



# Practice



# Outline

- Chapter 1: Interface with Github
- Chapter 2: Other git tools



# I. Interface with Github



# Interface with Github

1. Create Github account
2. Create new repository
3. Clone
4. First commit
5. Push to Github
6. Create pull request
7. Review pull request



# 1. Create Github account

➤ Go to the [GitHub sign up page](#)

The screenshot displays the GitHub sign-up process, specifically the 'Step 1: Create personal account' stage. At the top, a progress bar shows three steps: 'Step 1: Create personal account' (active), 'Step 2: Choose your plan', and 'Step 3: Tailor your experience'. The main form area is titled 'Create your personal account' and includes three input fields: 'Username', 'Email Address', and 'Password'. Below the 'Username' field, a note states: 'This will be your username — you can enter your organization's username next.' Below the 'Email Address' field, a note states: 'You will occasionally receive account related emails. We promise not to share your email with anyone.' Below the 'Password' field, a note states: 'Use at least one lowercase letter, one numeral, and seven characters.' At the bottom of the form, a green button labeled 'Create an account' is visible. To the right of the form, a section titled 'You'll love GitHub' lists three benefits: 'Unlimited collaborators', 'Unlimited public repositories', and 'Great communication', 'Frictionless development', and 'Open source community' (each preceded by a green checkmark).



# 1. Create Github account

- Enter a username, valid email address, and password.  
Use at least one lowercase letter, one numeral, and seven characters.

## Create your personal account

### Username



This will be your username — you can enter your organization's username next.

### Email Address




You will occasionally receive account related emails. We promise not to share your email with anyone.

### Password



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

By clicking on "Create an account" below, you are agreeing to the [Terms of Service](#) and the [Privacy Policy](#).





# 1. Create Github account

- Review carefully the [GitHub Terms of Service](#) and [Privacy Policy](#) before continuing.

Upon clicking the “Create an account” button you will simultaneously be agreeing to these documents.



# 1. Create Github account

➤ Choose a plan.

Choose your personal plan



☒ Unlimited public repositories for free.

☐ Unlimited private repositories for \$7/month.

Don't worry, you can cancel or upgrade at any time.

☐ Help me set up an organization next

Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees.

[Learn more about organizations.](#)



Continue





# 1. Create Github account

## ➤ Tailor experience.

✓ Completed  
Set up a personal account

🔧 Step 2:  
Choose your plan

⚙️ Step 3:  
Tailor your experience

How would you describe your level of programming experience?

☐ Totally new to programming ☐ Somewhat experienced ☐ Very experienced

What do you plan to use GitHub for? (check all that apply)

☐ Design ☐ Development ☐ Project Management  
☐ School projects ☐ Research ☐ Other (please specify)

Which is closest to how you would describe yourself?

☐ I'm a student ☐ I'm a hobbyist ☐ I'm a professional  
☐ Other (please specify)

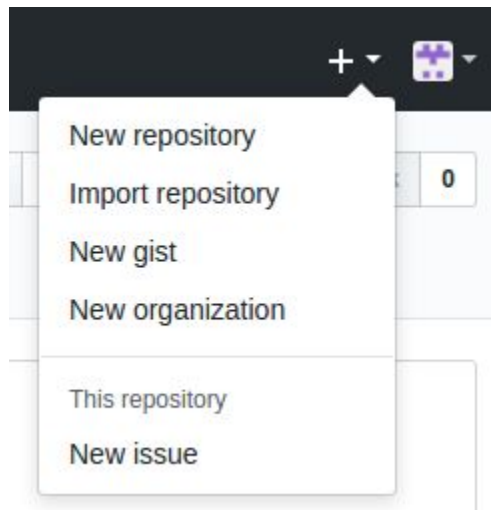
What are you interested in?

e.g. tutorials, android, ruby, web-development, machine-learning, open-source



## 2. Create new repository

- In the upper-right corner of any page, click **+** , and then click **New repository**.





## 2. Create new repository


### Create a new repository

A repository contains all the files for your project, including the revision history.

---

Owner

Repository name

 framgit ▾


 / 

training ✓


Great repository names are short and memorable. Need inspiration? How about **refactored-octo-engine**.

Description (optional)

---

☒  **Public**

Anyone can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.


---

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

 | 

Add a license: **None** ▾ 

---

Create repository



## 2. Create new repository



- In the page that create new repository, select the account you wish to create the repository on.
- Type a name for your repository, and an optional description.

Owner	Repository name
 framgit ▾	/ training ✓



## 2. Create new repository

- You can choose to make the repository either public or private. Public repositories are visible to the public, while private repositories are only accessible to you, and people you share them with. Your account must be on a paid plan to create a private repository.

