**Experiment No. 1**

**Aim:** To study & use Distributed Version Control System

**Objective:** To understand Version Control & Its use

**Outcome:** Able to use version Control system & its use

**Theory:**

Version control is a system that records changes to a file or set of files over time, so that you can recall specific versions later. It's most commonly used in software development, where multiple people may need to make changes to the same codebase. With version control, you can track who made which changes, when, and why, and revert back to earlier versions if necessary. The most widely used version control system is Git, but there are others such as SVN, Mercurial, and CVS. By using version control, teams can work together on code and collaborate more effectively, reducing the risk of lost or overwritten work.

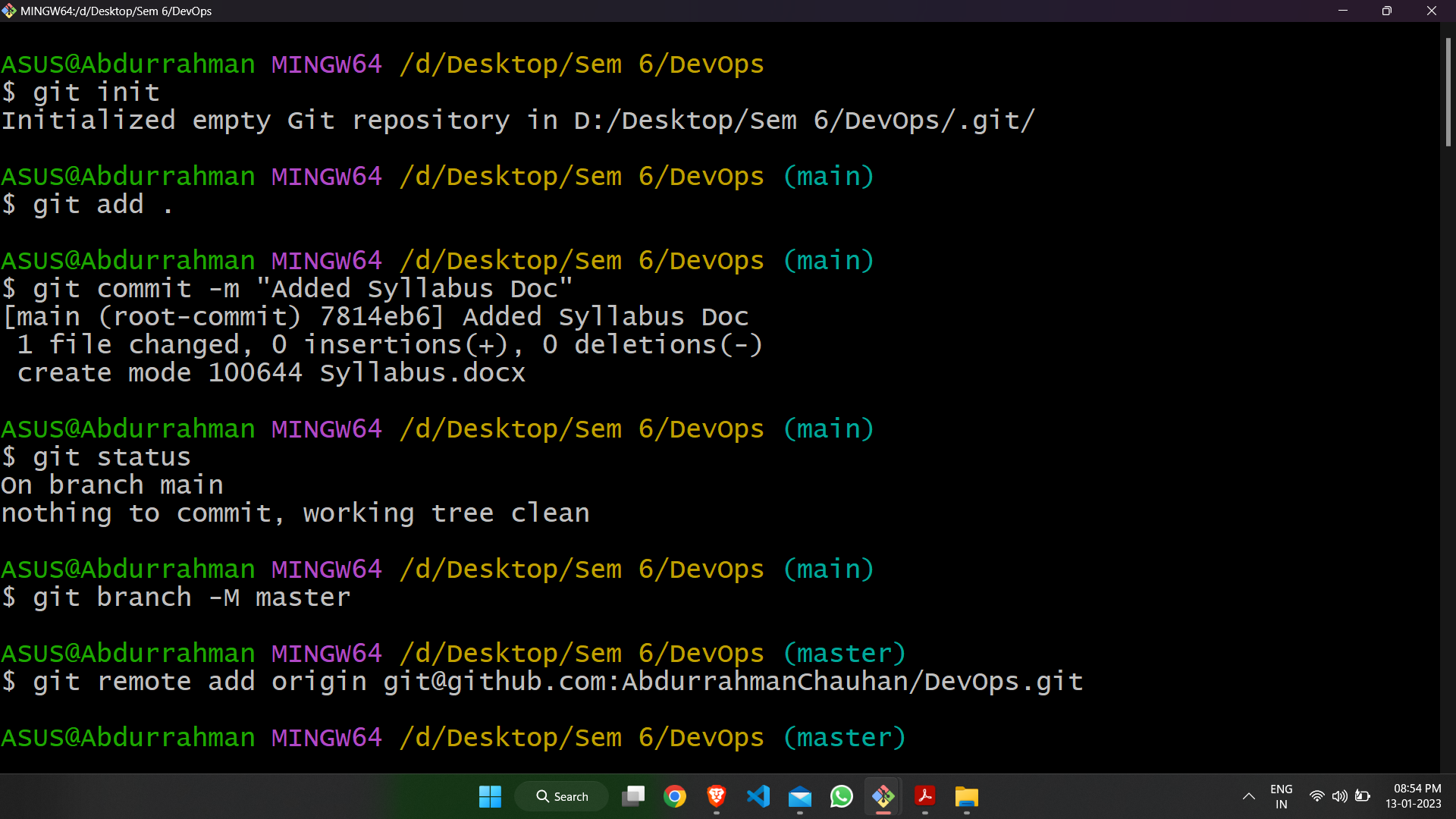
Here are some of the most commonly used types of version control systems:

