x

|  |  |
| --- | --- |
| **Ex. No: 11** | **Applying CI/CD Principles for Node JS application Using Jenkins, Git, using Docker Containers** |
| **24.10.2024** |

**Aim:**

Develop a **REST API** using Node.js. The API should provide an endpoint /status that returns a simple JSON response with a status message.

Containerize the API using Docker and automate the entire CI/CD pipeline using Jenkins.

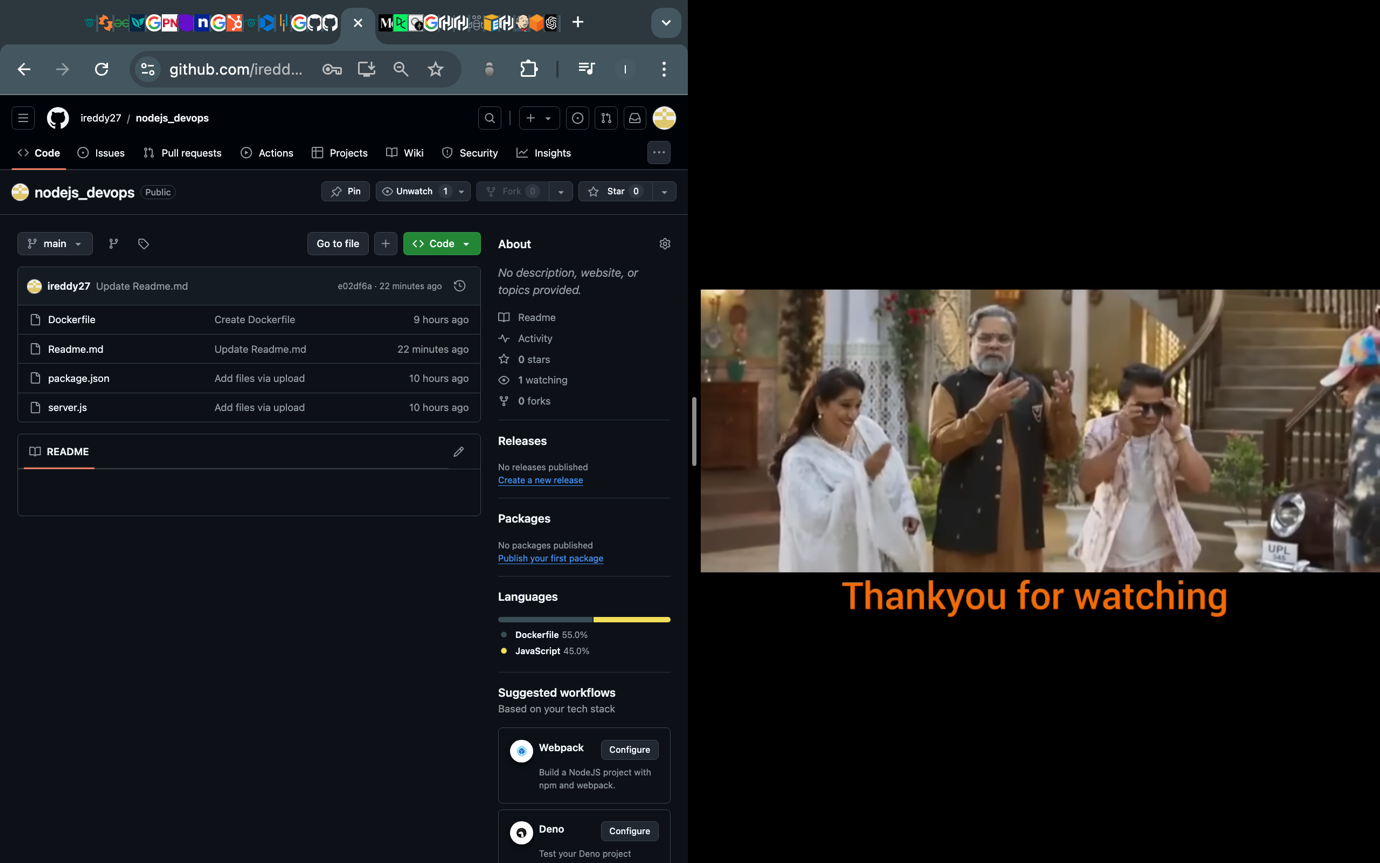
**PROCEDURE:**

**Step 1: Set Up the Web Application and Git Repository**

Create a simple web application or use an existing one. Ensure it can be

hosted in a Docker container.

