International Institute of Information Technology, Bangalore

CS 816 - Software Production Engineering

Samarpita Bhaumik
MT2023053

Sunnidhya Roy
MT2023079

Project - A React Native mobile application for movie recommendation using Devops pipeline

under guidance of

Prof. B. Thangaraju

Index

Relevant Links:

1. **Github Repository**: https://github.com/samarpita-bhaumik/Moviemate.git
2. **DockerHub**:
   frontend: https://hub.docker.com/repository/docker/samsun0196/spefrontend1/general
   backend: https://hub.docker.com/repository/docker/samsun0196/spebackend3/general

Problem Statement

A React-native mobile application for movie recommendation using Devops pipeline.

DevOps tool chain

The pipeline includes,

1. Using a source control management tool - like Git, GitHub.
2. Testing - test our code using unittest.
3. Continuous Integration - Continuous integrate our code using tool like Jenkins.
4. Containerize - Containerize our code using Docker.
5. Push our created Docker image to Docker hub.
6. Container Orchestration using Docker Compose,
7. Deployment - Deployment using Ansible.
8. For Monitoring & Logging we have used ELK Stack.

Devops

What is Devops?

DevOps is a collaborative approach that blends software development (Dev) and IT operations (Ops) to enhance the efficiency and quality of software delivery. It emphasizes breaking down barriers between teams, automating processes, and fostering a culture of continuous improvement. Central to DevOps is the concept of treating infrastructure as code, enabling teams to manage infrastructure through automation and version control systems. This methodology facilitates faster software delivery, increased deployment frequency, and closer alignment with business objectives. By embracing DevOps principles, organizations can respond more effectively to customer needs, shorten time-to-market, and gain a competitive advantage in today's dynamic digital landscape.

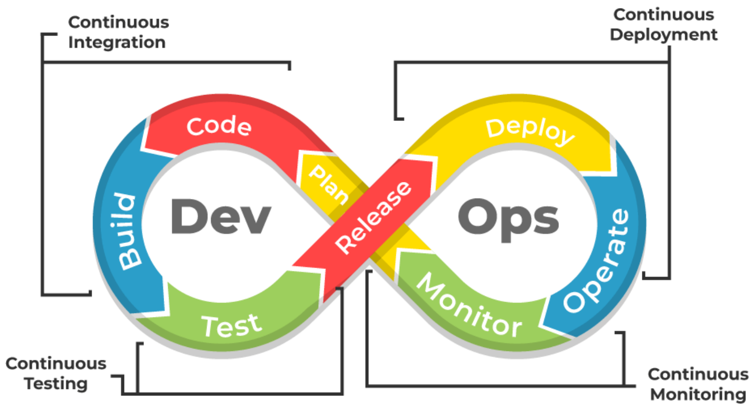


Fig: Phases in Devops

Why Devops?

Organizations embrace DevOps for several compelling reasons. Firstly, it cultivates collaboration and communication between development and operations teams, dismantling traditional silos and fostering a shared responsibility for the entire software delivery lifecycle. This collaboration results in expedited issue resolution, enhanced efficiency, and the delivery of higher-quality software releases.

Secondly, DevOps underscores the importance of automating manual tasks, empowering teams to streamline processes and mitigate human error. Automation facilitates continuous integration and continuous delivery (CI/CD), enabling organizations to deliver software updates more frequently, reliably, and with shorter lead times. Moreover, DevOps instills a culture of continuous improvement and feedback, motivating teams to iterate and innovate swiftly. By continually delivering value to customers and integrating feedback into the development process, organizations can remain competitive and adeptly address evolving customer needs.

Overall, DevOps equips organizations with greater agility, scalability, and resilience in their software delivery practices, ultimately driving business growth and success in today's dynamic and competitive market environment.

Tools used:

We will be using the following tools:

- **Source control management tool:** Git and GitHub.
- **Editor for the Project:** Visual Studio Code
- **Continuous Integration:** Jenkins
- **Containerising the code:** Docker and Docker Hub
- **Configuration Management and Continuous Deployment:** Ansible

GitHub: GitHub serves as a prominent platform for hosting code and managing version control, widely embraced in DevOps practices. It fosters collaboration among developers and organizations through features like pull requests, issue tracking, and code review. GitHub seamlessly integrates with CI/CD pipelines, automating workflows and streamlining code management and deployment processes efficiently.

Jenkins: Jenkins emerges as a popular open-source automation server utilized for implementing CI/CD pipelines. It facilitates continuous integration and delivery by automating building, testing, and deploying software. With its extensive ecosystem of plugins and integrations, Jenkins is adaptable to diverse development environments, empowering teams to achieve swift and reliable software delivery.

Docker: Docker emerges as a containerization platform streamlining the packaging, distribution, and execution of applications within lightweight, isolated containers. In DevOps, Docker ensures consistent and reproducible environments across development, testing, and production stages. Its portability and scalability facilitate seamless application deployment, thereby enhancing agility and efficiency within software delivery pipelines.

Ansible: Ansible serves as a potent automation tool catering to configuration management, application deployment, and orchestration needs. It enables DevOps teams to automate repetitive tasks, manage infrastructure as code, and streamline deployment processes across hybrid and multi-cloud environments. With its user-friendly interface, agentless architecture, and declarative language, Ansible emerges as an ideal choice for automating infrastructure and application deployments within DevOps workflows.

Expo App: We are using the expo go app to load the mobile app onto our phone through tunnelling. Expo Go is a sandbox that allows for quick experimentation in building native Android and iOS apps. It is designed to be the fastest way to start app development. With Expo Go, developers can easily create apps for Android, iOS, and the web using JavaScript and React. The platform offers various features like interactive gestures, graphics, and the ability to access every device API. Expo Go also provides tools for app store submission, automated end-to-end testing, and fast iteration cycles. Developers can preview updates using a QR code, manage app store credentials, and benefit from an active community for support and collaboration

Ngrok: It is a potent tool utilized for establishing secure tunnels to local servers and services. It offers a means to securely expose localhost or any other local server to the internet. Ngrok generates a public URL that directs traffic to a specified port on the local machine, permitting external access to local development environments or services. In the realm of DevOps, Ngrok proves especially valuable for locally testing webhooks or APIs prior to deployment to production environments. It enables developers to share ongoing work with colleagues or clients for feedback, simplifies troubleshooting, and accelerates development and testing iterations. Moreover, Ngrok seamlessly integrates into CI/CD pipelines, allowing for the testing of webhook integrations or receiving external webhooks in local development environments. This integration enhances workflow efficiency and promotes collaboration among team members.

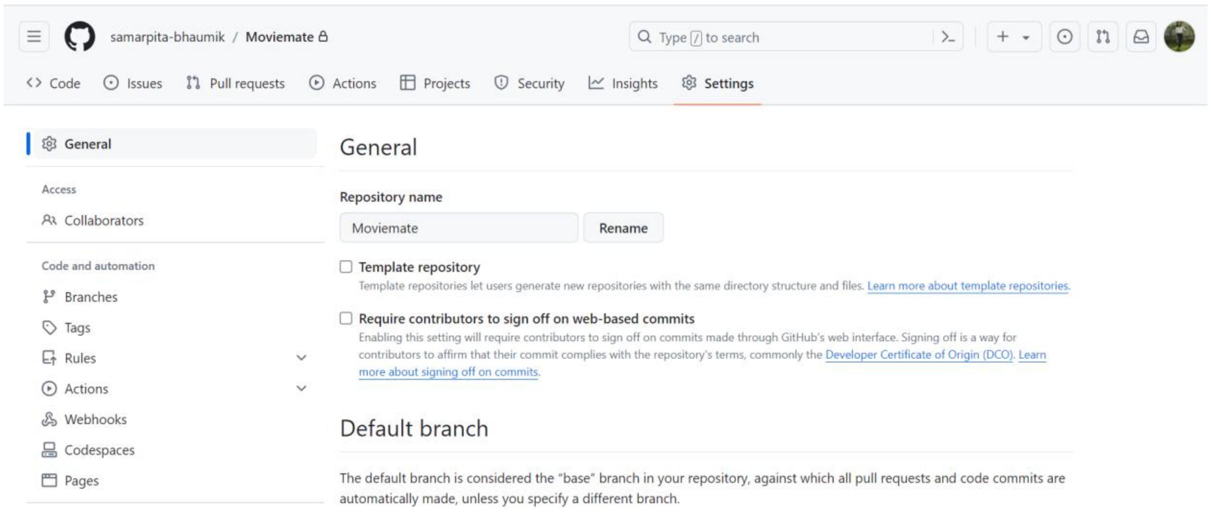
High Level Overview

1. **Step 1:** We will make a project folder and initialize Git in that folder.
2. **Step 2:** We will configure Jenkins to make a deployment pipeline, where Jenkins will clone code from Github. Then by using docker plugin we will containerize the code and push to docker hub. Then using Ansible we will deploy docker image on a local machine. We will do container orchestration using Docker Compose and monitoring and logging using ELK Stack.

Source Control Management

Github Remote Repository:

My repository after logging into Github account.



samarpita-bhaumik / Moviemate

Code · Issues · Pull requests · Actions · Projects · Security · Insights · Settings

General

Access
Collaborators

Code and automation
Branches
Tags
Rules
Actions
Webhooks
Codespaces
Pages

General

Repository name
Moviemate
Rename

Template repository
Template repositories let users generate new repositories with the same directory structure and files. Learn more about template repositories.

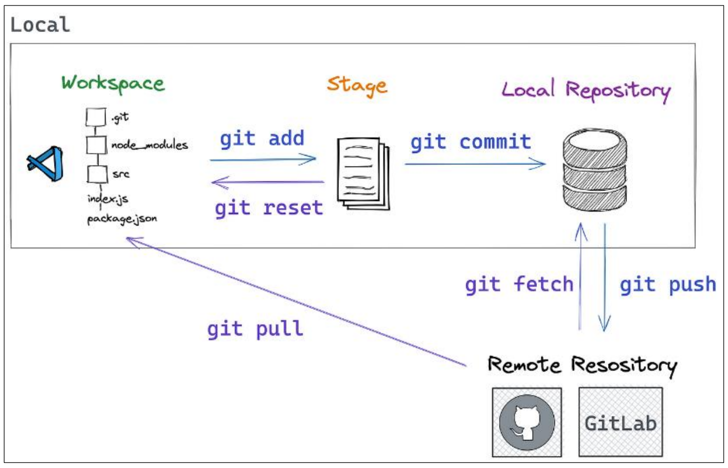
Require contributors to sign off on web-based commits
Enabling this setting will require contributors to sign off on commits made through GitHub's web interface. Signing off is a way for contributors to affirm that their commit complies with the repository's terms, commonly the Developer Certificate of Origin (DCO). Learn more about signing off on commits.

Default branch

The default branch is considered the "base" branch in your repository, against which all pull requests and code commits are automatically made, unless you specify a different branch.

What is Git?

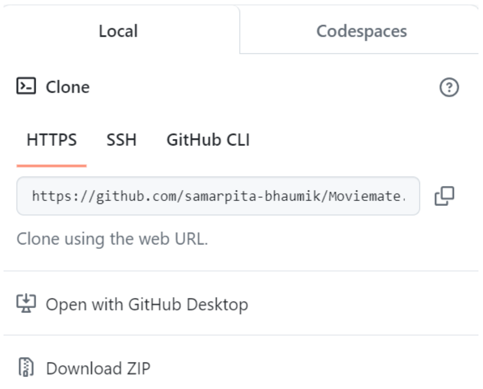
Git is a **distributed version control system**. The distributed **version control system** refers to the concept that a copy of the project files will be maintained at the client's system along with the copy being maintained at the remote server. This means that the developer's machine will also maintain a copy of backup files in addition to the repository maintained on the server.



The provided steps are used to create a local Git repository, initialize it, add a README.md file, commit changes, set the main branch name to 'main', link the local repository to a remote repository on GitHub, and push the changes to the remote repository. Here's a breakdown of the steps:

A folder is created with the name **Moviemate.**

We clone the remote repository into our local system as follows:



https://github.com/samarpita-bhaumik/Moviemate.

Clone using the web URL.

