



# Git-GitHub

Part - 1

# Contents

- What is GitHub?
- Signing up for a new GitHub account
- Creating a GitHub personal account
- Creating a GitHub repository
- Authenticating to GitHub
- Set alias of remote repo and check connection
- Inviting collaborators
- Managing a branch protection rule
- Cloning a GitHub repository
- Pushing your local changes to GitHub
- Create a pull request (PR) and Merge

# What is GitHub?

GitHub is an online hosting service for Git repositories. It facilitates the sharing of codebases among teams by providing a GUI to easily download repository to a local machine. It is the largest host of source code in the world, and has been owned by Microsoft since 2018.

- 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.
  - Free private repos for educational use: [github.com/edu](https://github.com/edu)
- You can collaborate with other developers from any location.

# Signing up for a new GitHub account

GitHub offers personal accounts for individuals and organizations. There are three types of accounts on GitHub.

- Personal accounts
- Organization accounts
- Enterprise accounts

Every person who uses GitHub signs into a personal account. Each personal account uses either GitHub Free or GitHub Pro. All personal accounts can own an unlimited number of public and private repositories, with an unlimited number of collaborators on those repositories. If you use GitHub Free, private repositories owned by your personal account have a limited feature set. You can upgrade to GitHub Pro to get a full feature set for private repositories.

# Creating a GitHub personal account

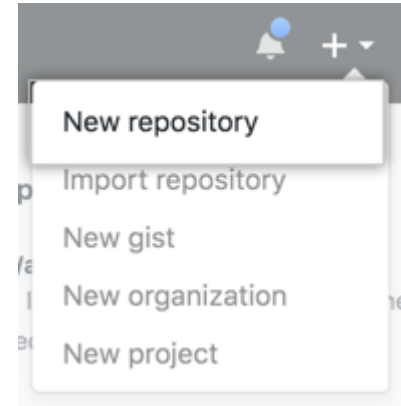
To be able to use GitHub, you will have to create an account first.

1. Go to [github.com](https://github.com) and click on Sign up
2. Enter your email
3. Provide your password
4. Enter a username
5. Like to receive product updates via email? Type "y"/"n"
6. Verify your account
7. Create account

# Creating a GitHub repository

A repository is usually used to organize a single project. Repositories can contain folders and files, images, videos, spreadsheets, and data sets -- anything your project needs. Often, repositories include a README file, a file with information about your project. GitHub lets you add a README file at the same time you create your new repository. GitHub also offers other common options such as a license file, but you do not have to select any of them now.


1. In the upper-right corner of any page, use the + drop-down menu, and select New repository.
2. In the Repository name box, enter Repo-Name.
3. In the Description box, write a short description.
4. Select Add a README file.
5. Select whether your repository will be Public or Private.
6. Click Create repository.



# Creating a GitHub repository

After clicking the button, GitHub will ask you to name your repo and provide a brief description:

Owner \*      Repository name \*


 octocat / hello-world ✓


Great repository names are short and memorable. Need inspiration? How about **ubiquitous-system**?

Description (optional)

My first repository

---

☐  **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.](#)

This will set  **main** as the default branch. Change the default name in your [settings](#).

Create repository

# Authenticating to GitHub

// To see if existing SSH keys are present, run command in Git Bash

```
$ ls -al ~/.ssh
```

// To create SSH key. When you're prompted to "Enter a file in which to save the key", you can press Enter to accept the default file location. Press enter twice for passphrase.

```
$ ssh-keygen -t ed25519 -C "ujjal.kumer@bjitgroup.com"
```

// Another way to create SSH key.

```
$ ssh-keygen -o -t rsa -C "ujjal.kumer@bjitgroup.com"
```

**Note:** If you are using a legacy system that doesn't support the Ed25519 algorithm, use:

```
$ ssh-keygen -t rsa -b 4096 -C "ujjal.kumer@bjitgroup.com"
```



# Authenticating to GitHub

// To add a new SSH key in GitHub

1. Click "Account Settings"
2. Click "SSH and GPG Keys"
3. Click "New SSH key"
4. Provide a title of your key
5. Paste your SSH key here

// To test your SSH connection

```
$ ssh -T git@github.com // Type "yes" to continue connecting
```

# Set alias of remote repo and check connection

// To rename the "master" branch in your local Git repositories

```
$ git branch -m master main
```

// To set alias for remote repository

```
$ git remote add origin https://github.com/uksaha77/bjet-b11.git
```

// To check remote connection

```
$ git remote -v
```

# Inviting collaborators

1. Ask for the username of the person you're inviting as a collaborator.
2. On GitHub.com, navigate to the main page of the repository.
3. Under your repository name, click Settings
4. In the "Access" section of the sidebar, click Collaborators & teams.
5. Click Invite a collaborator.
6. In the search field, start typing the name of person you want to invite, then click a name in the list of matches.
7. Click Add NAME to REPOSITORY.
8. The user will receive an email inviting them to the repository. Once they accept your invitation, they will have collaborator access to your repository.

