Git and GitHub

Lizen Shakya

What is Git?

- Git is a Version Control System (VCS)
- Designed to make it easier to have multiple versions of a code base, sometimes across multiple developers or teams
- Maintains history
- It allows you to see changes you make to your code and easily revert them.
- It is NOT GITHUB!

What is Github, then?

- GitHub is a for-profit company that offers a cloud-based Git repository hosting service.
- It makes it a lot easier for individuals and teams to use Git for version control and collaboration.
- Github.com is a website that hosts git repositories on a remote server.
- Hosting repositories on Github facilitates the sharing of codebases among teams by providing a GUI to easily fork or clone repos to a local machine

Cloud-based Git repository hosting service

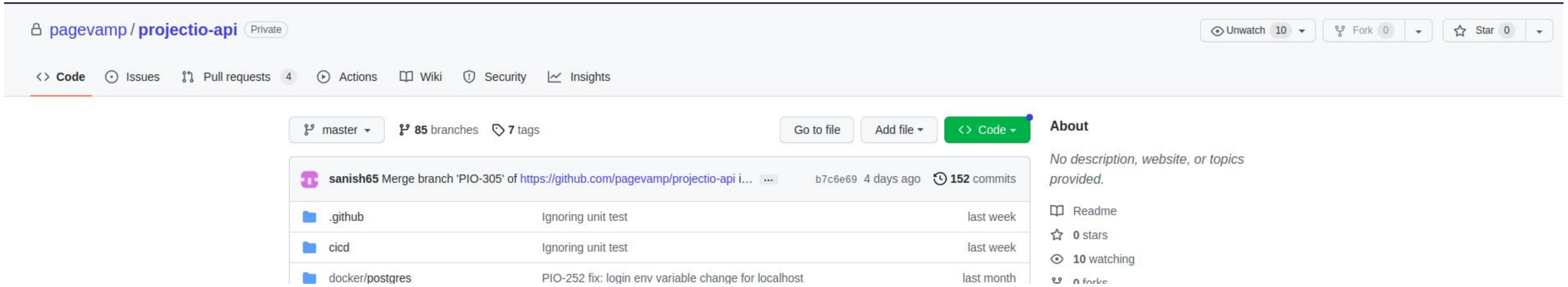
Exploring the github interface…

From here, you can view the various branches that are being worked on, as well as when someone made a commit (*this is kind of like "saving" a file*). Depending on how a repository is set up, you also might be able to create your own branch and make your own commits there.

And once you made some changes, you could submit that code back to a branch by making a pull request. A pull request is basically asking the person in charge of the branch to include your code. And it also helps that person see exactly what you've changed in the code.

Exploring the github interface…

If you wanted to edit some or all of the source code on your own account on a more permanent basis, you could also fork it by clicking the Fork button (a fork is similar in concept to a branch, but a fork is more permanent):



sanish65 Merge branch 'PIO-305' of https://github.com/pagevamp/projectio-api i… b7c6e69 4 days ago 152 commits

.github	Ignoring unit test	last week
cicd	Ignoring unit test	last week
docker/postgres	PIO-252 fix: login env variable change for localhost	last month

Install git…

- Open your terminal and run ‘git’
- If you see a ‘command not recognized’ error, you probably haven’t installed git yet.
- To solve this issue check the website below:

https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Configuring git

Your commits will have your name and email attached to them. To confirm that this information is correct, run the following commands:

```
$ git config --global user.name
```

> should be your name, i.e. Jon Rosado

```
$ git config --global user.email
```

> should be your email, i.e. jon.rosado42@gmail.com

To fix either, just add the desired value in quotes after the command:

```
$ git config --global user.name “Jon Rosado”

$ git config --global user.email “jon.rodado42@gmail.com"
```

