

DHARMSINH DESAI UNIVERSITY

MCA SEM -2

DATA STRUCTURE USING C

**THEDOCUMENTATIONOFTHE WORKINGOFGITANDGITHUB**

**NAME :GODIWALA.LIZA.V**

**ROLL NO:**MA003

**STUDENT ID :** 18MAPBG004

**SUBMITTED TO,**

**-----------------------------**

PROF. Himanshu Purohit

**Report on Git hub**

**What is GitHub?**

* **GitHub** is a web-based hosting service for version control using Git. It is mostly used for computer code. It offers all the distributed version control and source code management (SCM) functionality of Git as well as adding its own features.
* It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.
* GitHub offers plans for enterprise, team, pro and free accounts which are commonly used to host open-source software projects. As of January 2019, GitHub offers unlimited private repositories to all plans, including free accounts.
* As of June 2018, GitHub reports having over 28 million users and 57 million repositories (including 28 million public repositories), making it the largest host of source code in the world.
* GitHub and services like it, such as Bitbucket, GitLab or Goitrous use git. Git is a distributed version control system written by the creator of Linux, Linus Torvalds, after he got frustrated with proprietary software. Git is similar to other version control systems such as subversion or CVS, but it's distributed. What this means is that if you clone a git project, you have the entire project history. You can commit, branch and tag all on you local machine without interacting with a server at all. If you were working with subversion or another centralized VCS all of your interactions occur with the server.

**What are the three best reasons to use GitHub?**

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

* Today, nearly every open-source project uses GitHub to manage their project. GitHub is free if your project is open source and includes a wiki and issue tracker that make it easy to include more in-depth documentation and get feedback about your project. Symphony, Ruby on Rails, and Django all use GitHub (symphony, [rails](https://github.com/rails), Django) to develop their project and accept contributions from the community. Even the GitLab source is available at GitHub! If you want to contribute, you just fork their project, make your changes and then send them a pull request using the GitHub web interface.
* Interestingly, Drupal doesn't utilize GitHub (they host their own central git repository). Their main repository is [mirrored](https://github.com/drupal/drupal) on GitHub, but their process for contributing is a little bit different, both for core and for modules which are also all git repositories. You can still use git, but you need to send patches directly to their issue tracker.

2. Markdown

* Markdown is a way to write formatted documents using nothing more than a text editor. In fact, I wrote this post in markdown! And GitHub has turned everything into markdown: issue tracker, the wiki, comments, everything. Since we're already writing our READMEs and other documentation for the project in markdown so that it can be tracked easier using version control, it only makes it easier that we don't have to learn another system for formatting on other parts of the website.
* GitHub uses what they call "GitHub flavoured markdown", which makes a few changes to standard markdown which makes it more useful in a programming environment. Note that the other services also support markdown or have their own, similar flavoured markdown.

3. Documentation

* GitHub has excellent documentation. Their [help section](https://help.github.com/) and [guides](https://guides.github.com/) have articles for nearly any topic related to git that you can think of. Need to know how to generate SSH keys? They have an [article](https://help.github.com/articles/generating-ssh-keys). Wondering about a good git workflow? They have a [guide](https://guides.github.com/introduction/flow/). If you're looking for a sample Gi ignore for your next project they have many [samples](https://github.com/github/gitignore) for nearly anything you can think of. Are they missing one? You can fork their Gi ignore repository, add it and contribute it back!

What is Git?

* **Git** is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.
* Git was created by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development. Its current maintainer since 2005 is Junio Hamano.
* As with most other distributed version-control systems, and unlike most client–server systems, every Git directory on every computer is a full-fledged repository with complete history and full version-tracking abilities, independent of network access or a central server.

**Difference between Git and GitHub**

* **Git** is a revision control system, a tool to manage your source code history.
* **GitHub** is a hosting service for Git repositories.
* So,they are not the same thing: **Git** is the **tool**, **GitHub** is the **service for projects that use Git**.
* To get your code to GitHub

**what is git and github?**

**Git :**

Now, git is nothing but is a free and open source distributed version control system.which is designed to handle everything from small to very large project with speed and efficiency

* Git is easy to learn and has tiny footprint with lightning fast performance
* Git is released under the [GNU General Public License version 2.0](http://opensource.org/licenses/GPL-2.0), which is an [open source license](http://www.opensource.org/docs/osd).
* **Git is vey fast**. With Git, nearly all operations are performed locally, giving it a huge speed advantage on centralized systems that constantly have to communicate with a server somewhere.
* Git was built to work on the Linux kernel, meaning that it has had to effectively handle large repositories from day one.
* Git is written in C, reducing the overhead of runtimes associated with higher-level languages.
* Speed and performance has been a primary design goal of the Git from the start.

**another feature of git is that git has integrity:**

Now what does it means? Everything in Git is checksummed before it is stored and is then referred to by that checksum.

This means it’s impossible to change the contents of any file or directory without Git knowing about it. This functionality is built into Git at the lowest levels and is integral to its philosophy.

You can’t lose information in transit or get file corruption without Git being able to detect it.

**The another phase is that git is only adds data :**

When you do actions in Git, nearly all of them only add data to the Git database.It is hard to get the system to do anything that is not undoable or to make it erase data in any way.

As with any VCS, you can lose or mess up changes you haven’t committed yet, but after you commit a snapshot into Git, it is very difficult to lose, especially if you regularly push your database to another repository.

