International Institute of Information Technology, Bangalore

(IIIT Bangalore)

Software Production Engineering

Project Report

Swapsie - A Platform to Swap Items

Under the Guidance of Prof. B. Thangaraju

Teaching Assistant: Suchi

Shubham Pandey
MT2022111

Nitin Pandey
MT2022141

TABLE OF CONTENTS

Abstract

Swapsie is an innovative online platform specifically designed to encourage sustainable living practices by facilitating the sharing of expensive and infrequently used items among both individuals and organizations.

Oftentimes, individuals may find themselves needing to acquire certain products for a specific purpose, only to have them remain unused thereafter. As these items have been utilized only once, they typically retain their optimal condition and can be effectively repurposed, leading to a reduction of waste and potential cost savings through shared utilization. Swapsie seeks to address this issue by providing a convenient platform that connects different individuals and organizations, thereby promoting the reuse of items and contributing to the conservation of resources.

Swapsie's mission is to encourage responsible consumption and promote sustainable living by enabling the sharing of expensive and seldom-used items. The platform's convenience lies in its ability to connect individuals and organizations, allowing for the repurposing of items that would otherwise go to waste.

This model of sharing not only reduces costs and minimizes the impact on the environment, but it also fosters a sense of community and collaboration. For instance, graduating students on our college campus can pass on common hostel items such as buckets and chairs to the incoming batch. Swapsie can also be used by current students to share notes and other useful resources. Through Swapsie, we hope to inspire a culture of responsible consumption and create a more sustainable future.

INTRODUCTION -

Overview

This report outlines the Swapsie project, an innovative online platform designed to promote sustainable living practices by facilitating the sharing of expensive and infrequently used items among individuals and organizations.

The report provides an overview of the platform's mission, which is to encourage responsible consumption and promote sustainable living by enabling the sharing of expensive and seldom-used items. It explains how the platform works by connecting individuals and organizations, allowing for the repurposing of items that would otherwise go to waste.

The report highlights the benefits of this model of sharing, which not only reduces costs and minimizes the impact on the environment, but also fosters a sense of community and collaboration. It gives examples of how the platform can be used, such as graduating students passing on common hostel items to incoming students, or current students sharing notes and other useful resources.

Additionally, the report details the importance of promoting sustainable living practices, as it is critical for preserving our planet's natural resources and combating climate change. It concludes by emphasizing how Swapsie aims to inspire a culture of responsible consumption and create a more sustainable future.

The report will also discuss the secondary objective of the Swapsie project, which is to incorporate a complete DevOps framework to automate the entire software development lifecycle from development to deployment. This goal aims to ensure that the platform is always up-to-date, functioning optimally, and can be deployed with ease. The report will detail how this objective was achieved and its impact on the overall success of the project.

Github (frontend) - https://github.com/Shubhamp194/Swapsie-frontend
Github (backend) - https://github.com/Shubhamp194/Swapsie-backend

Features:

Swapsie offers a range of features that enable users to easily share and swap items with other members of the platform. These features include:

1. **User Registration and Login**: Swapsie provides a user-friendly registration and login process that allows users to create an account and access the platform's features.

2. **Add Product**: Users can add new products to the platform, providing details such as item name, description, and location.

3. **Update Product Details**: Users can update the details of their posted products, including item availability, location, and other relevant information.

4. **Delete Product**: Swapsie allows users to delete products from the platform when they have been swapped or are no longer available.

5. **Product Request**: Users can request items they need from other members of the platform, facilitating the sharing of resources.

6. **Product Swap**: Swapsie enables users to swap items with other members in the platform, giving them an opportunity to get something they need while also being able to lend out items they don't need anymore.

7. **Accept Swap** Request: Users can accept swap requests from other members, facilitating the exchange of items.

8. **Decline Swap Request**: Users can decline swap requests, providing flexibility and control over the sharing process.

9. **Delete Swap Request**: Swapsie allows users to delete swap requests that are no longer relevant or have been completed.

These features are designed to make Swapsie a user-friendly, accessible, and convenient platform for sharing and swapping items, promoting sustainable living practices, and fostering a sense of community and collaboration.

