Pioneering Excellence in Education,
Research & Innovation

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY , BANGALORE

CS816 – SOFTWARE PRODUCTION ENGINEERING

CUSTOMER RELATION MANAGEMENT USING DEVOPS

GUIDED BY:- PROF. B. THANGARAJU
TA:-VISHAL RAI

MADE BY:-
SWAROOP DIXIT
MT2022121
VIPUL MANOHAR AHUJA
MT2022132

GITHUB LINK:- https://github.com/vipul2097/SPECRM.git

Table of Content:-

Project Summary for Customer Relationship Management:

A customer relationship management(CRM) project using Django can be developed to manage interactions with customers, improve business relationships, and ultimately drive sales growth. A basic CRM (Customer Relationship Management) system typically includes several key components to help manage customer interactions and sales processes. These components can vary depending on the specific needs of the business, but some of the most common ones are:

1)Client Database: This is a repository of all the clients or customers of the business. The database typically contains information such as client contact details, demographic information, etc.
2)Leads Management: The lead management module helps track and manage the progress of these potential customers.

Tech Stack

Development:
Language: Python
Backend: Django
Database: MySQL
Frontend: HTML+Tailwind CSS

DevOps:
Version Control: Git and GitHub
CI/CD: Jenkins
Build: Pip(python)
Containerization: Docker + Docker-Compose
Configuration Management: Ansible
Monitoring: ELK Stack
Ngrok: For tunneling the Jenkins from localhost port to the public IP
Testing: Applying Test-Case on Models of each app in Django
Logging: Applied Logging through Django

Important Project Links:
GitHub: https://github.com/vipul2097/SPECRM.git
DockerHub (Image): https://hub.docker.com/repository/docker/vipul2097/spemajorproject/general
DevOps:

What is DevOps?

DevOps is a set of practices that combines software development (Dev) and IT operations (Ops) to shorten the systems development life cycle while delivering features, fixes, and updates frequently and reliably. DevOps emphasizes collaboration and communication between development and operations teams to automate infrastructure deployment, streamline workflows, and improve software quality.Some of the key principles of DevOps include continuous integration and delivery, automated testing, monitoring, and feedback loops. By implementing these practices, DevOps aims to reduce the time between code changes and their deployment to production, increase efficiency and productivity, and enhance customer satisfaction.DevOps also involves the use of tools and technologies such as version control systems, configuration management tools, containerization, and cloud computing. The ultimate goal of DevOps is to create a culture of continuous improvement and innovation, where teams can rapidly respond to changing business needs and deliver value to customers more quickly.

Why DevOps?

DevOps has become increasingly popular over the years because it offers a number of benefits to organizations that adopt it. Here are some of the key reasons why organizations choose to adopt DevOps:

Faster time-to-market: By automating software delivery processes and streamlining workflows, DevOps helps organizations to release new features and updates faster and more frequently. This allows organizations to stay ahead of the competition and respond more quickly to changing customer needs.

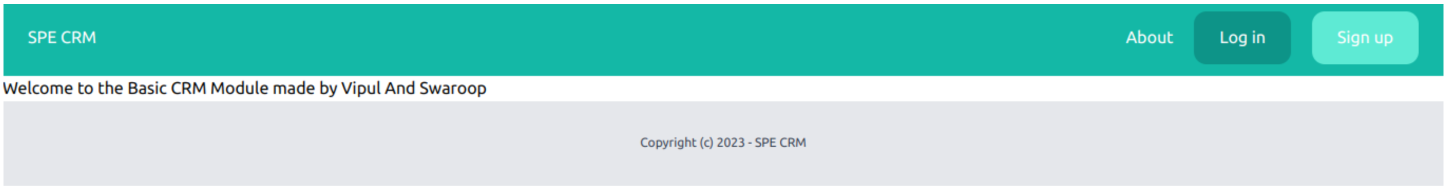
Increased collaboration: DevOps promotes collaboration and communication between development and operations teams, breaking down silos and encouraging cross-functional teamwork. This helps to improve the quality of software and infrastructure and reduce errors caused by miscommunication.

Improved efficiency and productivity: By automating manual processes and reducing the time spent on repetitive tasks, DevOps can improve the efficiency and productivity of development and operations teams. This allows teams to focus on more strategic work and deliver value to the business more quickly.

Better quality and reliability: By integrating testing and quality assurance into the development process, DevOps helps to improve the quality and reliability of software and infrastructure. This can reduce the number of bugs and errors in production and increase customer satisfaction.

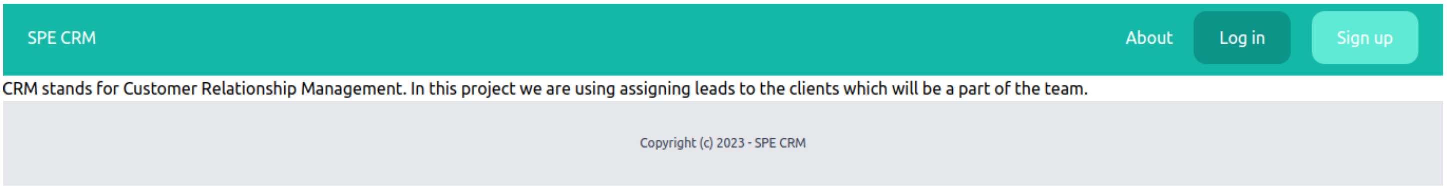
Cost savings: By reducing manual processes, automating tasks, and improving efficiency, DevOps can help organizations to reduce costs associated with software development and operations. This can free up resources for other strategic initiatives and improve the bottom line.

Application Screenshots:



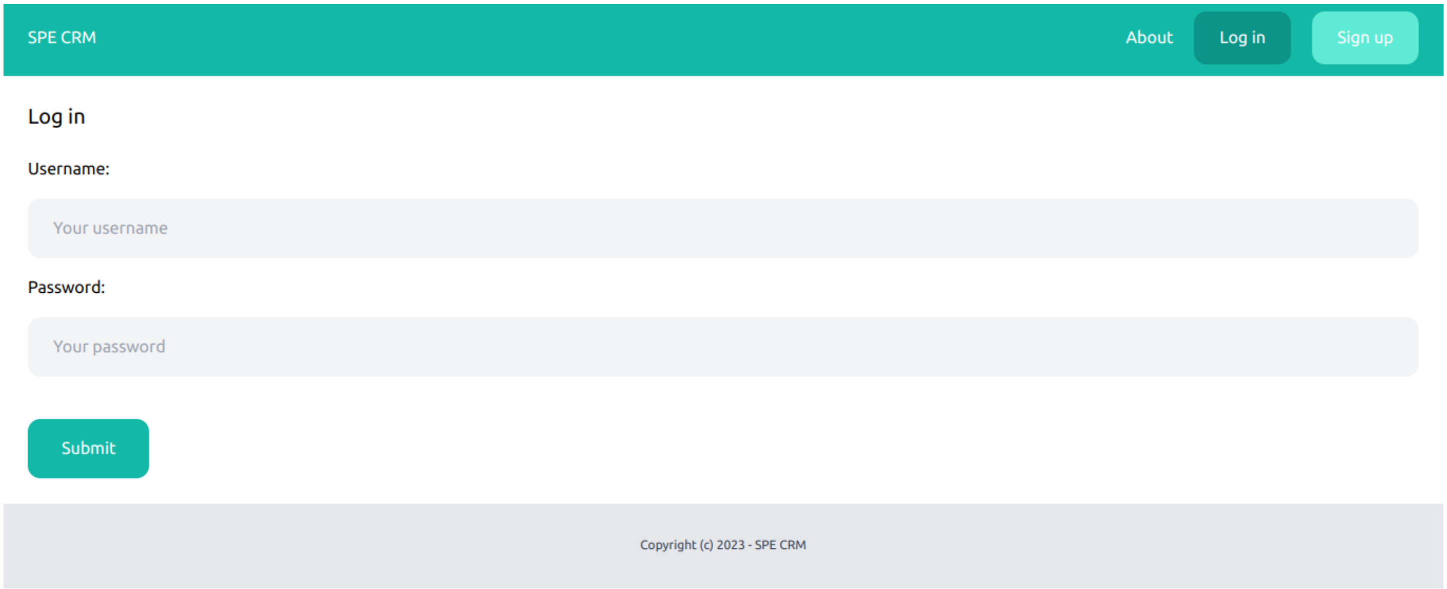
Welcome to the Basic CRM Module made by Vipul And Swaroop

Copyright (c) 2023 - SPE CRM



CRM stands for Customer Relationship Management. In this project we are using assigning leads to the clients which will be a part of the team.

