# Recap: Version Control

## Introduction

Generally, a software product is developed in collaboration by a group of developers, each of whom is responsible for certain functionality of the product.

Everyone modifies the source code accordingly. However, suppose some change in the code results in a particular bug in the product. In that case, there must be a way to track the cause of the problem and seamlessly restore the state of the product to its previously working position.

## A Version Control System

A Version Control System (VCS) records changes to one or more files over time, so you can fall back to any specific version if required. Different types of version control mechanisms are employed.

A manual way is to store each version of source code files in separate folders, preferably with a time stamp, so that one can restore the state to the desired version. This approach can be very cumbersome and unsuitable for a collaborative development environment.

## Centralized version control system

In comparison, a centralized version control system works well. A single server stores the various versions of the source code. The administrator of the server controls the activity of the other collaborators.

However, if the server encounters any issues, there is a risk of losing all the work done.

Distributed version control system

The distributed version control system is far more reliable and efficient. In this system, each user has a local copy of the code called a repository.

During the development, developers make changes to the working copy of the repository and commit those changes to the local repository.

The changes in the local repository updated are then pushed to the central repository.

Changes from two or more independent branches and subsequently merged.

The merging or integration operation reconciles the multiple changes to a version-controlled file collection.

## Git

Git is a popular distributed version control system (DVCS). It's free and open-source software designed to manage all source code history.

You can keep a history of commits, revert to previous versions, and share code.

To collaborate with it, download and install Git on your machine from <https://git-scm.com/downloads>

.

For example, open the Git terminal on your machine and initialize a local repository for the Little Lemon project.

$ git init littlelemon (to create also the folder)

$ git init (if you have already the folder)

**Add the project files to the folder and check the status**

**$ cd littlelemon**

$ git status

On branch master

No commits yet

**Add the files to the staging area of the repository**

**$ git add .**

**Configure the repository with relevant credentials**

**$ git config --global user.email "you@example.com"**

$ git config --global user.name "Your Name"