To clone the repository to our local system we write:

```
git clone https://github.com/samarpita-bhaumik/Moviemate.git
```

Created a new README.md file with the content "# Moviemate":

```
echo " # Moviemate " >> README.md
```

Initialize a new Git repository in the current directory:

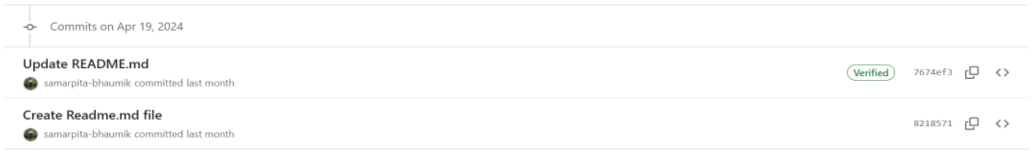
```
git init
```

Add the README.md file to the staging area:

```
git add README.md
```

Commit the changes with a commit message:

```
git commit -m "Create Readme.md file"
```



Commits on Apr 19, 2024

Update README.md
samarpita-bhaumik committed last month

7674ef3

Create Readme.md file
samarpita-bhaumik committed last month
8218571

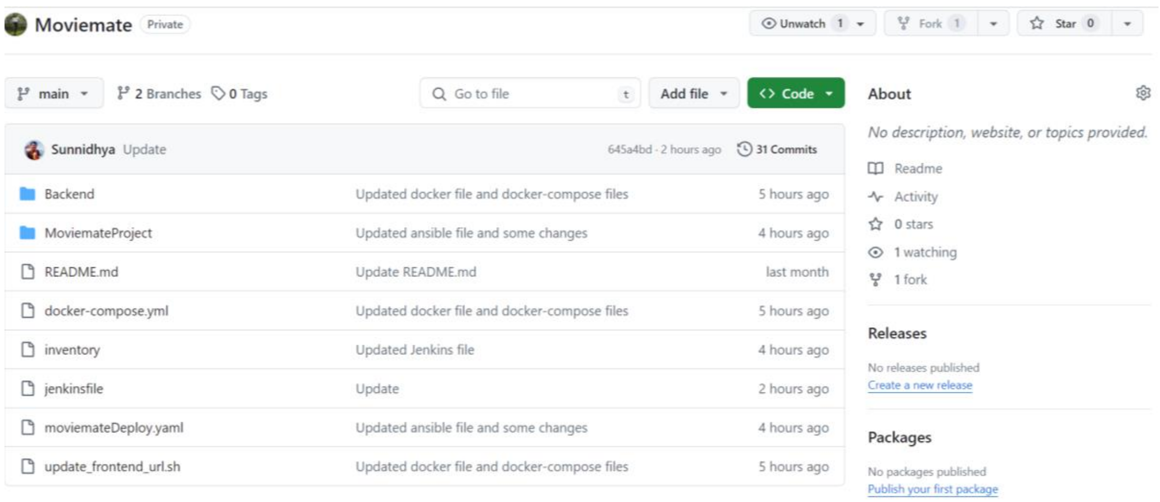
Link the local repository to a remote repository on GitHub:

```
git remote add origin https://github.com/samarpita-bhaumik/Moviemate.git
```

Push the committed changes from the local 'main' branch to the remote 'main' branch on GitHub:

```
git push -u origin main
```

Develop code of the project as per requirements and commit the changes to local repository first by git add . and then git commit -m "some message" after that push to remote repository by using git push -u origin <branch_name>


Moviemate Private

Unwatch 1 · Fork 1 · Star 0

main · 2 Branches · 0 Tags · Go to file · Add file · Code

Sunnidhya Update · 645a4bd · 2 hours ago · 31 Commits

Backend	Updated docker file and docker-compose files	5 hours ago
MoviemateProject	Updated ansible file and some changes	4 hours ago
README.md	Update README.md	last month
docker-compose.yml	Updated docker file and docker-compose files	5 hours ago
inventory	Updated Jenkins file	4 hours ago
jenkinsfile	Update	2 hours ago
moviemateDeploy.yaml	Updated ansible file and some changes	4 hours ago
update_frontend_url.sh	Updated docker file and docker-compose files	5 hours ago

About

No description, website, or topics provided.

Readme
Activity
0 stars
1 watching
1 fork

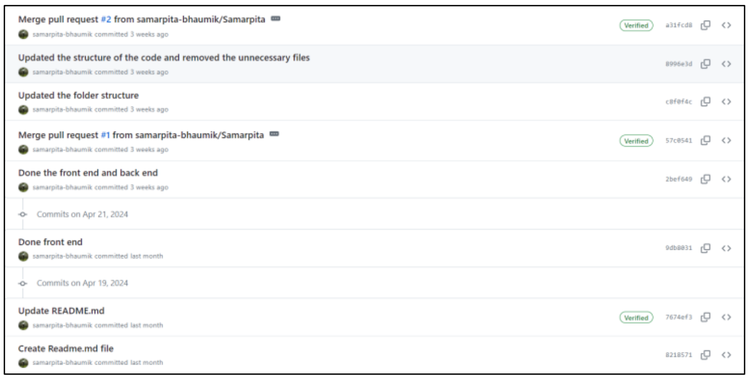
Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

<u>Our commit history:</u>

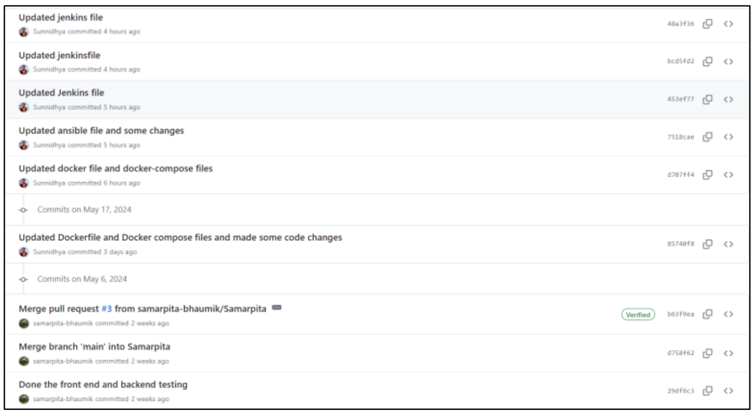

Merge pull request #2 from samarpita-bhaumik/Samarpita — samarpita-bhaumik committed 3 weeks ago — Verified — a31fcd8
Updated the structure of the code and removed the unnecessary files — samarpita-bhaumik committed 3 weeks ago — 8996e3d
Updated the folder structure — samarpita-bhaumik committed 3 weeks ago — c8f0f4c
Merge pull request #1 from samarpita-bhaumik/Samarpita — samarpita-bhaumik committed 3 weeks ago — Verified — 57c0541
Done the front end and back end — samarpita-bhaumik committed 3 weeks ago — 2bef649

Commits on Apr 21, 2024

Done front end — samarpita-bhaumik committed last month — 9db8031

Commits on Apr 19, 2024

Update README.md — samarpita-bhaumik committed last month — Verified — 7674ef3
Create Readme.md file — samarpita-bhaumik committed last month — 8218571

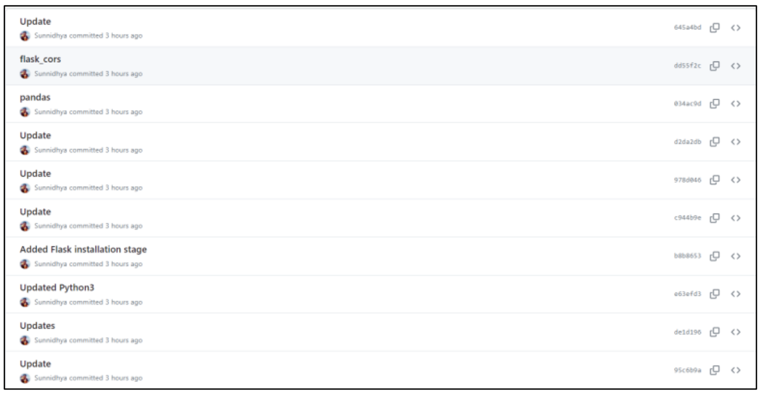

Updated jenkins file — Sunnidhya committed 4 hours ago — 48a3f36
Updated jenkinsfile — Sunnidhya committed 4 hours ago — bcd5fd2
Updated Jenkins file — Sunnidhya committed 5 hours ago — 453ef77
Updated ansible file and some changes — Sunnidhya committed 5 hours ago — 7518cae
Updated docker file and docker-compose files — Sunnidhya committed 6 hours ago — d707ff4

Commits on May 17, 2024

Updated Dockerfile and Docker compose files and made some code changes — Sunnidhya committed 3 days ago — 85740f8

Commits on May 6, 2024

Merge pull request #3 from samarpita-bhaumik/Samarpita — samarpita-bhaumik committed 2 weeks ago — Verified — b63f9ea
Merge branch 'main' into Samarpita — samarpita-bhaumik committed 2 weeks ago — d758f62
Done the front end and backend testing — samarpita-bhaumik committed 2 weeks ago — 29df6c3


Update — Sunnidhya committed 3 hours ago — 645a4bd
flask_cors — Sunnidhya committed 3 hours ago — dd55f2c
pandas — Sunnidhya committed 3 hours ago — 034ac9d
Update — Sunnidhya committed 3 hours ago — d2da2db
Update — Sunnidhya committed 3 hours ago — 978d046
Update — Sunnidhya committed 3 hours ago — c944b9e
Added Flask installation stage — Sunnidhya committed 3 hours ago — b8b8653
Updated Python3 — Sunnidhya committed 3 hours ago — e63efd3
Updates — Sunnidhya committed 3 hours ago — de1d196
Update — Sunnidhya committed 3 hours ago — 95c6b9a

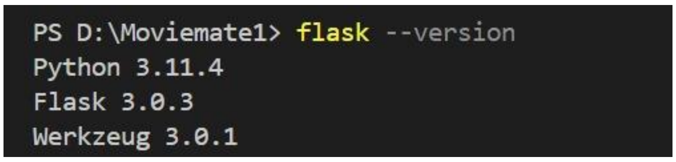
React Native Installation:

1. Installation of React Native in the local system:
   $ npm install -g expo-cli
   $ expo init MoviemateProject
   $ cd MoviemateProject
   $ npx expo start --tunnel

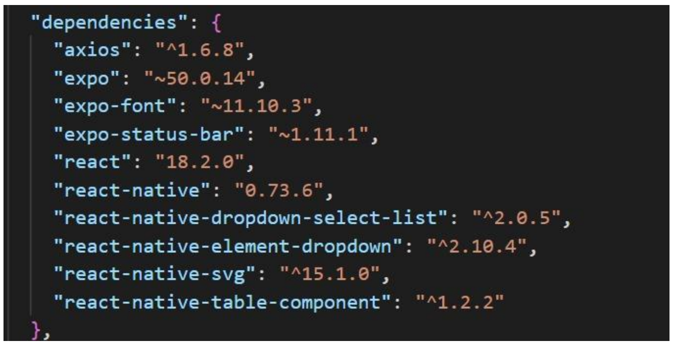
Flask Installation:

2. Installation of flask in the local system:
   $ pip install flask

3. **Check the version:**

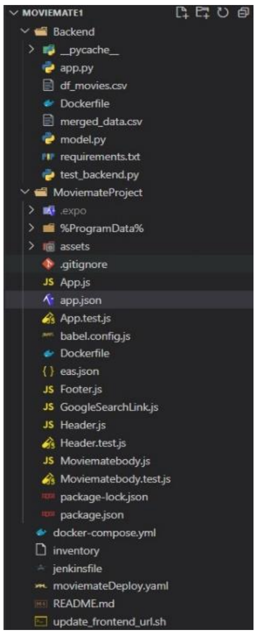


```
PS D:\Moviemate1> flask --version
Python 3.11.4
Flask 3.0.3
Werkzeug 3.0.1
```



```
"dependencies": {
  "axios": "^1.6.8",
  "expo": "~50.0.14",
  "expo-font": "~11.10.3",
  "expo-status-bar": "~1.11.1",
  "react": "18.2.0",
  "react-native": "0.73.6",
  "react-native-dropdown-select-list": "^2.0.5",
  "react-native-element-dropdown": "^2.10.4",
  "react-native-svg": "^15.1.0",
  "react-native-table-component": "^1.2.2"
},
```

This is the structure of our project:



```
MOVIEMATE1
  Backend
    __pycache__
    app.py
    df_movies.csv
    Dockerfile
    merged_data.csv
    model.py
    requirements.txt
    test_backend.py
  MoviemateProject
    .expo
    %ProgramData%
    assets
    .gitignore
    App.js
    app.json
    App.test.js
    babel.config.js
    Dockerfile
    eas.json
    Footer.js
    GoogleSearchLink.js
    Header.js
    Header.test.js
    Moviematebody.js
    Moviematebody.test.js
    package-lock.json
    package.json
  docker-compose.yml
  inventory
  jenkinsfile
  moviemateDeploy.yaml
  README.md
  update_frontend_url.sh
```

Steps followed to build the code in Visual Studio Code:

1. First we have written the Frontend code under the MoviemateProject.

App.js

```
import React from 'react';
import { View, StyleSheet, ScrollView } from 'react-native';
import Header from './Header';
import Moviematebody from './Moviematebody';
import Footer from './Footer';


const App = () => {
  return (

    <View style={styles.header}>
      <Header title1="Moviemate"  />
      <Moviematebody />
      <Footer text="About Us" />
    </View>
 );
};

const styles = StyleSheet.create({
  header: {
    flex: 1,
    backgroundColor: 'white',
    justifyContent:'center'
  },
});

export default App;
```

Footer.js

```
import React from 'react';
import { View, Text, StyleSheet } from 'react-native';

const Footer = ({ text }) => {
  return (
    <View style={styles.footer}>
      <Text style={styles.text}>{text}</Text>
    </View>
  );
};

const styles = StyleSheet.create({
  footer: {
    height: 40,
    backgroundColor: '#70A597',
    justifyContent: 'center',
    alignItems: 'center',
  },
  text:{
    color: 'white',
    fontSize: 15,
    fontWeight: 'bold',
    fontFamily: 'Roboto',
  }
});

export default Footer;
```

