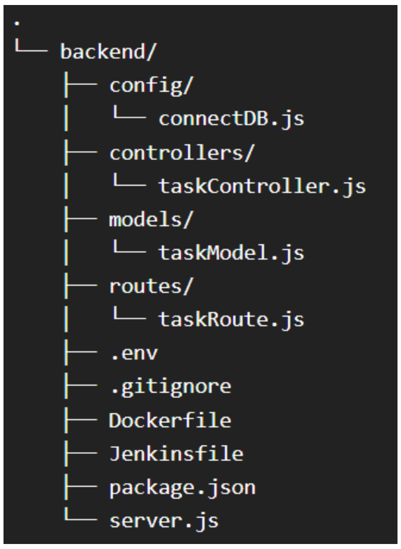
EE6254: DEVOPS ENGINEERING

ASSIGNMENT 2 – WRITING A PIPELINE TO DOCKERIZE APPLICATIONS

NAME	: GURUNAYAKE G.M.R.S.B
REG No.	: EG/2020/4368
SEMESTER	: 06
DATE	: 22/04/2024

- First, I have selected a repository for a node backend server. The file structure of the repository is as follows.

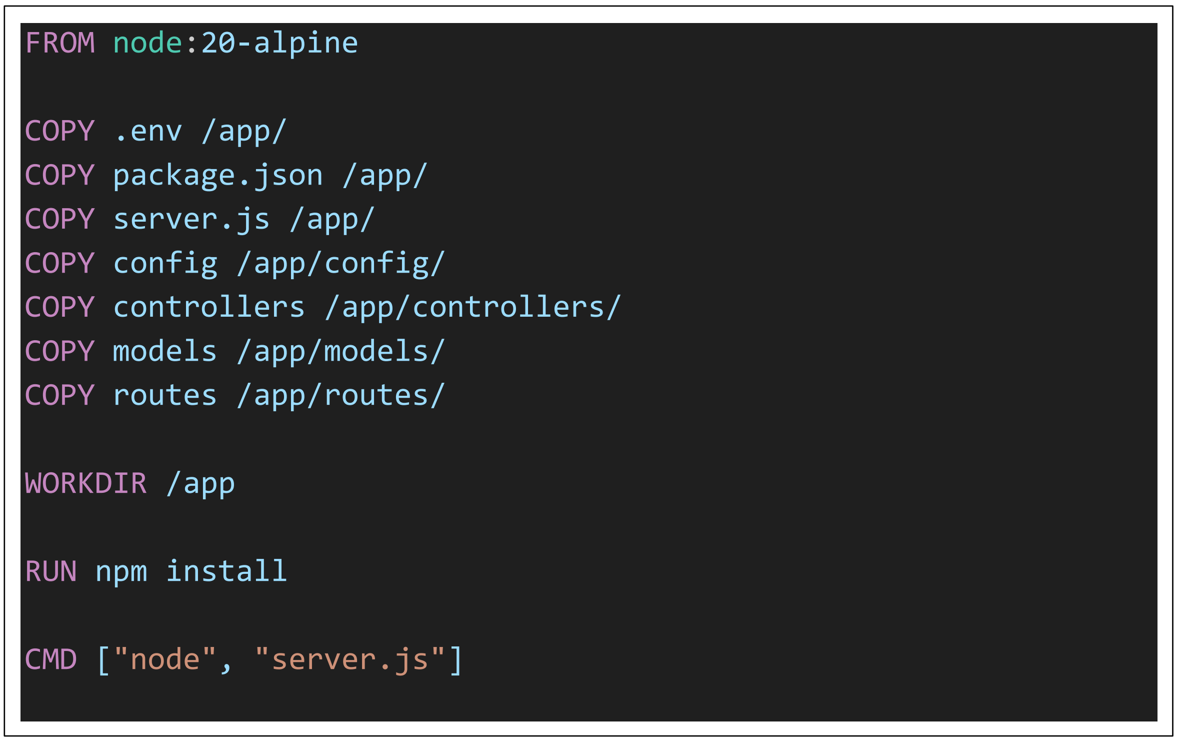


```
.
└── backend/
    ├── config/
    │   └── connectDB.js
    ├── controllers/
    │   └── taskController.js
    ├── models/
    │   └── taskModel.js
    ├── routes/
    │   └── taskRoute.js
    ├── .env
    ├── .gitignore
    ├── Dockerfile
    ├── Jenkinsfile
    ├── package.json
    └── server.js
```

- The GitHub link for this repository: https://github.com/SHASHI4368/4368-Gurunayake.git