Copyright (c) 2023 - SPE CRM

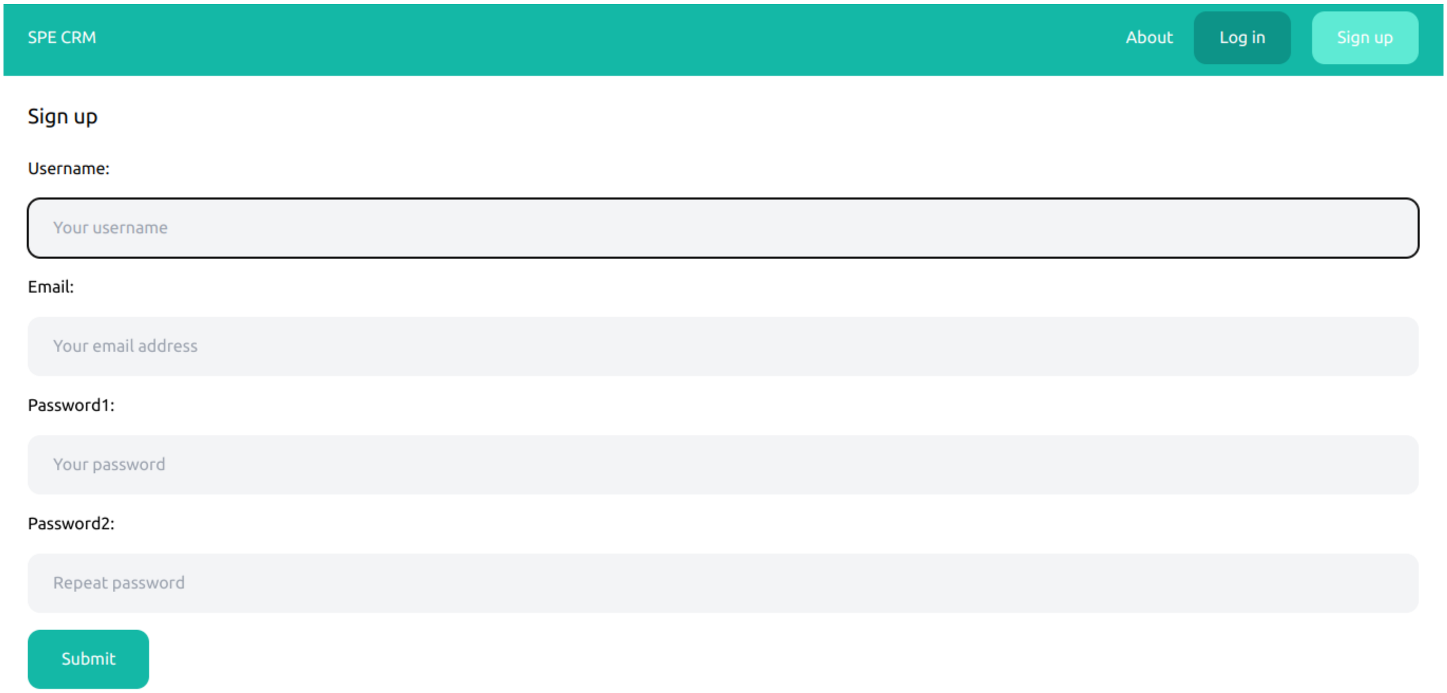


Log in

Username:

Password:

Copyright (c) 2023 - SPE CRM



Sign up

Username:

Your username

Email:

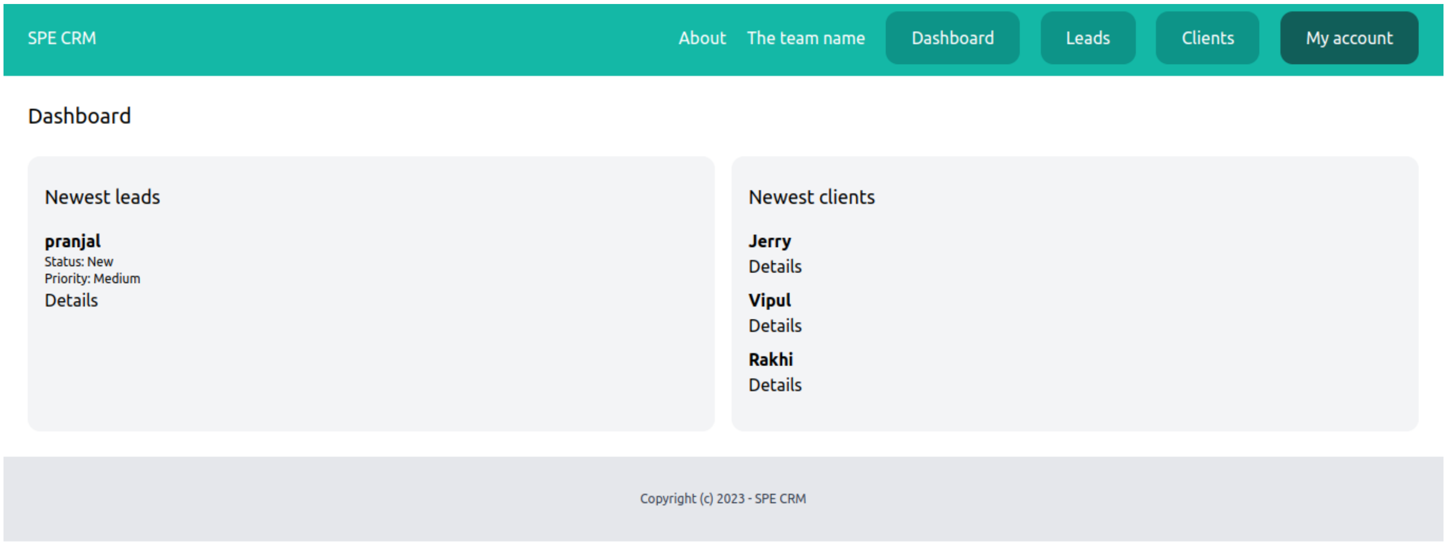
Your email address

Password1:

Your password

Password2:

