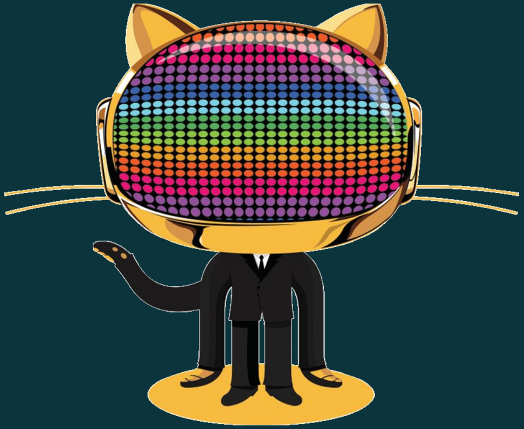
Coding Club
IIT Guwahati

Advanced Github Workshop

28 - 29 Aug

WORKSHOP STRUCTURE

- WHAT IS GIT AND GITHUB
- INSTALLATION AND SETUP
- CREATING A PULL REQUEST
- BRANCHING
- ADVANCE GIT COMMANDS
- MORE ABOUT GITHUB



Well, **GITHUB is not GIT.** Both are different Things.
Git is a VCS or Version Control System which tracks changes code changes over a period of time.

While, **Github** is a hosting platform service layered over GIT (i.e., uses GIT) for tracking the code changes but on cloud.

INTRODUCTION

- We will start this module by downloading **Git Bash** from the official website **(**https://git-scm.com/).
- After downloading launch the **Git Bash** setup. The very first page contains the important info regarding the *licenses* and *term & conditions*. After reading it click on the next button.
- The next page consists of the components that needs to be installed. Please confirm that the check box regarding *Additional icons* has been marked so that it would create a Desktop shortcut. And click the next button.
- The next page contain the info about the **Default editor** (eg. VS Code, Sublime Text, Vim etc.)that we want to set for **Git** and in which you are comfortable. After that click on the next button.
- The next page consist of the info about **adjusting the name of the initial branch in new repository**. Let it to be the default option i.e "let git decide" option. Click on next.
- After that it contains the info about the adjustment of **Path environment**. Keep this to be the recommended option (i.e Git from the command line and also from 3rd-party software). Click on next.
- In the next page choose the **Use the OpenSSL Library** option(default option) and click on next.
- In the next page choose the option (**Checkout Windows-style, commit Unix-style line endings**) and click on next.
- Again click on next after checking ***Use MinTTY*** option.Further click on next with default options till you reach *Installation* option. Get it installed and click on *Finish.*

--- We have Successfully Installed **GIT BASH---**

INSTALLATION

- Launch the installed **Git Bash** application and type ***git config*** and we will get all the functions related to git configuration will appear. We can read through them.
- Now type **git config --global user.name "Your UserName"** and then press enter. This would configure your Username.
- Now type **git config --global user.email "Your email-id"** and then press enter. Now this would configure your email-id.
- Now if we want to check whether our name and email-id has been configured successfully or not, we can simply type **git config user.name** and **git config user.email** to check our configured name and email respectively.
- If we want to see all the details, type **git config --list** and all the details would be available on the screen. After that if we want to get out of that simply press **q**-button and that's all.

