Introduction to Git and Github

Sai Rithwik M

CS-303(P) Software Engineering Lab

International Institute of Information Technology, Bangalore

sai.rithwik@iiitb.ac.in

August 6, 2021

Overview

What is Git?

Case 1

Imagine you are working on your programming assignment. You have implemented a feature 1. Now you want to implement a feature 2.

What is Git?

Case 1

Imagine you are working on your programming assignment. You have implemented a feature 1. Now you want to implement a feature 2.

You find that your code crashes! You want to go back to feature 1, and undoing doesnt help much.

What is Git?

Case 2

Imagine you are working on your course project. You want to test out new features and as a precaution from Case-1, you save the code in different folders.

Case 2

Imagine you are working on your course project. You want to test out new features and as a precaution from Case-1, you save the code in different folders.

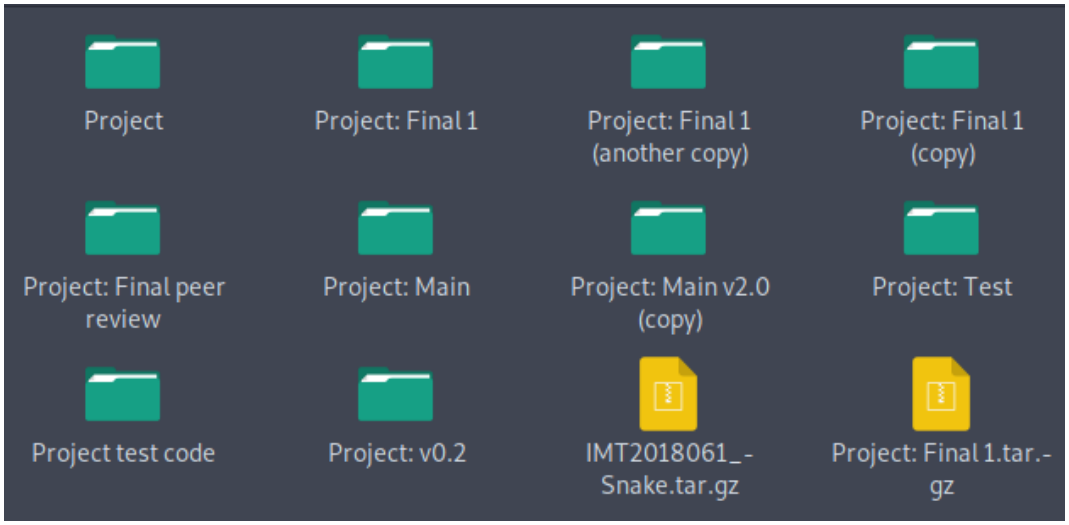


Figure: The pain

Case 2

Imagine you are working on your course project. You want to test out new features and as a precaution from Case-1, you save the code in different folders.

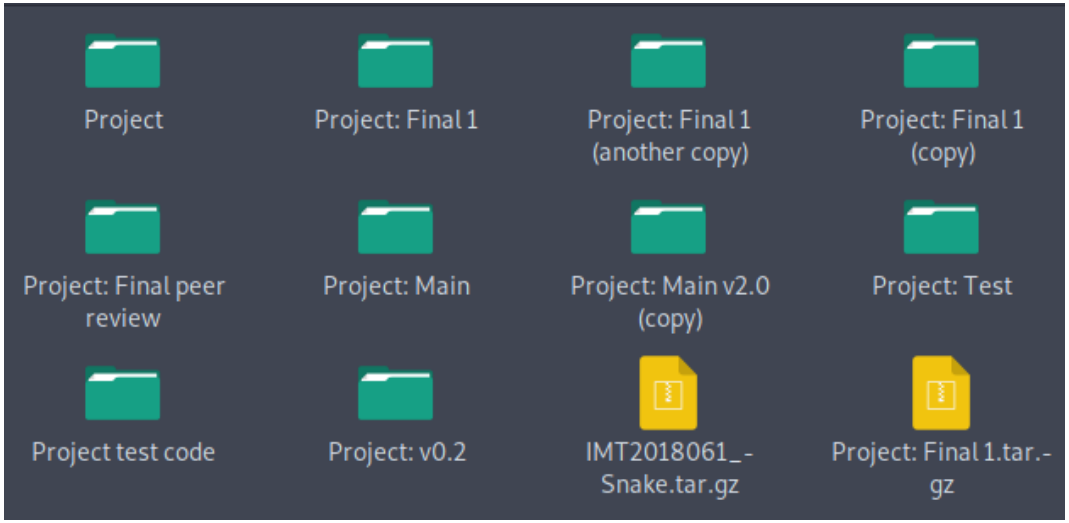


Figure: The pain

Not so efficient right?

What is Git?

We begin to understand that we need some sort of system to help you create checkpoints and versions to maintain track of your documents.

