1. GitHub Basics

- **What is GitHub?**
  - Introduction to version control and GitHub as a platform.
  - Key concepts: repositories, branches, commits, pull requests.
- **Getting Started**
  - Creating and cloning repositories.
  - Navigating the GitHub web interface.

---

2. Git Basics for GitHub

- **Version Control with Git**
  - Initializing a repository: `git init`.
  - Adding files: `git add`.
  - Committing changes: `git commit`.
  - Viewing history: `git log`, `git diff`.
- **Branching and Merging**
  - Creating and switching branches: `git branch`, `git checkout`.
  - Merging branches: `git merge`.
  - Resolving merge conflicts.
- **Remote Repositories**
  - Connecting to GitHub: `git remote`.
  - Pushing changes: `git push`.
  - Pulling updates: `git pull`.
  - Fetching changes: `git fetch`.

---

3. Repository Management

- **Repository Settings**
  - Configuring repository settings (visibility, branch protection).
  - Setting up webhooks and integrations.
- **Collaborating on Repositories**
  - Inviting collaborators and managing permissions.
  - Creating and managing teams in GitHub organizations.
- **Managing Large Repositories**
  - Using `.gitignore` to exclude files.
  - Cleaning up history with `git rebase`.
  - Git Large File Storage (LFS) for handling large files.

---

4. Pull Requests and Code Reviews

- **Creating Pull Requests**
  - Opening a pull request from a feature branch.
  - Writing effective pull request descriptions.
- **Code Review**
  - Commenting on code in pull requests.
  - Approving or requesting changes.
  - Understanding pull request statuses (mergeable or not).
- **Merging Pull Requests**
  - Squash and merge, rebase and merge, or create a merge commit.
  - Automating pull request merges with rules.

---

5. Automation in GitHub

- **GitHub Webhooks**

    - Setting up webhooks for triggering events.
    - Integrating GitHub with Jenkins, Terraform, or other DevOps tools.
- **Integrations**
    - Connecting GitHub with third-party tools like Slack, Jira, or .
    - Using GitHub API for custom automations.

---

6. Collaboration Best Practices

- **Commit Guidelines**
    - Writing meaningful commit messages.
    - Using tools like `git commit --amend` to refine commits.
- **Reviewing and Approving PRs**
    - Best practices for collaborative reviews.
    - Using labels and milestones to organize work.
- **Documenting Repositories**
    - Writing effective `README.md` files.
    - Adding contribution guides (`CONTRIBUTING.md`).
    - Using issue templates and PR templates.