----- We are now ready to move ahead with other functions -----

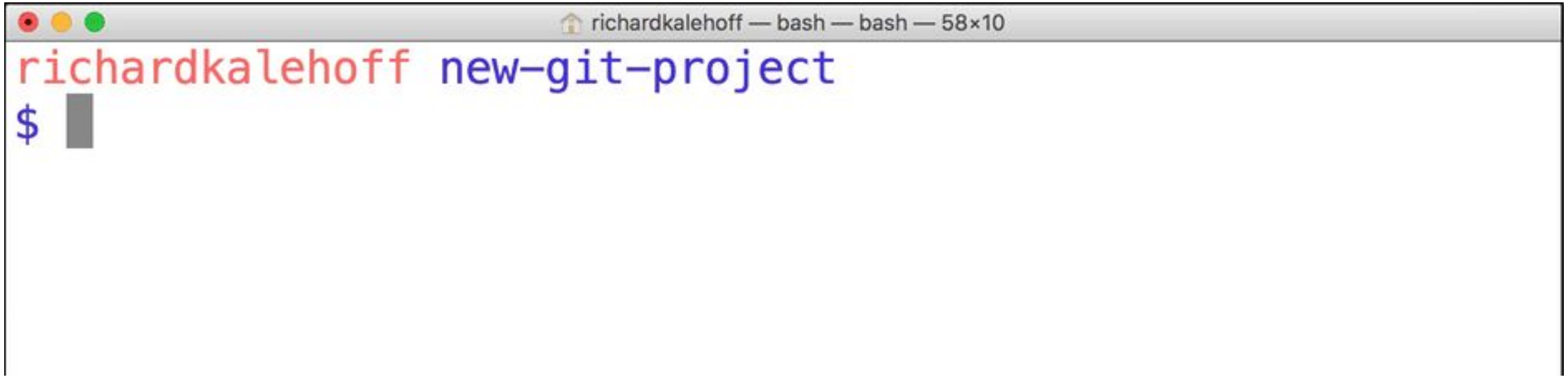
CONFIGURATION

- cd command: Allows you to traverse to other directories.
- mkdir command: Allows you to make directory.
- touch command: Allows you to make file.
- Nano: A Command Line Text Editor for editing files in bash. (You can also use vim, but for beginners nano is user friendly)

Further Commands will be told in the tutorial ahead.

BASIC BASH COMMANDS AND TOOLS

- Inside your Project Folder type: “git init”   to initialize git repository.
- It will make a .git folder inside your Projects Folder.
- .git folder is where, all your tracking data goes.
- To check the current status of git repo type: “git status”



```
richardkalehoff new-git-project
$ 
```

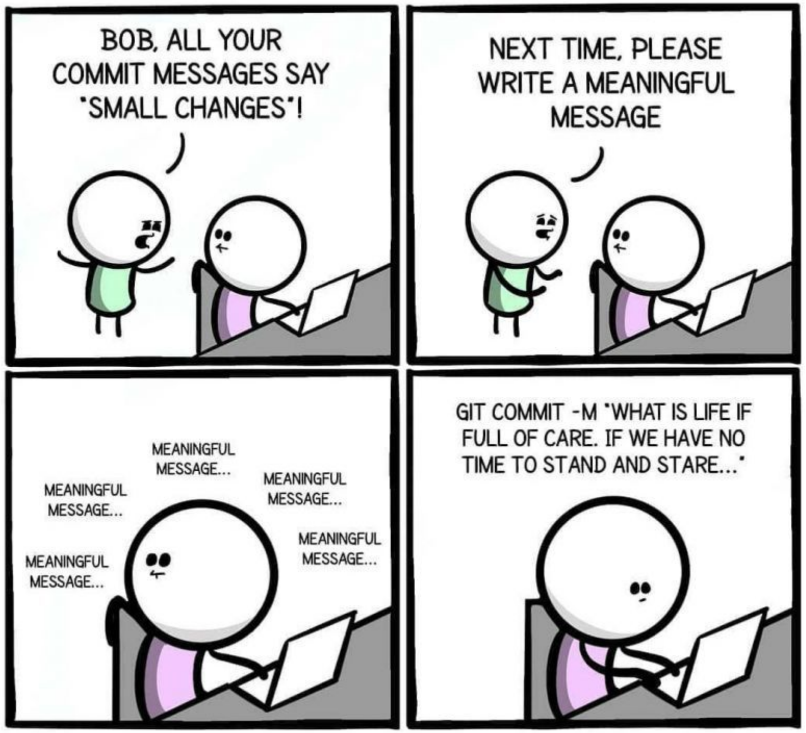
Initializing git repo

- The Staging Area is when git starts tracking and saving changes that occur in files.
- You tell git that I want to track these specific files, then git says okay and moves them from you Working Tree to the Staging Area and says “Cool, I know about this file in its entirety.”
- Type: “git add <filename>”   (For adding a single file to staging area)
- Type: “git add .”   (For adding all the files which you changed)

(Here . means the current directory)

STAGING AREA

- Now comes the committing part.
- If all thing is good you may want to commit changes to release the new version of your project.
- To do this we will use: “git commit” command
- Type: “git commit -m <message>”
- Here take care of the message part as it should be concise, clear and short.


