

# OSP THEORY DA 1 ( GitHub)

P ASHWATH

19BIT0222

FACULTY :

JAYAKUMAR S

# GITHUB

GitHub repository link:

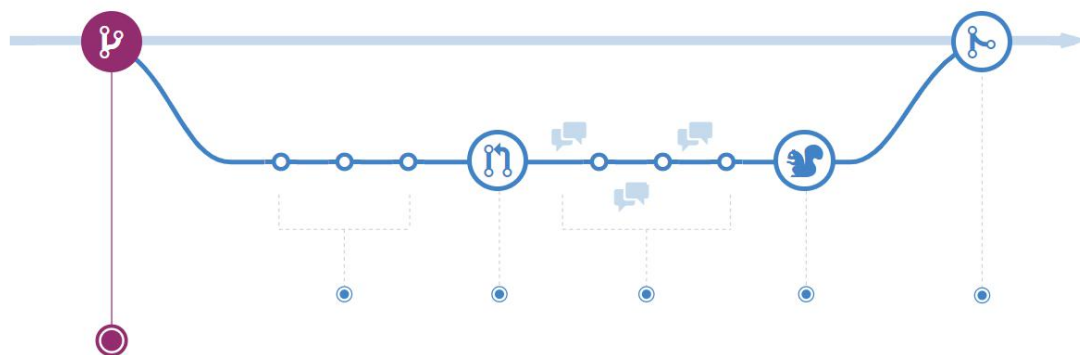
<https://github.com/ashwath-p/Portfolio>

GitHub is a platform that helps people solve problems by building software together. Here, a user can work on his project with his teammates. Even though they all work on the same project, they may not be in the same part of the world. Many people come up with a good idea but would not be able to implement it because of the lack of knowledge about building software. So now, they can open an issue in the git hub, and anyone could see or review it. One person may see it and could tag another person and tell them to take on this issue and solve it. The person who opened this issue is not the only one who is using the software. There are thousands of other people who use it too. So the person working needs a dedicated place to experiment and resolve this issue, so none of it would end up the software before they are ready. So he creates a branch of this code in an alternate timeline where he can safely make changes to the software. Now he is good to make changes to his code, and once he is complete, he could commit his work. Git hub tracks his progress and has snapshots of his progress. When he is ready to collaborate with his team, he opens a PULL REQUEST. Which lets you show others the changes you are proposing so they can review and discuss them. Which lets his teammates remove roadblocks and even make improvements. In Github, everyone has a copy of the project. So if another project member has an idea that would make the project better, he can add it to the branch himself. GitHub keeps a record of the contribution of everyone.

Once the team has signed up on the changes, the main contributor incorporates the new code into the project by the merge request method. The new feature will be available to everyone as soon as he merges it in. Now people from everywhere can benefit from this new feature.

## Quick steps to access Github:

### Step 5: Create a new branch.



## Working methodology of GitHub:

Merge.

Some commonly used words by the users of GitHub.

**Command Line:**

The computer program we use to input Git commands. On a Mac, it's called Terminal. On a PC, it's a non-native program that you download when you download Git for the first time (we'll do that in the next section). In both cases, you type text-based commands, known as prompts, into the screen, instead of using a mouse.

**Repository:**

A directory or storage space where your projects can live. Sometimes GitHub users shorten this to "repo." It can be local to a folder on your computer, or it can be a storage space on GitHub or another online host. You can keep code files, text files, image files, you name it, inside a repository.

**Version Control:**

Basically, Git has been designed to serve. When you have a Microsoft Word file, you either overwrite every saved file by saving it again, or you save multiple versions. With Git, you don't have to. It keeps "snapshots" of every point in time in the project's history, so you can never lose or overwrite it.

**Commit:**

This is the command that gives Git its power. When you commit, you are taking a "snapshot" of your repository then, giving you a checkpoint to which you can reevaluate or restore your project to any previous state.

**Branch:**

How do multiple people work on a project at the same time without Git getting them confused? Usually, they "branch off" of the main project with their versions full of changes they have made.