Initialise a Git repository for your web application and push it to GitHub.



**Step 2: Install and Configure Jenkins**

Install Jenkins on your computer or server following the instructions for your operating system (https://www.jenkins.io/download/).

Open Jenkins in your web browser (usually at http://localhost:8080) and

complete the initial setup, including setting up an admin user and installing

necessary plugins.

Configure Jenkins to work with Git by setting up Git credentials in the Jenkins

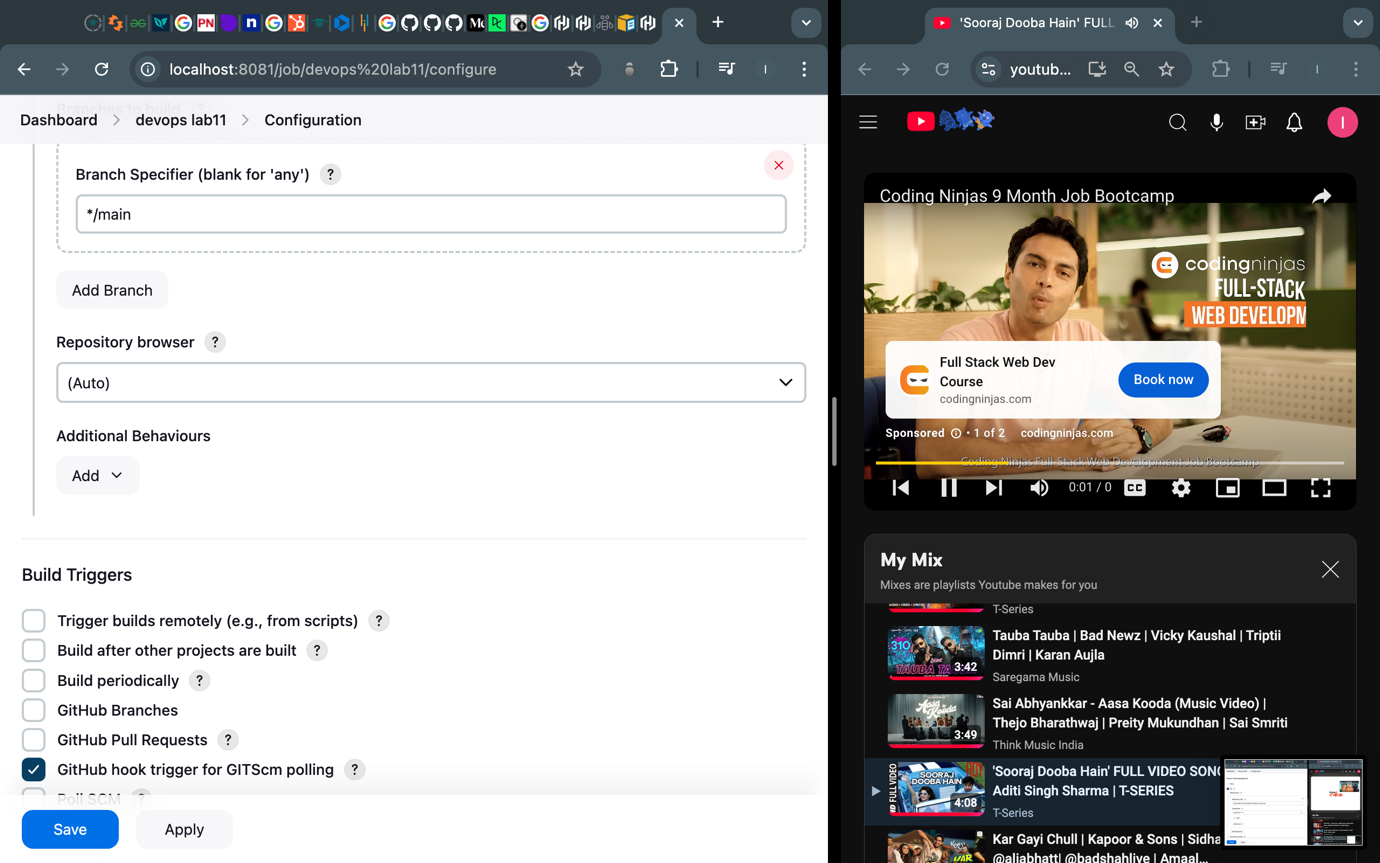
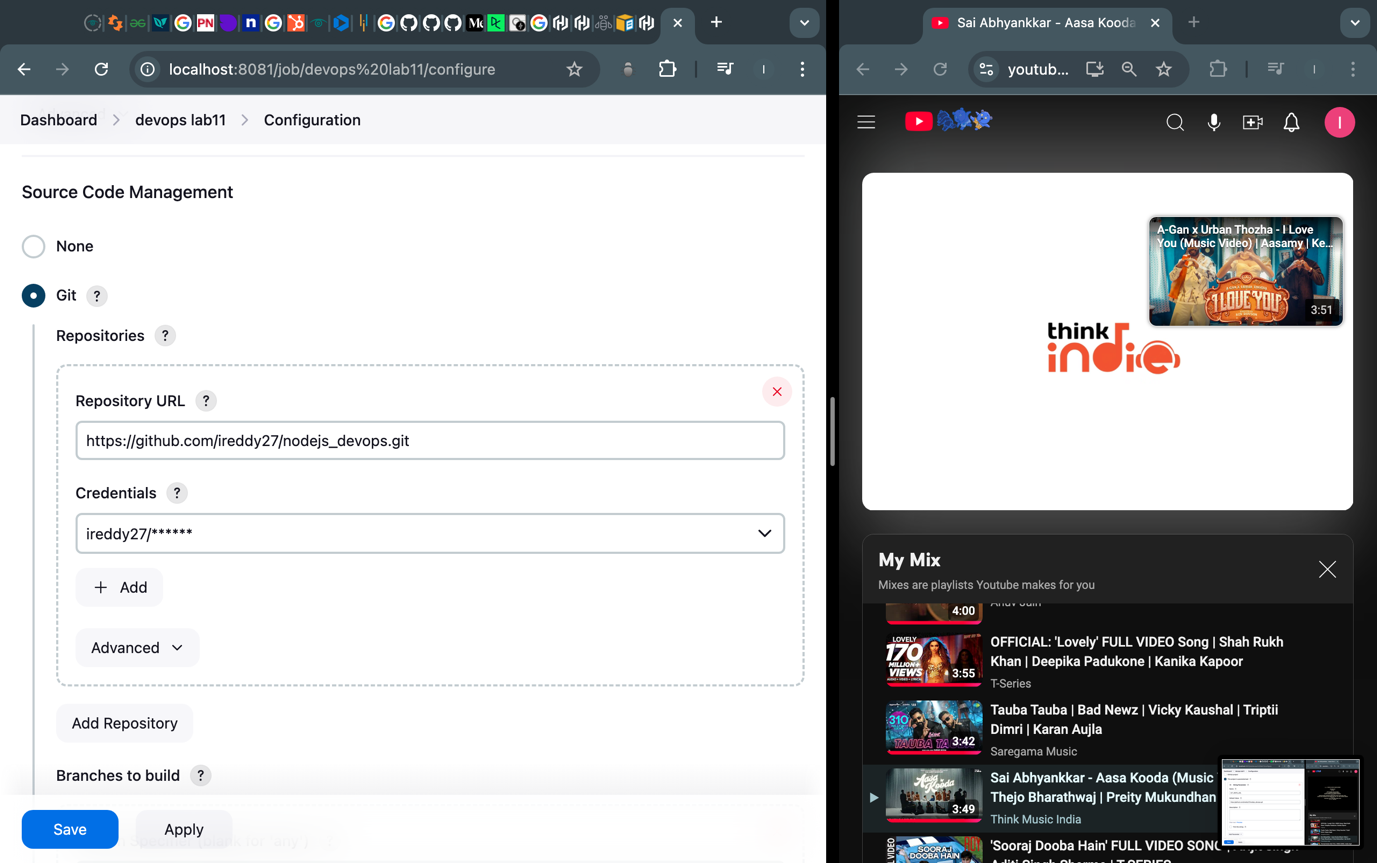
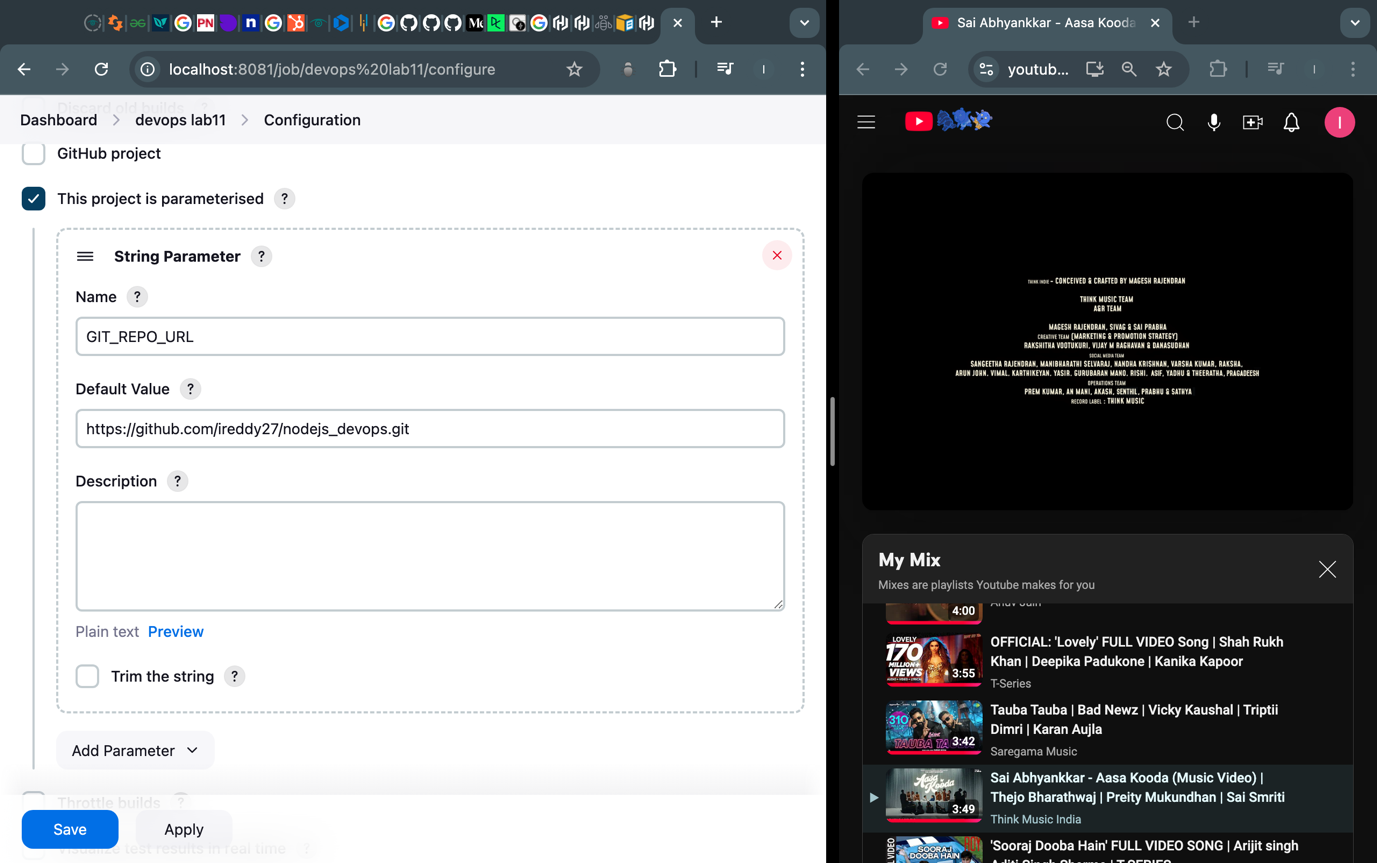
Credential Manager.

