**GitHub**

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere..

**Here are some important aspects of GitHub:**

1. **Repositories:**

A repository, or "repo" for short, is a collection of files and folders that make up a project. It can be either public (visible to everyone) or private (accessible only to selected collaborators).

1. **Version Control:**

GitHub utilizes Git, a distributed version control system, to manage and track changes in your codebase over time. Git allows you to create branches, commit changes, merge code, and revert to previous states.

1. **Pull Requests:**

A pull request is a way to propose changes to a repository. When working on a project, you can create a branch, make modifications, and then submit a pull request to merge your changes into the main codebase. It enables collaboration and code review among team members.

**How to create Github Account**

To create a GitHub account, you can follow these steps:

1. Open your web browser and go to the GitHub website at <github.com>.
2. On the GitHub homepage, you will see a "Sign up" button. Click on it.
3. You will be presented with two options: "Sign up for GitHub" or "Sign up with Google."

Choose the option that suits you best.

* If you choose "Sign up for GitHub," proceed to the next step.
* If you choose "Sign up with Google," you will be redirected to a Google sign-in page. Sign in with your Google account, and then follow the instructions in the next steps.

1. On the "Create your account" page, provide the following information:

* **Username:** Choose a unique username for your GitHub account. This will be part of your GitHub profile URL.
* **Email address:** Enter a valid email address that you want to associate with your GitHub account.
* **Password:** Choose a strong and secure password.

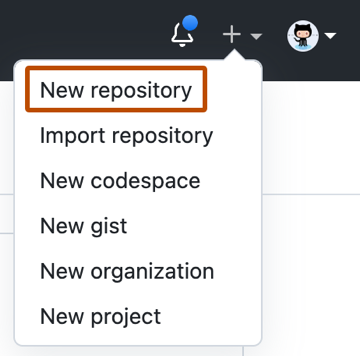
1. After filling in the required information, click on the "Verify account" button.
2. GitHub may prompt you to complete a verification process to ensure that you're not a robot. Follow the instructions on the screen to complete the verification.
3. Once your account is verified, you will be asked to choose your personal plan. GitHub offers free and paid plans. Select the plan that suits your needs and click on the "Continue" button.
4. Optionally, GitHub may ask you to provide additional information, such as your areas of interest or experience level. You can choose to fill in this information or skip it by clicking on the "Skip this step" link at the bottom.
5. Congratulations! You have successfully created your GitHub account. You will be directed to your GitHub profile page.

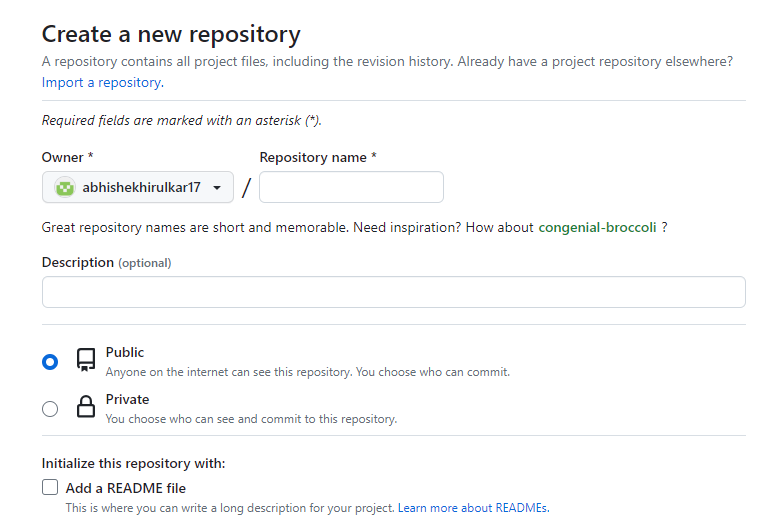
## [Creating a repository](https://docs.github.com/en/get-started/quickstart/hello-world" \l "creating-a-repository)

A repository is usually used to organize a single project. Repositories can contain folders and files, images, videos, spreadsheets, and data sets -- anything your project needs. Often, repositories include a README file, a file with information about your project. README files are written in the plain text Markdown language.

Your hello-world repository can be a place where you store ideas, resources, or even share and discuss things with others.

1. In the upper-right corner of any page, use the  drop-down menu, and select **New repository**.





* In the "Repository name" box, type Repository Name.
* In the "Description" box, type a short description.
* Select whether your repository will be **Public** or **Private**.
* Select **Add a README file**.
* Click **Create repository**.

