# 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 unit test and two security scans 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 improves software quality.

## Learning Objectives

* Understand the purpose of CI in software development.
* Set up and trigger a basic CI pipeline on GitLab.
* Write and run unit tests using Jest.
* Perform two security scans:  
   - A JSHint scan to identify potential errors and enforce coding standards.  
   - A Secret Detection scan to look for sensitive information.
* Use Git to push code to a GitLab repository.
* Reflect on the role of CI in collaborative development.

## Part 1: Develop Your JavaScript Program

1. Create your main JavaScript file (validatepalindrome.js):

This file should:  
- Accept an array of strings as input.  
- Use a function to determine if each string is a palindrome.  
- Return a list of strings that are palindromes.  
- Print the result using console.log().  
- The palindrome check must ignore case and whitespace.  
  
Example:  
Input: ["racecar", "hello", "Level", "I", "world", "mad am", "12321", "not a palindrome"]  
Output: racecar, Level, I, mad am, 12321

2. Export your function for testing:

* At the bottom of validatepalindrome.js, add:

module.exports = { getPalindromes };

3. Write unit tests using Jest:

- Create a file named test\_validatepalindrome.js.  
- Import your function using:  
 const { getPalindromes } = require('./validatepalindrome');  
- Write at least three unit tests using Jest:  
 - “racecar” should be identified as a palindrome.  
 - “hello” should not be identified as a palindrome.  
 - An empty string or single-character string should be identified as a palindrome.

4. Set up your project with Jest:

In your project folder:  
npm init -y  
npm install --save-dev jest  
  
Then, in your package.json, add the following under 'scripts':  
"scripts": {  
 "test": "jest"  
}

## Part 2: Push to GitLab

1. Create a GitLab repository

- Sign in to gitlab.com.  
- Create a new project:  
 - Group: INFO465  
 - Project Name: simple\_pipeline\_build  
 - Select blank project.

2. Upload your code

From your terminal:  
git clone <your-gitlab-repo-url>  
cd simple\_pipeline\_build  
# Copy your project files here  
git add .  
git commit -m "Initial commit"  
git push -u origin main

## Part 3: Configure CI Pipeline

1. Create a .gitlab-ci.yml file

In GitLab, go to your project and click the ➕ button, then "New file".  
Name the file .gitlab-ci.yml.  
Add the following content:

stages:  
 - test  
 - security  
  
test:  
 stage: test  
 image: node:20  
 script:  
 - npm install  
 - npm test  
  
secret\_detection:  
 stage: security  
 image: docker:latest  
 services:  
 - docker:dind  
 variables:  
 SECURE\_ANALYZERS\_PREFIX: registry.gitlab.com/gitlab-org/security-products/analyzers  
 script:  
 - echo "Running secret detection"  
 artifacts:  
 paths: [gl-secret-detection-report.json]  
  
jshint\_scan:  
 stage: security  
 image: node:20  
 before\_script:  
 - npm install -g jshint  
 script:  
 - jshint validatepalindrome.js

2. Run and monitor your pipeline

Go to CI/CD > Pipelines in your GitLab project. Verify that the pipeline runs and completes successfully.

## Part 4: Submission

Submit the following:

* A screenshot of your pipeline showing a successful run.
* Your JavaScript file (validatepalindrome.js).
* Download and submit the artifacts from your latest pipeline run (each job’s artifacts).
* A 100–150 word analysis of the scan results.
* A 100–150 word reflection on what you learned and how CI improves collaboration.

## Grading Rubric (100 Points)

- JavaScript Program: 20 points – Fully functional and meets spec. No partial credit.  
- Pipeline Setup & Results: 55 points  
 - 15 points – CI jobs defined correctly in .gitlab-ci.yml.  
 - 15 points – Successful pipeline run.  
 - 25 points – Thoughtful analysis of scan results and response to added code.  
- Reflection: 20 points  
 - 10 points – Clarity, grammar, and organization.  
 - 10 points – Insightful connection to CI concepts.  
- Clean Submission: 5 points – Clear formatting and clean artifacts.