What is Git?

We begin to understand that we need some sort of system to help you create checkpoints and versions to maintain track of your documents.

This is exactly the problem that **Git** is trying to help us with. Git is a distributed Version Control System (VCS).

What is Git?

We begin to understand that we need some sort of system to help you create checkpoints and versions to maintain track of your documents.

This is exactly the problem that **Git** is trying to help us with. Git is a distributed Version Control System (VCS).

VCS is the practice of tracking and managing your code.

What is Git?

We begin to understand that we need some sort of system to help you create checkpoints and versions to maintain track of your documents.

This is exactly the problem that **Git** is trying to help us with. Git is a distributed Version Control System (VCS).

VCS is the practice of tracking and managing your code.

Figure: Popular VC tools

Git Essentials

There are hundreds of Git commands out there, but good news! You just need a few regularly. Here are some popular commands.

- git init
- git add
- git commit
- git push
- git pull
- git clone
- git branch

Workflow commands

- git status
- git diff
- git checkout
- git remote

Helper commands

`$ git init`

Used to initialise your repo with git. Essentially tells git to start tracking your files.

Once you use this command you should be able to see a `.git` folder in your repo.

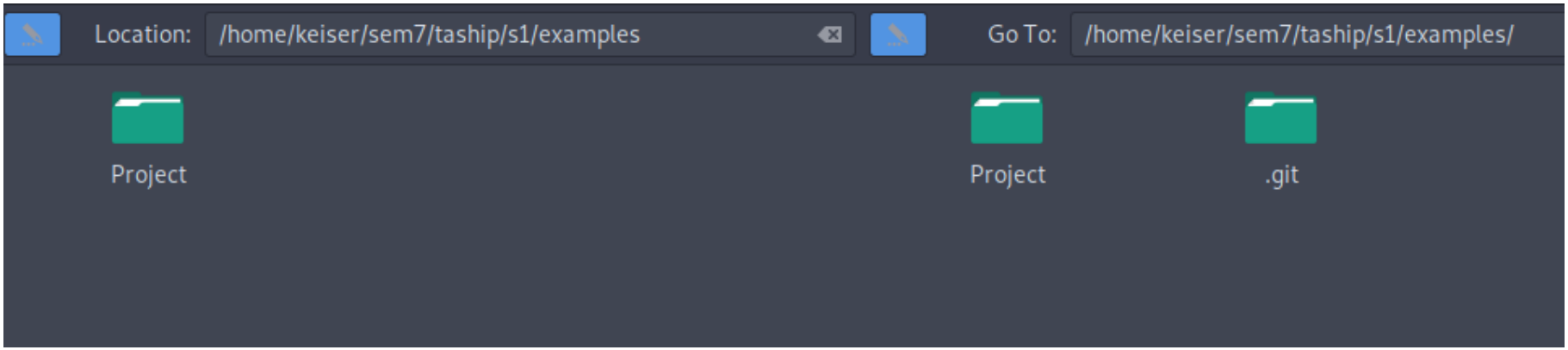


Figure: Before and after `git init`

$ git add

Once you have made changes to your code, you use `git add` to add your file to the staging area.

Staging area[1]essentially stores the changes in the files which are going to be used while committing.

`git add -A` to add all files
`git add "filename"` to add a specific file

[1]Checkout a similar command: `git stash` https://www.atlassian.com/git/tutorials/saving-changes/git-stash

$ git commit

Once you feel that you have made sufficient changes to accommodate for a feature, you use `git commit`.

Data from the staging area is then passed on to lock in changes.

A commit can be uniquely identified with its hash. This can be used to perform various operations like moving `HEAD`[2] to a previous commit etc.

```
git commit -m "Sensible commit message"
```

[2] `HEAD` points to the latest commit in your workflow

$ git clone

You could think this command like downloading a repo from local. All clone does is gets a copy of remote repository to local. This is usually done in the beginning.

$ git pull

This command is used to pull[3] your `git` initialised repo from a remote location(ex. repo on Github) to your local space(ie. your PC). Usually done whenever you want to update local repo with changes made in remote.

$ git push

This command is used to push the commits you have made on your local system to github.