GoogleSearchLink.js

```
import React from 'react';

import { TouchableOpacity, Linking, Text, View } from 'react-native';



const GoogleSearchLink = ({ searchText }) => {

  const handlePress = () => {

    const query = encodeURIComponent(searchText);

    const url = `https://www.google.com/search?q=${query}`;

    Linking.openURL(url);

  };

   return (

    <TouchableOpacity onPress={handlePress}>

      <View style={{ flexDirection: 'row', flexWrap: 'wrap', maxWidth: 300 }}>

        <Text style={{ color: 'blue' }} numberOfLines={4}>{`Search on Google : ${searchText}`}</Text>

      </View>

    </TouchableOpacity>

  );

};



export default GoogleSearchLink;
```

Header.js

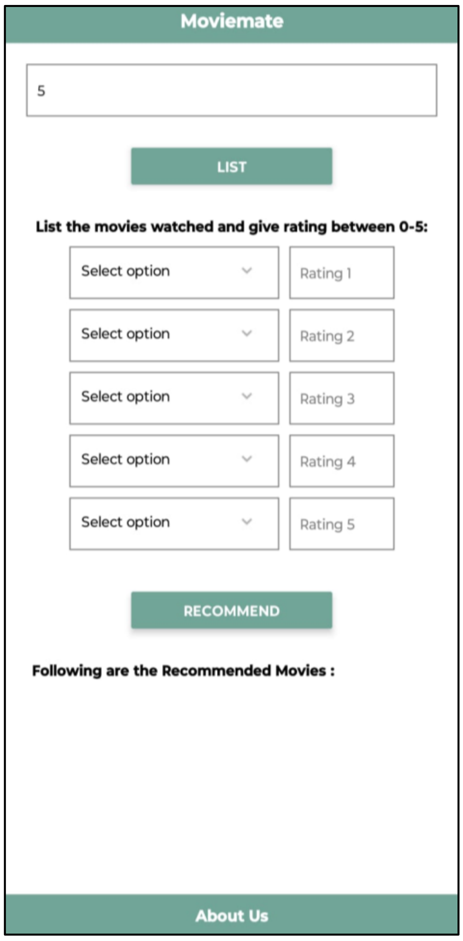
```
import React from 'react';
import { View, Text, StyleSheet, StatusBar } from 'react-native';

const Header = ({ title1 }) => {
  return (
    <View style={[styles.header,{height:40,paddingBottom:10}]} testID='status-bar'>
      <StatusBar backgroundColor="#70A597" barStyle="light-content"/>
      <Text style={styles.headerText} testID='text-bar'>{title1}</Text>
    </View>
  );
};

const styles = StyleSheet.create({
  header: {
    width: '100%',
    height: 60,
    backgroundColor: '#70A597',
    alignItems: 'center',
    justifyContent:'flex-end'
  },
  headerText: {
    color: 'white',
    fontSize: 18,
    fontWeight: 'bold',
    fontFamily: 'Roboto',

  },
});

export default Header;
```

Moviematebody.js

```
import axios from 'axios';
import React, { useState, useEffect } from 'react';
import { View, Text, StyleSheet, TextInput, Button, Alert, ScrollView } from 'react-native';
import { SelectList } from 'react-native-dropdown-select-list';
import Footer from './Footer';
import GoogleSearchLink from './GoogleSearchLink';

// Use the internal Docker network address
const backendUrl = 'http://localhost:5000';

const Moviematebody = () => {
  const [text, setText] = useState('');
  const [value, setValue] = useState(0);
  const [ratings, setRatings] = useState([]);
  const [visible, setVisible] = useState(false);
  const [selected, setSelected] = useState('');
  const [keyForSelectList, setKeyForSelectList] = useState(0);
const [moviesData, setMoviesData] = useState([]);
const [filteredMoviesData, setFilteredMoviesData] = useState([]);
const [selectedMovies, setSelectedMovies] = useState([]);
const [recommendation, setRecommendation] = useState([]);

const onChangeText = (inputText) => {
  setText(inputText);
};

const handleButtonPress = () => {
  setValue(text);
  setVisible(text !== '0');
  setKeyForSelectList((prevKey) => prevKey + 1);
  setRatings([]);
  setRecommendation([]);
};

const handleRatingInputChange = (index, value) => {
  if (!isNaN(value) && value >= 0 && value <= 5) {
    const updatedRatings = [...ratings];
    updatedRatings[index] = value;
    setRatings(updatedRatings);
  } else {
    Alert.alert('Please enter a valid rating between 0 and 5');
  }
};

useEffect(() => {
  fetchMoviesData();
}, []);

const fetchMoviesData = async () => {
  axios(
    {
      method: "GET",
      url: `${backendUrl}/listofmovies`,
      headers: {
        'Content-Type': 'application/json'
      }
    })
    .then(response => response.data)
    .then(movies => {
      console.log('Response:', movies);
      setMoviesData(movies);
      setFilteredMoviesData(movies);
    })
    .catch((error) => {
      console.error('Error fetching movies data:', error);
    });

};
const handleMovieChange = (selected, index) => {
  const updatedSelectedMovies = [...selectedMovies];
```

```
    updatedSelectedMovies[index] = selected;
    setSelectedMovies(updatedSelectedMovies);
  };

  const handleSearchChange = (searchText) => {
    const filteredData = moviesData.filter(movie =>
movie.Title.toLowerCase().includes(searchText.toLowerCase()));
    setFilteredMoviesData(filteredData);
  };


  const handleRecommendPress = async () => {
    try {
      const dataToSend = selectedMovies.map((movie, index) => ({
        movie: movie,
        rating: ratings[index]
      }));
      console.log(dataToSend)
      const response = await axios.post(`${backendUrl}/moviename`, dataToSend, {
        headers: {
          'Content-Type': 'application/json'
        }
      });
      setRecommendation(response.data);
      console.log('Recommendation response:', response.data);
    } catch (error) {
      console.error('Error sending recommendation:', error);

    }
  };


  const renderMovieInputs = () => {
    return Array.from({ length: value }, (_, index) => (
      <View key={index} style={styles.movieContainer}>
        <View style={styles.inputContainer}>
          <SelectList
            setSelected={(val) => setSelected(val)}
            data={filteredMoviesData && Array.isArray(filteredMoviesData) ?
filteredMoviesData.map((movie) => ({ key: movie.MovieID, value: movie.Title })) : []}
            key={keyForSelectList}
            boxStyles={{
              height: 50,
              borderColor: 'gray',
              borderWidth: 1,
              paddingLeft: 10,
              marginRight: 5,
              marginBottom: 10,
              width: 200,
              borderRadius: 0,
            }}
            dropdownStyles={{ borderRadius: 0, marginBottom: 10, width: 200}}
            save="value"
            searchBarPlaceholder="Search movies..."
            onChangeText={handleSearchChange} // Handle search input change
            onSelect={() => handleMovieChange(selected, index)}

          />
          <TextInput
            style={[styles.input, styles.ratingInput]}
            onChangeText={(value) => handleRatingInputChange(index, value)}
            value={ratings[index]}
            placeholder={`Rating ${index + 1}`}
            placeholderTextColor="gray"
            keyboardType="numeric"
          />
        </View>

      </View>
    )
    );
  };



  return (
    <ScrollView style={styles.scrollview}>
      <View style={styles.container}>
        <TextInput
          style={styles.input}
          onChangeText={onChangeText}
          value={text}
          placeholder="Enter number of movies you have watched ..."
          placeholderTextColor="gray"
        />
        <View style={styles.button}>
          <Button
            title="LIST"
            onPress={() => handleButtonPress(text)}
            color="#70A597"
            accessibilityLabel="Submit Button"
          />
        </View>
        <View style={styles.dynamicinput}>
          {visible && <Text style={styles.space}>List the movies watched and give rating between 0-
5:</Text>}
          {renderMovieInputs()}
        </View>
        <View style={styles.button}>
          <Button
            title="Recommend"
            onPress={() => handleRecommendPress()}
            color="#70A597"
            accessibilityLabel="Submit Button"
          />
      </View>
      <Text style={styles.space1}>
        Following are the Recommended Movies : {"\n"}{"\n"}
        {recommendation && Array.isArray(recommendation) ? (
          recommendation.map((movie, index) => (
            <Text key={index}>
              Movie Name : {movie.Title} {"\n"}
              Genres : {movie.Genre}{"\n"}
              <Text>
                <GoogleSearchLink searchText={`${movie.Title}`} />
              </Text>
              {"\n\n"}
              </Text>
          ))
        ) : (
          <Text>No recommendations available</Text>
        )}
      </Text>

     </View>
    </ScrollView>
  );
};

const styles = StyleSheet.create({
  container: {
    padding: 20,
  },
  input: {
    height: 50,
    borderColor: 'gray',
    borderWidth: 1,
    paddingLeft: 10,
    paddingRight: 10,
  },
  button: {
    margin: 30,
    paddingLeft: 70,
    paddingRight: 70,
  },
  inputContainer: {
    flexDirection: 'row',
  },
  movieInput: {
    marginRight: 5,
    marginBottom: 10,
    width: 200,
  },
  ratingInput: {
    marginLeft: 5,
    marginBottom: 10,
    width: 100,
  },
    space: {
      marginBottom: 10,
      fontWeight: 'bold',
    },
    space1: {
      fontWeight: 'bold',
      marginBottom: 10,
      marginLeft:5,
      marginRight:5
    },
    dynamicinput: {
      alignItems: 'center',
    },
    scrollview:{
      marginBottom:10,
    }
});

export default Moviematebody;
```

FrontEnd:



2. Then we have written the Backend code under the Backend.

app.py

```
from flask import Flask, request
import subprocess
import json
import os
import pandas as pd
from flask import jsonify  # Import jsonify
from flask_cors import CORS,cross_origin

app = Flask(__name__)
CORS(app, resources={r"/*": {"origins": "*"}})

@app.route('/listofmovies',methods=['GET'])
def list_of_movies():
    try:
        df_movies = pd.read_csv("df_movies.csv")

        # Convert DataFrame to a list of dictionaries
        movies_list = df_movies.to_dict(orient='records')

        # # Modify each dictionary to include 'id' and 'name' keys
        # for movie in movies_list:
        #     movie['Title'] = movie.pop('Title')  # Assuming 'Title' is the column name for movie names

        return jsonify(movies_list)
    except Exception as e:
        return f"An error occurred: {str(e)}", 500



@app.route('/moviename', methods=['POST'])
def process():
    try:
        data = request.get_json()
        data_str = json.dumps(data)
         # Run the model.py script
        result = subprocess.run(['python3', 'model.py', data_str], capture_output=True, text=True)
        print(data_str)
        # Check if the subprocess was successful
        if result.returncode == 0:
            output = result.stdout.strip()
            return output
        else:
            error_output = f"Error executing model.py: {result.stderr.strip()}"
            return error_output, 500
    except Exception as e:
        return f"An error occurred: {str(e)}", 500

if __name__ == '__main__':
    app.run(host = '0.0.0.0', port = 5000, debug=True)
```

