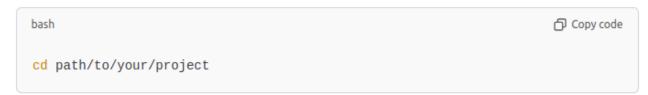
To upload a project to GitHub using Git commands, follow these steps. I'll break down each step with explanations and relevant commands.

Prerequisites

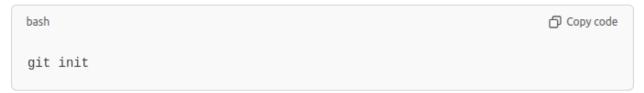
- 1. Install Git: If Git isn't installed, download and install it from git-scm.com.
- Create a GitHub repository: Log in to GitHub and create a new repository. Note the
 repository URL (it will look something like https://github.com/yourusername/repositoryname.git).

Steps

Navigate to Your Project Directory Open a terminal and go to your project folder. You can
use the cd command to do this:

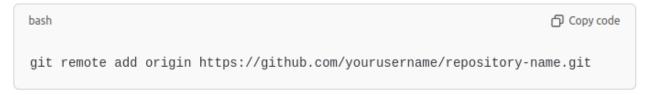


2. Initialize Git Repository If your project isn't already a Git repository, initialize it with:



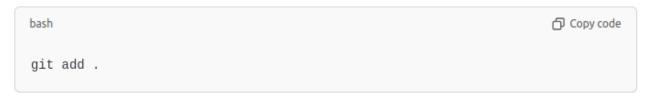
This command creates a .git directory in your project folder, setting it up as a Git repository.

 Add Remote Repository Link your local repository to the GitHub repository you created by adding a remote origin:



Replace https://github.com/yourusername/repository-name.git with the actual URL of your GitHub repository.

4. Add Files to Staging Area Add all files in your project to Git's staging area:



The . means "add everything in this directory." You can also specify individual files instead of . if you don't want to add everything at once.

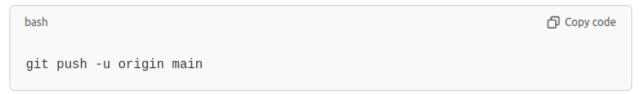
5. Commit Your Changes Make a commit, which is like a save point for your changes. Include a message to describe the commit:

```
bash

git commit -m "Initial commit"
```

Replace "Initial commit" with a message that describes the changes you made.

 Push Changes to GitHub Push the changes from your local repository to the GitHub repository:

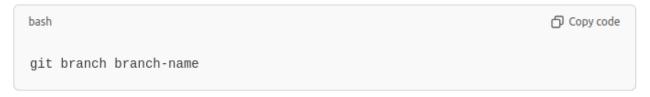


- The -u option sets origin as the default remote, so future pushes can be done with just git push.
- If your repository's main branch is named master instead of main, replace main with master.

To create a new branch in Git, switch to it, and add files specifically to that branch, follow these steps.

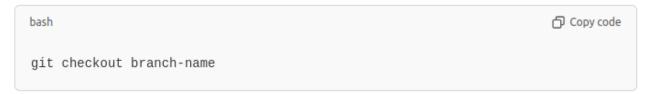
Steps

Create a New Branch To create a new branch, use:

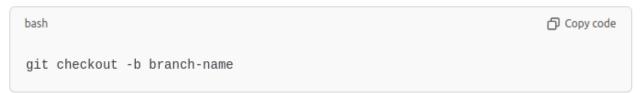


Replace branch-name with your preferred branch name (for example, feature-branch).

Switch to the New Branch After creating the branch, switch to it using:



Alternatively, you can create and switch to a new branch in a single command:



3. Add Files to the Branch

- Add new files to your project directory or make changes to existing files.
- Add these files to the staging area with:

```
bash
git add .
```

This command stages all changes in the current directory. To add only specific files, replace . with the file names.

4. Commit Changes Commit your changes to this branch:

```
bash
git commit -m "Added files to branch branch-name"
```

Replace the message with something meaningful about your changes.

5. Push the Branch to GitHub To upload this branch to GitHub, use:

```
bash
git push -u origin branch-name
```

- The -u flag sets origin as the upstream (default) for this branch, so future pushes can be done with just git push.
- Replace branch-name with the name of the branch.

