NAME:MIDHUNA R

ROLLNO:22BCS063

Experiment No. 3

Title: Implement GitLab Operations using Git.

Objective:

The objective of this experiment is to guide you through the process of using Git commands to interact with GitLab, from creating a repository to collaborating with others through merge requests.

Introduction:

GitLab is a web-based platform that offers a complete DevOps lifecycle toolset, including version control, continuous integration/continuous deployment (CI/CD), project management, code review, and collaboration features. It provides a centralized place for software development teams to work together efficiently and manage the entire development process in a single platform.

Key Features of GitLab:

- **Version Control:** GitLab provides version control capabilities using Git, allowing developers to track changes to source code over time. This enables collaboration, change tracking, and code history maintenance.
- **Repositories:** Repositories on GitLab are collections of files, code, documentation, and assets related to a project. Each repository can have multiple branches and tags, allowing developers to work on different features simultaneously.
- **Continuous Integration/Continuous Deployment (CI/CD):** GitLab offers robust CI/CD capabilities. It automates the building, testing, and deployment of code changes, ensuring that software is delivered rapidly and reliably.
- **Merge Requests:** Merge requests in GitLab are similar to pull requests in other platforms. They enable developers to propose code changes, collaborate, and get code reviewed before merging it into the main codebase.
- **Issues and Project Management:** GitLab includes tools for managing project tasks, bugs, and enhancements. Issues can be assigned, labeled, and tracked, while project boards help visualize and manage work.
- **Container Registry:** GitLab includes a container registry that allows users to store and manage Docker images for applications.
- **Code Review and Collaboration:** Built-in code review tools facilitate collaboration among team members. Inline comments, code discussions, and code snippets are integral parts of this process.

- **Wiki and Documentation:** GitLab provides a space for creating project wikis and documentation, ensuring that project information is easily accessible and well-documented.
- **Security and Compliance:** GitLab offers security scanning, code analysis, and compliance features to help identify and address security vulnerabilities and ensure code meets compliance standards.
- **GitLab Pages:** Similar to GitHub Pages, GitLab Pages lets users publish static websites directly from a GitLab repository.

Benefits of Using GitLab:

- **End-to-End DevOps:** GitLab offers an integrated platform for the entire software development and delivery process, from code writing to deployment.
- **Simplicity:** GitLab provides a unified interface for version control, CI/CD, and project management, reducing the need to use multiple tools.
- **Customizability:** GitLab can be self-hosted on-premises or used through GitLab's cloud service. This flexibility allows organizations to choose the hosting option that best suits their needs.
- **Security:** GitLab places a strong emphasis on security, with features like role-based access control (RBAC), security scanning, and compliance checks.
- **Open Source and Enterprise Versions:** GitLab offers both a free, open-source Community Edition and a paid, feature-rich Enterprise Edition, making it suitable for individual developers and large enterprises alike.