# Advantages and disadvantages of using GitHub:

## **Advantages of using GITHUB:**

### **1. It makes it easy to contribute to your open source projects**

Almost every open-source project uses GitHub. Using GitHub is free if your project is open source and includes a wiki and issue tracker that makes it easy to include more in-depth documentation and get feedback about your project. If you want to contribute, you just fork a project, make your changes and then send them a pull request using GitHub web interface.

### **2. Documentation**

By using GitHub, you make it easier to get excellent documentation. Their help section and guides have articles for nearly any topic related to git that you can think of.

### **3. Showcase your work**

Are you a developer and wishes to attract recruiters? GitHub is the best tool you can rely on for this. Today, when searching for new recruits for their project, most companies look into the GitHub profiles. If your profile is available, you will have a higher chance of being recruited even if you are not from a great university or college.

### **4. Markdown**

Markdown allows you to use a simple text editor to write formatted documents. GitHub has revolutionized writing by channeling everything

through Markdown: from the issue tracker, user comments, everything. With so many other programming languages to learn for setting up projects, it's really a big benefit to have your content inputted in a format without having to learn yet another system.

## **5. GitHub is a repository**

This was already mentioned before, but it's important to note, GitHub is a repository.

What this means that it allows your work to get out there in front of the public. Moreover, GitHub is one of the largest coding communities around right now, so it's wide exposure for your project.

## **6. Track changes in your code across versions**

When multiple people collaborate on a project, it's hard to keep track revisions—who changed what, when, and where those files are stored. GitHub takes care of this problem by keeping track of all the changes that have been pushed to the repository. Much like using Microsoft Word or Google Drive, you can have a version history of your code so that previous versions are not lost with every iteration.

## **7. Integration options**

GitHub can integrate with common platforms such as Amazon and Google Cloud, services such as Code Climate to track your feedback, and can highlight syntax in over 200 different programming languages.

## **Potential Drawbacks:**

### **1.Security**

GitHub does offer private repositories, but this isn't necessarily perfect for many. For high value intellectual property, you're putting all of this in

the hands of GitHub as well as anyone who has a login, which like many sites has had security breaches before and is targeted constantly. It is often better than nothing, but it's not perfect. In addition, some clients/employers will only allow code on their own secure internal Git as a matter of policy.

## 2.Pricing

Some of GitHub features, as well as features on other online repositories, are locked behind a SaaS paywall. If you have a large team, this can add up fast. Those who already have a dedicated IT team and their own internal servers are often better off using their own internal git for cost reasons, but for most the cost isn't outrageous.

## Setting Up GitHub And Git For The First Time



GitHub' s signup page.

First, you'll need to [sign up for an account](#) on GitHub.com. It's as simple as signing up for any other social network. Keep the email you picked handy; we'll be referencing it again soon.

You could stop there and GitHub would work fine. But if you want to work on your project on your local computer, you need to have Git installed. In fact, GitHub won't work on your local computer if you don't install Git. Install Git for Windows, Mac or Linux [as needed](#).



<http://git-scm.com/>, where you download Git.

Now it's time to go over to the command line. On Windows, that means starting the Git Bash app you just installed, and on OS X, it's regular old Terminal. It's time to introduce yourself to Git. Type in the following code:

```
git config --global user.name "Your Name Here"
```

Of course, you'll need to replace "Your Name Here" with your own name in quotations. It can be your legal name, your online handle, anything. Git doesn't care, it just needs to know to whom to credit commits and future projects.

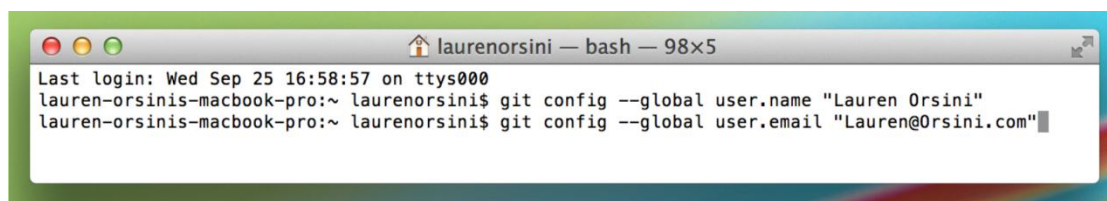


Next, tell it your email and make sure it's the same email you used when you signed up for a GitHub.com account just a moment ago. Do it like this:

```
git config --global user.email "your_email@youremail.com"
```

That's all you need to do to get started using Git on your computer.

