Crash Course on GitHub

A Complete Introduction for Students and Developers

---

1. Introduction

GitHub is the world's largest platform for version control and collaborative software development. It is built around **Git**, a distributed version control system created by Linus Torvalds (the founder of Linux). GitHub adds a friendly web interface, collaboration tools, issue tracking, and integrations with CI/CD, cloud services, and AI assistants like **GitHub Copilot**.

GitHub allows individuals and teams to:

- Host and manage code repositories online.
- Track and merge changes across contributors.
- Collaborate through pull requests and issues.
- Automate testing, deployment, and documentation.
- Showcase projects, portfolios, and open-source contributions.

Whether you are a student, a solo developer, or part of a large enterprise team, GitHub is an essential tool in the modern software development workflow.

---

2. Understanding Git vs GitHub

Concept	Git	GitHub
Type	Version control system	Cloud-based platform for hosting Git repositories
Scope	Runs locally on your computer	Runs online and enables collaboration
Function	Tracks changes in files	Adds UI, collaboration, issue tracking, and automation
Usage	Used via command line	Used via web interface, desktop app, or IDE integration

Think of **Git** as the engine that powers version control and **GitHub** as the platform that connects developers and projects through that engine.

---

3. Key Concepts

3.1 Repository (Repo)

A **repository** is a container for your project. It holds files, folders, and version history. Repositories can be **public** (visible to everyone) or **private** (restricted access).

3.2 Commit

A **commit** is a snapshot of your project at a specific point in time. Each commit includes a message describing the change.

3.3 Branch

A **branch** allows you to develop features or fixes in isolation without affecting the main codebase (usually called `main` or `master`).

3.4 Merge

Combines changes from one branch into another, usually after review.

3.5 Pull Request (PR)

A **pull request** proposes merging code from one branch or fork into another. It enables collaboration, code review, and discussion before integration.

3.6 Fork

A **fork** is a copy of a repository that allows you to experiment or contribute to someone else's project without affecting the original.

3.7 Clone

A **clone** is a local copy of a remote repository that allows you to work on your machine.

3.8 Remote and Origin

A **remote** refers to a version of your repository hosted online. By default, `origin` is the name Git assigns to the main remote repository.

---

4. Setting Up GitHub

4.1 Create a GitHub Account

1. Visit https://github.com
2. Sign up with your email or use your school email to qualify for **GitHub Student Developer Pack**.
3. Verify your email address.

4.2 Install Git

If you don't have Git installed:

- Windows: https://git-scm.com/download/win
- macOS/Linux: Install via terminal (`sudo apt install git` or `brew install git`)

Verify installation:

```
git --version
```

4.3 Configure Git

Set your name and email for commit tracking:

```
git config --global user.name "Your Name"
git config --global user.email "you@example.com"
```

---

5. Creating Your First Repository

Option 1: Create on GitHub Web

1. Click **New Repository**.
2. Give it a name (e.g., `my-first-project`).
3. Choose **Public** or **Private**.
4. Initialize with a **README** file (recommended).
5. Click **Create Repository**.

Option 2: Create Locally

```
mkdir my-first-project
cd my-first-project
git init
```

Then add a remote:

```
git remote add origin https://github.com/username/my-first-project.git
```

---

6. Basic Git Workflow

Step 1: Add Files

```
git add .
```

Step 2: Commit Changes

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

Step 3: Push to GitHub

```
git push -u origin main
```

Now your code is stored in the remote GitHub repository.

---

7. Cloning a Repository

To copy a repository from GitHub to your computer:

```
git clone https://github.com/username/repo-name.git
cd repo-name
```

---

8. Working with Branches

Create a new branch:

```
git checkout -b feature-branch
```

Switch between branches:

```
git checkout main
```

Merge a branch:

```
git merge feature-branch
```

Delete a branch:

```
git branch -d feature-branch
```

---

9. Collaboration Workflow

1. **Fork** a repository from someone else's project.
2. **Clone** it to your local machine.
3. Create a **new branch** for your changes.
4. Make edits, add commits.
5. **Push** the branch to your fork.
6. Open a **Pull Request** on GitHub to merge your changes into the original repository.

