

SET-222

Software Operations & Maintenance

Experiment # 03

**Experiment Title**

**Configuring Continuous Integration (CI) Pipelines**

**Assessment of CLO(s): 03**

**Performed on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name:** |  | | |
| **Roll No.** |  | **Group** |  |
| **Semester** |  | **Session** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S. No.** | **Perf. Level**  **Criteria** | **Excellent**  **(2.5)** | **Good**  **(2)** | **Satisfactory**  **(1.5)** | **Needs Improvement**  **(0 ~ 1)** | **Marks Obtained** |
| **1** | Project Execution & Implementation | Fully functional, optimized, and well-structured. | Minor errors, mostly functional. | Some errors, requires guidance. | Major errors, non-functional, or not Performed. |  |
| **2** | Results & Debugging  Or Troubleshooting | Accurate results with effective debugging  Or Troubleshooting. | Mostly correct, some debugging Or Troubleshooting needed. | Partial results, minimal debugging  Or Troubleshooting. | Incorrect results, no debugging Or Troubleshooting, or not attempted. |  |
| **3** | Problem-Solving & Adaptability  (VIVA) | Creative approach, efficiently solves challenges. | Adapts well, minor struggles. | Some adaptability, needs guidance. | Lacks innovation or no innovation, unable to solve problems. |  |
| **4** | Report Quality & Documentation | Clear, structured, with detailed visuals. | Mostly clear, minor gaps. | Some clarity issues, missing details. | Poorly structured, lacks clarity, or not submitted. |  |
| **Total Marks Obtained Out of 10** | | | | | |  |

**Experiment evaluated by**

|  |  |  |  |
| --- | --- | --- | --- |
| **Instructor’s Name** | **Ms. Shagufta Aftab** | | |
| **Date** |  | **Signature** |  |

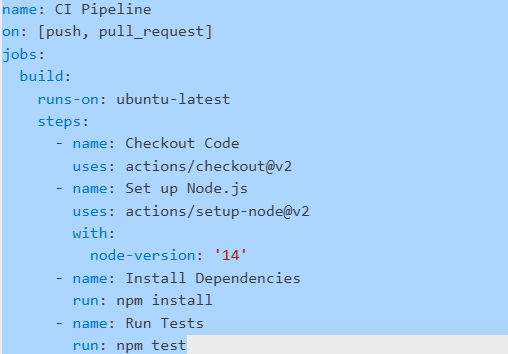
## Copyright © Department of Engineering & Technology – UIT University Karachi

**Objective:**

* Understand the concept of CI and its benefits.
* Set up a CI pipeline using a CI/CD tool.
* Automate code testing and integration.
* Analyze pipeline execution results and troubleshoot errors.

### **Exercise 1: Setting Up a CI Pipeline with GitHub Actions**

1. Create a repository on GitHub.
2. Navigate to the **Actions** tab and set up a new workflow.
3. Create a .github/workflows/ci-pipeline.yml file with the following example content:



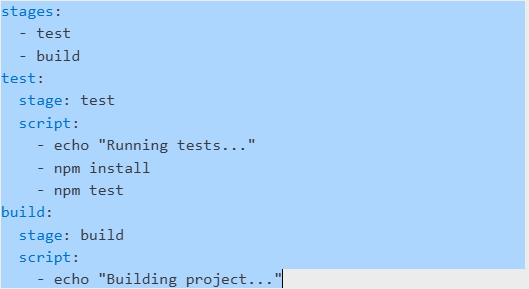
4. Commit and push the file to the repository.

5. Observe the pipeline execution under the **Actions** tab.

**Exercise 2: Configuring a CI Pipeline in Jenkins**

1. Install Jenkins and required plugins (Git, Pipeline, NodeJS, etc.).
2. Create a new **Freestyle Project** or **Pipeline Project**.
3. Configure the Git repository URL and branch.
4. Add build steps to:
   * Clone the repository.
   * Install dependencies.
   * Run tests.
5. Save and trigger the pipeline to test execution.

**Exercise 3: Implementing CI with GitLab CI/CD**

1. Create a .gitlab-ci.yml file in the root of your GitLab repository.
2. Add the following sample configuration:
3. Push the file to GitLab and check the **CI/CD > Pipelines** section.

**Troubleshooting Tips:**

* Ensure your repository contains a valid project structure.
* Check for syntax errors in configuration files.
* Review CI logs to identify and fix pipeline failures.

**Conclusion:** By completing this lab, you have successfully configured CI pipelines using different tools. You now have a foundation for automating code testing and deployment processes in real-world software projects.