**Commit the files that have been added to the staging area**

**$ git commit -m "first commit"**

**Then create a repo on Github and copy the https link, the link your local repo with that one online**

**$ git remote add origin** https://github.com/RobMae82/littlelemon.git

**the push it if you want it on your GITHUB**

**$ git push -u origin master**

**Username for 'https://github.com': RobMae82**

**Password for 'https://RobMae82@github.com':**

**Enumerazione degli oggetti in corso: 6, fatto.**

**Conteggio degli oggetti in corso: 100% (6/6), fatto.**

**Compressione delta in corso, uso fino a 4 thread**

**Compressione oggetti in corso: 100% (6/6), fatto.**

**Scrittura degli oggetti in corso: 100% (6/6), 371.06 KiB | 8.83 MiB/s, fatto.**

**6 oggetti totali (0 delta), 0 riutilizzati (0 delta), 0 riutilizzati nel file pack**

**remote:**

**remote: Create a pull request for 'master' on GitHub by visiting:**

**remote: https://github.com/RobMae82/littlelemon/pull/new/master**

**remote:**

**To https://github.com/RobMae82/littlelemon.git**

**\* [new branch] master -> master**

**ramo 'master' impostato per tracciare 'origin/master'.**

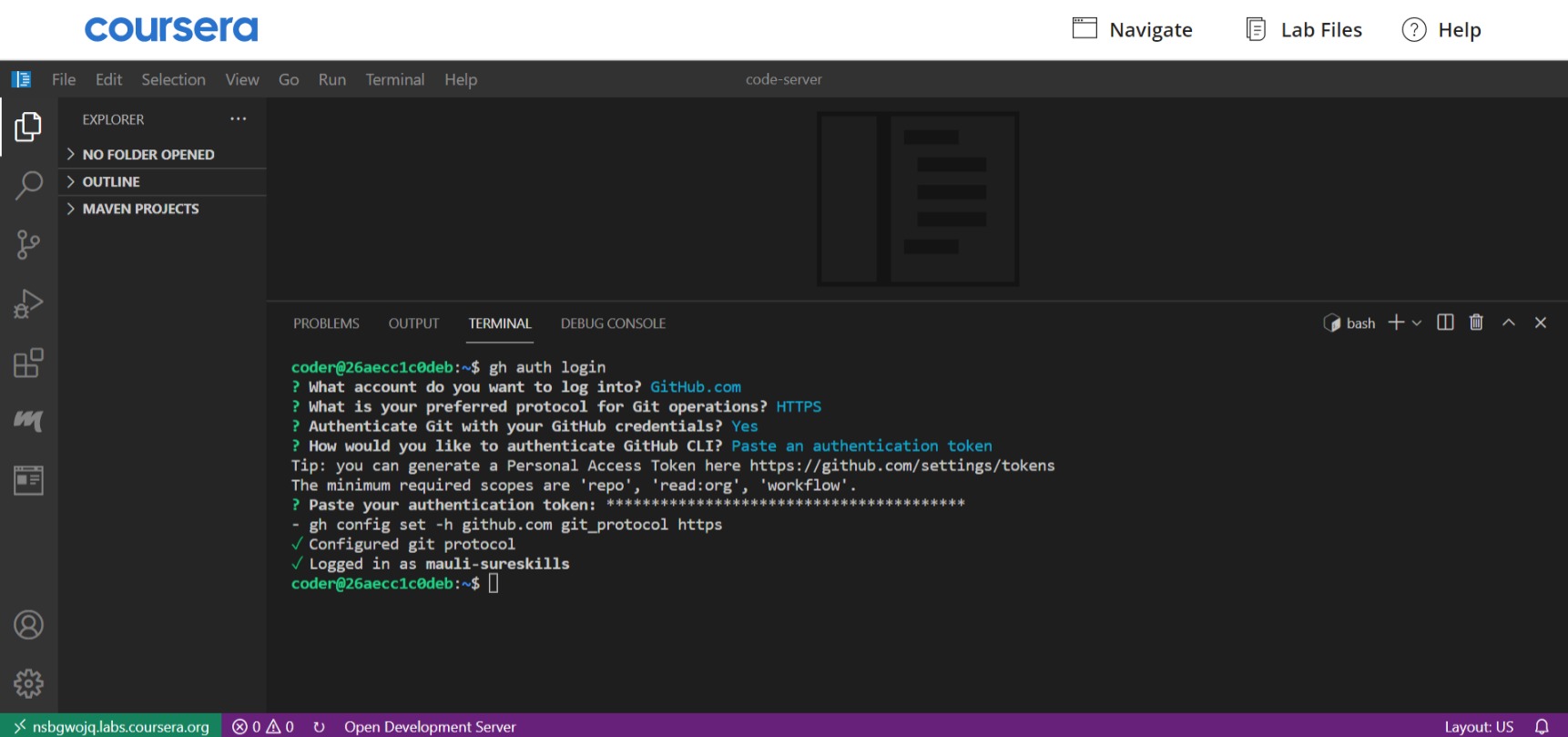
## ****The best is to start from GITHUB****

****Create a repo there then:****

Open the Terminal and authenticate using gh (Github CLI):

$ gh auth login

(install it if you don’t have apt install gh)



**Clone the repository using its GitHub CLI.**

gh repo clone <YOUR USERNAME>/<REPOSITORY-NAME>

╭─tommy@hp ~/GITHUB\_projects

╰─➤ gh repo clone RobMae82/littlelemon

Clone in 'littlelemon' in corso...

remote: Enumerating objects: 3, done.

remote: Counting objects: 100% (3/3), done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

Ricezione degli oggetti: 100% (3/3), fatto.

Then $ cd littlemen and $ git status

****Then** Move to the repo directory by using cd <REPOSITORY-NAME> and you can finally push to the repo.**

## ****GitHub****

Usually, you store Git repositories on GitHub. It has the source control management (SCM) features of Git and some other features such as project management, support ticket management, and bug tracking.

Developers use GitHub to share their repositories, access other developers' repositories, and store remote copies of repositories to serve as backups.

Sign up to <https://github.com/>

and create a new repository on it.

Go back to the Git terminal and push the local repository to the GitHub repository.

## Summary

In this Reading, you learned what version control and Git is. You also learned how to create and push a local repository on GitHub.