

MIT WORLD PEACE UNIVERSITY

Full Stack Development  
Third Year B. Tech, Semester 5

---

---

VERSION CONTROL WITH GIT

---

---

LAB ASSIGNMENT 1

Prepared By

Krishnaraj Thadesar  
Cyber Security and Forensics  
Batch A1, PA 20

September 21, 2023

# Contents

<b>1 Aim</b>	<b>1</b>
<b>2 Objectives</b>	<b>1</b>
<b>3 Problem Statement</b>	<b>1</b>
<b>4 Theory</b>	<b>1</b>
<b>5 What is Git</b>	<b>1</b>
5.1 History . . . . .	1
5.2 Advantages . . . . .	1
5.3 Differences between Github and Git . . . . .	2
<b>6 What is Version Control</b>	<b>2</b>
<b>7 How to use Git for version controlling</b>	<b>2</b>
<b>8 Platform</b>	<b>3</b>
<b>9 Input and Output</b>	<b>3</b>
<b>10 Screenshots</b>	<b>4</b>
<b>11 Conclusion</b>	<b>7</b>
<b>12 FAQ</b>	<b>8</b>

## 1 Aim

Version control with Git.

## 2 Objectives

- To introduce the concepts and software behind version control, using the example of Git.
- To understand the use of 'version control' in the context of a coding project.
- To learn Git version control with Clone, commit to, and push, pull from a git repository.

## 3 Problem Statement

Created a public git repository for your team and submit the repo URL as a solution to this assignment, Learn Git concept of Local and Remote Repository, Push, Pull, Merge and Branch.

## 4 Theory

## 5 What is Git

**Definition 1** *Git is a distributed version control system that allows developers to track changes to their code over time. It was created by Linus Torvalds in 2005 and has since become one of the most popular version control systems in use today. Git is designed to be fast, efficient, and flexible, and it can be used for projects of any size, from small personal projects to large enterprise applications.*

### 5.1 History

- Git was created by Linus Torvalds in 2005.
- It was originally designed for Linux kernel development, but it has since been adopted by many other projects.
- Git is a distributed version control system that allows developers to track changes to their code over time.
- It is designed to be fast, efficient, and flexible, and it can be used for projects of any size, from small personal projects to large enterprise applications.
- Git is free software distributed under the terms of the GNU General Public License version 2.
- It is available for Linux, macOS, and Windows operating systems.

### 5.2 Advantages

1. Git is a distributed version control system, which means that every developer has a full copy of the repository on their local machine.
2. This allows developers to work offline and commit changes without having to connect to a central server.

3. Git is designed to be fast, efficient, and flexible.
4. It can be used for projects of any size, from small personal projects to large enterprise applications.
5. Git is free software distributed under the terms of the GNU General Public License version 2.
6. It is available for Linux, macOS, and Windows operating systems.

### **5.3 Differences between Github and Git**

1. Git is a version control system that allows developers to track changes to their code over time.
2. Github is a web-based hosting service for Git repositories.
3. Git is a command-line tool that can be used locally or remotely.
4. Github is a web-based service that allows developers to host their Git repositories online.
5. Git is free software distributed under the terms of the GNU General Public License version 2.
6. Github is a proprietary service owned by Microsoft.

## **6 What is Version Control**

**Definition 2** *Version control is a system that allows multiple people to work on a project simultaneously, keeps track of changes made to the project's files and directories, and provides the ability to revert to previous versions of the project if necessary. It is essential for collaborative software development and is used to manage code, documents, and other digital assets efficiently.*

## **7 How to use Git for version controlling**

1. First, you need to initialize a Git repository in your project's directory using the following command:

```
git init
```

2. Add your project files to the Git repository by using the following command:

```
git add .
```

3. Commit your changes with a descriptive message using the following command:

```
git commit -m "Initial commit"
```

4. You can create branches to work on specific features or fixes. To create a new branch, use the following command:

```
git branch feature-branch
```

5. Switch to the newly created branch:

```
git checkout feature-branch
```

6. Make your changes and commit them to the branch.
7. To merge your changes back to the main branch (e.g., master), use the following command:

```
git checkout master  
git merge feature-branch
```

8. Finally, push your changes to a remote repository (e.g., GitHub) for collaboration and backup:

```
git push origin master
```

## **8 Platform**

**Operating System:** Arch Linux x86-64

**IDEs or Text Editors Used:** Visual Studio Code

**Compilers or Interpreters:** None Required.

## **9 Input and Output**

1. The Repository "Anti Brutus" was created as part of the mini project in Full Stack Development.
2. It was created as a group effort between me and my team. And they were all added as collaborators.
3. It was then forked on everyone's Github Account, and then cloned to their local machines.
4. Several branches were created for different features and then merged to the main branch. The images of the log graph are attached as reference.