# Managing a branch protection rule

1. On GitHub.com, navigate to the main page of the repository.
2. Under your repository name, click Settings.
3. In the "Code and automation" section of the sidebar, click Branches.
4. Next to "Branch protection rules", click Add rule.
5. Under "Branch name pattern", type the branch name or pattern you want to protect.
6. Optionally, enable required pull requests.
7. Click Create.

## Branch protection rules

Add rule

Define branch protection rules to disable force pushing, prevent branches from being deleted, and optionally require status checks before merging. New to branch protection rules? [Learn more](#).

enterprise-2.\*--release

Currently applies to 0 branches

Edit

Delete

update-readme-\*

Currently applies to 2 branches

Edit

Delete

# Cloning a GitHub repository

// To clone a GitHub repository to your computer with SSH

```
$ git clone git@github.com:USERNAME/REPOSITORY.git
```

```
$ git clone git@github.com:uksaha77/test_project.git
```

// To clone a GitHub repository to your computer with HTTPS

```
$ git clone https://github.com/USERNAME/REPOSITORY.git
```

```
$ git clone https://github.com/uksaha77/test_project.git
```

// To grab remote updates and merges them with your local work

```
$ git pull origin main
```

# Pushing your local changes to GitHub

// To add a new file

```
$ touch test.txt
```

// To add all files (new, modified, and deleted) in the current directory to the Staging Environment

```
$ git add --all
```

// To commit your staged update

```
$ git commit -m "Your commit message here. Refs #22333"
```

// To push changes to GitHub

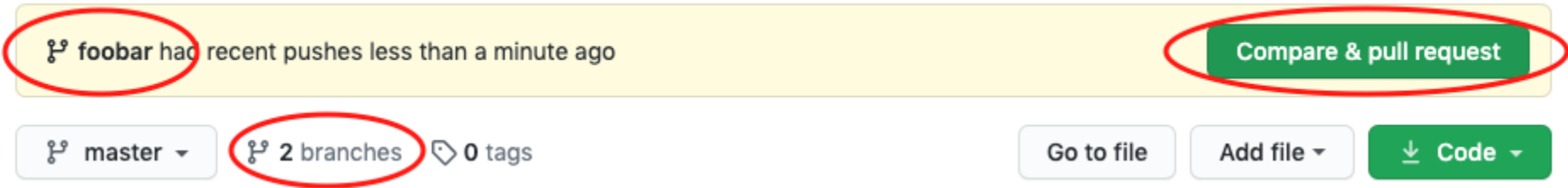
```
$ git push --set-upstream origin localbranch
```

```
$ git push -u origin localbranch
```

```
$ git push origin my-new-branch
```

# Pushing your local changes to GitHub

If you refresh the GitHub page, you'll see note saying a branch with your name has just been pushed into the repository. You can also click the 'branches' link to see your branch listed there.




# Create a pull request (PR) and Merge

A pull request (or PR) is a way to alert a repo's owners that you want to make some changes to their code. It allows them to review the code and make sure it looks good before putting your changes on the primary branch. This is what the PR page looks like before you've submitted it:

## 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: foobar








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




Add you to repo

Write

Preview

H B I  <>     @  

Leave a comment

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



Create pull request





# Create a pull request (PR) and Merge

This is what it looks like once you've submitted the PR request. You might see a big green button at the bottom that says 'Merge pull request'. Clicking this means you'll merge your changes into the primary branch.


## Add you to repo #1

 Open dfryer1193 wants to merge 1 commit into `master` from `foobar` 


Conversation 0 Commits 1 Checks 0 Files changed 1


 dfryer1193 commented now  ...


No description provided.

 Add you to repo 0e3b650

Add more commits by pushing to the `foobar` branch on `dfryer1193/newrepo`.

 Continuous integration has not been set up  
[GitHub Actions](#) and [several other apps](#) can be used to automatically catch bugs and enforce style.