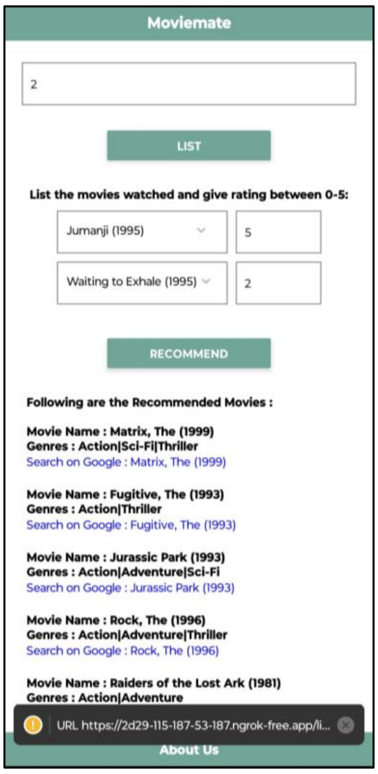
model.py

```
import sys
import json
import pandas as pd
import numpy as np



def pearson_correlation(user_ratings, other_user_ratings):
    # Mean of user ratings
    mean_user = np.mean(user_ratings)
    mean_other_user = np.mean(other_user_ratings)

    # Compute covariance
    cov = np.sum((user_ratings - mean_user) * (other_user_ratings - mean_other_user))

    # Compute standard deviations
    std_user = np.sqrt(np.sum((user_ratings - mean_user) ** 2))
    std_other_user = np.sqrt(np.sum((other_user_ratings - mean_other_user) ** 2))

    # Compute Pearson correlation coefficient
    pearson_corr = cov / (std_user * std_other_user)

    return pearson_corr
def cosine_similarity(user_ratings, other_user_ratings):
    # Compute dot product between the two vectors
    dot_product = np.dot(user_ratings, other_user_ratings)

    # Compute magnitudes of each vector
    magnitude_user = np.sqrt(np.sum(user_ratings ** 2))
    magnitude_other_user = np.sqrt(np.sum(other_user_ratings ** 2))

    # Compute cosine similarity
    cosine_sim = dot_product / (magnitude_user * magnitude_other_user)

    return cosine_sim
# Function to convert a DataFrame row to a JSON object
def row_to_json(row):
    return {
        'Title': row[1]['Title'],
        'Genre': row[1]['Genres'],

    }

# Function to send top k recommended movies of a genre as JSON
def send_top_k_movies(top_recommended_movies_genre, k):
    # Initialize an empty list to store JSON objects
    json_objects = []

    # Iterate over the first k rows of top_recommended_movies_genre
    for row in top_recommended_movies_genre.head(k).iterrows():
        # Convert the row to a JSON object
        json_obj = row_to_json(row)
        # Append the JSON object to the list
        json_objects.append(json_obj)

    # Serialize the list of JSON objects to JSON format
    json_data = json.dumps(json_objects)

    # Return the JSON data
    return json_data


def main():


    df_new= pd.read_csv("merged_data.csv")
    df_movies= pd.read_csv("df_movies.csv")



    user_movie_matrix = df_new.pivot_table(index='UserID',columns='Title',values='Rating').fillna(0)


    new_row_index = len(user_movie_matrix)  # Index for the new row
    new_row_values = [0] * len(user_movie_matrix.columns)  # Values for the new row

    # Appending the new row to the DataFrame
    user_movie_matrix.loc[new_row_index+1] = new_row_values

    index_number = 6041


    # Read the serialized JSON data from command line arguments
    data_str = sys.argv[1]

    # Deserialize the JSON data
    data = json.loads(data_str)



    # Iterate through the array of objects and construct movie information
    for item in data:
        movie_title = item.get('movie_title')

        rating = float(item.get('rating'))

        user_input_value = movie_title
        user_rating = rating
        user_movie_matrix.at[index_number, user_input_value] = user_rating

    user_movie_matrix_norm = user_movie_matrix.subtract(user_movie_matrix.mean(axis=1), axis =
'rows')
```

```
 # Choose a particular user
 user_id = 6041

 # Select the row corresponding to the chosen user
 user_ratings = user_movie_matrix_norm.iloc[user_id-1]

 # Compute Pearson correlation with other users

 user_similarity = {}
 for other_user_id, other_user_ratings in user_movie_matrix_norm.iterrows():
    if other_user_id != user_id:
       similarity = pearson_correlation(user_ratings, other_user_ratings)
       user_similarity[other_user_id] = similarity



 # Remove the user's own similarity score from the dictionary
 user_similarity.pop(user_id, None)

 # Convert the dictionary to a pandas Series
 user_similarity = pd.Series(user_similarity)
 # Take a look at the data
 user_similarity.head()

 # Number of similar users
 n = 10
 # Get top n similar users
 similar_users = user_similarity.sort_values(ascending=False)[:n]

 similar_user_movies =
user_movie_matrix_norm[user_movie_matrix_norm.index.isin(similar_users.index)].dropna(axis=1,
how='all')

 # A dictionary to store movie scores
 item_score = {}

 # Loop through the similar movies
 for i in similar_user_movies.columns:

    # Retrieve the ratings for movie i
    movie_rating = similar_user_movies[i]

    # Create a variable to store the movie score
    total = 0

    # Create a variable to store the number of scores
    count = 0

    # Loop through similar users
    for u in similar_users.index:

       # Score is the sum of user similarity score multiply by the movie rating
       score = similar_users[u] * movie_rating[u]
```

```
            # Add the score to the total score for the movie so far
            total += score
            count +=1

        # Get the average score for the item
        item_score[i] = total / count # Formula: Summation(user_similarity_value * rating given by the user)/total number of similar users.

    # Convert dictionary to pandas dataframe
    item_score = pd.DataFrame(item_score.items(), columns=['movie', 'movie_score'])

    # Sort the movies by score
    top_recommended_movies = item_score.sort_values(by='movie_score', ascending=False)

    top_recommended_movies.rename(columns={'movie': 'Title'}, inplace=True)

    temp_df = top_recommended_movies[['Title']].copy()

    top_recommended_movies_genre = pd.merge(temp_df, df_movies, on='Title')


    top_recommended_movies_genre.drop(columns=['MovieID'], inplace=True)


    # Select top k movies
    k = 20
    # top_recommended_movies_genre.head(k)

    json_data = send_top_k_movies(top_recommended_movies_genre, k)

    # Now you can return json_data or use it as needed
    print(json_data)



if __name__ == "__main__":
    main()
```

<u>Backend response displayed on the mobile app:</u>



Moviemate

2

List the movies watched and give rating between 0-5:

Jumanji (1995)	5
Waiting to Exhale (1995)	2

Following are the Recommended Movies :

Movie Name : Matrix, The (1999)
Genres : Action|Sci-Fi|Thriller
Search on Google : Matrix, The (1999)

Movie Name : Fugitive, The (1993)
Genres : Action|Thriller
Search on Google : Fugitive, The (1993)

Movie Name : Jurassic Park (1993)
Genres : Action|Adventure|Sci-Fi
Search on Google : Jurassic Park (1993)

Movie Name : Rock, The (1996)
Genres : Action|Adventure|Thriller
Search on Google : Rock, The (1996)

Movie Name : Raiders of the Lost Ark (1981)
Genres : Action|Adventure

URL https://2d29-115-187-53-187.ngrok-free.app/li...

<u>Test cases for Backend:</u>

We have written the test cases for backend under the test_backend.py

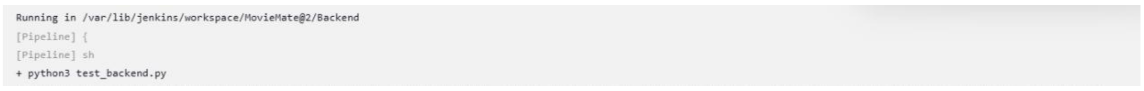
<u>test_backend.py</u>

```
import unittest
import requests
import json
from app import app

class TestBackend(unittest.TestCase):

  def test_list_of_movies(self):
    response = requests.get('http://127.0.0.1:5000/listofmovies')
    self.assertEqual(response.status_code, 200)
    data = response.json()
    print("Response from /listofmovies endpoint:", data)
    self.assertIsInstance(data, list)
    self.assertTrue(len(data) > 0)
  def test_process(self):
    payload = json.dumps(
      [{
      "movie": "Titan A.E. (2000)",
      "rating":"5"
      }]
    )
    headers = {'Content-Type': 'application/json'}
```

```
        response = requests.post('http://127.0.0.1:5000/moviename', data=payload, headers=headers)
        self.assertEqual(response.status_code, 200)
        print("Response from /moviename endpoint:", response.text)


if __name__ == '__main__':
    unittest.main()
```

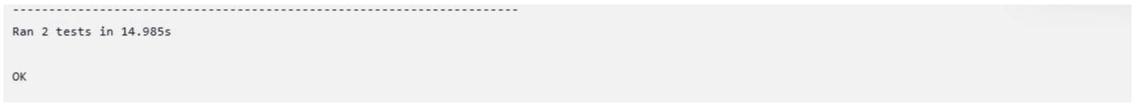
Output:

We will write the command:



```
Running in /var/lib/jenkins/workspace/MovieMate@2/Backend
[Pipeline] {
[Pipeline] sh
+ python3 test_backend.py
```

Response got:



```
----------------------------------------------------------------------
Ran 2 tests in 14.985s

OK
```

Using Jenkins for Continuous Integration

Here is the steps to install Jenkins in ubuntu:

1. First we will update the package list:

```
sudo apt update
```

2. We need to install Java:
   Jenkins requires Java to run, so you'll need to install it if it's not already on your system.
   We will run the following command to install OpenJDK 11:

```
sudo apt install openjdk-11-jdk
```

3. We will add the Jenkins repository key:

```
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
```

4. We have to add the Jenkins repository to our system:

```
sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
```

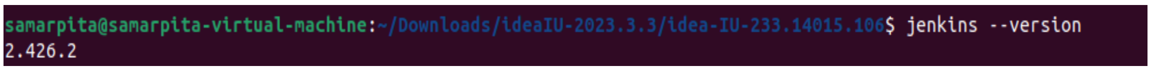
5. Now we will update the package list again:

```
sudo apt update
```

6. We have to install the Jenkins:

```
sudo apt install jenkins
```

7. We will check the Jenkins version:



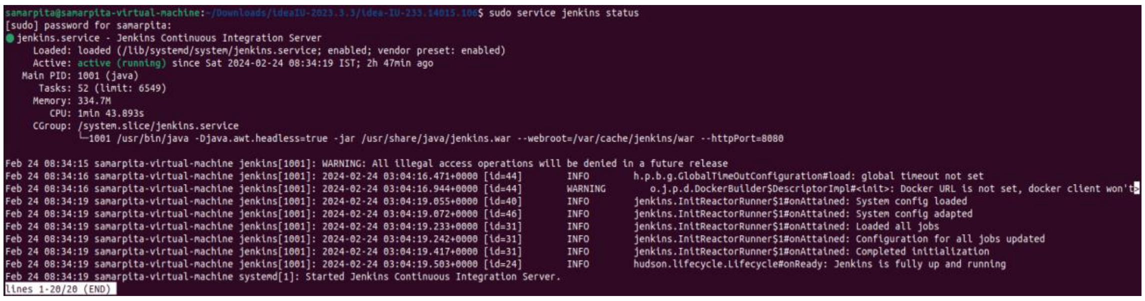
```
samarpita@samarpita-virtual-machine:~/Downloads/ideaIU-2023.3.3/idea-IU-233.14015.106$ jenkins --version
2.426.2
```

8. Starting the Jenkins service:

```
sudo service jenkins start
```

9. We will check the status of the Jenkins service:

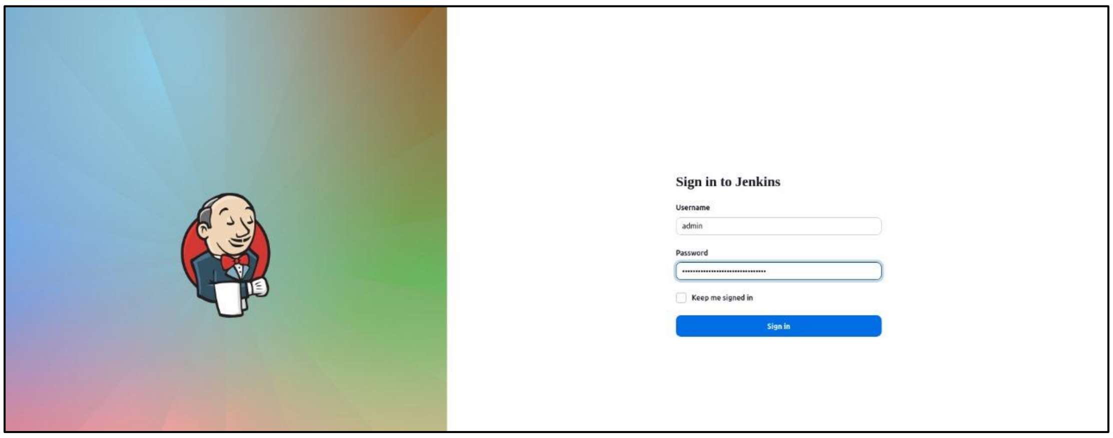
```
sudo service jenkins status
```



```
samarpita@samarpita-virtual-machine:~/Downloads/ideaIU-2023.3.3/idea-IU-233.14015.106$ sudo service jenkins status
[sudo] password for samarpita:
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2024-02-24 08:34:19 IST; 2h 47min ago
   Main PID: 1001 (java)
      Tasks: 52 (limit: 6549)
     Memory: 334.7M
        CPU: 1min 43.893s
     CGroup: /system.slice/jenkins.service
             └─1001 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Feb 24 08:34:15 samarpita-virtual-machine jenkins[1001]: WARNING: All illegal access operations will be denied in a future release
Feb 24 08:34:16 samarpita-virtual-machine jenkins[1001]: 2024-02-24 03:04:16.471+0000 [id=44]       INFO       h.p.b.g.GlobalTimeOutConfiguration#load: global timeout not set
Feb 24 08:34:16 samarpita-virtual-machine jenkins[1001]: 2024-02-24 03:04:16.944+0000 [id=44]       WARNING      o.j.p.d.DockerBuilder$DescriptorImpl#<init>: Docker URL is not set, docker client won't>
Feb 24 08:34:19 samarpita-virtual-machine jenkins[1001]: 2024-02-24 03:04:19.055+0000 [id=40]       INFO       jenkins.InitReactorRunner$1#onAttained: System config loaded
Feb 24 08:34:19 samarpita-virtual-machine jenkins[1001]: 2024-02-24 03:04:19.072+0000 [id=46]       INFO       jenkins.InitReactorRunner$1#onAttained: System config adapted
Feb 24 08:34:19 samarpita-virtual-machine jenkins[1001]: 2024-02-24 03:04:19.233+0000 [id=31]       INFO       jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
Feb 24 08:34:19 samarpita-virtual-machine jenkins[1001]: 2024-02-24 03:04:19.242+0000 [id=31]       INFO       jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
Feb 24 08:34:19 samarpita-virtual-machine jenkins[1001]: 2024-02-24 03:04:19.417+0000 [id=31]       INFO       jenkins.InitReactorRunner$1#onAttained: Completed initialization
Feb 24 08:34:19 samarpita-virtual-machine jenkins[1001]: 2024-02-24 03:04:19.503+0000 [id=24]       INFO       hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
Feb 24 08:34:19 samarpita-virtual-machine systemd[1]: Started Jenkins Continuous Integration Server.
lines 1-20/20 (END)
```