However, since you did set up a GitHub.com account, it's likely you don't just want to manage your project locally, but also online. If you want you can also set up Git so it doesn't ask you to log in to your GitHub.com account every time you want to talk to it. For the purposes of this tutorial, it isn't a big deal since we'll only be talking to it once. The full tutorial to do this, however, is [located on GitHub](#).

A screenshot of a macOS terminal window. The title bar shows a home icon, the name 'laurenorsini', and '— bash — 98x5'. The terminal text shows the last login time and two successful 'git config' commands: 'git config --global user.name "Lauren Orsini"' and 'git config --global user.email "Lauren@Orsini.com"'.

```
laurenorsini — bash — 98x5
Last login: Wed Sep 25 16:58:57 on ttys000
lauren-orsinis-macbook-pro:~ laurenorsini$ git config --global user.name "Lauren Orsini"
lauren-orsinis-macbook-pro:~ laurenorsini$ git config --global user.email "Lauren@Orsini.com"
```

## Baby's first Git commands. Creating Your Online Repository

Now that you're all set up, it's time to create a place for your project to live. Both Git and GitHub refer to this as a repository, or "repo" for short, a digital directory or storage space where you can access your project, its files, and all the versions of its files that Git saves.

Go back to GitHub.com and click the tiny book icon next to your username. Or, go to the [new repository page](#) if all the icons look the same. Give your repository a short, memorable name. Go ahead and make it public just for kicks; why hide your attempt to learn GitHub?

Creating a new repository on GitHub.

Don't worry about clicking the checkbox next to "Initialize this repository with a README." A Readme file is usually a text file that explains a bit about the project. But we can make our own Readme file locally for practice.

Click the green "Create Repository" button and you're set. You now have an online space for your project to live in.

### **Creating Your Local Repository**

So we just made a space for your project to live online, but that's not where you'll be working on it. The bulk of your work is going to be done on your computer. So we need to actually mirror that repository we just made as a local directory.

This—where we do some heavy command line typing—is the part of every Git tutorial that really trips me up, so I'm going to go tediously, intelligence-insultingly slow.

First type:

```
mkdir ~/MyProject
```

**mkdir** is short for make directory. It's not actually a Git command, but a general navigational command from the time before visual computer interfaces. The **~/** ensures that we're building the repository at the top level of your computer's file structure, instead of stuck inside some other directory that would be hard to find later. Actually, if you type **~/** into your browser window, it'll bring up your local computer's top level directory. For me, using Chrome on a Mac, it displays my Users folder.

Also, notice that I called it MyProject, the very same name I called my GitHub repository that we made earlier. Keep your name consistent, too.

Next, type:

```
cd ~/MyProject
```

**cd** stands for change directory, and it's also a navigational command.

We just made a directory, and now we want to switch over to that directory and go inside it. Once we type this command, we are transported inside MyProject.

Now we're finally using a Git command. For your next line, type:

```
git init
```

You know you're using a Git command because it always begins with **git**. **init** stands for "initialize." Remember how the previous two commands we typed were general command-line terms? When we type this code in, it tells the computer to recognize this directory as a local Git repository. If you open up the folder, it won't look any different, because this new Git directory is a hidden file inside the dedicated repository.

