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

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 collbaorators.

3. It was then forked on everyones 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

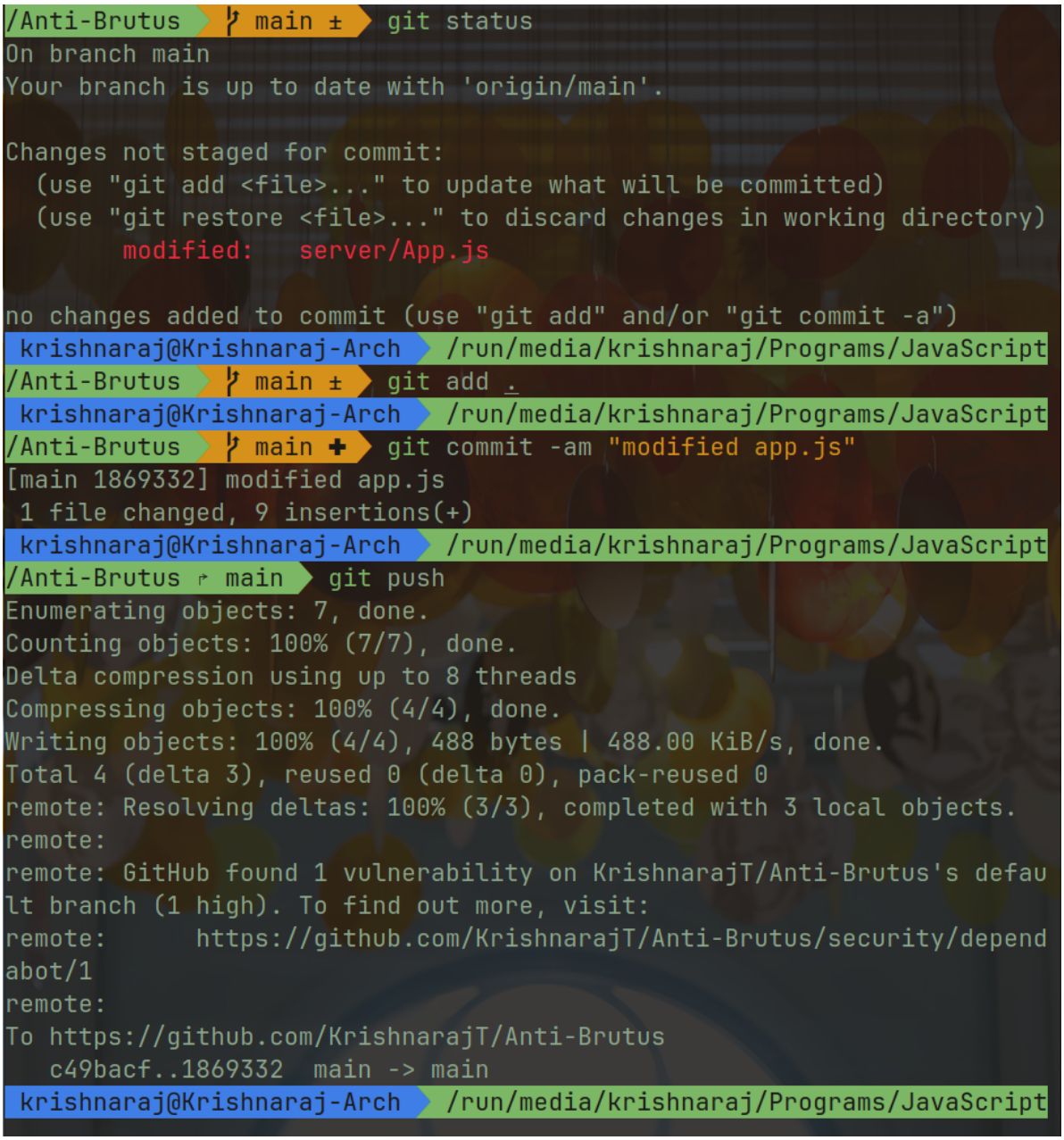

Figure 1: Status and Commit to the Repository.

