THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

Streamlining Application Deployment with Jenkins and Docker: CI/CD Pipeline Implementation

In today's fast-paced software development environment, organizations are increasingly relying on automated solutions to streamline the deployment process. DevOps practices, which emphasize collaboration and automation between development and operations teams, are essential for enabling continuous software delivery. This project focuses on the design and implementation of a Continuous Integration (CI) and Continuous Deployment (CD) pipeline using Jenkins and Docker to automate the software development lifecycle.

The CI/CD pipeline automates key processes such as code integration, testing, and deployment, allowing for faster feedback and more reliable deployments. By using Jenkins, an open-source automation tool, the project ensures that all code changes are automatically fetched, built, tested, and deployed. Docker plays a crucial role in containerizing the application, ensuring consistency and scalability across different environments, from development to production.

The goal of this project is to provide a fully automated, scalable, and efficient pipeline that eliminates manual intervention in the deployment process. The integration of Jenkins and Docker facilitates a smooth, error-free transition of code from development through testing and to production, ensuring that all code changes are continuously deployed in an optimized environment.

Throughout this project, I have designed the pipeline, configured Jenkins with necessary plugins, and set up Docker containers to facilitate smooth deployment workflows. This setup not only improves productivity by reducing manual steps but also enhances the overall quality of software releases by automating error detection and resolution.

THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

Checking Node Setup and test file work or not?

```
PS E:\Youtube Projects\devopstest2\GitHub-Docker-and-Jenkins-CI-CD-Pipeline\nodeapp> npm start

> node-app@0.0.1 start
> node index.js

App listening on port 3000!
^C^CTerminate batch job (Y/N)? y
PS E:\Youtube Projects\devopstest2\GitHub-Docker-and-Jenkins-CI-CD-Pipeline\nodeapp> cd test
PS E:\Youtube Projects\devopstest2\GitHub-Docker-and-Jenkins-CI-CD-Pipeline\nodeapp\test> npm test

> node-app@0.0.1 test
> mocha ./test/test.js

App listening on port 3000!


  GET /will
    ✓ respond with hello world


  1 passing (40ms)
```

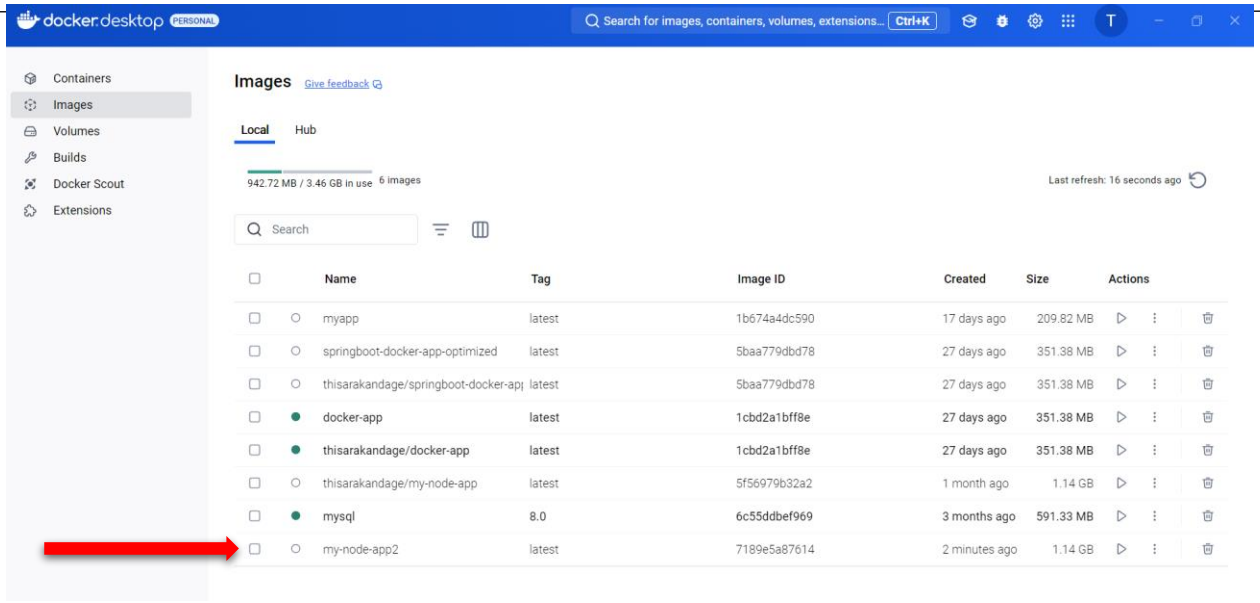
Create a docker image

```
PS E:\Youtube Projects\devopstest2\GitHub-Docker-and-Jenkins-CI-CD-Pipeline> docker build -t my-node-app2 .
[+] Building 157.9s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 609B
 => [internal] load metadata for docker.io/library/node:latest
 => [auth] library/node:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/4] FROM docker.io/library/node:latest@sha256:1745a99b66da41b5ccd6f7be3810f74ddab16eb4579de10de378adb50d2e6e6f
 => => resolve docker.io/library/node:latest@sha256:1745a99b66da41b5ccd6f7be3810f74ddab16eb4579de10de378adb50d2e6e6f
 => => sha256:1745a99b66da41b5ccd6f7be3810f74ddab16eb4579de10de378adb50d2e6e6f 6.41kB / 6.41kB
 => => sha256:b536d17a0f8a33c85b1ad262f50d3ffd82446c46c2240ba7657b9e66f3817d8d 6.39kB / 6.39kB
 => => sha256:54c7be425079efba0003054ee884bf72f1ffebca733bedd6f077d2809ee9aa6f 23.87MB / 23.87MB
 => => sha256:7aa8176e6d893aff4b57b2c22574ec2afadff4673b8e0954e275244bfd3d7bc1 64.39MB / 64.39MB
 => => sha256:8b2b79dcd4164985acb914c82222c2a1e8277399a2b02c681109a04327baaf5b 2.49kB / 2.49kB
 => => sha256:0a96bdb8280554b560ffee0f2e5f9843dc7b625f28192021ee103ecbcc2d629b 48.50MB / 48.50MB
 => => sha256:1523f4b3f5602bf41caf8d8e649536ac0b3e56984c81a9f518fb20c6516ca075 211.31MB / 211.31MB
```