PROJECT ARCHITECTURE

Overview -

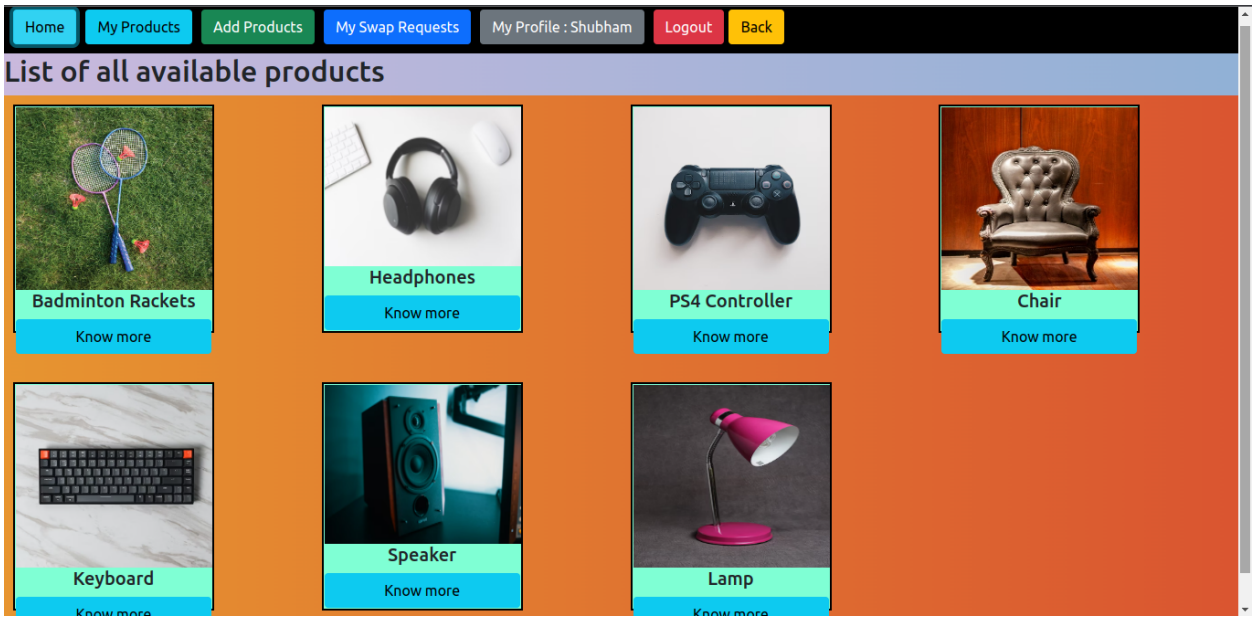
Framework - Java Spring Boot Framework
Database - MYSQL
Frontend - React
Deployment Server - Tomcat
Repository Hosting - Github
Artifact - JAR file
Container Tool - Docker
Github Repo (frontend) - https://github.com/Shubhamp194/Swapsie-frontend
Github Repo (backend) - https://github.com/Shubhamp194/Swapsie-backend

Automation Tools -

SCM - Git
Build - Maven
Testing - Junit
Configuration - Ansible
Automation server - Jenkins
Container - Docker
Monitoring - ELK Stack

