THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

End-to-End CI/CD Pipeline with Jenkins, GitLab, and Docker

Introduction & Project Overview:

This project focuses on building a complete Continuous Integration and Continuous Deployment (CI/CD) pipeline using Jenkins, GitLab, and Docker to automate the process of building, testing, and deploying a web application. The goal was to design and execute a real-world DevOps workflow that ensures reliability, repeatability, and visibility in the software delivery process.

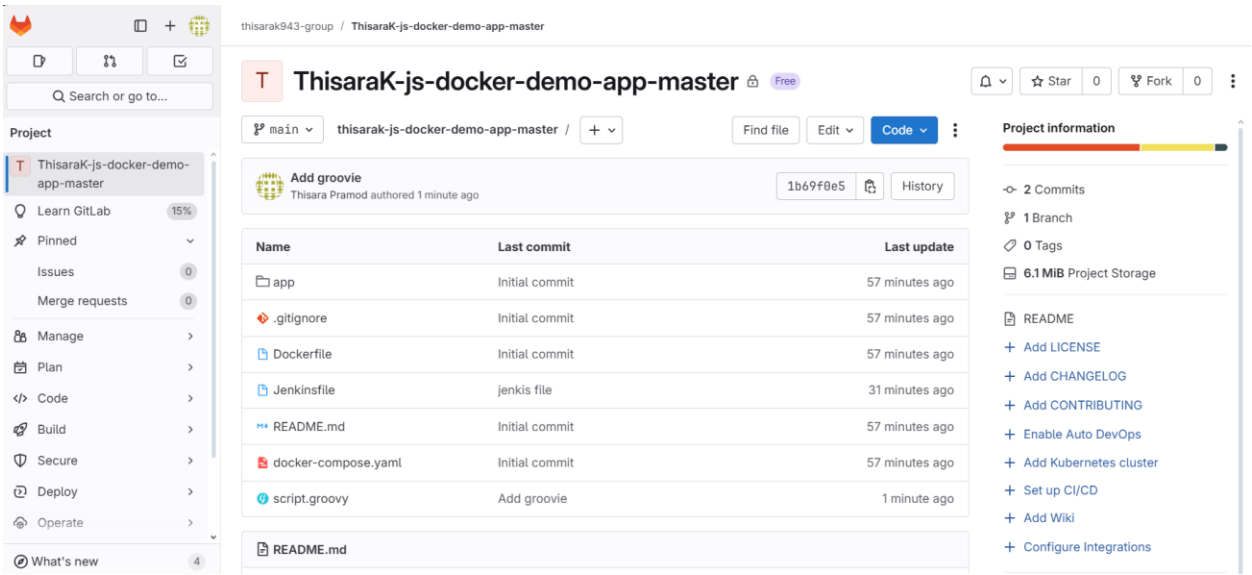
The process began with initializing a GitLab repository where the source code, Dockerfile, and Jenkinsfile were committed. A Dockerfile was created to containerize the application, enabling consistent deployment across environments. On the Jenkins side, a fresh installation was set up, followed by installing essential plugins like Git, Docker Pipeline, GitLab integration, and Pipeline Stage View.

To streamline automation, a **Multibranch Pipeline** job was created in Jenkins, configured to scan branches from the connected GitLab repository. Authentication was securely handled using a **Personal Access Token (PAT)**, which was added as a credential in Jenkins to fetch code and configuration from GitLab.

