**Uploading your project to GitHub**

## Step 1: Planning the move

Uploading your project to GitHub gives you the feature-rich tools and collaboration needed to elevate your project to the next level. Not to mention, it's also pretty exciting. If you're doing this for the first time, you have a few options when uploading your project to GitHub. This course will guide you through the necessary steps to upload a local project to be hosted on GitHub.

I know some people like to get straight to the point while others like more information. For those who like more information, be sure to check out the drop-downs like this one ⬇️

Why move to GitHub?

### Why move to GitHub?

You may be wondering what this GitHub thing is all about and why you should use it. If this sounds like you, here are a few reasons to make GitHub your project's new home:

* **Version control** — Everything on GitHub is stored in [Git](http://git-scm.com), the best version control system around. Version control allows you to experiment and make mistakes in code without messing up your final product.
* **Keep your code in one place** — Whether you work on multiple computers or just want to get some important projects off your computer, GitHub is the perfect place to store your projects online.
* **Collaboration** — Once your code is on GitHub, you can invite others to work on your code with you, share it with the world, or send a link to a friend to help you debug a problem.

### Where is your project?

Most users find it is easiest to upload a project that is already located on their local machine, so **the goal of this first step is to make a local copy of the repository.** First, let's make sure this course is going to give you the right steps:

### Is your project using version control

If you aren't sure whether or not your code is under version control, it probably isn't. However, here are a few tests you can apply to know for certain:

* Can you view a history of the changes you have made?
* Can you easily roll back to a previous version of your project?
* Are you required to provide "messages" or "commits" when you make changes?

If none of these are true, your project isn't using version control.

### ⌨️ Activity: Exporting your project

Choose the drop-down below that best fits your current situation.

### Your project is already on your local machine

✨ Terrific! [@shahnewazibnu](https://github.com/shahnewazibnu) since you already have the project locally, you are almost ready to move it to GitHub.

To confirm: You have a project directory on your computer and you want to save it on GitHub.

* **If this is correct**, close this issue to signal you are finished with this step. I will open a new issue to show you how to optimize your repository for Git operations.
* **If this is incorrect**, please use the next drop-down to learn how to export your project to your local machine or join the [Migrating your project to GitHub](https://lab.github.com/courses/migrating-your-repository-to-github) course to migrate your project to GitHub.

Your project is on a non-version controlled site, such as CodePen or Glitch

### General instructions

There are many platforms that allow users to create and store projects. We can't cover them all, but we will do our best to cover the more common examples. First, let's cover general instructions:

* Export your project using the tools available on the current site. This will usually happen via a .zip, or some other compressed format, downloaded directly to your local machine
* Save the .zip file
* Extract the .zip file

Now let's talk about specific platforms:

#### Exporting from CodePen

From the main page of your CodePen project:

1. Click the **Export** button in the bottom right corner
2. Save the exported .zip file in your local directory
3. Extract the .zip file

#### Exporting from Glitch

From your Glitch project page:

1. Click the dropdown next to your project name in the top right corner
2. Select **Advanced Options**
3. Select **Download Project**
4. Save the exported file in your local directory
5. Extract the file
6. Rename the app folder as desired

## Step 2: Prepare the project

### Working with Binary files

In general, there are two types of files: text files and binary files.

Text files, like most code files, are easily tracked with Git and are very lightweight.

However, binary files like spreadsheets, presentations with slides, and videos don't work well with Git. If your repository already has some of these files, it's best to have a plan in place before you enable Git version control.

You could choose to remove the binary files, or use another tool like [git-lfs](https://git-lfs.github.com/) (Git Large File Storage). We won't get into detail on how to set up git-lfs in this course, but we will talk about .gitignore files next, which are key to protecting your code from becoming bloated with binaries.

### Add a .gitignore

As we convert your project to a Git repository, it should only include the source code necessary to build or compile your project. In addition to avoiding binaries as we discussed above, you will also want to keep build artifacts out of your version controlled code.

To do this, you will create a file in your current project named .gitignore. Git will use the .gitignore to determine which files and directories should not be tracked under version control. The [.gitignore file](https://help.github.com/articles/ignoring-files/) is stored in your repository in order to share the ignore rules with any other users that interact with the repository.

Since the files to be ignored are dependent on the language you are using, the open source community has contributed some great templates for .gitignore files in the [github/gitignore](https://github.com/github/gitignore) repository.

### ⌨️ Activity: Prepare your repository

1. Remove any binary files from your repository.
2. In your local environment, [create a .gitignore file](https://help.github.com/articles/ignoring-files/). You can use a [template](https://github.com/github/gitignore) or create your own.

## Step 3: Make the move

Having a project already stored locally enables you to move it to GitHub rather quickly. The following activity provides instructions to move your local project to GitHub using various tools. Select the tool you are most comfortable with and get importing 😄.

### ⌨️ Activity: Moving your local project

1. In the **Code** tab of this repository, copy the URL shown under **Quick Setup**.
2. Follow the instructions below based on what tool you'd like to use locally.

Using the command line

### Using the command line

1. In your command line, navigate to your project directory. Type git init to initialize the directory as a Git repository.
2. Type git remote add origin https://github.com/shahnewazibnu/github-upload.git
3. Type git add .
4. Type git commit -m "initializing repository"
5. Type git push -u origin main to push the files you have locally to the remote on GitHub. (You may be asked to log in.)

**Note:** You can also use a password protected SSH key to connect to GitHub. See [Connecting to GitHub with SSH](https://help.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh) in our documentation to learn more.

Using GitHub Desktop ( not shown here)

Using Visual Studio Code ( not shown here)

Using Atom ( not shown here)

Using Eclipse ( not shown here)

## Step 4: Private or Public?

Right now, your repository is set to public.

You can change the visibility of a repository to Private or Public at any time in your repository's **Settings** tab, but there are some things you should know.

### Private Repositories

If your repository is private, the only people who can see your code are you and the collaborators [📖](https://help.github.com/articles/github-glossary/#collaborator) you've invited.

### Public Repositories

In public repositories, anybody can see your code. Millions of open source repositories on GitHub are public, too!

Licenses, code of conduct, and other files are important when you create a public repository. There are many benefits to this, but it's also a large responsibility. Keep in mind that once a repository is public and open source, there are certain implications about keeping it public, depending on the license chosen.