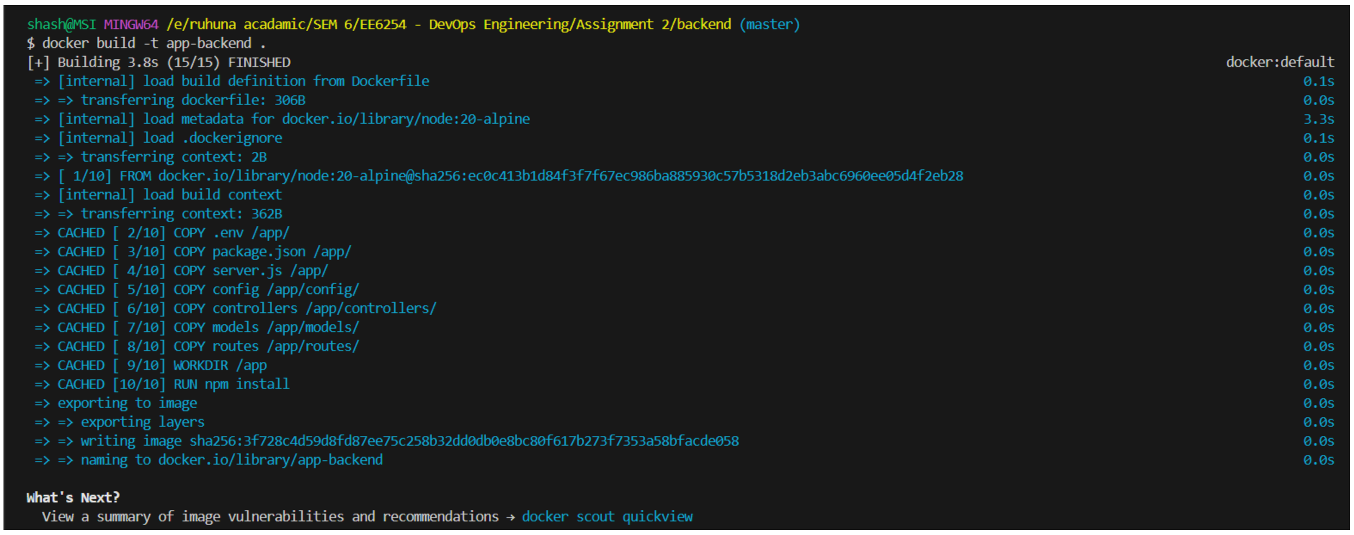
Manually Dockerizing the application

1. Creating the “Dockerfile”.



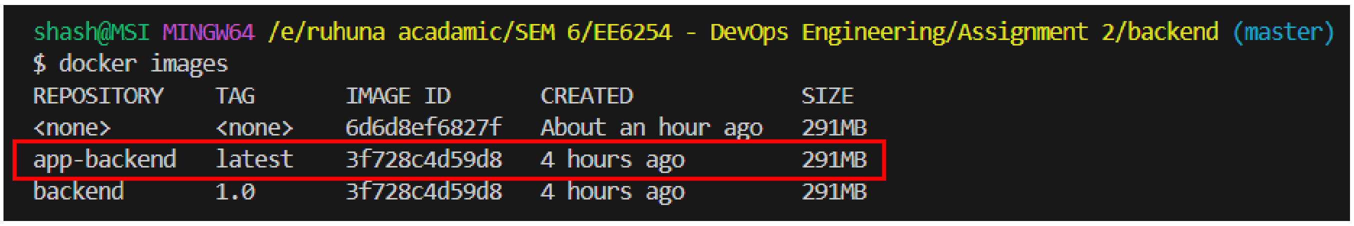
```
FROM node:20-alpine

COPY .env /app/
COPY package.json /app/
COPY server.js /app/
COPY config /app/config/
COPY controllers /app/controllers/
COPY models /app/models/
COPY routes /app/routes/

WORKDIR /app

RUN npm install

CMD ["node", "server.js"]
```

