GitHub

Date Created	@June 9, 2022 10:31 AM
Status	Research
Tags	Research and Development
Deadline	
Created By	
Engineer	
Reviewer	
Tracking	
Master Card	
Property	
Estimates	
Sprint	SPRINT#1
Property 1	
Property 2	

GitHub is a code hosting platform for version-control and collaboration. It helps to us and other work together on project from anywhere.

Types of GitHub accounts

1. **Personal accounts :-** Every person who uses GitHub.com signs into a personal account. Your personal account is your identity on GitHub.com and has a username and profile.

Your personal account can own resources such as repositories, packages, and projects. Any time you take any action on GitHub.com, such as creating an issue or reviewing a pull request, the action is attributed to your personal account.

All personal accounts can own an unlimited number of public and private repositories, with an unlimited number of collaborators on those repositories.

2. **Organization accounts**

Organizations are shared accounts where an unlimited number of people can collaborate across many projects at once.

Like personal accounts, organizations can own resources such as repositories, packages, and projects. However, you cannot sign into an organization. Instead, each person signs into their own personal account, and any actions the person takes on organization resources are attributed to their personal account. Each personal account can be a member of multiple organizations.

The personal accounts within an organization can be given different roles in the organization, which grant different levels of access to the organization and its data. All members can collaborate with each other in repositories and projects, but only organization owners and security managers can manage the settings for the organization and control access to the organization's data with sophisticated security and administrative features.

3. **Enterprise accounts**

GitHub Enterprise Cloud and GitHub Enterprise Server include enterprise accounts, which allow administrators to centrally manage policy and billing for multiple organizations and enable inner-sourcing between the organizations.

Signing up for a new GitHub account

1. If you want to create a new personal account, make sure you are currently signed out of GitHub.
2. Go to GitHub's Pricing page.
3. Read the information about the different products and subscriptions that GitHub offers, then click the upgrade button under the subscription you'd like to choose.
4. Follow the prompts to create your personal account or organization.
5. Verify your email address :- You can verify your email address after signing up for a new account, or when you add a new email address. If an email address is undeliverable or bouncing, it will be unverified.

GitHub CLI

GitHub CLI is an open source tool for using GitHub from your computer's command line. When you're working from the command line, you can use the GitHub CLI to save time and avoid switching context.

GitHub CLI includes GitHub features such as:

- View, create, clone, and fork repositories

- Create, close, edit, and view issues and pull requests
- Review, diff, and merge pull requests
- Run, view, and list workflows
- Create, list, view, and delete releases
- Create, edit, list, view, and delete gists
- List, create, delete, and connect to a codespace

Keyboard Shortcut

Typing ”?” on GitHub brings up a dialog box that lists the keyboard shortcuts available for that page.

Keyboard shortcut	Description
s or /	Focus the search bar
G + N	Go to your notifications
esc	focus on user, issue, or pull request hovercard, closes the hovercard and refocuses on the element the hovercard is in
cmd+k(mac)or ctrl+k(windows)	Opens the github command palette
G C	goto the code tab
G I	goto the issue tab
G P	goto pull request tab
G A	goto action tab
G B	goto projects tab
G W	goto wiki tab
G G	goto discussion tab
cmd+b(mac) or ctrl+b(windows)	insert markdown formatting for bolding text
cmd+I(mac) or ctrl+I(windows)	for italicizing text
cmd+k(mac) or ctrl+k(windows)	for creating a link
cmd+shift+7(mac) or ctrl+shift+7(windows)	insert markdown formatting for an ordered list
cmd+shift+.(mac) or ctrl+shift+.(windows)	insert markdown formatting for a quote

E	open source code file in Edit file tab
cmd+f(mac) or ctrl+f(windows)	Start searching in file editor
cmd+g(mac) or ctrl+g(W/L)	find next
cmd+shift+g(mac) or ctrl+shift+g(W/L)	find previous
cmd+option+f(mac) or ctrl+shift+f(W/F)	replace
cmd+shift+option+f(mac) or ctrl+shift+r(W/L)	replace all
alt+g	jump to line
cmd+z or ctrl+z	undo
cmd+y or ctrl+y	redo
cmd+shift+p	toggle between edit file and preview changes tab
cmd+s or ctrl+s	write a commit message

GitHub Command Palette

Use the command palette in GitHub to navigate, search, and run commands directly from your keyboard.

Main advantage of command palette:-

1. **Fast navigation**
2. **Easy access to commands**

Steps:

a. Open the command palette using default keyboard shortcuts: • Windows and Linux: Ctrl+K or Ctrl+Alt+K

b. start typing the path you want to navigate to.

c. **“#”** Search for issues, pull requests, discussions, and projects

d. **“!”** Search for projects

e. **“@”** Search for users, organizations, and repositories

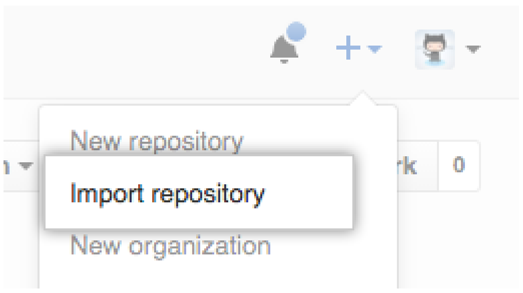
f. **“/”** Search for files within a repository scope

You can navigate, search, and run commands on GitHub with the GitHub Command Palette. You can open the command palette with a keyboard shortcut from anywhere on GitHub, which saves you time and keeps your hands on the keyboard.