A screenshot of a computer

Description automatically generated

**Step 3: Create a Jenkins Job**

* Create a new Jenkins job using the "Freestyle project" type.
* In the job configuration, specify a name for your job and choose "This project is parameterized."
* Add a "String Parameter" named GIT\_REPO\_URL and set its default value to your Git repository URL.
* Set Branches to build -> Branch Specifier to the working Git branch (ex\*/master)
* In the job configuration, go to the "Build Triggers" section and select the
* "GitHub hook trigger for GITScm polling" option. This enables Jenkins to
* listen for GitHub webhook triggers.



**Step 4: Configure Build Steps**

In the job configuration, go to the "Build" section.

Add build steps to execute Docker commands for building and deploying the

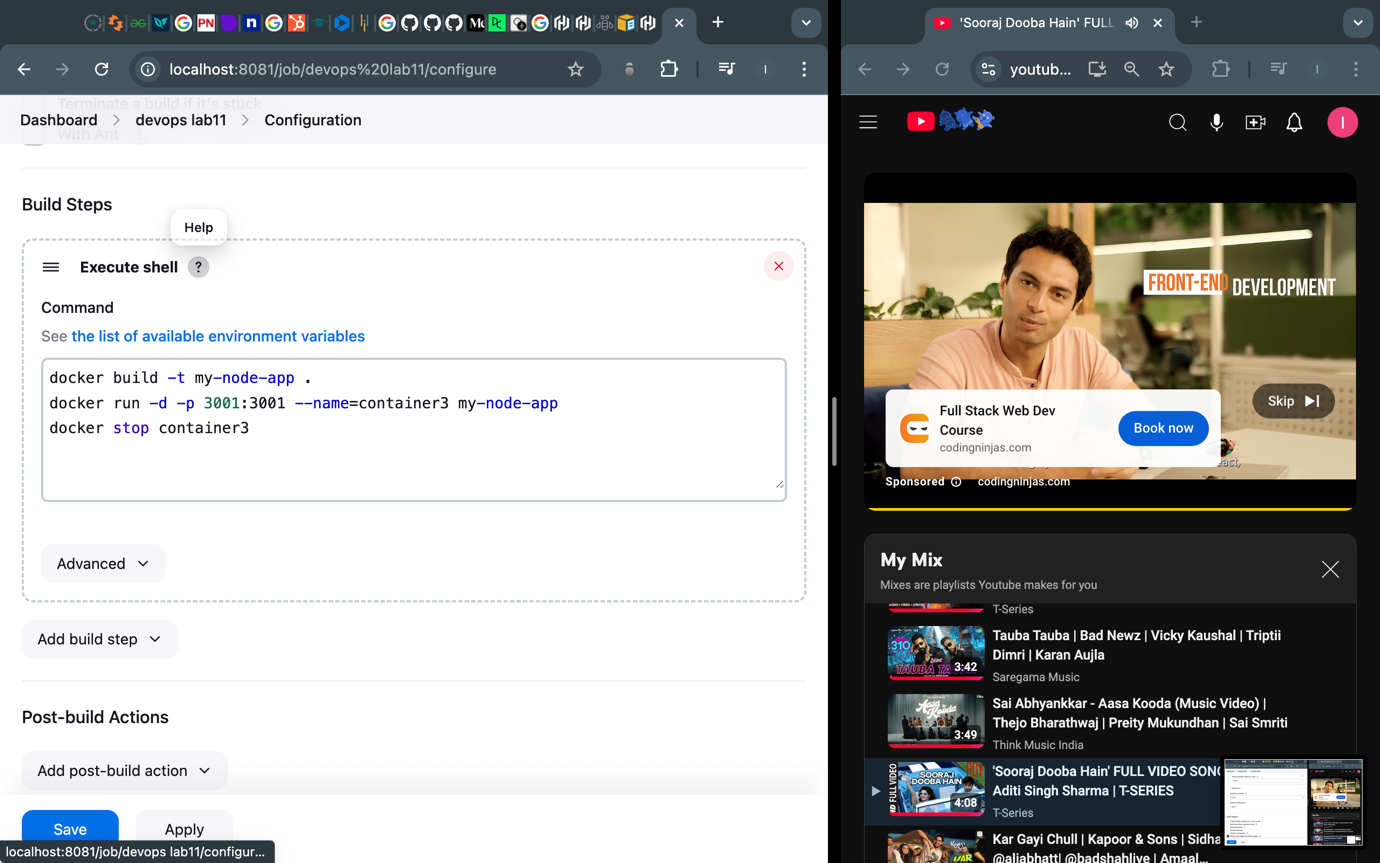
containerized web application.

**# Build a new Docker image**

docker build -t my-node-api .

**# Run the Docker container**

docker run -d -p 3001:3000 –name=container3 my-node-app



**Step 5: Set Up a GitHub Webhook**

In your GitHub repository, navigate to "Settings" and then "Webhooks."

[Create a new webhook, and configure it to send a payload to the Jenkins

webhook URL (usually http://jenkins-server/github-webhook/). Set the content

type to "application/json."

A screenshot of a computer

Description automatically generated

**Step 6: Trigger the CI/CD Pipeline**

Push changes to your GitHub repository. The webhook will trigger the Jenkins

job automatically, executing the build and deployment steps defined in the job

configuration.

