**Lesson 03 Demo 05**

**Rebasing in Git**

**Objective:** To perform rebase in Git for integrating changes from one branch to another while maintaining a linear commit history

**Tools required:** Git and GitHub

**Prerequisites:** None

Steps to be followed:

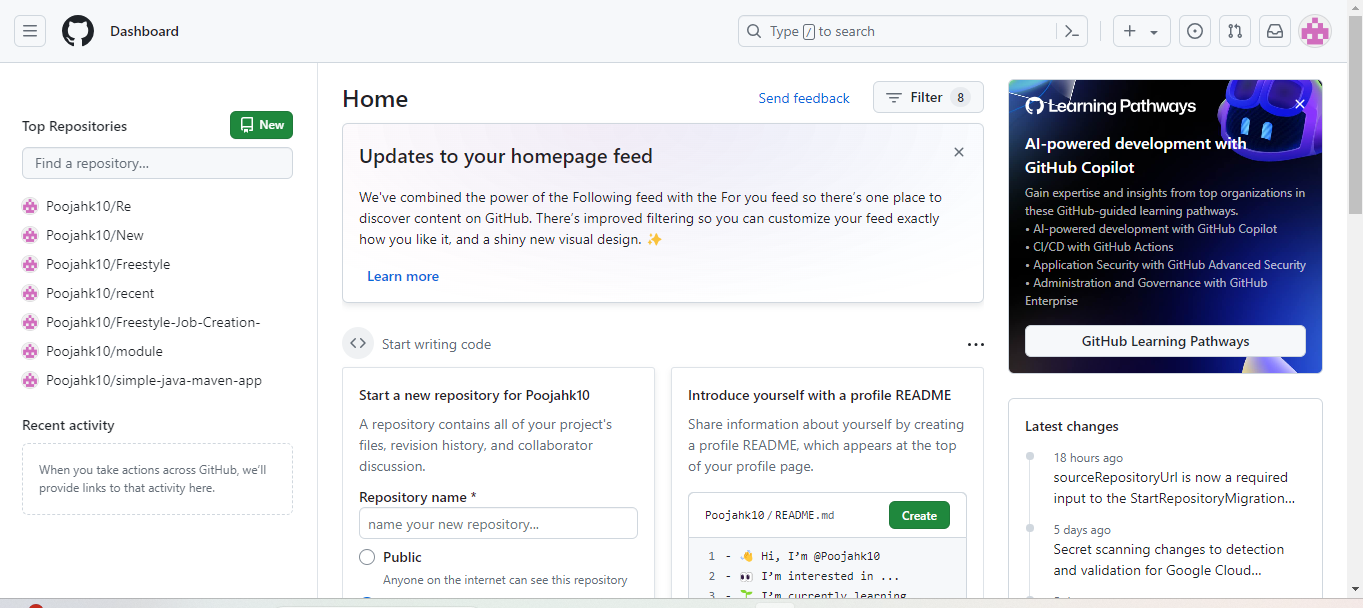
1. Create a repository
2. Clone the Git repository and rebase the branch to integrate the changes

**Step 1: Create a repository**

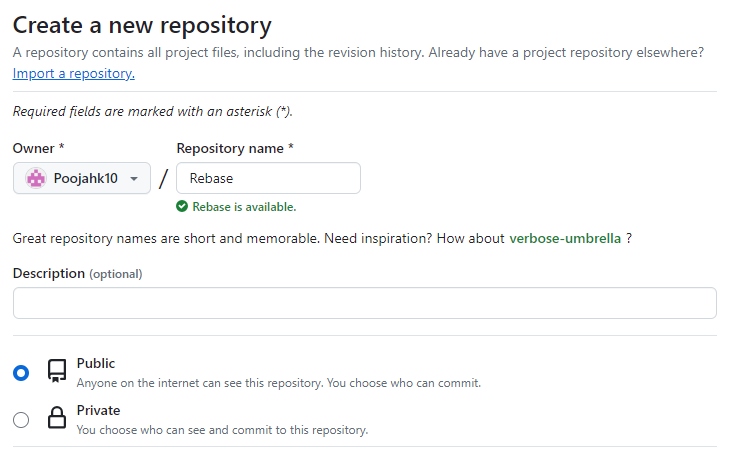
|  |
| --- |
| **Note**: If you do not have a GitHub account, visit the official website at <https://github.com/signup> and create a new account |

1. Open the browser in your practice lab, navigate to **github.com**, and sign in to your account as shown in the screenshot below:  
     
   A screenshot of a login form

   Description automatically generated
2. Click on the **New** button as shown in the screenshot below:

****

1. Enter a desired name for your repository and choose **Public** as shown in the screenshot below:

****

1. Click on the **Add a README file** checkbox and then click on **Create repository** as shown in the screenshot below:

**A screenshot of a computer

Description automatically generated**

1. Click on **Code** as shown in the screenshot below:

A screenshot of a computer

Description automatically generated

1. Copy the repository URL as shown in the screenshot below:

A screenshot of a computer

Description automatically generated

**Step 2: Clone the Git repository and rebase the branch to integrate the changes**

1. Open the Linux terminal in your lab and clone the repository using the following command:

**git clone <RepositoryURL>**

A close-up of a computer code

Description automatically generated

**Note**: Add the URL of your GitHub account repository in the place of **<RepositoryURL>**

* 1. Navigate inside the repository that you had created using the following command:

**cd RepositoryName/**



1. List the branches using the following command:

**git branch**

A green numbers on a white background

Description automatically generated

1. Create a new branch using the following command:

**git branch feature1**



1. Switch to the branch that you created in the previous step using the following command:

**git checkout feature1**

A close-up of numbers

Description automatically generated

1. List the branches using the following command:

**git branch**

A number with green numbers

Description automatically generated with medium confidence

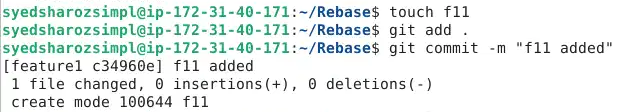
1. Create a file, stage the changes, and commit the staged changes in **feature1** branch using the following commands:

**touch f1  
git add .  
git commit -m “f1 added”**

A screenshot of a computer code

Description automatically generated

1. Create another file, stage the changes, and commit the staged changes using the following commands:

**touch f11  
git add .  
git commit -m “f11 added”**  
  


1. Run the following command to display the commit history of a repository:  
   **git log**

A screenshot of a computer code

Description automatically generated

1. Switch to the master branch using the following command:

**git checkout master**

A close-up of a computer code

Description automatically generated

1. List the branches using the following command:  
    **git branch**

A number and numbers on a white background

Description automatically generated

1. Create a file inside the master branch, stage the changes, and commit the staged   
    changes using the following commands:

**touch f22  
git add .  
git commit -m “f22 added”**  
   
A screenshot of a computer code

Description automatically generated

1. Run the following command to display the commit history of a repository:  
   **git log**

A screen shot of a computer code

Description automatically generated

1. Run the following command to rebase both branches: **git rebase feature1 master**



1. Run the following command to check the commit history of a repository after rebasing:  
   **git log**

A screenshot of a computer code

Description automatically generated   
  
By following these steps, you have successfully performed a rebasing in Git to integrate changes from one branch into another for seamless collaboration and code management.