2. Creating the Docker Image Manually.



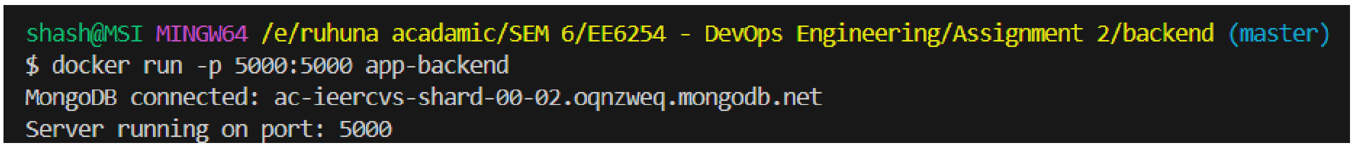
```
shash@MSI MINGW64 /e/ruhuna acadamic/SEM 6/EE6254 - DevOps Engineering/Assignment 2/backend (master)
$ docker build -t app-backend .
[+] Building 3.8s (15/15) FINISHED                                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                                            0.1s
 => => transferring dockerfile: 306B                                                                                            0.0s
 => [internal] load metadata for docker.io/library/node:20-alpine                                                               3.3s
 => [internal] load .dockerignore                                                                                               0.1s
 => => transferring context: 2B                                                                                                 0.0s
 => [ 1/10] FROM docker.io/library/node:20-alpine@sha256:ec0c413b1d84f3f7f67ec986ba885930c57b5318d2eb3abc6960ee05d4f2eb28       0.0s
 => [internal] load build context                                                                                               0.0s
 => => transferring context: 362B                                                                                               0.0s
 => CACHED [ 2/10] COPY .env /app/                                                                                              0.0s
 => CACHED [ 3/10] COPY package.json /app/                                                                                      0.0s
 => CACHED [ 4/10] COPY server.js /app/                                                                                         0.0s
 => CACHED [ 5/10] COPY config /app/config/                                                                                     0.0s
 => CACHED [ 6/10] COPY controllers /app/controllers/                                                                           0.0s
 => CACHED [ 7/10] COPY models /app/models/                                                                                     0.0s
 => CACHED [ 8/10] COPY routes /app/routes/                                                                                     0.0s
 => CACHED [ 9/10] WORKDIR /app                                                                                                 0.0s
 => CACHED [10/10] RUN npm install                                                                                              0.0s
 => exporting to image                                                                                                          0.0s
 => => exporting layers                                                                                                         0.0s
 => => writing image sha256:3f728c4d59d8fd87ee75c258b32dd0db0e8bc80f617b273f7353a58bfacde058                                    0.0s
 => => naming to docker.io/library/app-backend                                                                                  0.0s

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
```

3. Visualizing the created image.



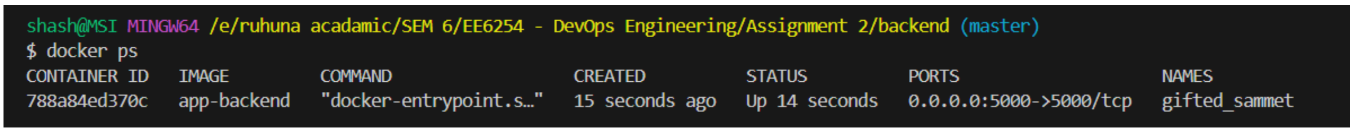
```
shash@MSI MINGW64 /e/ruhuna acadamic/SEM 6/EE6254 - DevOps Engineering/Assignment 2/backend (master)
$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED             SIZE
<none>        <none>    6d6d8ef6827f   About an hour ago   291MB
app-backend   latest    3f728c4d59d8   4 hours ago         291MB
backend       1.0       3f728c4d59d8   4 hours ago         291MB
```

4. Running the Docker Image Manually.



```
shash@MSI MINGW64 /e/ruhuna acadamic/SEM 6/EE6254 - DevOps Engineering/Assignment 2/backend (master)
$ docker run -p 5000:5000 app-backend
MongoDB connected: ac-ieercvs-shard-00-02.oqnzweq.mongodb.net
Server running on port: 5000
```

5. Visualizing the running containers.



```
shash@MSI MINGW64 /e/ruhuna acadamic/SEM 6/EE6254 - DevOps Engineering/Assignment 2/backend (master)
$ docker ps
CONTAINER ID   IMAGE         COMMAND                  CREATED          STATUS          PORTS                    NAMES
788a84ed370c   app-backend   "docker-entrypoint.s…"   15 seconds ago   Up 14 seconds   0.0.0.0:5000->5000/tcp   gifted_sammet
```

