**PRINCIPLE OF DEVOPS**

**LAB-03**

**NAME**: KRITTIKA V

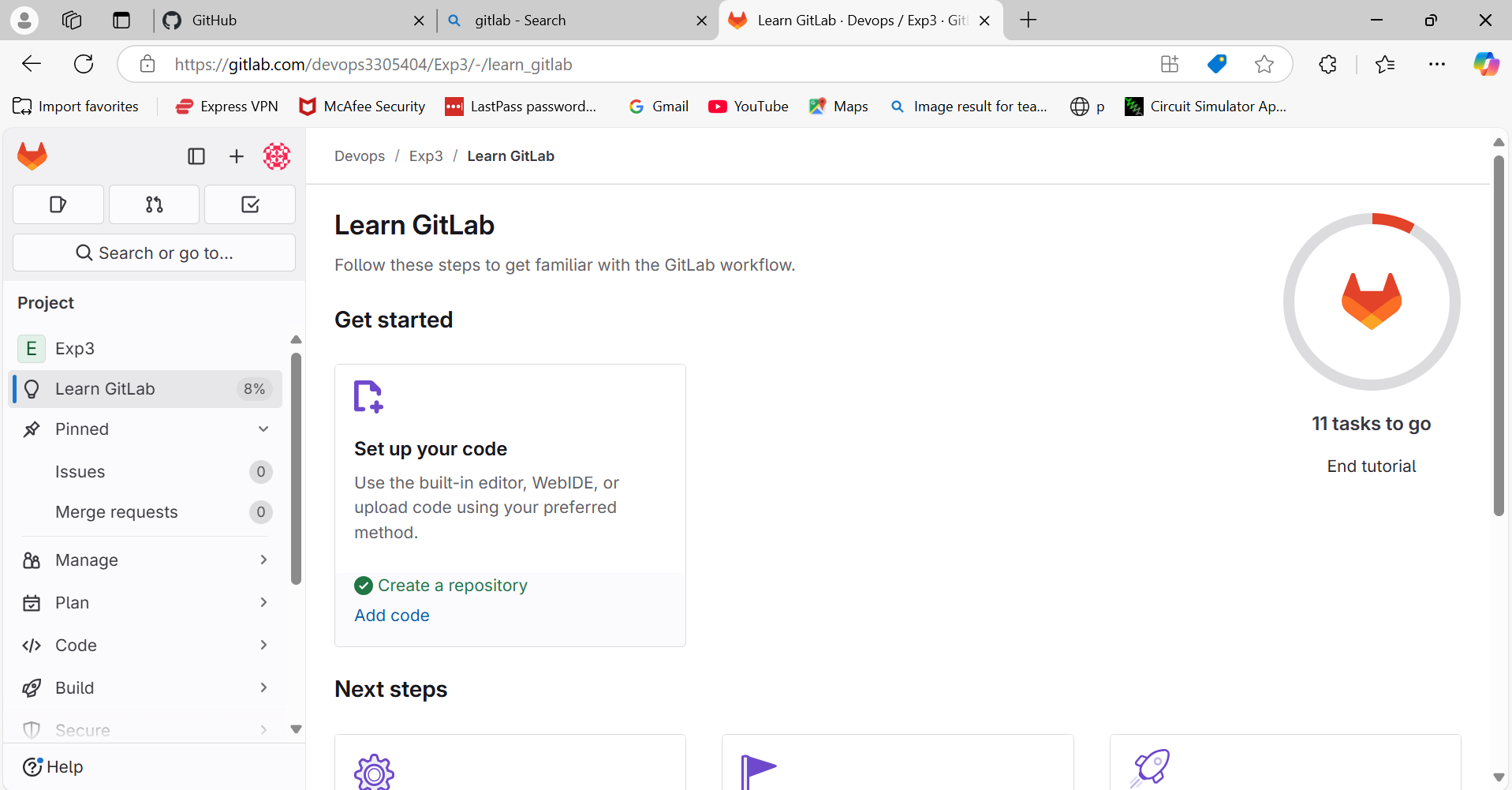
**ROLL NO:** 22BCS053

**Implement Git Lab Operations using Git.**

Experimental steps:

**Step 1: Creating a Repository**

* Sign in to your Git Lab account.



* Click the "New" button to create a new project.
* Choose a project name, visibility level (public, private), and other settings.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Click "Create project."

A screenshot of a computer

Description automatically generated

**Step 2: Cloning a Repository**

* Open your terminal or command prompt.
* Navigate to the directory where you want to clone the repository.
* Copy the repository URL from GitLab.
* Run the following command:

***git clone <repository\_url>***

* Replace <repository\_url> with the URL you copied from GitLab.
* This will clone the repository to your local machine.