BOB, ALL YOUR COMMIT MESSAGES SAY "SMALL CHANGES"!
NEXT TIME, PLEASE WRITE A MEANINGFUL MESSAGE
MEANINGFUL MESSAGE...
MEANINGFUL MESSAGE...
MEANINGFUL MESSAGE...
MEANINGFUL MESSAGE...
MEANINGFUL MESSAGE...
GIT COMMIT -M "WHAT IS LIFE IF FULL OF CARE. IF WE HAVE NO TIME TO STAND AND STARE..."

COMMIT CHANGES

- Let’s suppose you want to revert back to last version of the file as the current version is buggy or you made some serious mistake.
- So first type: “git diff <filename>”   (diff shows you the previous version of the file)
- And now to move back to last version type:

  “git checkout <filename>”

  And the file will revert back to the last version.

REVERTING BACK TO PREVIOUS VERSION

- There are many remote repo hosting service platforms of which some are: Github, Gitlab, Bitbucket, etc. (You can even host your own)
- The main idea about remote repo is that you can collaborate to other users, more backup safety, etc.
- As a starting point you can create a Github Account.
- Get yourself familiarise by exploring some

  of its features.

WORKING WITH REMOTE REPOSITORIES

- Go to your project folder(where you have git initialised)
- Add your github repo link as remote for pushing files to github repo
- By convention we add a github repo url remote as “origin” name (Although you can name it anything)
- Type the command: “git remote add origin <your github url>”
- Now to push all your commits to the repo type:

  “git push -u origin <your branch name>”

ADDING FILES TO REMOTE REPO

- The most important part of a git repository is a README.md file
- This file summarises about the project and helps other user to understand about your code
- .md refers to Markdown File
- Some basics of Markdown:

  # Text -> Main Heading Level

  ## Text -> 2nd Heading Level

  Google for detail cheatsheet

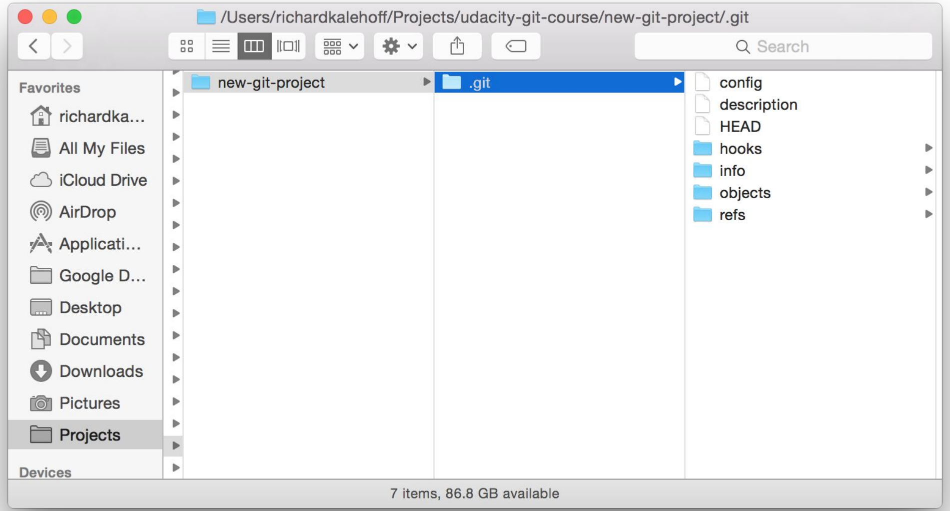
README

- Let’s suppose you are developing a webapp which employs various APIs. For accessing those APIs you will surely be using some Private Access Keys for accessing those APIs.
- Let’s say you kept all these API Keys in a file. You surely will not want git to track that file. Here’s where .gitignore comes into play.
- .gitignore tells git about the files which it should not track.
- Then add the files name (with extension) which you don’t want to get Tracked. And git will not track those files.
- https://github.com/github/.gitignore

KEEPING YOUR SECRETS PRIVATE (.gitignore)