[3] Checkout a similar command: `git fetch`

Git Essentials

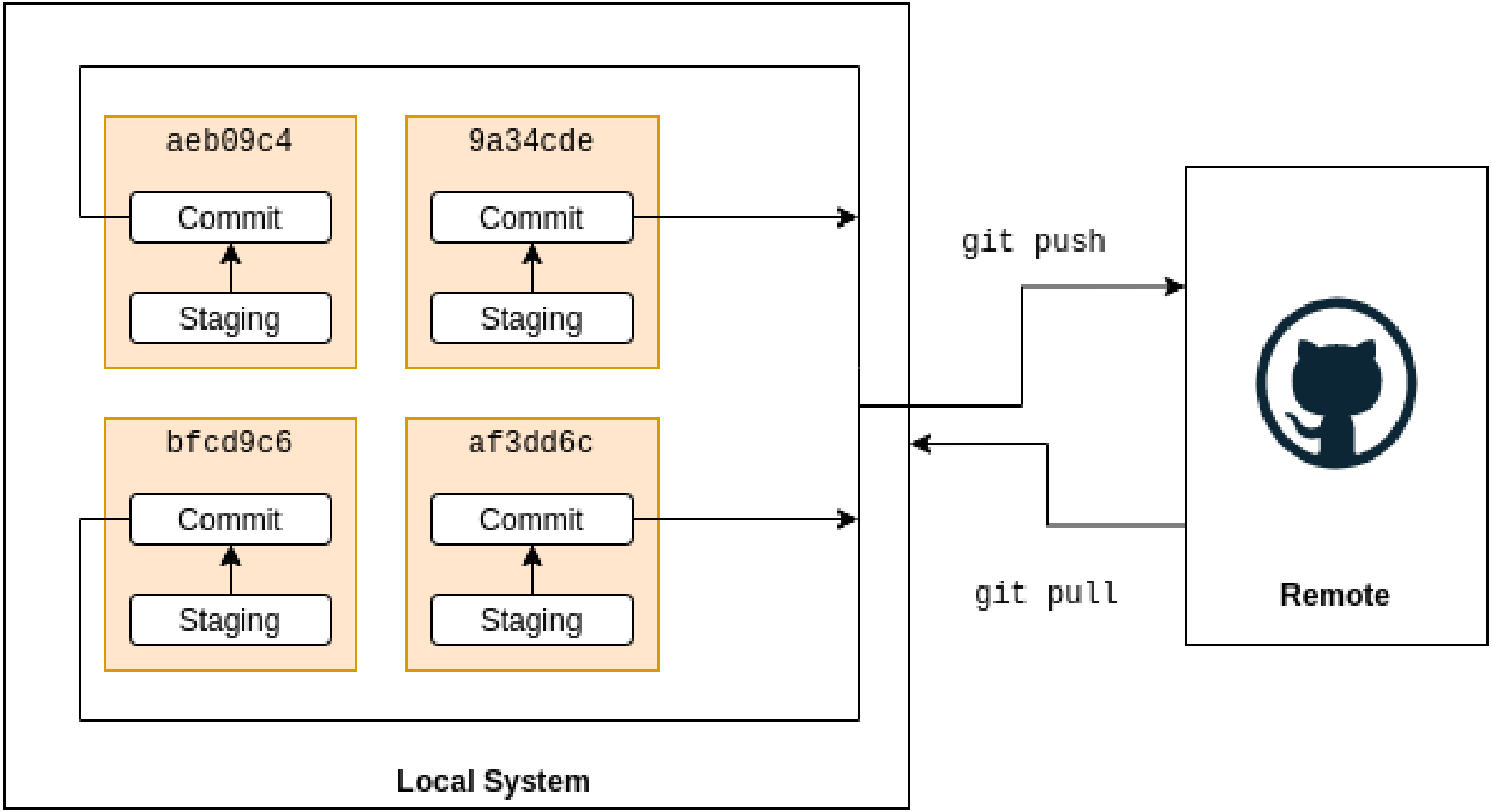
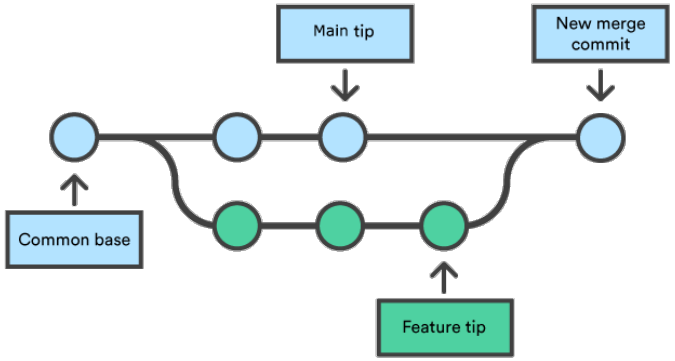


Figure: Commands we have learnt

Git Essentials: Branches[4]

Branches are one of the most widely used features in git. You as a developer would want to maintain a stable piece of code and perform any tests on a different copy of the code. Once you feel confident about the code then you could merge your code to your base branch.

`git checkout -b "branch_name"` to create a new branch.
`git branch -v` to view local branches.



???

[4] Checkout `git tags` and `git merge`

Git Essentials

There are a few more commands you could look at:

- git status: Displays the state of staging area and your repo
- git diff: Shows the difference between any two commits
- git remote: Helpful to check your connection with remote repositories

Github

Github is a remote collaboration tool which helps you and your teams collaborate on a project and stay in sync with each other. It acts as a remote git repository for your teams to collaborate.

