1. Project Overview

The project is an arithmetic calculator application built using Node.js and Express. It provides functionalities to perform basic arithmetic operations such as addition, subtraction, multiplication, and division. The application includes a backend server implemented with Express.js and a set of API endpoints to handle arithmetic operations. Additionally, automated tests are implemented to ensure the correctness of arithmetic functions. The project leverages a stack of technologies to facilitate its functionality. It is primarily built using Node.js and Express.js, which provide the foundation for developing the backend server and handling HTTP requests. Additionally, Jest, a widely-used testing framework, is employed to automate the testing process and ensure the reliability of the application's arithmetic functionalities.

2. Pipeline Stages

Stage	Title	Description	Outcome
1	Ensure Docker Daemon is Running:	This stage checks if the Docker daemon is running on the Jenkins server.	If Docker daemon is running, the pipeline proceeds. Otherwise, it prompts to start Docker Desktop.
2	Clone the Code	This stage clones the code repository from GitHub.	Code is successfully cloned from the repository.
3	Initialize	This stage initializes Docker tools.	Docker tools are configured for use in subsequent stages.
4	Building Stage: Building Docker Image and Run Container	This stage builds a Docker image and runs a container based on the image.	Docker image is built and container is started successfully.
5	Testing Stage: Test the Code using Jest Testing Units	This stage executes tests inside the Docker container using Jest testing units.	Code is tested successfully.
6	Deploying Stage: Using Docker Compose	This stage deploys the code to a test environment using Docker Compose.	Code is deployed to the test environment successfully.

3. Testing framework

The project utilizes Jest as its testing framework to automate the testing process and validate the correctness of its arithmetic functionalities. Through Jest, a suite of tests is executed, covering various scenarios for each arithmetic operation:

- Addition: Ensures the accuracy of the addition function.
- Subtraction: Validates the correctness of the subtraction function.
- Multiplication: Verifies the precision of the multiplication function.
- Division: Ensures the reliability of the division function, handling edge cases appropriately.

4. Deployment Tool Used

For managing the deployment and orchestrating the test environment, Docker Compose serves as the primary tool. Leveraging Docker Compose, the project seamlessly deploys the application into a Docker container environment. This ensures consistency across different testing environments and facilitates efficient integration and deployment processes.

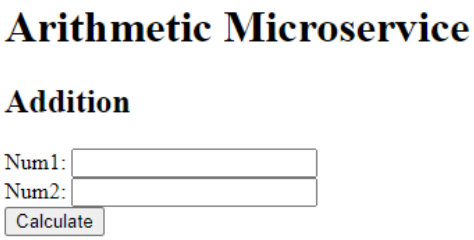
Conclusion

The implementation of the Jenkins pipeline ensures the automation of the build, test, and deployment processes for the arithmetic calculator application. By leveraging Docker and Docker Compose, the application can be easily deployed to different environments, facilitating continuous integration and delivery practices. The use of Jest for automated testing ensures the reliability and correctness of the application's functionalities.

Link to GitHub

https://github.com/HadiRastin/Task6_2.git

Screenshots




```
PASS ./app.test.js
  Arithmetic Functions
    √ Addition (7 ms)
    √ Subtraction (2 ms)
    √ Multiplication (1 ms)
    √ Division (6 ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        1.46 s, estimated 2 s
```

Jenkinsfile

```
pipeline {
    agent any

    environment {
        DOCKER_IMAGE = 'pro_prac'
    }

    stages {
        stage('Ensure Docker Daemon is Running') {
            steps {
                script {
                    def dockerStatus = sh(script: 'docker info', returnStatus: true)
                    if (dockerStatus != 0) {
                        error 'Docker daemon is not running. Please start Docker Desktop on the Jenkins
server.'
                    } else {
                        echo 'Docker daemon is running.'
                    }
                }
            }
        }

        stage('Clone the code') {
            steps {
                script {
                    git branch: 'main', url: 'https://github.com/HadiRastin/Task6_2'
                    echo 'Code cloned successfully'
                }
            }
        }

        stage('Initialize') {
            steps {
                script {
```

```
                    def dockerHome = tool 'myDocker'
                    env.PATH = "${dockerHome}/bin:${env.PATH}"
                }
            }
        }

        stage('Building stage: Building Docker image and run container') {
            steps {
                script {
                    // Build Docker image
                    sh "docker build -t ${DOCKER_IMAGE}:v1 ."
                    // Run Docker container
                    sh "docker run -d -p 8700:3040 --name my-container ${DOCKER_IMAGE}:v1"
                    echo 'Docker image built and container started successfully'
                }
            }
        }

        stage('Testing stage: Test the code using jest testing units') {
            steps {
                script {
                    // Execute tests inside Docker container
                    sh "docker exec my-container /bin/bash -c 'cd /app && npm test'"
                    echo 'Code tested successfully'
                }
            }
        }

        stage('Deploying stage: Using Docker Compose') {
            steps {
                script {
                    // Use Docker Compose to deploy to a test environment
                    sh 'docker-compose -f Docker-compose.yaml up -d'
                    echo 'Code deployed to test environment successfully'
                }
            }
        }
    }

    post {
        always {
            // Clean up Docker containers
            script {
                sh 'docker stop my-container'
                sh 'docker rm my-container'
                echo 'Clean up completed.'
            }
        }
    }
}
```