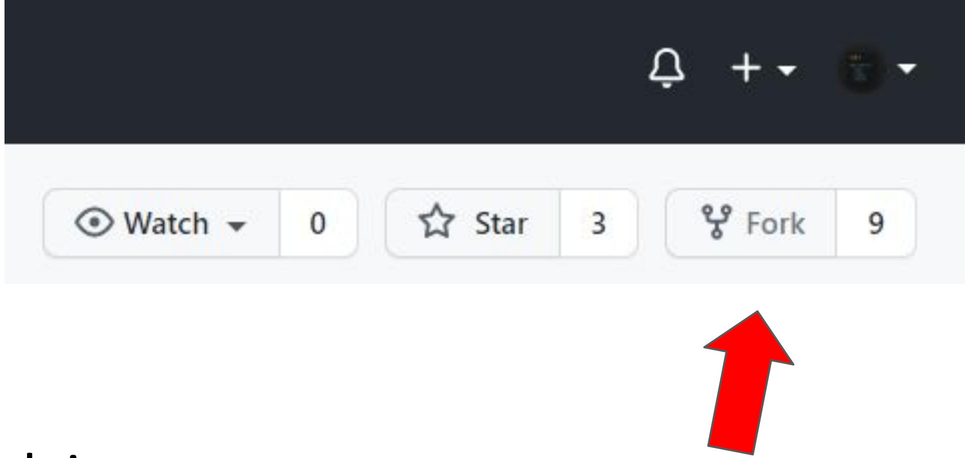
- Cloning is making a copy of the repo on your local machine
- For Cloning type: “git clone <your-remote-git-repo-url>”
- You will get the copy of the remote repo on your local machine

CLONING



CREATING A PULL REQUEST

- Pull Request basically is a request for merging your code changes to someone else codebase.
- For doing this we first need to fork it (https://github.com/GeekTuts/First_PR)
- Go the github repo where you want to contribute
- You will find a fork option on top right area
- Click on it to fork it

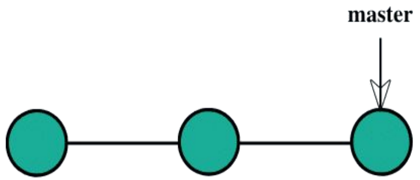


CREATING A PULL REQUEST

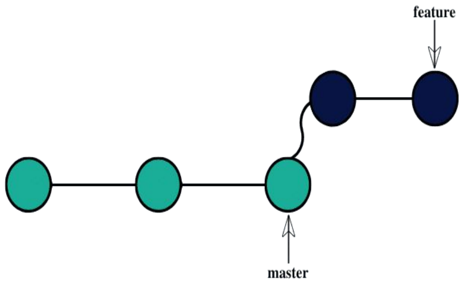
Why Branching? Let's see an example!

Developing a Project through branching:-

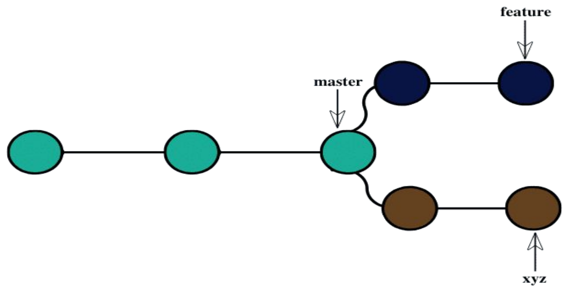
- Suppose you are working on a project for a client

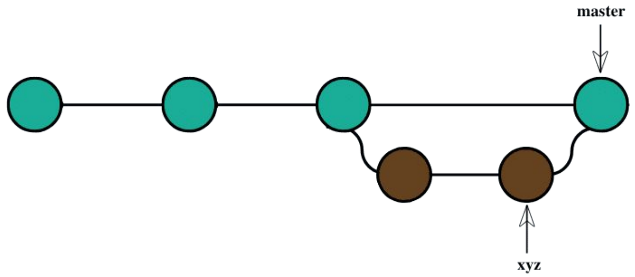


After this you decide to develop a feature and create a New branch feature and start working.



Since you had some time left you decided to work on one more feature names "xyz". You now show it to client And he disproves the "feature" but allows the "xyz" . Now as you used branching you just need to remove the "feature" branch and merge the "xyz" branch into master.





Branches give you the freedom to independently work on different modules (*not necessarily though*) and merge the modules when you finish developing them. It might sound a cumbersome process, but git branches are swift to be created and destroyed. Just a simple command can perform these processes, and they are very cheap, considering the size they take. Branches in Git help the team, which are in different parts of the world, work independently on independent features that would ultimately combine to produce a great project. Moreover, the branches are very flexible. Using branches does not mean you are using them for different features.