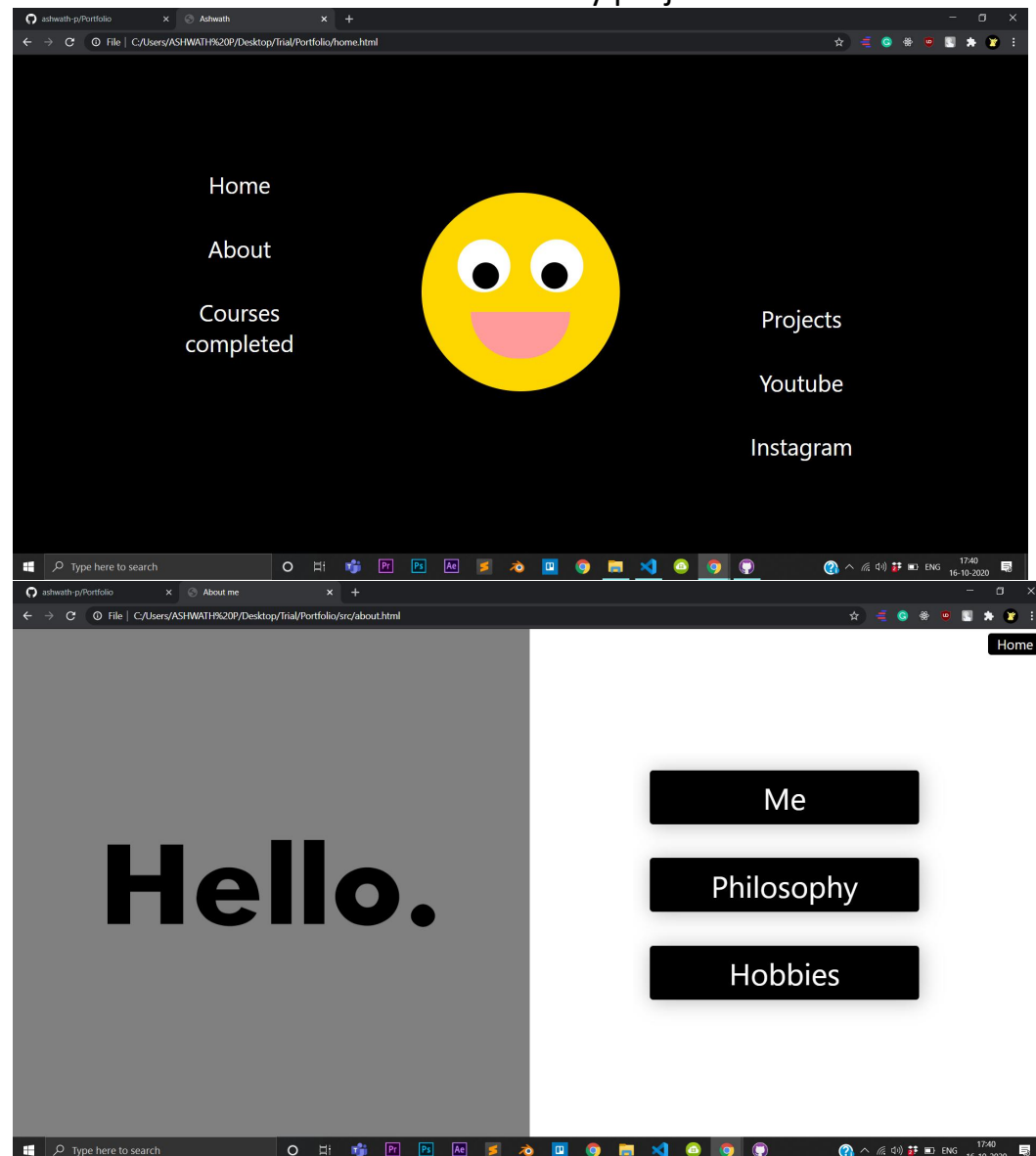
Creating a local Git repository in three steps.


However, your computer now realizes this directory is Git-ready, and you can start inputting Git commands. Now you've got both an online and a local repo for your project to live inside.

# Personal Portfolio

I have created my own personal portfolio step by step without using boot-strap or semantic UI. This is completely filled with HTML, CSS and JS.

Here are some of the screenshots of my project.





# Ashwath

Hello, I'm Ashwath P.  
Currently pursuing my bachelor degree in VIT Vellore.  
2nd year B.Tech Information Technology.  
I'm all about front-end development.  
Well versed in HTML, CSS, JS and node JS.  
Trying to spread happiness and positivity.

ashwathp.2019@vitstudent.ac.in  
+91 6383660273

ashwath-p/Portfolio

Me

+

File | C:/Users/ASHWATH%20P/Desktop/Trial/Portfolio/src/me.html

Back

Udemy

Categories

Search for anything

Udemy for Business

Teach on Udemy

My courses

Shopping cart

Notifications

Profile

Back

Development > Web Development

## The Web Developer Bootcamp 2020


COMPLETELY REDONE - The only course you need to learn web development - HTML, CSS, JS, Node, and More!