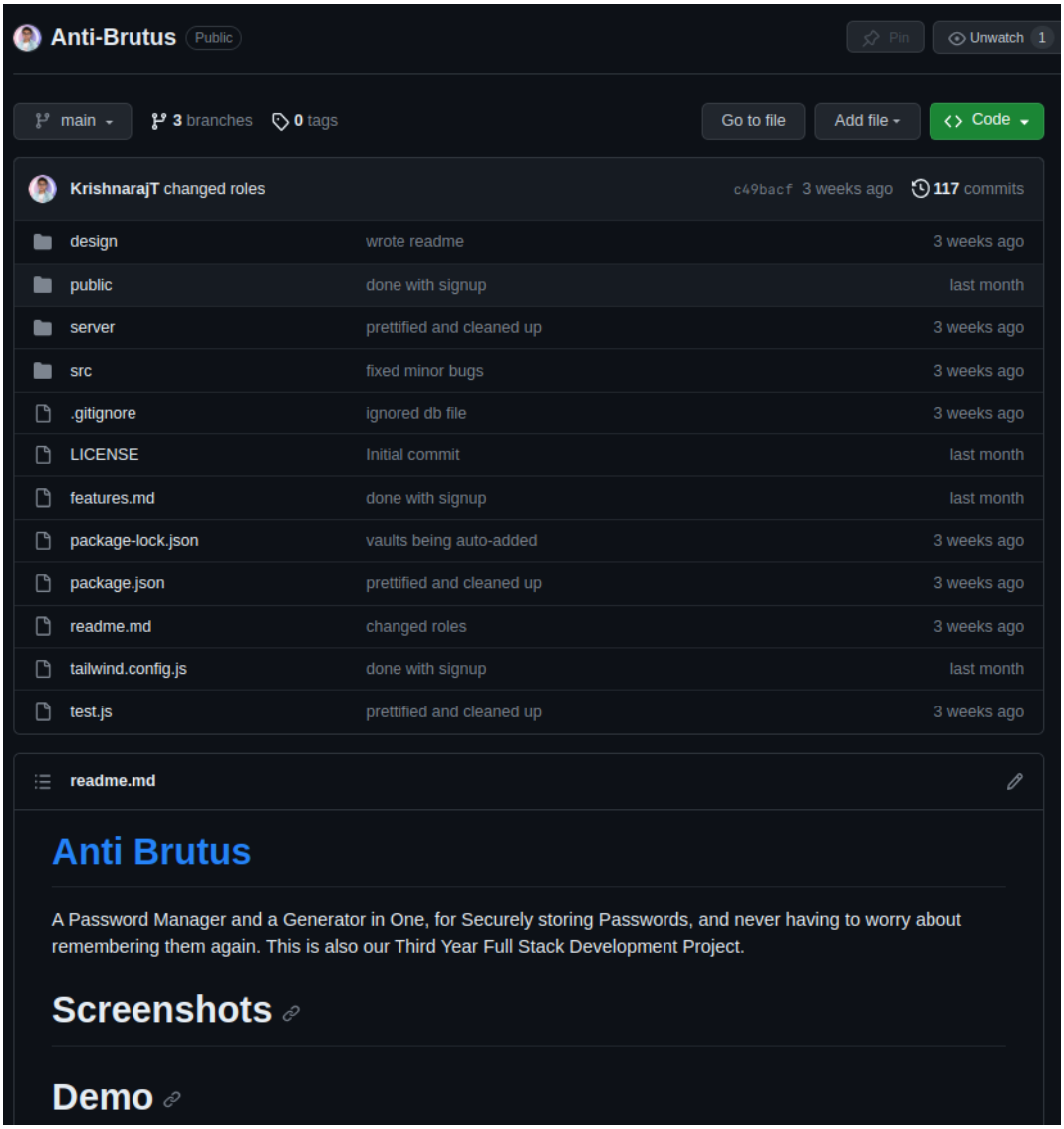

Figure 2: The Repository on Github.