 This branch has no conflicts with the base branch  
Merging can be performed automatically.

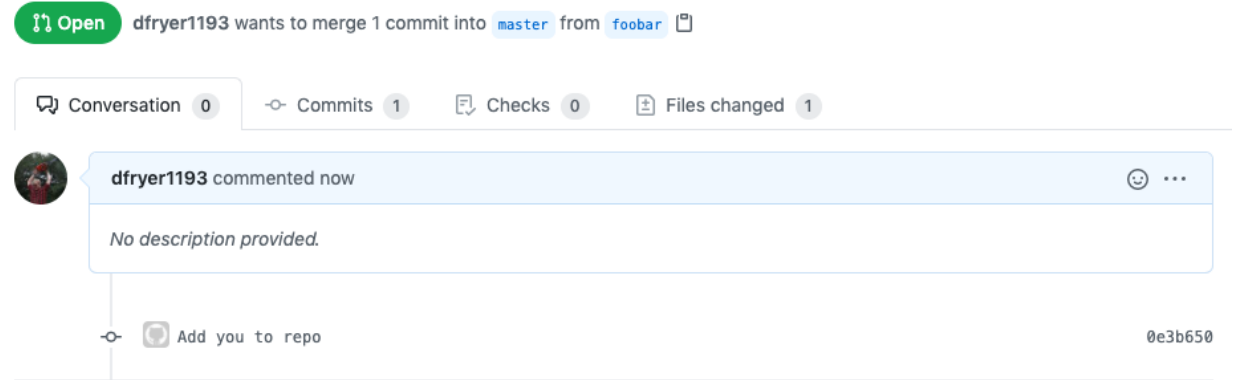
 Merge pull request You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

# Create a pull request (PR) and Merge

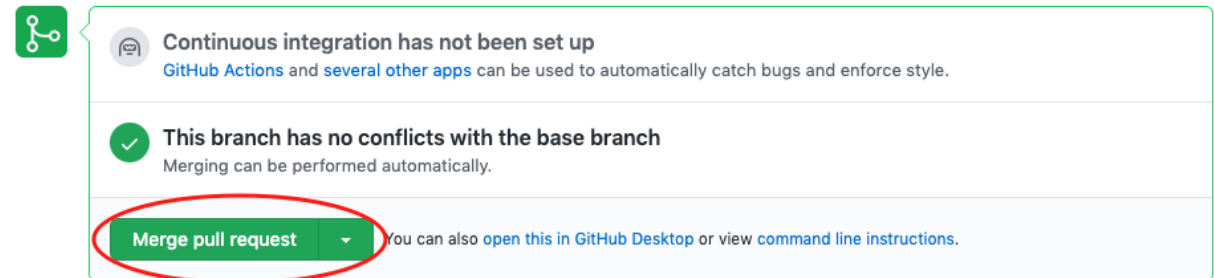
This is what it looks like once you've submitted the PR request. You might see a big green button at the bottom that says 'Merge pull request'. Clicking this means you'll merge your changes into the primary branch.

Go ahead and click the green 'Merge pull request' button. This will merge your changes into the primary branch.

## Add you to repo #1



Add more commits by pushing to the **foobar** branch on **dfryer1193/newrepo**.



# Create a pull request (PR) and Merge

When you're done, I recommend deleting your branch (too many branches can become messy), so hit that grey 'Delete branch' button as well.

## Add you to repo #1

The screenshot displays a GitHub pull request interface. At the top, a purple banner indicates the pull request has been merged: "Merged dfryer1193 merged 1 commit into master from foobar now". Below this, a navigation bar shows tabs for Conversation (0), Commits (1), Checks (0), and Files changed (1). A comment by dfryer1193, posted 1 minute ago, states "No description provided." and includes a commit link "Add you to repo" with hash 0e3b650. Below the comment, a merge summary shows "dfryer1193 merged commit 7d52acd into master now" with a "Revert" button. At the bottom, a purple-bordered box confirms the pull request was successfully merged and closed, stating "You're all set—the foobar branch can be safely deleted." and features a "Delete branch" button.

A white L-shaped graphic consisting of a horizontal line and a vertical line meeting at a right angle, positioned to the left of the "Thank you for watching" text.

Thank you for watching

A white L-shaped graphic consisting of a horizontal line and a vertical line meeting at a right angle, positioned to the right of the "Thank you for watching" text.

Get in touch with us:

[www.bjitgroup.com](http://www.bjitgroup.com)