Repeat password



Dashboard

Newest leads

pranjal
Status: New
Priority: Medium
Details

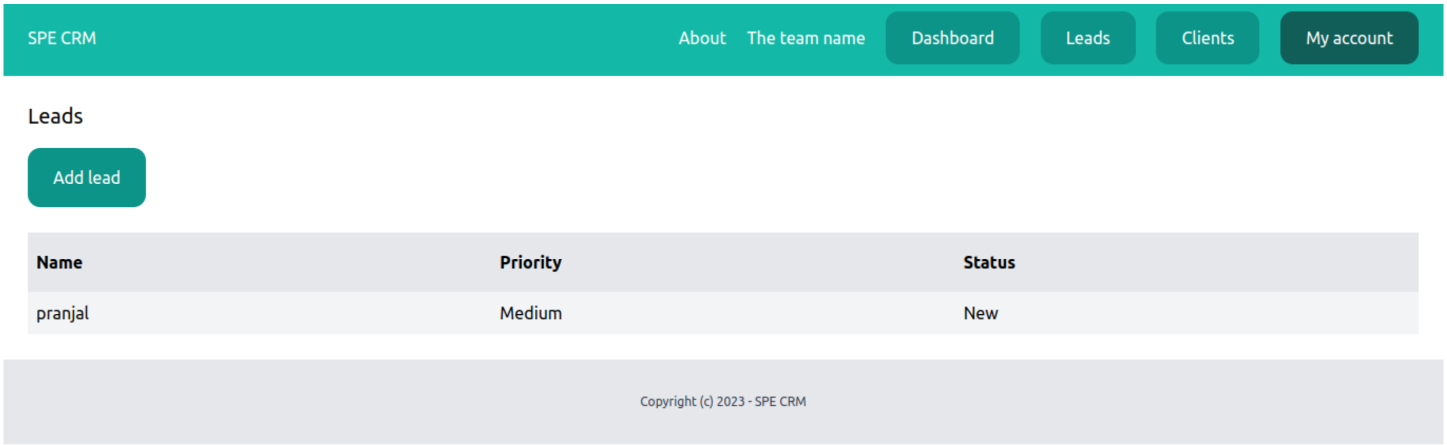
Newest clients

Jerry
Details

Vipul
Details

Rakhi
Details

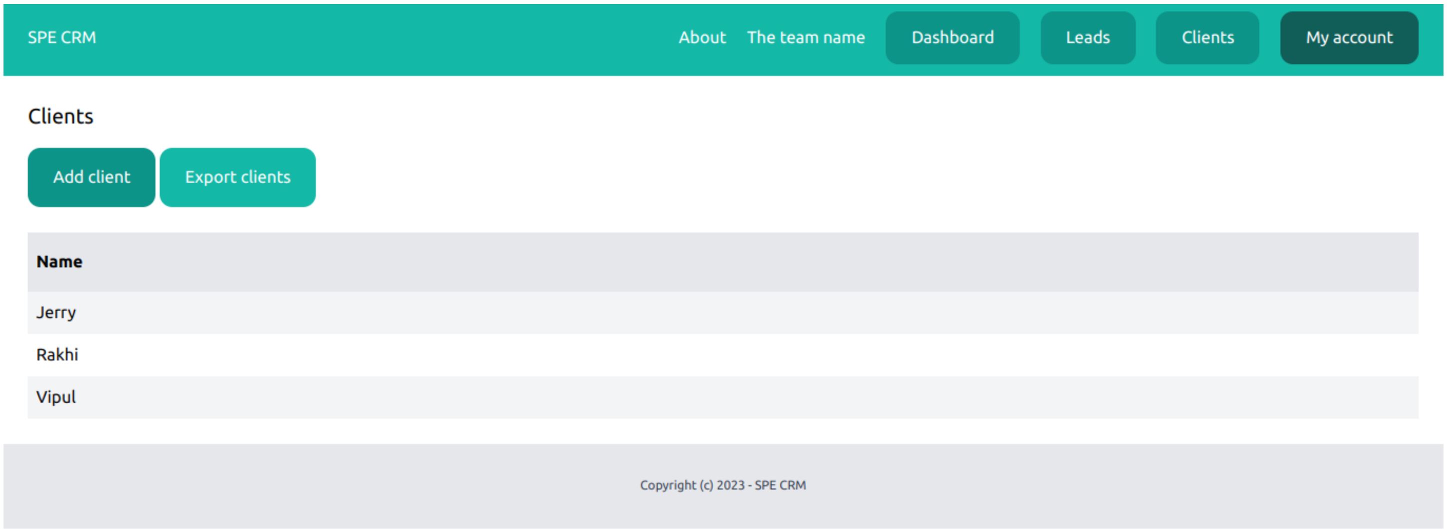
Copyright (c) 2023 - SPE CRM



Leads

Name	Priority	Status
pranjal	Medium	New

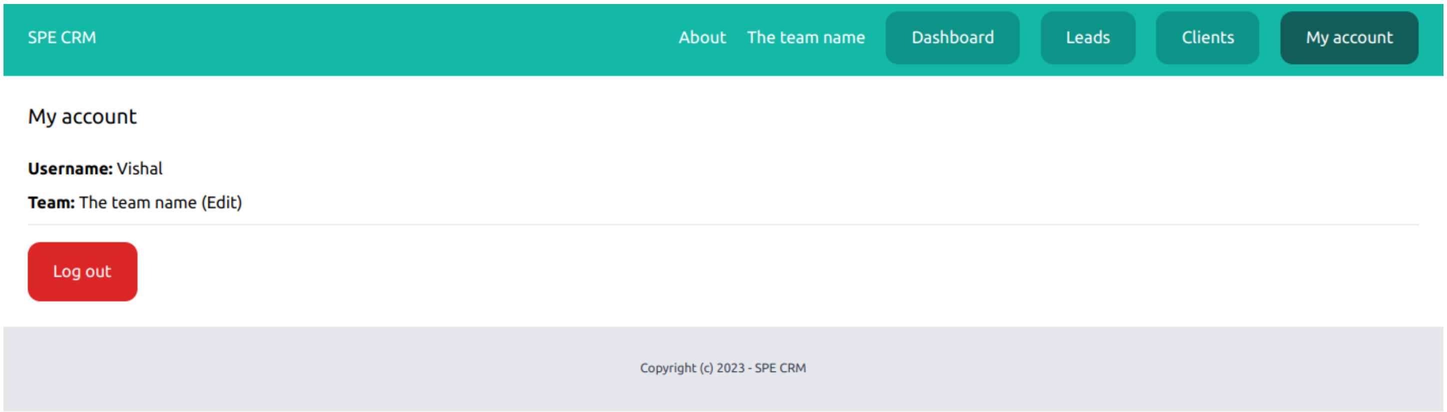
Copyright (c) 2023 - SPE CRM



Clients

Name
Jerry
Rakhi
Vipul

Copyright (c) 2023 - SPE CRM



My account

Username: Vishal

Team: The team name (Edit)

Copyright (c) 2023 - SPE CRM

Project Description:

Our project is developed in Django. Django is a high-level Python web framework that enables rapid development of secure and maintainable web applications. It follows the Model-View-Controller (MVC) architectural pattern, but in Django terminology, this pattern is referred to as Model-View-Template (MVT).

Django provides a range of built-in features and tools that make web development faster and easier. Some of its notable features include:
Object-Relational Mapping (ORM): Django provides a powerful ORM that enables developers to interact with databases using Python code, eliminating the need for writing SQL queries.
URL Routing: Django's URL routing system maps URLs to view functions, allowing developers to create clean and readable URLs.

Template System: Django's template system enables developers to create reusable HTML templates that can be rendered dynamically with data from a database.
Authentication and Authorization: Django provides a built-in authentication system that enables developers to add user authentication and authorization to their applications.
Administration Interface: Django provides a built-in administration interface that enables developers to manage the application's data and perform CRUD (Create, Read, Update, Delete) operations on the database.
Apps in our Project:
1)client
2)core
3)dashboard
4)lead
5)tealcrm
6)userprofile

In each app there are different files such forms.py, models.py, tests.py, views.py, urls.py, template folders, migration folders, etc.
So, here is a basic idea about creating the Django application.

Creating a Django application involves the following steps:

Install Django: If you don't have Django installed on your system, you can install it using pip command:

```
pip install django
```

Create a Django project: A Django project is a collection of settings and configurations for a specific website. You can create a new Django project by running the following command:

```
django-admin startproject projectname
```

This will create a new Django project with the given name.

Create a Django app: A Django app is a module that performs a specific functionality. You can create a new Django app within the project by running the following command:

```
python manage.py startapp appname
```

This will create a new app with the given name.
Define models: Models are used to define the structure of the database tables in the app. You can define models in the models.py file within the app directory.Create database tables: Once you have defined the models, you can create the database tables by running the following commands:

```
python manage.py makemigrations
python manage.py migrate
```

The first command creates a migration file based on the changes you made to the models, and the second command applies those changes to the database.

Define views: Views are used to handle user requests and render responses. You can define views in the views.py file within the app directory.

Define URLs: URLs are used to map URLs to views. You can define URLs in the urls.py file within the app directory.

Create templates: Templates are used to render HTML pages. You can create templates in the templates directory within the app directory.

Define Tests: Tests are used to check whether the application works on the desired input properly and gives error on undesired input. You can define the test case in tests.py file within the app directory.

```
Python manage.py test
```

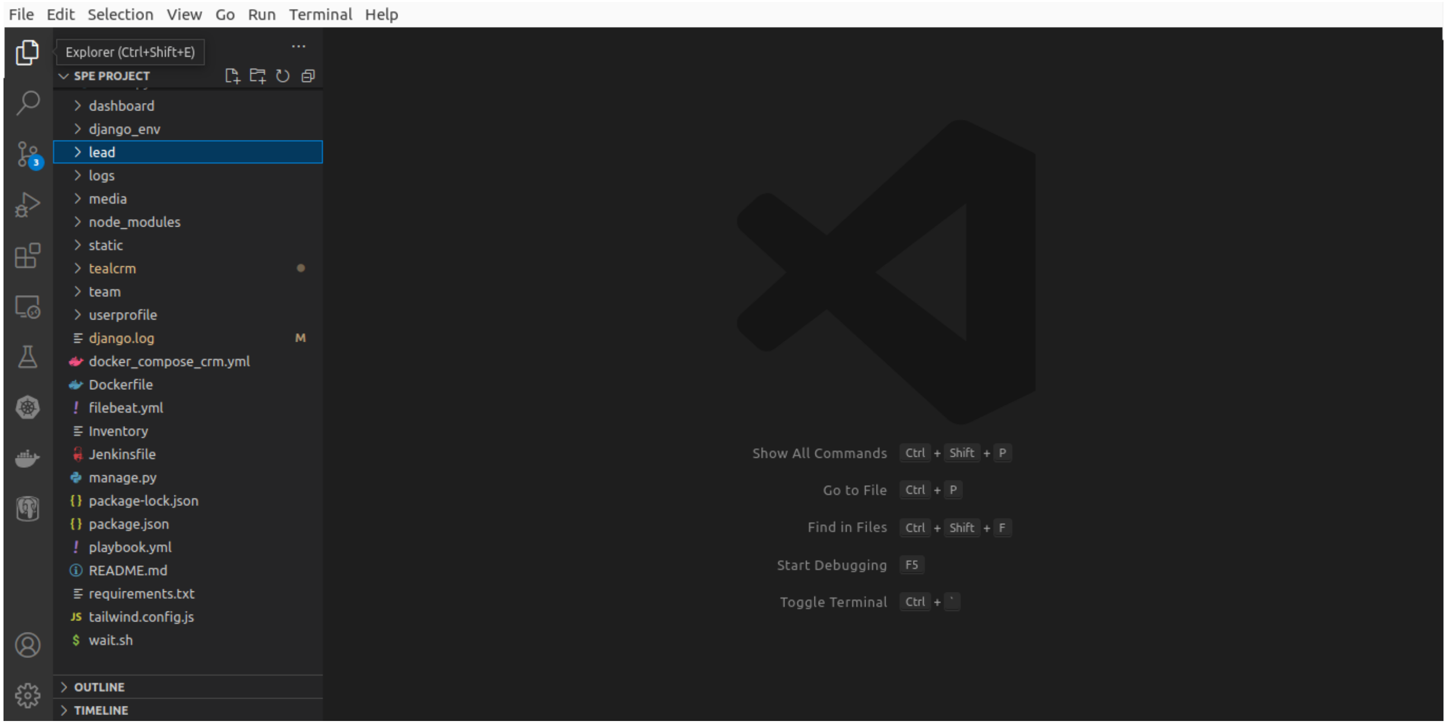
Start the application : Finally, you can start the application by running the following command:

```
python manage.py runserver
```