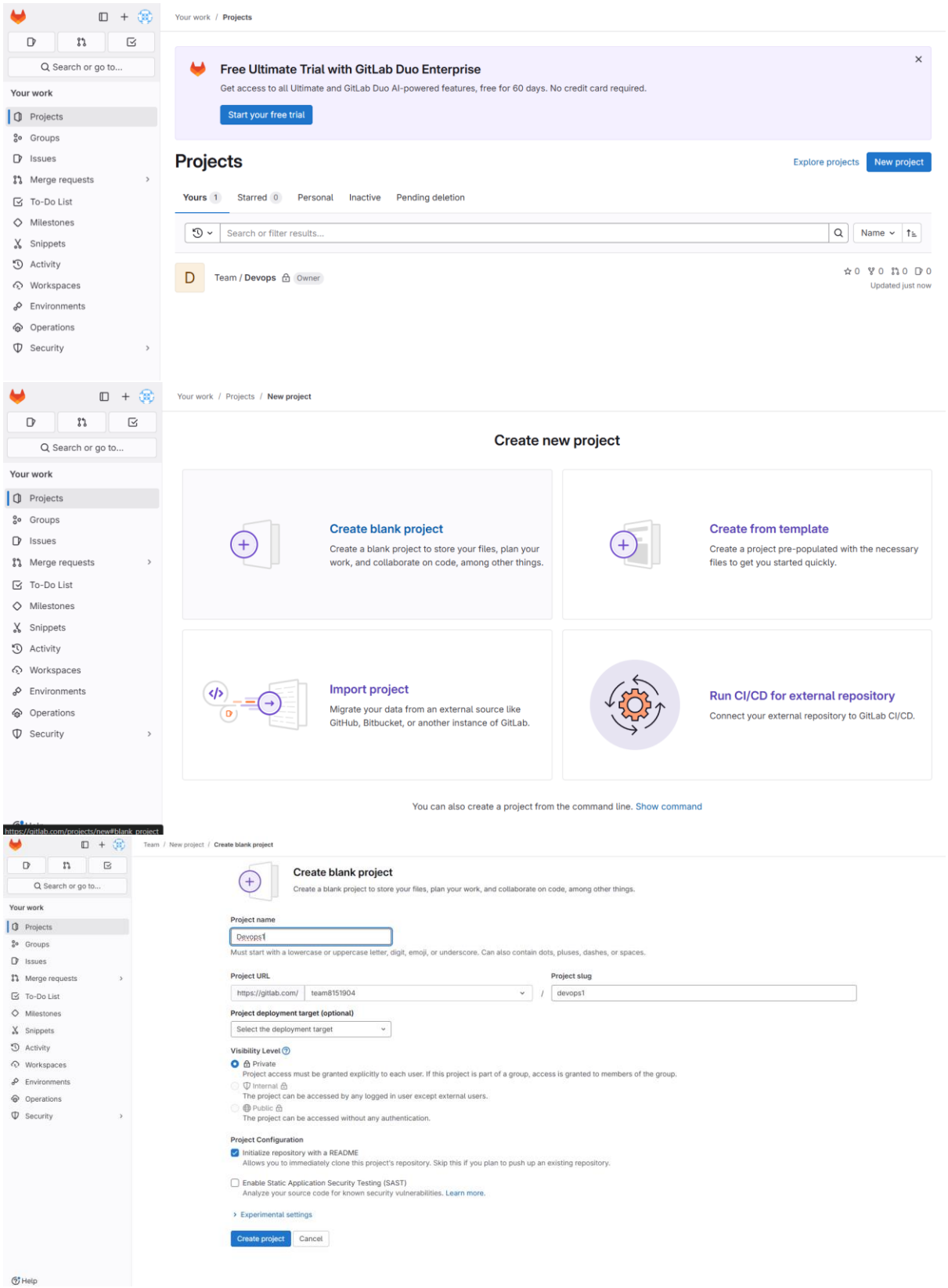
Materials:

- Computer with Git installed (https://git-scm.com/downloads)
- GitLab account (https://gitlab.com/)
- Internet connection

Experiment Steps:

Step 1: Creating a Repository

- Sign in to your GitLab account.
- Click the "New" button to create a new project.
- Choose a project name, visibility level (public, private), and other settings.
- Click "Create project."



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.

```
C:\Users\midhu\Downloads\Devops>git clone https://gitlab.com/team8151904/devops1
Cloning into 'devops1'...
info: please complete authentication in your browser...
warning: redirecting to https://gitlab.com/team8151904/devops1.git/
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```

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.
- Check the status of the repository:

git status

- Stage the changes for commit:

```
C:\Users\midhu\Downloads\Devops\devops1>git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        example.txt

nothing added to commit but untracked files present (use "git add" to track)
```

git add example.txt

```
C:\Users\midhu\Downloads\Devops\devops1>git add example.txt
```

- Commit the changes with a descriptive message:

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

```
C:\Users\midhu\Downloads\Devops\devops1>git commit -m "old version"
[main 5c96bbb] old version
 1 file changed, 1 insertion(+)
 create mode 100644 example.txt
```

- Create a new branch named "feature":

git branch feature

- Switch to the "feature" branch:

git checkout feature

