**Note:**

Given that it's a standalone, individual assignment, I recommend placing it near the **end of the semester**. Here's why:

1. **Solidifying Concepts**: By then, students will have developed a clearer understanding of Git, GitHub, and collaborative workflows, which will make the transition to GitLab smoother.
2. **Standalone Context**: Since the assignment doesn't depend on teamwork, it won't disrupt sprint dynamics or require alignment with team schedules.
3. **Capstone Feel**: Introducing CI/CD at the end ties together the semester's learning, showcasing how all the pieces (Git, collaboration, quality assurance) work in a real-world context.

You could frame it as a final skill-building exercise, preparing them for technical interviews or giving them an edge as future business analysts by understanding software delivery pipelines.

**Assignment: Continuous Integration with GitLab**

**Overview**

In this assignment, you will gain hands-on experience with Continuous Integration (CI) by setting up a pipeline on GitLab to run a security scan on a JavaScript program. This activity will introduce you to key CI concepts, such as pipelines, stages, and jobs, and provide a foundation for understanding how CI can improve software quality.

**Learning Objectives**

1. Understand the purpose of CI in software development.
2. Set up and trigger a basic CI pipeline on GitLab.
3. Perform a security scan using ESLint on a JavaScript program.
4. Use Git to push code to a GitLab repository.
5. Reflect on the role of CI in collaborative development.

**Assignment Instructions**

**Part 1: Setup**

1. **Write a JavaScript Program:**
   * Create a JavaScript file (validatepalindrome.js) that:
     + Accepts an array of strings as input.
     + Use a function to determine if each string is a palindrome. A palindrome is a word, phrase, number, or sequence of characters that reads the same backward as forward, ignoring capitalization and spaces. A string of length one or zero is a palindrome.
     + Return a list of strings that are palindromes.
     + Recommend: pre-populate the array as a constant or as a var in order to simplify the programming.
   * Example input and output:
     + Input: ["racecar", "hello", "Level", “I”, "world", "mad am", "12321", "not a palindrome"]
     + Output: racecar, Level, I, mad am, “\12321]
     + Output should be displayed on the console using console.log()
   * Tips:
     + Divide the program into functions
     + Use the toLowerCase() and replace() functions for cleaning the input.
       - Example: in\_string.toLowerCase(); // converts in\_string to all lower case
       - Example: new\_string.replace(" ", ""); // replaces spaces with nulls, eliminates spaces
     + Use the split(), reverse(), and join() functions to reverse a string.
       - Example: return in\_string.split('').reverse().join(''); // This code reverses in\_string, for example, if in\_string = “Mike” this code will produce “ekiM”
2. **Initialize Git:**
   * Commit your validatepalindrome.js program to your local Git repository.
3. **Push to GitLab:**
   * Create an account on GitLab at [www.gitlab.com](http://www.gitlab.com).
   * When registering, you can say that you will be the only person using GitLab.
   * Create a new project on GitLab.
     + Group should be INFO465
     + Project name should be “simple\_pipeline\_build”
     + Do not select a template.
   * Two options to add your .js program
     + Option 1: Add the GitLab remote to your local repository and push your code:
       - Will need to create a SSH key. Instructions are posted on Canvas in the file “ssh\_key\_for\_gitlab.docx”
       - git remote add origin <GitLab-repository-URL>
       - git push -u origin main
     + Option 2: Use the GitLab GUI to add and commit a file.
       - Within your project, click on the + sign.
       - Select “Upload file”

**Part 2: Configure CI Pipeline**

1. **Add the .gitlab-ci.yml File:**
   * Create a .gitlab-ci.yml file in your project directory:
     + Click the **"+" (plus)** button then **"New file"**
     + In the "File name" field, enter .**gitlab-ci.yml**.
     + Define the CI/CD pipeline. The pipeline has a single stage that uses a Node.js container is used to run ESLint, a static code analysis tool, to check JavaScript files for code quality and security issues. The code is:

*stages:*

*- security-scan*

*security\_scan:*

*stage: security-scan*

*image: node:latest*

*script:*

*- npm install eslint*

*- npx eslint --init*

*- npx eslint \*.js artifacts:*

*when: always*

*paths:*

*- eslint\_report.json*

*reports:*

*codequality: eslint\_report.json*

1. **Push the File to GitLab:**
   * Commit and push the .gitlab-ci.yml file to trigger the pipeline.
   * git add .gitlab-ci.yml
   * git commit -m "Add GitLab CI pipeline for security scan"

git push

1. **Run the Pipeline:**
   * Check your GitLab project’s CI/CD > Pipelines section to monitor the pipeline execution.
   * Verify that the pipeline completes successfully.

**Part 3: Submission**

Submit the following:

1. A screenshot of your pipeline showing a successful run.
2. Your JavaScript file (validateEmail.js).
3. A brief reflection (150-200 words) addressing:
   * What you learned about CI.
   * How CI can improve software development quality and collaboration.

**Rubric**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criteria** | **Exemplary (90-100)** | **Proficient (75-89)** | **Developing (50-74)** | **Incomplete (<50)** |
| **JavaScript Program** | Fully functional and meets the specification. | Functional with minor issues. | Significant errors or incomplete implementation. | Missing or non-functional. |
| **GitLab Setup** | Code successfully pushed and visible on GitLab. | Code pushed with minor setup issues. | Errors in pushing code; setup incomplete. | Code not pushed to GitLab. |
| **CI Pipeline Execution** | Pipeline executes successfully without errors. | Pipeline executes with minor errors or warnings. | Pipeline setup incomplete or fails to execute. | No evidence of pipeline execution. |
| **Reflection** | Insightful, detailed, and connects CI concepts to practice. | Adequate but lacks depth or connection to CI concepts. | Superficial or vague reflection. | Missing or fails to address the reflection prompts. |
| **Submission Quality** | All required files and screenshots submitted; well-organized. | Most files submitted but minor omissions. | Several required components missing. | Incomplete submission or disorganized. |

**Instructor Notes**

* **Pre-Assignment Support:**
  + Provide a live demo or video walkthrough of:
    - Creating a GitLab account and project.
    - Basic Git commands (e.g., commit, push).
    - Navigating GitLab’s CI/CD pipeline interface.
* **Troubleshooting:**
  + Host a Q&A session or lab time to assist students.
  + Share a troubleshooting guide for common Git and GitLab issues.

**Advanced Challenge (Optional)**

For students seeking more challenge, allow them to:

1. Modify the ESLint rules.
2. Add another stage (e.g., run unit tests or check for specific code patterns).

**Final Note**

This assignment is designed to balance technical learning with practical exposure. It introduces CI concepts without requiring deep technical expertise, preparing students for future roles where CI/CD is essential.