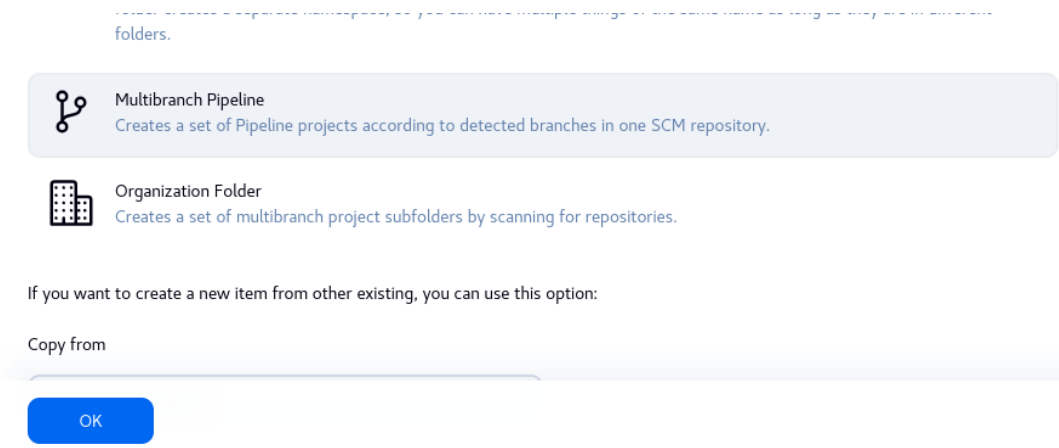
A declarative Jenkinsfile was used to define the CI/CD stages: init, build, test, and deploy. Additionally, an external script.groovy file was used to modularize and simplify the pipeline logic. Parameters were added to allow flexibility, including selectable versions and a toggle for executing tests. Once everything was configured, the pipeline was triggered, and Jenkins automatically executed all stages while displaying each step using the Stage View UI.

This hands-on project provided deep insight into configuring CI/CD pipelines using industry-standard tools and applying best practices in automation, containerization, and continuous delivery.