## 10 Screenshots

```
/Anti-Brutus > | main ± git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   server/App.js

no changes added to commit (use "git add" and/or "git commit -a")
krishnaraj@Krishnaraj-Arch > /run/media/krishnaraj/Programs/JavaScript
/Anti-Brutus > | main ± git add .
krishnaraj@Krishnaraj-Arch > /run/media/krishnaraj/Programs/JavaScript
/Anti-Brutus > | main + git commit -am "modified app.js"
[main 1869332] modified app.js
 1 file changed, 9 insertions(+)
krishnaraj@Krishnaraj-Arch > /run/media/krishnaraj/Programs/JavaScript
/Anti-Brutus > | main git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 488 bytes | 488.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
remote:
remote: GitHub found 1 vulnerability on KrishnarajT/Anti-Brutus's default branch (1 high). To find out more, visit:
remote:   https://github.com/KrishnarajT/Anti-Brutus/security/dependabot/1
remote:
To https://github.com/KrishnarajT/Anti-Brutus
   c49bacf..1869332  main -> main
krishnaraj@Krishnaraj-Arch > /run/media/krishnaraj/Programs/JavaScript
```

Figure 1: Status and Commit to the Repository.

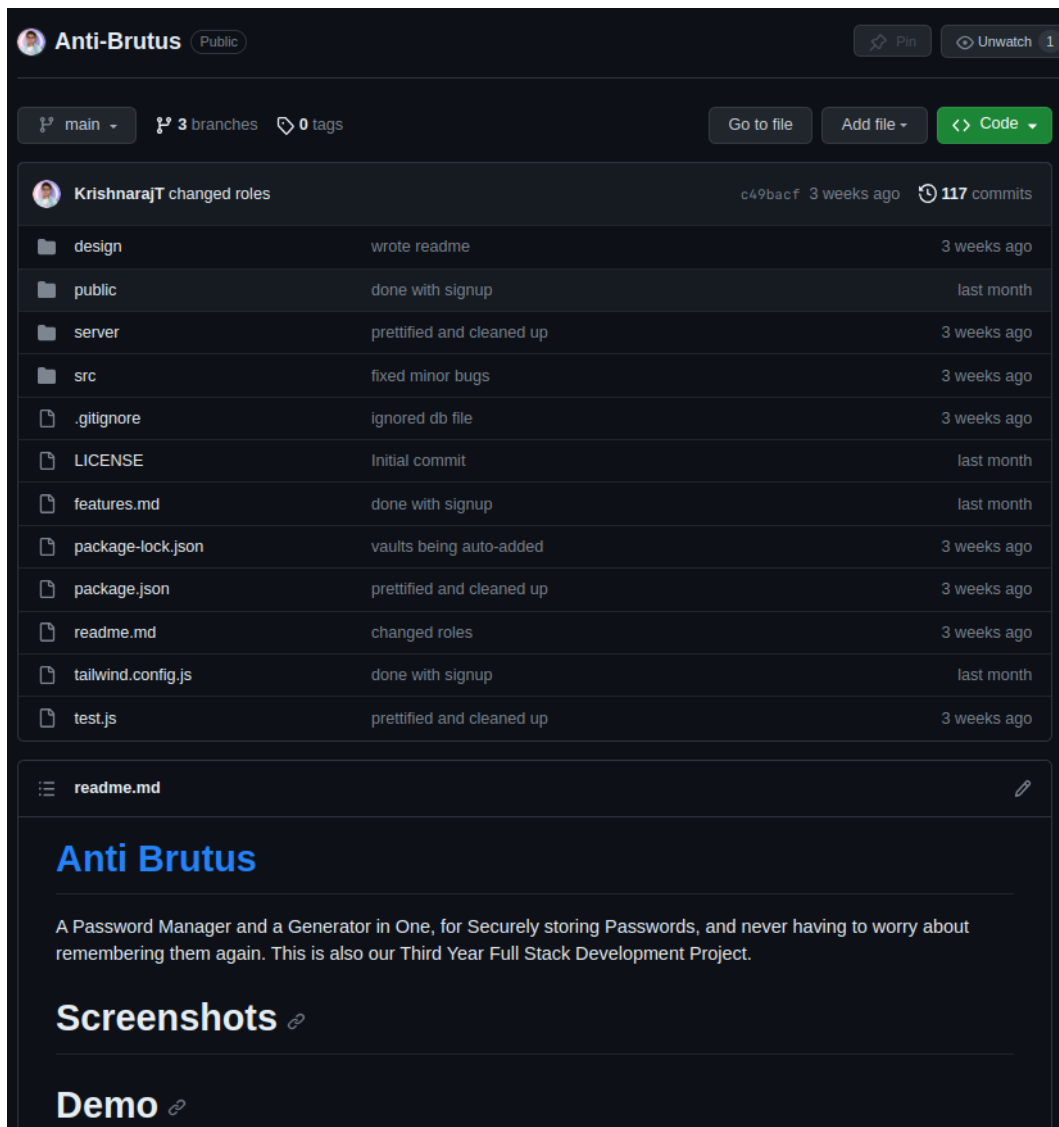


Figure 2: The Repository on Github.

```

Merge pull request #38 from KrishnarajT/dev

tested more functions, add new routes

*   commit 1e064b8
|\  Merge: e841dbc 61ece91
| | Author: Krishnaraj Thadesar <krishnaraj.kpt@outlook.com>
| | Date:   3 weeks ago
| |
| | Merge branch 'dev' of https://github.com/KrishnarajT/Anti-Brutus into dev
| |
*   commit 61ece91
|\  Merge: 57d47e6 de878c5
| | Author: Krishnaraj Thadesar <krishnaraj.kpt@outlook.com>
| | Date:   3 weeks ago
| |
| | Merge pull request #37 from Parth4123/main
| |
| | delete functionalities
| |
*   commit de878c5
| | Author: Parth <parthzarekar@gmail.com>
| | Date:   3 weeks ago
| |
| | delete functionalities
| |
*   commit 57d47e6
|\  Merge: b69a002 71904c5
| | Author: Krishnaraj Thadesar <krishnaraj.kpt@outlook.com>
| | Date:   3 weeks ago
| |
| | Merge pull request #36 from Parth4123/main
| |
:|
```

Figure 3: Log Graphs of Commits and Branches.

```

* | | | | commit 88acb9e
| | | | | Author: Parth Zarekar <90327206+Parth4123@users.noreply.github.com>
| | | | | Date:   5 weeks ago
| | | | |
| | | | | readme.md (#4)
| | | | |
| | | | | Co-authored-by: Krishnaraj Thadesar <krishnaraj.kpt@outlook.com>
| | | | |
* | | | | commit 1c8fb3e
| | | | | Author: sourab777karad <119159021+sourab777karad@users.noreply.github.com>
| | | | | Date:   5 weeks ago
| | | | |
| | | | | krish (#3)
| | | | |
| | | | | * Update readme.md
| | | | |
| | | | | * Update readme.md
| | | | |
| | | | | -----
| | | | |
| | | | | Co-authored-by: Krishnaraj Thadesar <krishnaraj.kpt@outlook.com>
