# What is a VCS?

A version control system or VCS, also know as revision control or source control system, is a software utility that tracks and manages changes to a filesystem. A VCS also offers collaborative utilities to share and integrate these filesystem changes to other VCS users. When operating at the filesystem level, a VCS will track the addition, deletion, and modification actions applied to files and directories. A repository is a VCS term which describes when VCS is tracking a filesystem. In the scope of individual source code files, a VCS will track additions, deletions, modifications of the lines of text within that file.

VCS is an invaluable tool with many benefits to a collaborative software team workflow. Any software project that has more than one developer maintaining source code files should absolutely use a VCS. Additionally, sole-maintainer projects will also greatly benefit from utilizing a VCS. There is arguably no valid reason to forgo the use of a VCS in any modern software development project.

# Why to use it?

**Collaboration**

With a VCS, everybody on the team is able to work absolutely freely — on any file at any time. The VCS will later allow you to merge all the changes into a common version. There’s no question where the latest version of a file or the whole project is. It’s in a common, central place: your version control system.

**Rollback and undo changes to source code**

Once a VCS has begun tracking a source code file system, it keeps a history of changes and the state of the source code throughout a project’s history. This enables the possibility to “undo” or rollback a source code project to a last well-known state. If a bug is discovered in a live application, the code can be quickly reverted to a known stable version.

**Understanding What Happened**

Every time you save a new version of your project, your VCS requires you to provide a short description of what was changed. Additionally (if it’s a code / text file), you can see what exactly was changed in the file’s content. This helps you understand how your project evolved between versions.

# How to setup Git in Angular project and Commit to Github/GitLab

Github is used to track code changes and manage issues for an application. It helps to maintain a history of code changes by multiple team members and contributors. Using git versioning we can travel back in time to fetch the old version of code.

This Angular post is compatible with Angular 4 upto latest versions, Angular 7, Angular 8, Angular 9, Angular 10, Angular 11 & Angular 12

In this article, we will go through steps to quickly set up Git versioning in the Angular project.

Let’s start from scratch by creating a new Angular project.

**Step 1) Create a new Angular project**

Using NG CLI tool, make sure you have installed the latest version by running below code:

$ npm install -g @angular/cli

create a new project by running following command in the terminal window:

$ ng new my-git-project

After creating project go to its root ~cd my-git-project and open with VS code by running ~code . command.

**Step 2) Download and Install [Git](https://git-scm.com/" \t "_blank)**

If you are creating project in other then **C drive**, you may need to set **Environment Variable** path to this *C:\Program Files\Git\bin*

After that reopen the VS Code to take this in effect.

You can also open VS Code first from its launcher and open your project's folder from the code editor.

Now, you can use the [built-in terminal](https://code.visualstudio.com/docs/editor/integrated-terminal) in Visual Studio Code which can be opened via the keyboard shortcut **CTRL + `** on Linux, macOS, or Windows.

When you have created a project with Angular CLI, it has also initialized an empty GIT repository in your project and even made a first commit for the generated source code.

You can make sure of that by looking at the log of the git commits inside your project's folder by simply running the following command:

$ git log

You'll have a similar output:

commit b2f1d7463e2287035a0f94d011d32a4b432d3fbe (HEAD -> master)

Author: techiediaries <email@gmail.com>

Date: Fri Apr 17 12:50:28 2020 +0100

initial commit

**Step 3) Run git init command**

Now open the terminal window, by pressing **Ctr+`**if in VS code to run git init command to initialize git.

This will create a **.git** folder and **.gitignore** file keeping track of files and folder which **will not** be versioned by Git.

**Step 4) Execute Git Command to Commit locally**

Before committing files to Github repository online, we will execute some commands to create a bunch of files to be committed.

git status displays the files which are new/ changed but not committed yet.

git add . add files and folders need to be committed. The **.** means all new/ changed files.

git commit -m "Initial commit" commits the file with a message.

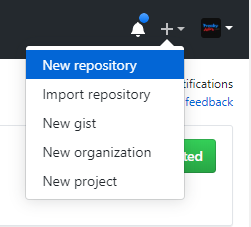
**Step 5) Register**[**GitHub**](https://github.com/)**and Create a repository**

Next, we are ready to move our committed files to an online repository like GitHub or GitLab.