BRANCHING

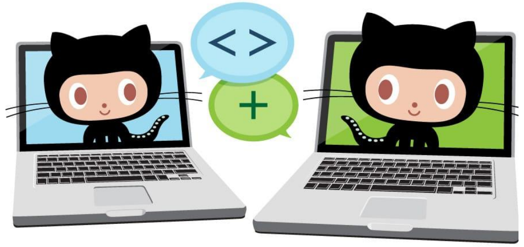
Operations in Git Branches

- The `git branch` Command
  - To list all branch name in the repository
  - Create a branch
    - `git branch <branchname>`
- The `git checkout` Command
  - To switch between different branches
    - `git checkout <branchname>`
- Delete a branch
  - `git branch -d <branchname>` (will show error if you are on that branch)
  - `git branch -D <branchname>` (force deletes the branch)

Some more Operations in Git Branches

- To clone a specific branch
  - `git clone -b <branchname> <repo_url>`
- The create + checkout Command
  - `git checkout  -b <branchname>`
- Push to a specific branch
  - `git push -u origin <branchname>`
- Merging two branches (Note : you should be on the branch in which you want to merge a specific branch)
  - `git merge <branchname>` (here branchname is the branch you want to merge)

Collaborating with GitHub



To collaborate with multiple people working on the same project, there are two major options:

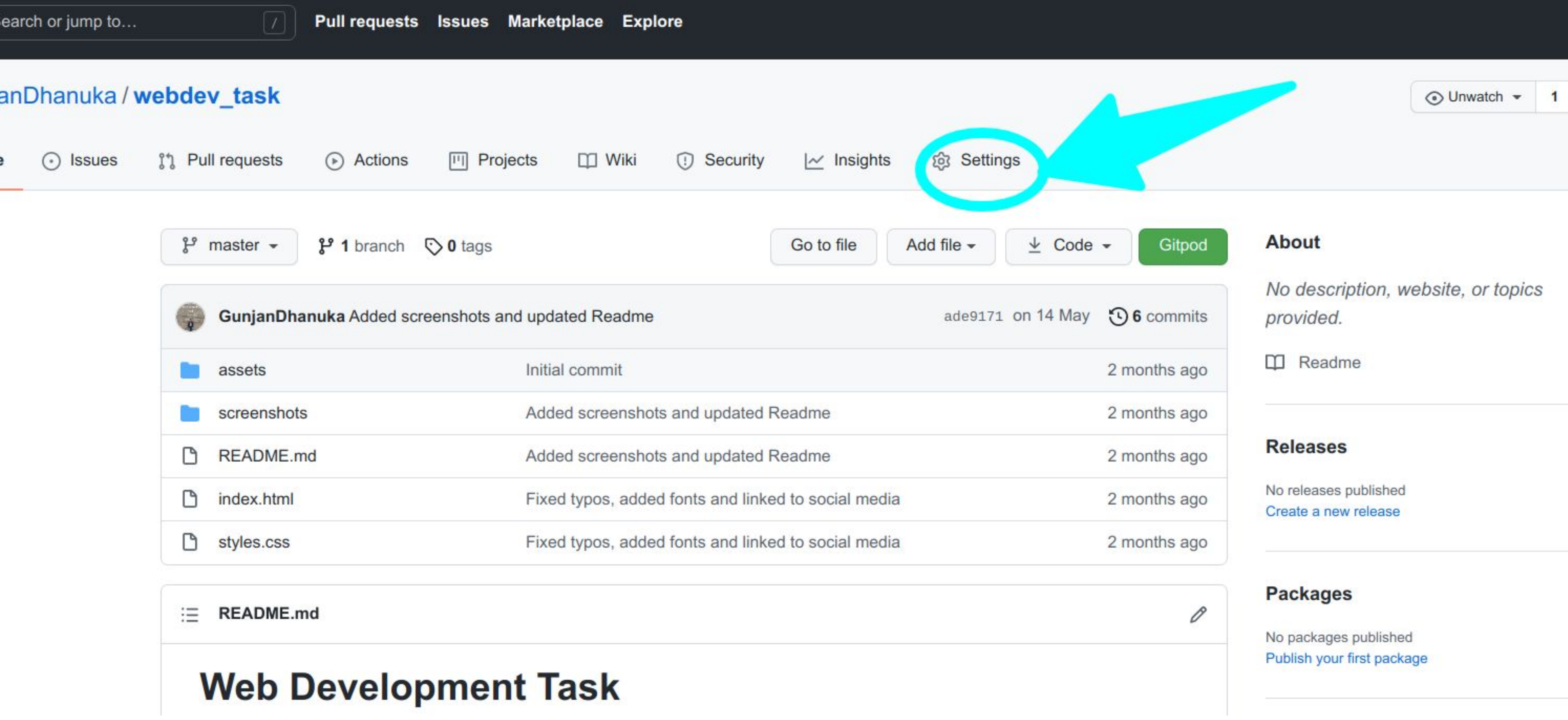
1. Add COLLABORATORS (If you are the owner of the repo)
2. Fork + Pull Request (If you want to someone else's repo)

