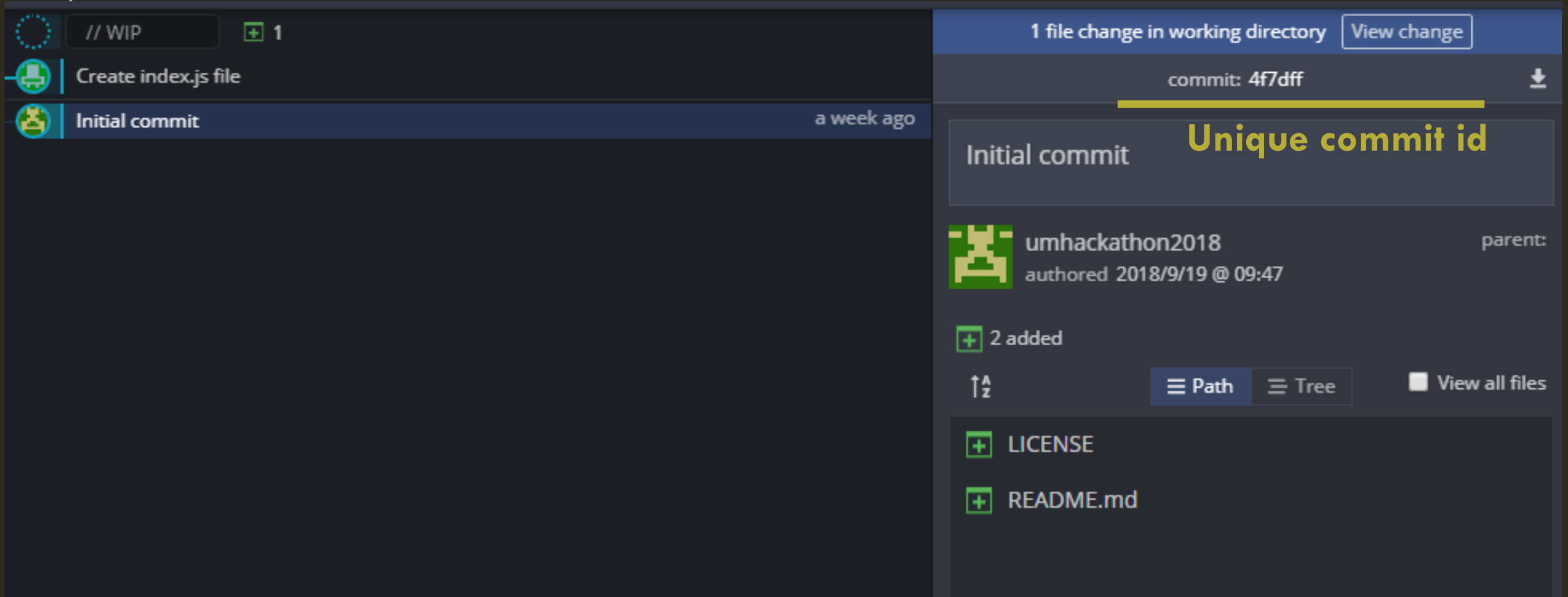
# GIT CONCEPT - REPOSITORY

File edit histories + code files = Repository



# GIT CONCEPT — COMMIT

7

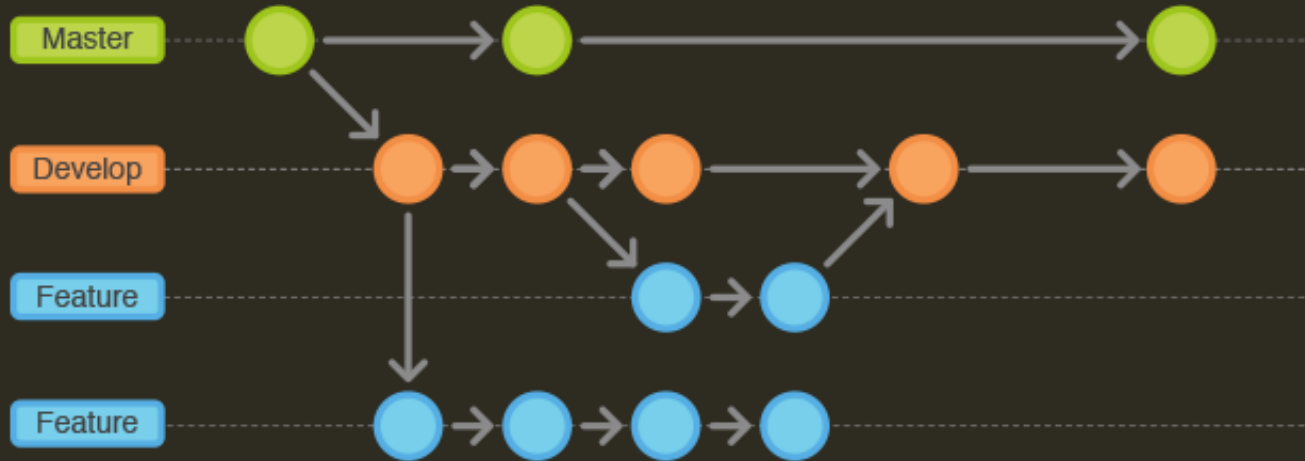


A Git repository contains mainly a set of **commits**.

A commit contains a set of file changes histories (e.g. added lines to a file), date, description of the changes and a unique hash code identified the commit.

It can be considered as a game saving checkpoint, you can restore the save whenever you need.

# GIT CONCEPT - BRANCH



A function for collaborative development.

A default branch called “**master**” will be created when the repository initializes.



# LEARN GIT & GITHUB THROUGH EXAMPLE

# INTRO

In this section, we will go through some situations you will encounter when using Git and GitHub during the UM Hackathon.

In the following part, we assume there are two programmers in your team. One of them is the remote repository owner (**team leader**). All commits are made under branch **master**.



**Team Leader** / Programmer  
Username: **umhackathon2018**



Team member  
Username: **umhackathon2018\_prog**

# GIT INSTALLATION

## Git Command Line

Download Git from this URL <https://git-scm.com/downloads>

## Git GUI

- Git Kraken <https://www.gitkraken.com/>

(support all platforms, we will show you how to use it)



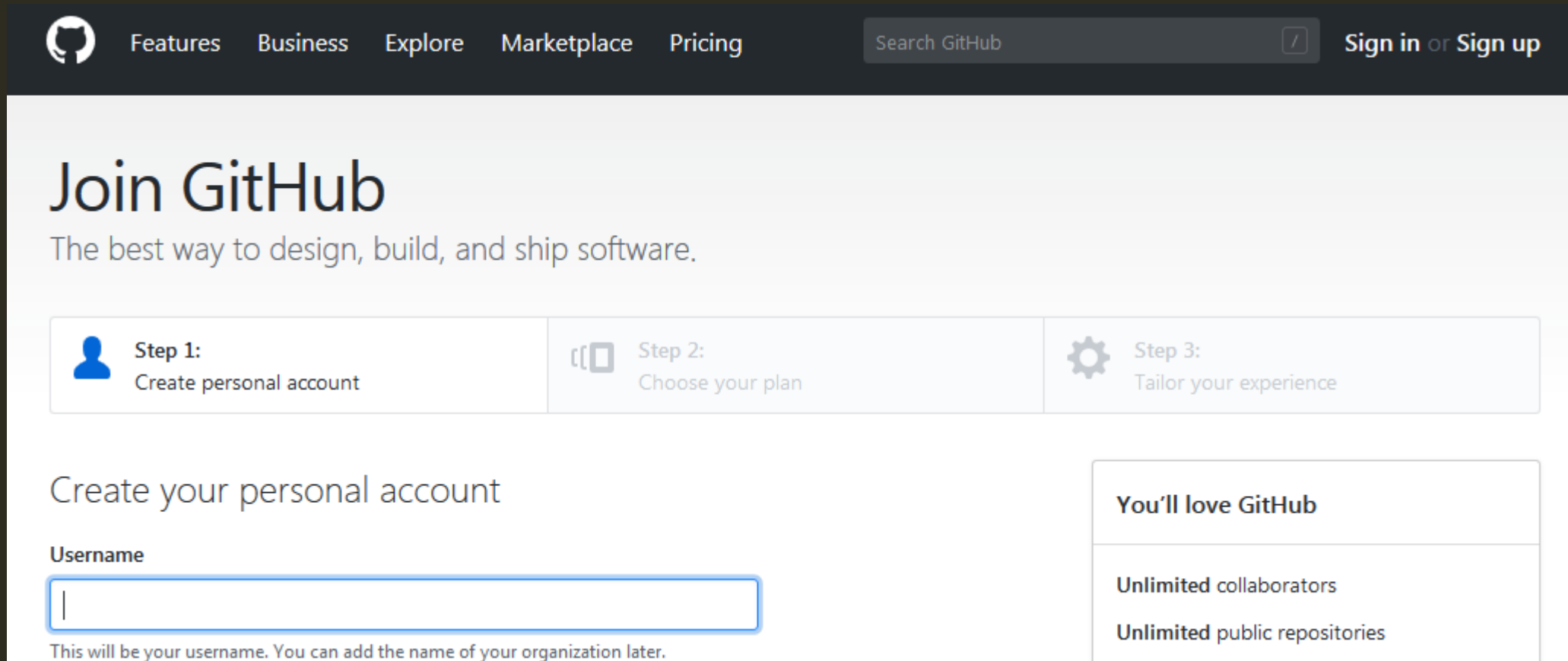
## Other GUI options:

- SourceTree (MacOS, Windows)
- TortoiseGit (Windows)


# SIGN UP A GITHUB ACCOUNT

All developers should sign up a GitHub account before the Hackathon through this website:

<http://www.github.com>





The screenshot shows the GitHub homepage with a dark navigation bar at the top. The navigation bar includes the GitHub logo, links for Features, Business, Explore, Marketplace, and Pricing, a search bar labeled 'Search GitHub', and links for 'Sign in' or 'Sign up'. Below the navigation bar, the main heading is 'Join GitHub' with the tagline 'The best way to design, build, and ship software.' A three-step process is outlined: Step 1: Create personal account (with a person icon), Step 2: Choose your plan (with a document icon), and Step 3: Tailor your experience (with a gear icon). The first step is expanded, showing a form titled 'Create your personal account' with a 'Username' label and an input field. Below the input field, a note states: 'This will be your username. You can add the name of your organization later.' To the right of the form, a box titled 'You'll love GitHub' lists 'Unlimited collaborators' and 'Unlimited public repositories'.


