# MLA// EXTENDED ASSESSMENT TASK

*Please make a copy of this document*

| | |
|---|---|
| **NAME OF SESSION** | CI/CD Extended |
| **STUDENT NAME** | |

| | |
|---|---|
| **ASSESSMENT FOCUS** | Understanding CI/CD with GitLab |

**Scenario:**

You've joined a team of developers working on a web application for a local library. The team is currently using Git for version control, but they are considering migrating to GitLab to take advantage of its integrated CI/CD features.

**Task:**

Explain to your team how GitLab can improve their development workflow. Your explanation should cover the following aspects:

1. **Version Control:** Describe how GitLab's version control features compare to Git. Explain the benefits of using GitLab for managing code repositories and collaborating on code changes.

2. **CI/CD Integration:** Explain how GitLab's built-in CI/CD features can streamline the process of building, testing, and deploying their web application. Provide examples of how GitLab CI/CD can automate various tasks in their workflow.

3. **Collaboration and Issue Tracking:** Discuss how GitLab's issue tracking and project management tools can improve team communication and collaboration. Explain how these features can help them organize tasks, track progress, and resolve issues more efficiently.

# 1. Version Control

GitLab doesn't replace Git - it enhances it. Git remains the underlying version control system, but GitLab provides a powerful web-based interface and collaboration platform built around Git repositories. Think of it as Git with a comprehensive management layer.

**Key GitLab Version Control Benefits:**

# MLA// EXTENDED ASSESSMENT TASK

*Please make a copy of this document*

GitLab offers an intuitive web interface for visualizing repository history, branches, and commits without command-line complexity. Team members less comfortable with Git commands can browse code, review changes, and understand project structure through the browser. The platform provides protected branches, ensuring critical branches like `main` require code reviews before merging, preventing accidental direct commits that could break production.

Merge requests (GitLab's equivalent to pull requests) facilitate structured code review with inline comments, approval workflows, and automated checks. Reviewers can comment on specific code lines, suggest changes, and approve or reject submissions - all tracked in one place. GitLab also offers code snippets for sharing reusable code fragments and wikis for documentation, keeping everything centralized.

Repository management becomes easier with access controls, allowing granular permissions - some team members get read-only access while others can merge code. GitLab's search functionality helps locate code across repositories quickly, and the file browser with syntax highlighting makes code exploration seamless. For the library application, this means better collaboration, clearer code review processes, and reduced risk of problematic code reaching production while maintaining all Git's powerful version control capabilities underneath.

## 2. CI/CD Integration

GitLab's built-in CI/CD eliminates the need for separate tools, streamlining your library application's development pipeline. Unlike GitHub Actions or Jenkins that require additional setup, GitLab CI/CD is fully integrated from the start.

**How It Works:** You define pipelines in a `gitlab-ci.yml` file at your repository root. This configuration specifies stages like build, test, and deploy, with jobs running automatically on every code commit or merge request.

**Practical Examples for Your Library App:**

**Automated Testing:** When a developer pushes code adding a new book search feature, GitLab automatically runs unit tests, integration tests, and end-to-end tests. If tests fail, the merge request is blocked, preventing broken code from reaching production.

**Build Automation:** The pipeline compiles your application, bundles JavaScript/CSS assets, optimizes images, and generates production-ready builds without manual intervention. Each environment (development, staging, production) gets consistent builds.

**Automated Deployment:** After successful tests, GitLab automatically deploys to staging environments for review. Once approved, a single click deploys to production. For the library app, this means new

features like patron notifications or catalogue updates deploy smoothly without manual server configuration.

**Security Scanning:** GitLab runs dependency scanning, checking for vulnerable libraries in your application. If a security issue is detected in a package your library app uses, you're notified immediately before deployment.

This automation saves hours weekly, reduces human error, and ensures every deployment meets quality standards.

## 3. Collaboration and Issue Tracking

GitLab's integrated issue tracking and project management tools transform how your library app team collaborates, replacing scattered tools like Trello, Jira, and email threads with one unified platform.

**Issue Management:** Team members create issues for bugs ("Book checkout returns error for overdue items"), feature requests ("Add email notifications for due dates"), or technical debt ("Refactor authentication module"). Issues support markdown formatting, file attachments, labels for categorization (bug, enhancement, urgent), and assignments to specific developers. Unlike external trackers, issues link directly to code - you can reference commits, merge requests, and code snippets within issues, creating complete context.

**Boards and Milestones:** GitLab's Kanban-style boards visualize workflow stages (Backlog, In Progress, Review, Done), letting the team see who's working on what at a glance. Milestones group related issues for releases - "Version 2.0: Mobile Interface" might include 15 issues that need completion before launch, tracking overall progress automatically.

**Merge Request Integration:** When fixing an issue, developers create merge requests that automatically close related issues upon merging. Comments, code changes, and discussions stay connected, creating a complete audit trail. Team members receive notifications about mentions, assignments, and updates, eliminating the "I didn't know about that" problem.

For your library application, this means the entire team - developers, designers, project managers - operates from a single source of truth, improving communication, reducing duplicate work, and ensuring nothing falls through the cracks.