**Lesson-End Project**

**Implementing Advance Operations in Git**

**Project agenda:** To execute advanced Git operations for enhanced version control and collaboration

**Description**: Imagine you are a software developer who has been asked to work on the Codex repository on GitHub. The goal is to solve issues using Git. First, create the Codex repo. Then, perform key Git tasks like tagging and branching. Understand Git rebase versus Git revert, explore Git log, and use Git rm with Git status. This project aims to improve teamwork and issue tracking for the Codex repository.

**Tools required:** Git and GitHub

**Prerequisites:** You must have Git installed in the lab to proceed.

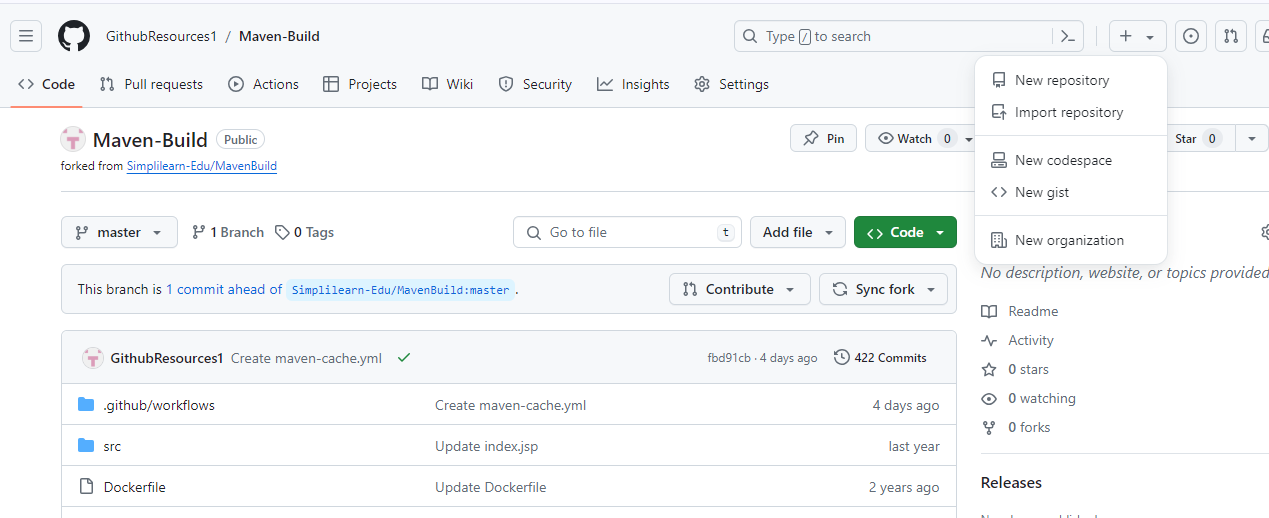
**Expected deliverables**: Creation of a GitHub repository named Codex with specified operations executed

Steps to be followed:

1. Create a new repository in Git
2. Create a tag in Git
3. Create a new branch in Git
4. Revert to the previous commit
5. Rebase the branch to integrate the changes
6. Remove the files from the Git index

**Step 1:** **Create a new repository in Git**

1. Click on the **+** icon in the upper-right corner of the page and select **New repository** from the drop-down menu



1. Provide the **Repository name** as **Codex**, select the **Add a README file**, choose **Public** for the repository type, and click on **Create repository**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Step 2:** **Create a tag in Git**

* 1. Open the terminal tab and execute the following command to create and navigate into the **Codex** project directory:

**mkdir Codex**

**cd Codex**

**A screenshot of a computer

Description automatically generated**

* 1. Execute the following command to clone the repository:

**git clone “repo path”**

**cd “repo name”**

**A computer code with numbers and text

Description automatically generated with medium confidence**

* 1. Execute the following command to create a new file:

**touch testfile1**

****

* 1. Execute the following command to commit the changes and create a tag in Git:

**git commit -m "soruce ver 1.0"**

**git tag -a 1.0 -m"version 1.0"**

A screenshot of a computer code

Description automatically generated

* 1. Execute the following command to push a tag:

**git push origin “tag name”**

A screenshot of a computer code

Description automatically generated

* 1. Navigate back to your GitHub repository to check the tag

A screenshot of a computer

Description automatically generated

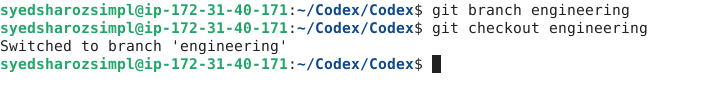
You can see the tag has been pushed.