 [Features](#) [Business](#) [Explore](#) [Marketplace](#) [Pricing](#)  [Sign in](#) or [Sign up](#)

## Join GitHub

The best way to design, build, and ship software.

 **Step 1:**  
Create personal account

 **Step 2:**  
Choose your plan

 **Step 3:**  
Tailor your experience

### Create your personal account

Username

This will be your username. You can add the name of your organization later.

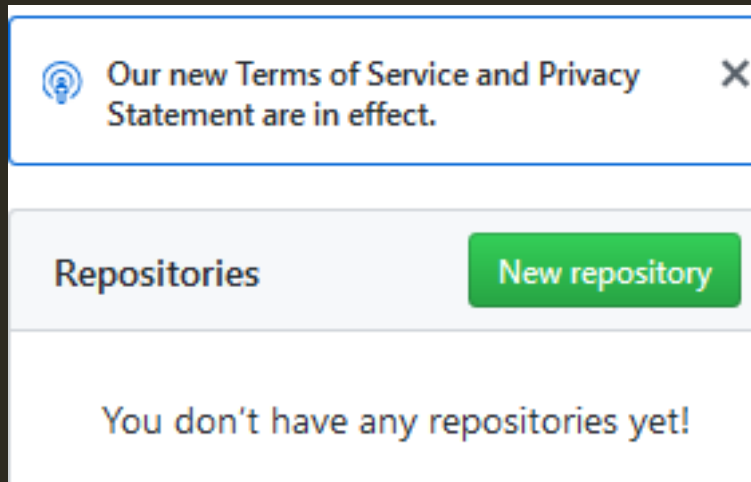
#### You'll love GitHub

- Unlimited collaborators
- Unlimited public repositories



# CREATE A REMOTE GIT REPOSITORY (1)

1



These steps are for **team leader**.

1.1 Login your GitHub account

1.2 Click on “New repository” button.

2.1 Assume the repository name is demo01

2.2 Select “Public” repository

2.3 Tick “Initialize this repository with a README”

2.4 Choose “None” license.

2



# CREATE A REMOTE GIT REPOSITORY (2)

This step is for **team leader**.

Share the repository page URL [https://github.com/<owner\\_name>/<repository\\_name>](https://github.com/<owner_name>/<repository_name>) to other team members.

3

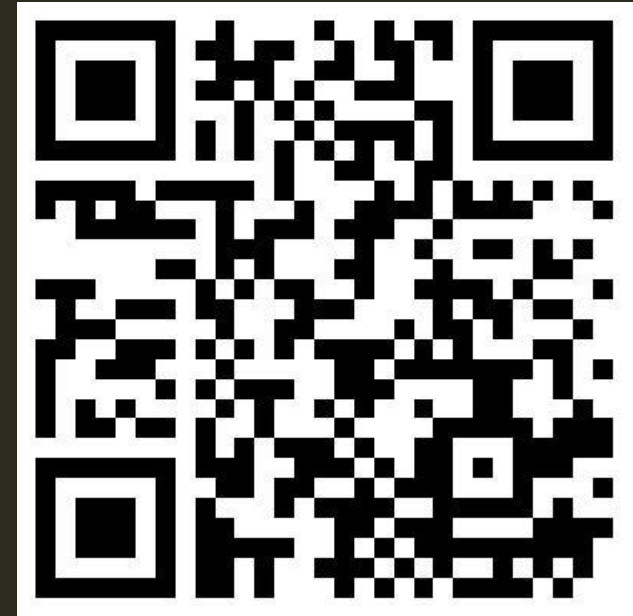
The screenshot shows a web browser displaying the GitHub repository page for `umhackathon2018/demo01`. The browser's address bar shows the URL `https://github.com/umhackathon2018/demo01`. The GitHub interface includes a search bar, navigation links for Pull requests, Issues, Marketplace, and Explore, and a notification bell. The repository name `umhackathon2018 / demo01` is displayed, along with buttons for Watch (0), Star (0), and Fork (0). Below the repository name, there are tabs for Code (selected), Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings. The main content area shows the text "It is a demo." with an "Edit" button and a link to "Manage topics". At the bottom, a summary bar displays repository statistics: 1 commit, 1 branch, 0 releases, 1 contributor, and the MIT license.

# COLLECT TEAM'S GIT REPO

IMPORTANT!

To save time during the competition,  
please fill in this form in advance:

<https://goo.gl/forms/az3oTgVfdVgRwm812>



# ADD MEMBERS (1)



16

**Team leader** needs to add team members to the repository in order to grant push right to them.

These steps are for the **team leader**:

1. Click on “Settings” tab
2. Click on “Collaborators” button
3. Search your member’s username
4. Click “Add collaborator”

The screenshot shows the GitHub repository settings page. The 'Settings' tab is selected in the top navigation bar, indicated by a red box and the number 1. In the left sidebar, the 'Collaborators' option is selected, also indicated by a red box and the number 2. The main content area is titled 'Collaborators' and includes a sub-header 'Push access to the repository'. Below this, a message states: 'This repository doesn't have any collaborators yet. Use the form below to add a collaborator.' A search bar is provided with the placeholder text 'Search by username, full name or email address'. Below the search bar, a dropdown menu shows search results for 'umhackathon20' and 'umhackathon2018-prog', with the first result highlighted in blue. A red box and the number 3 are placed over the search input. To the right of the search bar is an 'Add collaborator' button, which is also highlighted with a red box and the number 4.






# ADD MEMBERS (2)

Collaborators

Push access to the repository

 Awaiting umhackathon2018-prog's response

Copy invite link ▼

Cancel invite

Search by username, full name or email address


You'll only be able to find a GitHub user by the username or email address instead.

Add collaborator

Copy invite link

umhackathon2018-prog's shareable invitation link.

<https://github.com/umhackathon2018/demo>



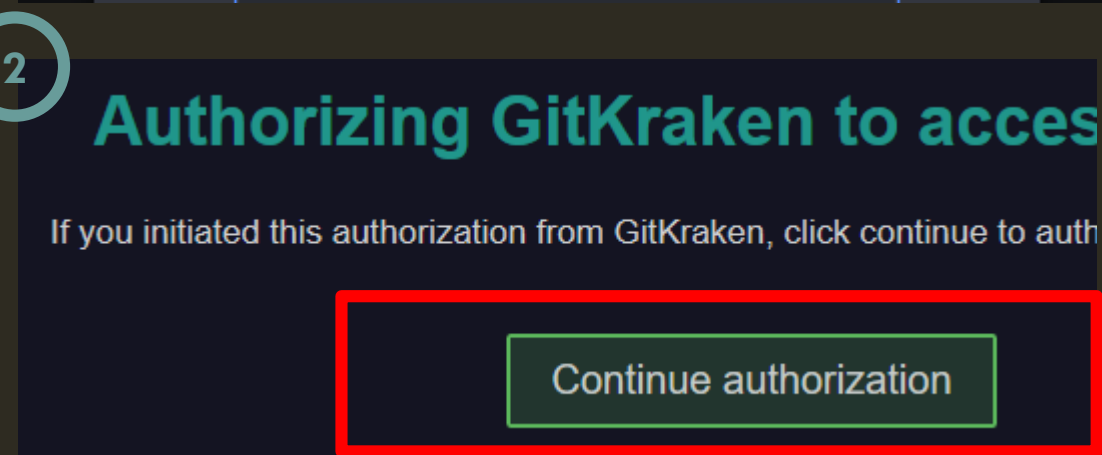
**Team leader** sends the invitation link to the team member.  
Team member needs to accept this invitation.

# GIT KRAKEN SETUP



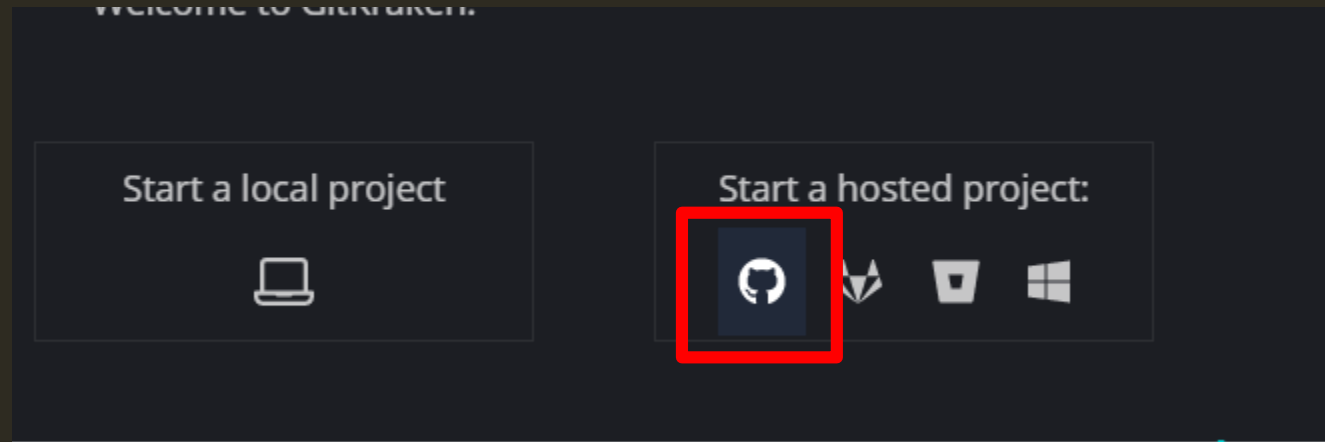
1. A box will pop up when you launch Git Kraken at the first time. Click “Sign in with GitHub”