- ☒  **Public**  
Anyone can see this repository. You choose who can commit.
  - ☐  **Private**  
You choose who can see and commit to this repository.
-



## 2. Create new repository

- There are a number of optional items you can pre-populate your repository with. If you're importing an existing repository to GitHub, don't choose any of these options, as you may introduce a merge conflict. You can choose to add these files using the command line later.

---

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▼

Add a license: **None** ▼





## 2. Create new repository

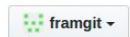
- When you're finished, click Create repository.

### Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

Repository name



/ training



Great repository names are short and memorable. Need inspiration? How about **refactored-octo-engine**.

Description (optional)

☒ **Public**

Anyone can see this repository. You choose who can commit.

☐ **Private**

You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None**

Add a license: **None**



Create repository



## 2. Create new repository

### ➤ Create a new repository on the command line

- Initialize the local directory as a Git repository.

```
$ git init
```

- Add the files in your new local repository. This stages them for the first commit.

```
$ git add .
```





# Create new repository

- Commit the files that you've staged in your local repository.

```
$ git commit -m "First commit"
```

- In Terminal, add the URL for the remote repository where your local repository will be pushed.

```
$ git remote add origin remote repository URL
```



## 2. Create new repository

- Push the changes in your local repository to GitHub.

```
$ git push origin master
```



## 3. Clone

- On GitHub, navigate to the main page of the repository.
- Under the repository name, click Clone or download.

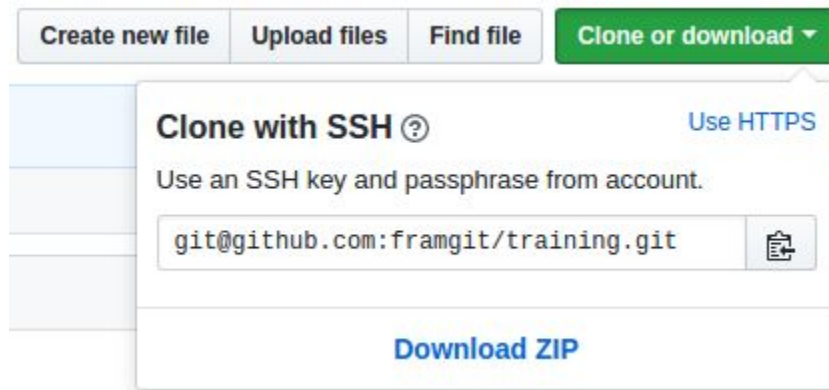
Find file

Clone or download ▾



## 3. Clone

- In the Clone with HTTPs section, click to copy the clone URL for the repository.





## 3. Clone

- Open Terminal.
- Change the current working directory to the location where you want the cloned directory to be made.
- Type `git clone`, and then paste the URL you copied in Step 2.

```
$ git clone https://github.com/your-username/your-repository
```



## 3. Clone

- Press Enter. Your local clone will be created.

```
$ git clone https://github.com/your-username/your-repository
```



## 4. First commit

- Stage the file for commit to your local repository.

```
$ git add .
```

- Commit the file that you've staged in your local repository.

```
$ git commit -m "Add existing file"
```



## 5. Push to Github

- Stage the file for commit to your local repository.

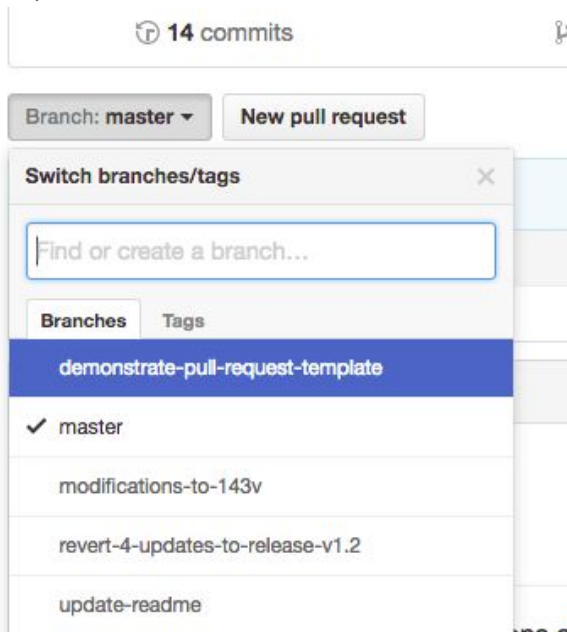
```
$ git push origin your-branch
```





## 6. Create pull request

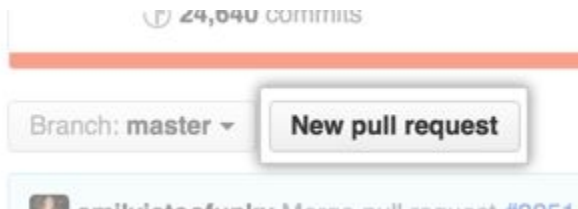
- On GitHub, navigate to the main page of the repository.
- In the "Branch" menu, choose the branch that contains your commits.