Using Git / Github

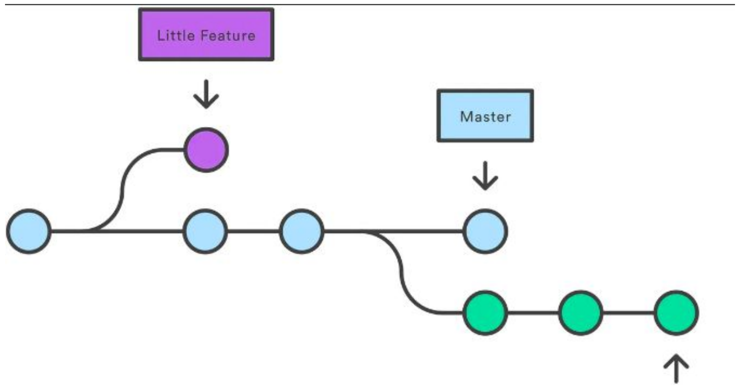
- In order to prevent having to enter your password each time you push up to Github, you must configure git and Github to recognize Secured Shell (SSH) keys that you generate.
- To check and see if you have any recognized SSH keys active on Github, go to https://github.com/settings/keys
- If you do not see any SSH keys listed, you do not have SSH configured with Github. We can assist with this later.
- If using Mac OS X, you can also configure your keychain to automatically enter your password via HTTPS.

Getting Started with Github(Connect the project)

- From your project directory, run `git init` to initialize a git repository.
- Go to Github, and create a new repository with the name of your project.
- Follow the instructions on Github to connect your initialized git repository to the remote server on Github.
- *Please Note: you must have files in your project directory to commit in order to push anything to your remote server.

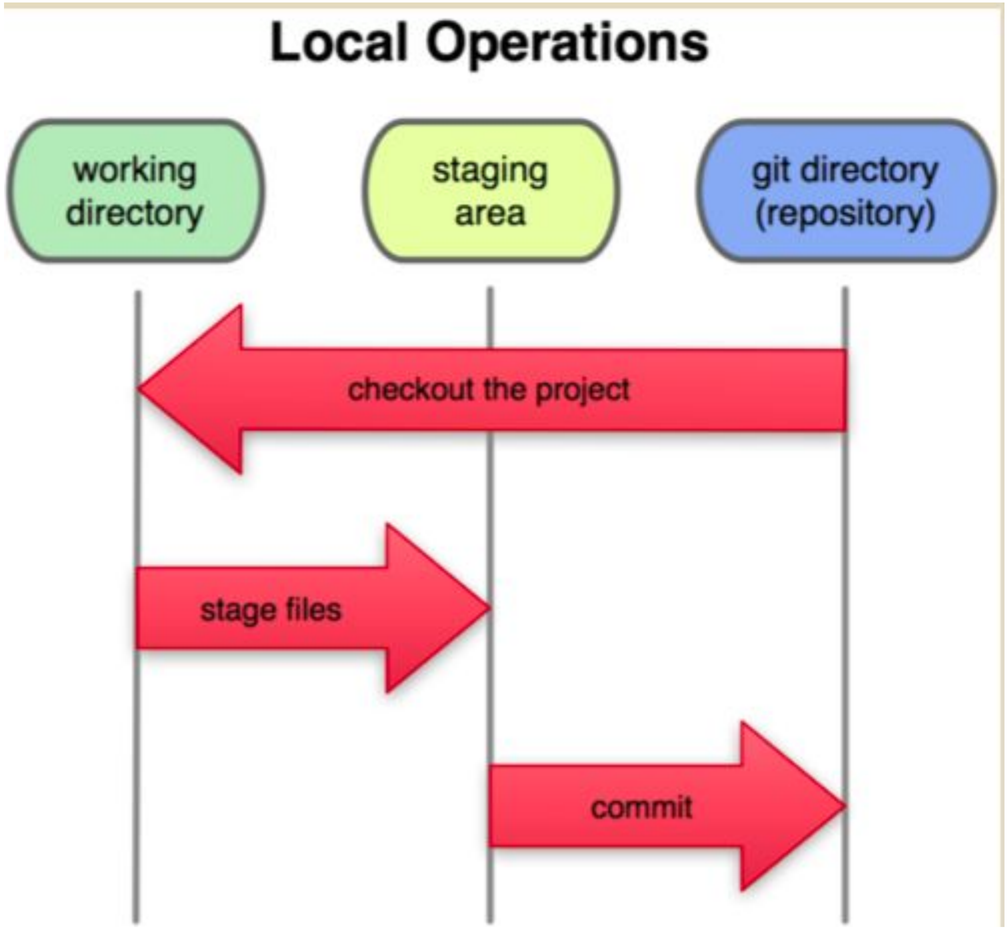
Before committing to git….