2. Authorize access to GitHub by login your GitHub account



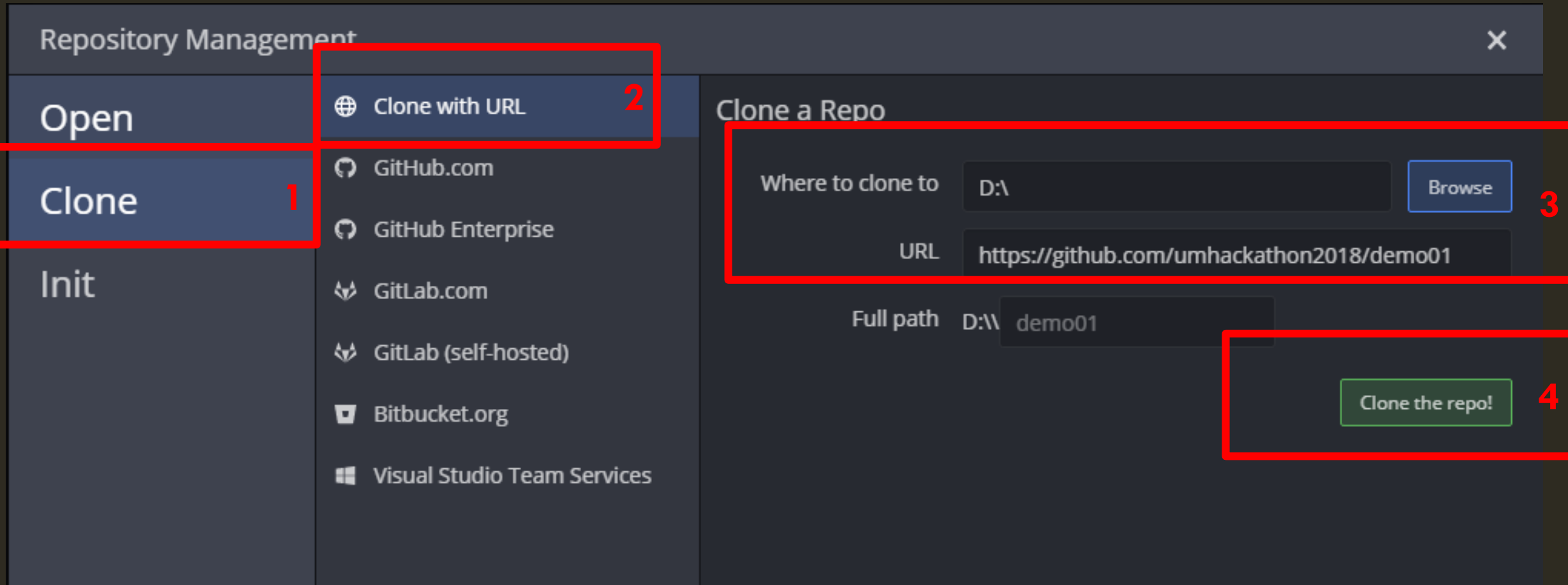
# CLONE THE REPOSITORY TO LOCAL (1)

1. Launch Git Kraken
2. Click “Start a hosted project”



# CLONE THE REPOSITORY TO LOCAL (2)

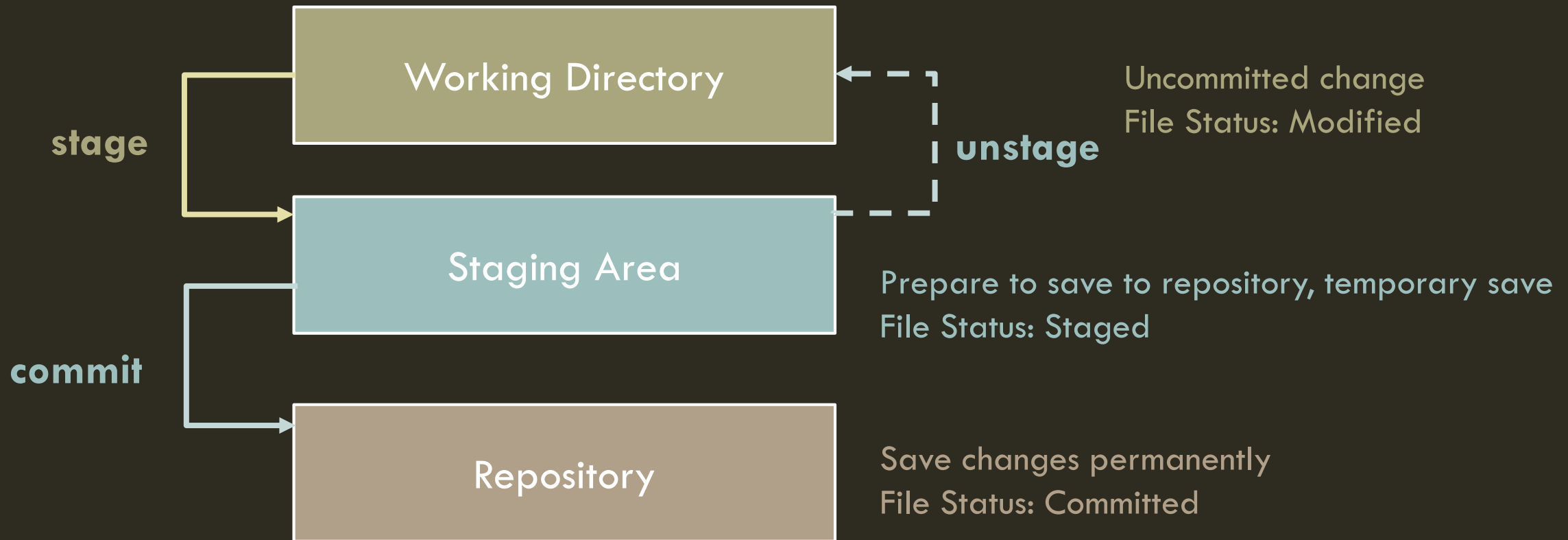
3. Paste the shared repository URL [https://github.com/<owner\\_name>/<repository\\_name>](https://github.com/<owner_name>/<repository_name>)



# CREATE COMMIT PRINCIPLE

How commits are created in Git?

Files have three status in local Git repository:  
**Modified**, **Staged**, **Committed**



# EXAMPLE: CREATE COMMIT



Create file index.html, index.js

Undo Redo Pull Push Branch Stash Pop

// WIP + 2

Initial commit a week ago

2 file changes in working directory [View changes](#)

commit: 4f7dff

Initial commit

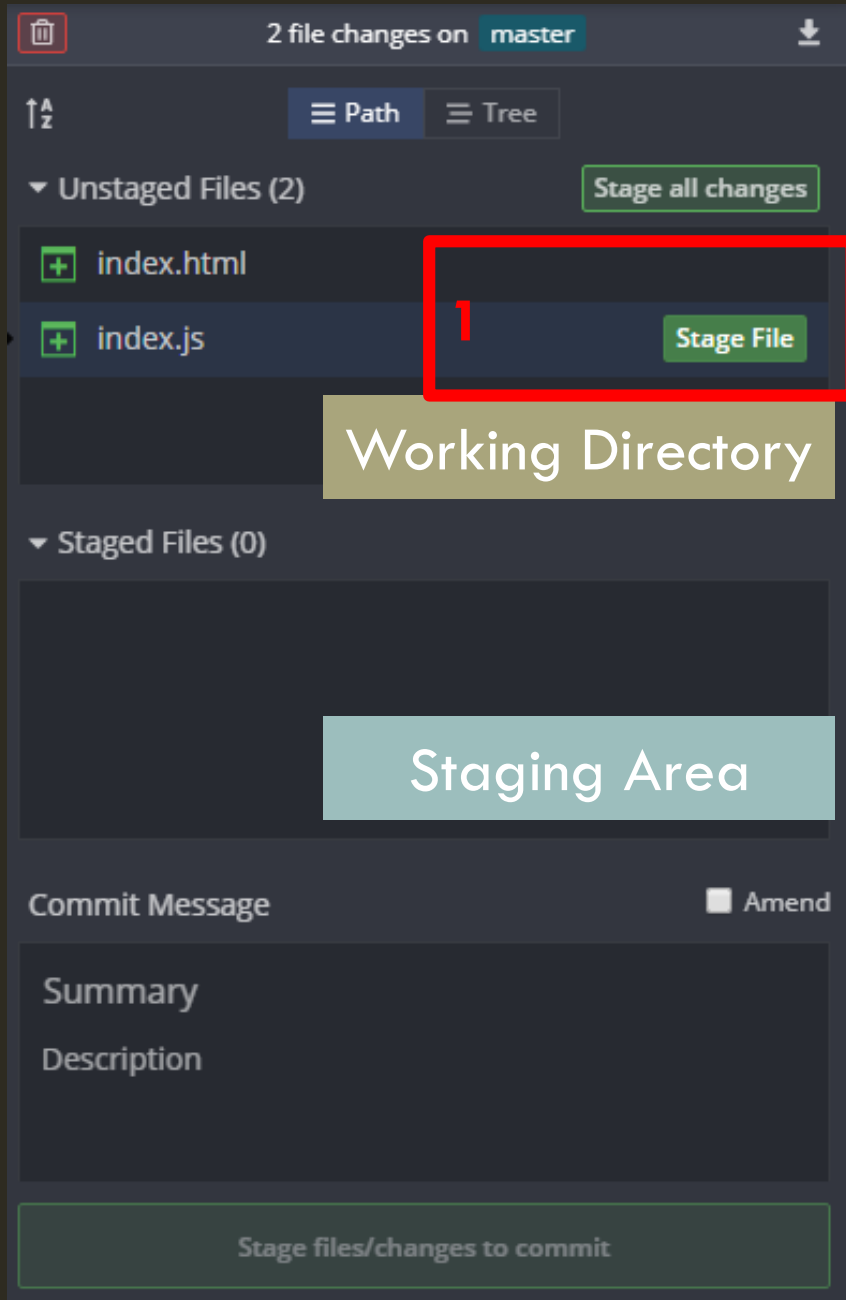
umhackathon2018 parent:  
authored 2018/9/19 @ 09:47