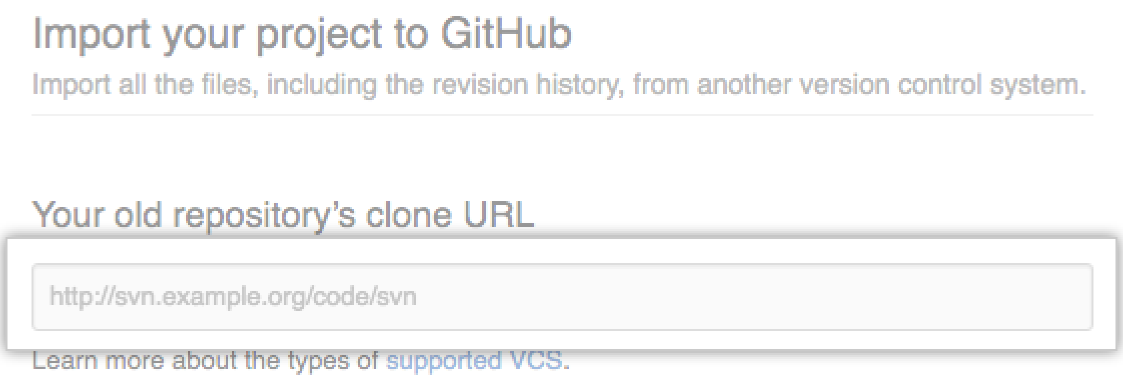
Importing a repository with GitHub Importer

If you have a project hosted on another version control system, you can automatically import it to GitHub using the GitHub Importer tool.

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



2. Under "Your old repository's clone URL", type the URL of the project you want to import.


Import your project to GitHub

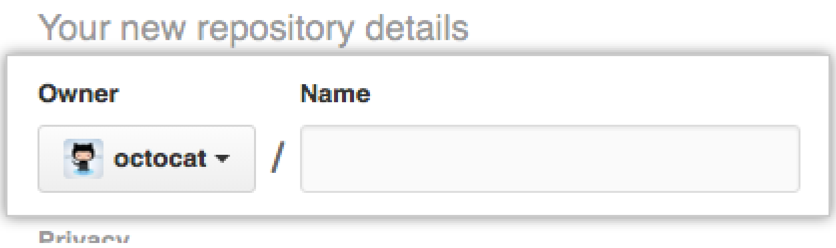
Import all the files, including the revision history, from another version control system.

Your old repository's clone URL

http://svn.example.org/code/svn

Learn more about the types of supported VCS.

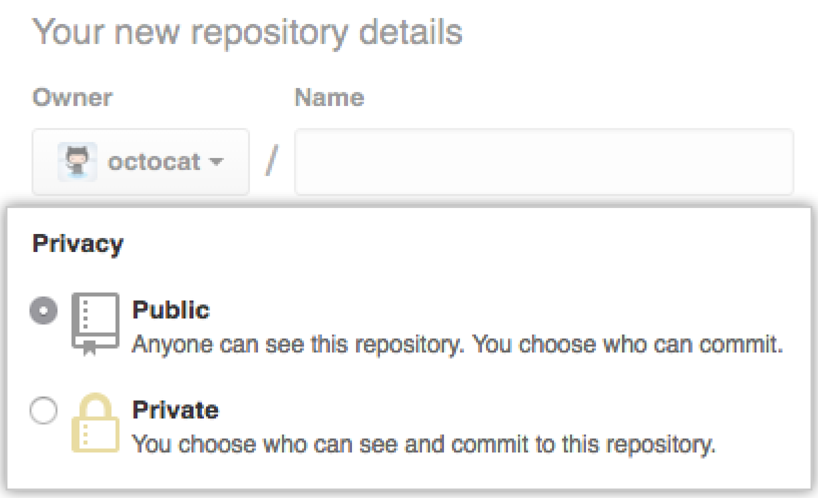
3. Choose your personal account or an organization to own the repository, then type a name for the repository on GitHub.


Your new repository details

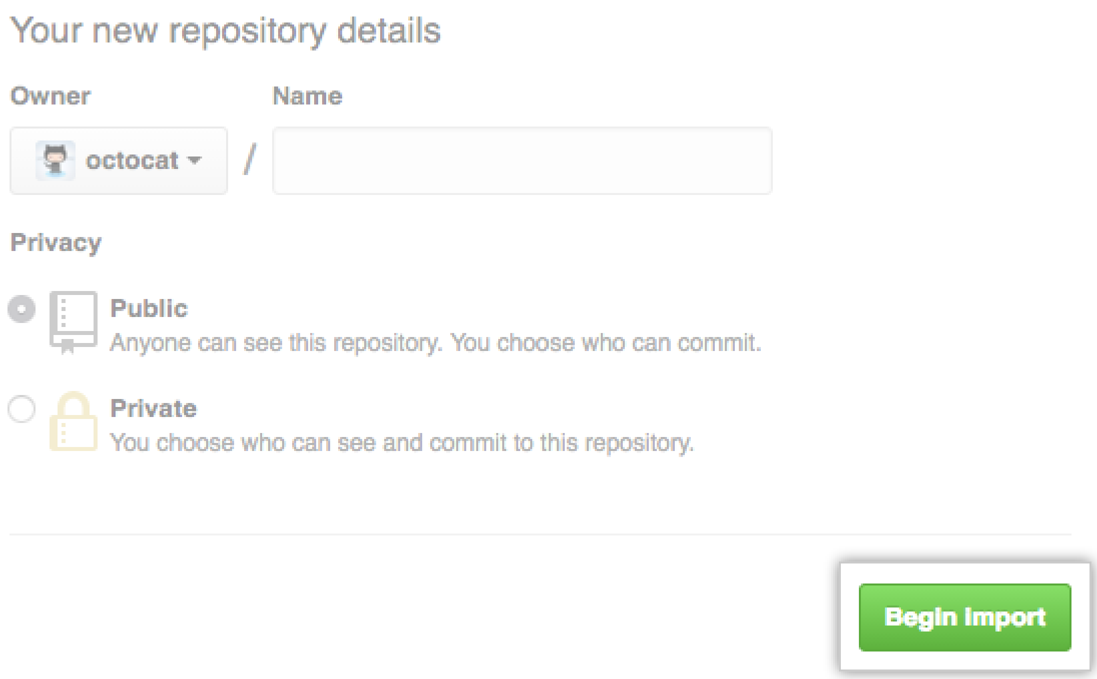
Owner: octocat / Name:

