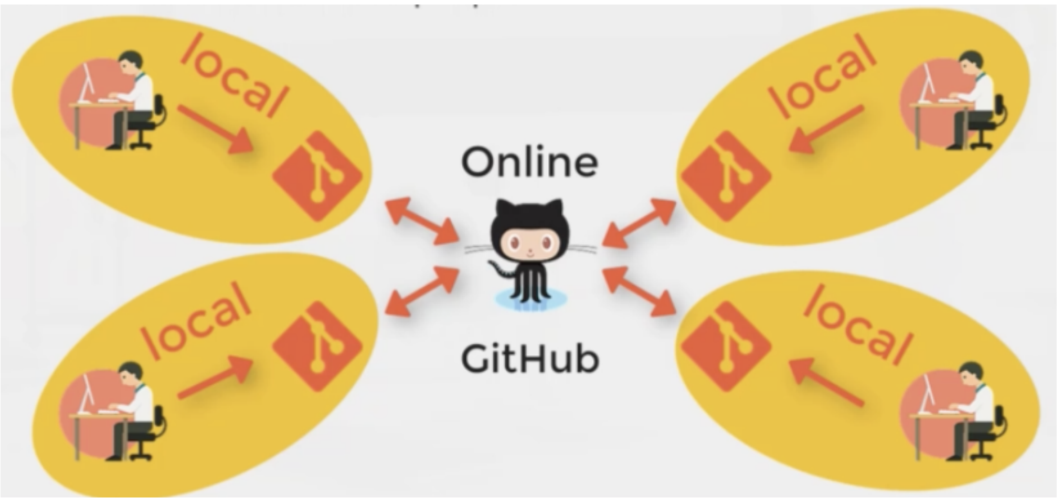
Git with GitHub

GitHub

- GitHub is an online service where you can share your code or files in order to collaborate with different people.
- To put in simple words, GitHub would have installed in one centralized machine where ever developers would commit their code from their local machine.



Difference between Git & GitHub

- Often people get confuse with Git & GitHub.

Git :

- As we know Git is an version control tool which will allow us to manage project locally.
- Additionally, we can say git helps us to push & pull the project files to an central server.

GitHub

- GitHub is an hosting platform which allows you to host your project files in an remote server.
- Beside, we can also consider GitHub is an user friendly public platform with nice visual interface and millions of users sharing their projects to entire world.

Some of more hosting tools are GitLab, Bit-Bucket etc..

- If you are new to GitHub, please go-ahead and create your own account in GitHub with your mail ID.

Create first repository in GitHub

- Navigate to GitHub page and login with your account.
- Click the tab “Repositories”
- Click “New”
- Provide the ”Repository name”
- Choose if it should be “Public or Private”
- You can also add ReadMe files which can help others to understand your project.
- Create the Repository once all done.

Overview of GitHub dashboard

- Overview of what can be done after creating new repo.
- How to create folders and files in GitHub.
- How to make commit after creating file in GitHub
- How to download a file to local machine
- Delete the file.
- How to find the branch which we are into.
- Other widgets like (watch, star, Fork, Settings etc..)

Pushing Local Repo to GitHub

- In our case, we have already created the local repository. We just need to push it to GitHub.
- To push files to remote server we may need to first setup the remote in local server.

```
git remote add origin <git-link>
```

(origin is an optional name here, you can create remote with any name)

- Once remote is set our local repo is linked with remote GitHub.

```
git push origin master
```

(Right now we are just pushing our codes to Master branch)

- Provide your credentials.

Manage GitHub credentials in environment

- Every time when you push the file to GitHub, you may need to provide your GitHub credentials which is bit annoying. Let’s see how to avoid it by storing credentials in as an environment variable.
- For that we are going to use credential helper.

```
git config credential.helper store
```

```
git push -u origin main
```

```
cat ~/.git-credentials
```

Making change in GitHub and pulling the changes to local

- In this scenario, let’s make the change in file at GitHub.
- This change will not reflect in local repo however. Let’s see how to pull the changes from GitHub to local repo.