+ 2 added

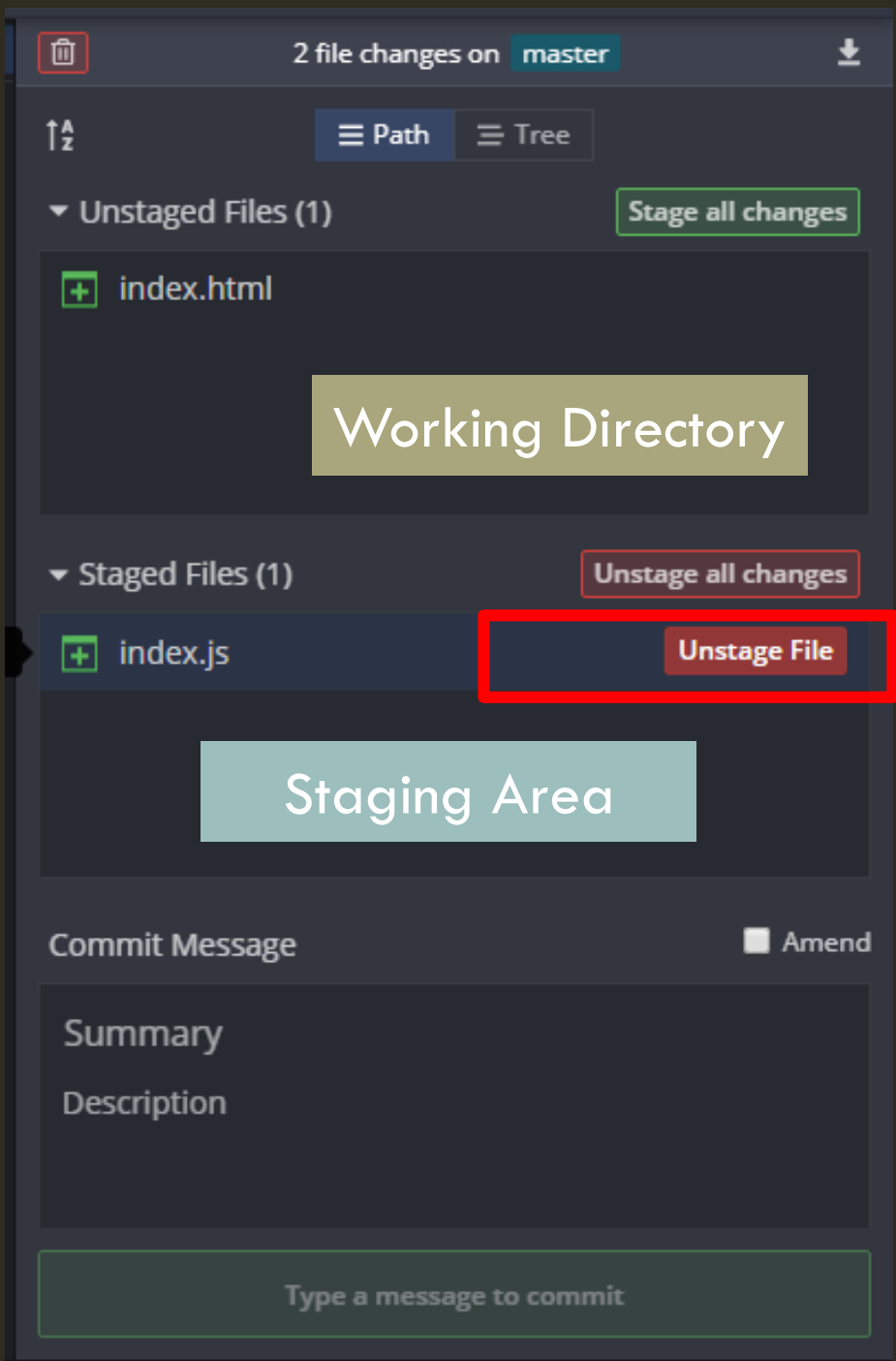
↑↓ Path Tree View all files

Git Kraken automatically detects changes in your repository. You can click the “View changes” button located in the upper right corner.

# EXAMPLE: CREATE COMMIT



Click the “Stage File” button to select files added to the staging area. Here we add the index.js file to the staging area.



# EXAMPLE: CREATE COMMIT

24

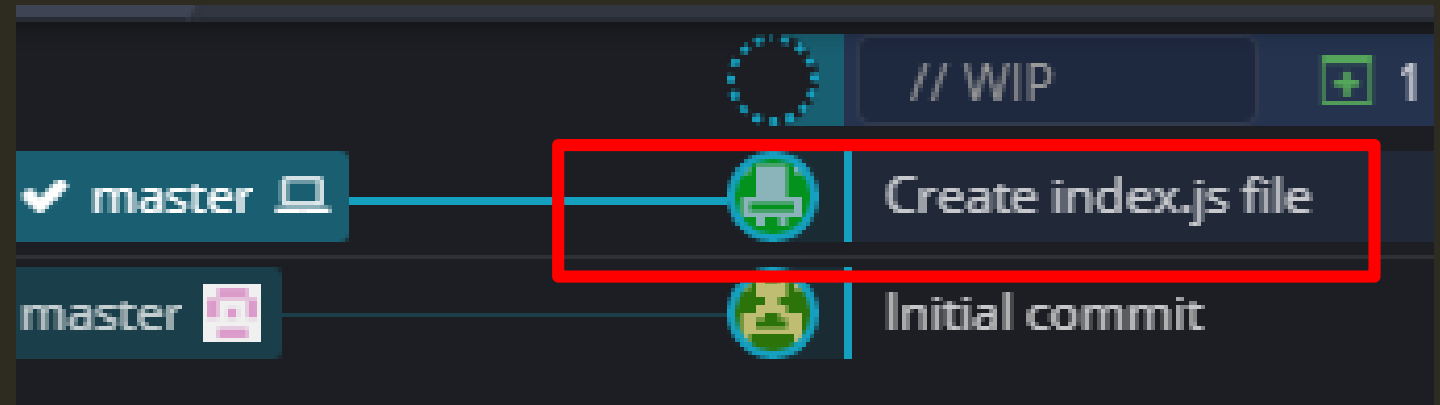
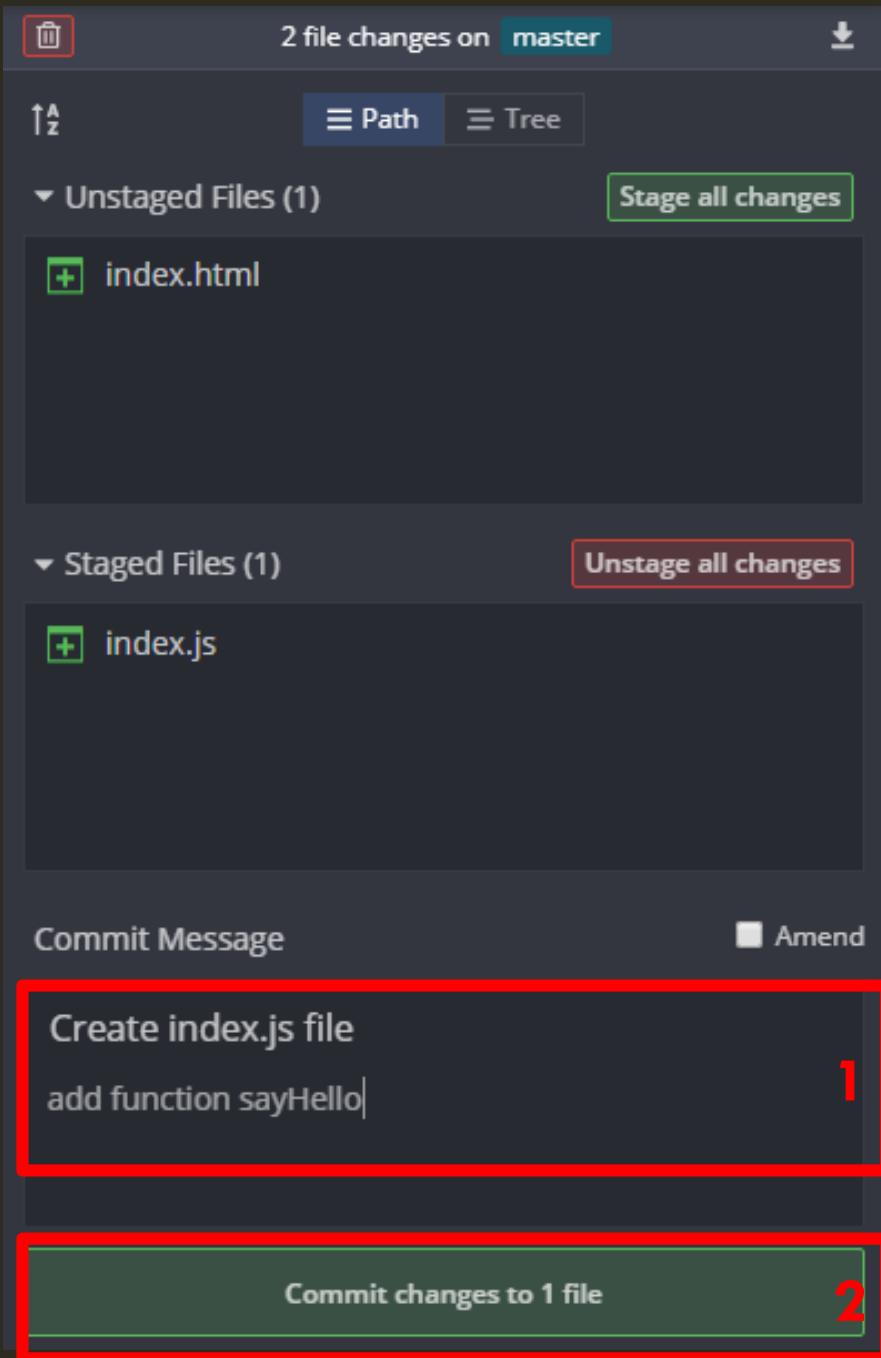
You can click the “Unstage File” button if you do not want to commit the changes.



