Introduction to GitHub

What is GitHub?

- GitHub is a cloud-based platform for version control and collaboration
- Built on Git, a distributed version control system
- Used by developers to store, manage, and collaborate on code
- Supports open-source and private projects

Why Use GitHub?

- Track changes in source code over time
- Collaborate with teams efficiently
- Backup code in the cloud (Manages code history)
- Integrates with CI/CD tools and project management tools

Continuous Integration and Continuous Delivery/Deployment (CI and CD)

- CI and CD is a DevOps practice that automates building, testing, and deploying applications. It helps developers deliver code faster, with fewer errors, and more confidence.

- **Continuous Integration (CI):** Developers frequently merge code changes into a shared repository. Automated builds and tests run to catch bugs early and ensure the codebase remains stable.

- **Continuous Delivery (CD):** Ensures that code is always in a deployable state. After CI, the application can be released to production at any time with minimal manual effort.

- **Continuous Deployment (CD):** Extends delivery by automatically deploying every validated change to production, making releases routine instead of stressful events

Benefits of CI and CD

Faster Development Cycles

- Automates repetitive tasks like builds and tests.
- Enables rapid feedback on code changes, reducing waiting time.

Improved Code Quality

- Early bug detection through automated testing.
- Reduces integration conflicts by merging code frequently.

Reduced Deployment Risks

- Smaller, incremental updates are easier to test and roll back.
- Releases become routine instead of high-stress events.

Better Collaboration

- Multiple developers can work on the same codebase without chaos.
- CI/CD pipelines manage changes systematically.

Higher Productivity

- Developers spend less time on manual testing and deployment.

Key GitHub Features

- Repositories
- Branches
- Commits
- Pull Requests
- Issues
- GitHub Actions

Repositories

- A repository (repo) is a storage space for a project
- Contains files, folders, and revision history
- Can be public or private
- Example: A repository for a Python web application

Branches

- Branches allow you to work on different versions of a project
- Main (or master) is the default branch
- Used for developing new features or fixing bugs
- Example: feature-login branch for adding login functionality

Commits

- A commit is a saved change in the repository
- Each commit has a unique ID (hash)
- Includes a message describing the change
- Example: 'Fixed login validation bug'

Pull Requests

- Pull Requests (PRs) propose changes to a repository
- Used for code review and discussion
- Merge changes from one branch into another
- Example: Merging feature-login into main branch

Issues

- Issues are used to track bugs, tasks, and enhancements
- Can be assigned to team members
- Support labels and milestones
- Example: Issue #12 – Fix password reset bug

GitHub Actions

- Automation tool for CI/CD workflows
- Triggered by events like push or pull request
- Used for testing, building, and deploying code
- Example: Automatically run tests on every push

Basic GitHub Workflow

- Create a repository
- Create a new branch
- Make changes and commit
- Open a pull request
- Review and merge

Tutorial: Creating a Repository

- Go to github.com and log in
- Click 'New Repository'
- Enter repository name and description
- Choose public or private
- Click 'Create Repository'

Tutorial: Cloning a Repository

- Copy repository URL from GitHub
- Open terminal or command prompt
- Run: git clone <repository-url>
- Example: git clone https://github.com/user/project.git

Tutorial: Making a Commit

- Modify files in the project
- Run: git status
- Run: git add .
- Run: git commit -m 'Your commit message'

Tutorial: Pushing Changes to GitHub

- Run: git push origin branch-name
- Uploads local commits to GitHub
- Example: git push origin main

Example Project Workflow

- Create a repository for a website
- Create a branch for a new feature
- Commit HTML/CSS changes
- Open a pull request
- Review and merge to main

Benefits for Students & Teams

- Improves collaboration and code quality
- Provides portfolio through public repositories
- Industry-standard tool
- Supports learning and open-source contribution

Conclusion

- GitHub is essential for modern software development
- Provides powerful collaboration and automation features
- Easy to learn with hands-on practice
- Start using GitHub to manage and share your projects