Since the service is running, we can see the **Active: "active (running)".**

10. **Open our web browser and go to http://localhost:8080. We can see the Jenkins login page. We have to provide the following credentials and enter the Jenkins system.**



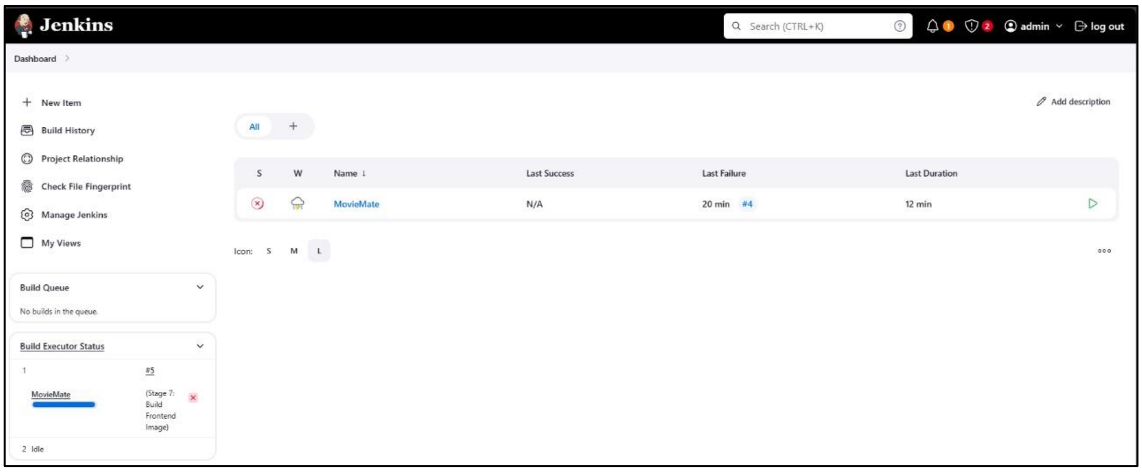
Sign in to Jenkins

Username: admin

Password: ••••••••••••••••••••••••••••••••

Keep me signed in

After logging into Jenkins:



S	W	Name ↓	Last Success	Last Failure	Last Duration
		MovieMate	N/A	20 min #4	12 min

Build Queue

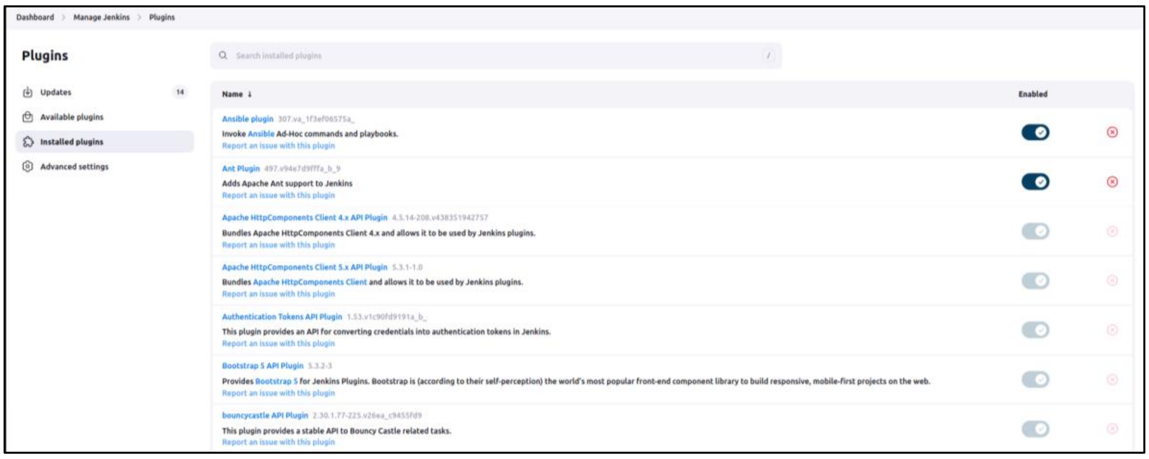
No builds in the queue.

Build Executor Status

1 #5 MovieMate (Stage 7: Build Frontend Image)

2 Idle

11. Now we have to install all the plugins that are required:
    Install : **Maven, Git, Ansible, Docker** plugins from Plugin Manager.



Dashboard > Manage Jenkins > Plugins

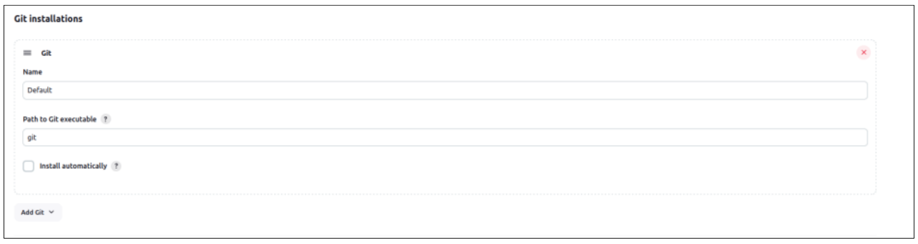
Plugins

Search installed plugins

- Updates 14
- Available plugins
- Installed plugins
- Advanced settings

Name ↓	Enabled
Ansible plugin 307.va_1f3ef06575a_ Invoke Ansible Ad-Hoc commands and playbooks. Report an issue with this plugin	✓
Ant Plugin 497.v94e7d9fffa_b_9 Adds Apache Ant support to Jenkins Report an issue with this plugin	✓
Apache HttpComponents Client 4.x API Plugin 4.5.14-208.v438351942757 Bundles Apache HttpComponents Client 4.x and allows it to be used by Jenkins plugins. Report an issue with this plugin	✓
Apache HttpComponents Client 5.x API Plugin 5.3.1-1.0 Bundles Apache HttpComponents Client and allows it to be used by Jenkins plugins. Report an issue with this plugin	✓
Authentication Tokens API Plugin 1.53.v1c90fd9191a_b_ This plugin provides an API for converting credentials into authentication tokens in Jenkins. Report an issue with this plugin	✓
Bootstrap 5 API Plugin 5.3.2-3 Provides Bootstrap 5 for Jenkins Plugins. Bootstrap is (according to their self-perception) the world's most popular front-end component library to build responsive, mobile-first projects on the web. Report an issue with this plugin	✓
bouncycastle API Plugin 2.30.1.77-225.v26ea_c9455fd9 This plugin provides a stable API to Bouncy Castle related tasks. Report an issue with this plugin	✓

12. Add Global Tool Configuration from Manage Jenkins.



Git installations

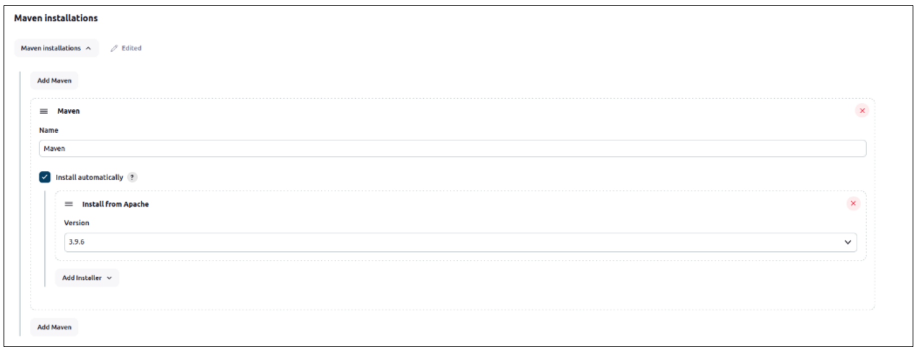
Git

Name: Default

Path to Git executable: git

☐ Install automatically

Add Git



Maven installations

Maven installations ^ Edited

Add Maven

Maven

Name: Maven

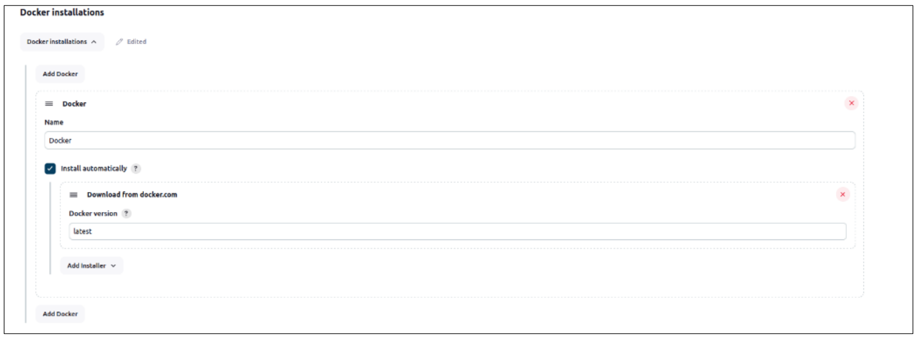
☑ Install automatically

Install from Apache

Version: 3.9.6

Add Installer

Add Maven



Docker installations

Docker installations ^ Edited

Add Docker

Docker

Name: Docker

☑ Install automatically

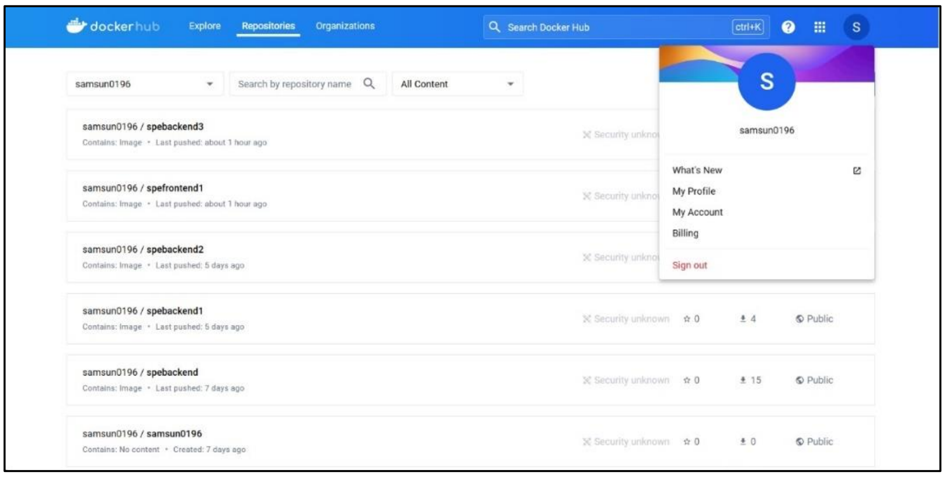
Download from docker.com

Docker version: latest

Add Installer

Add Docker

13. Now we will go to Docker hub and create an account.


docker hub · Explore · Repositories · Organizations · Search Docker Hub · ctrl+K

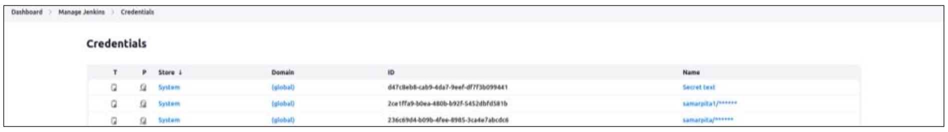
samsun0196 · Search by repository name · All Content

Repository	Details	Security	Stars	Pulls	Visibility
samsun0196 / spebackend3	Contains: Image • Last pushed: about 1 hour ago	Security unkno			
samsun0196 / spefrontend1	Contains: Image • Last pushed: about 1 hour ago	Security unkno			
samsun0196 / spebackend2	Contains: Image • Last pushed: 5 days ago	Security unkno			
samsun0196 / spebackend1	Contains: Image • Last pushed: 5 days ago	Security unknown	0	4	Public
samsun0196 / spebackend	Contains: Image • Last pushed: 7 days ago	Security unknown	0	15	Public
samsun0196 / samsun0196	Contains: No content • Created: 7 days ago	Security unknown	0	0	Public

samsun0196 · What's New · My Profile · My Account · Billing · Sign out

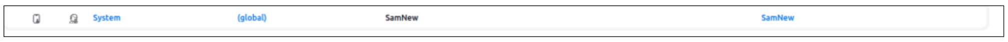
This are our Docker hub credentials.

