git 101: git and github for everyone

by subhajit

we’re gonna learn:

- what is git and github (briefly)
- basic git terminologies
- hands-on experience with git and github
    - part 1: start your own project
    - part 2: contribute to a existing github project

what is git and github

- git is a free, open-source distributed version control system (vcs) to manage and keep track of source code changes
    - vcs (version control system) is a system that keeps track of changes made to a file or a set of files.
- created by Linus Torvalds in 2005 for the development of the Linux
- Github is a git repository hosting platform.

basic git terminologies

basic git terminologies : **repository**

- often shortened to ‘repo’
- a collection of all the files and their history
- can live on local machine (you computer) or on remote server (github)
- act of making a copy of a repository from remote server to local machine is called ‘cloning’
- think of it as a main folder/directory that contains some files.

basic git terminologies : **branch**

- branch is a separate/new version of main repo
- it allows to work on different part of the project without impacting the main branch .
    - helps to build new feature without breaking existing features.
- command:

    create a new branch:

    ```
    git branch <branch-name>
    ```

    switch to a branch:

    ```
    git checkout <branch-name>
    ```

basic git terminologies : **commits**

- the commit command saves the changes to the local repo
- puts the changes into staging area
    - staging area is where your changes live before you push them into remote repo.
- command :

```
git commit -m <commit-message>
```

basic git terminologies : **pull**

- get latest updates from remote repo
- it’s like refresh button. it grabs the latest changes from the shared project
- command :

```
git pull

git pull origin <branch-name>
```

basic git terminologies : **push**

- Uploads contents/changes from local repo to remote repo
- command :

```
git push origin <branch-name>
```

basic git terminologies : **fork**

- a fork is a new repository that shares code and visibility settings with the original.
- you cannot always make a branch or pull an existing branch and push back to it, because you are not registered as a collaborator for that specific project.
- you can work on your ideas in isolation.

Let’s do some hands-on

part 1: start your own project

- **create a repo on github**
- clone the repo
- make some changes to the main branch
- push the changes to github
- create a new branch
- Make more changes to new branch
- Push the changes to the newly created branch

1. in the upper-right corner of any page, use the **+** drop-down menu, and select new repository.
2. type a short, memorable name for your repository.
3. choose a repository visibility. public or private
4. select initialize this repository with a readme.
5. click create repository.

part 1: start your own project

- create a repo on github
- **clone the repo**
- make some changes to the main branch
- push the changes to github

- create a new branch
- Make more changes to new branch
- Push the changes to the newly created branch

1. navigate to the main page of the repository.
2. above the list of files, click <> Code.
3. copy the URL for the repository, under "HTTPS"
4. on terminal, type **git clone**, and then paste the URL you copied earlier

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

5. press Enter to create your local clone.

part 1: start your own project

- create a repo on github
- clone the repo
- **make some changes to the main branch**
- push the changes to github

- create a new branch
- Make more changes to new branch
- Push the changes to the newly created branch

1. Move to the cloned project repo
   cd <dir-name>
2. Open the code repo in vscode
3. Make some changes to the existing file, or create a new file.
4. Make some changes to the newly created file, if you have created one.

part 1: start your own project

- create a repo on github
- clone the repo
- make some changes to the main branch
- **push the changes to github**

1. Check the status of your repo
   git status
2. Add everything to staged area
   git add .
3. Commit everything that you have in your staging area
   git commit -m “<commit-message>”
4. Push everything to main branch
   git push origin main

- create a new branch
- Make more changes to new branch
- Push the changes to the newly created branch

part 1: start your own project

- create a repo on github
- clone the repo
- make some changes to the main branch
- push the changes to github

1. Create a new branch
   git branch <branch-name>
2. Switch to the newly created branch
   Git checkout <branch-name>

- **create a new branch**
- Make more changes to new branch
- Push the changes to the newly created branch

part 1: start your own project

- create a repo on github
- clone the repo
- make some changes to the main branch
- push the changes to github

- create a new branch
- **Make more changes to new branch**
- Push the changes to the newly created branch

1. Move to the cloned project repo
   cd <dir-name>
2. Open the code repo in vscode
3. Make some changes to the existing file, or create a new file.
   a. Changing existing file
4. Make some changes to the README.md,

part 1: start your own project

- create a repo on github
- clone the repo
- make some changes to the main branch
- push the changes to github

- create a new branch
- Make more changes to new branch
- **Push the changes to the newly created branch**

1. Check the status of your repo
   git status
2. Add everything to staged area
   git add .
3. Commit everything that you have in your staging area
   git commit -m “<commit-message>”
4. Push everything to main branch
   git push origin <branch-name>

part 2: contribute to a existing project (DIY)

- Fork a repo on github
- clone the repo
- create a new branch
- Make more changes to new branch
- Push the changes to the newly created branch