Privacy

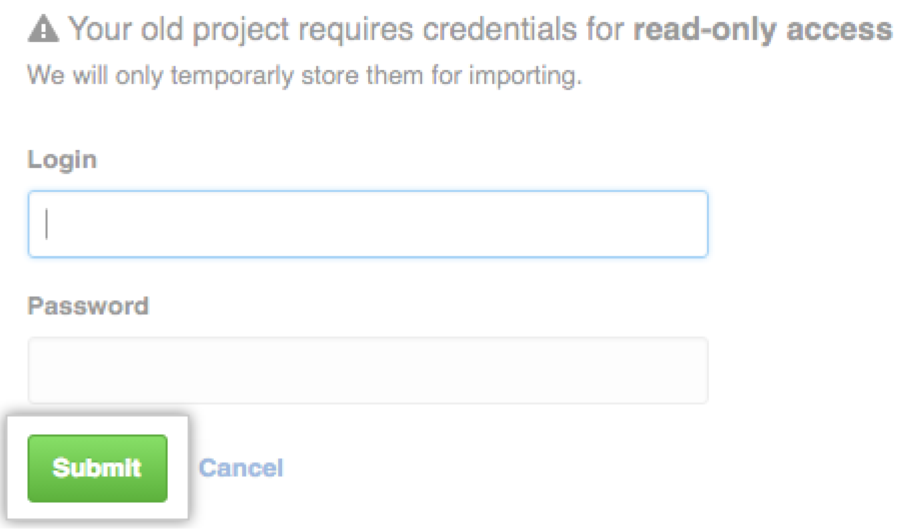
4. Specify whether the new repository should be *public* or *private*


Your new repository details
Owner
Name
octocat
/
Privacy
Public
Anyone can see this repository. You choose who can commit.
Private
You choose who can see and commit to this repository.

5. Review the information you entered, then click **Begin import**


Your new repository details
Owner
Name
octocat
/
Privacy
Public
Anyone can see this repository. You choose who can commit.
Private
You choose who can see and commit to this repository.
Begin Import

6. If your old project requires credentials, type your login information for that project, then click **Submit**
   . If SAML SSO or 2FA are enabled for your user account on the old project, enter a personal access token with repository read permissions in the "Password" field instead of your password.

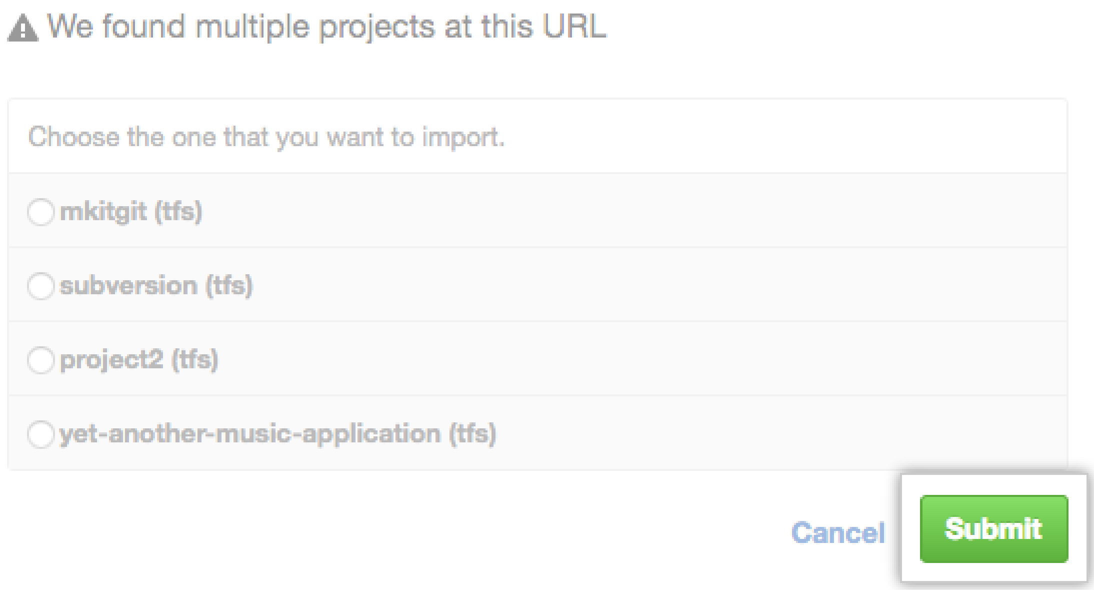

Your old project requires credentials for **read-only access**

We will only temporarily store them for importing.

