

SET-222

Software Operations & Maintenance

Experiment # 04

**Experiment Title**



**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** |  |

**Objective:**

* To understand and implement automated deployment pipelines for continuous integration and deployment (CI/CD) in software development.

**Theory:**  
Automating deployment pipelines is crucial in modern DevOps practices. It involves setting up Continuous Integration (CI) and Continuous Deployment (CD) processes to ensure code is built, tested, and deployed automatically. Tools like Jenkins, GitHub Actions, GitLab CI/CD, and Docker help streamline the deployment workflow, reducing manual errors and accelerating software releases.

**Key Concepts and Commands to be Covered:**

**Version Control with Git**

1. git clone <repo\_url> – Clone a repository.
2. git commit -m "message" – Commit changes.
3. ain – Push code to the main branch.

**CI/CD with Jenkins (or GitHub Actions/GitLab CI/CD)**

1. **Jenkins:**
   * sudo systemctl start jenkins – Start Jenkins service.
   * Configure a Jenkins pipeline using a Jenkinsfile.
   * Install necessary plugins (e.g., Git, Docker, Build tools).
2. **GitHub Actions/GitLab CI/CD:**
   * Define a .github/workflows/main.yml file for GitHub Actions.
   * Define a .gitlab-ci.yml file for GitLab CI/CD.

**Containerization with Docker (Optional)**

1. docker build -t app-image . – Build a Docker image.
2. docker run -d -p 8080:80 app-image – Run the application in a container.

**Procedure:**

**Part 1: Setting Up a CI/CD Pipeline**

1. Clone a Git repository using git clone <repo\_url>.
2. Create a sample application (Python/Node.js/Java).
3. Configure a CI/CD pipeline in Jenkins, GitHub Actions, or GitLab CI/CD.
4. Define build, test, and deployment stages in Jenkinsfile, .github/workflows/main.yml, or .gitlab-ci.yml.

**Part 2: Automating Deployment**

1. Push code changes to the repository.
2. Observe the automated build and test execution.
3. Deploy the application automatically to a server or container.
4. Verify deployment by accessing the application via URL or container logs.

**Observations:**

* Note down build and test results.
* Identify any pipeline failures and troubleshoot errors.
* Verify successful deployment using logs or application URLs.

**Conclusion:**  
Automating deployment pipelines enhances software development by enabling rapid, reliable, and repeatable deployments. This experiment demonstrates configuring and automating CI/CD workflows using industry-standard tools.

**References:**

* Jenkins Documentation (https://www.jenkins.io/doc/)
* GitHub Actions (https://docs.github.com/en/actions)
* GitLab CI/CD (https://docs.gitlab.com/ee/ci/)