14. Now we will go to **Credentials in Jenkins**:


Dashboard > Manage Jenkins > Credentials

Credentials

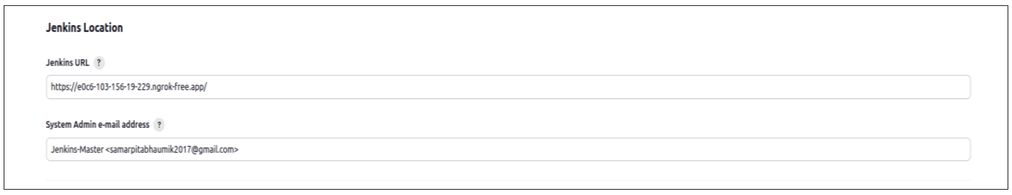
T	P	Store ↓	Domain	ID	Name
		System	(global)	d47c8eb8-cab9-4da7-9eef-df7f3b099441	Secret text
		System	(global)	2ce1ffa9-b0ea-480b-b92f-5452dbfd581b	samarpita1/******
		System	(global)	236c69d4-b09b-4fee-8985-3ca4e7abcdc6	samarpita/******



		System	(global)	SamNew	SamNew

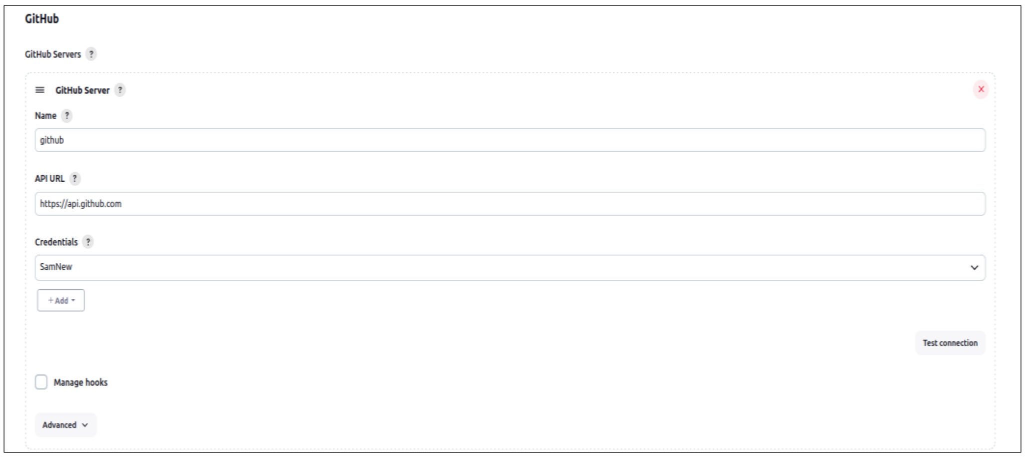
We will add Dockerhub and Github Credentials for Jenkins access.

15. Now we will go to Manage Jenkins -> System.


Jenkins Location

Jenkins URL: https://e0c6-103-156-19-229.ngrok-free.app/

System Admin e-mail address: Jenkins-Master <samarpitabhaumik2017@gmail.com>


GitHub

GitHub Servers

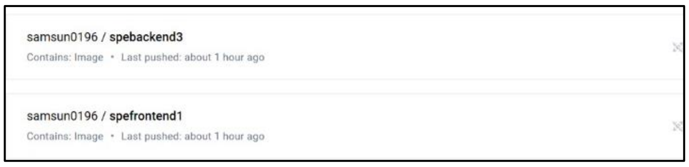
GitHub Server

Name: github

API URL: https://api.github.com

Credentials: SamNew

We will a create two repositories in Docker Hub named “**spebackend3**” and “**spefrontend1**”


samsun0196 / spebackend3
Contains: Image • Last pushed: about 1 hour ago

samsun0196 / spefrontend1
Contains: Image • Last pushed: about 1 hour ago

Containerising Our Code

We will be using docker to containerising the code. Installing Docker I have followed the instructions given in this.

sudo apt-get update: This command updates the package lists for upgrades and new package installations. It ensures that your system has the latest information about available packages.

sudo apt-get -y install docker-ce: This command installs Docker Community Edition (CE) on your system. The -y flag is used to automatically confirm the installation without prompting for confirmation.

sudo usermod -aG docker $USER: This command adds your user to the docker group. Users in the docker group have permission to interact with the Docker daemon. Adding yourself to this group allows you to run Docker commands without needing to use sudo each time.

newgrp docker: This command starts a new shell session with the docker group's permissions activated. After running this command, you should be able to execute Docker commands without using sudo.

After running these commands, you should have Docker installed on your system

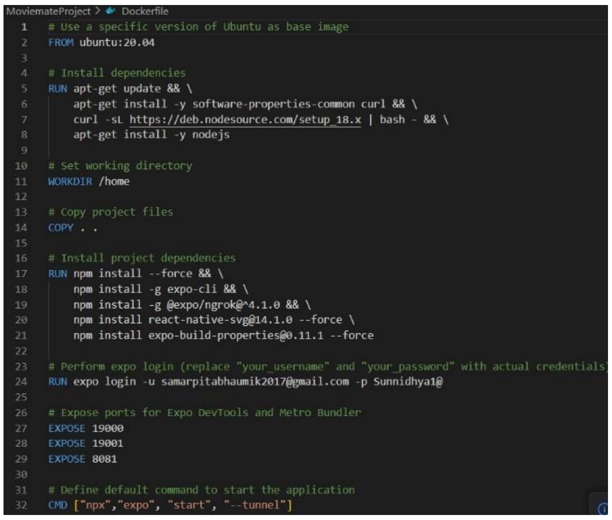
Checking the docker installation by using:

```
docker --version
```

Now we will see how to create the Docker file, which will be used to create docker image. And this docker image can be used to create docker containers which will be used to deploy our code onto local machine.

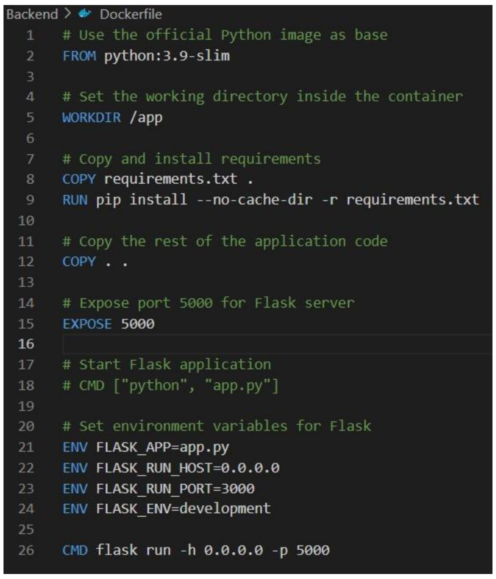
We will create two **“Dockerfile”** in our project folder. One for Frontend and another for Backend.

DockerFile for Frontend:



```
# Use a specific version of Ubuntu as base image
FROM ubuntu:20.04

# Install dependencies
RUN apt-get update && \
    apt-get install -y software-properties-common curl && \
    curl -sL https://deb.nodesource.com/setup_18.x | bash - && \
    apt-get install -y nodejs

# Set working directory
WORKDIR /home

# Copy project files
COPY . .

# Install project dependencies
RUN npm install --force && \
    npm install -g expo-cli && \
    npm install -g @expo/ngrok@^4.1.0 && \
    npm install react-native-svg@14.1.0 --force \
    npm install expo-build-properties@0.11.1 --force

# Perform expo login (replace "your_username" and "your_password" with actual credentials
RUN expo login -u samarpitabhaumik2017@gmail.com -p Sunnidhya1@

# Expose ports for Expo DevTools and Metro Bundler
EXPOSE 19000
EXPOSE 19001
EXPOSE 8081

# Define default command to start the application
CMD ["npx","expo", "start", "--tunnel"]
```

DockerFile for Backend



Backend > Dockerfile

```
# Use the official Python image as base
FROM python:3.9-slim

# Set the working directory inside the container
WORKDIR /app

# Copy and install requirements
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application code
COPY . .

# Expose port 5000 for Flask server
EXPOSE 5000

# Start Flask application
# CMD ["python", "app.py"]

# Set environment variables for Flask
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0.0
ENV FLASK_RUN_PORT=3000
ENV FLASK_ENV=development

CMD flask run -h 0.0.0.0 -p 5000
```

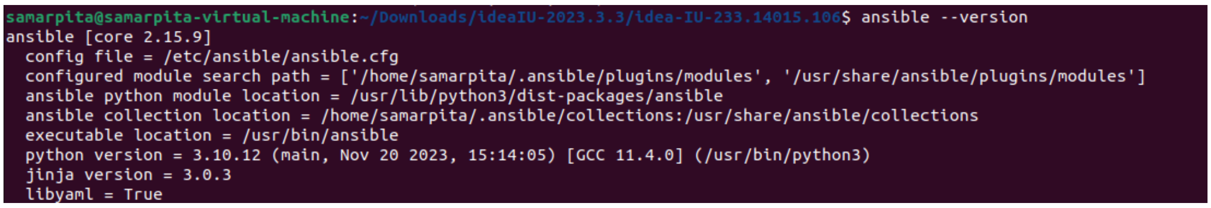
Deployment

Setting up Ansible

Ansible Installation:
Installing Ansible on a Linux system is a straightforward process. Execute the following command.

• **sudo apt update**
• **sudo apt install Ansible**

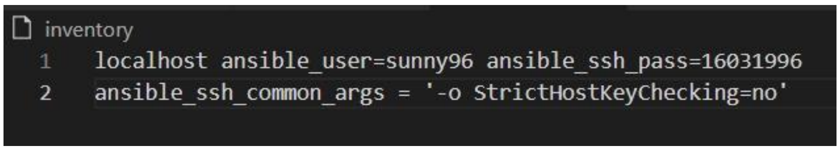
After the installation is complete, you can verify it by checking the
Ansible version:



```
samarpita@samarpita-virtual-machine:~/Downloads/ideaIU-2023.3.3/idea-IU-233.14015.106$ ansible --version
ansible [core 2.15.9]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/samarpita/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/samarpita/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] (/usr/bin/python3)
  jinja version = 3.0.3
  libyaml = True
```

We will Create "Inventory" file under Moviemate

Inventories: Inventories are files that list the hosts (managed nodes) and group them into categories. This categorization allows for easier management of resources and targeted execution of tasks on specific groups.

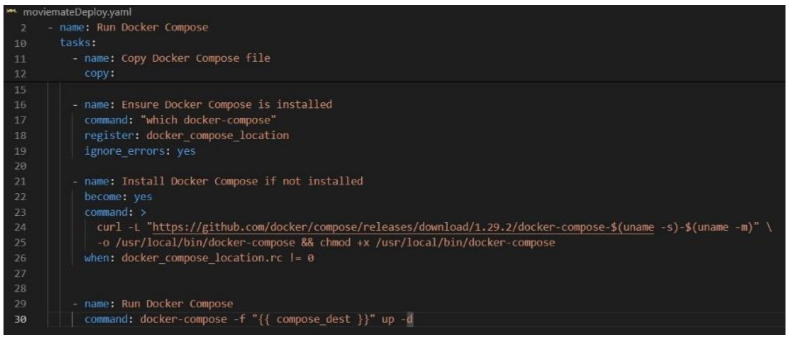


inventory

```
localhost ansible_user=sunny96 ansible_ssh_pass=16031996
ansible_ssh_common_args = '-o StrictHostKeyChecking=no'
```

moviemateDeploy.yaml

This Ansible playbook automates running a Docker Compose file on localhost. It first copies the specified Docker Compose file to the local machine. Then, it checks if Docker Compose is installed by running the **which docker-compose** command and registering its location. If Docker Compose is not found, it installs the latest version using **curl** and grants execution permissions. Finally, the playbook uses the copied Docker Compose file to bring up the defined services with **docker-compose up -d**. This setup ensures that Docker Compose is available and properly configured before attempting to run the services.