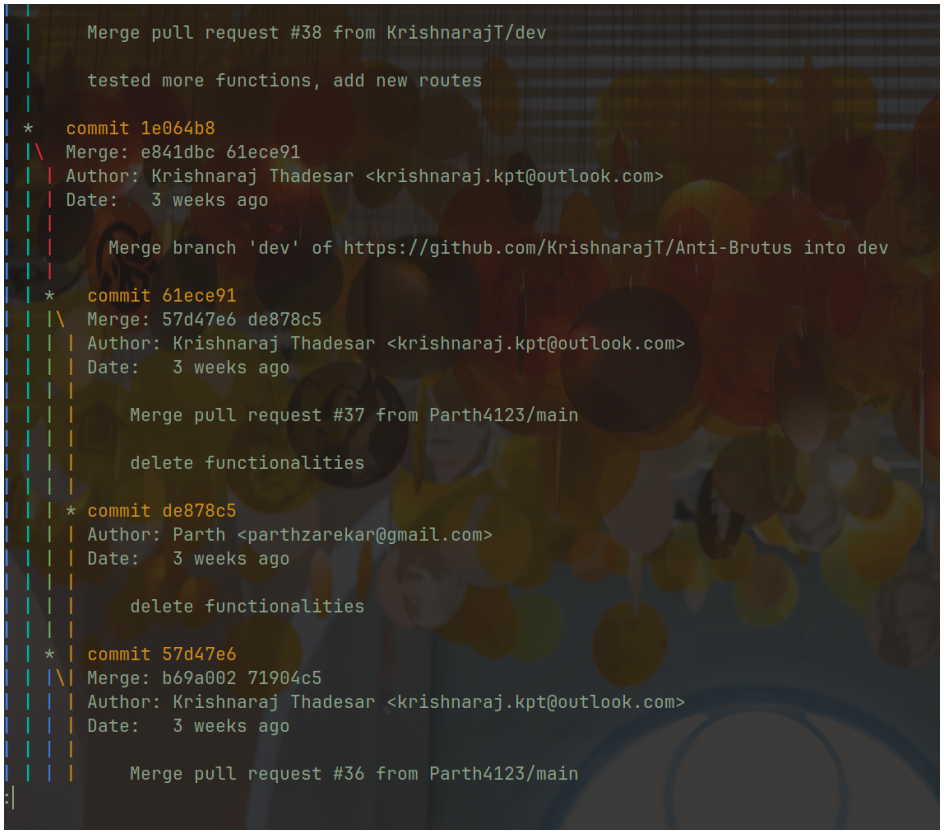

Figure 3: Log Graphs of Commits and Branches.