- Your local repo can be updated using command.

  ```
  git pull origin master
  ```

Branches

- As you know default branch in every repository is Master (Main) branch.
- Your “master” branch contains the original code of project.
- Sometime you may need to add new feature to your project at the same time you don’t want to touch or break the current state of your project.
- In such cases the best solution would be creating new branch and commit all your modified project contents over there.
- Once you tested your new feature and find it working fine, you can go-ahead and merge it with your “Master” branch.
- You can create many number of branches as per requirement.
- This approach would be helpful when several developers are working in same project and each of them can work on different branch without breaking the master code.

Create Branches

Local

To check what branch we are working on

```
git branch
```

Create new branch

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

To switch between branches we are going to checkout command.

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

Checking

```
git branch (To check which branch we are into)
```

```
git log (To check Head is pointing to which branch)
```

Note :

- Git is going to inherit all the files from master to new branch once it is created.
- Once after branch created, make commit & check if its not making any change in master.

GitHub

Create branch in GitHub GUI.

Push & Pull your local branch to GitHub

Push your local branch to GitHub.

- First switch to branch

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

Make required modifications with files

```
git add <file-name> (stage the file)
```

```
git commit –m <message>
```

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

Pull branch from GitHub to local machine.

- Create branch manually in GitHub (GUI)
- Add couple of files in it.
- Create the same branch locally.

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

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

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

Delete branch

GitHub

- Click branch tab in GitHub.
- Go-ahead click delete option.

Delete branch in GitHub from command line

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

Local

- To delete branch, you can just run branch command with string -D

```
git branch –D <branch name>
```

Remember, you should not be in same branch when you are trying to delete it.

Merge

- In the real world, though you are working in different branches you may need to combine it with **”Master”** branch at one point in time.

- This process is called **“Merging Branches”**

Create new branch and commit code

```
git branch dev
git checkout dev
git add
git commit -m ”Version 1”
```

Now you can see the difference between the commits of your Master and Dev branch

Create pull request

- Once commit is done, you can see new branch and create pull request in GitHub.
- Ask your team to verify and give the feedback about new commit.
- If things are fine you can goahead and merge your code.
- You can also merge your branch with Master from command line.

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

Merge conflicts

- Solving conflicts are more like, when two persons modified the same line in an file.
- Step 1 : Modify some line in code at Master branch
- Step 2 : Checkout to dev branch and Modify the same line
- Step 3 : Try merging both branched now

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

  ```
  git merge dev
  ```

- Step 4 : Solve the conflicts before proceeding.

- Step 5 : While solving the conflicts you can compare the conflicts and either accept any one of the change.
- Step 6 : Once conflict is solved, go-ahead and add and commit the files again and then do merge.

  ```
  git add .
  ```

  ```
  git commit
  ```

  Command to abort merge if incase required “git merge –abort”

Git Rebase

Git Rebase is going to help us to copy the new changes from one branch to other.

Step 1 : Add some additional feature to your Master branch code.

Step 2 : Commit the changes

Step 3 : Checkout to dev branch

Step 4 : You might not find the changes which are recently made in Master.

Step 5 : To update all changes & updated code use rebase command as below

```
git rebase <branch from where the update code should be copied>
```

Git difference

- Diff command is going to be really help to understand what have modified in your file.
- It will just display the difference between the recent modification with last commit.

  ```
  git diff <branch-1> <branch-2>
  ```

  ```
  git difftool
  ```

Git Fork

- Using Fork you can create the copy of original repo. But this repo remains in your local account.
- For example : You want to contribute to some open source community, where as you cannot directly modify the code of originator.
- Instead, you can fork the identical of that repo to you machine and do whatever changes which remains within your account.

Git clone

- Using clone option you can download the complete repo to your local machine

  ```
  git clone <link>
  ```