Login

Password

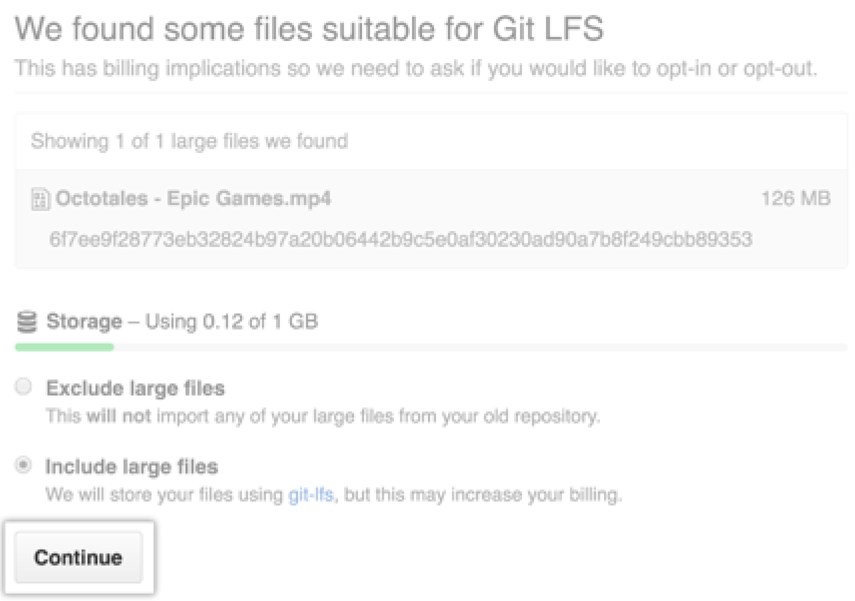
7. If there are multiple projects hosted at your old project's clone URL, choose the project you'd like to import, then click **Submit**


We found multiple projects at this URL

Choose the one that you want to import.

- mkitgit (tfs)
- subversion (tfs)
- project2 (tfs)
- yet-another-music-application (tfs)

8. If your project contains files larger than 100 MB, choose whether to import the large files using Git Large File Storage
, then click **Continue**



We found some files suitable for Git LFS

This has billing implications so we need to ask if you would like to opt-in or opt-out.

Showing 1 of 1 large files we found

Octotales - Epic Games.mp4 126 MB

6f7ee9f28773eb32824b97a20b06442b9c5e0af30230ad90a7b8f249cbb89353

Storage – Using 0.12 of 1 GB

○ **Exclude large files**
This **will not** import any of your large files from your old repository.

[✓ ● **Include large files**]
We will store your files using git-lfs, but this may increase your billing.

Importing a Git repository using the command line

1. 1. Create a new repository on GitHub. You'll import your external Git repository to this new repository.
2. On the command line, make a "bare" clone of the repository using the external clone URL. This creates a full copy of the data, but without a working directory for editing files, and ensures a clean, fresh export of all the old data.

```
git clone --bare https://external-host.com/ extuser / repo.git
```

3. Push the locally cloned repository to GitHub using the "mirror" option, which ensures that all references, such as branches and tags, are copied to the imported repository.

```
cd  repo.git
git push --mirror https://github.com/ ghuser / repo.git
# Pushes the mirror to the new repository on GitHub.com
```

4. Remove the temporary local repository.

```
cd ..
rm -rf  repo.git
```

Adding a local repository to GitHub with GitHub CLI

—> In the command line, navigate to the root directory of your project.

—> Initialize the local directory as a Git repository. `git init -b main`

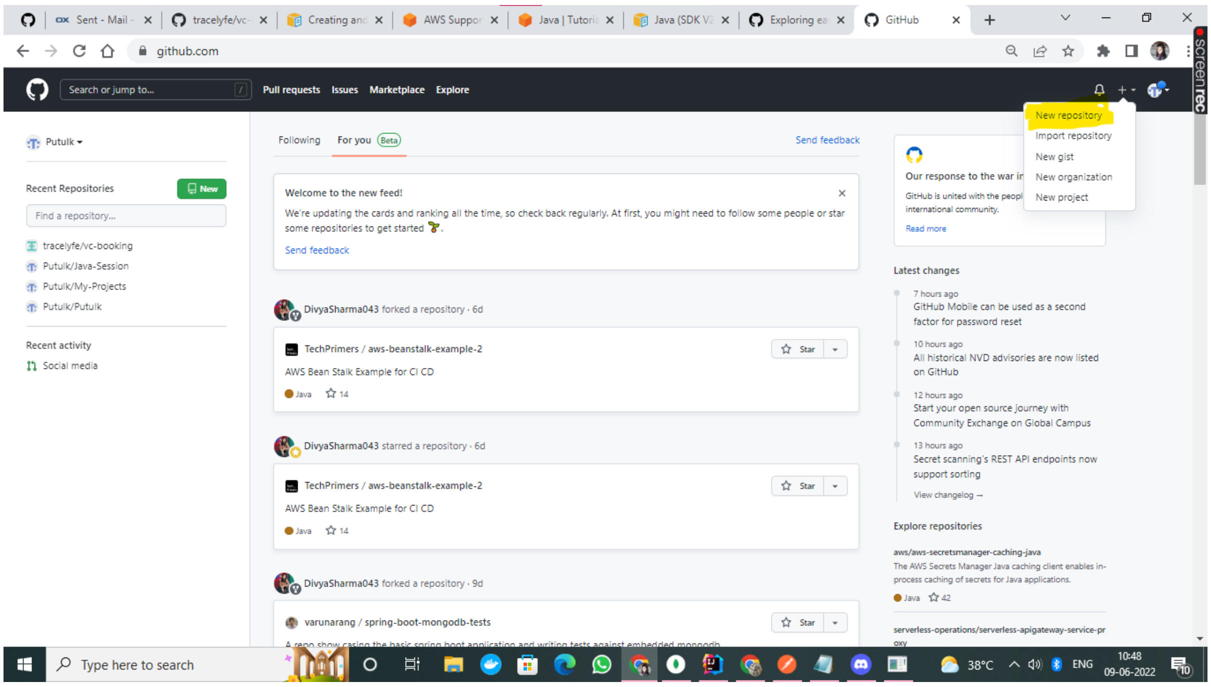
—> Stage and commit all the files in your project `git add . && git commit -m "initial commit"`

Setting your username in Git

1. Open Git Bash.
2. Set a Git username: `git config --global user.name "putulk"`
3. Confirm that you have set the Git username correctly: `git config --global user.name`

Creating a Repository

Repository usually used to organize a single project. It can contains folders and files, images, videos, spreadsheets and data sheets.