## 6. Create pull request

- To the right of the Branch menu, click New pull request.





## 6. Create 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: master ... compare: test ✓ Able to merge. These branches can be automatically merged.



## 6. Create 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

Milesto

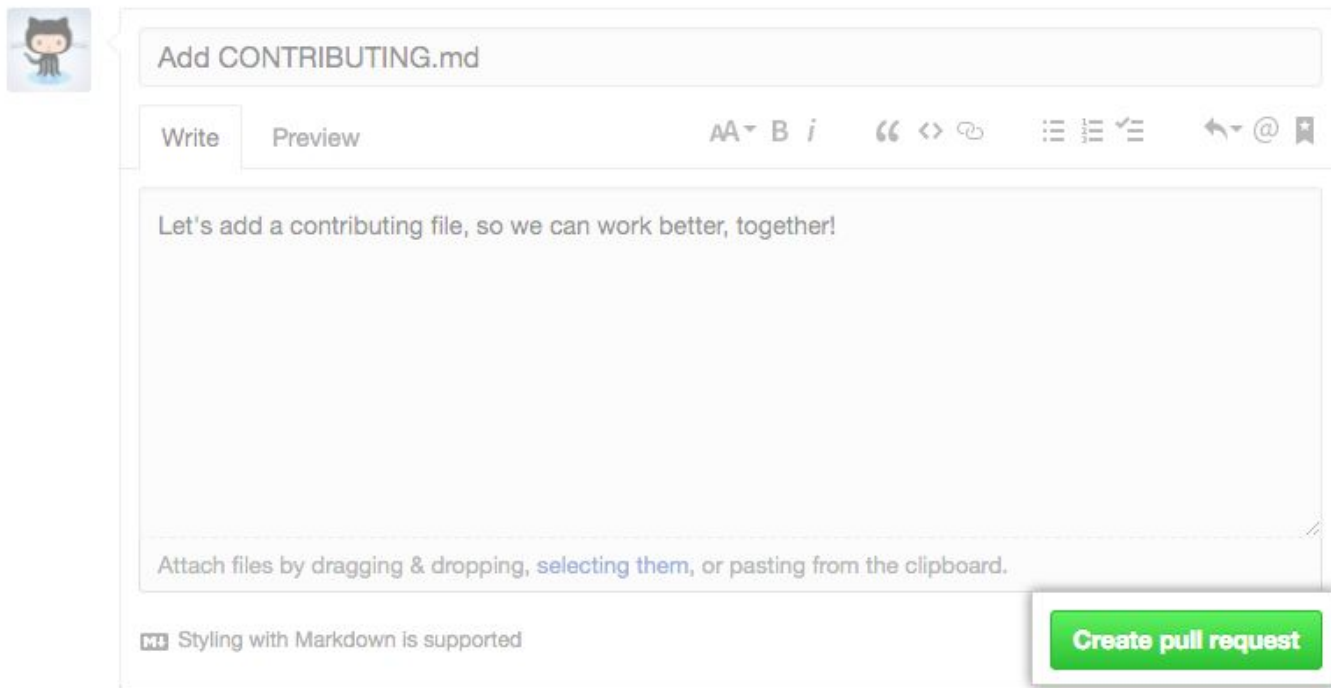
No mile

Assign

No one

## 6. Create pull request

- Click Create pull request.



The screenshot shows the GitHub interface for creating a pull request. At the top left is the GitHub Octocat logo. The main heading is 'Add CONTRIBUTING.md'. Below this is a text editor with two tabs: 'Write' (active) and 'Preview'. The 'Write' tab contains a rich text editor with a toolbar showing options for bold, italic, quote, code, link, list, and other formatting. The text area contains the placeholder text: 'Let's add a contributing file, so we can work better, together!'. Below the text area is a dashed line indicating where to attach files, with the text: 'Attach files by dragging & dropping, selecting them, or pasting from the clipboard.' At the bottom left, there is a note: 'Styling with Markdown is supported'. At the bottom right, there is a prominent green button labeled 'Create pull request'.

## 7. Review pull request



octocat requested changes 28 days ago

[View changes](#)

This is looking ✨! I've left a few comments that should be addressed before this gets merged. 🐱

data/reusables/open-source.yml

... @@ -0,0 +1,5 @@

1 +open-source-handbook-repositories: |

2 + For more information on open source, specifically how to create and grow an open



octocat 28 days ago

"provide best practices relating to creating repositories for your open source project."