This is the standard open-source contribution model used worldwide.

---

10. Issues and Discussions

- **Issues** are GitHub’s built-in bug tracker. You can report bugs, request features, or discuss tasks.
- **Discussions** allow for longer conversations, ideas, and design feedback within a community.
- You can assign issues, add labels, and link pull requests to keep projects organized.

---

11. README and Documentation

Every repository should include a `README.md` file in Markdown format. It serves as the front page of your project and typically includes:

- Project description
- Installation instructions
- Usage examples
- Contributors and licensing information

Example:

~~~
# My First Project
This is a simple example project.

## Installation
```bash
pip install -r requirements.txt
~~~

Usage

```
python main.py
```

```


---

## 12. GitHub Pages

GitHub Pages allows you to host static websites directly from your repositories—
for portfolios, documentation, or projects.

### Steps:
1. Create a repository named `<username>.github.io`.
2. Add HTML/CSS/JS files.
3. Push to the `main` branch.
4. Visit `https://<username>.github.io`.

```

~~~
Your website will be live within minutes.

---

## 13. GitHub Actions

**GitHub Actions** automate workflows such as:
- Running tests automatically.
- Building and deploying applications.
- Sending notifications or performing code analysis.

Example workflow file (`.github/workflows/test.yml`):

```yaml
name: Run Tests
on: [push]
jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: "3.11"
      - name: Install dependencies
        run: pip install -r requirements.txt
      - name: Run tests
        run: pytest
~~~

This runs your test suite automatically whenever you push changes.

---

14. GitHub Desktop and VS Code Integration

14.1 GitHub Desktop

A graphical application that simplifies Git commands:

- Clone repositories
- Commit changes
- Create branches
- Push and pull updates

Download: https://desktop.github.com/

14.2 GitHub Extension for VS Code

The **GitHub Pull Requests and Issues** extension integrates GitHub directly into VS Code.

Features:

- View and manage issues and pull requests inside VS Code.
- Review code and add inline comments.
- Authenticate with your GitHub account for direct commits and PR creation.
- Access GitHub Copilot and Actions through the Command Palette.

To install:

1. Open VS Code → Extensions (`Ctrl+Shift+X`)
2. Search for **GitHub Pull Requests and Issues**
3. Install and sign in with GitHub.

---

15. GitHub Student Developer Pack

GitHub offers a **Student Developer Pack**, which provides free access to premium tools including:

- GitHub Pro and Copilot for Students
- Cloud credits (AWS, Azure, DigitalOcean)
- IDEs and developer tools (JetBrains, Canva, MongoDB Atlas, etc.)

Apply at: https://education.github.com/pack

Verification typically requires a school email or student ID.

---

16. Best Practices

1. Commit frequently with meaningful messages.
2. Always work on branches instead of directly on `main`.
3. Use `.gitignore` to exclude unnecessary files.
4. Review pull requests before merging.
5. Write clear documentation and maintain your `README`.
6. Keep repositories organized with labels, milestones, and projects.
7. Use GitHub Actions for continuous integration and testing.
8. Contribute to open-source projects to gain real-world experience.

---

17. Common Git Commands Reference

Action	Command
Initialize repo	`git init`
Clone repository	`git clone <url>`
Add changes	`git add <file>`
Commit changes	`git commit -m "message"`
View status	`git status`
View history	`git log`

Action	Command
Switch branch	`git checkout <branch>`
Merge branches	`git merge <branch>`
Push changes	`git push origin main`
Pull updates	`git pull`

---

18. Summary

GitHub is the foundation of modern software collaboration. It enables developers to:

- Version control their code with Git.
- Collaborate seamlessly using branches and pull requests.
- Automate workflows through Actions.
- Host web pages and share projects publicly.
- Integrate directly with IDEs like VS Code.

For students, GitHub is both a **learning platform** and a **professional portfolio**. By mastering its workflow, you gain the ability to contribute to open-source projects, collaborate with others globally, and showcase your technical skills to future employers.

GitHub’s combination of version control, collaboration, automation, and education resources makes it an essential part of every developer’s toolkit.