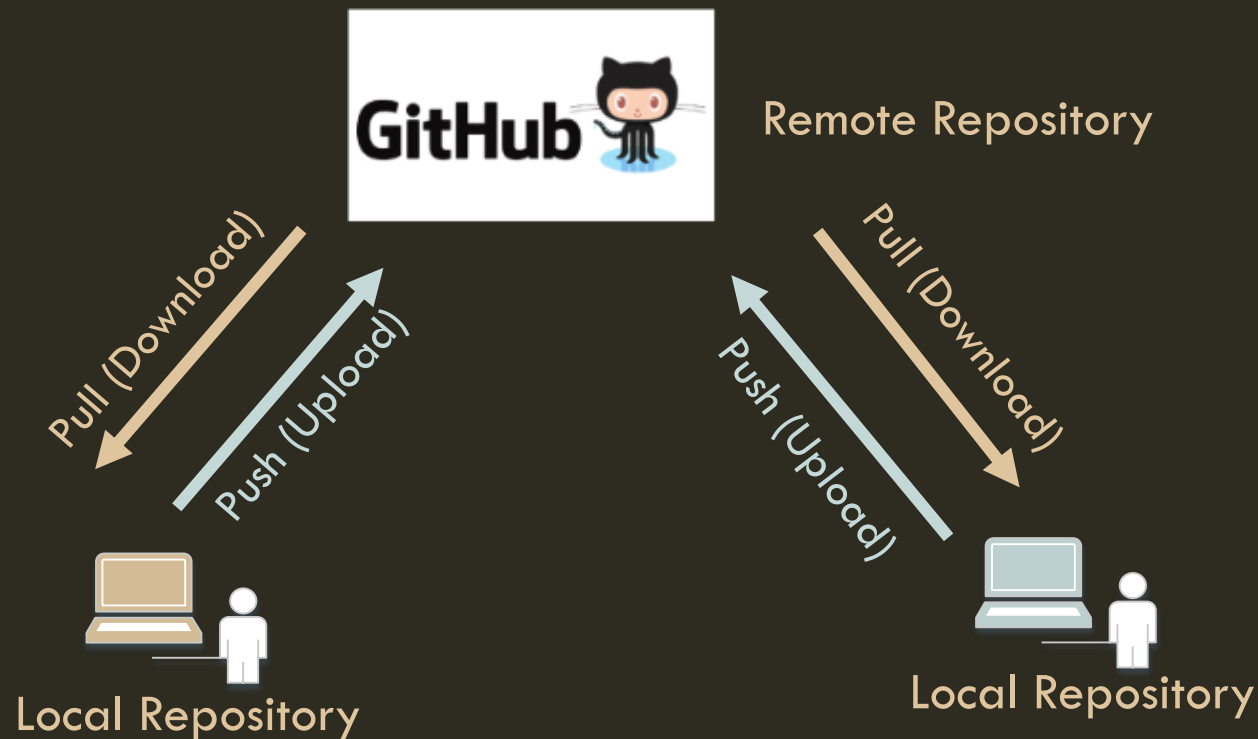
# EXAMPLE: CREATE COMMIT

1. Briefly describe what you have done
2. Click the button to commit

An extra commit is created in the local repository.



# PULL/PUSH COMMIT (DOWNLOAD/UPLOAD CHANGES)



# EXAMPLE: PULL/PUSH COMMIT

27



Share changes



Remote repository

Assume team leader has made 2 commits under his local repository.  
Team member has made 1 commit under her local repository.

They want to share changes to each other.

# EXAMPLE: PULL/PUSH COMMIT

28



Click “Push” button to upload changes to remote repository

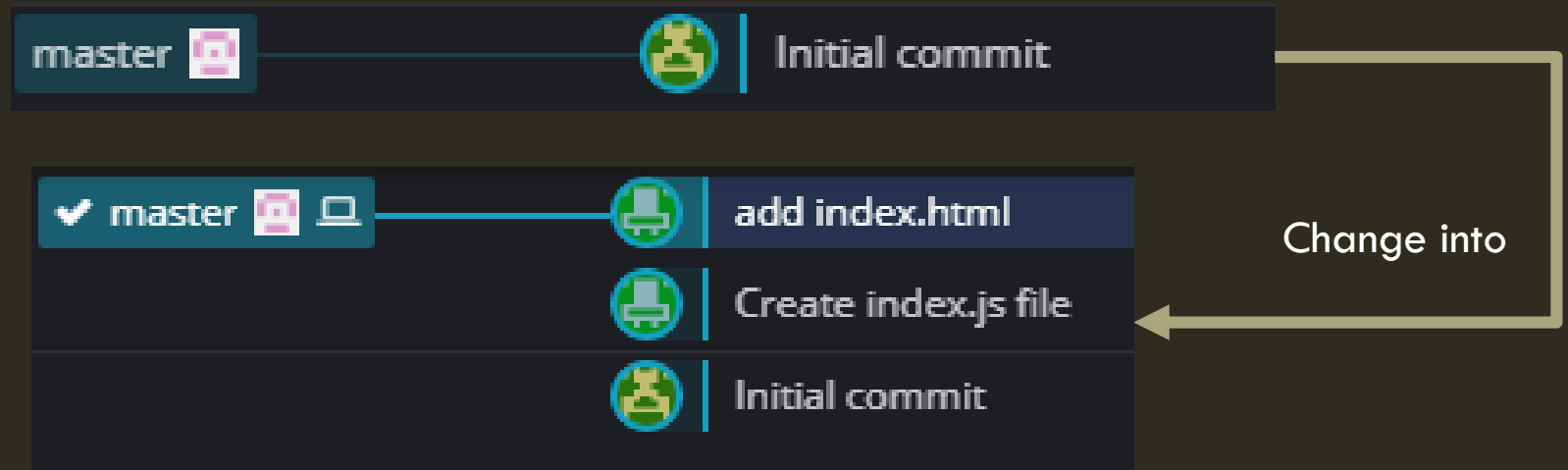
Push (Upload)

The screenshot shows a code editor interface. At the top, there is a toolbar with four buttons: 'Undo' (curved arrow left), 'Redo' (curved arrow right), 'Pull' (downward arrow), and 'Push' (upward arrow). The 'Push' button is highlighted with a red rectangular border. Below the toolbar, there is a list of three commit items, each with a circular icon on the left and text on the right: 1. A green circle with a computer icon, followed by 'add index.html'. 2. A green circle with a computer icon, followed by 'Create index.js file'. 3. A green circle with a person icon, followed by 'Initial commit'.

Assume team leader pushes changes to remote repository first.



Remote repository



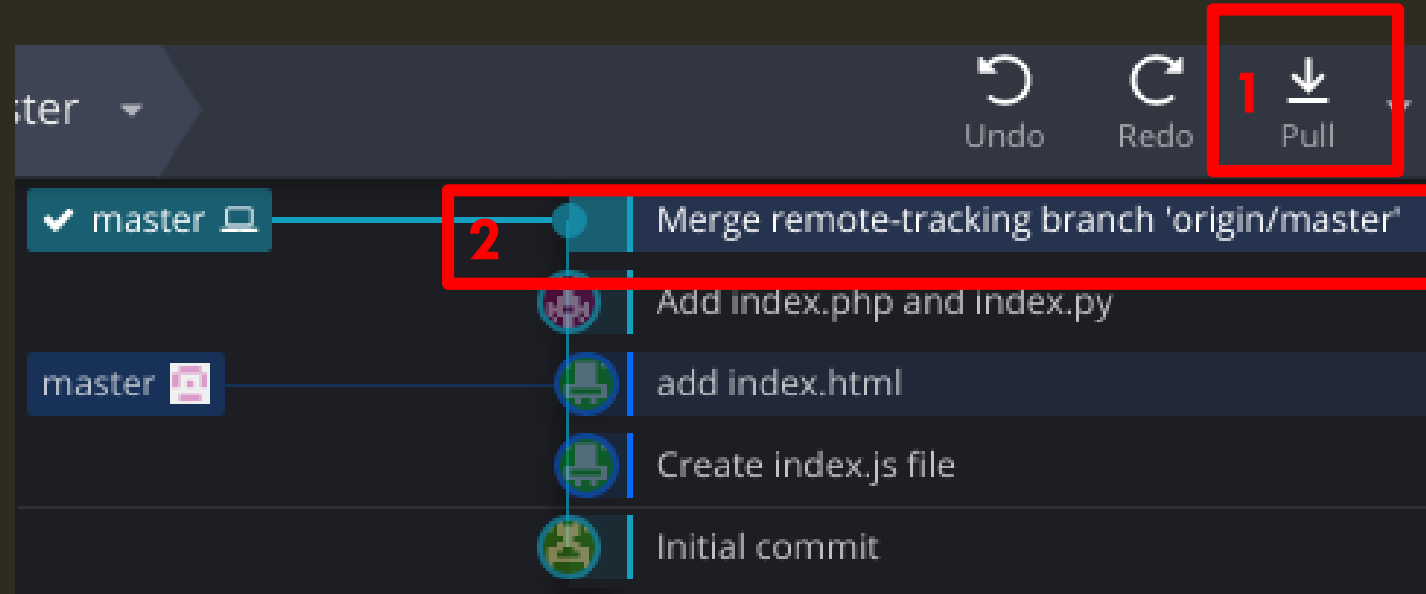
# EXAMPLE: PULL/PUSH COMMIT

29



↑  
Pull (Download)