Welcome to the new feed!

We're updating the cards and ranking all the time, so check back regularly. At first, you might need to follow some people or star some repositories to get started.

DivyaSharma043 forked a repository · 6d

TechPrimers / aws-beanstalk-example-2

AWS Bean Stalk Example for CI CD

Java 14

DivyaSharma043 starred a repository · 6d

TechPrimers / aws-beanstalk-example-2

AWS Bean Stalk Example for CI CD

Java 14

DivyaSharma043 forked a repository · 9d

varunarang / spring-boot-mongodb-tests

Our response to the war in

GitHub is united with the peopl international community.

Read more

Latest changes

7 hours ago
GitHub Mobile can be used as a second factor for password reset

10 hours ago
All historical NVD advisories are now listed on GitHub

12 hours ago
Start your open source journey with Community Exchange on Global Campus

13 hours ago
Secret scanning's REST API endpoints now support sorting

View changelog →

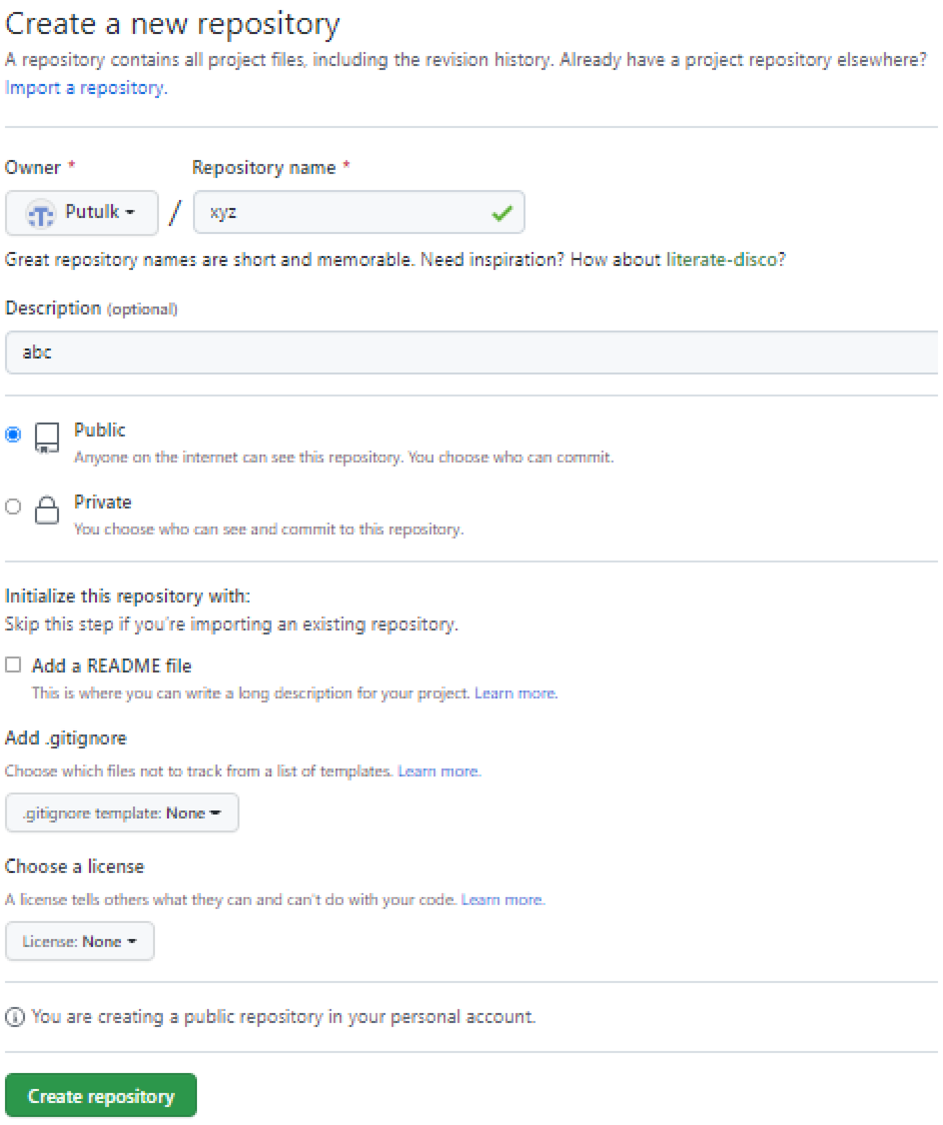
Explore repositories

aws/aws-secretsmanager-caching-java

The AWS Secrets Manager Java caching client enables in-process caching of secrets for Java applications.

Java 42

serverless-operations/serverless-apigateway-service-proxy



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * Putulk / Repository name * xyz

Great repository names are short and memorable. Need inspiration? How about literate-disco?

Description (optional)

abc

[✓ 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.
.gitignore template: None

Choose a license
A license tells others what they can and can't do with your code. Learn more.
License: None

You are creating a public repository in your personal account.