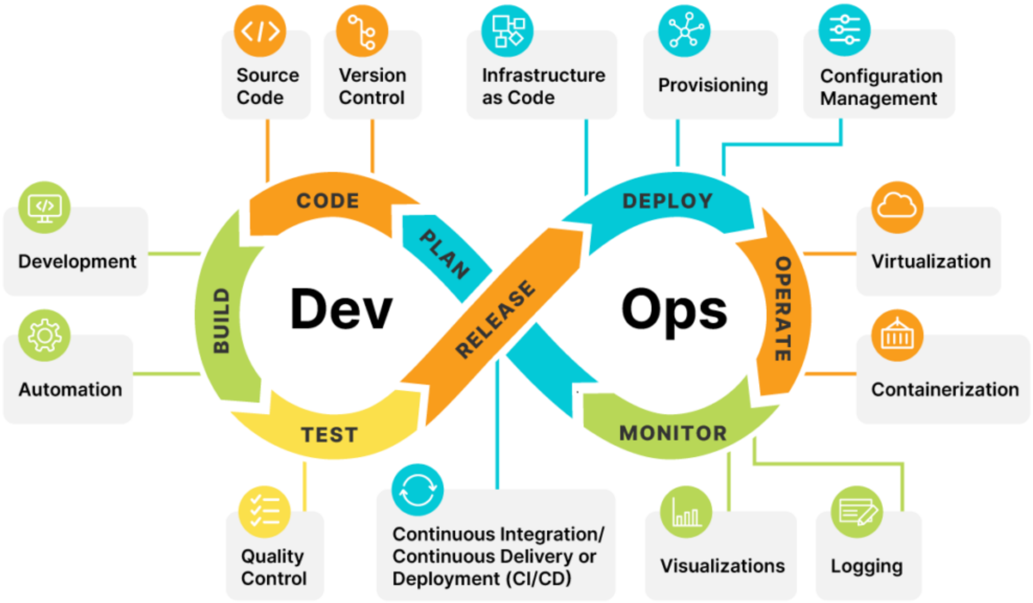
This will start the development server, and you can access the application in your web browser at http://localhost:8000/.

These are the basic steps to create a Django application.

Image of our VS Code Application which shows different apps in my project.



DEVOPS:



Version Controlling System:

A version control system (VCS) is a software tool that manages changes to source code, documentation, or any other set of files. It allows multiple developers to collaborate on a project while keeping track of changes made to the files over time.

There are two main types of VCS: centralized and distributed. Centralized version control systems have a central repository that all developers connect to, while distributed version control systems allow each developer to have their own repository, which can be synchronized with other developers' repositories.

Some popular version control systems include Git, SVN (Subversion), Mercurial, and CVS (Concurrent Versions System). Git, in particular, has become very popular in recent years and is widely used in the open-source community.

Using a version control system is an essential part of modern software development, as it helps developers collaborate more effectively and ensures that changes can be tracked, reviewed, and rolled back if necessary. It also makes it easier to maintain multiple versions of a project and manage releases.

We have used github as VCS.

GitHub is a web-based platform that is primarily used for version control and collaboration in software development projects. It allows developers to store and manage their code repositories, track changes made to the code over time, and collaborate with other developers on the same project. GitHub uses Git, which is a distributed version control system that allows for decentralized collaboration on code.
In addition to version control and collaboration, GitHub also offers a variety of features such as issue tracking, code reviews, continuous integration and deployment, and project management tools. It also has a vast community of users who contribute to open-source projects and share their code with others.
GitHub is widely used in the software development industry, from small startups to large corporations. It is also used by individual developers as a platform to showcase their work and build their portfolio.

CI/CD Pipeline using Jenkins:

CI/CD (Continuous Integration/Continuous Deployment) is a software development practice that involves the automatic building, testing, and deployment of code changes. Jenkins is an open-source automation server that can be used to create a CI/CD pipeline for your application. Here's an example of how to set up a basic pipeline using Jenkins:

Install Jenkins on your server or computer and make sure it's up and running.

Install any necessary plugins for your pipeline. For example, if you are using Git as your version control system, you'll need the Git plugin.

Create a new job in Jenkins and select "Pipeline" as the job type.

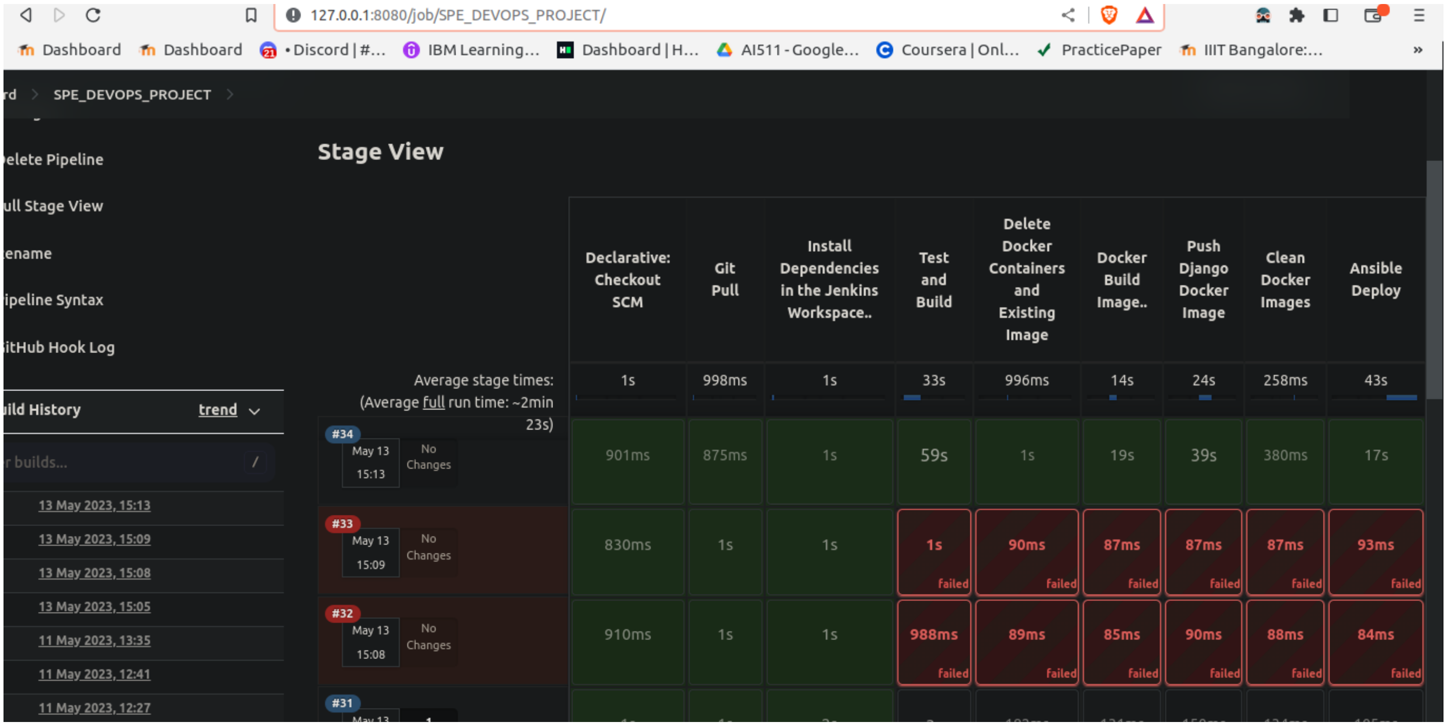
In the Pipeline section, select "Pipeline script" as the definition.