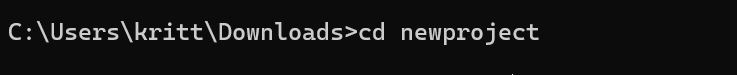
A computer screen with white text

Description automatically generated

**Step 3: Making Changes and Creating a Branch**

* Navigate into the cloned repository:

***cd <repository\_name>***



* Create a new text file named "example.txt" using a text editor.
* Add some content to the "example.txt" file.
* Save the file and return to the command line.

A screenshot of a computer

Description automatically generated

* Check the status of the repository:

***git status***

A screen shot of a computer code

Description automatically generated

* Stage the changes for commit:

***git add example.txt***

***git commit -m "Add content to example.txt"***

A computer screen shot of a program code

Description automatically generated

* Commit the changes with a descriptive message:
* Create a new branch named "feature":

***git branch feature***

* Switch to the "feature" branch:

***git checkout feature***

A black screen with white text

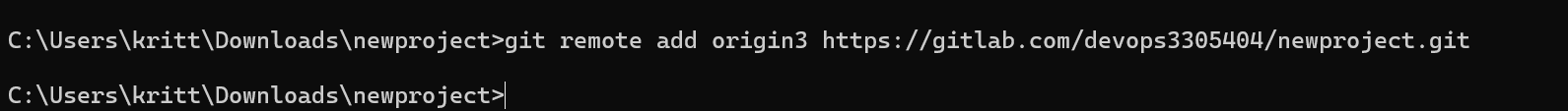
Description automatically generated

**Step 4: Pushing Changes to GitLab**

* Add Repository URL in a variable

***git remote add origin  <repository\_url>***

* Replace <repository\_url> with the URL you copied from GitLab.



* Push the "feature" branch to GitLab:

**git push origin feature**

* Check your GitLab repository to confirm that the new branch "feature" is available.

A computer screen with white text

Description automatically generated

**Step 5: Collaborating through Merge Requests**

1. Create a merge request on GitLab:
   * Go to the repository on GitLab.
   * Click on "Merge Requests" and then "New Merge Request."

A screenshot of a computer

Description automatically generated

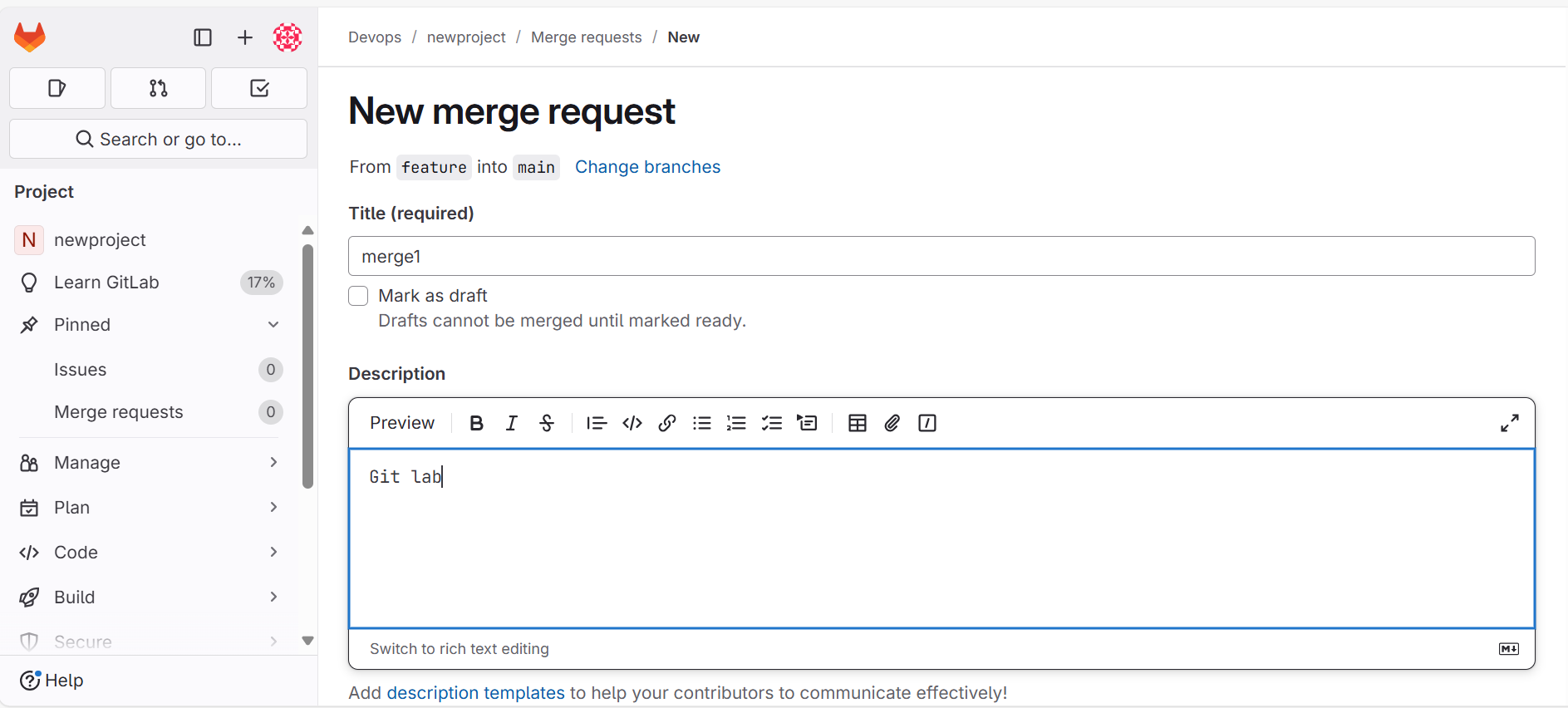
* + Choose the source branch ("feature") and the target branch ("main" or "master").