- **GitLab repo**



- **Create a new build (Multibranch pipeline)**

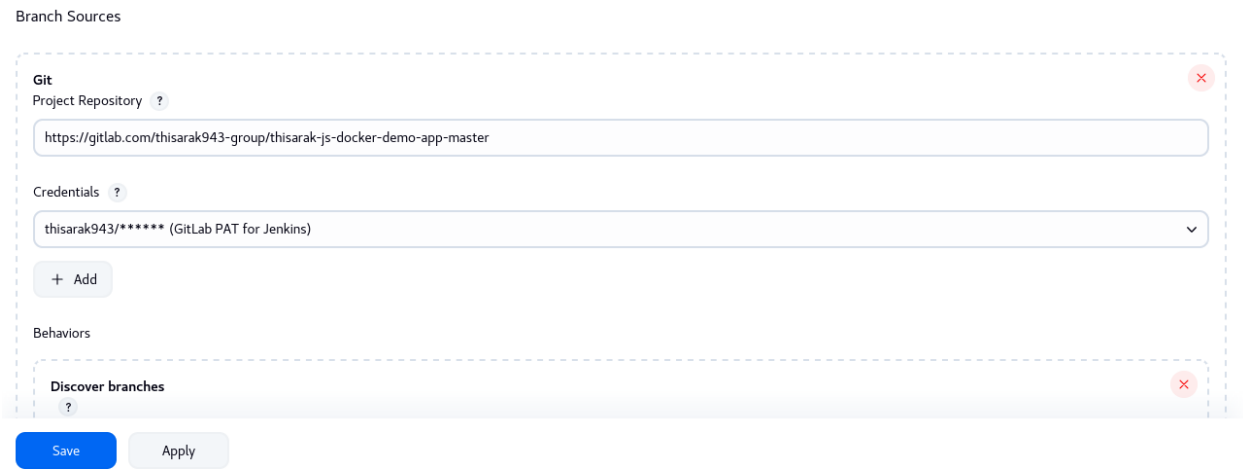


THISARA KANDAGE

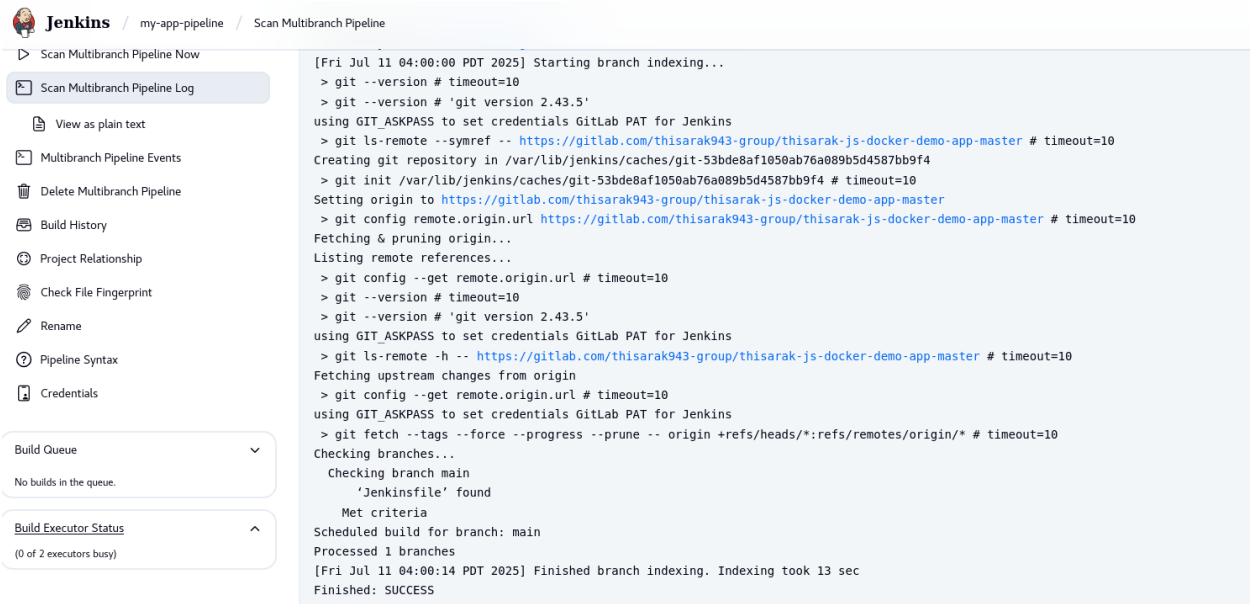
UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

- **Add Branch sources (Access token credentials already created)**



- **Pipeline logs**



- **Jenkinsfile**

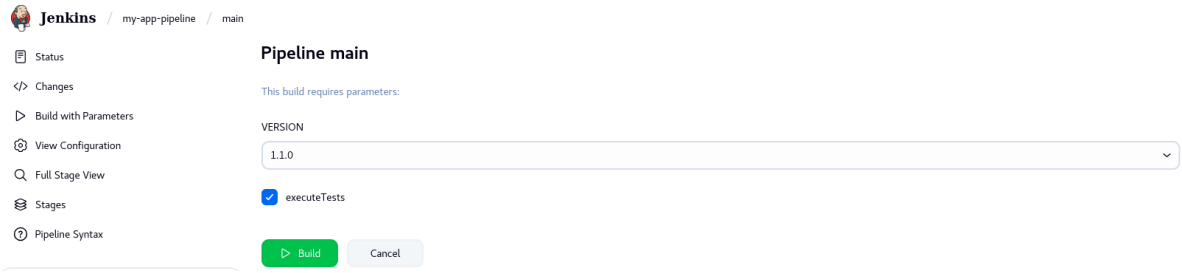
```
pipeline {
  agent any
  parameters {
    choice(name: 'VERSION', choices: ['1.1.0', '1.2.0', '1.3.0'], description: '')
    booleanParam(name: 'executeTests', defaultValue: true, description: '')
  }
  stages {
    stage("init") {
      steps {
        script {
          gv = load "script.groovy"
        }
      }
    }
    stage("build") {
      steps {
        script {
          gv.buildApp()
        }
      }
    }
    stage("test") {
      when {
        expression {
```

```
            params.executeTests
        }
    }
    steps {
        script {
            gv.testApp()
        }
    }
}
stage("deploy") {
    steps {
        script {
            gv.deployApp()
        }
    }
}
}
}
```