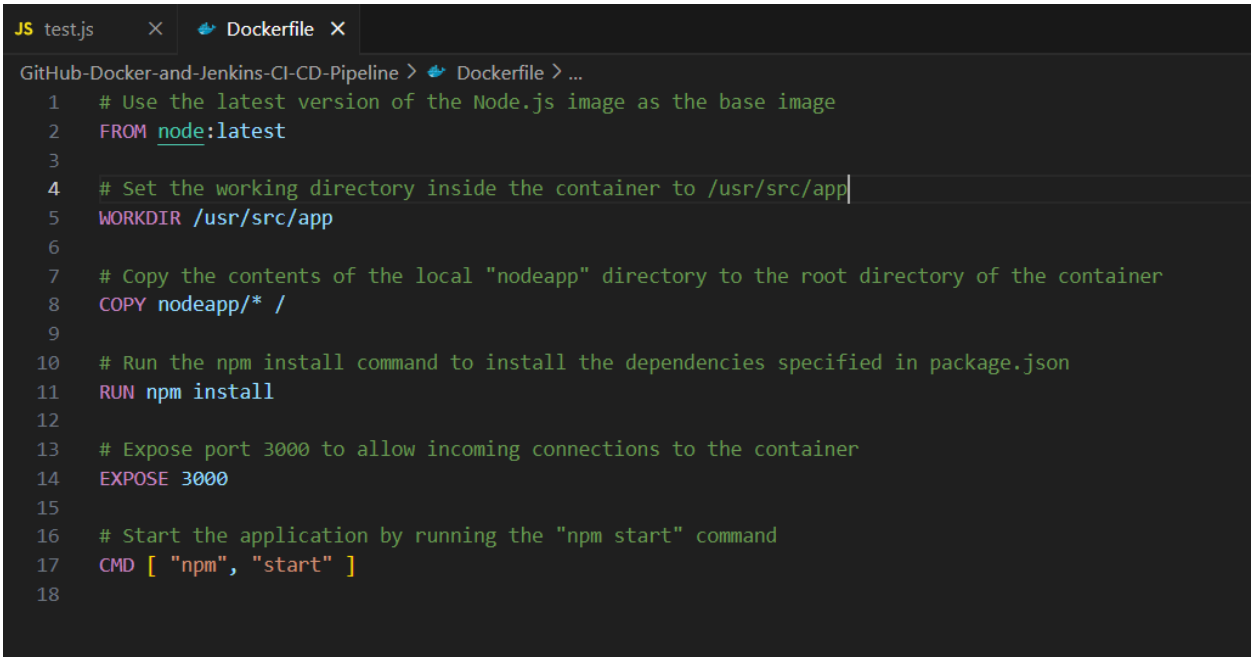
THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website