Example Command List

```
git checkout -b branch-name

# Add or modify files
git add .
git commit -m "Added files to branch branch-name"
git push -u origin branch-name
```

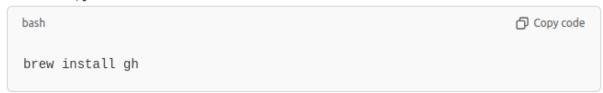
Additional Notes

- You can switch between branches using git checkout branch-name.
- Only the changes committed in a specific branch will be visible when you are checked out to that branch.

Creating a new GitHub repository directly from the command line requires using GitHub's command-line interface (CLI), known as gh. Here's how to set it up and create a repository:

Step 1: Install the GitHub CLI

- 1. Install GitHub CLI: Download and install the GitHub CLI from GitHub CLI's installation page.
 - On macOS, you can use Homebrew:



- On Linux, Windows, or other operating systems, follow the instructions on the installation page.
- 2. Authenticate the GitHub CLI: Run the following command to log in and authenticate:



Follow the prompts to log in with your GitHub account.

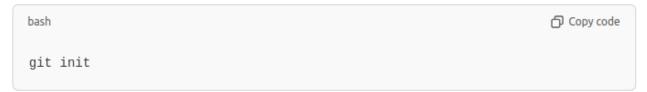
Step 2: Create a New GitHub Repository

Once you're authenticated, use the following commands to create a new repository:

1. Navigate to Your Project Directory:



2. Initialize a Git Repository (If Not Already Initialized):



3. **Create the GitHub Repository**: Run the following command to create a new repository on GitHub:

```
bash

gh repo create repository-name --public
```

Replace repository-name with your desired repository name. Use --public to make the repository public, or --private to make it private.

• If you want to create the repository with a remote link right away, you can use:

```
bash

Ghordon Copy code

ghordon reported repository-name --public --source --- remote -- remote -- remote -- remote --- remote -- remote --- r
```

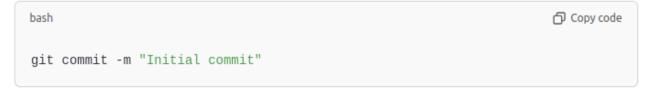
This command will automatically add the GitHub repository as a remote named origin for your project.

Step 3: Push Your Local Project to GitHub

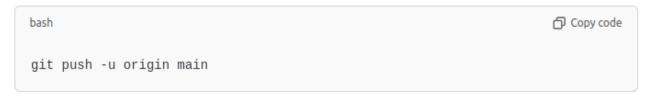
1. Add All Files to Git:



2. Commit the Changes:



3. Push the Repository to GitHub:



Replace main with master if that's the default branch in your repository.

Example Command Summary

```
cd path/to/your/project
git init
gh repo create repository-name --public --source=. --remote=origin
git add .
git commit -m "Initial commit"
git push -u origin main
```

Additional Notes

- You can also set repository details interactively by simply using gh repo create.
- · If you encounter any issues, make sure you're authenticated with gh auth login.

PAT (Personal Access Token)

A **Personal Access Token (PAT)** in GitHub is a secure way to authenticate yourself with GitHub when performing actions via the Git command line, scripts, or API. It functions as a substitute for your GitHub password when accessing GitHub repositories over HTTPS.

Why Use a Personal Access Token?

Enhanced Security:

- GitHub removed basic authentication with passwords for Git operations over HTTPS as of August 2021. PATs are now the secure alternative.
- PATs can have specific permissions, limiting access to certain actions or data, which adds a layer of security over using your password.

2. Automated or Programmatic Access:

 PATs are often used in scripts or applications that interact with GitHub repositories or the GitHub API. They are essential for any programmatic access, such as CI/CD pipelines or API-based operations.

3. Access Control and Permissions:

 You can generate tokens with only the permissions needed for specific actions, such as read-only access to repositories, write access to a specific repository, or admin-level permissions across repositories.

When Should You Use a PAT?

You should use a PAT in the following scenarios:

1. Cloning, Pulling, or Pushing Code Over HTTPS:

 When accessing private repositories over HTTPS, you'll need a PAT to authenticate rather than a password.

Authenticating GitHub CLI or API Requests:

 When using GitHub's command-line tool (gh), API requests, or third-party tools that require GitHub access, you'll need a PAT.

3. Setting Up CI/CD Pipelines:

 If your CI/CD pipeline or automated workflow requires access to a GitHub repository or needs to make API calls, you'll need a PAT to grant the necessary permissions.

4. Third-Party Integrations:

 Some third-party tools that integrate with GitHub require a PAT with specific permissions to access your repositories, issues, or workflows.

