



IBM

GitHub

Training Material

TABLE OF CONTENTS

CHAPTER 1: WHAT IS GITHUB?	3
VERSION CONTROL.....	3
REPOSITORIES.....	3
COLLABORATION TOOLS	3
BRANCHING AND MERGING.....	3
CONTINUOUS INTEGRATION & CONTINUOUS DEPLOYMENT	4
CODE REVIEW AND FEEDBACK.....	4
DOCUMENTATION	4
INTEGRATION WITH OTHER TOOLS	4
CHAPTER 2: HOW GITHUB WORKS.....	4
CREATING A REPOSITORY	4
CLONING THE REPOSITORY	5
MAKING CHANGES	5
PUSHING CHANGES	5
CREATING A BRANCH	5
CREATING A PULL REQUEST.....	5
REVIEW AND DISCUSS.....	5
MERGING	5
RESOLVING ISSUES	5
CONTINUOUS INTEGRATION AND DEPLOYMENT	5
CHAPTER 3: THE NEED FOR GITHUB	6
CHAPTER 4: WHY GITHUB SO POPULAR.....	7
CHAPTER 5: GITHUB SIGNIN PAGE.....	7
CREATE A REPOSITORY	10
CREATE A BRANCH.....	12
MAKE A COMMIT	14
OPENING A PULL REQUEST.....	15
MERGING A PULL REQUEST.....	16

Chapter 1: What is GitHub?

GitHub is a prevalent web-based stage that gives adaptation control utilizing Git, a conveyed form control framework made by Linus Torvalds. GitHub encourages collaboration on program advancement ventures by facilitating stores, following changes, and giving a extend of apparatuses to oversee code and collaborate with other engineers. Here's a closer see at what GitHub offers and how it works:

Version Control

- **Git Integration:** GitHub employments Git to oversee changes to source code over time. It permits different designers to work on the same extend at the same time without interferometer with each other's work.
- **Commit History:** Clients can track and audit the history of changes made to the code base, counting who made changes and why.

Repositories

- **Public and Private Repositories:** Users can make open storehouses (unmistakable to everybody) or private stores (limited get to). Open storehouses are frequently utilized for open-source ventures, whereas private ones are utilized for exclusive or touchy work.
- **Forking and Cloning:** Users have the ability to clone (download) repositories to their local machines for development purposes and fork (make a copy of) repositories to work on their own version.

Collaboration Tools

- **Pull Requests:** Developers propose changes to a project via pull requests. These changes can be reviewed and discussed before being merged into the main codebase.
- **Issues:** To keep track of work, feature requests, and bugs, users can create issues. Within the repository, issues can be assigned, tagged, and discussed.
- **Dividing and Combining:** Using GitHub, individuals can establish branches to separately work on different features or repairs. This makes it easier to manage various development streams without interfering with the primary codebase.

Branching and Merging

- **Branches:** GitHub allows users to create branches to work on different features or fixes independently. This helps in managing different lines of development without affecting the main codebase.

- **Merging:** After a branch's development is finished, it can be merged back into the main branch, often known as main or master, to incorporate the modifications into the main source code.

Continuous Integration and Continuous Deployment (CI/CD)

- **GitHub Actions:** GitHub Actions are integrated continuous integration and delivery technologies. With these, users can respond to events like pull requests or code contributions by automating work routines, including launching apps or performing tests.

Code Review and Feedback

- **Code Review:** Pull requests help with code review by enabling team members to offer suggestions for enhancements and to remark on individual lines of code prior to changes being merged.
- **Collaborative Feedback:** GitHub offers a central location for teams to discuss code, review changes, and provide comments.

Documentation

- **README Files:** Each repository can include a README file that provides information about the project, such as how to install, use, and contribute to it.
- **Wiki:** GitHub supports a wiki for more extensive documentation and project information.

Integration with Other Tools

- **Third-Party Integrations:** GitHub can be integrated with other external tools and services, such as deployment platforms, testing frameworks, and project management systems.

Chapter 2: How GitHub Works

Users of GitHub can build coding projects, upload files, and create accounts. However, when users start collaborating, that's when GitHub really gets to work.

Creating a Repository

- **Setup:** On GitHub, create a new repository that will act as the hub of your project.

Cloning the Repository

- **Local Development:** Use Git to clone the repository to your local computer. This makes a copy of the code and lets you operate offline.