- Before committing to git you should know about branches.
- In Git, a branch is a new/separate version of the main repository.
- Branches allow you to work on different parts of a project without impacting the main branch.
- When the work is complete, a branch can be merged with the main project.
- You can even switch between branches and work on different projects without them interfering with each other.



Git branch

- git branch to view current branches in repo
- git checkout -b <branch name> to create a new branch with branch name
- git checkout <branch name> without ‘-b’ flag to switch to existing branches


Local Operations
working directory
staging area
git directory (repository)
checkout the project
stage files
commit

Basic git/github workflow

- From the command line, use git to create a new branch off of master to make your edits to.
- Stage edits to be committed to your git repository
- Commit changes using `git commit -m <message>` and be sure to leave a short but descriptive message detailing what the commit will change when merged to the master branch.
- Push changes to save them by using `git push origin <branch name>`

• On Github, create a new pull request for the branch that you have just pushed, and add any clarifying comments that you deem necessary.

• In order to close the issue associated with the pull request, tell GitHub to do so by adding ‘closes <issue number>’ in the comments.

(you can also use ‘closed, fix, fixes, resolve, resolves, and resolved’ before the issue number as well).

• Github will automatically check for merge conflicts. If there are any, check to see what they are and resolve them.

• Once everything is up to date, Github will allow you to merge using three separate options: Merge; Squash and Merge; Rebase and Merge.

Merge Conflict

- Merge conflicts arise when two members of the same development team work on the same file and try to merge in their respective changes.
- The proper way to avoid merge conflicts would be to ensure that only one branch is fully committed, pushed, and merged to master, allowing the other branch to integrate any changes before attempting to push and merge to master.
- If merge conflicts arise, don’t fret! Many text editors (as well as Github) provide tools to help track down conflicts and resolve them. Often times, it will show incoming changes juxtaposed with the current state of your file, and allow you to choose which to keep (one or both).

.gitignore

When sharing your code with others, there are often files or parts of your project, you do not want to share.