1. As the owner of a repo you:

- add people as collaborators
- each collaborator can read/write files in the repo
- each collaborator is adding files and other content → making branches → and either directly merging changes in or via pull requests

YOU ADD COLLABORATORS IF THEY ARE A CORE PART OF THE TEAM!

MORE ABOUT GITHUB



GunjanDhanuka Added screenshots and updated Readme ade9171 on 14 May 6 commits

assets	Initial commit	2 months ago
screenshots	Added screenshots and updated Readme	2 months ago
README.md	Added screenshots and updated Readme	2 months ago
index.html	Fixed typos, added fonts and linked to social media	2 months ago
styles.css	Fixed typos, added fonts and linked to social media	2 months ago

README.md

Web Development Task

About

No description, website, or topics provided.

Readme

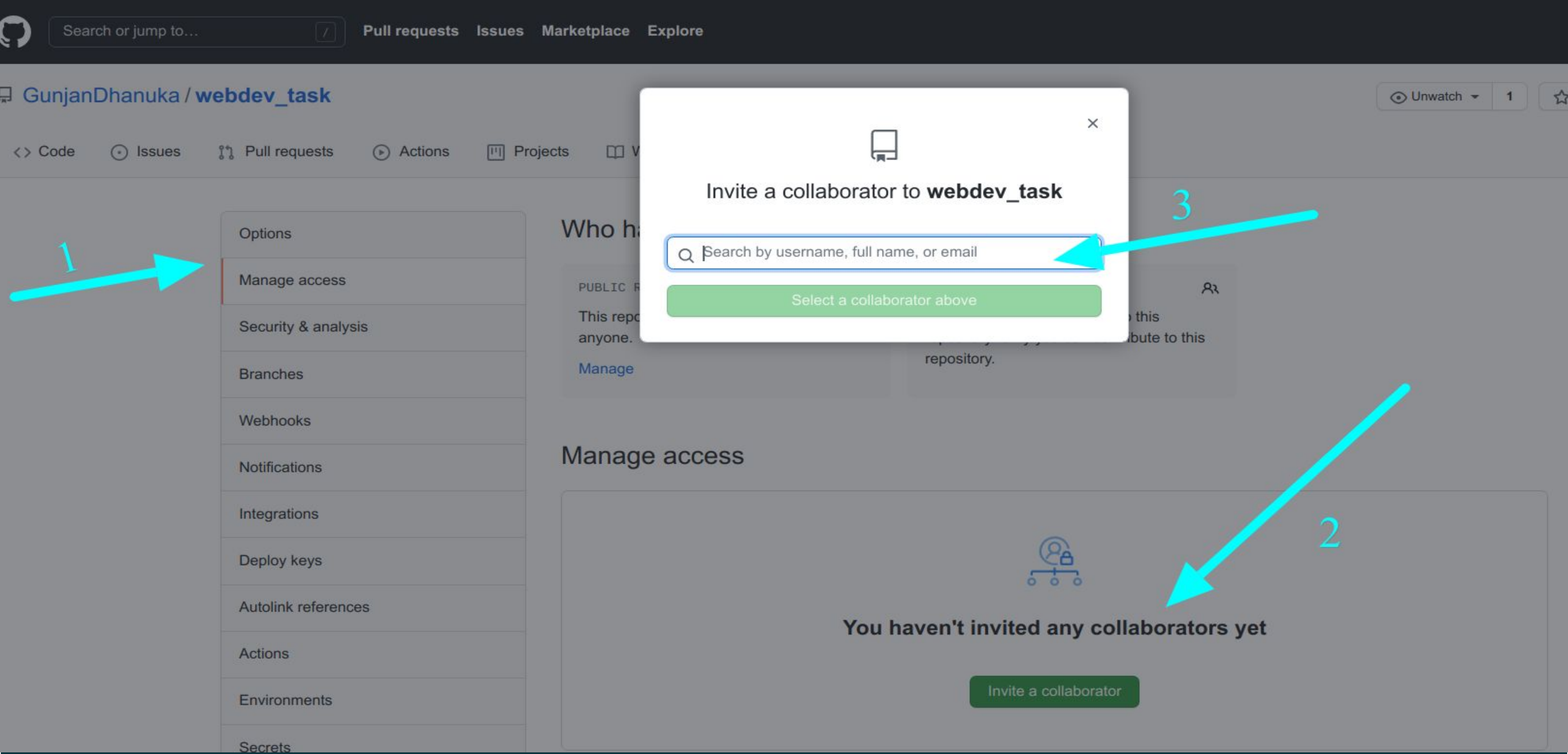
Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