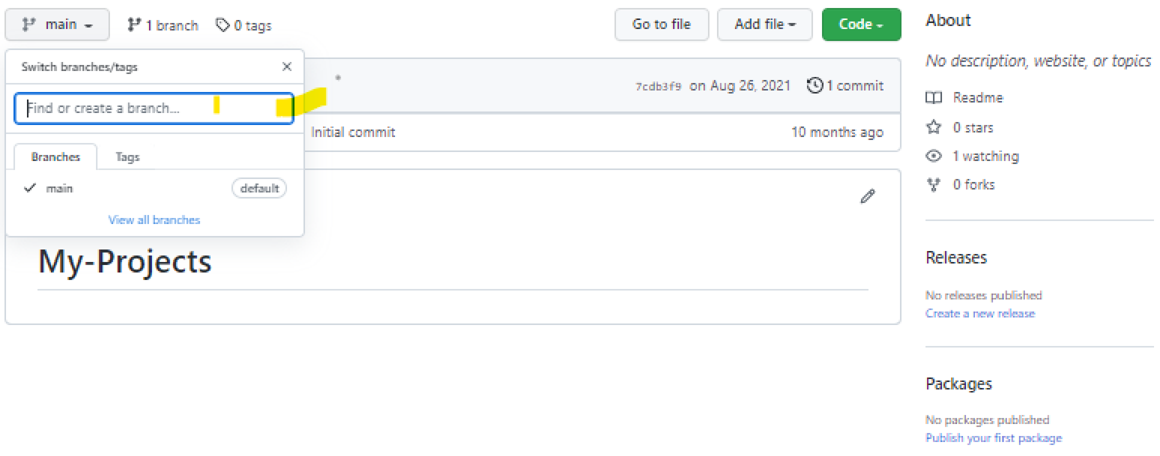
Create repository

ReadmeFile :- contain information about our project which is written in plain text.

Create Branch :- By default our repo having one branch called “main” or “master”

We can also create a new branch off of main in our repository. This is helpful when we want to add new feature to a project without changing the main source of code.

The work done in different branch is not shown in main until we merge it.



Clone a repo

1. goto the repository which you want to get
2. goto the code and copy the url for repository
3. open your ide and paste the url in vcs project

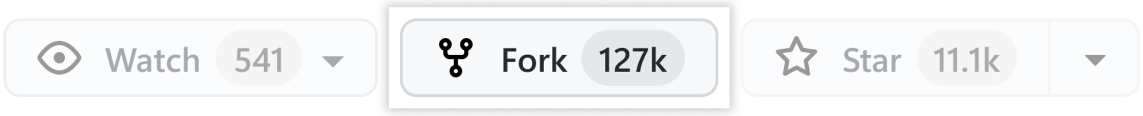
Fork a repo

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

We can use forks to propose changes related to fixing a bug. Rather than logging an issue for a bug you have found, you can:

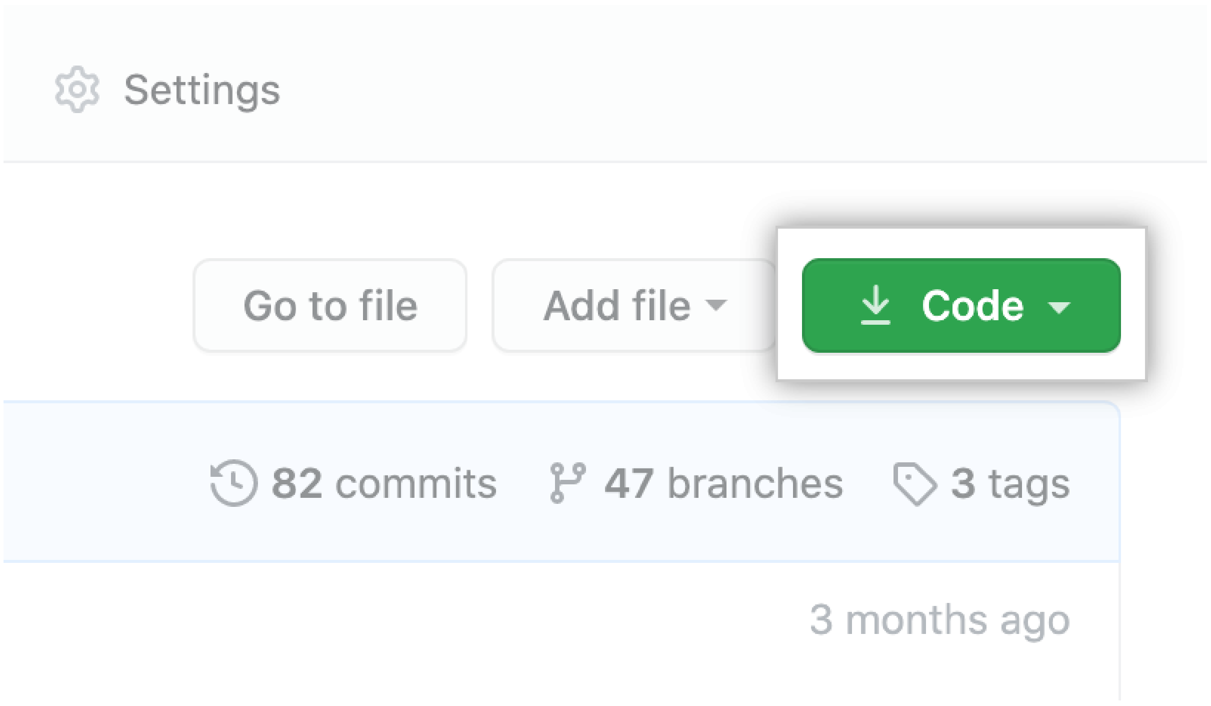
- Fork the repository.
- Make the fix.
- Submit a pull request to the project owner.