Examples

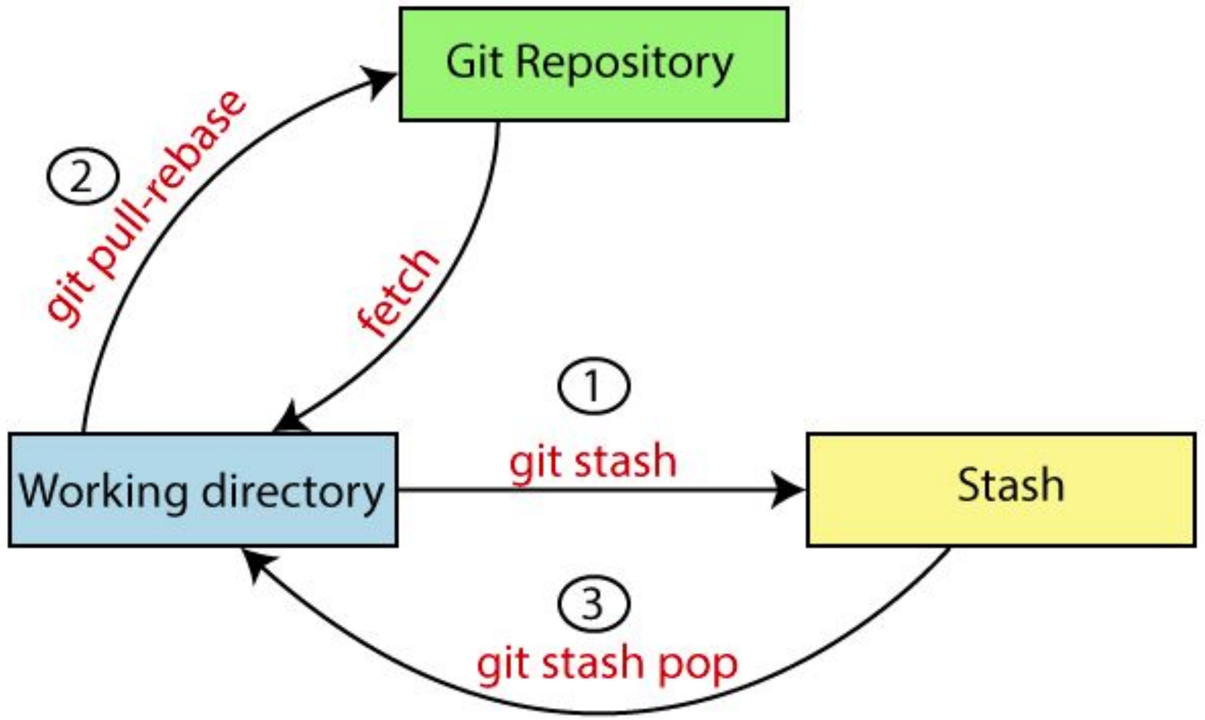
- log files
- temporary files
- hidden files
- personal files
- etc.

Git can specify which files or parts of your project should be ignored by Git using a .gitignore file.

Git will not track files and folders specified in .gitignore. However, the .gitignore file itself IS tracked by Git.

Git stash

- Sometimes you want to switch the branches, but you are working on an incomplete part of your current project. You don't want to make a commit of half-done work. Git stashing allows you to do so. The **git stash command** enables you to switch branches without committing the current branch.
- Generally, the stash's meaning is "**store something safely in a hidden place**." The sense in Git is also the same for stash; Git temporarily saves your data safely without committing.
- Stashing takes the messy state of your working directory, and temporarily save it for further use.


Git Repository
② git pull-rebase
fetch
① git stash
Working directory
Stash
③ git stash pop

Git Cheat Sheet

REBELLABS by ZEROTURNAROUND

Create a Repository

From scratch -- Create a new local repository
```
$ git init [project name]
```

Download from an existing repository
```
$ git clone my_url
```

Observe your Repository

List new or modified files not yet committed
```
$ git status
```

Show the changes to files not yet staged
```
$ git diff
```

Show the changes to staged files
```
$ git diff --cached
```

Show all staged and unstaged file changes
```
$ git diff HEAD
```

Show the changes between two commit ids
```
$ git diff commit1 commit2
```

List the change dates and authors for a file
```
$ git blame [file]
```

Show the file changes for a commit id and/or file
```
$ git show [commit]:[file]
```

Show full change history
```
$ git log
```

Show change history for file/directory including diffs
```
$ git log -p [file/directory]
```

Working with Branches

List all local branches
```
$ git branch
```

List all branches, local and remote
```
$ git branch -av
```

Switch to a branch, my_branch, and update working directory
```
$ git checkout my_branch
```

Create a new branch called new_branch
```
$ git branch new_branch
```

Delete the branch called my_branch
```
$ git branch -d my_branch
```

Merge branch_a into branch_b
```
$ git checkout branch_b
$ git merge branch_a
```

Tag the current commit
```
$ git tag my_tag
```

Make a change

Stages the file, ready for commit
```
$ git add [file]
```

Stage all changed files, ready for commit
```
$ git add .
```

Commit all staged files to versioned history
```
$ git commit -m "commit message"
```