my-app-pipeline 30 min log N/A 13 sec

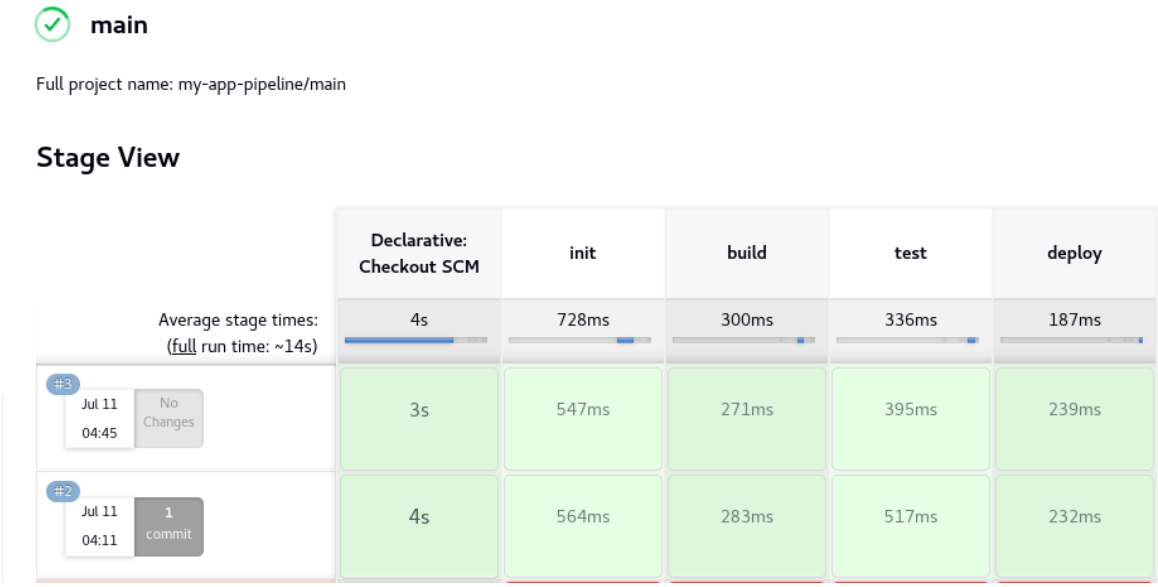
Icon: S M L

- **Build with parameters**



- **After building parameters**

All environmental variables in Jenkins file are available in the groovy script



THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

Summary of Key Learnings

- **Version Control & Repository Structuring**
  Learned how to properly structure and manage a GitLab repository to support CI/CD processes. This included organizing application code, configuration files (Dockerfile, Jenkinsfile, script.groovy), and preparing the repository for automatic integration with Jenkins. Gained experience in branch-based development and remote repository linking.
- **Jenkins Configuration & Plugin Management**
  Installed and configured Jenkins on a Linux environment. Set up essential plugins such as Git, Docker Pipeline, GitLab Authentication, and Pipeline Stage View. Learned to manage Jenkins system settings, global tool configurations, and how to secure credentials for interacting with external tools and repositories.
- **Multibranch Pipeline Setup and GitLab Integration**
  Configured a Multibranch Pipeline in Jenkins that dynamically detects and builds all branches in the GitLab repository. Used a securely stored **GitLab Personal Access Token (PAT)** as credentials, allowing Jenkins to clone the repository, trigger builds, and continuously monitor for changes in code or branch structure.
- **Pipeline Development Using Declarative Jenkinsfile**
  Designed a structured and modular Jenkinsfile using declarative pipeline syntax. Defined core CI/CD stages — init, build, test, and deploy. Utilized a separate Groovy file (script.groovy) to encapsulate logic for better readability and reusability, enhancing long-term maintainability of the pipeline code.
- **Parameterization and Conditional Logic in Pipelines**
  Implemented build parameters such as version selection and toggling test execution. Gained an understanding of how Jenkins pipelines can respond dynamically to user input and how to control conditional logic using when blocks and parameter-based decision-making.
- **Pipeline Execution Visualization with Stage View**
  Successfully visualized the entire CI/CD workflow using Jenkins Stage View, enabling real-time insights into stage duration, execution results, and pipeline flow. This helped monitor performance, quickly identify bottlenecks or failures, and enhance the overall observability of the automation process.
- **End-to-End CI/CD Automation Experience**
  Executed the full CI/CD cycle — from committing code to the GitLab repository to automated building, testing, and deploying the application using Jenkins. Developed a deep, practical understanding of how modern DevOps tools integrate to support continuous software delivery, rapid feedback, and process automation.