**Step 3:** **Create a new branch in Git**

* 1. Open the terminal tab and execute the following command to create and switch to a new branch:

**git branch “branch name”**

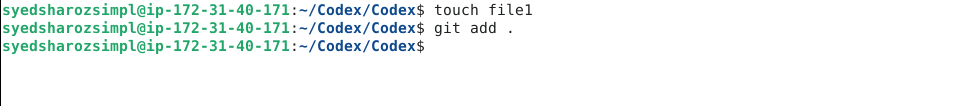
**git checkout “branch name”**

****

* 1. Execute the following commands to create and add a file in the **engineering** branch:

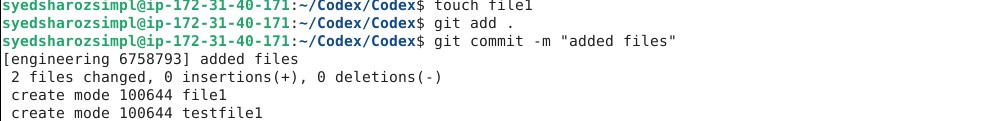
**touch file1**

**git add .**

****

* 1. Execute the following command to commit the changes:

**git commit -m “added files”**



* 1. Execute the following command to push the changes to the remote repository:

**git push origin “new branch name”**

A screenshot of a computer program

Description automatically generated

* 1. Navigate to the **Codex** repository in GitHub, and click on **Branches** to verify whether the branch has been pushed or not

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

You can see that a new branch is created.

**Step 4:** **Revert to the previous commit**

* 1. Use the command given below to create and switch to a new branch:

**git checkout -b test-branch**

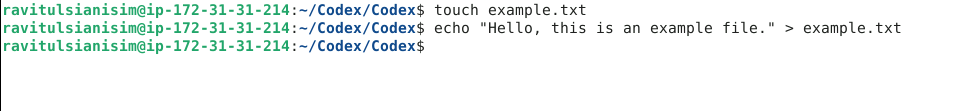
A screenshot of a computer

Description automatically generated

* 1. Execute the following commands to create a new file and modify it:

**touch example.txt**

**echo "Hello, this is an example file." > example.txt**



* 1. Run the following commands to add and commit the changes:

**git add example.txt**

**git commit -m "Add example file"**

**A screenshot of a computer code

Description automatically generated**

* 1. Run the following command to push the changes to the **test-branch** of the remote repository:

**git push origin test-branch**

A screenshot of a computer

Description automatically generated

* 1. Run the following commands to modify the **example.txt** file further:

**touch example.txt**

**echo "Hello world" > example.txt**

**A close up of a computer screen

Description automatically generated**

* 1. Run the following commands to add and commit the changes:

**git add example.txt**

**git commit -m "Modify example file"**

**A close-up of a computer code

Description automatically generated**

* 1. Execute the following command to inspect the commit history and identify the commit for reversal:

**git log --oneline**

A computer screen shot of text

Description automatically generated

* 1. Run the given command to revert to the preceding commit:

**git reset HEAD** **~1**

**A close-up of a computer screen

Description automatically generated**

* 1. Run the provided command to verify the commit history and confirm the successful reversal:

**git log --oneline**

**A computer code with text

Description automatically generated with medium confidence**

**Step 5:** **Rebase the branch to integrate the changes**

* 1. Execute the below command to create and switch to a new branch:

**git branch testing**

**git checkout testing**

A close-up of a computer code

Description automatically generated

* 1. Create a file, stage the changes in the current working directory, and commit the staged changes using the following commands:

**touch file5**

**git add .**

**git commit -m “files 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 file6**

**git add .**

**git commit -m “file6 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 screenshot of a computer code

Description automatically generated

* 1. Switch to the main branch using the following command:

**git checkout main**

A screenshot of a computer code

Description automatically generated

* 1. List the branch that you are currently working on using the following command:

**git branch**

A close-up of a computer code

Description automatically generated

* 1. Create a file inside a main branch, stage the changes in the current working directory, and commit the staged changes using the following commands:

**touch file7**

**git add .**

**git commit -m “file7 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 screenshot of a computer code

Description automatically generated

* 1. Run the following commands to rebase both the branches and check the commit history of a repository after rebasing:

**git rebase testing main**

**git log**

**A screenshot of a computer code

Description automatically generated**

**Step 6:** **Remove the files from the Git index**

* 1. Execute the following command to list all the files in the current branch:

**ls**

**A white background with black text

Description automatically generated**

* 1. Execute the following commands to remove **file5** in the current branch and check the status:

**git rm file5**

**ls**

**git status**

**A screenshot of a computer program

Description automatically generated**

You can see that **file5** has been removed.

By following these steps, you have successfully implemented advanced Git operations to enhance version control and collaboration.