Write your pipeline script using the Jenkins DSL (domain-specific language) syntax. Here's a pipeline script of our project:

```
pipeline {
    environment{
        dockerimg = ''
        DB_NAME = "mydatabase"
        DB_HOST = "127.0.0.1"
        DB_PORT = "3306"
        DB_USER = "root"
        DB_PASSWORD = "Vipul*20"
    }

    agent any

    stages {
        stage('Git Pull') {
            steps {
                echo 'Git Pull'
                git branch: 'main', url: 'https://github.com/vipul2097/SPECRM.git',
                credentialsId: 'githubid'
            }
        }
        stage('Install Dependencies in the Jenkins Workspace..') {
            steps {
                echo 'Installing Dependencies'
                sh 'pip3 install django mysqlclient '
            }
        }
        stage('Test and Build') {
            steps {
                echo 'We dont need to Build our code. Just do the Model testing.'
                sh 'python3 manage.py test client.tests'
                sh 'python3 manage.py test lead.tests'
                sh 'python3 manage.py test team.tests'
                sh 'python3 manage.py test userprofile.tests'
            }
        }
        stage('Delete Docker Containers and Existing Image') {
            steps {
                script{
                    // here we are checking if there are any containers running in our system if so then delete them.
                    def running_containers = sh (returnStdout: true, script: 'docker ps -q').trim()
                    if (running_containers) {
                        sh 'docker rm -f $(docker ps -aq)'
                    }

                    sh 'docker image rm -f vipul2097/spemajorproject'
                }
            }
        }
        stage('Docker Build Image..') {
            steps {
                 script {
                     echo 'Docker Build Image.'
                     dockerimg = docker.build("vipul2097/spemajorproject")
                }
            }
        }
        stage('Push Django Docker Image') {
            steps {
                script{
                    docker.withRegistry('','dockerhub'){
                    dockerimg.push()
                    }
                }
            }
        }

        stage('Clean Docker Images') {
            steps{
            // sh '''
            //    # Remove all images with the tag <none>
            //    docker rmi --force $(docker images | grep "<none>" | awk '{print $3}')
            // '''
            sh 'docker image prune --force'

            }
         }
         stage('Ansible Deploy'){
            steps{
              ansiblePlaybook colorized:true, disableHostKeyChecking:true, installation:'Ansible',
inventory:'Inventory', playbook:'playbook.yml'
            }
         }

    }
}
```

Save and run your pipeline. Jenkins will automatically build, test, and deploy your code changes according to the defined pipeline.

Monitor your pipeline for any failures or errors and make necessary changes to your script to improve your pipeline's functionality and reliability.



127.0.0.1:8080/job/SPE_DEVOPS_PROJECT/

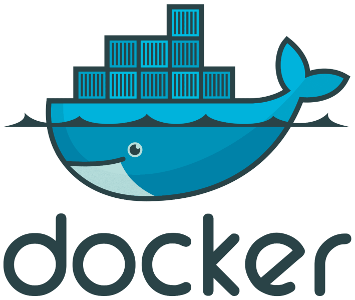
rd > SPE_DEVOPS_PROJECT >

13 May 2023, 15:13
13 May 2023, 15:09
13 May 2023, 15:08
13 May 2023, 15:05
11 May 2023, 13:35
11 May 2023, 12:41
11 May 2023, 12:27

Stage View

	Declarative: Checkout SCM	Git Pull	Install Dependencies in the Jenkins Workspace..	Test and Build	Delete Docker Containers and Existing Image	Docker Build Image..	Push Django Docker Image	Clean Docker Images	Ansible Deploy
Average stage times: (Average full run time: ~2min 23s)	1s	998ms	1s	33s	996ms	14s	24s	258ms	43s
#34 May 13 15:13 No Changes	901ms	875ms	1s	59s	1s	19s	39s	380ms	17s
#33 May 13 15:09 No Changes	830ms	1s	1s	1s failed	90ms failed	87ms failed	87ms failed	87ms failed	93ms failed
#32 May 13 15:08 No Changes	910ms	1s	1s	988ms failed	89ms failed	85ms failed	90ms failed	88ms failed	84ms failed
#31 May 13									

Containerization:



Docker is a popular platform for developing, shipping, and running applications in containers. Containers are lightweight, standalone, and executable packages that contain everything needed to run an application, including the application code, runtime, system tools, libraries, and settings. Docker makes it easy to create, deploy, and manage containers by providing a simple and consistent interface for developers and system administrators.
Docker uses a client-server architecture and consists of several components, including the Docker Engine, Docker CLI, Dockerfile, Docker Hub, and Docker Compose. The Docker Engine is the core component that runs and manages containers. The Docker CLI is a command-line tool that developers use to interact with the Docker Engine. The Dockerfile is a text file that developers use

to define the configuration of a Docker image. Docker Hub is a cloud-based repository for storing and sharing Docker images. Docker Compose is a tool for defining and running multi-container Docker applications.Docker provides several benefits for developers and system administrators, including faster application deployment, better portability, improved scalability, and reduced infrastructure costs. Docker has become increasingly popular in recent years, and many organizations use it to streamline their development and deployment workflows.

Dockerfile of our project-

```
#Creating the Base Image
FROM python:3.10.7-alpine3.15

#Setting Up Environment Variables in Docker Container
ENV DockerHOME=/app

#Setting Work Directory (In this your actual code resides)
RUN mkdir -p $DockerHOME

#Present Work Directory
WORKDIR $DockerHOME

# set environment variables
ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

RUN mkdir -p crm_logs

# install dependencies
RUN pip install --upgrade pip

# install dependencies mysqlclient using mariadb connector and delete it after installation.
RUN apk update && \
    apk add --no-cache mariadb-connector-c-dev && \
    apk add --no-cache python3 python3-dev mariadb-dev build-base && \
    pip3 install --no-cache-dir mysqlclient && \
    apk del python3-dev mariadb-dev build-base
```

```
# checking the network adapters inside the docker container
RUN apk add netcat-openbsd

# Copy the code contents to the /app directory.
COPY . $DockerHOME/

#Install the required dependencies.
RUN pip install -r requirements.txt

COPY wait.sh $DockerHOME/wait.sh
RUN chmod +x $DockerHOME/wait.sh
```

Explanation:-

The code starts by specifying the base image as python:3.10.7-alpine3.15. This means that the image will be built on top of an existing image that has Python 3.10.7 installed and runs on Alpine Linux 3.15.

The next instruction sets an environment variable DockerHOME to /app. This variable will be used later in the code.

The following instruction creates a directory at the path specified by the DockerHOME environment variable.

The WORKDIR instruction sets the working directory for subsequent instructions in the Dockerfile.

The next two instructions set environment variables PYTHONDONTWRITEBYTECODE and PYTHONUNBUFFERED to 1. This will ensure that Python does not write bytecode files and that the standard output and error streams are unbuffered.

The instruction RUN mkdir -p crm_logs creates a directory called crm_logs inside the Docker container.

The next set of instructions installs the dependencies required for connecting to a MySQL database using Python. It installs the mariadb-connector-c-dev package, Python 3 and development libraries, mariadb-dev package, and build-base package, which is used to build Python packages from source. It then installs the mysqlclient package using pip3. Finally, it removes the previously installed packages that are no longer needed.

The instruction RUN apk add netcat-openbsd installs the netcat-openbsd package, which is used to check network adapters inside the Docker container.

The instruction COPY . $DockerHOME/ copies the contents of the current directory to the directory specified by the DockerHOME environment variable.

The instruction RUN pip install -r requirements.txt installs the required dependencies listed in the requirements.txt file.

The instruction COPY wait.sh $DockerHOME/wait.sh copies a shell script called wait.sh to the directory specified by the DockerHOME environment variable.

The instruction RUN chmod +x $DockerHOME/wait.sh makes the wait.sh script executable.

Docker-Compose File code:-

```
version: "3"

services:
  web:
    image: vipul2097/spemajorproject
    container_name: spemajorproject
    environment:
      - DB_HOST=mydb2
      - DB_PORT=3306
      - DB_NAME=mydatabase
      - DB_USER=root
      - DB_PASSWORD=root
    ports:
      - "8000:8000"
    depends_on:
      - mydb2
    entrypoint: ["./wait.sh"]
    volumes:
      - crm_logs:/app
    networks:
      - efk

  mydb2:
    image: mysql:8.0.33
    ports:
      - "3308:3306"
    environment:
      #These environment variables are to be not included in the Settings.py file of Django.
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: mydatabase
    # This will create the database spemajorproject here if it not exists.
    volumes:
      - vipul_db_CRM:/var/lib/mysql
    networks:
      - efk

  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.8.0
    container_name: elasticsearch
    ports:
      - 9200:9200
      - 9300:9300
    environment:
      discovery.type: single-node
    networks:
      - efk
  kibana:
    image: docker.elastic.co/kibana/kibana:7.8.0
    ports:
      - 5601:5601
    links:
      - elasticsearch
    depends_on:
      - elasticsearch
    networks:
      - efk

  filebeat:
    image: docker.elastic.co/beats/filebeat:7.8.0
    volumes:
      - filebeat.yml:/filebeat.yml:ro
      - crm_logs:/app
    environment:
      ELASTICSEARCH_URL: http://elasticsearch:9200
    links:
      - kibana
      - elasticsearch
    depends_on:
      - elasticsearch
    networks:
      - efk

networks:
  efk:
    driver: bridge

# Creating a volume named mydb_data and store the data of the MYSQL running in the container by doing the mapping.
volumes:
  vipul_db_CRM:
  crm_logs:
```

```
  filebeat.yml:
    driver: local
```

Explanation:-

This is a docker-compose YAML file, which describes a multi-container Docker application with multiple services and networks.