```
C:\Users\midhu\Downloads\Devops\devops1>git branch feature

C:\Users\midhu\Downloads\Devops\devops1>git checkout feature
Switched to branch 'feature'
```

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

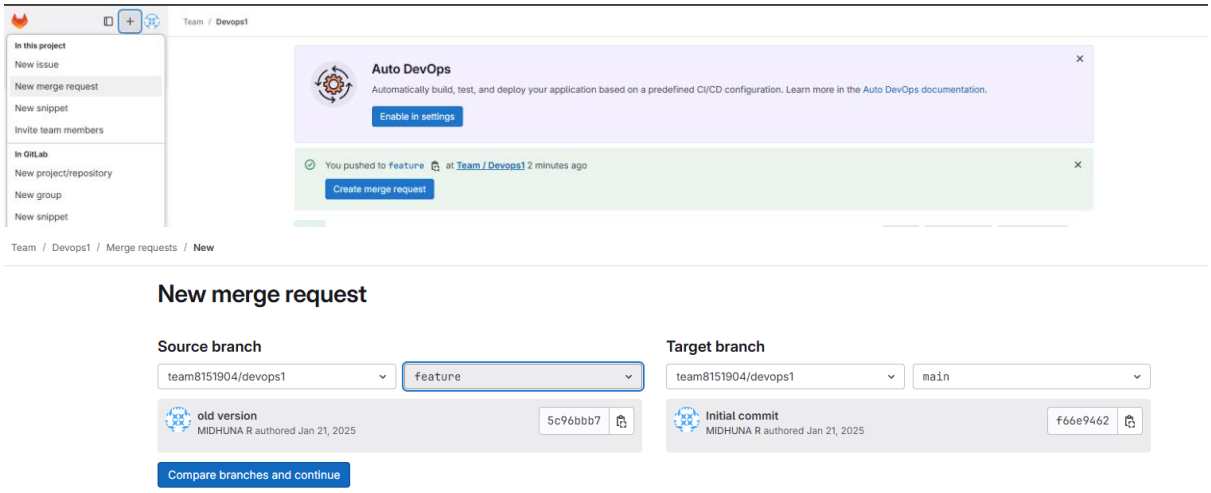
- Check your GitLab repository to confirm that the new branch "feature" is available.

```
:\Users\midhu\Downloads\Devops\devops1>git remote add origin https://gitlab.com/team8151904/devops1.git
error: remote origin already exists.
```

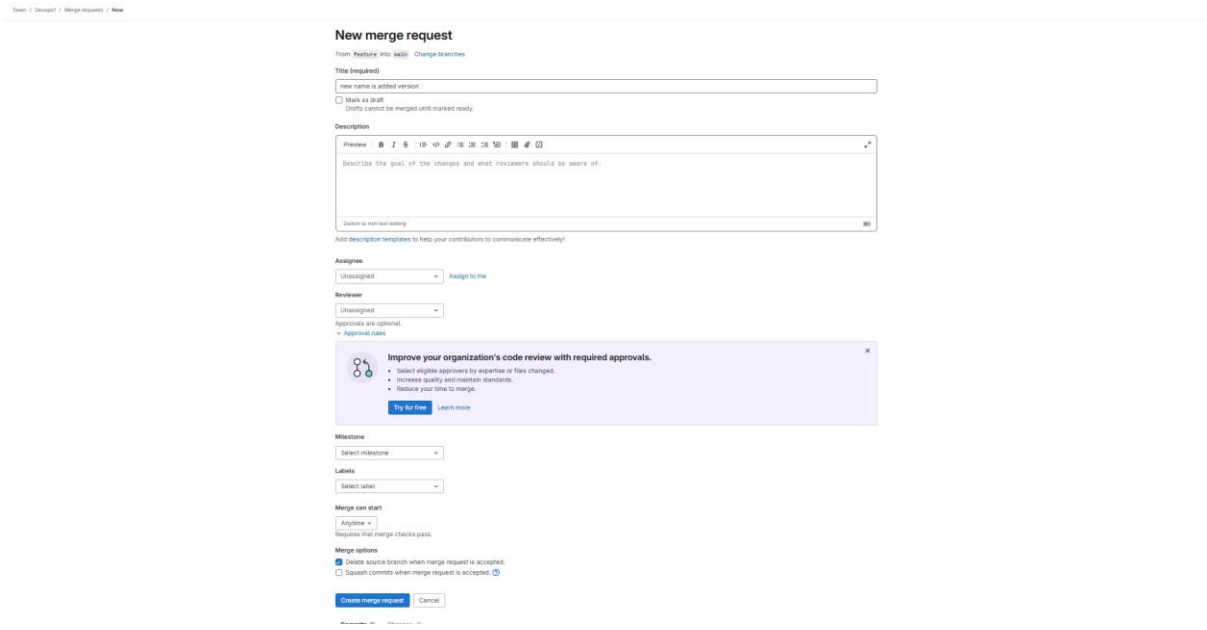
```
C:\Users\midhu\Downloads\Devops\devops1>git add example.txt

