

**Dharmsinh Desai University**

**MCA SEM-II**

**DATA STRUCTURE USING C**

**ASSIGNMENT-1**

|  |  |  |
| --- | --- | --- |
| **ROLL NO** | **:** | **MA006** |
| **NAME** | **:** | **SHIVANI KACHHIYA** |
| **ID** | **:** | **18MAPOG001** |

**SUBMITTED TO**

**Prof. Himanshu Purohit**

**DIFFERENCE BETWEEN GIT AND GITHUB**

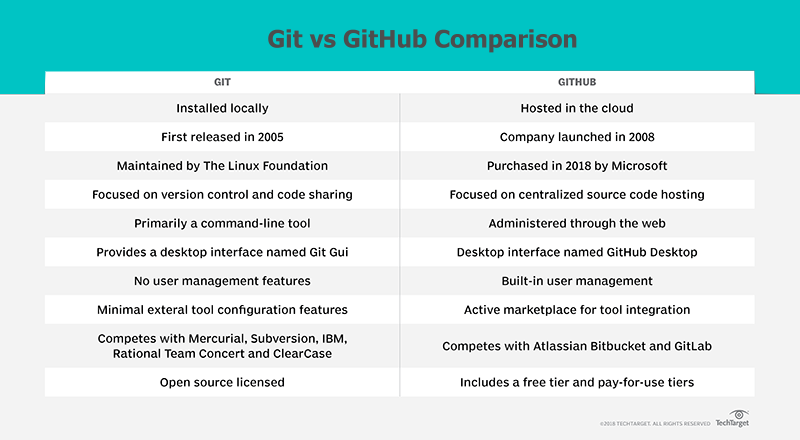
* **What is the difference between Git and GitHub?**

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. 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.

The Git tool runs locally on a computer, is completely self-sufficient, and it does not require the use of any external, cloud hosting service to function. Git was released in 2005, predating GitHub by three years.While cloud-based hosting services like GitHub and GitLab can complement the tool, Git itself does not require them, and a team of developers can version files, merge source code and work collaboratively in a distributed manner just fine without any SaaS or PaaS offerings.

If you look at the full [list of commands](https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet), you'll notice that Git focuses exclusively on tasks related to source code management, without any departure into peripheral topics, such as security, authentication, software quality or CI. For most developers, [Git commands](https://www.theserverside.com/tutorial/Five-basic-Git-commands-every-beginner-needs-to-know), such as commit, reset, push, revert, checkout, fetch and merge are self-evident in their purpose and thus easy to understand. And there are only about a dozen other Git commands, in addition to the seven just mentioned, that can be considered part of the porcelain. As such, you can learn Git and begin using it at an advanced level quickly, and since it supports all languages and file types, you can use it across the enterprise, regardless of whether development is happening in Java, Node.js, Python, HTML or .NET



### DVCS with Git vs. GitHub

The Git tool is popular with developers because is stays true to its purpose of versioning source code, [managing commit histories](https://www.theserverside.com/video/How-to-use-the-git-reset-hard-command-to-change-a-commit-history) and making it possible to share code between developers without deviating into peripheral fields. There is no feature bloat with Git. It does what it does, it does nothing else, and it makes no apologies for that fact.

Of course, the fact that Git does not extend its tentacles beyond the world of version control is both a blessing and a curse. After all, enterprises need a variety of services built around and integrated into their version control systems, from progress tracking systems, like Jira, to CI servers like [Jenkins](https://www.theserverside.com/tutorial/Attain-Jenkins-Git-integration-with-a-GitHub-pull-request). Furthermore, Git doesn't provide any out-of-the-box features for authenticating users against a centralized credentials repository, nor does it provide the ability to protect blessed master branches from inadvertent commits by an unwitting developer.

This is where vendors such as GitLab, Atlassian and GitHub enter the picture. They build services around Git and make those services available through a cloud-based platform. GitHub and GitLab charge nominal fees to enterprise customers, while often offering basic services at no cost to hobbyists and qualifying open source projects.

* **Useful Git and GitHub features**

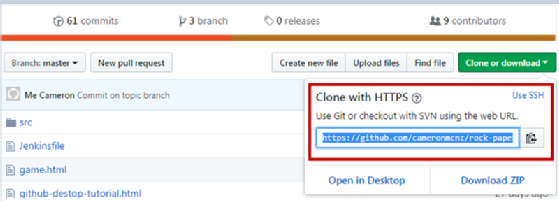
Beyond its role as a central source code repository, GitHub offers the following features that are not native to the Git tool:

* lightweight task tracking;
* the GitHub Desktop GUI tool for managing GitHub repositories;
* project wikis with extensive emoji support;
* online editing of files;
* gists for storing code snippets;
* a marketplace for third-party tool integration;
* branch protection rules;
* forking capabilities;
* organizational tools for managing teams;
* email notifications and alerting;
* [Secure Shell](https://searchsecurity.techtarget.com/definition/Secure-Shell) and GNU Privacy Guard security and encryption key management;
* project insights regarding traffic, code commit frequency and dependency graphs; and
* interaction limits for when arguments about software development heat up.

These are all features completely absent in Git but are nevertheless invaluable for enterprise software development teams. Building services and features around underlying Git technology is the value-add that vendors like GitLab, BitBucket and GitHub provide.

* **What is Git and GitHub integration?**

As you can see, the pertinent question to ask is not about GitHub vs. Git, but instead: *What is Git and GitHub integration, and how does it happen?* It's actually a relatively straightforward process.

The first step in answering this question is setting up a GitHub account and creating a repository. The GitHub repository will be given a URL that uniquely identifies it, such as: http://github.com/abc/rps.git.

GitHub URL uniquely identifies the Git repository.

With the repository created, developers can install Git locally and then clone the repository. Developers can then make local changes to project files and then push commits back to GitHub. To get updates to the [central GitHub repository](https://www.theserverside.com/tutorial/Attain-Jenkins-Git-integration-with-a-GitHub-pull-request), developers use pull and fetch commands.

### Are Git and GitHub the same thing or not?

For those asking "Is Git and GitHub the same thing", the answer is definitely no. There is clearly a difference between Git and GitHub. Git is implemented with a laser sharp focus on the tasks a software engineer needs to perform in order to version code and share it. GitHub plays the role of the cloud-hosted, online, Git repository that teams can use to store code centrally.

As you can see, there's no need to pit these two tools against each other. The two tools are complementary. It's not about arguing the merits of Git vs. GitHub. After all, the best [DevOps integrations](https://www.theserverside.com/video/Tackle-these-10-sample-DevOps-interview-questions-and-answers) are achieved when Git and GitHub are configured to work together.