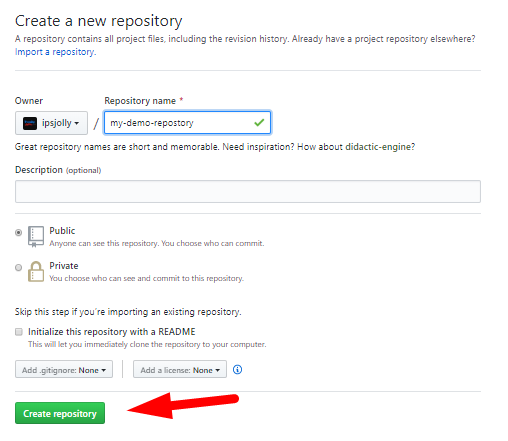
Here we will discuss GitHub steps to create a new repository:

Sign Up/ Sign In on the [GitHub](https://github.com/) portal

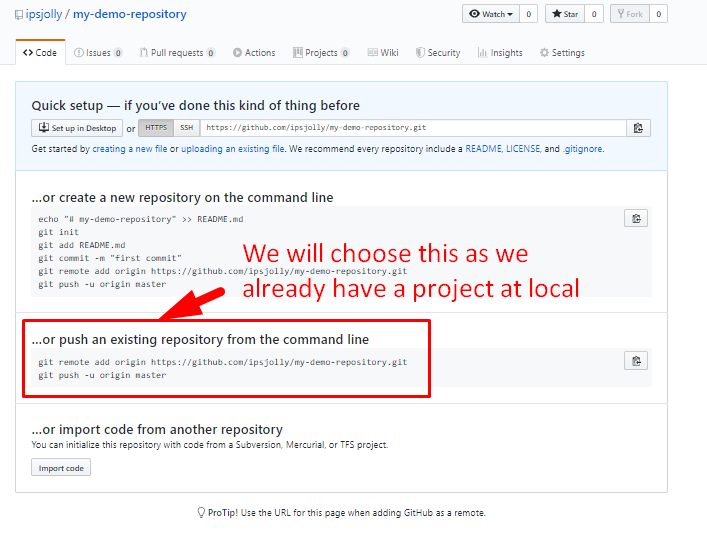
Now hover on the top right **+** icon, then click the **New repository** link



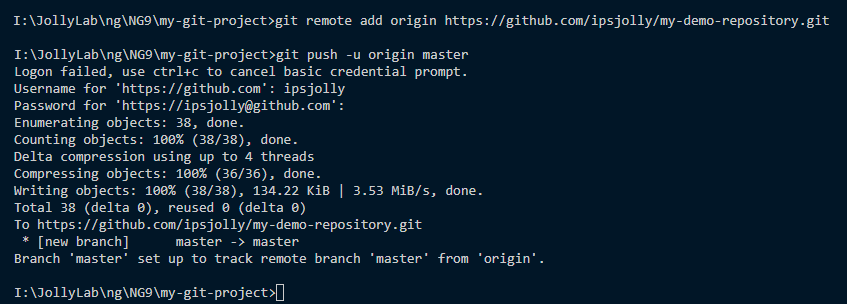
Fill the project repository name and visibility setting private/ public, then click Create repository button



After that, you need to run some commands shown on the next screen



That’s it now you have successfully committed your files on GitHub repository!

[](https://www.freakyjolly.com/wp-content/uploads/2020/03/Screenshot_2_.png)

Just refresh the repository page on GitHub to see committed files:

Now when you will make modifications to any files or folders, follow the same procedure in step 4 above to commit your changes.

git status

git add .

git commit -m "changes in the index.html"

git push origin

## Linking a Remote GitHub Repository

Git is a distributed source control system which doesn't need a central repository server like GitHub but for the sake of sharing your code with other developers in your team or worldwide, GitHub is the right platform nowadays!

If you want to collaborate with your team on the same Angular project or you just want to keep a remote copy of your project's source code, then you need to add a remote GitHub repository to your Git repository which will allow you to push, pull and synchronize the code between the local and remote repositories at any point.

The first step is to set up your GitHub user's credentials (email and name used when you signed up for your GitHub account) with Git.

Head back to your terminal and run the following commands:

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

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

You can actually add any name you want but make sure you use the right email address.

Next, you need to go to your GitHub account and create a new empty repository. Make sure to note the URL of your repository.

### Pushing your Angular 10 Code to GitHub

Open the src/index.html file and add some changes. You can change the title for example.

Head to the **Source Control panel** where you can see that your changes are recognized by Git. You'll see the name of the changed file with the letter **U** next to it. **U** stands for untracked file, which marks a file that is new or changed, but has not yet been added to the repository.

Next, you need to click on the **plus** icon for the index.html file to track the file by the repository.

After that, the letter next to the file will become **A**. which means the file that is added to the repository.

Next, you need to commit your changes, enter a commit message in the input box at the top of the **Source Control panel** and, click the **check** icon to make the commit.

After adding your remote repository to your local Git repository using the following command:

$ git remote add origin <URL-OF-YOUR-REPOSITORY>

You can push your code to GitHub using either the command-line or VS Code:

$ git push