MORE ABOUT GITHUB



MORE ABOUT GITHUB

2. If you don't own a repo/ not a official contributor:

a. You will FORK a repo.

b. Work and make changes on the forked copy of the repo.

c. Make a PULL REQUEST inorder to make your changes accepted into the original repo.

d. The owner of the original repo will check if your changes are helpful - and MERGE them in!

YAY YOU NOW HAVE SOME COOL INTERNET FRIENDS!

MORE ABOUT GITHUB

Starring and Following:

a. You can star the repositories you find interesting; so you may come back to them later for contributing or just using them in your own projects.
b. Follow the people whose progress and activity you want to follow on GitHub. You willl see the repos they starred, people they followed and you get to learn a lot.

GITHUB IS THE SOCIAL NETWORK FOR CODERS!

GitHub Profile README:

a. It is a special README file that shows whenever someone opens your GitHub profile.
b. An awesome place to tell people about yourself, your interests, projects and just basically leave an impression onto the visitor.
c. Can include a variety of stuff like Badges, Stats, various cards like your latest blogs, the songs you listen to, your email-id and other cool things to show-off!

How to create Profile README?

a. Create a repository with the exact same name as your GitHub Username. Eg: GunjanDhanuka
b. Then create a README.md file inside that repo and start editing it in GitHub itself.
c. https://rahuldkjain.github.io/gh-profile-readme-generator/ to create beautiful README's very quickly. Then edit as per your likings.

THE README IS YOUR CANVAS!!

Cool things to add to Profile README!

a. Add your Medium blogs: https://github.com/bxcodec/github-readme-medium-recent-article/

b. Show your music taste: https://github.com/kittinan/spotify-github-profile

c. Flex your GitHub stats (my favourite part about Profile README ;)): https://github.com/anuraghazra/github-readme-stats

d. Many more cool stuff to add (emojis, GIFs, skill badges, social media handles): https://towardsdatascience.com/build-a-stunning-readme-for-your-github-profile-9b80434fe5d7/

Portfolio website using GitHub Pages:

a. You get your own domain to host your personal website with your GitHub account. All for free!

b. Very easy to deploy and it supports everything from basic HTML, CSS based websites to full-fledged React Apps.

c. You have your own personal website that re-deploys whenever you make any changes in the code and push it to the repo. No hassles!

Create personal website on GitHub Pages:

a. Create a repository with the name: [Your Git User Name].github.io. For ex: gunjandhanuka.github.io
b. Download a template for your website and make changes to it or push your own files to the above repository. (*The entire process of cloning, committing and pushing to remote origin has been shown earlier in the workshop!*).
c. As soon as you push, you will be able to see the deployment in a few minutes. Also you can deploy your own projects as well, for ex: https://gunjandhanuka.github.io/corona-tracker-app/. Just push your React repository to GitHub and then go to settings and deploy it at your preferred domain-name. Make sure to make a production build of the app and push it to the repo on GitHub. More on this: https://github.com/gitname/react-gh-pages/

WE WILL HOST A LIVE DOUBT SESSION FROM 6 P.M. TODAY (i.e. 28 August 2021) ON OUR DISCORD CHANNEL.

WE WILL RELEASE A DETAILED VIDEO REGARDING THE TASKS TOMORROW (i.e. 29 August 2021) MORNING.

DOUBT SESSION AND TASKS

MEET THE TEAM

OPEN SOURCE MODULE

ATISHAY JAIN

GUNJAN DHANUKA

ADITYA RAJ

HARSH AGRAWAL

SHUBHAM PATHAK

AMRESH P SINHA

CODING CLUB
IIT GUWAHATI

THANK YOU FOR JOINING