Making Changes

- **Edit Code:** Edit the documents within your local repository. Stage the changes and commit them to your local branch when you're ready.
- **Commit Changes:** Utilize Git to generate commits that include your modifications and informative notes.

Pushing Changes

- **Update Remote:** Your modifications are reflected in the remote repository when you push your contributions to GitHub.

Creating a Branch

- **Feature Development:** Make a new branch dedicated to working on a particular feature or fix. This contributes to the main branch's stability.

Creating a Pull Request

- **Merge Proposal:** When your modifications are complete, create a pull request on GitHub. This indicates that your branch should be merged into another branch, usually the main branch.

Review and Discuss

- **Peer Review:** Members of the team examine the pull request, offer comments, and talk about any adjustments that should be made. Based on comments, you can add further commits to the pull request.

Merging

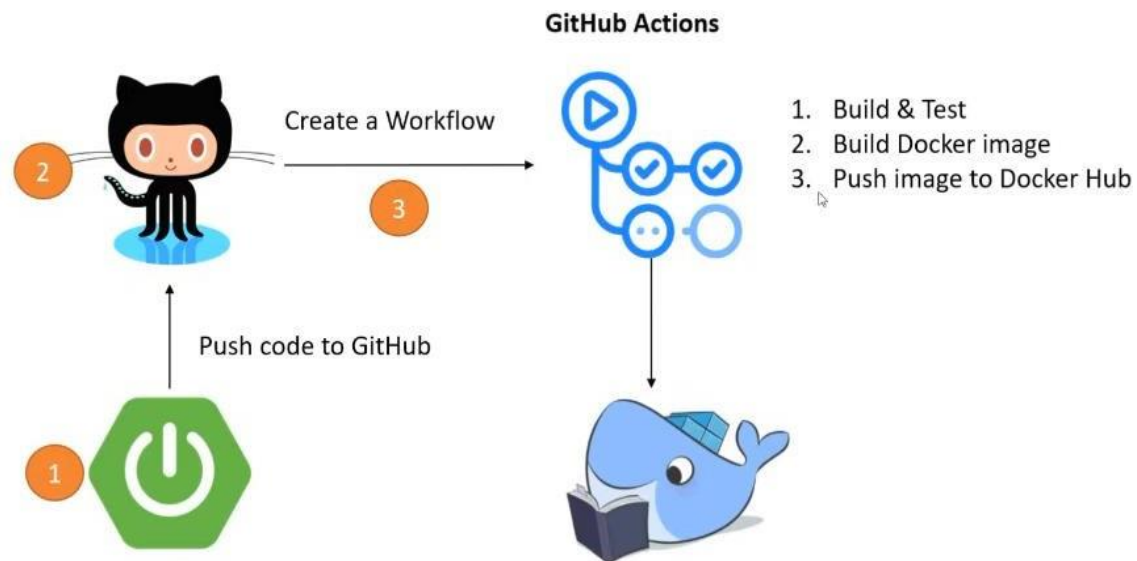
- **Integration:** The pull request can be merged into the target branch after it has been approved. Your modifications are now incorporated into the main code base.

Resolving Issues

- **Track Progress:** Track tasks, features, or bugs using issues. To give context for how they solve particular issues or requests, you can link issues to pull requests.

Continuous Integration and Deployment

- **Automation:** When configured, GitHub Actions responds to events like commits or pull requests by launching automated work routines, such as tests. This makes deployment easier and helps guarantee code quality.



Chapter 3: The Need for GitHub

GitHub addresses several key needs and challenges in modern software development and project management. Here's a detailed look at why GitHub is essential for many teams and projects:

1. **Version Control**
2. Collaboration
3. Code Review
4. Documentation and Issue Tracking
5. Continuous Integration and Deployment (CI/CD)
6. Collaboration with Open Source Community
7. Project Management
8. Security and Access Control
9. Integration with Other Tools
10. Visibility and Communication

With its ability to handle requirements for version control, collaboration, code review, project management, and automation, GitHub is an essential tool for modern software development. It is an essential platform for both lone workers and big teams working on a variety of projects because it improves productivity, quality, and transparency in the development process.

Chapter 4: Why GitHub so popular?

The reason GitHub is so popular is because of its vast integration possibilities, community support, powerful collaboration tools, and user-friendly design. Both small and large teams use it because it provides an all-in-one platform for version control, project management, and automation. Its widespread adoption is further facilitated by its function as the focal point for open-source projects, as well as by its ongoing development and the backing of significant IT firms.