**Flow of how you can use GitHub for collaborative software development:**

* **Create a Repository:** Start by creating a new repository on GitHub. This will serve as a central location to store your project's code, track changes, and collaborate with others. You can create a repository by clicking the "New" button on the GitHub website and providing a name, description, and other optional details.
* **Clone the Repository:** Clone the repository to your local machine using the git clone command. This creates a local copy of the repository on your computer, allowing you to make changes and work offline.
* **Create a Branch:** Create a new branch for your changes using the git branch command. Branches allow you to isolate your work from the main codebase. Use a descriptive branch name that reflects the feature or bug fix you're working on.
* **Switch to the Branch:** Switch to the branch you just created using the git checkout command. This ensures that any changes you make will be applied to the correct branch.
* **Make Changes:**  Start making changes to the files in your local repository using your preferred text editor or IDE. Add, modify, or delete files as necessary to implement the desired functionality or fix a bug.
* **Stage and Commit Changes:**  Use the git add command to stage the changes you made to your files. This prepares them for a commit. Then, use the git commit command to create a commit with a descriptive commit message explaining the changes you made.
* **Push Changes to GitHub:**  Push your committed changes to the GitHub repository using the git push command. This uploads your local commits to the remote repository on GitHub.
* **Open a Pull Request:**  If you want to merge your changes into the main codebase, open a pull request on GitHub. Go to your repository's page on GitHub, click on the "Pull requests" tab, and create a new pull request. Provide a title and description that clearly explain the purpose of your changes.
* **Review and Discuss:**  Collaborators or project maintainers can review your changes, leave comments, and suggest modifications directly on the pull request. Engage in discussions to address any questions or concerns.
* **Make Changes and Update the Pull Request:**  If feedback or changes are requested, make the necessary modifications locally in your branch. Stage and commit the changes as before, and push them to the remote branch associated with the pull request. The pull request will be automatically updated with your latest changes.
* **Merge the Pull Request:** Once the pull request has been reviewed and approved, you can merge it into the main codebase. Click the "Merge" button on the pull request page to incorporate your changes.
* **Pull Changes:**  If you are working with a team and others have made changes to the repository, you can pull those changes to your local repository using the git pull command. This updates your local repository with the latest changes from the remote repository.

**Git Setup**

**Installation Link of Git Bash Command Line For Windows**

:- <git-for-windows.github.io>

1. **Open Git Bash:**

Once the installation is complete, open Git Bash by searching for "Git Bash" in your applications or by right-clicking in any folder and selecting "Git Bash Here" to open a Git Bash terminal in that location.

**To check the git Version us**e :

:- $ git version

1. **Configure Git:**

Before you start using Git Bash, it's recommended to configure your Git username and email address. Open the Git Bash terminal and run the following commands, replacing Your Name with your desired username and your.email@example.com with your email address:

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

:- git config --global user.email your.email@example.com

These configurations will be used when you make commits in Git.

1. **Navigate to a Directory:**

Use the cd command to navigate to the directory where you want to work with Git repositories. For example, to navigate to the "Documents" directory, you would use:

:- cd Documents

You can use commands like ls (list files) and pwd (print working directory) to explore the current directory and see its contents.

1. **Clone a Repository:**

To clone an existing Git repository from a remote server, use the git clone command followed by the repository URL. For example, to clone a repository named "my-repo" hosted on GitHub, you would use:

:- git clone <https://github.com/your-username/my-repo.git>

This command creates a local copy of the repository in the current directory.

1. **Initialize a New Repository:**

If you want to create a new Git repository, navigate to the desired directory using the cd command and run the following command:

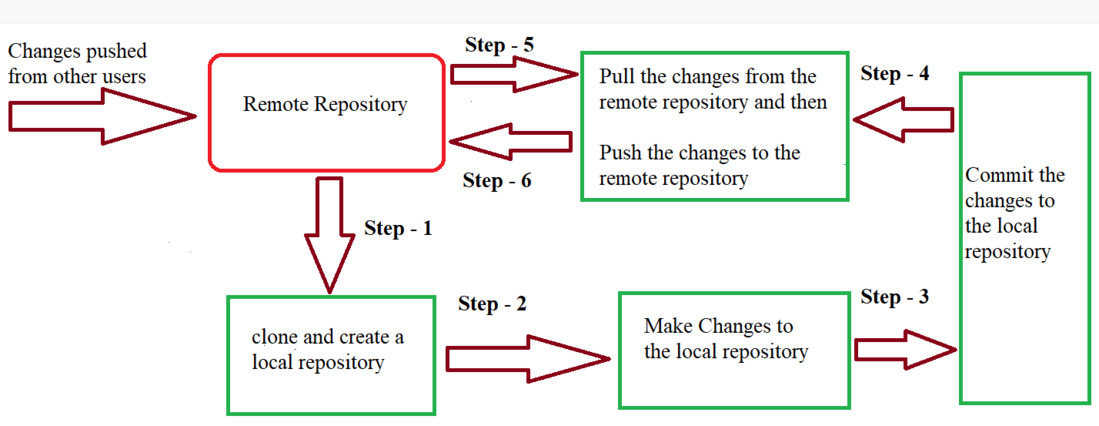
:- git init

This initializes a new Git repository in the current directory.