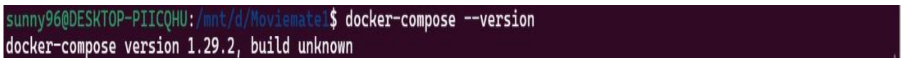


moviemateDeploy.yaml

```yaml
  - name: Run Docker Compose
    tasks:
      - name: Copy Docker Compose file
        copy:

      - name: Ensure Docker Compose is installed
        command: "which docker-compose"
        register: docker_compose_location
        ignore_errors: yes

      - name: Install Docker Compose if not installed
        become: yes
        command: >
          curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" \
          -o /usr/local/bin/docker-compose && chmod +x /usr/local/bin/docker-compose
        when: docker_compose_location.rc != 0


      - name: Run Docker Compose
        command: docker-compose -f "{{ compose_dest }}" up -d
```

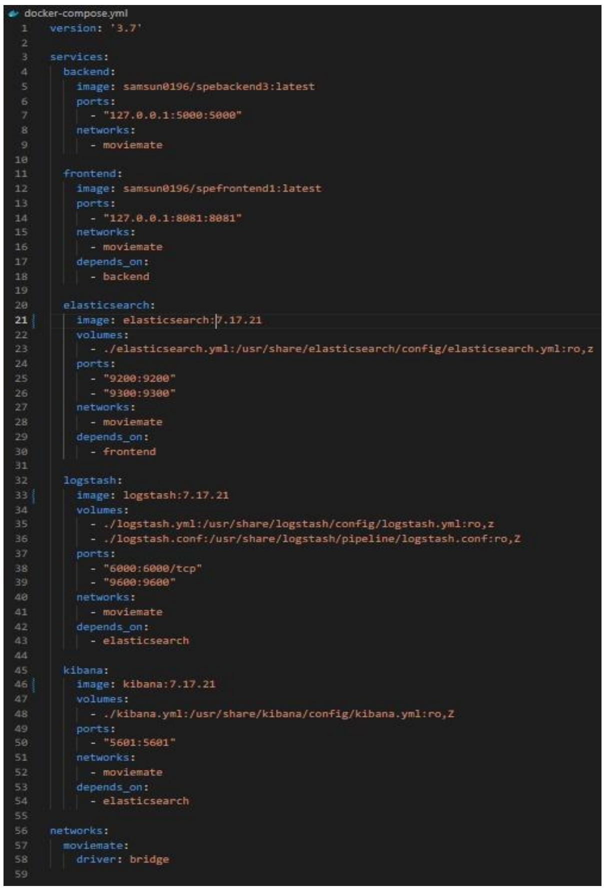
Docker orchestration ,Monitoring and Logging

Using Docker Compose:



```
sunny96@DESKTOP-PIICQHU:/mnt/d/Moviemate1$ docker-compose --version
docker-compose version 1.29.2, build unknown
```

docker-compose.yml



docker-compose.yml

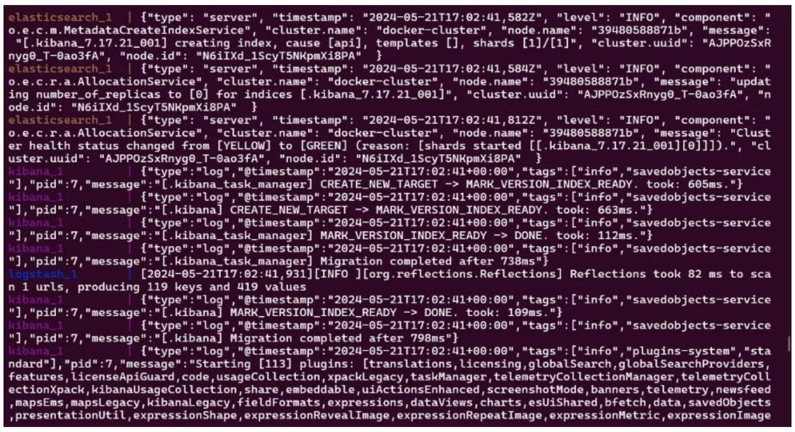
```yaml
version: '3.7'

services:
  backend:
    image: samsun0196/spebackend3:latest
    ports:
      - "127.0.0.1:5000:5000"
    networks:
      - moviemate

  frontend:
    image: samsun0196/spefrontend1:latest
    ports:
      - "127.0.0.1:8081:8081"
    networks:
      - moviemate
    depends_on:
      - backend

  elasticsearch:
    image: elasticsearch:7.17.21
    volumes:
      - ./elasticsearch.yml:/usr/share/elasticsearch/config/elasticsearch.yml:ro,z
    ports:
      - "9200:9200"
      - "9300:9300"
    networks:
      - moviemate
    depends_on:
      - frontend

  logstash:
    image: logstash:7.17.21
    volumes:
      - ./logstash.yml:/usr/share/logstash/config/logstash.yml:ro,z
      - ./logstash.conf:/usr/share/logstash/pipeline/logstash.conf:ro,Z
    ports:
      - "6000:6000/tcp"
      - "9600:9600"
    networks:
      - moviemate
    depends_on:
      - elasticsearch

  kibana:
    image: kibana:7.17.21
    volumes:
      - ./kibana.yml:/usr/share/kibana/config/kibana.yml:ro,Z
    ports:
      - "5601:5601"
    networks:
      - moviemate
    depends_on:
      - elasticsearch

networks:
  moviemate:
    driver: bridge

```

This Docker Compose configuration defines a multi-service application composed of several interdependent services, which are interconnected through a network called "**moviemate**". Each service runs in its own container, specified by the Docker images and configurations provided.

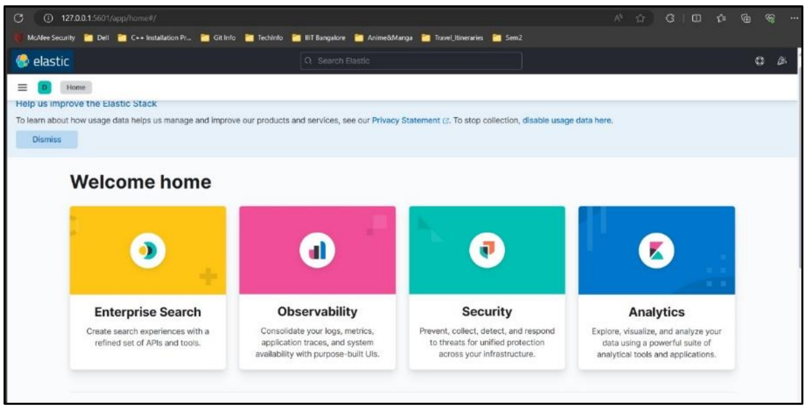
The backend service is based on the image **samsun0196/spebackend3:latest** and listens on port 5000, but it is only accessible from the local machine at 127.0.0.1:5000. It connects to the moviemate network for communication with other services. The frontend service uses the image **samsun0196/spefrontend1:latest** and listens on port 8081, also restricted to local access (127.0.0.1:8081). It depends on the backend service, ensuring the backend starts first.

For data indexing and search, the elasticsearch service uses the image **elasticsearch:7.17.21**, with its configuration provided through a mounted elasticsearch.yml file. It exposes ports **9200 and 9300** to the host and depends on the frontend service. The logstash service uses the **logstash:7.17.21** image and is configured with **logstash.yml** and **logstash.conf** files, mapping its internal ports **6000** and **9600** to the host. It depends on the elasticsearch service.

Lastly, the kibana service, using the **kibana:7.17.21** image, is configured through **kibana.yml** and exposes port **5601** to the host. It also depends on the elasticsearch service, ensuring Elasticsearch is up before Kibana starts. All services are connected through the moviemate network, which is defined as a bridge network, facilitating inter-container communication.



```
elasticsearch_1  | {"type": "server", "timestamp": "2024-05-21T17:02:41,582Z", "level": "INFO", "component": "
o.e.c.m.MetadataCreateIndexService", "cluster.name": "docker-cluster", "node.name": "39480588871b", "message":
 "[.kibana_7.17.21_001] creating index, cause [api], templates [], shards [1]/[1]", "cluster.uuid": "AJPPOzSxR
nyg0_T-0ao3fA", "node.id": "N6iIXd_1ScyT5NKpmXi8PA"  }
elasticsearch_1  | {"type": "server", "timestamp": "2024-05-21T17:02:41,584Z", "level": "INFO", "component": "
o.e.c.r.a.AllocationService", "cluster.name": "docker-cluster", "node.name": "39480588871b", "message": "updat
ing number_of_replicas to [0] for indices [.kibana_7.17.21_001]", "cluster.uuid": "AJPPOzSxRnyg0_T-0ao3fA", "n
ode.id": "N6iIXd_1ScyT5NKpmXi8PA"  }
elasticsearch_1  | {"type": "server", "timestamp": "2024-05-21T17:02:41,812Z", "level": "INFO", "component": "
o.e.c.r.a.AllocationService", "cluster.name": "docker-cluster", "node.name": "39480588871b", "message": "Clust
er health status changed from [YELLOW] to [GREEN] (reason: [shards started [[.kibana_7.17.21_001][0]]]).", "cl
uster.uuid": "AJPPOzSxRnyg0_T-0ao3fA", "node.id": "N6iIXd_1ScyT5NKpmXi8PA"  }
kibana_1         | {"type":"log","@timestamp":"2024-05-21T17:02:41+00:00","tags":["info","savedobjects-service
"],"pid":7,"message":"[.kibana_task_manager] CREATE_NEW_TARGET -> MARK_VERSION_INDEX_READY. took: 605ms."}
kibana_1         | {"type":"log","@timestamp":"2024-05-21T17:02:41+00:00","tags":["info","savedobjects-service
"],"pid":7,"message":"[.kibana] CREATE_NEW_TARGET -> MARK_VERSION_INDEX_READY. took: 663ms."}
kibana_1         | {"type":"log","@timestamp":"2024-05-21T17:02:41+00:00","tags":["info","savedobjects-service
"],"pid":7,"message":"[.kibana_task_manager] MARK_VERSION_INDEX_READY -> DONE. took: 112ms."}
kibana_1         | {"type":"log","@timestamp":"2024-05-21T17:02:41+00:00","tags":["info","savedobjects-service
"],"pid":7,"message":"[.kibana_task_manager] Migration completed after 738ms"}
logstash_1       | [2024-05-21T17:02:41,931][INFO ][org.reflections.Reflections] Reflections took 82 ms to sca
n 1 urls, producing 119 keys and 419 values
kibana_1         | {"type":"log","@timestamp":"2024-05-21T17:02:41+00:00","tags":["info","savedobjects-service
"],"pid":7,"message":"[.kibana] MARK_VERSION_INDEX_READY -> DONE. took: 109ms."}
kibana_1         | {"type":"log","@timestamp":"2024-05-21T17:02:41+00:00","tags":["info","savedobjects-service
"],"pid":7,"message":"[.kibana] Migration completed after 798ms"}
kibana_1         | {"type":"log","@timestamp":"2024-05-21T17:02:41+00:00","tags":["info","plugins-system","sta
ndard"],"pid":7,"message":"Starting [113] plugins: [translations,licensing,globalSearch,globalSearchProviders,
features,licenseApiGuard,code,usageCollection,xpackLegacy,taskManager,telemetryCollectionManager,telemetryColl
ectionXpack,kibanaUsageCollection,share,embeddable,uiActionsEnhanced,screenshotMode,banners,telemetry,newsfeed
,mapsEms,mapsLegacy,kibanaLegacy,fieldFormats,expressions,dataViews,charts,esUiShared,bfetch,data,savedObjects
,presentationUtil,expressionShape,expressionRevealImage,expressionRepeatImage,expressionMetric,expressionImage
```



127.0.0.1:5601/app/home#/

Help us improve the Elastic Stack

To learn about how usage data helps us manage and improve our products and services, see our Privacy Statement. To stop collection, disable usage data here.

Welcome home

Enterprise Search: Create search experiences with a refined set of APIs and tools.

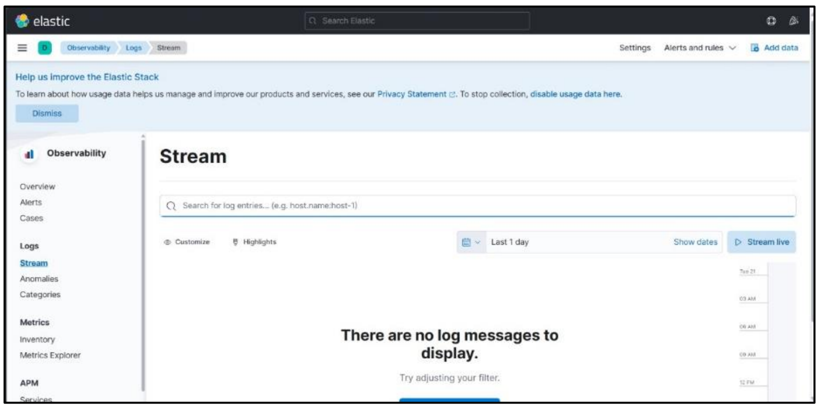
Observability: Consolidate your logs, metrics, application traces, and system availability with purpose-built UIs.

Security: Prevent, collect, detect, and respond to threats for unified protection across your infrastructure.

Analytics: Explore, visualize, and analyze your data using a powerful suite of analytical tools and applications.

$\longrightarrow$

We can see port number 5601 is exposed to the host



Help us improve the Elastic Stack

To learn about how usage data helps us manage and improve our products and services, see our Privacy Statement. To stop collection, disable usage data here.

Stream

There are no log messages to display.

Try adjusting your filter.