**Now, moving further about the core part of git is that the three states.**

if you want the rest of your learning process to go smoothly. Git has three main states that your files can reside in: committed, modified, and staged:

* Committed means that the data is safely stored in your local database.
* Modified means that you have changed the file but have not committed it to your database yet.
* Staged means that you have marked a modified file in its current version to go into your next commit snapshot.



This leads us to the three main sections of a Git project: the Git directory, the working tree, and the staging area.

* The Git directory is where Git stores the metadata and object database for your project. This is the most important part of Git, and it is what is copied when you clone a repository from another computer.
* The working tree is a single checkout of one version of the project. These files are pulled out of the compressed database in the Git directory and placed on disk for you to use or modify.
* The staging area is a file, generally contained in your Git directory, that stores information about what will go into your next commit.

The basic git workflows goes on like as below:

1. You modify files in your working tree.
2. You selectively stage just those changes you want to be part of your next commit, which adds onlythose changes to the staging area.
3. You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

If any particular version of a file in the git directory, It is considered as committed. if it has been modified and was added to the staging area, it is staged. And if it was changed since it was checked out but has not been staged ,it is modified.

**Github:**

Before moving further about working of github what is github?

* GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.
* Github has certain essential like repositories, branches, commits, and Pull Requests.
* You can create your own repositories in github.

Certain steps regarding to create repositories in github which briefly explain as below :

**Step 1: to create repositories**

* A **repository** is usually used to organize a single project.
* Repositories can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs.
* We recommend including a README, or a file with information about your project.
* GitHub makes it easy to add one at the same time you create your new repository.
* It also offers other common options such as a license file.
* Your any repository can be a place where you store ideas, resources, or even share and discuss things with others.

Hence from the above briefs we can say that its works in simple way and very useful for backup.

Now, to create new repository certain steps has follow.

1. In the upper right corner, next to your avatar or identicon, click + and then select **New repository**.
2. Name your repository (i.e. datastructure ).
3. Write a short description.
4. Select **Initialize this repository with a README**.

**Step 2: to create new branch**

* **Branching** is the way to work on different versions of a repository at one time.
* By default your repository has one branch named master which is considered to be the definitive branch.
* We use branches to experiment and make edits before committing them to master.
* When you create a branch off the master branch, you’re making a copy, or snapshot, of master as it was at that point in time.
* If someone else made changes to the master branch while you were working on your branch, you could pull in those updates.

Now , how to create a branch

1. Go to your new repository (i.e. datastructure)
2. Click the drop down at the top of the file list that says **branch: master**.
3. Type a branch name, readme-edits, into the newbranch text box.
4. Select the blue **Create branch** box or hit “Enter” on your keyboard.

**step 3: make and commit changes**

* On GitHub, saved changes are called commits.
* Each commit has an associated commit message, which is a description explaining why a particular change was made.
* Commit messages capture the history of your changes, so other contributors can understand what you’ve done and why.

**Make and commit changes**

1. Click the README.md file.
2. Click the  pencil icon in the upper right corner of the file view to edit.
3. In the editor, write a bit about yourself.
4. Write a commit message that describes your changes.
5. Click **Commit changes** button.

Now, after a commit changes occurs another phase is that open the pull request.

**Step 4: Open a pull request**

Pull request are the core or we can say that the heart of the collobation of github.

When you open a pull request, you’re proposing your changes and requesting that someone review and pull in your contribution and merge them into their branch.

* Pull requests show diffs, or differences, of the content from both branches.
* The changes, additions, and subtractions are shown in green and red.
* As soon as you make a commit, you can open a pull request and start a discussion, even before the code is finished.
* You can even open pull requests in your own repository and merge them yourself.
* It’s a great way to learn the GitHub flow before working on larger projects.

**What is difference between git and github ?**

major or we can say that core difference between git and github is that Git is a distributed version control tool that can manage a development project's source code history, while GitHub is a cloud based platform built around the Git tool.

Git is a tool a developer installs locally on their computer, while GitHub is an online service that stores code pushed to it from computers running the Git tool.

Another key difference between git and github is that The key difference between Git and GitHub is that Git is an open-source tool developers install locally to manage source code, while GitHub is an online service to which developers who use Git can connect and upload or download resources.



One way to examine the differences between GitHub and Git is to look at their competitors.

 Git competes with centralized and distributed version control tools such as Subversion, Mercurial, ClearCase and IBM's Rational Team Concert.

On the other hand, GitHub competes with cloud-based [SaaS](https://searchcloudcomputing.techtarget.com/definition/Software-as-a-Service) and PaaS offerings, such as GitLab and Atlassian's Bitbucket.

Difference between git and github continue…

* Git is maintained by linux foundation since 2005.
* While github is maintaining by Microsoft since 2018.
* Git has have not user management tool.
* While github has built-in usermanagement tool.

**Conclusion**

As per the above explanation about working process of git and github we can conclude from that is git is distributed version control system written by the creator of linux.while github and similar services bring all of the benefits of a decentralized VCS to a centralized services.

**Bibliography**

[https://www.quora.com](https://www.quora.com/)

<https://guides.github.com/>

[https://www.theserverside.com](https://www.theserverside.com/)

[https://guides.github.com](https://guides.github.com/)