**GIT CASE STUDIES**

**---INDIVIDUAL---**

**Case Study 1: Setting Up a New Repository**

**User Story:**

As a software developer, I want to create a new repository for my project on GitHub and set it up locally on my machine so that I can start version controlling my code.

**Tasks:**

* Create a new repository on GitHub.
* Clone the repository to your local machine.
* Add a simple README.md file.
* Commit the changes and push them to GitHub.

**Solution:**

1. Create a new repository on GitHub.
   * Log in to GitHub.
   * Click on the "+" sign in the upper right corner and choose "New repository."
   * Follow the instructions to create the repository.
2. Clone the repository to your local machine.

*git clone <repository\_url>*

1. Add a simple README.md file.

*cd <repository\_directory> echo "# My Project" > README.md*

1. Commit the changes and push them to GitHub.

*git add README.md*

*git commit -m "Add README.md"*

*git push origin master*

**=================================================================**

**Case Study 2: Working on a Feature Branch**

**User Story:**

As a developer, I need to work on a new feature without affecting the main codebase until the feature is complete.

**Tasks:**

* Create a new branch named "feature/new-feature."
* Make changes to a file related to the new feature.
* Commit the changes.
* Push the feature branch to GitHub.

**Solution:**

1. Create a new branch named "feature/new-feature."

*git checkout -b feature/new-feature*

1. Make changes to a file related to the new feature.
2. Commit the changes.

*git add <file>*

*git commit -m "Implement new feature"*

1. Push the feature branch to GitHub.

*git push origin feature/new-feature*

**=================================================================**

**Case Study 3: Collaborative Branch Development**

**User Story:** As a development team, we want to work on a new feature collaboratively by creating feature branches.

**Tasks:**

1. Create a new branch named "feature/user-authentication."
2. Multiple team members make changes to the branch simultaneously.
3. Each team member commits their changes.
4. Merge the feature branch into the main branch.

**Solution:**

1. Create a new branch:

*git checkout -b feature/user-authentication*

1. Multiple team members make changes and commit.
2. Merge the feature branch into the main branch:

*git checkout main*

*git merge feature/user-authentication*

**=================================================================**

**Case Study 4: Repository Fork and clone with commit**

**User Story:** As an open-source contributor, I want to contribute to a project by forking the repository, making changes, and creating a pull request.

**Tasks:**

1. Fork a public repository on GitHub.
2. Clone the forked repository to your local machine.
3. Make changes and commit.

**Solution:**

1. Fork the repository on GitHub.
2. Clone the forked repository to your local machine:

*git clone <forked\_repository\_url>*

1. Make changes and commit.

==================================================================

**Case Study 5: Undoing Changes**

**User Story:** As a developer, I accidentally committed changes that shouldn't be part of the project. I want to undo the last commit.

**Tasks:**

1. Identify the last commit that needs to be undone.
2. Revert the last commit and keep changes in the working directory.

**Solution:**

1. Identify the last commit:

*git log*

1. Revert the last commit and keep changes in the working directory:

*git revert HEAD*

==================================================================

**Case Study 6: Renaming a Branch**

**User Story:** As a developer, I want to rename a branch to better reflect its purpose.

**Tasks:**

1. Identify the branch to be renamed.
2. Rename the branch locally.
3. Push the renamed branch to the remote repository.

**Solution:**

1. Identify the branch to be renamed:

*git branch -m old-branch new-branch*

1. Rename the branch locally:

*git branch -m old-branch new-branch*

1. Push the renamed branch to the remote repository:

*git push origin -u new-branch*

==================================================================

**Case Study 7: Git Log for Viewing Commit History**

**User Story:** As a developer, I want to view the commit history of the project.

**Tasks:**

1. Display a concise log of recent commits.
2. View a detailed log with commit messages and changes.

**Solution:**

1. Display a concise log of recent commits:

*git log --oneline*

1. View a detailed log with commit messages and changes:

*git log*

==================================================================

**Case Study 8: Git Pull for Updating Local Repository**

**User Story:** As a developer, I want to update my local repository with the latest changes from the remote repository.

**Tasks:**

1. Ensure you have committed or stashed any local changes.
2. Pull the latest changes from the remote repository.

**Solution:**

1. Commit or stash local changes.
2. Pull the latest changes from the remote repository:

*git pull origin main*

==================================================================

**Case Study 9: Deleting a Branch**

**User Story:** As a developer, I want to delete a feature branch after it has been merged into the main branch.

**Tasks:**

1. Ensure the branch to be deleted has been merged.
2. Delete the branch locally.
3. Delete the branch on the remote repository.

**Solution:**

1. Ensure the branch to be deleted has been merged.
2. Delete the branch locally:

*git branch -d feature-branch*

1. Delete the branch on the remote repository:

*git push origin --delete feature-branch*

==================================================================

**Case Study 10: Git Remote**

**User Story:** As a developer, I want to add a new remote repository to my local repository.

**Tasks:**

1. Identify the URL of the new remote repository.
2. Add the new remote to your local repository.

**Solution:**

1. Identify the URL of the new remote repository.
2. Add the new remote to your local repository:

*git remote add new-remote <repository\_url>*

==================================================================

**---GROUP---**

**Case Study: Setting Up and Pushing to GitHub**

**User Story 1: Initialization, First Commit, and GitHub Connection**

**User Story:** As a developer starting a new project, I want to initialize a Git repository, add some initial files, commit them, create a GitHub repository, and connect it to my local Git repository.

**Tasks:**

* 1. Navigate to the project's root directory.
  2. Initialize a new Git repository.
  3. Add a README.md file with project information.
  4. Commit the initial changes.
  5. Go to GitHub and create a new repository.
  6. Copy the repository URL.
  7. Add the GitHub repository as a remote to the local repository.

**Solution:**

bashCopy code

*# Navigate to the project's root directory*

*cd /path/to/project*

*# Initialize a new Git repository*

*git init*

*# Add a README.md file with project information*

*echo "# My Project" > README.md*

*# Commit the initial changes git add README.md*

*git commit -m "Initial commit"*

*# Go to GitHub and create a new repository*

*# Copy the repository URL*

*# Add the GitHub repository as a remote*

*git remote add origin https://github.com/your-username/your-repo.git*

**============================================================**

**User Story 2: Making Changes and Pushing to GitHub**

**User Story:** As a developer, I want to make changes to my project and push them to GitHub.

**Tasks:**

* 1. Make some changes to the README.md file.
  2. Add and commit the changes.
  3. Push the changes to the GitHub repository.

**Solution:**

bashCopy code

*# Make changes to the README.md file*

*echo "Additional information about my project" >> README.md*

*# Add and commit the changes*

*git add README.md*

*git commit -m "Update README"*

*# Push the changes to GitHub*

*git push -u origin main*

**=============================================================**

**User Story 3: Branching and Pull Requests**

**User Story:** As a developer, I want to create a new branch for a feature, make changes, and create a pull request on GitHub.

**Tasks:**

* 1. Create a new branch named "feature/my-feature."
  2. Make changes to a file.
  3. Commit the changes.
  4. Push the feature branch to GitHub.
  5. Create a pull request on GitHub.

**Solution:**

# Create a new branch

*git checkout -b feature/my-feature*

# Make changes, commit, and push the feature branch

# (Similar to the steps in previous case studies)

# Create a pull request on GitHub

# Navigate to GitHub, go to the repository, and create a pull request for the feature branch.