The version key specifies the version of the Docker Compose file format being used. In this case, it is version 3.

The first service defined is web. It uses an image from Docker Hub, specified by image: vipul2097/spemajorproject, and sets the container name to spemajorproject. The environment key is used to set environment variables inside the container. The ports key maps port 8000 of the container to port 8000 of the host machine. The depends_on key specifies that the web service depends on the mydb2 service, and the entrypoint key specifies a shell script to be executed when the container starts. The volumes key creates a named volume named crm_logs and maps it to the /app directory inside the container. Finally, the networks key adds the web service to the efk network.

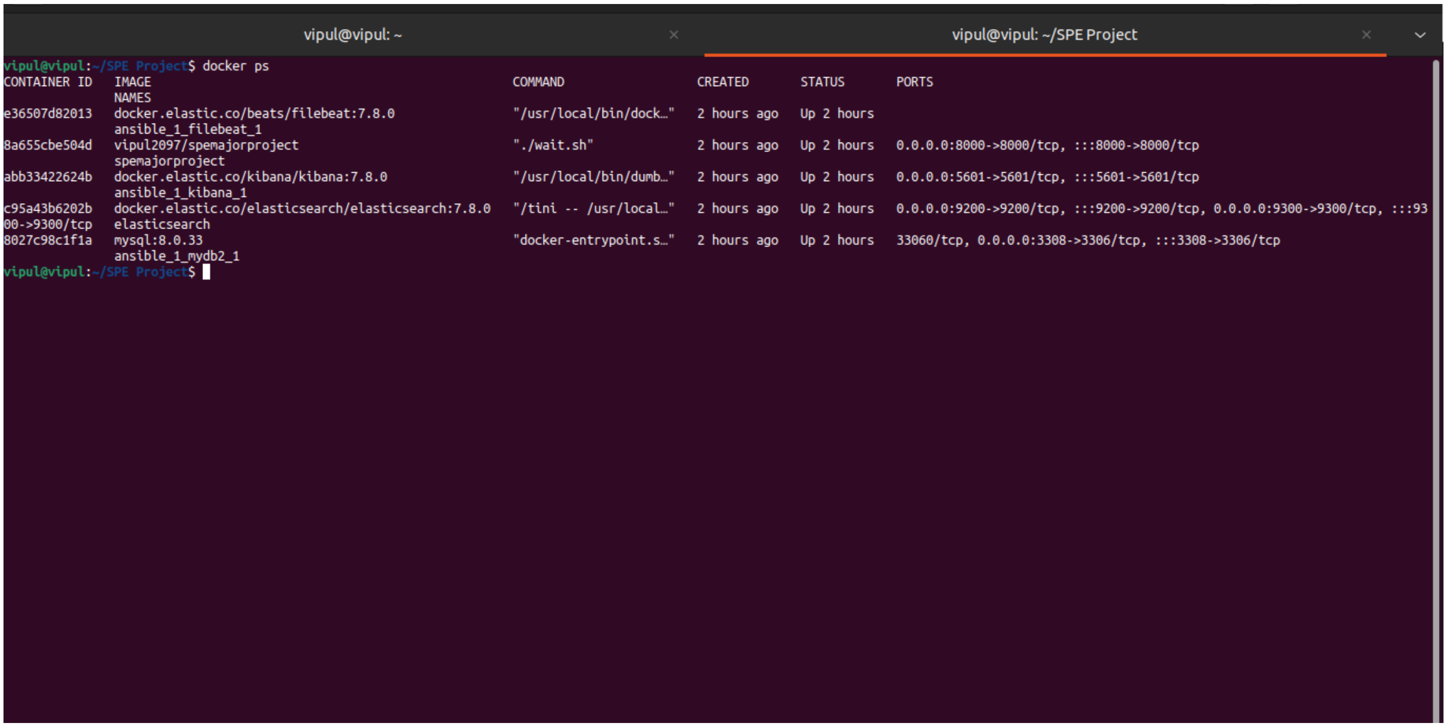
The mydb2 service uses an image from Docker Hub, specified by image: mysql:8.0.33, and sets the container name to mydb2. The ports key maps port 3306 of the container to port 3308 of the host machine. The environment key sets environment variables for the MySQL container. The volumes key creates a named volume named vipul_db_CRM and maps it to the /var/lib/mysql directory inside the container. Finally, the networks key adds the mydb2 service to the efk network.

The elasticsearch service uses an image from the Elastic Docker Registry, specified by image: docker.elastic.co/elasticsearch/elasticsearch:7.8.0, and sets the container name to elasticsearch. The ports key maps port 9200 and port 9300 of the container to the same ports on the host machine. The environment key sets the discovery.type variable to single-node. Finally, the networks key adds the elasticsearch service to the efk network.

The kibana service uses an image from the Elastic Docker Registry, specified by image: docker.elastic.co/kibana/kibana:7.8.0, and sets the container name to kibana. The ports key maps port 5601 of the container to port 5601 of the host machine. The links key specifies that the kibana service depends on the elasticsearch service. The depends_on key specifies that the kibana service also depends on the elasticsearch service. Finally, the networks key adds the kibana service to the efk network.

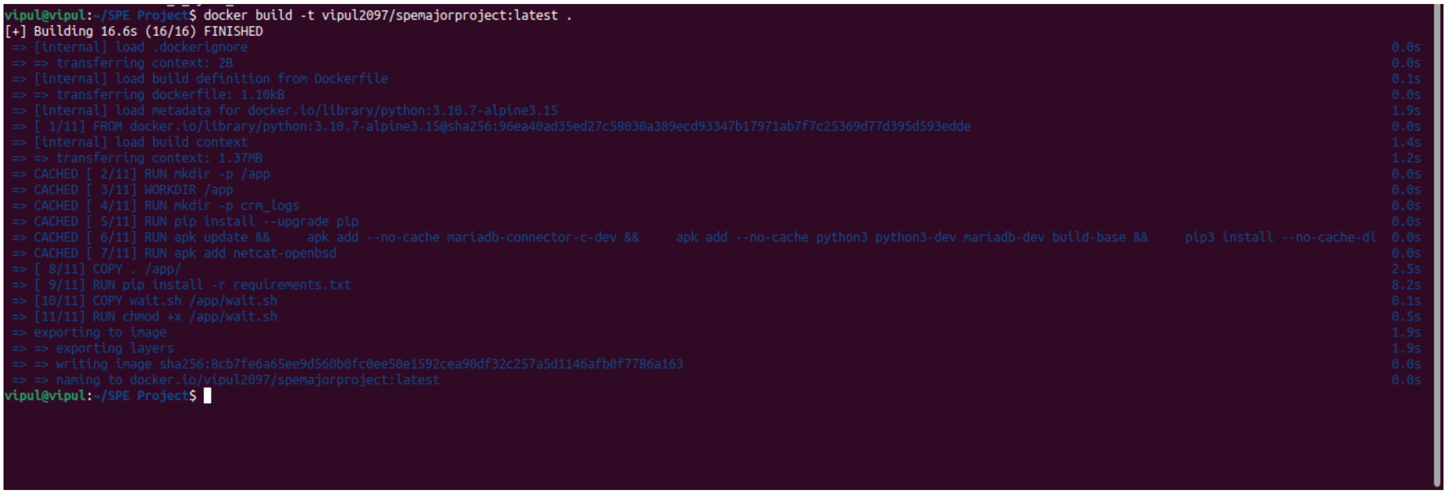
The filebeat service uses an image from the Elastic Docker Registry, specified by image: docker.elastic.co/beats/filebeat:7.8.0. The volumes key maps the filebeat.yml file to the /filebeat.yml file inside the container, and maps the crm_logs named volume to the /app directory inside the container. The environment key sets the ELASTICSEARCH_URL variable to http://elasticsearch:9200. The links key specifies that the filebeat service depends on the kibana and elasticsearch services. The depends_on key specifies that the filebeat service depends on the elasticsearch service. Finally,the file defines three volumes: vipul_db_CRM, crm_logs, and filebeat.yml. The vipul_db_CRM volume is used by the mydb2 service to store the data of the MySQL container, and the crm_logs volume is used by the web and filebeat services to store logs. The filebeat.yml file is also mapped to the filebeat service.