1. **Ease of Use**
2. **Collaboration Features**
3. **Community and Open Source**
4. **Integration and Extensible**
5. **Automation and CI/CD**
6. **Documentation and Project Management**
7. **Security and Compliance**
8. **Visibility and Transparency**
9. **Support for Diverse Work flows**
10. **Brand and Ecosystem**

Chapter 5: GitHub SignIn page

Open GitHub's Website

Go to <https://github.com/login>

Access the Sign-In Page

Click on the "**Sign in**" button located in the top-right corner of the page.

Enter Your Credentials

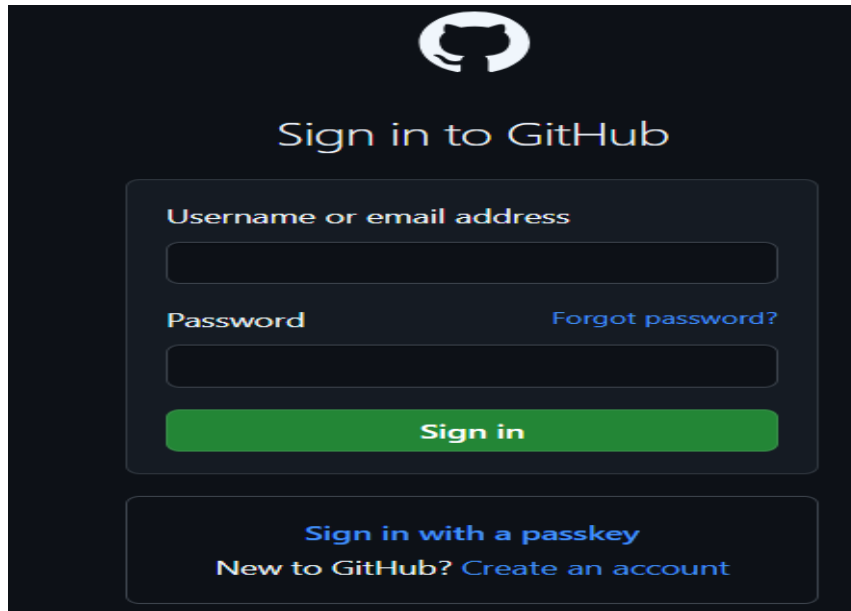
Username or Email Address: Enter your GitHub username or the email address associated with your GitHub account.

Password: Enter your GitHub password.

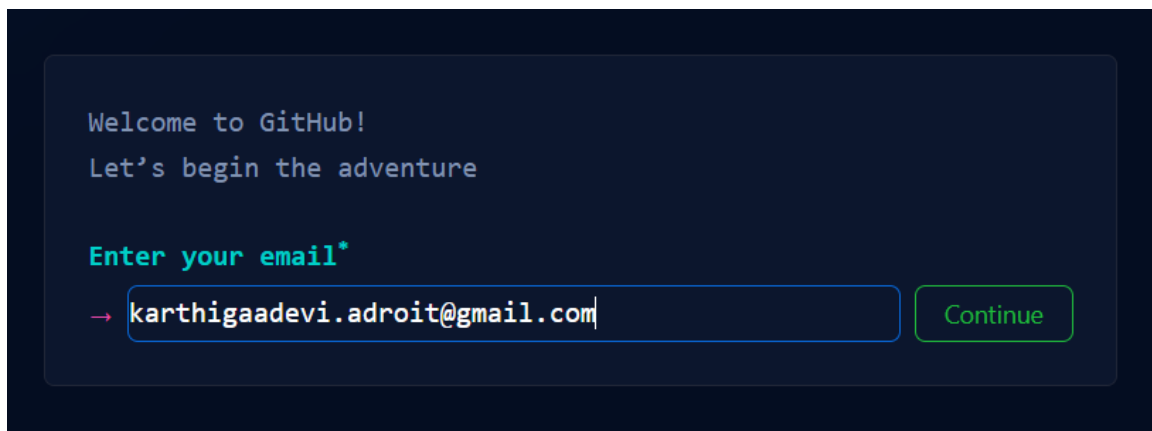
- If you have enabled two-factor authentication (2FA), you will also need to enter the authentication code generated by your 2FA method (such as an authenticator app or a code sent to your phone).