4.7 ★★★★★ (194,172 ratings) 600,346 students

Created by Colt Steele

Last updated 10/2020 English English [Auto], French [Auto], 5 more

Wishlist Share Gift this course



Preview this course

You purchased this course on Mar. 21, 2020

Go to course

30-Day Money-Back Guarantee

**This course includes:**

- 61.5 hours on-demand video
- 46 articles
- 88 downloadable resources
- 62 coding exercises
- Full lifetime access
- Access on mobile and TV

**Training 5 or more people?**  
Get your team access to 5,000+ top Udemy courses anytime, anywhere.

### What you'll learn

✓ Make REAL web applications using cutting-edge technologies	✓ Continue to learn and grow as a developer, long after the course ends
✓ Create a blog application from scratch using Express, MongoDB, and Semantic UI	✓ Create a complicated yelp-like application from scratch
✓ Write your own browser-based game	✓ Create static HTML and CSS portfolio sites and landing pages
✓ Think like a developer. Become an expert at Googling code questions!	✓ Create complex HTML forms with validations and responsive layouts
✓ Write web apps with full authentication	✓ Use Bootstrap to create good-looking responsive layouts
✓ Implement responsive navbars on websites	✓ Use JavaScript variables, conditionals, loops, functions, arrays, and objects
✓ Write JavaScript functions, and understand scope and higher order functions	✓ Create full-stack web applications from scratch
✓ Manipulate the DOM with vanilla JS	✓ Manipulate the DOM using jQuery