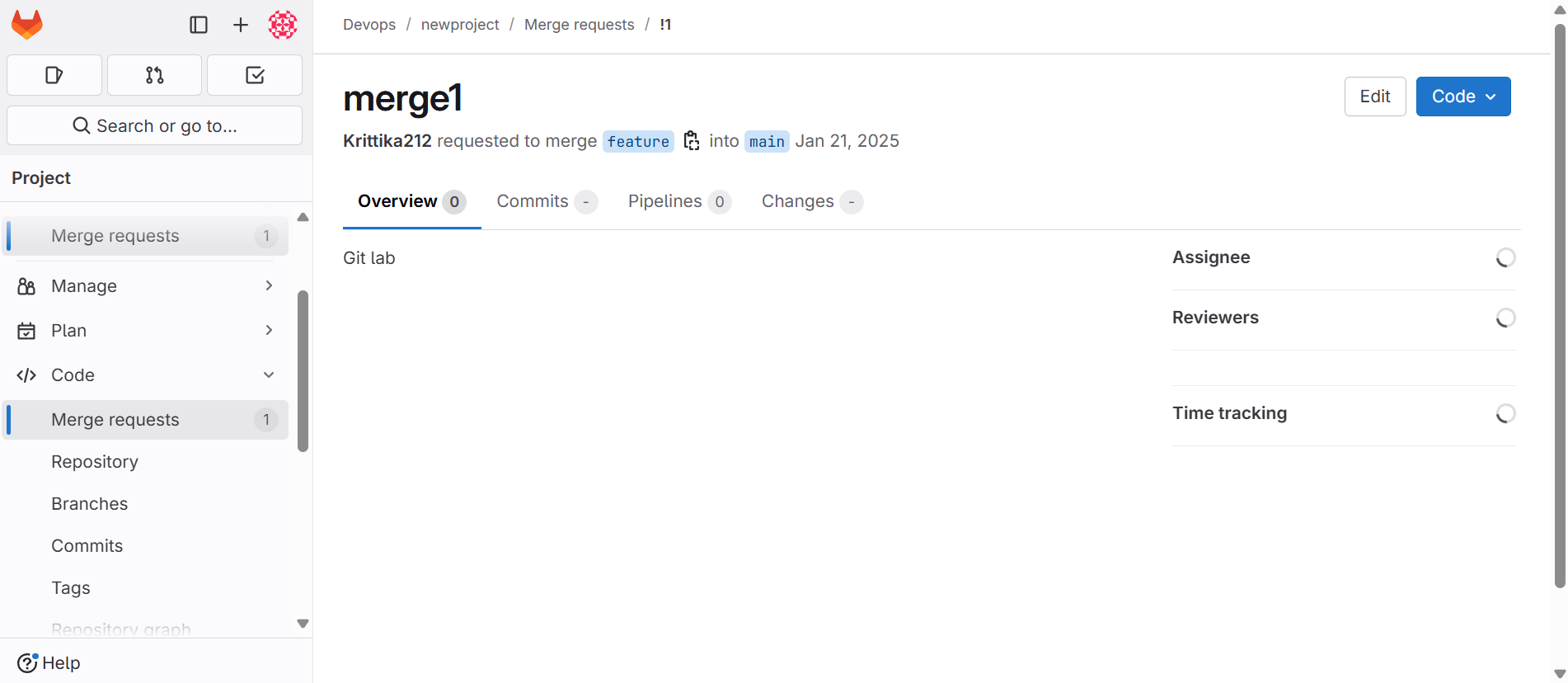
A screenshot of a computer

Description automatically generated

* + Review the changes and click "Submit merge request."