Click "Sign in"

Once you've entered your credentials, click the **"Sign in"** button to access your GitHub account.



Click "Create an account"



Create a Password and Click continue

Enter email id and click "Continue"

Welcome to GitHub!
Let's begin the adventure

Enter your email*

✓ karthigaadevi.adroit@gmail.com

Create a password*

→

Password may be compromised
Password is in a list of passwords commonly used on other websites

Enter a "Username" and click Continue

Welcome to GitHub!
Let's begin the adventure

Enter your email*

✓ karthigaadevi.adroit@gmail.com

Create a password*

✓

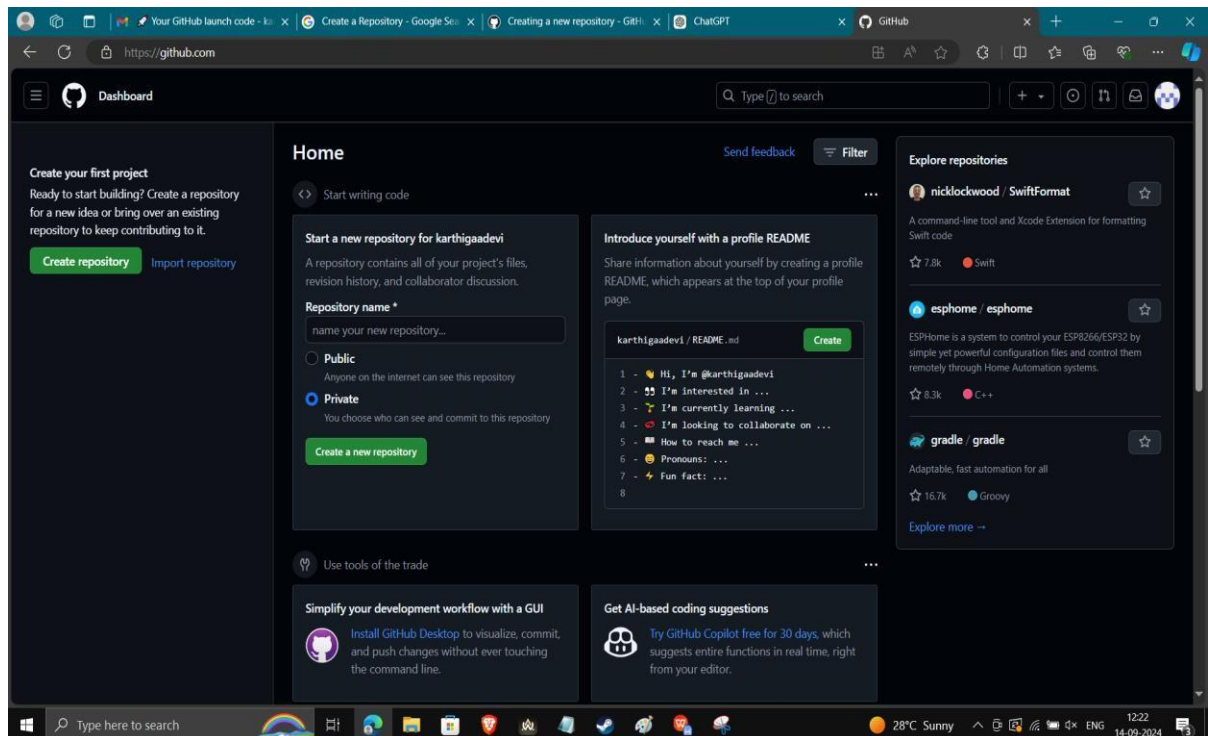
Enter a username*

→

karthigaadevi is available.

Once the account was created, have to verify the email id by giving the launch code. Then Sign In to the GitHub page.

Create a Repository



1. Log In to GitHub

- Go to [GitHub's homepage](https://github.com) and click the "Sign in" button in the top-right corner.
- Enter your credentials and log in to your GitHub account.

2. Access the New Repository Page

- Once logged in, click the "+" icon in the upper-right corner of the page.
- Select "New repository" from the drop down menu.

3. Fill Out Repository Details

- **Repository Name:** Enter a name for your repository. The name should be descriptive and relevant to the project.
- **Description (Optional):** Provide a short description of what the repository is for. This helps others understand the purpose of the repository.
- **Visibility:** Choose the visibility of your repository:

Public: Anyone can view and contribute to this repository.