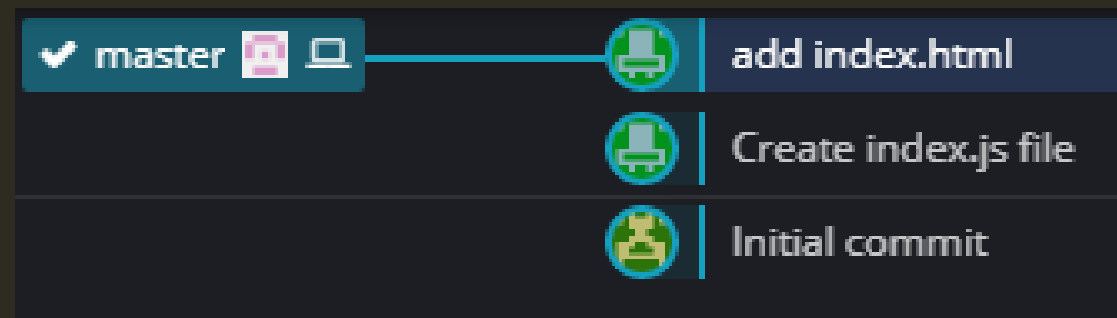
Click “Pull” button to download changes from remote repository before upload her commits



A new merged commit will be automatically generated.



Remote repository

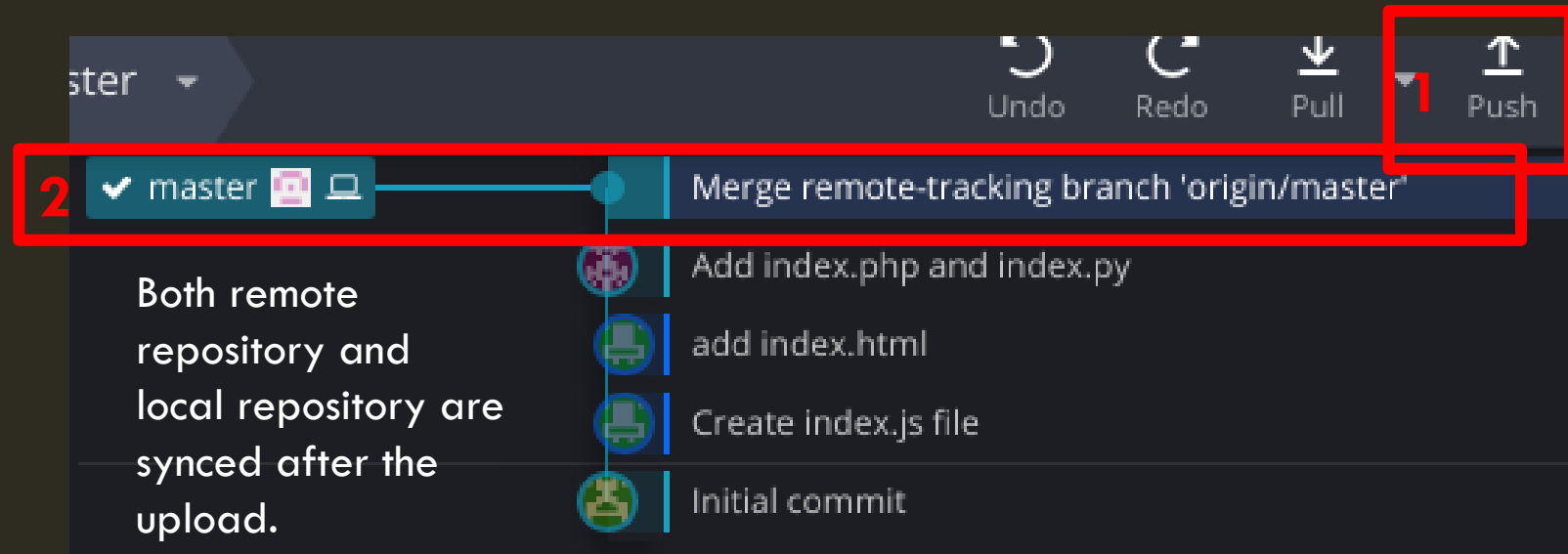


# EXAMPLE: PULL/PUSH COMMIT

30



Click “Push” button to upload changes to remote repository



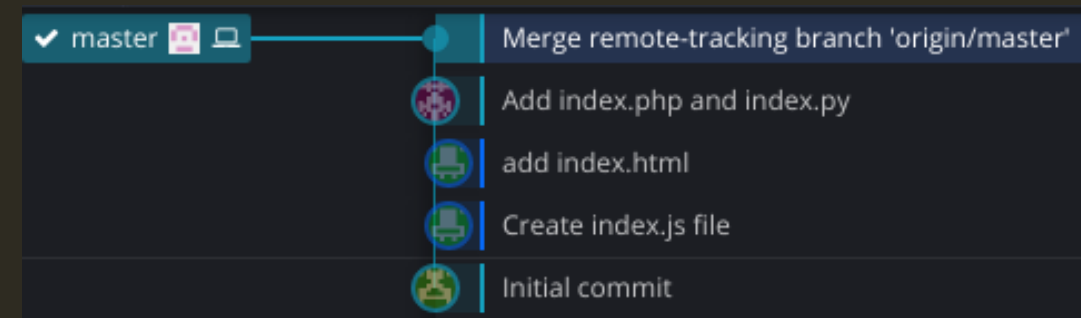
Push (Upload)



Remote repository



Change into



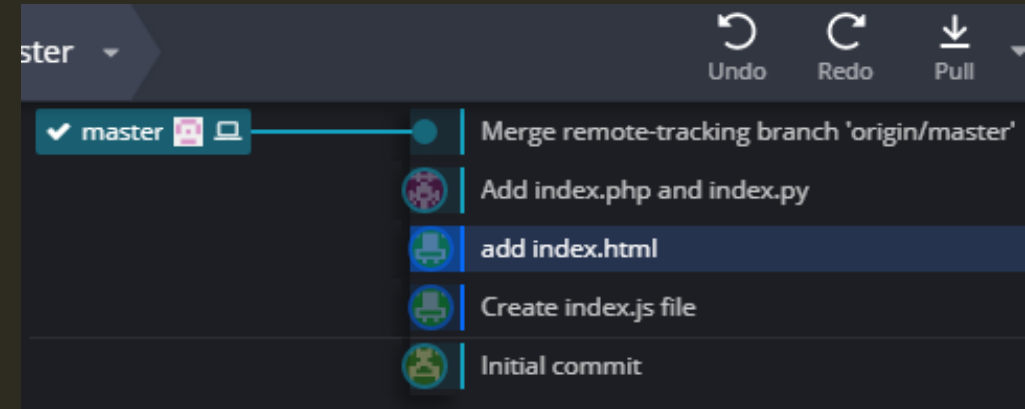
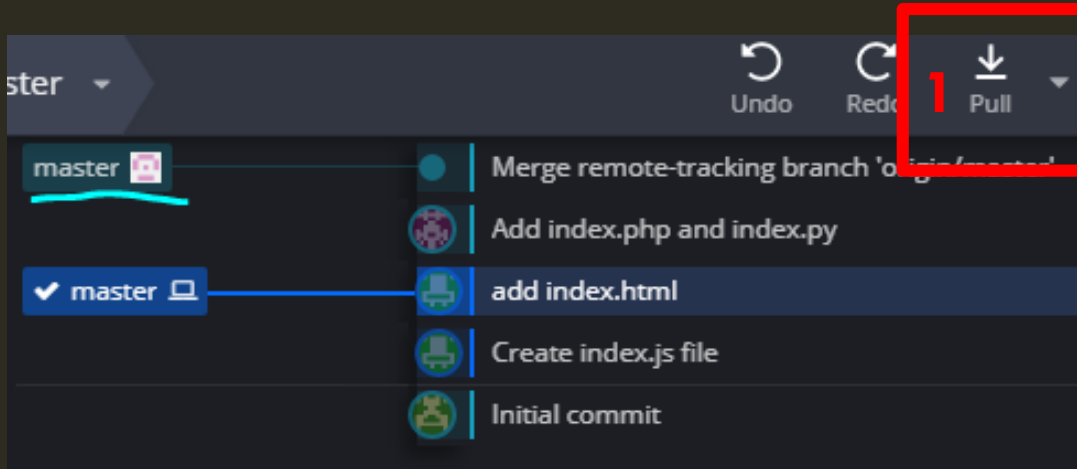
# EXAMPLE: PULL/PUSH COMMIT