1. Review and merge the merge request:
   * Add a title and description for the merge request.
   * Assign reviewers if needed.
   * Once the merge request is approved, merge it into the target branch.





**Step 6: Syncing Changes**

* After the merge request is merged, update your local repository:

***git checkout main***

A screen shot of a computer code

Description automatically generated

***git pull origin main***



**Conclusion:**

This experiment provided you with practical experience in performing GitLab operations using Git commands. You learned how to create repositories, clone them to your local machine, make changes, create branches, push changes to GitLab, collaborate through merge requests, and synchronise changes with remote repositories. These skills are crucial for effective collaboration and version control in software development projects using GitLab and Git.

**Questions/Exercises:**

1. **What is GitLab, and how does it differ from other version control platforms?**

GitLab is a DevOps platform that integrates version control (using Git) with a wide range of development and operations tools, enabling end-to-end software development and delivery. Unlike other platforms like GitHub or Bitbucket, GitLab provides built-in CI/CD pipelines, advanced project management tools, and a single application for the entire software lifecycle.

1. **Explain the significance of a GitLab repository. What can a repository contain?**

A GitLab repository is a storage location for project files, including code, documentation, and configurations. It tracks changes and facilitates collaboration among developers. A repository can contain:

* Source code
* Documentation files
* CI/CD pipeline configurations (.gitlab-ci.yml)
* Project assets (e.g., images, data files)
* Issue and merge request history

1. **What is a merge request in GitLab? How does it facilitate the code review process?**

A merge request (MR) in GitLab is a feature used to propose, discuss, and review changes made in a branch before merging them into the main branch. It facilitates the code review process by:

* Allowing team members to comment on specific code lines
* Enforcing approval rules
* Automating tests and checks with CI/CD integration
* Ensuring quality and minimizing errors before deployment

1. **Describe the steps involved in creating and submitting a merge request on GitLab.**

* Create a branch for your changes from the repository's main branch.
* Commit your changes to the new branch.
* Push the branch to the remote repository on GitLab.
* Navigate to the **Merge Requests** section in the GitLab project.
* Click **New Merge Request**, select the source and target branches, and fill out the details.
* Submit the merge request and notify reviewers for feedback.
* Resolve comments, address issues, and finalize the MR for approval and merging.

1. **What are GitLab issues, and how are they used in project management?**

GitLab issues are tools for tracking tasks, bugs, feature requests, or any actionable work items. They are used in project management to:

* Create, assign, and prioritize tasks
* Track progress with labels, milestones, and due dates
* Facilitate discussions through comments and threads
* Integrate with project boards for visual task organization

1. **Explain the concept of a GitLab project board and its purpose in organising tasks.**

A GitLab project board is a visual representation of tasks, organized using cards in customizable columns (e.g., To-Do, In Progress, Done). It is used to:

* Manage workflow and track progress
* Enhance team collaboration
* Automate task movements using issue status changes
* Provide a clear overview of project milestones and deliverables

1. **How does GitLab address security concerns in software development? Mention some security-related features.**

GitLab addresses security concerns with robust features, including:

* **Static Application Security Testing (SAST):** Detects vulnerabilities in source code.
* **Dynamic Application Security Testing (DAST):** Identifies issues in running applications.
* **Dependency Scanning:** Detects vulnerabilities in project dependencies.
* **Secret Detection:** Scans for sensitive information like API keys in code.
* **Security Dashboards:** Centralizes vulnerabilities and their status for resolution.
* **Two-Factor Authentication (2FA):** Secures user accounts.

1. **Describe the role of compliance checks in GitLab and how they contribute to maintaining software quality.**

Compliance checks in GitLab enforce adherence to organizational and regulatory standards during software development. They contribute to quality by:

* Automating compliance policy enforcement (e.g., mandatory approvals, audit trails).
* Verifying code integrity with signed commits.
* Tracking security and quality metrics through reports.
* Ensuring only compliant code progresses through pipelines.