C:\Users\midhu\Downloads\Devops\devops1>git commit -m "Name is added in the version"
[feature 6f084b4] Name is added in the version
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\midhu\Downloads\Devops\devops1>git push origin feature
OAuth token info request failed: A task was canceled.
warning: redirecting to https://gitlab.com/team8151904/devops1.git/
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 305 bytes | 305.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: View merge request for feature:
remote:   https://gitlab.com/team8151904/devops1/-/merge_requests/1
remote:
To https://gitlab.com/team8151904/devops1
   5c96bbb..6f084b4  feature -> feature
```

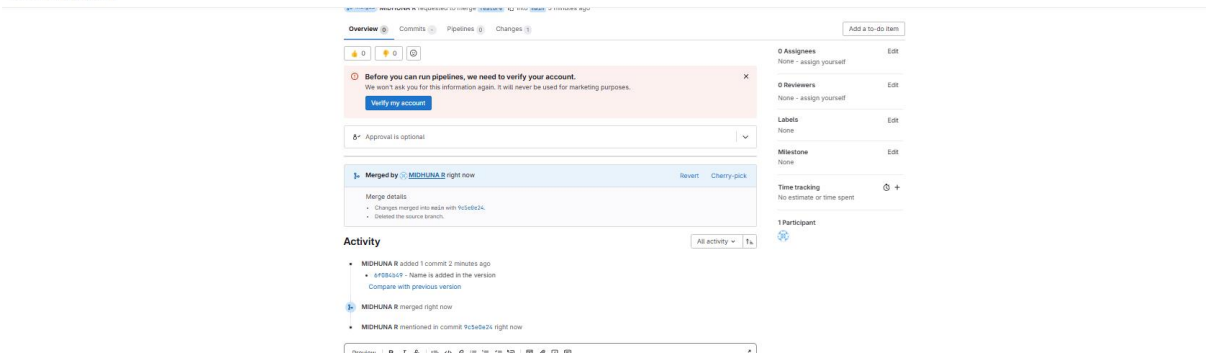
Step 5: Collaborating through Merge Requests

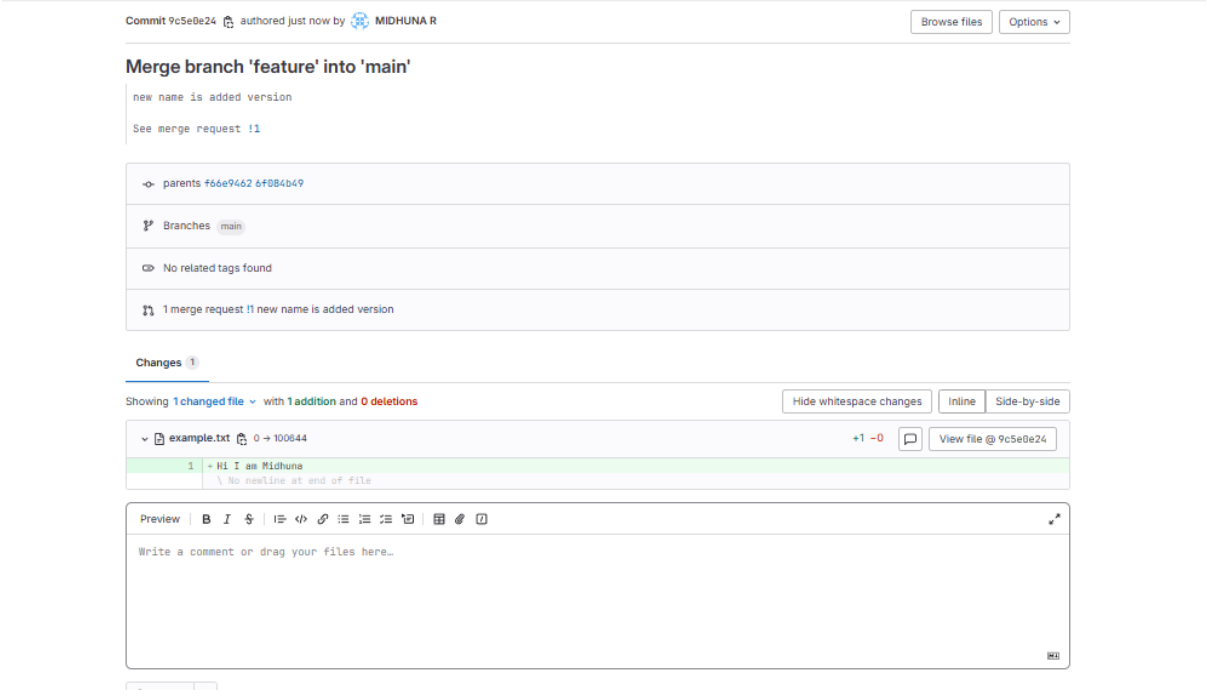
1. Create a merge request on GitLab:
    - Go to the repository on GitLab.
    - Click on "Merge Requests" and then "New Merge Request."
    - Choose the source branch ("feature") and the target branch ("main" or "master").
    - Review the changes and click "Submit merge request."

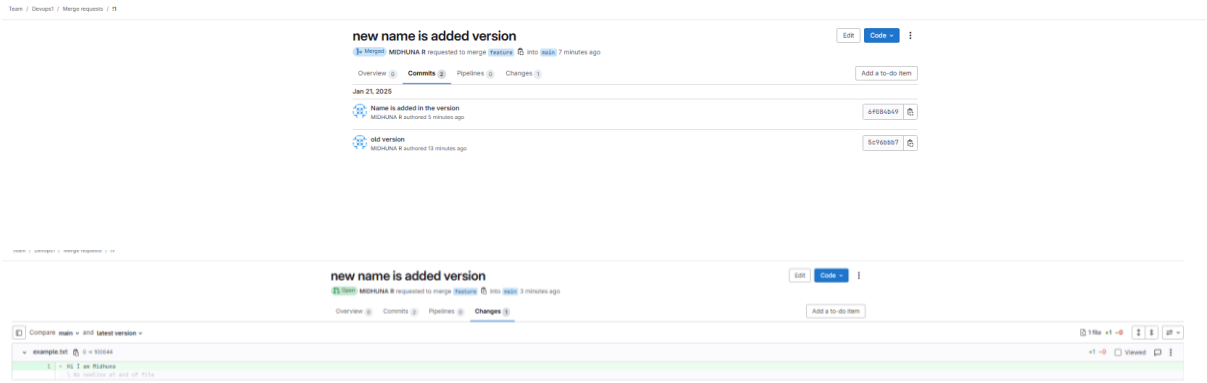


2. Review and merge the merge request:
    - Add a title and description for the merge request.
    - Assign reviewers if needed.
    - Once the merge request is approved, merge it into the target branch.


New merge request
From feature into main Change branches
Title (required)
new name is added version
Mark as draft
Drafts cannot be merged until marked ready.
Description
Preview
Describe the goal of the changes and what reviewers should be aware of.
Add description templates to help your contributors to communicate effectively!
Assignee
Unassigned
Assign to me
Reviewer
Unassigned
Approvals are optional.
Approval rules
Improve your organization's code review with required approvals.
