CI/CD Pipeline for Node.js Application Using Jenkins, Git, and Docker

This document outlines the process of setting up a CI/CD pipeline for a Node.js REST API using Jenkins, Git, and Docker. The goal is to automate the build, test, and deployment stages, ensuring continuous integration and delivery.

Project Overview

We will develop a simple Node.js REST API with an endpoint `/status` that returns a JSON response. The application will be containerized with Docker, and Jenkins will automate the CI/CD process.

Prerequisites

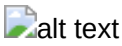
- **Node.js** and **npm** installed locally
- **Git** repository (e.g., on GitHub or GitLab) to store the project code
- **Jenkins** installed and configured with access to Docker
- **Docker** installed and running on the Jenkins server

Steps

1. Set Up the Node.js Application

1. **Initialize the Node.js project:**

```
mkdir nodejs-api
cd nodejs-api
npm init -y
```



2. **Install Express and create the main application file app.js:**

```
npm install express
```

Contents of app.js

```
const express = require('express');
const app = express();
const PORT = process.env.PORT || 3000;

app.get('/status', (req, res) => {
  res.json({ status: "API is running" });
});
```

```
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

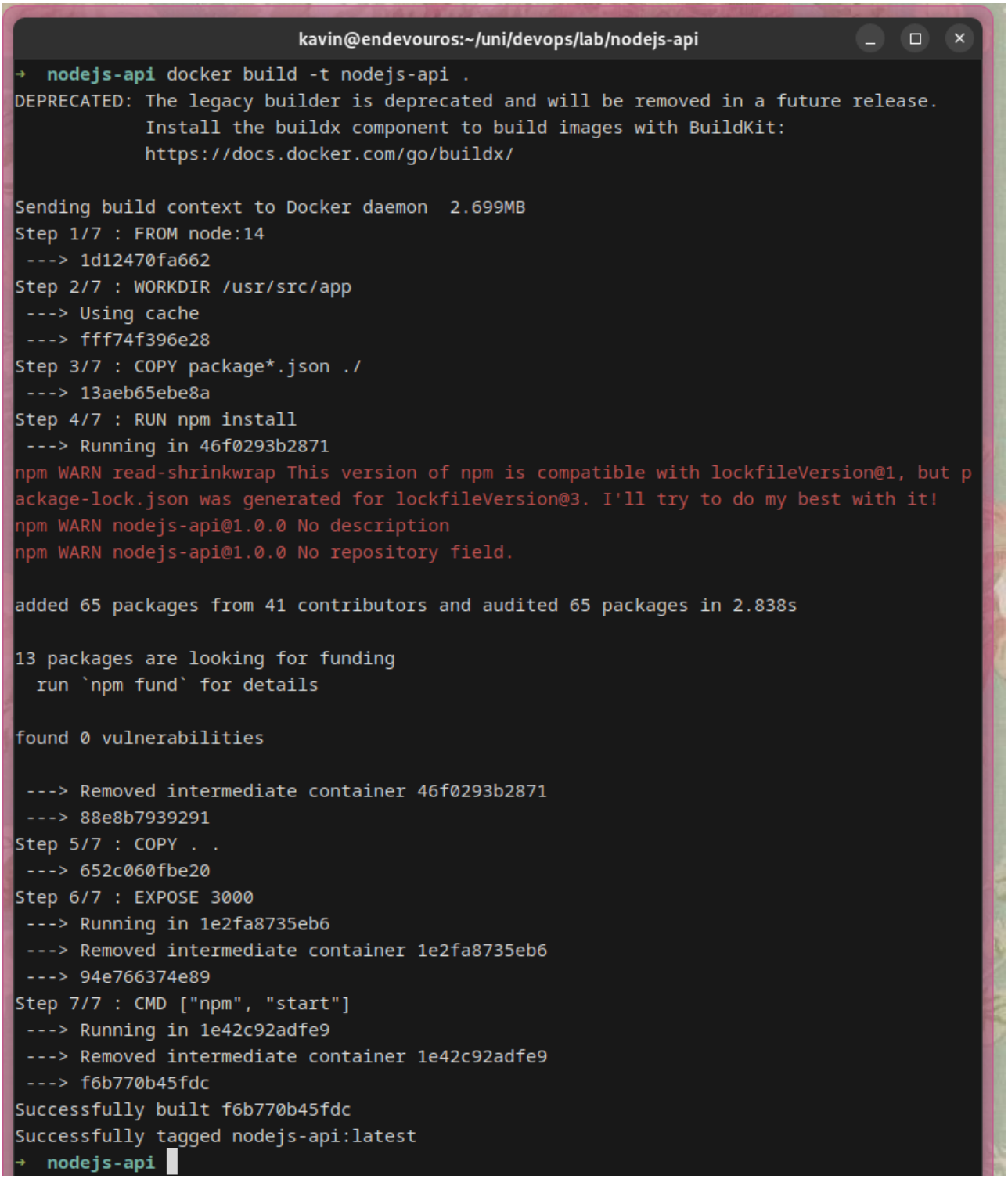
Step2: Containerize the Application with Docker

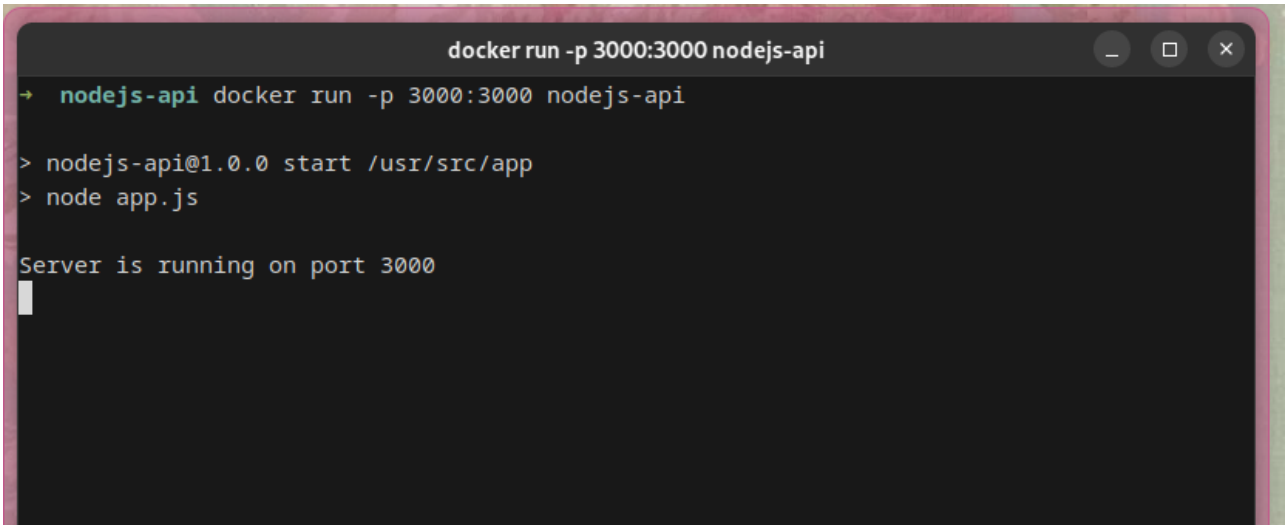
1. **Create a Dockerfile in the project root:**