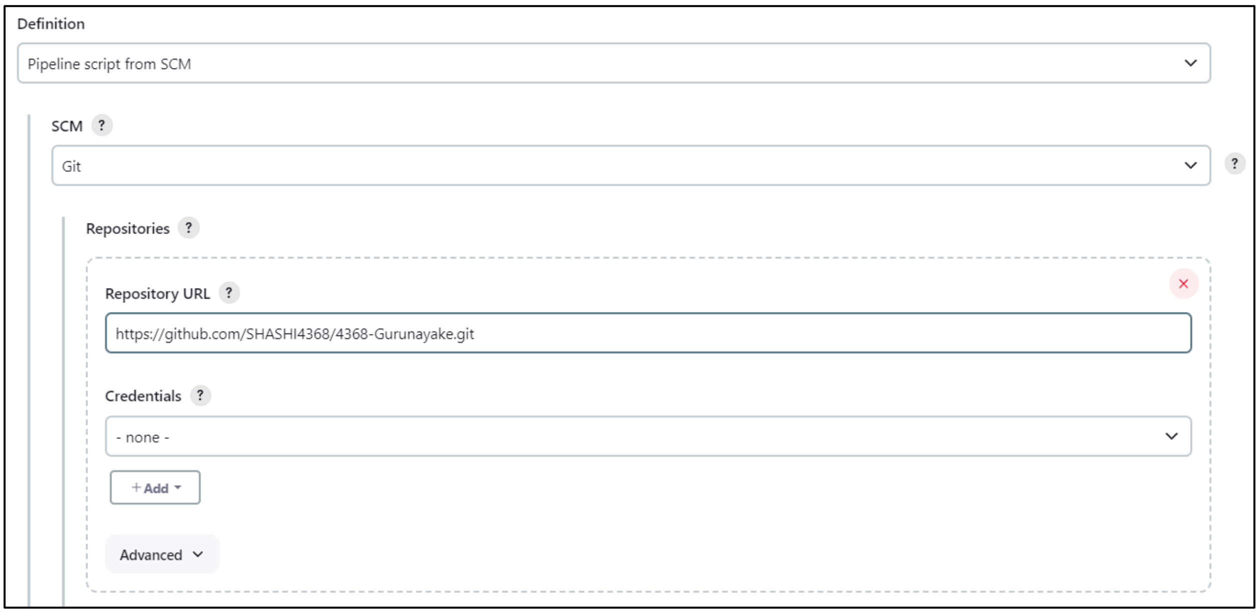
- As we can see, our Dockerfile is created and executed successfully using manual commands.
- We now automate this process using a Jenkins pipeline

Automating the Dockerizing process using Jenkins

1. Creating the "Jenkinsfile"

```
pipeline {
    agent any

    environment {
        GITHUB_REPO_URL = 'https://github.com/SHASHI4368/4368-Gurunayake.git'
    }

    stages {
        stage('Checkout') {
            steps {
                git branch: 'master', url: "${env.GITHUB_REPO_URL}"
            }
        }

        stage('Build Docker Image') {
            steps {
                bat 'docker build -t app-backend .'
            }
        }

        stage('Run Docker Image') {
            steps {
                bat 'docker run -p 5000:5000 app-backend'
            }
        }
    }

    post {
        always {
            echo 'Cleaning up docker containers'
            bat 'docker stop app-backend'
            bat 'docker rm app-backend'
        }
    }
}
```

2. Linking the repository to Jenkins



Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

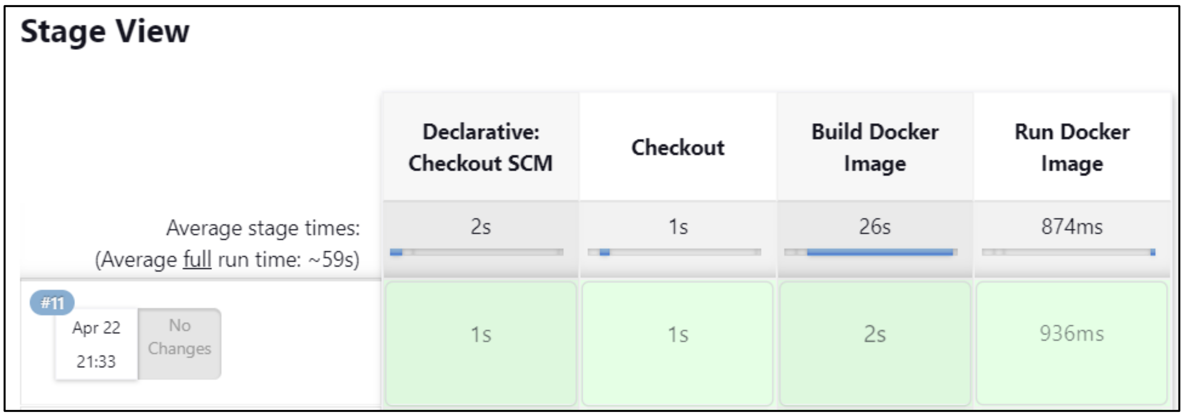
Repository URL ?

https://github.com/SHASHI4368/4368-Gurunayake.git

Credentials ?

- none -

3. Building the Jenkins pipeline



Stage View

	Declarative: Checkout SCM	Checkout	Build Docker Image	Run Docker Image
Average stage times: (Average full run time: ~59s)	2s	1s	26s	874ms
#11 Apr 22 21:33 No Changes	1s	1s	2s	936ms

- The complete log of this build is in the "/report" directory of this repository.