Docker file



THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

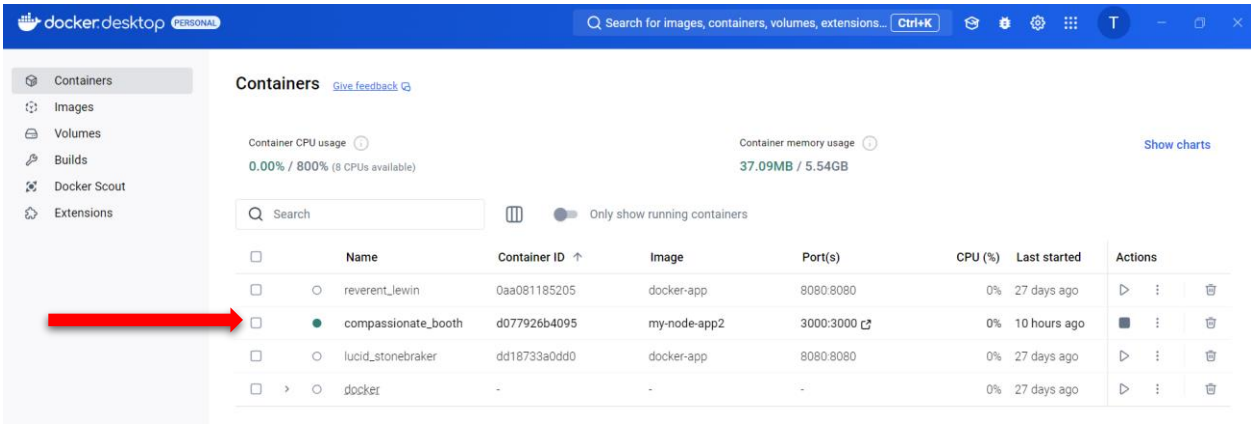
Create docker container using docker image

```
PS E:\Youtube Projects\devopstest2\GitHub-Docker-and-Jenkins-CI-CD-Pipeline> docker run -p 3000:3000 my-node-app2

> node-app@0.0.1 start
> node index.js

App listening on port 3000!
```