Select eligible approvers by expertise or files changed.
Increase quality and maintain standards.
Reduce your time to merge.
Try for free
Learn more
Milestone
Select milestone
Labels
Select label
Merge can start
Anytime
Requires that merge checks pass.
Merge options
Delete source branch when merge request is accepted.
Squash commits when merge request is accepted.
Create merge request
Cancel


Overview
Commits
Pipelines
Changes
Add a to-do item
Before you can run pipelines, we need to verify your account.
We won't ask you for this information again. It will never be used for marketing purposes.
Verify my account
Approval is optional
Merged by MIDHUNA R right now
Revert
Cherry-pick
Merge details
Changes merged into main with 9c5e0e24.
Deleted the source branch.
Activity
All activity
MIDHUNA R added 1 commit 2 minutes ago
6fDB4b49 - Name is added in the version
Compare with previous version
MIDHUNA R merged right now
MIDHUNA R mentioned in commit 9c5e0e24 right now
0 Assignees
None - assign yourself
Edit
0 Reviewers
None - assign yourself
Labels
None
Milestone
None
Time tracking
No estimate or time spent
1 Participant


Commit 9c5e0e24 authored just now by MIDHUNA R
Browse files
Options
Merge branch 'feature' into 'main'
new name is added version
See merge request !1
parents f66e9462 6fDB4b49
Branches main
No related tags found
1 merge request !1 new name is added version
Changes 1
Showing 1 changed file with 1 addition and 0 deletions
Hide whitespace changes
Inline
Side-by-side
example.txt 0 → 100644
+1 −0
View file @ 9c5e0e24
1 + Hi I am Midhuna
\ No newline at end of file
Preview
Write a comment or drag your files here…



Step 6: Syncing Changes

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

git checkout main

git pull origin main