Commit all your tracked files to versioned history
```
$git commit -am "commit message"
```

Unstages file, keeping the file changes
```
$ git reset [file]
```

Revert everything to the last commit
```
$ git reset --hard
```

Synchronize

Get the latest changes from origin (no merge)
```
$ git fetch
```

Fetch the latest changes from origin and merge
```
$ git pull
```

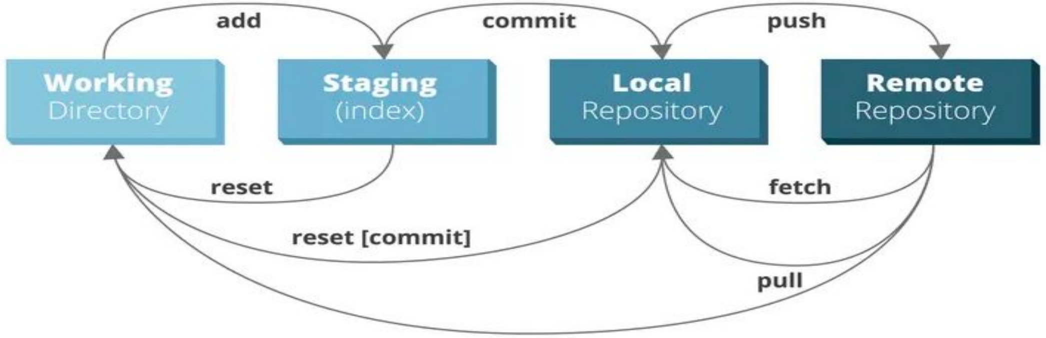
Fetch the latest changes from origin and rebase
```
$ git pull --rebase
```

Push local changes to the origin
```
$ git push
```

Finally!

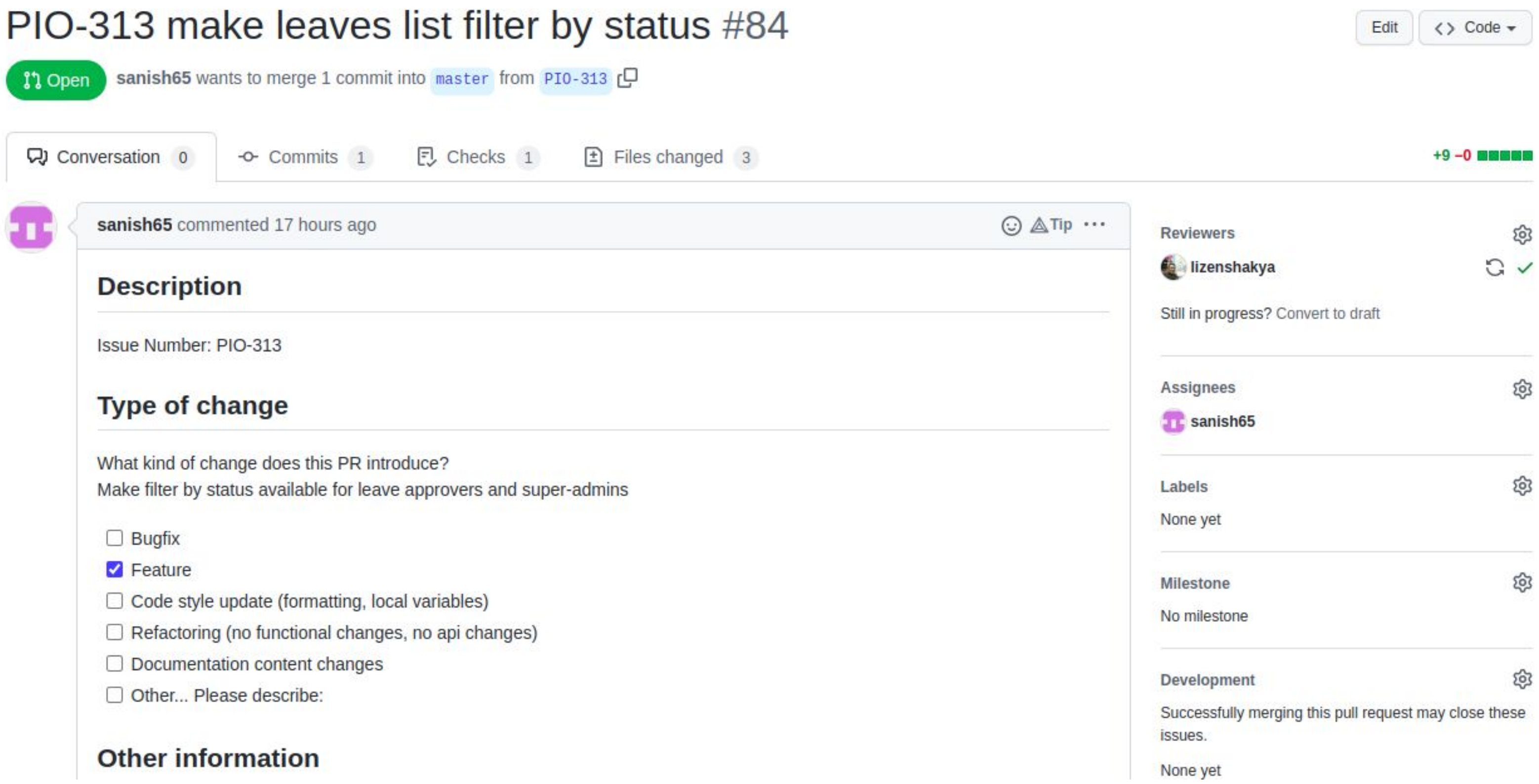
When in doubt, use git help
```
$ git command --help
```