Docker Hub

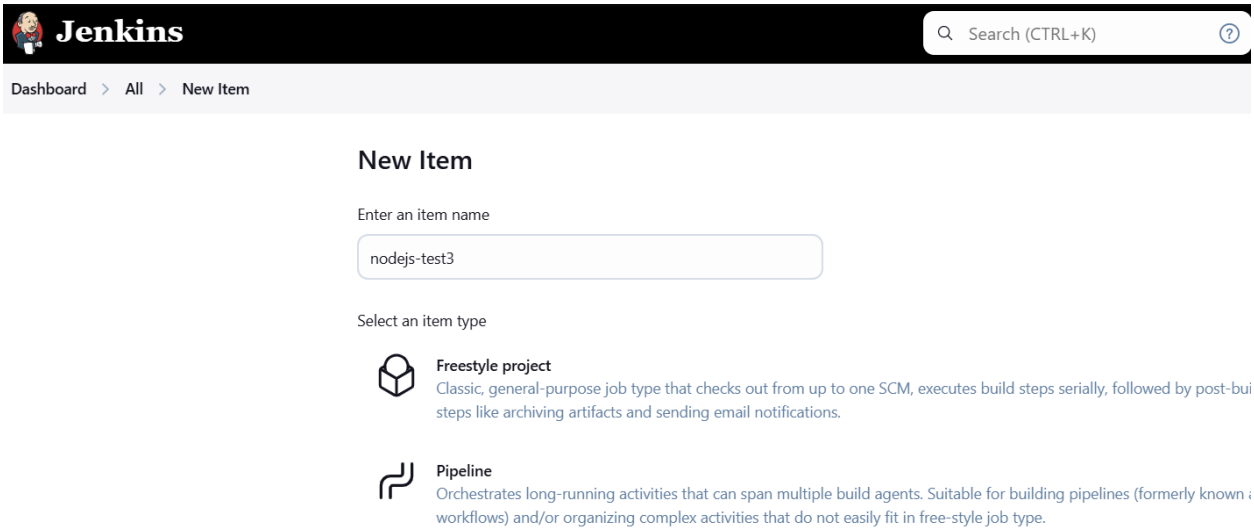


THISARA KANDAGE

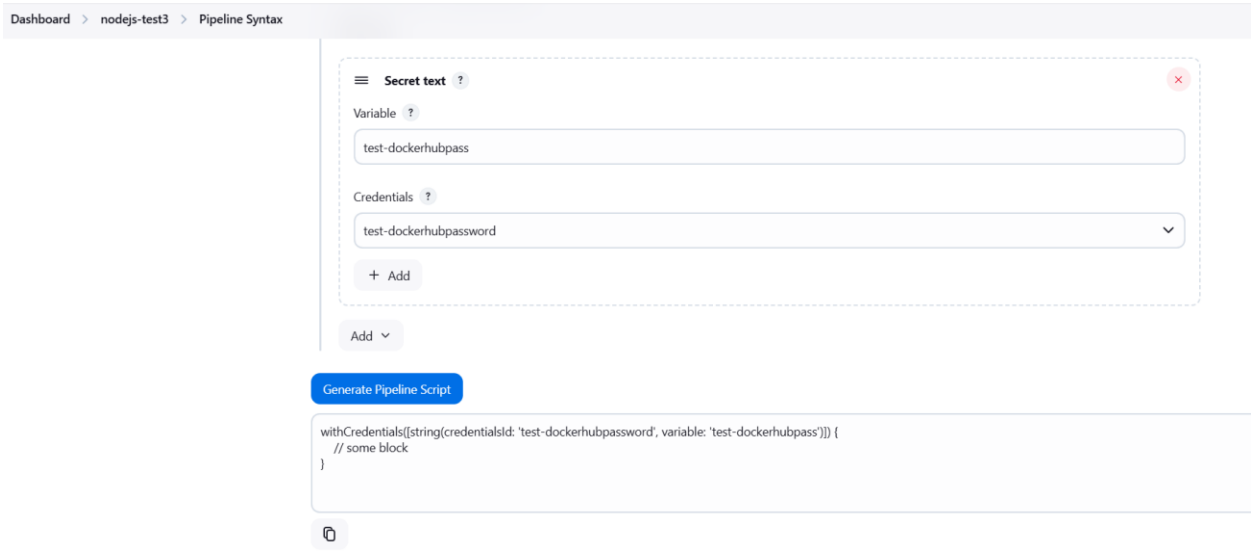
UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

Create Jenkins new item



Generate pipeline script



THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

Jenkins file/Pipeline Script

```
pipeline {
    agent any

    stages {
        stage('SCM Checkout') {
            steps {
                retry(3) {
                    // Pull the latest code from the GitHub repository
                    git branch: 'main', url: 'https://github.com/Thisarak943/DockerJenkinsProject.git'
                }
            }
        }

        stage('Build Docker Image') {
            steps {
                // List the directory contents to verify the Dockerfile is present
                bat 'dir'

                // Use the correct format for Windows environment variables
                bat '''
                    echo Building Docker image...
                    docker build -t thisarakandage/nodeapp2:%BUILD_NUMBER% .
                '''
            }
        }
```

```
        stage('Login to Docker Hub') {
            steps {
                withCredentials([string(credentialsId: 'test-dockerhubpassword', variable: 'test-dockerhubpass')]) {
                    // Login to Docker Hub using Jenkins credentials
                    bat 'docker login -u thisarakandage -p %test-dockerhubpass%'
                }
            }
        }

        stage('Push Image') {
            steps {
                // Push the image to Docker Hub with a unique tag based on build number
                bat 'docker push thisarakandage/nodeapp2:%BUILD_NUMBER%'
            }
        }
    }

    post {
        always {
            // Logout from Docker Hub to ensure security
            bat 'docker logout'
        }
    }
}
```

THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

Conclusion

1. **Challenges Faced:**
    - **Configuring Jenkins and Docker integration.**
    - **Resolving misconfigurations in pipeline scripts.**
    - **Handling failed authentication during Docker Hub push.**
2. **Solutions Implemented:**
    - **Set up Jenkins and Docker integration, following best practices.**
    - **Used retry mechanisms and corrected pipeline scripts.**
    - **Configured secure authentication for Docker Hub login.**
3. **Key Learnings:**
    - **Gained practical experience with Jenkins, Docker, and CI/CD.**
    - **Improved troubleshooting skills in build and deployment stages.**
    - **Learned containerization best practices for consistent deployments.**
4. **Impact on Knowledge:**
    - **Enhanced proficiency in Jenkins and Docker.**
    - **Strengthened problem-solving abilities in DevOps workflows.**
    - **Acquired hands-on experience in automating deployment processes.**