Here is the screenshot of docker ps:-



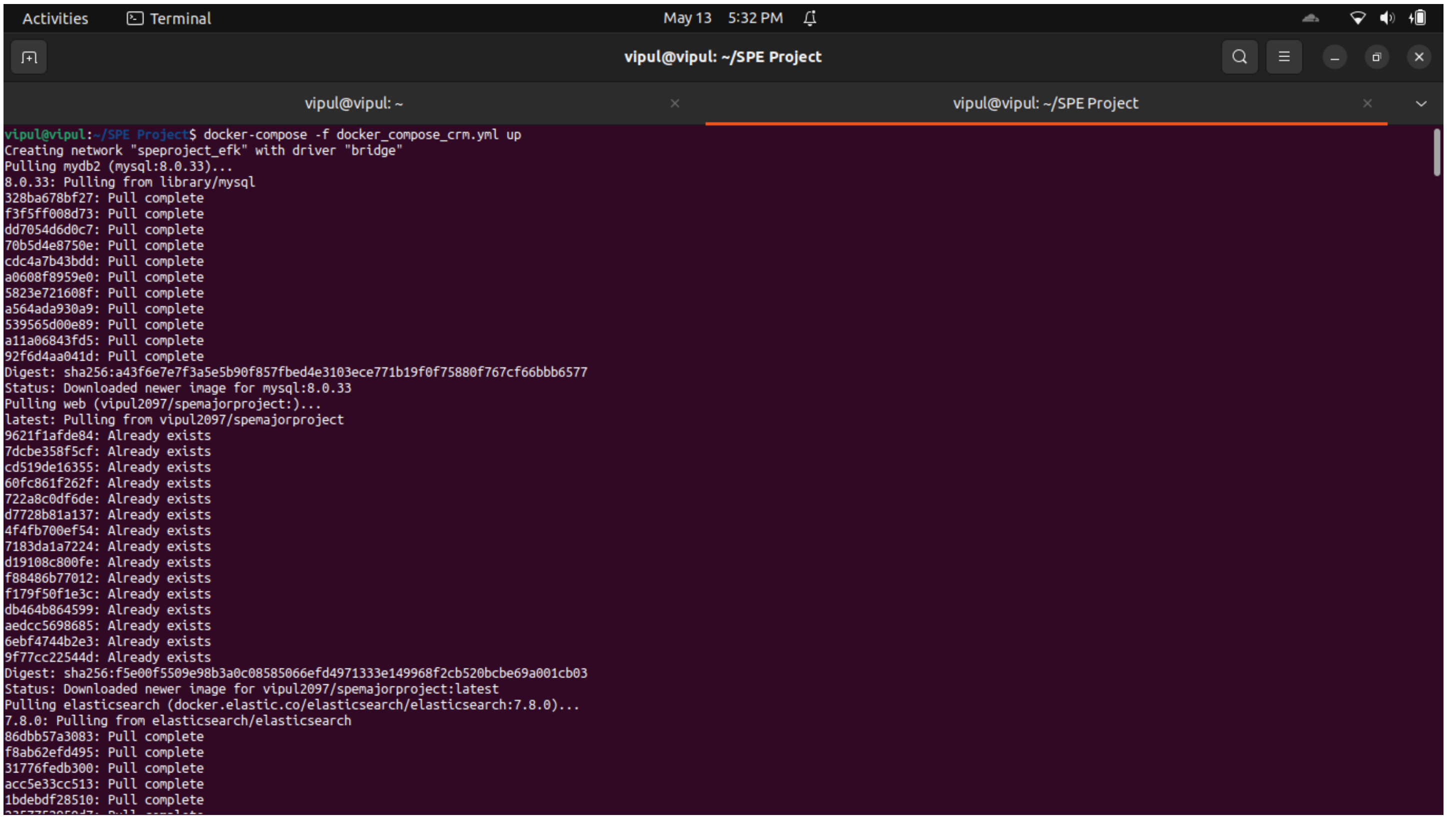
```
vipul@vipul:~/SPE Project$ docker ps
CONTAINER ID   IMAGE                                                  COMMAND                  CREATED       STATUS       PORTS
               NAMES
e36507d82013   docker.elastic.co/beats/filebeat:7.8.0                 "/usr/local/bin/dock…"   2 hours ago   Up 2 hours
               ansible_1_filebeat_1
8a655cbe504d   vipul2097/spemajorproject                              "./wait.sh"              2 hours ago   Up 2 hours   0.0.0.0:8000->8000/tcp, :::8000->8000/tcp
               spemajorproject
abb33422624b   docker.elastic.co/kibana/kibana:7.8.0                  "/usr/local/bin/dumb…"   2 hours ago   Up 2 hours   0.0.0.0:5601->5601/tcp, :::5601->5601/tcp
               ansible_1_kibana_1
c95a43b6202b   docker.elastic.co/elasticsearch/elasticsearch:7.8.0    "/tini -- /usr/local…"   2 hours ago   Up 2 hours   0.0.0.0:9200->9200/tcp, :::9200->9200/tcp, 0.0.0.0:9300->9300/tcp, :::93
00->9300/tcp   elasticsearch
8027c98c1f1a   mysql:8.0.33                                           "docker-entrypoint.s…"   2 hours ago   Up 2 hours   33060/tcp, 0.0.0.0:3308->3306/tcp, :::3308->3306/tcp
               ansible_1_mydb2_1
vipul@vipul:~/SPE Project$
```

Again we are using docker build command to build the image and output is shown below in the form of image:-



```
vipul@vipul:~/SPE Project$ docker build -t vipul2097/spemajorproject:latest .
[+] Building 16.6s (16/16) FINISHED
 => [internal] load .dockerignore                                                                                                                   0.0s
 => => transferring context: 2B                                                                                                                     0.0s
 => [internal] load build definition from Dockerfile                                                                                                0.1s
 => => transferring dockerfile: 1.10kB                                                                                                              0.0s
 => [internal] load metadata for docker.io/library/python:3.10.7-alpine3.15                                                                         1.9s
 => [ 1/11] FROM docker.io/library/python:3.10.7-alpine3.15@sha256:96ea40ad35ed27c58030a389ecd93347b17971ab7f7c25369d77d395d593edde                 0.0s
 => [internal] load build context                                                                                                                   1.4s
 => => transferring context: 1.37MB                                                                                                                 1.2s
 => CACHED [ 2/11] RUN mkdir -p /app                                                                                                                0.0s
 => CACHED [ 3/11] WORKDIR /app                                                                                                                     0.0s
 => CACHED [ 4/11] RUN mkdir -p crm_logs                                                                                                            0.0s
 => CACHED [ 5/11] RUN pip install --upgrade pip                                                                                                    0.0s
 => CACHED [ 6/11] RUN apk update &&     apk add --no-cache mariadb-connector-c-dev &&     apk add --no-cache python3 python3-dev mariadb-dev build-base &&     pip3 install --no-cache-di  0.0s
 => CACHED [ 7/11] RUN apk add netcat-openbsd                                                                                                       0.0s
 => [ 8/11] COPY . /app/                                                                                                                            2.5s
 => [ 9/11] RUN pip install -r requirements.txt                                                                                                     8.2s
 => [10/11] COPY wait.sh /app/wait.sh                                                                                                               0.1s
 => [11/11] RUN chmod +x /app/wait.sh                                                                                                               0.5s
 => exporting to image                                                                                                                              1.9s
 => => exporting layers                                                                                                                             1.9s
 => => writing image sha256:8cb7fe6a65ee9d560b0fc0ee50e1592cea90df32c257a5d1146afb0f7786a163                                                        0.0s
 => => naming to docker.io/vipul2097/spemajorproject:latest                                                                                         0.0s
vipul@vipul:~/SPE Project$
```

