# Collaborating with GitHub

---

**1. Introduction to GitHub**

**What is GitHub?**

- A cloud-based platform built on top of Git, designed to host and manage Git repositories.
- Enables collaboration, code sharing, and project management.
- Key Features:
    - Pull Requests (PRs).
    - Issues and project boards.
    - GitHub Actions for CI/CD.
    - Wikis and documentation hosting.

**Purpose of GitHub:**

- Central repository for source code.
- Supports collaboration via pull requests and code reviews.
- Enables open-source contribution and enterprise-level source control.

**Real-World Example:**

- An enterprise hosts its internal projects on GitHub Enterprise, collaborating across multiple teams globally.
- Open-source projects (e.g., Kubernetes, Ansible) are maintained on GitHub, enabling community collaboration.

**Where to Implement:**

- Software development teams.
- DevOps automation and infrastructure teams.
- Open-source communities.

---

**2. Pushing Code to GitHub**

**git remote:**

- Links local repository to a remote repository.
- Example:

git remote add origin https://github.com/your-username/project.git
git remote -v

**git push:**

- Push local commits to remote repository.
- Example:

  git push -u origin main

**git pull:**

- Fetch and merge changes from remote repository.
- Example:

  git pull origin main

**git fetch:**

- Download changes from remote without merging.
- Example:

  git fetch origin

**Real-World Example:**

- A DevOps engineer pushes updated Terraform scripts to a shared GitHub repository, enabling the team to access the latest infrastructure configurations.

---

**3. Pull Requests (PRs) and Code Reviews**

**What is a Pull Request (PR)?**

- A formal request to merge code from one branch into another (usually feature branch to main).

**Purpose:**

- Enables collaborative code reviews.
- Ensures code quality and peer validation.
- Integrates CI/CD pipelines for automated tests.

**Creating a Pull Request:**

- After pushing feature branch:
  1. Navigate to GitHub repository.
  2. Click “Compare & pull request”.

3. Add PR description and reviewers.
4. Submit PR.

**Code Reviews:**

- Team members review code via GitHub's review interface.
- Approve, request changes, or comment inline.

**Real-World Example:**

- Feature development workflow: Developers submit PRs for every new feature; senior engineers perform code reviews before merging to production branches.

---

**4. Forking and Cloning**

**Forking:**

- Create a personal copy of another user's repository under your account.
- Used primarily in open-source contributions.

**Real-World Example:**

- A developer forks an open-source DevOps tool to propose changes via PRs without impacting the original project.

**Cloning:**

- Download (clone) a repository to local machine.
- Example:

  git clone https://github.com/your-username/project.git

**Real-World Example:**

- New team members clone the main project repository to their local environment to begin development or testing.

---

**5. Hands-On: Create GitHub Repository, Push Code, Create PR**

**Steps:**

1. **Create a Repository on GitHub:**
   - Navigate to https://github.com/
   - Click "New Repository".
   - Name repository and initialize with README.

2. **Push Local Code to GitHub:**

   ```
   git init
   git add .
   git commit -m "Initial commit"
   git remote add origin https://github.com/your-username/project.git
   git push -u origin main
   ```

3. **Create a Feature Branch Locally:**

   ```
   git checkout -b feature/new-feature
   # Make changes
   git add .
   git commit -m "Added new feature"
   git push origin feature/new-feature
   ```

4. **Open a Pull Request:**
   - Go to GitHub.
   - Compare branches and submit Pull Request.
5. **Perform Code Review and Merge PR:**
   - Reviewer reviews code, approves PR.
   - Author merges PR after approval.

**Real-World Example:**

- In a DevOps pipeline, feature branches with changes to deployment scripts are submitted via PRs, reviewed for security and quality, and merged into mainline infrastructure repositories.

---

**Webhooks & Integrations**

Webhooks notify external systems of GitHub events:

- Trigger Jenkins on push
- Send Slack notifications
- Integrate with Jira

**Example:** Jenkins GitHub Webhook Setup

1. Enable **GitHub plugin** in Jenkins.
2. Add webhook in GitHub repo settings → Payload URL = https://jenkins.example.com/github-webhook/.