1. **Start Using Git:**

With Git Bash set up, you can now start using Git commands to manage your repositories. Some common Git commands include **git add**, **git commit**, **git push**, **git pull**, and **git branch,** among others. You can refer to the Git documentation or use the **git --help** command for more information on each command.

# Git – Life Cycle



* In Step – 1, We first clone any of the code residing in the remote repository to make our own local repository.
* In Step-2 we edit the files that we have cloned in our local repository and make the necessary changes in it.
* In Step-3 we commit our changes by first adding them to our staging area and committing them with a commit message.
* In Step – 4 and Step-5 we first check whether there are any of the changes done in the remote repository by some other users and we first pull that changes.
* If there are no changes we directly proceed with Step – 6 in which we push our changes to the remote repository and we are done with our work.

**To use Git Bash with GitHub, you can follow these steps:**

1. **Create a Repository on GitHub:**

Once you have a GitHub account, you can create a new repository to host your code. On the GitHub website, click on the "New" button to create a new repository. Provide a name, description, and other optional details for your repository, and then click "Create repository".

1. **Clone the Repository:**

In Git Bash, navigate to the directory where you want to clone the repository using the cd command. For example, if you want to clone the repository into your "Documents" directory, you would run:

:- cd Documents

To clone the repository, go to the repository page on GitHub and click on the "Code" button. Copy the repository URL (HTTPS or SSH) and then run the following command in Git Bash:

:- git clone <repository\_url>

Replace **<repository\_url>** with the URL you copied.

1. **Configure Remote:**

After cloning the repository, navigate into its directory using cd <repository\_name>. To connect the local repository with your GitHub repository, run the following command:

:- git remote add origin <repository\_url>

1. **Make Changes and Commit:**

Start making changes to the files in your local repository using your preferred text editor or IDE. Once you've made changes, you need to stage them for commit. Use the following command to stage all changes:

:- git add .

You can also specify individual files instead of . to stage specific changes.

1. **Commit Changes:**

After staging the changes, you can create a commit with a message describing the changes. Use the following command:

:- git commit -m "Your commit message"

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

1. **Push Changes to GitHub:**

To push your committed changes to the GitHub repository, use the following command:

:- git push origin master

This pushes the changes to the remote repository on GitHub.

1. **Pull Changes from GitHub:**

If you are collaborating with others and they have made changes to the GitHub repository, you can pull those changes to your local repository using the following command:

:- git pull origin master

This updates your local repository with the latest changes from the remote repository.

**Git commands that you can use with GitHub:**

1. **git init:** Initializes a new Git repository in the current directory.
2. **git clone <repository\_url>:** Clones a remote Git repository to your local machine. Replace <repository\_url> with the URL of the repository you want to clone.
3. **git add <file>:** Adds a file or changes to the staging area, preparing it for a commit. Replace <file> with the name of the file you want to add. Use "." to add all changes.
4. **git commit -m "Commit message":** Creates a new commit with the changes from the staging area. Replace "Commit message" with a descriptive message that explains the changes made in the commit.
5. **git push:** Pushes committed changes from your local repository to a remote repository, typically on GitHub. This command is used to upload your local commits to the remote repository.
6. **git pull:** Fetches and merges changes from a remote repository into your local repository. It updates your local repository with the latest changes from the remote repository.
7. **git status:** Shows the current status of your repository, including the files that have been modified, added, or deleted. It provides an overview of the changes made and the state of your repository.
8. **git log:** Displays the commit history of your repository, showing the author, date, and commit messages of each commit.
9. **git branch:** Lists all the branches in your repository. The current branch is indicated with an asterisk.
10. **git checkout <branch>:** Switches to the specified branch. Replace <branch> with the name of the branch you want to switch to.
11. **git merge <branch>:** Merges changes from the specified branch into the current branch. Replace <branch> with the name of the branch you want to merge into the current branch.
12. **git remote add origin <repository\_url>:** Associates your local repository with a remote repository. Replace <repository\_url> with the URL of the remote repository.