Or visit https://training.github.com/ for official GitHub training.



Good Developer Practices.

- To tie the branch to your issue on Github, make sure to include the issue number after the branch name, e.g. branchName #1
- Title of the merge request should self define the branch.
- Description should contain jist all the changes made.



PIO-313 make leaves list filter by status #84

Edit | Code

Open sanish65 wants to merge 1 commit into `master` from `PIO-313`

Conversation 0 | Commits 1 | Checks 1 | Files changed 3 | +9 −0

sanish65 commented 17 hours ago

Description

Issue Number: PIO-313

Type of change

What kind of change does this PR introduce?
Make filter by status available for leave approvers and super-admins

- [ ] Bugfix
- [x] Feature
- [ ] Code style update (formatting, local variables)
- [ ] Refactoring (no functional changes, no api changes)
- [ ] Documentation content changes
- [ ] Other... Please describe:

Other information

Reviewers
lizenshakya ✓

Still in progress? Convert to draft

Assignees
sanish65

Labels
None yet

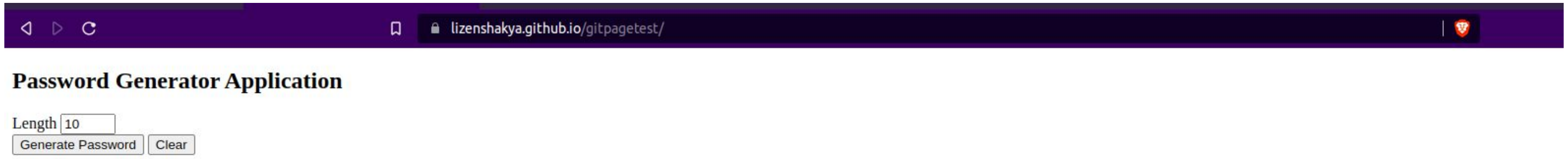
Milestone
No milestone

Development
Successfully merging this pull request may close these issues.

None yet

Github Pages

Host your static page on github



lizenshakya.github.io/gitpagetest/

Password Generator Application

Length 10

Any Feedbacks or queries?

Thank You!