```
C:\Users\midhu\Downloads\Devops\devops1>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

C:\Users\midhu\Downloads\Devops\devops1>git pull origin main
warning: redirecting to https://gitlab.com/team8151904/devops1.git/
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (1/1), 265 bytes | 66.00 KiB/s, done.
From https://gitlab.com/team8151904/devops1
 * branch            main       -> FETCH_HEAD
   f66e946..9c5e0e2  main       -> origin/main
Updating 5c96bbb..9c5e0e2
Fast-forward
 example.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

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 platform that helps developers work together on coding projects. It provides tools for version control, testing, deployment, and managing tasks all in one place.

How GitLab is Different from Other Platforms

1. **All-in-One**: Combines version control, testing, and project management in one platform.
2. **Self-Hosting**: Can be installed on your own servers, unlike some other platforms.
3. **Built-In CI/CD**: Includes tools to test and deploy code automatically.
4. **Security Tools**: Offers features like scanning for vulnerabilities and detecting secrets in code.
5. **Open Source**: The free version is open source, giving you more control.

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

A **GitLab repository** is a storage space for code, configuration files, and related data hosted on GitLab.

Contents:

- **Code Files**: Source code for applications or libraries.
- **Documentation**: README files, guidelines, or wikis.
- **CI/CD Configurations**: .gitlab-ci.yml files for setting up pipelines.
- **Issues and Merge Requests**: Track progress and code changes.
- **Artifacts**: Build outputs, logs, or test results generated by CI/CD pipelines.

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

A **merge request (MR)** in GitLab is a proposal to merge changes from one branch into another.

Role in Code Reviews:

- Allows team members to review, comment on, and discuss changes before merging.
- Provides tools for viewing diffs, testing pipelines, and resolving conflicts.
- Ensures code quality and alignment with team standards.

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

Create a Branch:

- Create a new branch for your feature or bug fix.

Make Changes:

- Commit and push changes to the branch.

Open Merge Request:

- Go to the GitLab project, navigate to **Merge Requests**, and click **"New Merge Request"**.

Select Branches:

- Choose the source branch (your branch) and target branch (e.g., main or develop).

Add Details:

- Provide a title, description, and any necessary labels or assignees.

Submit for Review:

- Click **"Submit"** to create the merge request.

Collaborate:

- Respond to feedback, make additional commits if required, and update the MR.

Merge:

- After approval, merge the request into the target branch.

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

   GitLab Issues: Built-in tools for tracking tasks, bugs, and feature requests.

   Use in Project Management:

- Assign issues to team members to track responsibility.
- Add **labels**, **milestones**, and **due dates** for organization.
- Discuss issues using comments and attach files or code snippets.
- Link issues to merge requests for better traceability.

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

   The **GitLab Project Board** is a Kanban-style tool for visualizing and organizing tasks.

   Purpose:

- Tracks project progress with columns (e.g., To Do, In Progress, Done).
- Allows drag-and-drop movement of issues between columns.
- Facilitates agile workflows like Scrum and Kanban.

- Improves team collaboration by providing a clear overview of tasks and priorities.

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

   GitLab provides several security-related features to protect code and ensure compliance:

- **Static Application Security Testing (SAST)**: Scans code for vulnerabilities.
- **Dependency Scanning**: Identifies vulnerabilities in libraries and packages.
- **Secret Detection**: Finds hardcoded sensitive information (e.g., API keys).
- **Container Scanning**: Detects vulnerabilities in Docker images.
- **Dynamic Application Security Testing (DAST)**: Scans running applications for security issues.
- **Audit Logs**: Tracks user actions and changes for accountability.

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

Compliance checks ensure that software adheres to organizational, regulatory, or industry standards.

Contribution to Software Quality:

- Enforces coding standards through automated rules.
- Requires approval from specific roles or groups before merging.
- Tracks compliance metrics via audit reports.
- Automates license compliance checks to avoid using unapproved dependencies.