31

Click “Pull” button to download changes from remote repository



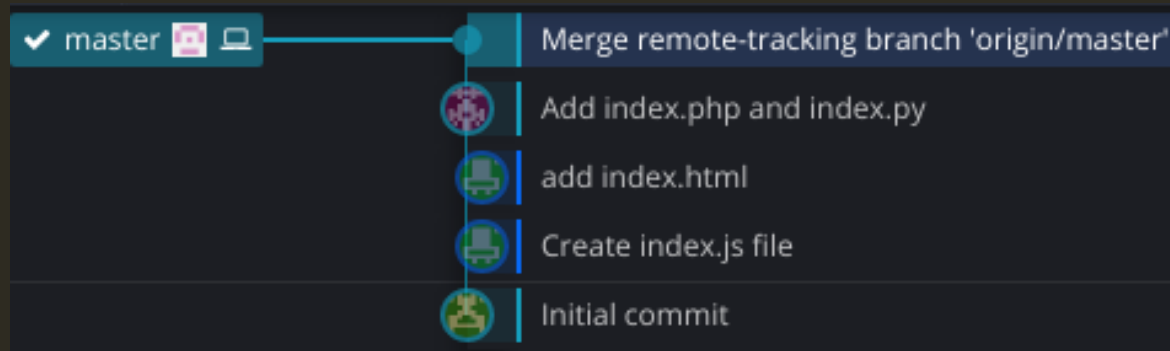
Pull (Download)



Change into



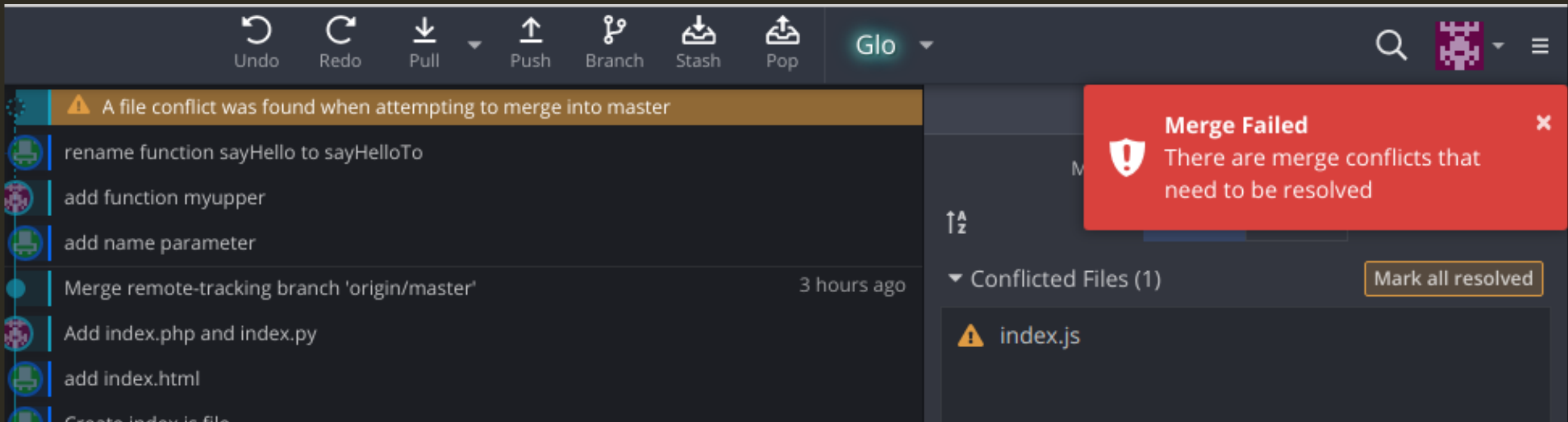
Remote repository



# RESOLVE CONFLICT

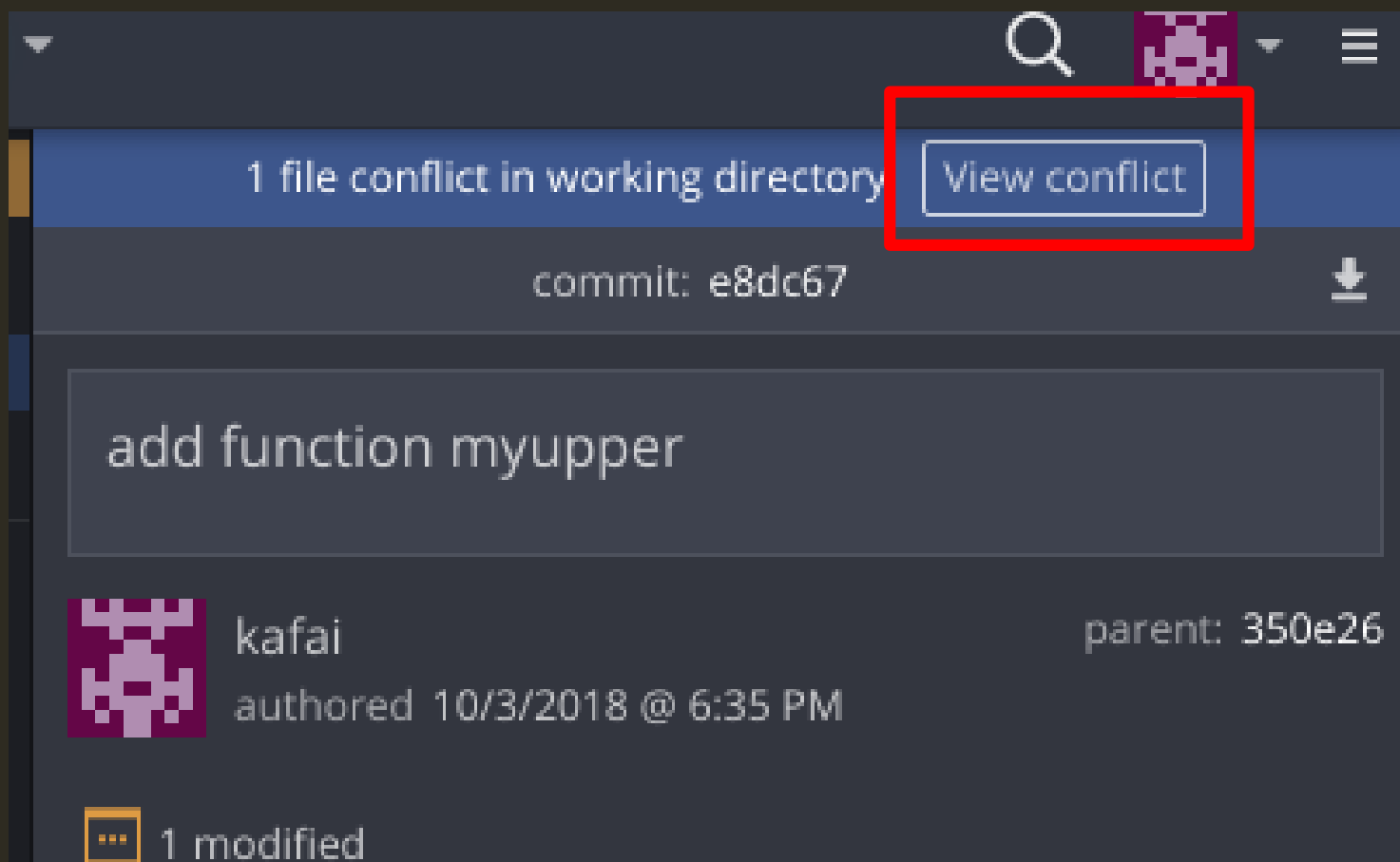
32

When you pull changes from remote repository, the following situation may occur. This is because both you and your teammate edit same line of the file. At this moment, you need to **resolve conflict by yourself** before pushing your commits to remote repository.





# EXAMPLE: RESOLVE CONFLICT





Merging `origin/master` into `master`



≡ Path

≡ Tree

▼ Conflicted Files (1)

Mark all resolved

⚠ index.js

▼ Resolved Files (0)

Commit Message

Merge remote-tracking branch 'origin/master'

Description

Commit and Merge

Abort Merge

# EXAMPLE: RESOLVE CONFLICT

34

The box listed out all conflicted files,  
click on it to resolve

# EXAMPLE: RESOLVE CONFLICT

35

⚠ index.js (1 conflict)

❑ **A** Commit e8dc67 on *master*

```
❑ 1 function sayHello(name){
  2 ✓ alert("Hello, "+myupper(name));
  3 }
  4
  5 function myupper(name){
  6   return name.toUpperCase();
  7 }
  8
```

Local Repository

❑ **B** Commit on *origin/master*

```
❑ 1 ✓ function sayHelloTo(name){
  2   alert("Hello, "+name);
  3 }
  4
```

Remote Repository

**The highlighted lines are the conflicts.** You can click on the lines to mark which line should be kept in the output. Here we keep the first line from the remote repository and keep the second line from the local repository.