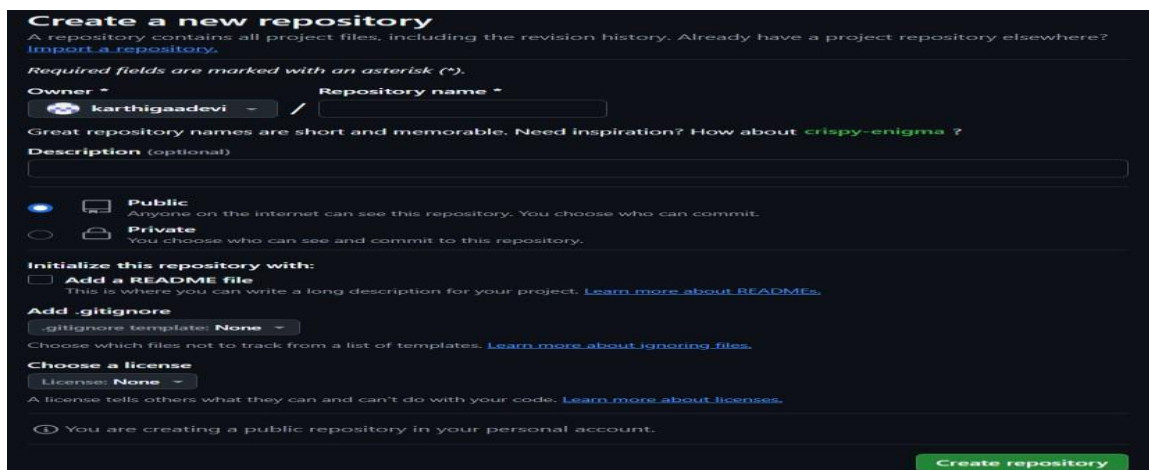
Private: Only you and people you invite can view and contribute to this repository.

Initialize This Repository with (Optional):

- **Add a README file:** Check this box if you want to include a README file in your repository. A README is useful for providing information about the project.
- **Add .gitignore:** Choose a template from the dropdown menu if you want to automatically create a .gitignore file for your project. This file specifies which files or directories to ignore in your Git repository.
- **Choose a license:** Select a license for your repository if you want to specify terms for how others can use, modify, and distribute your project.

4. Create the Repository

- Click the **"Create repository"** button to finalize the creation of your new repository.

The screenshot shows the GitHub 'Create a new repository' page. At the top, it says 'Create a new repository' and 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. Below this, it states 'Required fields are marked with an asterisk (*)'. The 'Owner' dropdown is set to 'karthigaadevi'. The 'Repository name' field is empty. A note says 'Great repository names are short and memorable. Need inspiration? How about [crispy-enigma](#) ?'. The 'Description (optional)' field is empty. Under 'Visibility', 'Public' is selected with the description 'Anyone on the internet can see this repository. You choose who can commit.' and 'Private' is unselected with 'You choose who can see and commit to this repository.'. The 'Initialize this repository with:' section has 'Add a README file' unselected, 'Add .gitignore' selected, and 'Choose a license' unselected. The '.gitignore template' dropdown is set to 'None'. A note at the bottom says 'You are creating a public repository in your personal account.' and a green 'Create repository' button is at the bottom right.

Clone the Repository

The following command in your terminal or command prompt:

“git clone <https://github.com/your-username/your-repository-name.git>”

Add Files and Commit

- Add files to your local repository, stage them, and commit changes using Git commands