Docker-compose command output:



```
vipul@vipul:~/SPE Project$ docker-compose -f docker_compose_crm.yml up
Creating network "speproject_efk" with driver "bridge"
Pulling mydb2 (mysql:8.0.33)...
8.0.33: Pulling from library/mysql
328ba678bf27: Pull complete
f3f5ff008d73: Pull complete
dd7054d6d0c7: Pull complete
70b5d4e8750e: Pull complete
cdc4a7b43bdd: Pull complete
a0608f8959e0: Pull complete
5823e721608f: Pull complete
a564ada930a9: Pull complete
539565d00e89: Pull complete
a11a06843fd5: Pull complete
92f6d4aa041d: Pull complete
Digest: sha256:a43f6e7e7f3a5e5b90f857fbed4e3103ece771b19f0f75880f767cf66bbb6577
Status: Downloaded newer image for mysql:8.0.33
Pulling web (vipul2097/spemajorproject:)...
latest: Pulling from vipul2097/spemajorproject
9621f1afde84: Already exists
7dcbe358f5cf: Already exists
cd519de16355: Already exists
60fc861f262f: Already exists
722a8c0df6de: Already exists
d7728b81a137: Already exists
4f4fb700ef54: Already exists
7183da1a7224: Already exists
d19108c800fe: Already exists
f88486b77012: Already exists
f179f50f1e3c: Already exists
db464b864599: Already exists
aedcc5698685: Already exists
6ebf4744b2e3: Already exists
9f77cc22544d: Already exists
Digest: sha256:f5e00f5509e98b3a0c08585066efd4971333e149968f2cb520bcbe69a001cb03
Status: Downloaded newer image for vipul2097/spemajorproject:latest
Pulling elasticsearch (docker.elastic.co/elasticsearch/elasticsearch:7.8.0)...
7.8.0: Pulling from elasticsearch/elasticsearch
86dbb57a3083: Pull complete
f8ab62efd495: Pull complete
31776fedb300: Pull complete
acc5e33cc513: Pull complete
1bdebdf28510: Pull complete
```

Continuous Deployment:

Ansible is an open-source automation tool that helps in configuring, managing, and deploying software applications and infrastructure. It is designed to be simple, powerful, and easy to learn. Ansible allows users to automate tasks such as installing software, configuring servers, and deploying applications on multiple machines at once.

Ansible uses a declarative language called YAML (YAML Ain't Markup Language) to describe tasks and configurations. The YAML file contains the desired state of the system, and Ansible ensures that the system reaches that state. It also uses SSH to connect to remote servers and execute tasks.

Ansible can be used to automate various tasks, including:

Configuration management: It helps in configuring servers and applications by using a single playbook for multiple machines.

Provisioning: Ansible can create, configure, and manage virtual machines, containers, and other cloud resources.

Application deployment: It helps in deploying applications across multiple servers, ensuring consistency and reliability.

Continuous delivery: Ansible can be used to automate the entire software delivery pipeline, from code deployment to testing and production.

Ansible is a powerful tool that is easy to learn and use. It is widely used in the DevOps industry for automation and orchestration of infrastructure and applications.

Inventory File:-

```
[ubuntu22.10]
192.168.1.7 ansible_user=ansible_1
```

In the context of Ansible, an inventory file is a text file that contains a list of hosts or groups of hosts that Ansible can connect to and manage using its modules. The inventory file specifies

information such as the IP addresses or domain names of the hosts, the username and password for connecting to the hosts, and any other parameters necessary to establish the connection.
The first line from an inventory file that specifies a single host with IP address 192.168.1.7 and the username ansible_1 for connecting to the host. This host can be accessed by Ansible using the SSH protocol, which allows Ansible to execute commands and tasks remotely on the host.
Note that the example you provided does not include any information about the host's group membership or any additional parameters for connecting to the host. In a complete inventory file, you would typically specify one or more groups of hosts, and additional parameters such as SSH keys, port numbers, and other connection settings.

Playbook.yml:-

```
---
- name: copy docker-compose file
 hosts: all
 tasks:
   - name: copy docker-compose file
     copy:
       src: ./docker_compose_crm.yml
       dest: ./
   - name:  docker-compose down
     command: docker-compose -f docker_compose_crm.yml down
   - name: run docker-compose file
     command: docker-compose -f docker_compose_crm.yml up -d
```

The purpose of this playbook is to copy a Docker Compose file to the current directory of all hosts and then run a Docker Compose stack defined in the file.

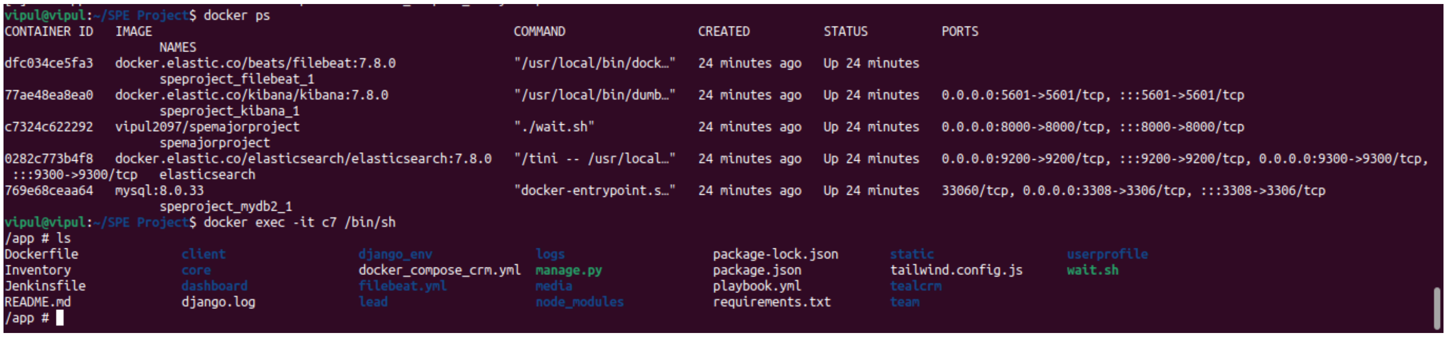
Here's a breakdown of what each task does:

"copy docker-compose file": Copies the file docker_compose_crm.yml from the current directory of the control node to the current directory of the target hosts. This is done using the Ansible module copy.
"docker-compose down": Runs the command docker-compose -f docker_compose_crm.yml down on the target hosts. This stops and removes containers, networks, and volumes defined in the Docker Compose file. This is done using the Ansible module command.

"run docker-compose file": Runs the command docker-compose -f docker_compose_crm.yml up -d on the target hosts. This creates and starts containers defined in the Docker Compose file in the background (-d flag). This is done using the Ansible module command.
Overall, this playbook automates the process of deploying a Docker Compose stack on a group of hosts by copying the Compose file and then starting and stopping the stack using the docker-compose command.

CONTINUOUS MONITORING:

docker exec -it c7 /bin/sh



```
vipul@vipul:~/SPE Project$ docker ps
CONTAINER ID   IMAGE                                                  COMMAND                  CREATED          STATUS          PORTS
      NAMES
dfc034ce5fa3   docker.elastic.co/beats/filebeat:7.8.0                 "/usr/local/bin/dock…"   24 minutes ago   Up 24 minutes
      speproject_filebeat_1
77ae48ea8ea0   docker.elastic.co/kibana/kibana:7.8.0                  "/usr/local/bin/dumb…"   24 minutes ago   Up 24 minutes   0.0.0.0:5601->5601/tcp, :::5601->5601/tcp
      speproject_kibana_1
c7324c622292   vipul2097/spemajorproject                              "./wait.sh"              24 minutes ago   Up 24 minutes   0.0.0.0:8000->8000/tcp, :::8000->8000/tcp
      spemajorproject
0282c773b4f8   docker.elastic.co/elasticsearch/elasticsearch:7.8.0   "/tini -- /usr/local…"   24 minutes ago   Up 24 minutes   0.0.0.0:9200->9200/tcp, :::9200->9200/tcp, 0.0.0.0:9300->9300/tcp,
 :::9300->9300/tcp   elasticsearch
769e68ceaa64   mysql:8.0.33                                           "docker-entrypoint.s…"   24 minutes ago   Up 24 minutes   33060/tcp, 0.0.0.0:3308->3306/tcp, :::3308->3306/tcp
      speproject_mydb2_1
vipul@vipul:~/SPE Project$ docker exec -it c7 /bin/sh
/app # ls
Dockerfile        client          django_env               logs            package-lock.json   static               userprofile
Inventory         core            docker_compose_crm.yml   manage.py       package.json        tailwind.config.js   wait.sh
Jenkinsfile       dashboard       filebeat.yml             media           playbook.yml        tealcrm
README.md         django.log      lead                     node_modules    requirements.txt    team
/app #
```

Now see where the log files are then fetch it and place it in your local using this command.

docker cp <container_name>:<path_to_file> <host_path>

Now we have our log with us.. We need to upload it to the ELK stack. But for the sake of simplicity, we will upload it manually in Kibana or we could use the Filebeat to upload it.(We tried to automate the logs file to ELK stack and we somehow able to load the elasticsearch page but on elastic search logs file are not able to get uploaded in index management on that).

Filebeat.yml:-

```
filebeat.inputs:
- type: log
 enabled: true
 paths:
   - "*.log"
 multiline.pattern: '^\['
 multiline.negate: true
 multiline.match: after

output.elasticsearch:
 hosts: ["http://elasticsearch:9200"]

setup.kibana:
 host: "http://kibana:5601"
```