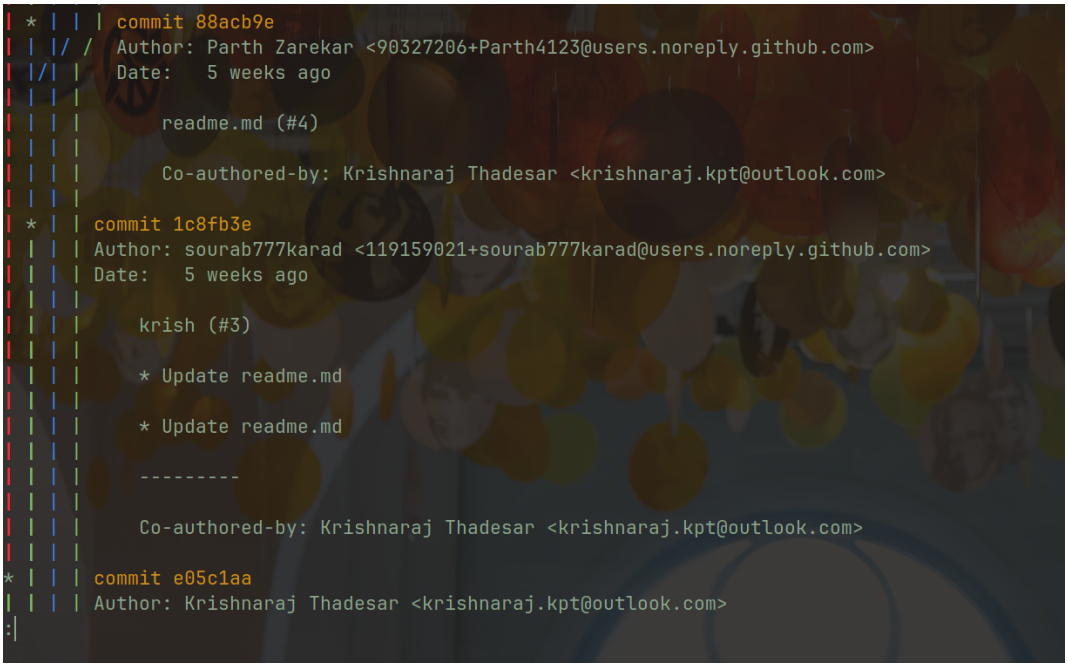

Figure 4: Log Graphs of Commits and Branches.

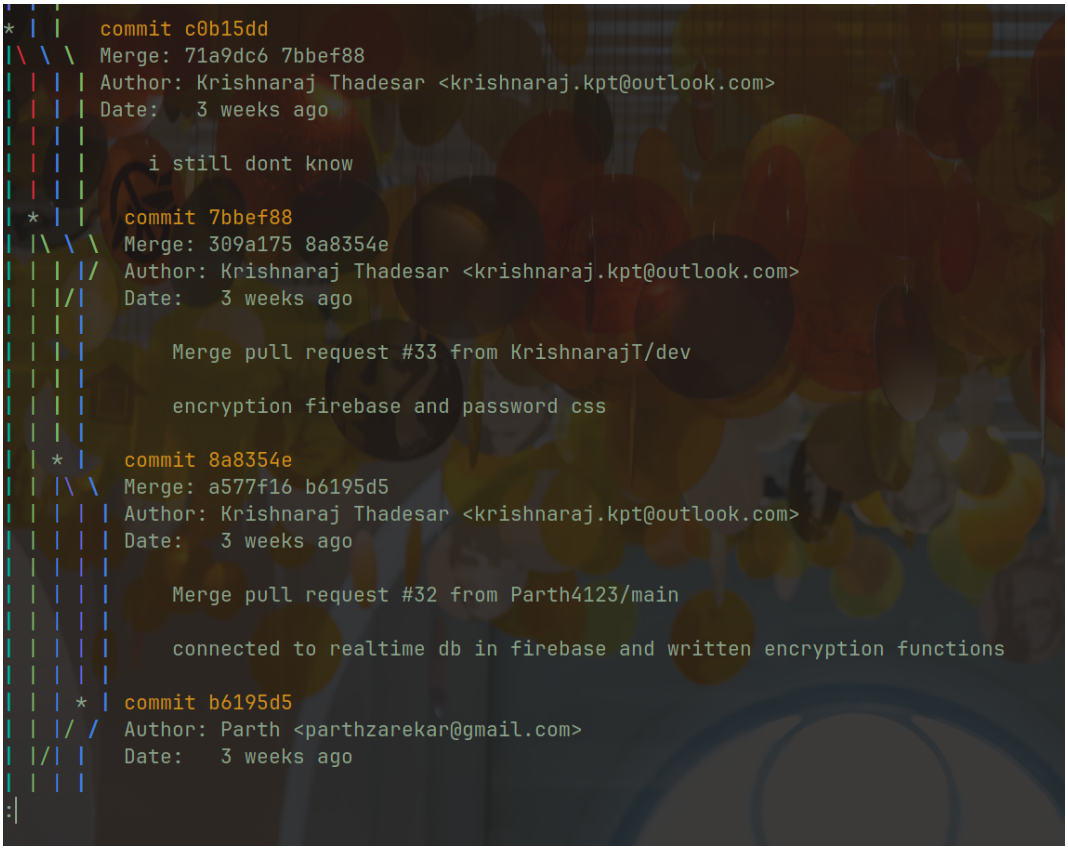

Figure 5: Log Graphs of Commits and Branches.

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.