“cd your-repository-name

git add

```
git commit -m "Initial commit"
```

```
git push origin main"
```

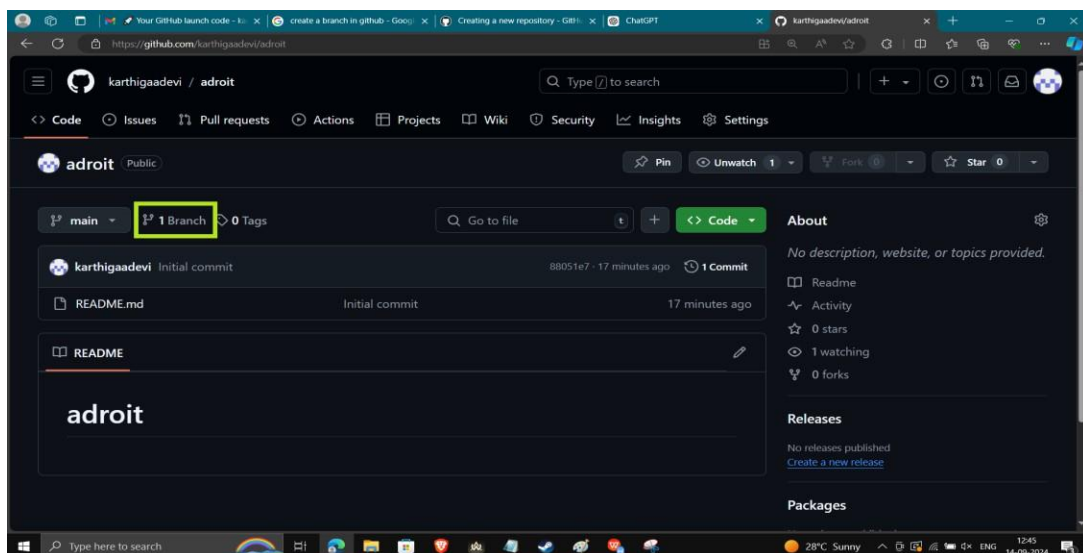
Manage Repository

- Managing the repository via the GitHub website, including creating issues, managing pull requests, and collaborating with others.

Create a Branch

In the Branch Selector

- Find the drop-down menu for the branch selector on the repository page. This is often located in the upper left corner of the repository and is frequently marked with the name of the active branch (main, master, etc.).

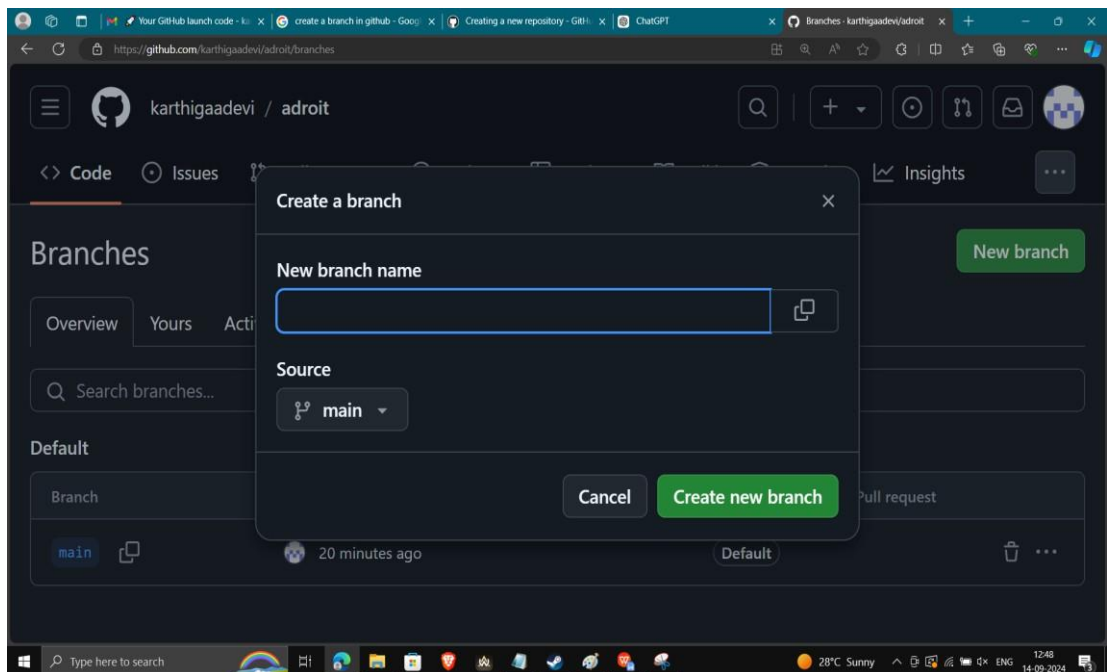
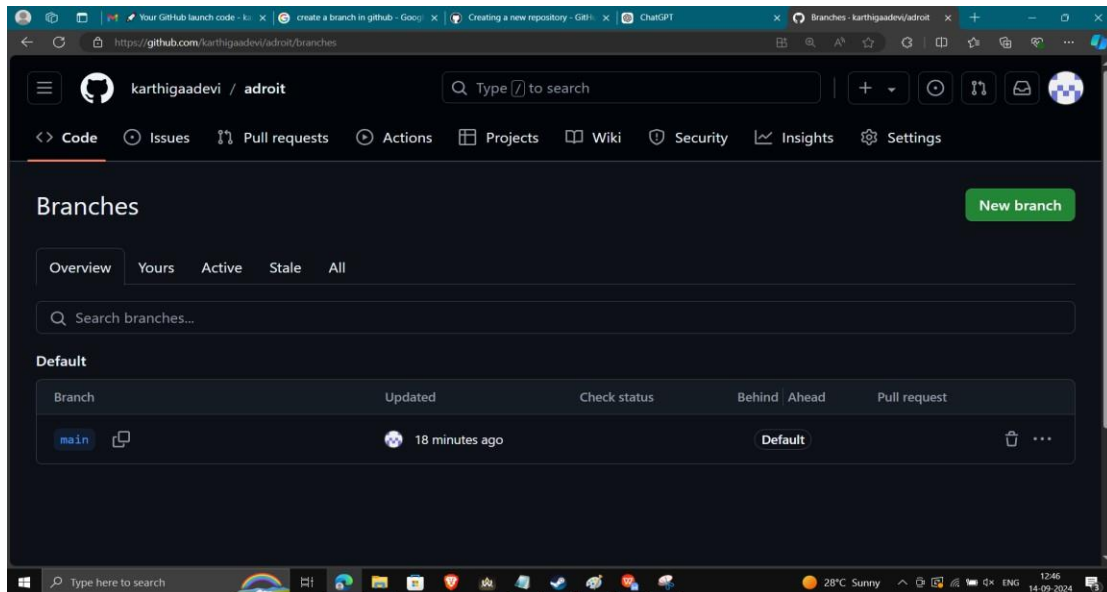