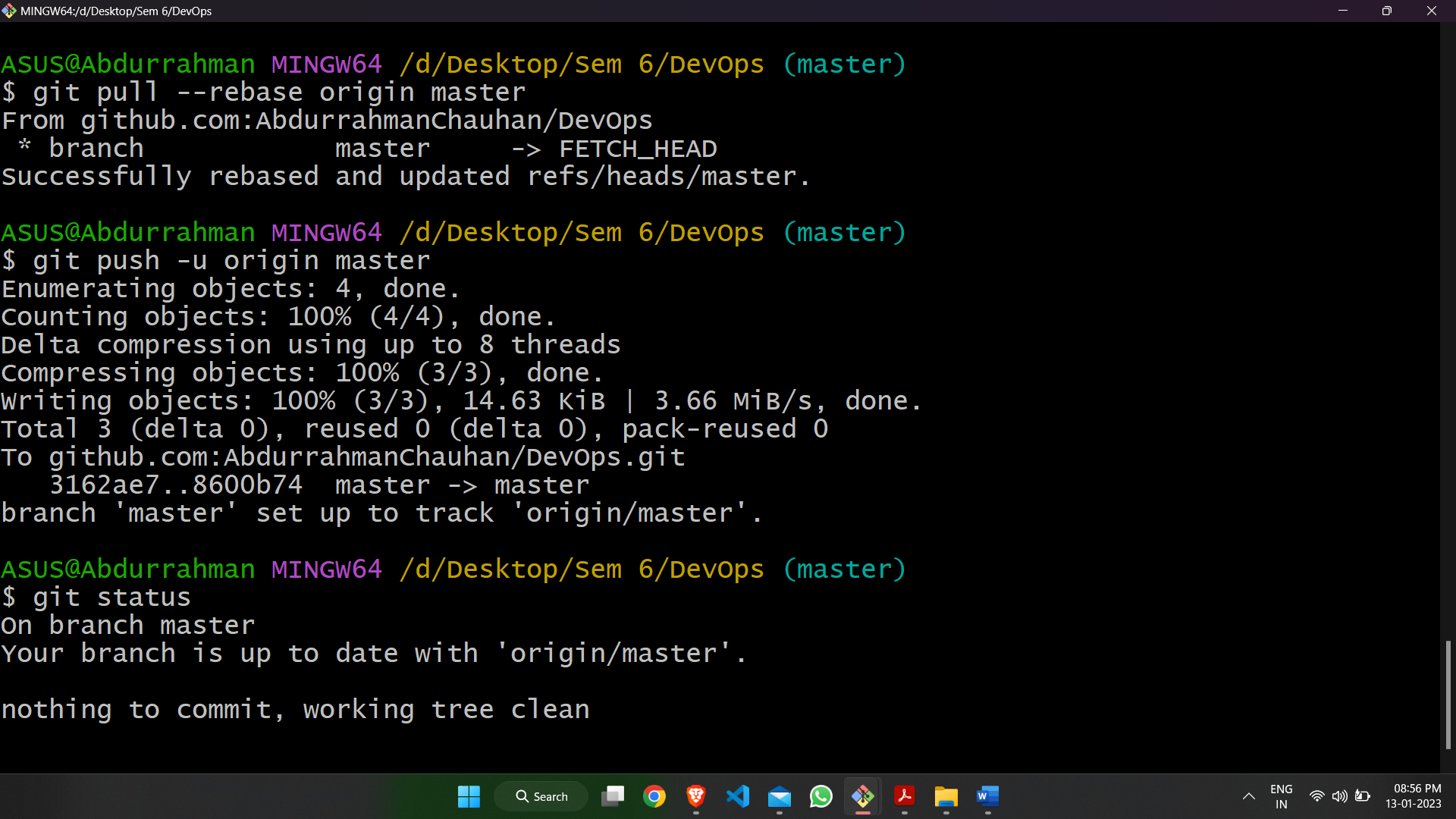
1. Centralized Version Control Systems (CVCS): CVCS use a central server to store all versions of a project's files. Examples include Subversion (SVN) and Concurrent Versions System (CVS).
2. Distributed Version Control Systems (DVCS): DVCS don't rely on a central server to store all versions of a project's files. Instead, each user has a full copy of the repository, and changes are synced between users. Examples include Git and Mercurial.
3. Lock-Modify-Unlock (LMU) Systems: LMU systems allow only one user to modify a file at a time. Other users can read the file, but they must wait for the first user to release the file before they can make changes.
4. Optimistic Concurrency Control (OCC) Systems: OCC systems allow multiple users to make changes to a file simultaneously. The version control system then merges the changes together and resolves any conflicts.
5. Label/Baseline Systems: Label/baseline systems allow you to "tag" a specific version of the code with a label, making it easy to recall that version later. This type of version control is often used in more structured environments, such as in regulated industries.