How to Generate a PAT on GitHub

- Go to Settings in your GitHub account.
- In the left sidebar, select Developer settings > Personal access tokens > Tokens (classic).
- Click Generate new token.
- Select the scopes (permissions) you want to grant, such as repo for full control over private repositories or workflow for GitHub Actions.
- Click Generate token and copy the token immediately (as you won't be able to see it again).

Using a PAT for Authentication in Git

After generating the PAT, use it in place of your password when Git prompts for authentication. For example, when pushing code:

bash
git push https://github.com/username/repository.git

Git will prompt for a username and password:

- Username: Your GitHub username
- **Password**: Your PAT (instead of your GitHub password)

Alternatively, you can configure Git to store your PAT locally using:

```
bash
git config --global credential.helper store
```

Then, when prompted for your credentials, Git will remember them for future commands.

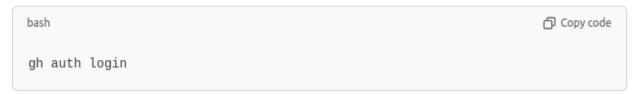
To delete a repository, branch, or file using GitHub's command line and Git commands, follow these instructions.

1. Deleting a GitHub Repository

To delete a repository directly on GitHub from the command line, you can use the **GitHub CLI** (gh).

Steps:

1. Ensure GitHub CLI is installed and authenticated:



2. **Delete the Repository**: Run the following command to delete a repository:



Replace username/repository-name with the actual repository name, like myusername/myrepo.

3. Confirm the Deletion:

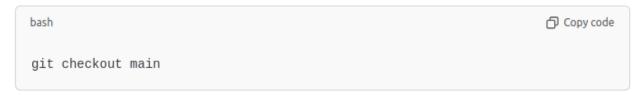
 You will be prompted to confirm whether you want to delete the repository locally, remotely, or cancel. Choose the appropriate option.

2. Deleting a Branch

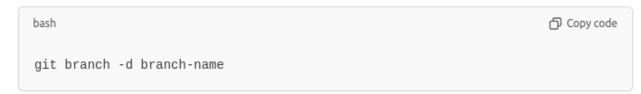
To delete a branch, you can use Git commands if it's a local branch, and gh or Git commands if it's a remote branch.

Delete a Local Branch:

1. Switch to a Different Branch (you cannot delete the branch you're currently on):



2. Delete the Branch:



Replace branch-name with the branch you want to delete. Use -D instead of -d to force-delete if there are unmerged changes.

Delete a Remote Branch:

1. Delete the Remote Branch:

```
bash
git push origin --delete branch-name
```

This deletes the branch branch-name from the remote repository.

3. Deleting a File

To delete a file from a GitHub repository using Git commands:

1. **Delete the File Locally**: Use the rm command to delete a file from your local repository:

```
bash
git rm path/to/file
```

Replace path/to/file with the path to the file you want to delete.

2. Commit the Deletion: Commit the change to register the deletion in Git:

```
bash

git commit -m "Deleted file path/to/file"
```

3. **Push the Change to GitHub**: Push the commit to update the remote repository:

```
bash
git push origin branch-name
```

Replace branch-name with the name of the branch you're pushing to (usually main or master).

Summary of Commands

```
# Delete a GitHub repository
gh repo delete username/repository-name

# Delete a local branch
git branch -d branch-name

# Delete a remote branch
git push origin --delete branch-name

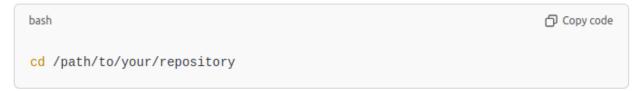
# Delete a file from the repository
git rm path/to/file
git commit -m "Deleted file path/to/file"
git push origin branch-name
```

Additional Notes

- Be cautious with these delete commands, especially with repositories and branches, as they
 may be irreversible.
- Ensure that you have backups or confirmation before deleting critical parts of your repository.

To change the Git remote URL for a repository, you can use the following commands in your terminal or command prompt:

1. Go to your Git repository directory:



2. Change the remote URL:

To update the URL of the existing remote, typically named origin:

```
bash
git remote set-url origin <new-repo-url>
```

 Replace <new-repo-url> with the new URL for your repository, such as https://github.com/username/repo.git or git@github.com:username/repo.git.

3. Verify the new URL:



This command shows the updated remote URLs to confirm the change.