Create a New Branch

Press the drop-down menu for the branch selector. An existing branch list will be displayed to you.

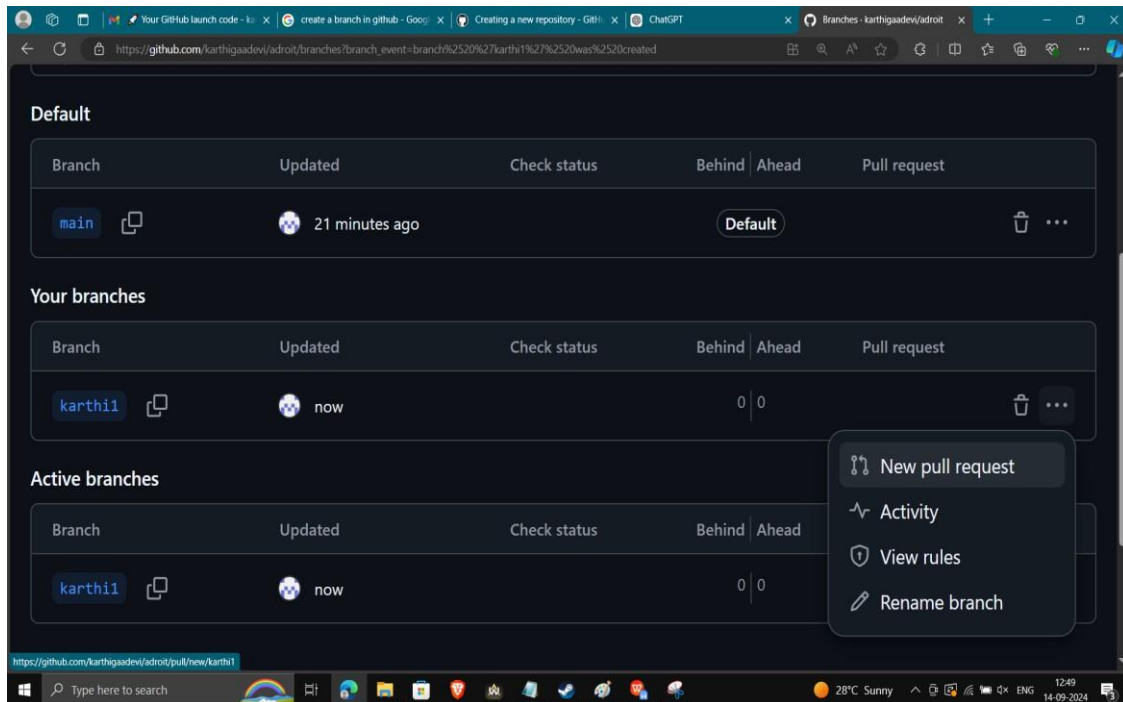
In the "Find or create a branch" input field, type the name of the new branch you wish to create.