$\longrightarrow$

Here we can see the logs which are logged from the app when api is called

Jenkins File

This “**jenkinsfile”** is a Declarative Pipeline script for Jenkins, which orchestrates the build and deployment process.

```
pipeline {
  agent any
  environment {
    DOCKER_IMAGE_NAME_FRONTEND = 'spefrontend1'
    DOCKER_IMAGE_NAME_BACKEND = 'spebackend3'
    GITHUB_REPO_URL = 'https://github.com/samarpita-bhaumik/Moviemate.git'
    GIT_CREDENTIALS = 'mmProj'
    DOCKER_HUB_USERNAME = 'samsun0196'
    DOCKER_HUB_PASSWORD = 'Sr@16031996'
    DOCKER_HUB_CREDENTIALS = 'Dockerhub_creds_samsun'
    LANG = 'en_US.UTF-8'
    LC_ALL = 'en_US.UTF-8'
  }
  stages {
    stage('Checkout') {
      steps {
        // Checkout the repository using GitSCM plugin with specified credentials
        checkout([$class: 'GitSCM', branches: [[name: '*/main']],
          userRemoteConfigs: [[url: env.GITHUB_REPO_URL, credentialsId: env.GIT_CREDENTIALS]]
        ])
      }
    }
    stage('Stage 2: Prepare Environment') {
      steps {
        // sh 'sudo apt-get update'
        sh 'chmod -R +r MoviemateProject Backend'
        sh 'ls -l MoviemateProject'
        sh 'ls -l Backend'
      }
    }

    stage('Install Python and Flask') {
      steps {
        // Update package index and install Python 3 and pip
        sh 'sudo apt update'

        sh 'sudo apt install -y python3 python3-pip'


        // Install Flask using pip

        sh 'pip3 install Flask pandas flask_cors requests'

    }

}


stage('Start Flask Server') {

    steps {

        script {

            // Start the Flask server in the background

            dir('Backend') {

                    sh 'nohup python3 app.py > /dev/null 2>&1 &'

            sleep 40 // Give Flask some time to start (adjust as needed)

                }

        }

    }

}


stage('Backend Testing') {

    steps {

        script {

            try {

                // Change directory to Backend

                dir('Backend') {

                    sh 'python3 test_backend.py'

                }

            } finally {

                // Ensure the Flask server is stopped after tests

                dir('Backend') {

                    sh 'pkill -f "python3 app.py"'

                }

            }

        }

    }

}
```

```
    stage('Setup ngrok') {
      steps {
        script {
          // Ensure the directory exists before writing the ngrok configuration file
          sh 'mkdir -p /var/lib/jenkins/.ngrok2'
          sh 'echo "authtoken: 2cGkRwBjSdrcSHpoKCNgqCqTvY4_2k3ore6rZvxLpw3SDZmwv" >
/var/lib/jenkins/.ngrok2/ngrok.yml'
          sh 'ngrok config upgrade' // Upgrade ngrok configuration file
        }
      }
    }
    stage('Stage 5: Start ngrok') {
      steps {
        script {
          // Start ngrok in the background
          sh 'nohup ngrok http 5000 &'
          sleep 10 // Give ngrok some time to start and provide the URL
        }
      }
    }

    stage('Install jq') {
      steps {
        script {
          // Install jq
          sh 'sudo apt-get update && sudo apt-get install -y jq'
        }
      }
    }

    stage('Get ngrok URL') {
      steps {
        script {
          // Get the ngrok URL
          def ngrokUrl = sh(script: "curl -s http://127.0.0.1:4040/api/tunnels | jq -r .tunnels[0].public_url", returnStdout:
true).trim()
          echo "ngrok URL: ${ngrokUrl}"
```

```
                // Replace the backendUrl in the frontend code
                sh """
                    sed -i 's|http://localhost:5000|${ngrokUrl}|g' MoviemateProject/Moviematebody.js
                """
            }
        }
    }

    stage('Stage 7: Build Frontend Image') {
        steps {
            // Build frontend Docker image
            sh "sudo docker build -t ${DOCKER_IMAGE_NAME_FRONTEND} MoviemateProject"
            sh "sudo docker tag ${DOCKER_IMAGE_NAME_FRONTEND}:latest
${DOCKER_HUB_USERNAME}/${DOCKER_IMAGE_NAME_FRONTEND}:latest"
        }
    }

    stage('Stage 8: Build Backend Image') {
        steps {
            // Build backend Docker image
            sh "sudo docker build -t ${DOCKER_IMAGE_NAME_BACKEND} Backend"
            sh "sudo docker tag ${DOCKER_IMAGE_NAME_BACKEND}:latest
${DOCKER_HUB_USERNAME}/${DOCKER_IMAGE_NAME_BACKEND}:latest"
        }
    }
    stage('Stage 9: Push Image to DockerHub') {
        steps {
            script {
                withCredentials([usernamePassword(credentialsId: env.DOCKER_HUB_CREDENTIALS, usernameVariable:
'DOCKER_HUB_USERNAME', passwordVariable: 'DOCKER_HUB_PASSWORD')]) {
                    // Log in to DockerHub
                    sh "echo ${DOCKER_HUB_PASSWORD} | docker login -u ${DOCKER_HUB_USERNAME} --password-stdin"

                    // Push frontend and backend images to DockerHub
                    sh "docker push ${DOCKER_HUB_USERNAME}/${DOCKER_IMAGE_NAME_FRONTEND}:latest"
                    sh "docker push ${DOCKER_HUB_USERNAME}/${DOCKER_IMAGE_NAME_BACKEND}:latest"
                }
            }
        }
    }


    stage('Stage 10: Clean Docker Images') {
      steps {
        script {
          // Prune unused Docker containers and images
          sh 'docker container prune -f'
          sh 'docker image prune -f'
        }
      }
    }


    stage('Stage 11: Ansible Deployment') {
      steps {
        script {
          // Execute Ansible playbook for deployment
          sh 'ansible-playbook -i inventory.yaml moviemateDeploy.yaml'
        }
      }
    }
  }
}
```

Explanation:

Agent and Environment Configuration:

- Specifies the agent for the pipeline.
- Sets environment variables for Docker image names, GitHub repository, credentials, Docker Hub login, and language settings.

Stage: Checkout:

- Checks out the repository from GitHub using the specified credentials.

Stage: Prepare Environment:

- Sets read permissions for the project directories.
- Lists the contents of the project directories.

Stage: Install Python and Flask:

- Updates the package index.
- Installs Python 3 and pip.
- Installs Flask and other necessary Python packages.

Stage: Start Flask Server:

- Starts the Flask server in the background and waits for it to be ready.

Stage: Backend Testing:

- Runs the backend tests.
- Ensures the Flask server is stopped after testing.

Stage: Setup ngrok:

- Creates the ngrok configuration directory and writes the ngrok auth token.
- Upgrades the ngrok configuration.

Stage: Start ngrok:

- Starts ngrok to expose the Flask server to the internet and waits for it to be ready.

Stage: Install jq:

- Installs the jq tool for JSON processing.

Stage: Get ngrok URL:

- Retrieves the public URL of the ngrok tunnel.
- Replaces the backend URL in the frontend code with the ngrok URL.

Stage: Build Frontend Image:

- Builds the Docker image for the frontend.
- Tags the frontend image with the Docker Hub username.

Stage: Build Backend Image:

- Builds the Docker image for the backend.
- Tags the backend image with the Docker Hub username.

Stage: Push Image to DockerHub:

- Logs into Docker Hub using credentials.
- Pushes both the frontend and backend Docker images to Docker Hub.

Stage: Clean Docker Images:

- Prunes unused Docker containers and images to free up space.

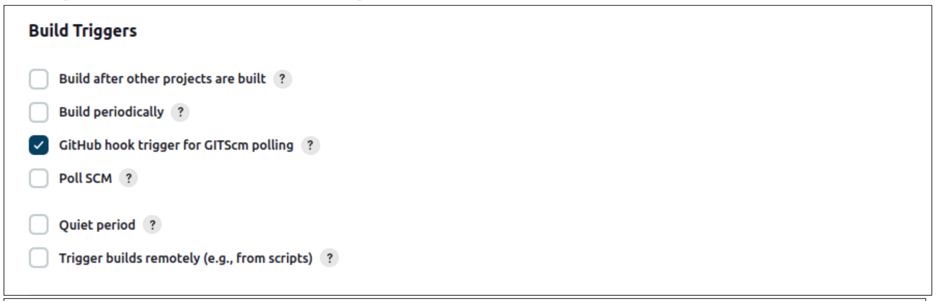
Stage: Ansible Deployment:

- Runs an Ansible playbook to deploy the application using the specified inventory file.

Jenkins Pipeline

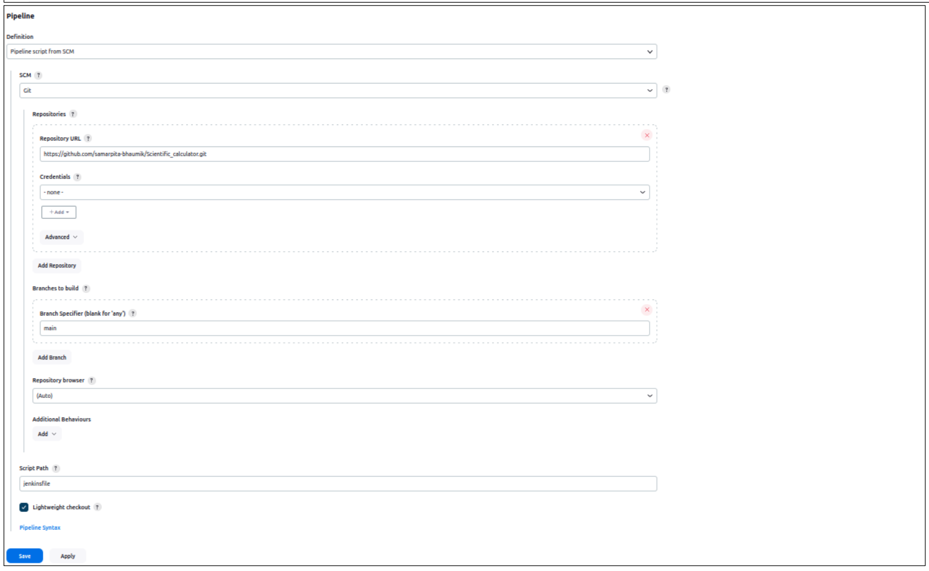
We will now go to Jenkins and follow the steps below.

1. We will select "**New Item**" from the Dashboard.
2. We will enter the Item Name "**MovieMate**" and choose "**Pipeline project**".
3. Now go to "**MovieMate** -> **Configuration**":



Build Triggers

- [ ] Build after other projects are built
- [ ] Build periodically
- [x] GitHub hook trigger for GITScm polling
- [ ] Poll SCM
- [ ] Quiet period
- [ ] Trigger builds remotely (e.g., from scripts)



Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories

Repository URL: https://github.com/samarpita-bhaumik/Scientific_calculator.git

Credentials: - none -

Branches to build

Branch Specifier (blank for 'any'): main

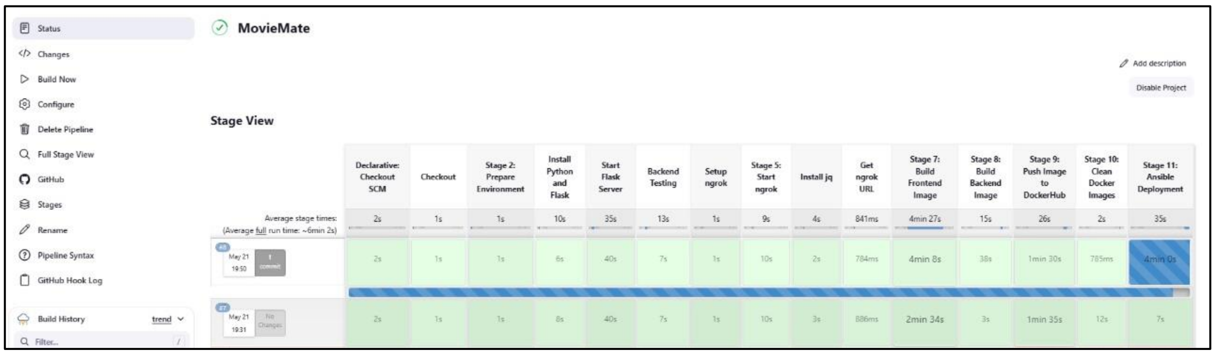
Repository browser: (Auto)

Additional Behaviours

Script Path: jenkinsfile

- [x] Lightweight checkout

4. Now we will "**MovieMate->Build Now**".



MovieMate

Stage View

	Declarative: Checkout SCM	Checkout	Stage 2: Prepare Environment	Install Python and Flask	Start Flask Server	Backend Testing	Setup ngrok	Stage 5: Start ngrok	Install jq	Get ngrok URL	Stage 7: Build Frontend Image	Stage 8: Build Backend Image	Stage 9: Push Image to DockerHub	Stage 10: Clean Docker Images	Stage 11: Ansible Deployment
Average stage times: (Average full run time: ~6min 2s)	2s	1s	1s	10s	35s	13s	1s	9s	4s	841ms	4min 27s	15s	26s	2s	35s
#8 May 21 19:50 1 commit	2s	1s	1s	6s	40s	7s	1s	10s	2s	784ms	4min 8s	38s	1min 30s	785ms	4min 0s
#7 May 21 19:31 No Changes	2s	1s	1s	8s	40s	7s	1s	10s	3s	886ms	2min 34s	3s	1min 35s	12s	7s

Thank You