## 7. Review pull request

A review has three possible statuses:

- **Comment:** Submit general feedback without explicitly approving the changes or requesting additional changes.
- **Approve:** Submit feedback and approve merging the changes proposed in the pull request.
- **Request changes:** Submit feedback that must be addressed before the pull request can be merged.



## 7. Review pull request

Unified

Split

Review changes 3

Submit your 3 pending comments

Review summary

This is looking ✨! I've left a few comments that should be addressed before this gets merged. 🐱

☐ Comment

Submit general feedback without explicit approval.

☐ Approve

Submit feedback and approve merging these changes.

☒ Request changes

Submit feedback that must be addressed before merging.

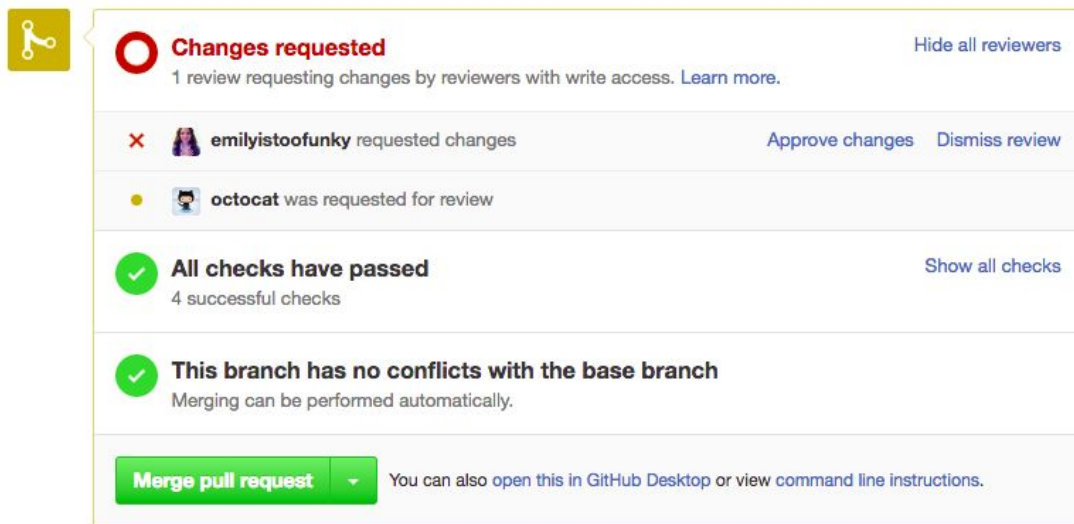
Submit review





## 7. Review pull request

- You can view all of the reviews a pull request has received in the Conversation timeline, and you can see reviews by repository owners and collaborators in the pull request's merge box.



The screenshot displays a GitHub pull request interface. At the top left is a yellow icon with a white fork symbol. The main section is titled 'Changes requested' in bold red text, with a red circle icon to its left. Below the title, it says '1 review requesting changes by reviewers with write access. [Learn more.](#)' and a link 'Hide all reviewers' on the right. Below this, there are two rows of activity: the first row shows a red 'X' icon, a user profile for 'emilyistooofunky', and the text 'requested changes', with links 'Approve changes' and 'Dismiss review' on the right; the second row shows a yellow dot icon, a user profile for 'octocat', and the text 'was requested for review'. Below the activity section, there are two green checkmark icons. The first is followed by the text 'All checks have passed' and '4 successful checks', with a link 'Show all checks' on the right. The second is followed by the text 'This branch has no conflicts with the base branch' and 'Merging can be performed automatically.' At the bottom, there is a green button labeled 'Merge pull request' with a dropdown arrow, and a note: 'You can also [open this in GitHub Desktop](#) or view [command line instructions](#).'



## II. Other git tools



# Outline

1. Gitk
2. Git gui



# 1. Gitk

- Name
- Synopsis
- Description
- Options



# 1. Gitk

- Name:
  - The Git repository browser
- Synopsis:

```
$ gitk [<options>] [<revision range>] [\--] [<path>...]
```

- Description:

Displays changes in a repository or a selected set of commits:

  - The commit graph.
  - Showing information related to each commit.
  - The files in the trees of each revision.



# 1. Gitk

- Options:
  - rev-list options and arguments
  - gitk-specific options



## 2. Git gui

- Name
- Synopsis
- Description
- Commands



## 2. Git gui

➤ **Name:**

- A portable graphical interface to Git.

➤ **Synopsis:**

```
$ git gui [<command>] [arguments]
```

➤ **Description:**

Unlike *gitk*, focuses on commit generation and single file annotation and does not show project history.





## 2. Git gui

### ➤ **Commands:**

- blame
- browser
- citool
- version