| | | | |
* | | | | commit e05c1aa
| | | | | Author: Krishnaraj Thadesar <krishnaraj.kpt@outlook.com>
| | | | |
:|
```

Figure 4: Log Graphs of Commits and Branches.



```
* | | | commit c0b15dd
| \ \ \ Merge: 71a9dc6 7bbef88
| | | | Author: Krishnaraj Thadesar <krishnaraj.kpt@outlook.com>
| | | | Date: 3 weeks ago
| | | |
| | | | i still dont know
| | | |
| * | | | commit 7bbef88
| | \ \ \ Merge: 309a175 8a8354e
| | | | / Author: Krishnaraj Thadesar <krishnaraj.kpt@outlook.com>
| | | | / Date: 3 weeks ago
| | | |
| | | | Merge pull request #33 from KrishnarajT/dev
| | | |
| | | | encryption firebase and password css
| | | |
| | * | | | commit 8a8354e
| | | \ \ \ Merge: a577f16 b6195d5
| | | | | Author: Krishnaraj Thadesar <krishnaraj.kpt@outlook.com>
| | | | | Date: 3 weeks ago
| | | | |
| | | | | Merge pull request #32 from Parth4123/main
| | | | |
| | | | | connected to realtime db in firebase and written encryption functions
| | | | |
| | | * | | | commit b6195d5
| | | / / | Author: Parth <parthzarekar@gmail.com>
| | | / / | Date: 3 weeks ago
| | | | |
| | | | |
```

Figure 5: Log Graphs of Commits and Branches.

## 11 Conclusion

Thus, we have successfully learnt, understood and implemented the concepts of Version Control with Git.

## 12 FAQ

### 1. *What is branching in Git?*

Branching in Git is a powerful feature that allows developers to work on multiple independent lines of development within the same repository. It essentially creates a copy of the codebase at a specific point, allowing you to make changes without affecting the main codebase. Branches are often used for implementing new features, fixing bugs, or isolating experimental changes.

### 2. *How to create and merge branches in Git? Write the commands used.*

To create and merge branches in Git, you can use the following commands:

#### **Creating a New Branch:**

To create a new branch and switch to it, you can use the following command:

```
git checkout -b new-branch-name
```

This command creates a new branch named "new-branch-name" and switches to it.

#### **Merging a Branch:**

To merge changes from one branch into another (e.g., merging a feature branch into the main branch), use the following commands:

First, switch to the target branch (e.g., main branch):

```
git checkout main
```

Then, merge the feature branch into the target branch:

```
git merge feature-branch
```

This command incorporates the changes from "feature-branch" into the "main" branch.

#### **Deleting a Branch:**

After the branch has been merged and is no longer needed, you can delete it:

```
git branch -d feature-branch
```

This command deletes the "feature-branch" locally.