1. On GitHub.com, navigate to any repository.
2. In the top-right corner of the page, click **Fork**.



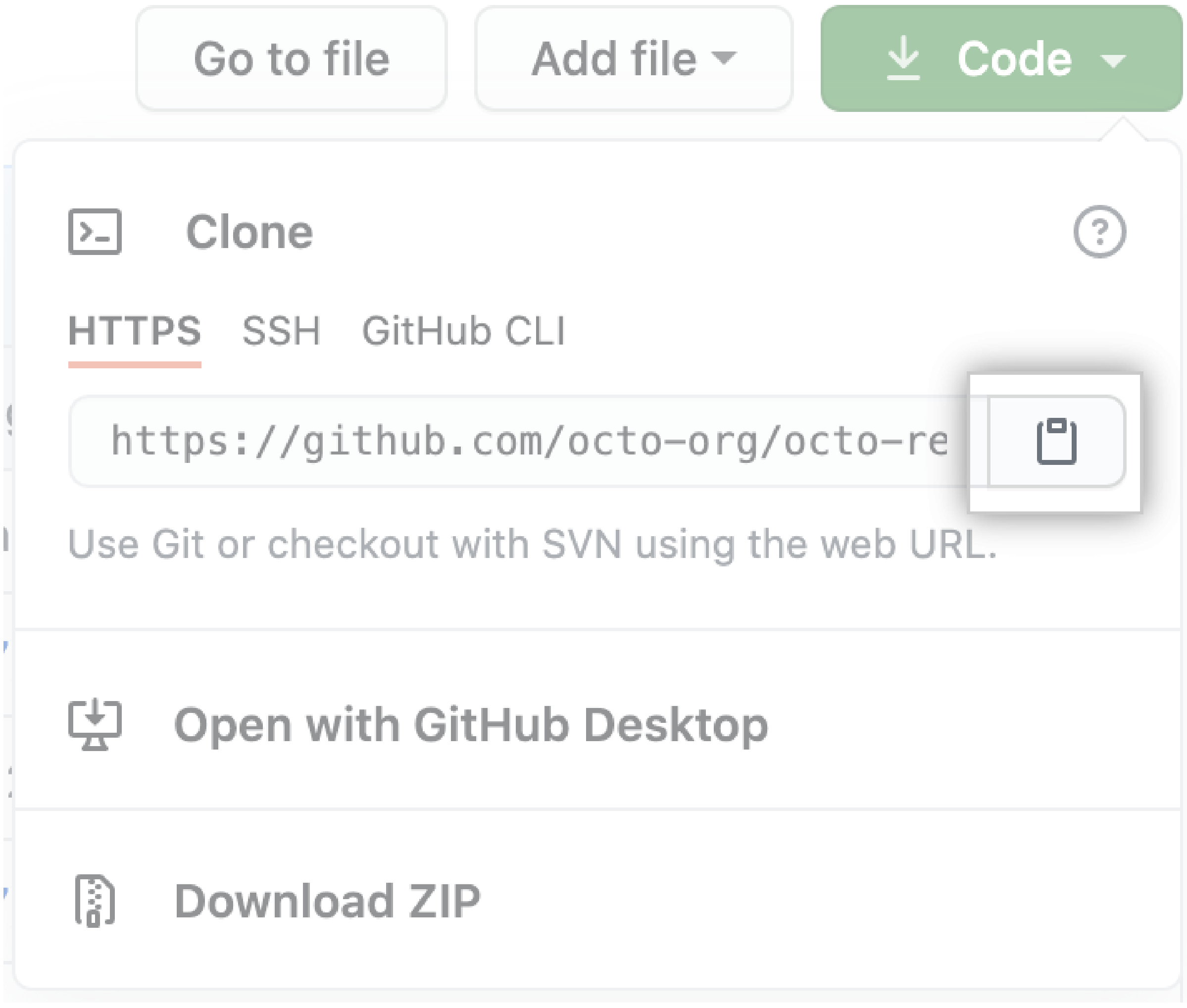
3. Right now, you have a fork of the Repository, but you do not have the files in that repository locally on your computer.

1. On GitHub.com, navigate to **your fork** of the Repository.
2. Above the list of files, click **Code**.



4. Copy the URL for the repository.

- To clone the repository using HTTPS, under "HTTPS", click .
- To clone the repository using an SSH key, including a certificate issued by your organization's SSH certificate authority, click **SSH**, then click .
- To clone a repository using GitHub CLI, click **GitHub CLI**, then click .



```
https://github.com/octo-org/octo-re
```

Use Git or checkout with SVN using the web URL.

5. Open Git Bash.
6. Change the current working directory to the location where you want the cloned directory.
7. Type `git clone` and then paste the URL you copied earlier. It will look like this, with your GitHub username instead of `YOUR-USERNAME`:

```
git clone https://github.com/ YOUR-USERNAME /repository-name
```

8. Press **Enter.** Your local clone will be created.

```
git clone https://github.com/ YOUR-USERNAME /repository-name
```

Create a pull request

Pull request are the heart of collaboration in GitHub. When we open the pull request, we proposing the our changes and requesting that someone review and pull our contribution and merge them into their branch.

Merge your pull request

Once your pull request is approved, merge your pull request. This will automatically merge your branch so that your changes appear on the default branch. GitHub retains the history of comments and commits in the pull request to help future contributors understand your changes.

GitHub will tell you if your pull request has conflicts that must be resolved before merging.