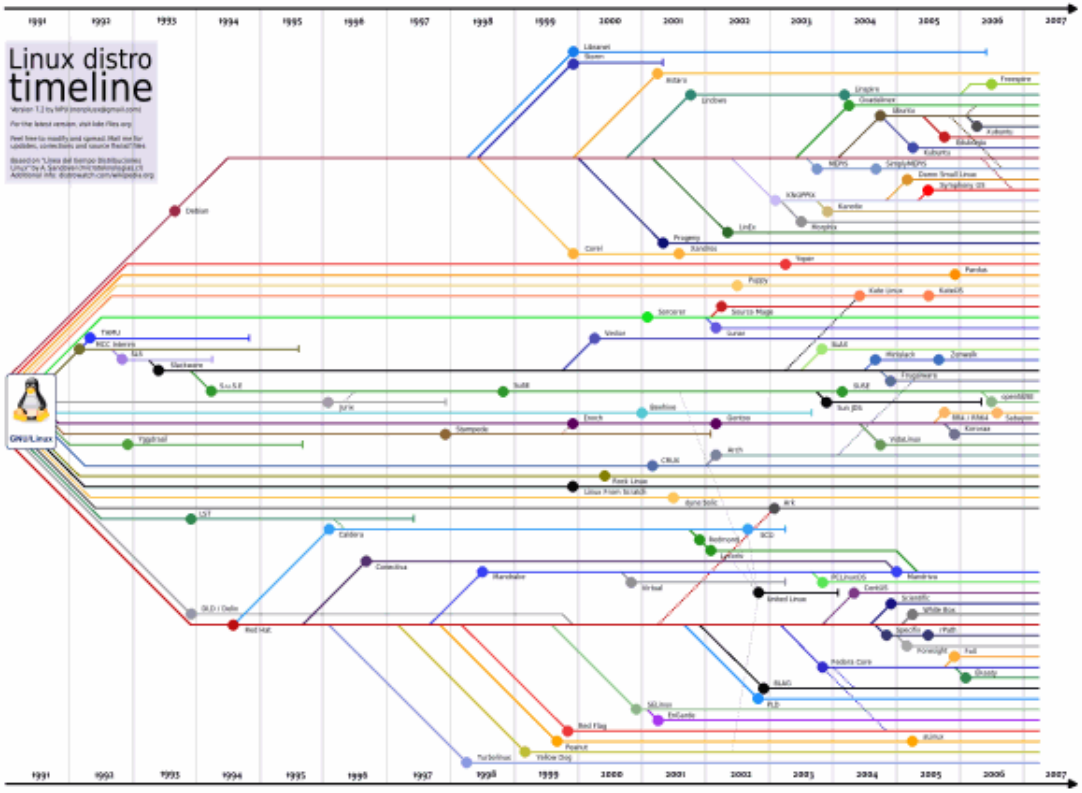
Key features of Github:

- Team collaboration and access management
- Forks
- Issues
- Pull Requests
- Code Reviews
- CI/CD pipelines

Checkout Github student developer pack, if you haven't already.

Github Essentials: Forks[5]

Imagine someone has a piece of code. You want to modify it to suit your own needs or build a product out of it. You could quickly create a fork of that project.



[5] Read about origin, upstream

Git & Github Extras

- Github Actions: Connects all our tools to make sure the development workflow is automated.
- README.md: A markdown file which essentially describes the motive and things to look at while working with the codebase.
- .gitignore: A special file in git which doesn't track the files mentioned in this file to the staging area. Used to manage secrets eg: API keys etc.

Interesting stuff to read about

Here's some additional reading material, questions to understand git workflow better.

- Git and Github tutorials
- Why do you need to add or stash when you can commit changes?
- What are merge conflicts and how to avoid them?
- How to revert back to older commits?
- RESET (Dangerous command)
- What's rebase?
- .git file structure [Unnecessary, but interesting]
- How to write good commits?

Graded Task 1 sneak peek

Develop your portfolio page using existing templates or create your own website and push it on Github. Also deploy using Github Pages or any other service like Heroku, Netlify or Vercel.

Details, score distribution and other instructions will be shared soon.

Ungraded Task: 1

Step 1: Fork the Repository

- Click on the fork button which is present on the top right of the repository dashboard.
- You will now have your personal fork.

Step 2:Commit changes

- Find the file name corresponding to your roll number in the repository.
- Change the name ‘torvalds‘ to your Github username. Make sure you do not put @ or any symbols that occur before it.
- Commit the file.

Step 3: Send a Pull Request

- Come back to the forked repo’s dashboard.
- You will find a ‘Contribute‘ button. Click on Open Pull Request.
- Click on Pull Request and add comments if you like to.

The End