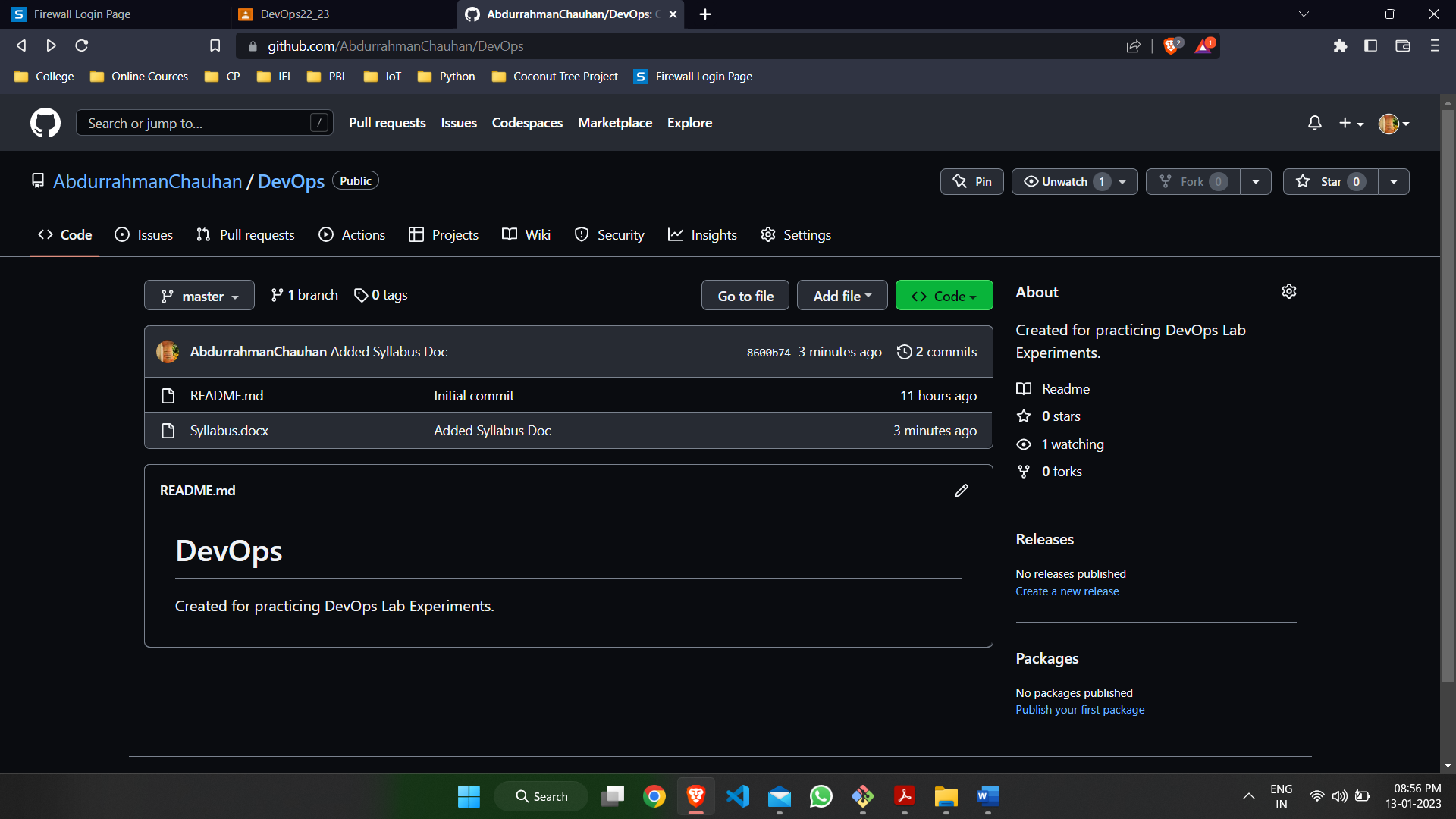
Distributed Version Control (DVCS) is a type of version control system where each user has a full copy of the repository, including the entire history of changes to the files. This allows users to work independently of a central server and makes it possible to keep working even when offline. When a user is connected, they can sync their changes with other users, creating a distributed network of repositories. The most popular example of a DVCS is Git.

In DVCS, users can commit changes to their local repository and then push those changes to remote repositories, where they can be pulled down by other users. This allows for more flexible and efficient collaboration, as users can work in parallel and merge their changes later. DVCS also provides greater reliability and safety, as users can recover from network failures or data loss on the central server. Overall, DVCS provides a more flexible, scalable, and resilient way to manage source code compared to centralized version control systems.

**Output:**









**Conclusion:**

Git is a widely used distributed version control system (DVCS) that plays an important role in many DevOps processes and workflows. DevOps emphasizes collaboration and communication between development and operations teams, and Git supports this by allowing multiple people to work on the same codebase and track changes over time.

Git's distributed architecture and branching and merging capabilities allow teams to work in parallel and collaborate effectively, reducing the risk of lost or overwritten work. Git also integrates well with other DevOps tools and workflows, such as continuous integration and delivery (CI/CD) pipelines, making it a valuable component in the DevOps toolchain.

In summary, Git is a key tool for DevOps teams to manage source code, collaborate efficiently, and improve the overall speed and quality of delivering software