Branch protection settings may block merging if your pull request does not meet certain requirements. For example, you need a certain number of approving reviews or an approving review from a specific team.

Delete your branch

After you merge your pull request, delete your branch. This indicates that the work on the branch is complete and prevents you or others from accidentally using old branches.

Don't worry about losing information. Your pull request and commit history will not be deleted. You can always restore your deleted branch or revert your pull request if needed.

Some Important Git Commands

1. `git init -b main`

   Initialize the local directory as a Git repository.

2. `git clone https://github.com/owner/repo.git`
3. `cd repo` Change into the repo directory
4. `git branch <branch-name>` Create new branch
5. `git checkout <branch-name>` Switch to new branch
6. `git add <file1.md> <file2.md>` add file to your branch
7. `git commit -m <message>` commit changes
8. `git push -u origin <branch-name>` push changes to git hub

Contribute to an existing branch on GitHub

`git pull` origin <branch-name>

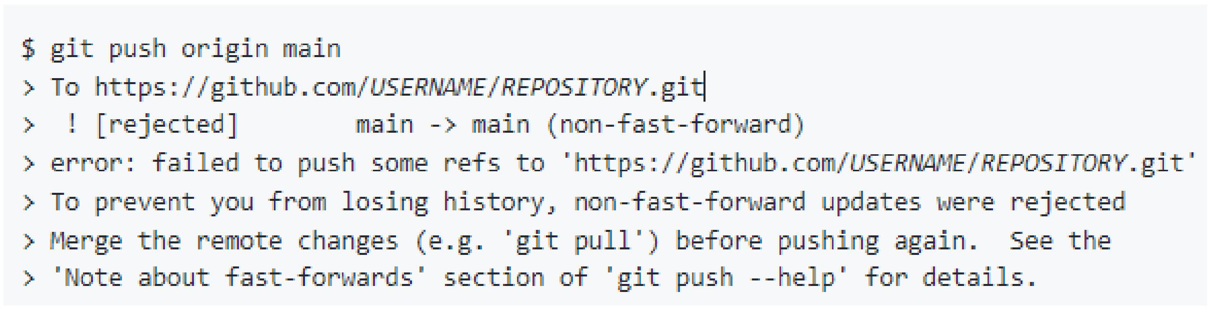
update all changes into your branch from done in another branch

`git merge` *`remotename`* `/` *`branchname`*

Merging combines your local changes with changes made by others.

Dealing with non-fast-forward errors

If another person has pushed to the same branch as you, Git won't be able to push your changes:



```
$ git push origin main
> To https://github.com/USERNAME/REPOSITORY.git
>  ! [rejected]        main -> main (non-fast-forward)
> error: failed to push some refs to 'https://github.com/USERNAME/REPOSITORY.git'
> To prevent you from losing history, non-fast-forward updates were rejected
> Merge the remote changes (e.g. 'git pull') before pushing again.  See the
> 'Note about fast-forwards' section of 'git push --help' for details.
```

You can fix this by fetching and merging the changes made on the remote branch with the changes that you have made locally:

`git fetch origin`

`git merge origin` *`YOUR_BRANCH_NAME`*

Or, you can simply use `git pull` to perform both commands at once:

`git pull origin` *`YOUR_BRANCH_NAME`*

Synchronizing with updates and changes

When a subproject is added, it is not automatically kept in sync with the upstream changes. You will need to update the subproject with the following command:

`git pull -s subtree` *`remotename`* *`branchname`*

Example : `git pull -s subtree spoon-knife main`

Git rebase

Typically, you would use `git rebase` to:

• Edit previous commit messages

• Combine multiple commits into one

• Delete or revert commits that are no longer necessary

`git rebase --interactive` *`other_branch_name`* *: To rebase all the commits between another branch and the current branch state,* `git rebase --interactive HEAD~7` : To rebase the last few commits in your current branch

There are six commands available while rebasing:

1. ***pick :-*** `pick` simply means that the commit is included. Rearranging the order of the `pick`
 commands changes the order of the commits when the rebase is underway. If you choose not to include a commit, you should delete the entire line.
2. ***reword :-*** The `reword` command is similar to `pick`, but after you use it, the rebase process will pause and give you a chance to alter the commit message.
3. ***edit :-*** If you choose to `edit` a commit, you'll be given the chance to amend the commit, meaning that you can add or change the commit entirely. This allows you to split a large commit into smaller ones, or, remove erroneous changes made in a commit.
4. ***squash :-*** *This command lets you combine two or more commits into a single commit. Git gives you the chance to write a new commit message describing both changes.*
5. ***fixup :-*** This is similar to `squash`, but the commit to be merged has its message discarded.
6. ***exec :-*** *This lets you run arbitrary shell commands against a commit.*

GitHub glossary

@mention

access token:- A token that is used in place of a password when performing Git operations over HTTPS with Git on the command line or the API. Also called a personal access token.

API preview :- A way to try out new APIs and changes to existing API methods before they become part of the official GitHub API.

assignee :- The user that is assigned to an issue.

authentication code :- A code you'll supply, in addition to your GitHub password, when signing in with 2FA via the browser. This code is either generated by an application or delivered to your phone via text message. Also called a "2FA authentication code."

base branch :- The branch into which changes are combined when you merge a pull request.

CA certificate :- A digital certificate issued by Certificate Authority (CA) that ensures there are valid connections between two machines, such as a user's computer and GitHub.com and verifies the ownership of a site.

code frequency graph :- A repository graph that shows the content additions and deletions for each week in a repository's history.

collaborator :- A collaborator is a person with read and write access to a repository who has been invited to contribute by the repository owner.