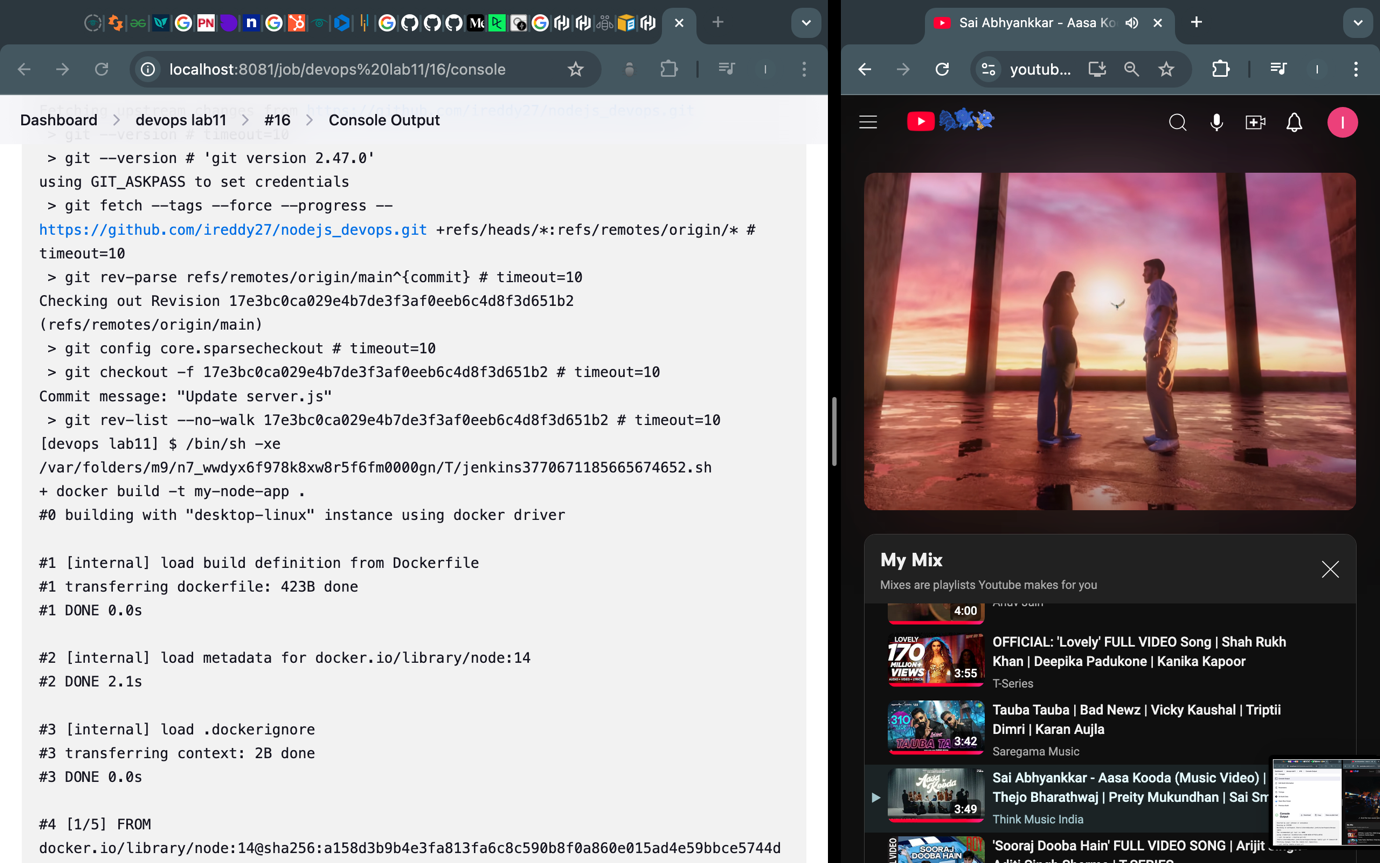
Monitor the Jenkins job's progress in the Jenkins web interface.

A screenshot of a computer

Description automatically generated

Here the Readme.md file is edited

Screens screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**Step 7: Verify the Deployment**

Access your web application by opening a web browser and navigating to

http://localhost:3001 (or the appropriate URL if hosted elsewhere).

A screenshot of a computer

Description automatically generated

**Result:**

Therefore, we've successfully applied CI/CD Principles for NodeJS application Using Jenkins, Gits, using Docker Containers