Output

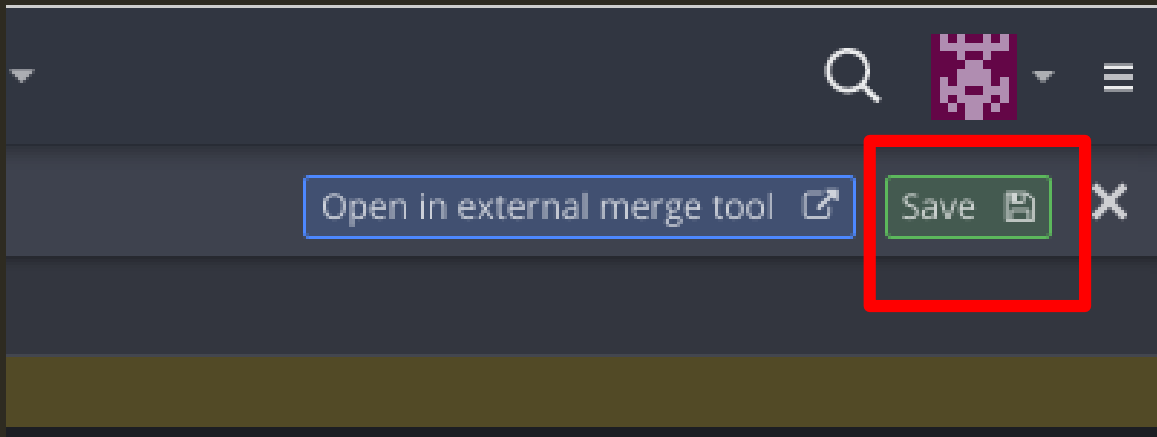
conflict 1 of 1 ^ v

```
B 1 function sayHelloTo(name){
A 2 alert("Hello, "+myupper(name));
  3 }
  4
  5 function myupper(name){
  6   return name.toUpperCase();
  7 }
  8
```

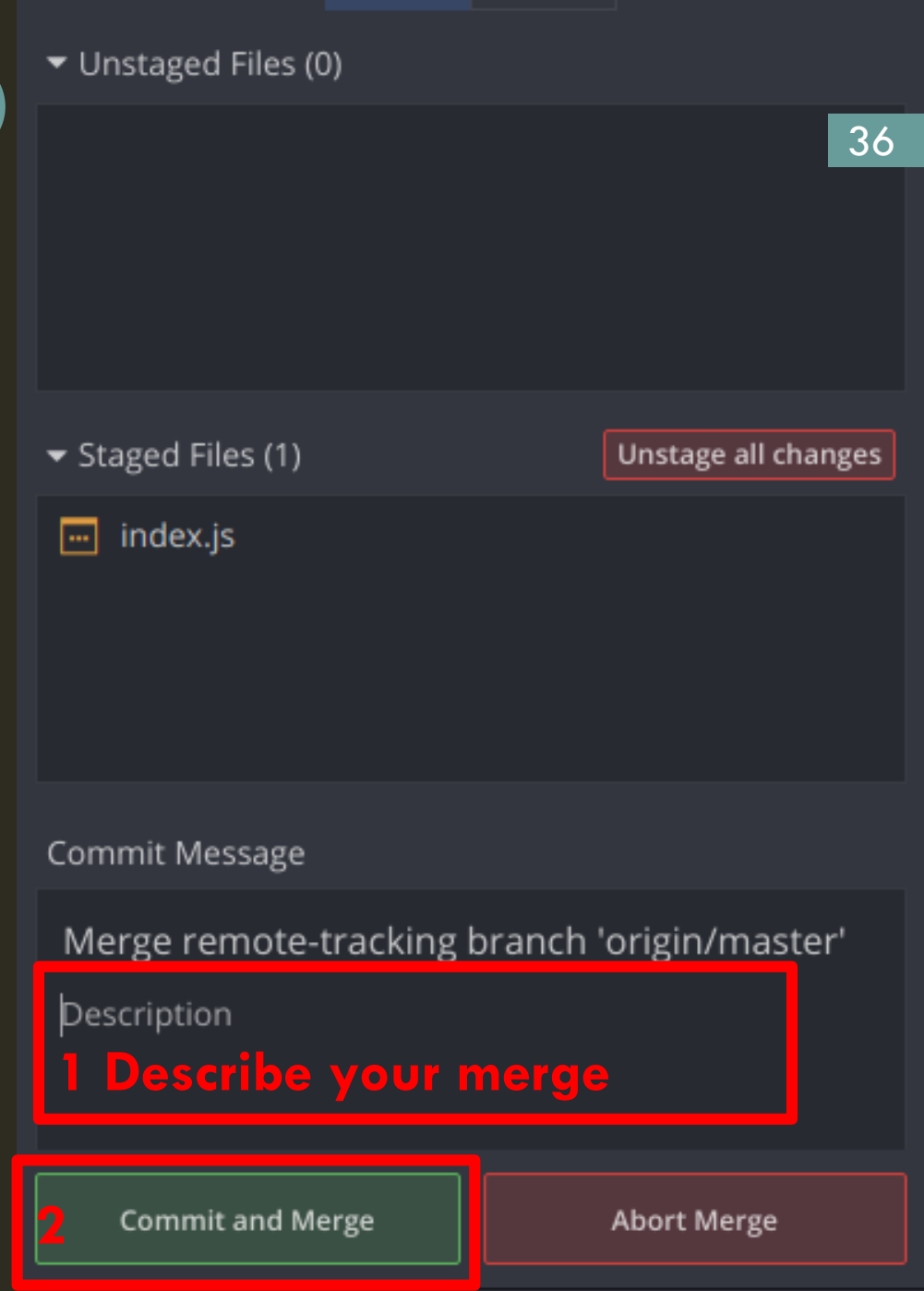
Expected Merge Result

# EXAMPLE: RESOLVE CONFLICT

1



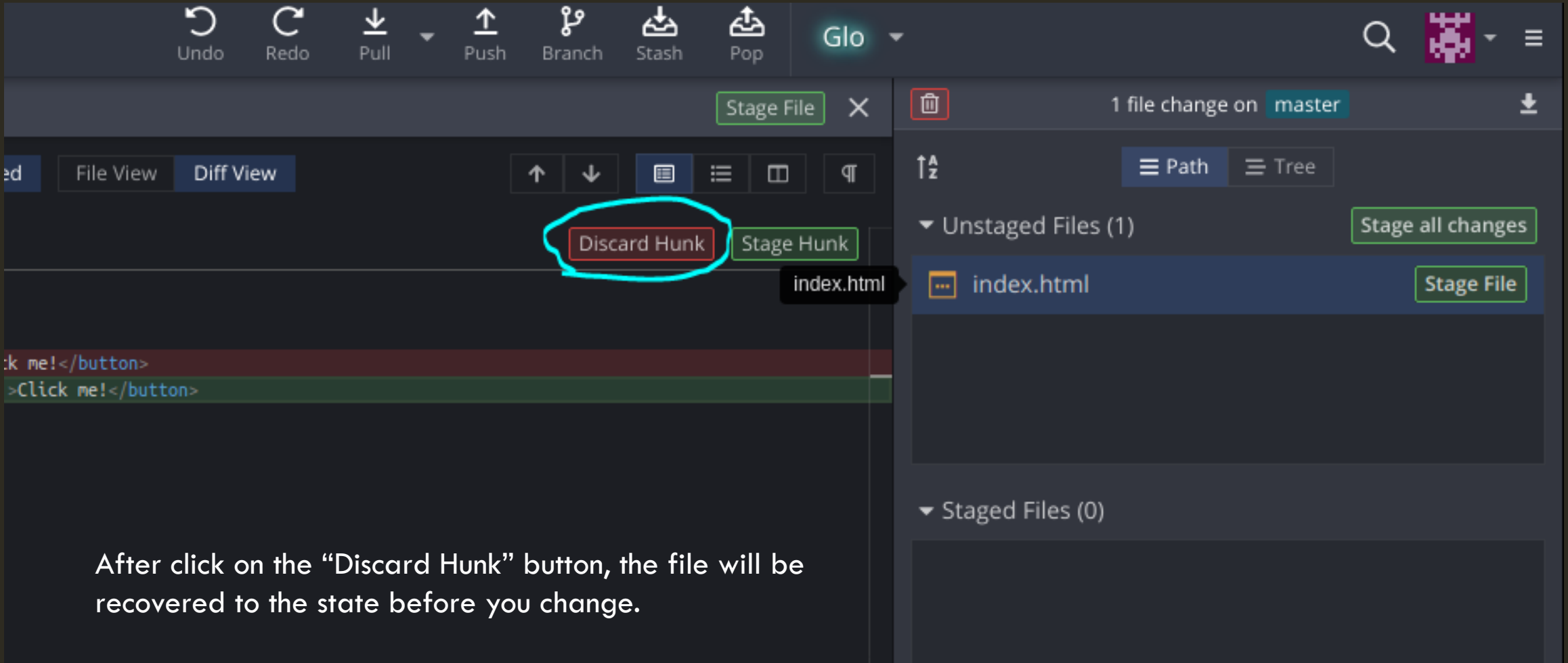
2



36

1. Click Save
2. Describe your merge and click “Commit and Merge” button

# ROLLBACK UNCOMMIT CODE



After click on the “Discard Hunk” button, the file will be recovered to the state before you change.

# RECOVER BY COMPARE COMMITS

The screenshot displays the Git GUI interface. On the left, the commit history is shown as a vertical list of commits. Two commits are highlighted with red boxes and numbered: 'modify html file after change index.js' (labeled 1) and 'add function myupper' (labeled 2). The right panel shows the diff view for the selected commits, titled 'Viewing diff between 2 commits'. It lists the two commits with their hashes and authors. Below this, a section titled '2 modified' shows the files 'index.html' and 'index.js' that were changed. A red box highlights the '2 modified' section, and the text '3 Compare result' is overlaid in red.

Undo Redo Pull Push Branch Stash Pop

2 commits selected

Viewing diff between 2 commits

modify html file after change index.js 52ced5  
2018/10/4 @ 10:14 by kafai <test2atc@outlook.com>

add function myupper e8dc67  
2018/10/4 @ 09:35 by kafai <test2atc@outlook.com>

2 modified

Path Tree

index.html

index.js

**3 Compare result**

How to compare commits?

1. Click on the first commit you want to compare
2. Press “Ctrl” key, then left click on the second commit you want to compare.
3. View the compare result to see the changes, and decide the recover strategy

# ADVANCED MATERIAL

<https://gitbook.tw/chapters/introduction/about-this-book.html> (highly recommended)

[https://backlog.com/git-tutorial/tw/intro/intro1\\_1.html](https://backlog.com/git-tutorial/tw/intro/intro1_1.html)

<https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000>

Practices:

<https://www.codecademy.com/learn/learn-git>

<https://learngitbranching.js.org/>

LAST BUT NOT LEAST...

We will judge your work in  
accordance with **the latest**  
**commit in master branch on**  
**27 Oct at 5 pm sharp**



**THANK YOU!**



Q&A