```
FROM node:14
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["npm", "start"]
```

2. **Build and Test Docker Image Locally (optional)**

```
docker build -t nodejs-api .
docker run -p 3000:3000 nodejs-api
```


kavin@endevouros:~/uni/devops/lab/nodejs-api
→ nodejs-api docker build -t nodejs-api .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/
Sending build context to Docker daemon 2.699MB
Step 1/7 : FROM node:14
---> 1d12470fa662
Step 2/7 : WORKDIR /usr/src/app
---> Using cache
---> fff74f396e28
Step 3/7 : COPY package*.json ./
---> 13aeb65ebe8a
Step 4/7 : RUN npm install
---> Running in 46f0293b2871
npm WARN read-shrinkwrap This version of npm is compatible with lockfileVersion@1, but p
ackage-lock.json was generated for lockfileVersion@3. I'll try to do my best with it!
npm WARN nodejs-api@1.0.0 No description
npm WARN nodejs-api@1.0.0 No repository field.
added 65 packages from 41 contributors and audited 65 packages in 2.838s
13 packages are looking for funding
run `npm fund` for details
found 0 vulnerabilities
---> Removed intermediate container 46f0293b2871
---> 88e8b7939291
Step 5/7 : COPY . .
---> 652c060fbe20
Step 6/7 : EXPOSE 3000
---> Running in 1e2fa8735eb6
---> Removed intermediate container 1e2fa8735eb6
---> 94e766374e89
Step 7/7 : CMD ["npm", "start"]
---> Running in 1e42c92adfe9
---> Removed intermediate container 1e42c92adfe9
---> f6b770b45fdc
Successfully built f6b770b45fdc
Successfully tagged nodejs-api:latest
→ nodejs-api


docker run -p 3000:3000 nodejs-api
→ nodejs-api docker run -p 3000:3000 nodejs-api
> nodejs-api@1.0.0 start /usr/src/app
> node app.js
Server is running on port 3000

Step3: Set Up Jenkins Job

1. Create a New Jenkins Job:
    - Go to Jenkins Dashboard > New Item.
    - Select Pipeline and name the job.
    - In the Pipeline section, choose Pipeline script from SCM and configure your Git repository URL.
2. Configure Permissions for Docker Access:
    - Add Jenkins to the Docker group and restart services:

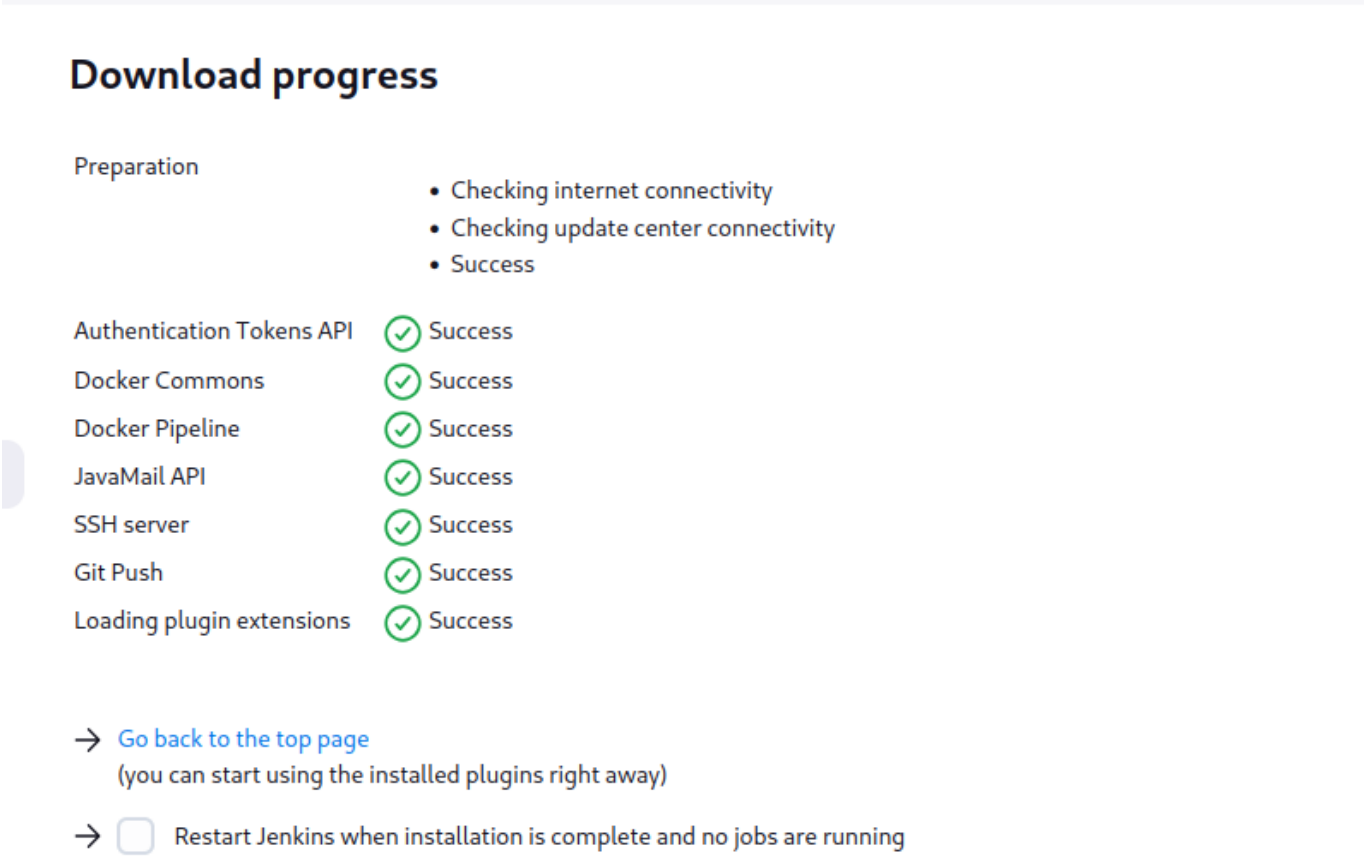
```
sudo usermod -aG docker jenkins
sudo systemctl restart jenkins
sudo systemctl restart docker
```

    - Test permission with:

```
sudo -u jenkins docker ps
```

Step4: Create a Jenkinsfile for Pipeline Automation

1. Install the necessary plugins into jenkins to make sure docker and git works.



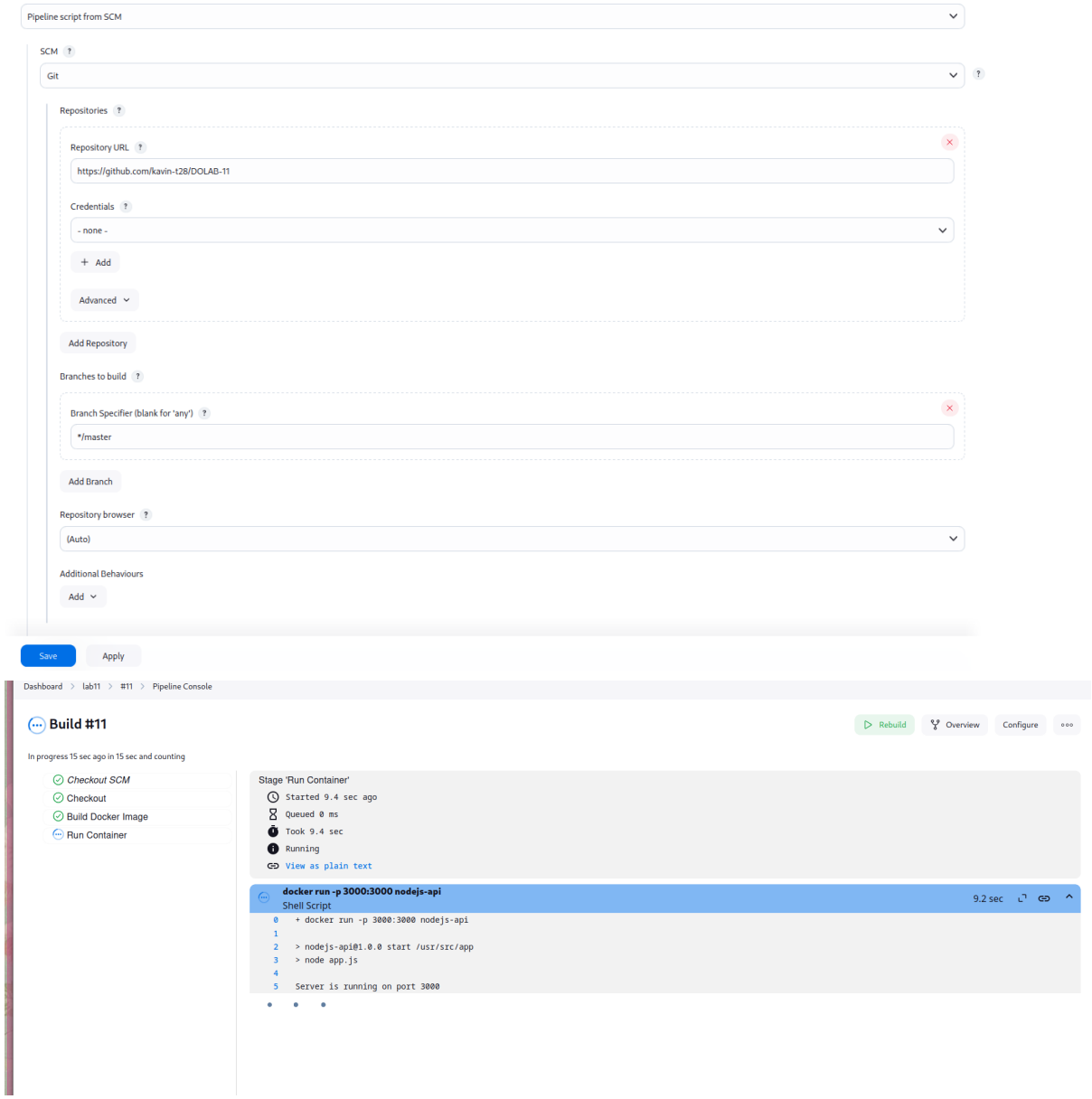
2. Add a `Jenkinsfile` to therepository:

```
pipeline {
  agent any
  environment {
    DOCKER_BUILDKIT = '1'
  }
  stages {
    stage('Checkout') {
      steps {
        git url: 'https://github.com/kavin-t28/DOLAB-11', branch:
'main'
      }
    }
    stage('Build Docker Image') {
      steps {
        script {
          sh 'docker build -t nodejs-api .'
        }
      }
    }
    stage('Run Container') {
      steps {
        script {
          sh 'docker run -p 3000:3000 nodejs-api'
        }
      }
    }
  }
}
```

3. Trigger the Build in Jenkins:

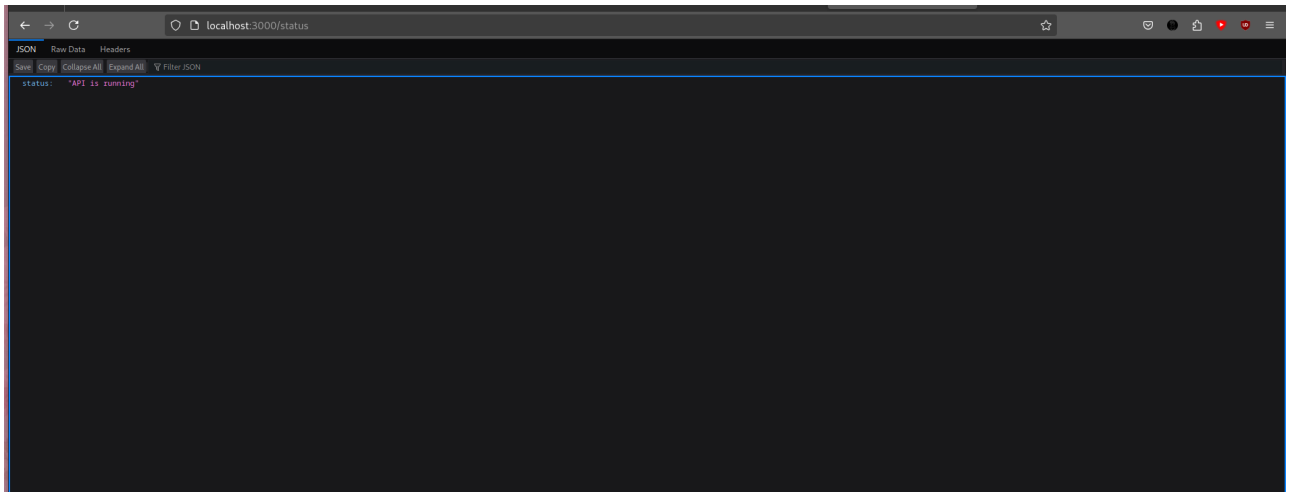
    - Save the job configuration and click Build Now.

- Jenkins will execute each stage, including Checkout, Build Docker Image, and Run Container.



Step5: Verify the Deployment

- After the pipeline completes, verify the /status endpoint on the Jenkins server by visiting



`http://<jenkins-server-ip>:3000/status.`

Submitted by Kavin.T (21011102047)