WebDev Bootcamp

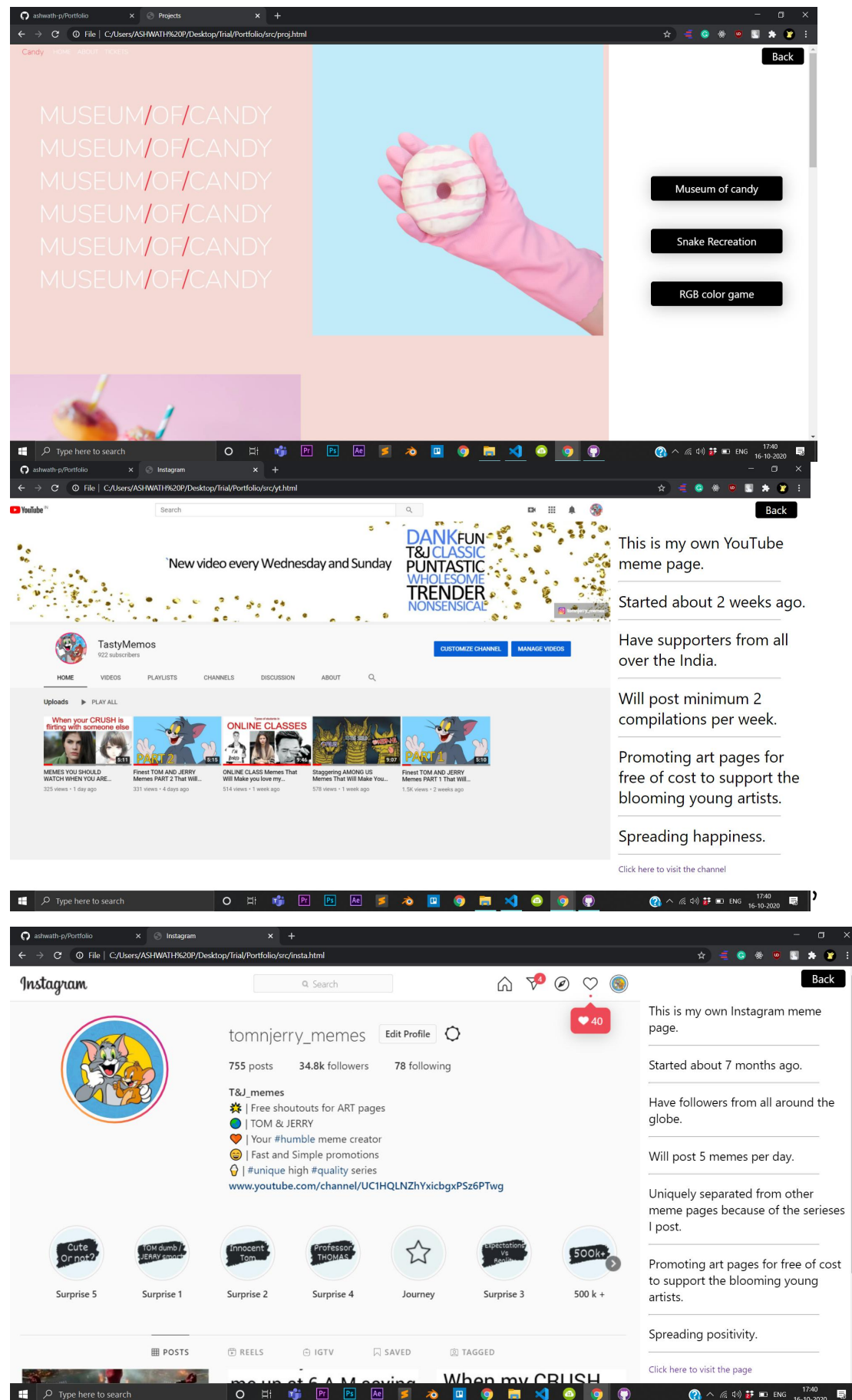
MySQL Bootcamp

React JS Bootcamp

WebDev Bootcamp

MySQL Bootcamp

React JS Bootcamp



## Project version history:

The image displays two screenshots of the GitHub repository 'ashwath-p/Portfolio'. The top screenshot shows the repository page with the commit history table. The bottom screenshot shows the same page but with the 'Environments' section expanded, showing a deployment failure for 'portfolio39ash'.

**Repository: ashwath-p/Portfolio**

**Commit History:**

Commit	Message	Time
4e4df99	Updating styles	1 minute ago
img	images	37 minutes ago
src	Updating styles	1 minute ago
stylesheet	Social media achievements	30 minutes ago
README.md	created readme	28 minutes ago
home.html	Updating styles	1 minute ago

**Files:**

- img
- src
- stylesheet
- README.md
- home.html

**README.md:**

### Portfolio

Welcome to my Personal Portfolio created for my Open Source Programming(OSP) course.

**Environments:**

- portfolio39ash: Failure

## GitHub repository link:

<https://github.com/ashwath-p/Portfolio>



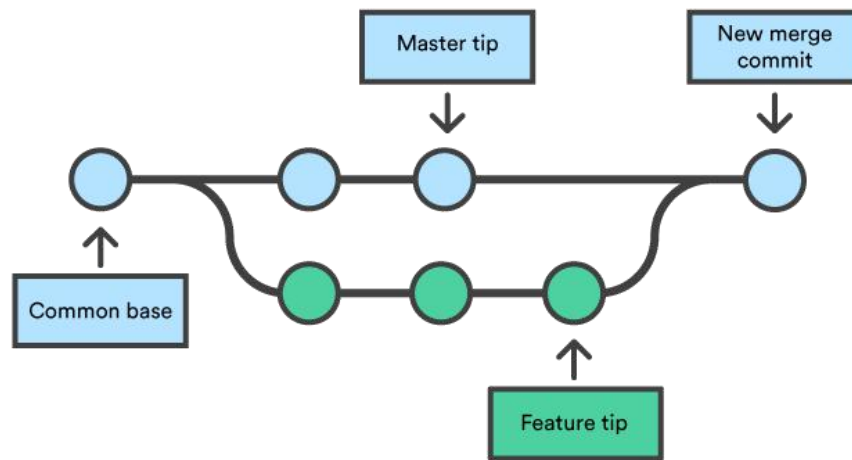
## Features needed to be added in GitHub:

1. The ability to suggest changes to multiple lines at once.
2. Even better code review in GitHub for mobile.
3. Fine-tune access to external actions.
4. Adopt more suggestions from the users.
5. Add website hosting for more than one for a user.
6. Improve the UI/UX a little bit more fluid.
7. Support a compiler.
8. Give achievement badges for the users who have achieved something incredible.