Tap the "Create branch: [branch-name]" option that slides into view beneath the text box. By doing this, the new branch will be created and switched to automatically.



Confirm Branch Creation

- Once created, you will be switched to the new branch. You can now make changes and commit them to this branch.



Make a Commit

Making a commit in Git involves recording changes to your local repository using both the Git command line and GitHub's web interface.

Using command line:

1. Navigate to Your Repository: "cd path/to/your-repository"
2. Check the Status: "git status"
3. Stage Your Changes:
To stage specific files for commit, "git add filename"
To stage all changes, "git add ."
4. Make the Commit: "git commit -m "Your commit message describing the changes""
5. Push the Commit to Remote Repository:
To push your commit to the remote repository (e.g., GitHub), "git push origin branch-name"
Replace branch-name with the name of the branch you're working on (e.g., main or feature-branch).

Using GitHub's web interface:

Edit a File

- Navigate to the file you want to edit and click on it.
- Click the "**Edit**" button (pencil icon) to open the file in edit mode.

Make Changes

- Edit the file as needed directly in the browser.

Commit Changes

- Scroll down to the "**Commit changes**" section.
- Enter a **commit message** describing the changes you've made.
- Optionally, add a description in the "**Extended description**" field.
- Choose whether to commit directly to the main branch or to a new branch (you can create a pull request later if you choose the latter).
- Click "**Commit changes**" to save your changes to the repository.

Open & Merge Pull request

Opening and merging pull requests are essential tasks in collaborative software development. Pull requests (PRs) are used to propose changes to a codebase and request that those changes be reviewed and merged into another branch, typically the main branch.

Opening a Pull Request

Navigate to Your Repository

- Go to GitHub and sign in if you aren't already.
- Navigate to the repository where you want to open a pull request.

Go to the Pull Requests Tab

- Click on the "Pull requests" tab near the top of the repository page.

Start a New Pull Request

- Click the "New pull request" button.

Select Branches for Comparison

- Base Branch: This is the branch you want to merge changes into (usually main or master).
- Compare Branch: This is the branch with your changes (the feature branch or bugfix branch).
- GitHub will compare these branches and display the differences.

Review Changes

- Review the changes that will be merged. You can see the files changed and the diffs between the branches.

Create the Pull Request

- Click the "Create pull request" button.
- Title: Provide a clear and descriptive title for the pull request.
- Description: Add a description explaining the changes made, the reason for the pull request, and any other relevant information.
- Assignees: Optionally, assign reviewers or team members who should review the pull request.
- Labels: Optionally, add labels to categorize the pull request (e.g., bug, feature, enhancement).

Submit the Pull Request

- Click the "Create pull request" button to submit your pull request for review.

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base: master ... compare: m1 ✓ Able to merge. These branches can be automatically merged.

Milestone m1 changes

Write Preview AA B i

Looking to merge #1 and #2 as required for [m1](#)

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported

Create pull request

2 commits 3 files changed 0 commit comments 1 contributor

Commits on Jun 21, 2016

jane-ss removed "Ordered by" info in Purchase order header. ... 6487d89

jane-ss Added product id to item detail ... 9cef2fa

Showing 3 changed files with 12 additions and 4 deletions.

Unified Split

Merging a Pull Request

Navigate to the Pull Requests Tab

- Go to the "Pull requests" tab in your repository.

Select the Pull Request

- Click on the pull request you want to merge. This will open the pull request details.

Review the Pull Request

- Ensure that the changes have been reviewed and approved by the necessary reviewers.
- Review the conversation, code changes, and any associated issues or

comments.

Merge the Pull Request

- Click the "Merge pull request" button.
- Confirm the Merge: A dialog will appear asking you to confirm the merge. You can edit the commit message if necessary.
- Click "Confirm merge" to complete the process.
- Delete the Branch (Optional)
- After merging, you might see an option to "Delete branch". This will remove the branch used for the pull request if it is no longer needed, keeping your repository clean.