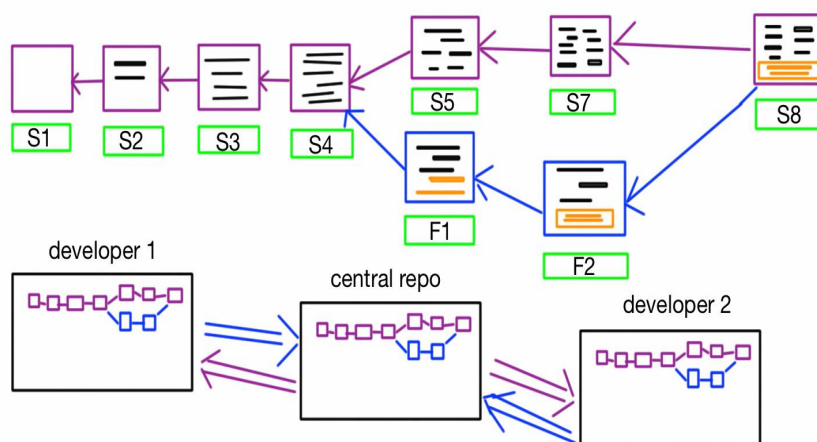
## What is version control?

Version control allows you to keep track of your work and helps you to easily explore the changes you have made, be it data, coding scripts, notes, etc. You are probably already doing some type of version control, if you save multiple files, such as Dissertation\_script\_25thFeb.R, Dissertation\_script\_26thFeb.R, etc. This approach will leave you with tens or hundreds of similar files, making it rather cumbersome to directly compare different versions, and is not easy to share among collaborators. With version control software such as Git, version control is much smoother and easier to implement. Using an online platform like Github to store your files means that you have an online back up of your work, which is beneficial for both you and your collaborators.





## Version Control System



## Software Development:

The primary use of version control systems has generally been in software development, where the ability to revisit older instances of a file may be useful, if not absolutely essential. There are many circumstances in which having some sort of file versioning is imperative.

For example, when a bug appears in a piece of software, the first step to tracking down the bug is to know what changed in the source code since the last working version. If necessary, the changes made can be "rolled back", essentially removing the changes that introduced the error.

If a given piece of source code may be edited by multiple people, version control systems may assist here as well to reduce the number of conversations developers need to have with each other over code management--leaving more time to discuss design, documentation and testing. Some version control systems will lock files so that only one person may edit it at one time. When done, the user releases the lock and the next person may edit the file. Other systems allow multiple developers to work on a given file and when ready merge the changes in a meaningful way.

For software projects in which the development team is scattered over a large area--be it a city, a country, or the globe--managing these changes and meaningful information describing them requires a modern version control system.

## **Documentation:**

Many documents are "living documents". Since they describe things that change, they must themselves change, or become useless. The ability to revisit historical versions of a document may be used to satisfy curiosity or keep up with those changes, or document changes in the subject itself.

If a document describes something relatively complex, the process of writing and editing it may be an iterative one, as with software. As well, multiple people may have multiple roles in the document's development, and the principle of version control may be applied to the document in the same way as they are for larger pieces of software. An example is The Linux Documentation Project, which is a global effort to provide documentation for the Linux operating system.

# Systems Management:

In a large computing environment, typically there are many similar machines whose configurations are modifications to a common base. For example, there might be a several machines with the same operating system, similar startup and shutdown scripts, and common network configurations, underneath their applications and services. It may be useful to manage their configurations in a central, version-controlled repository, where variations are easily tracked and common elements are easily replicated. If a change is made to the configuration that affects all machines, it becomes a relatively simple affair to visit each machine and update its configuration with the new one.

GitHub repository link:

<https://github.com/ashwath-p/Portfolio>