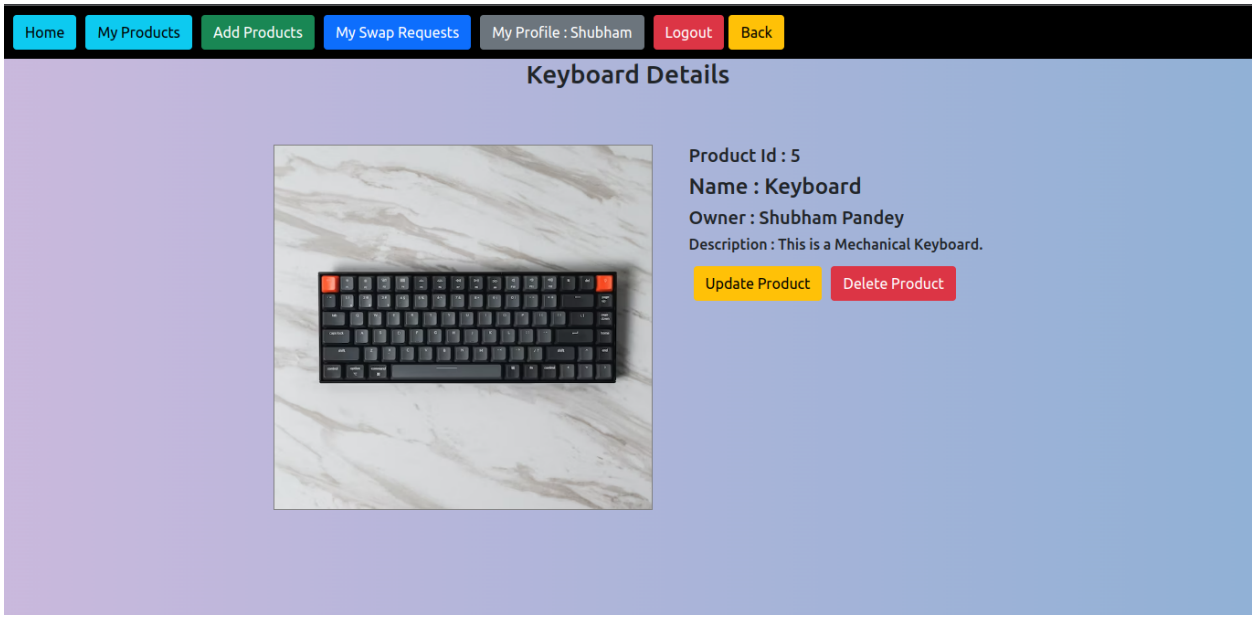
APPLICATION SCREENSHOTS

Home Page -

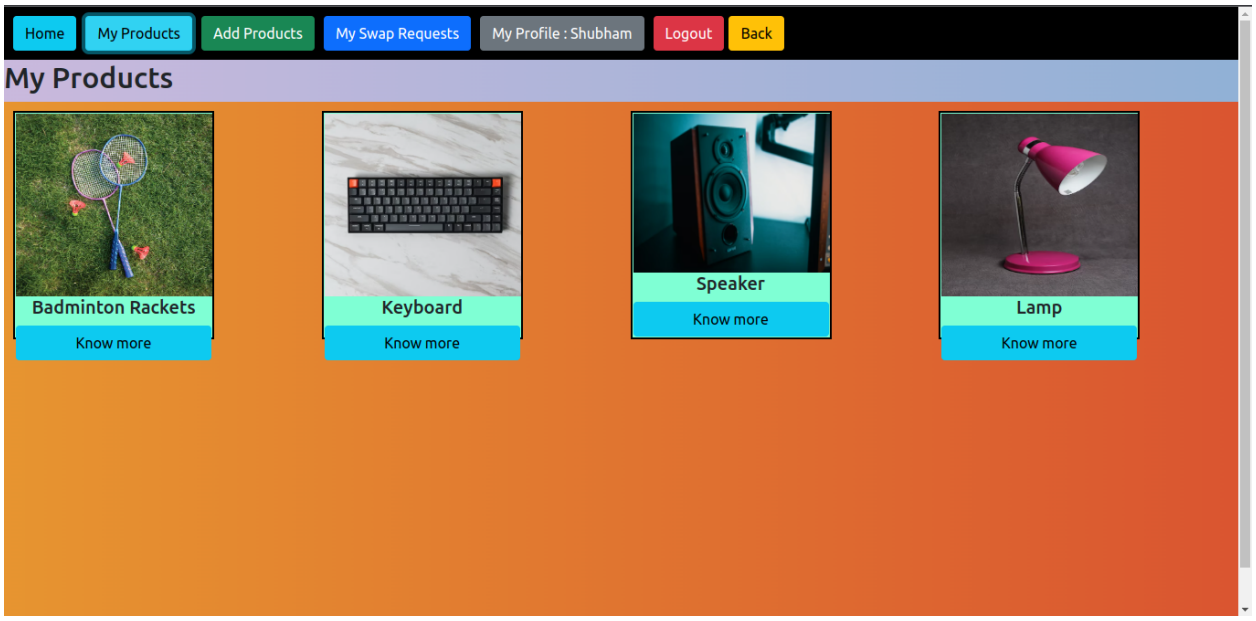


Product Info Page -

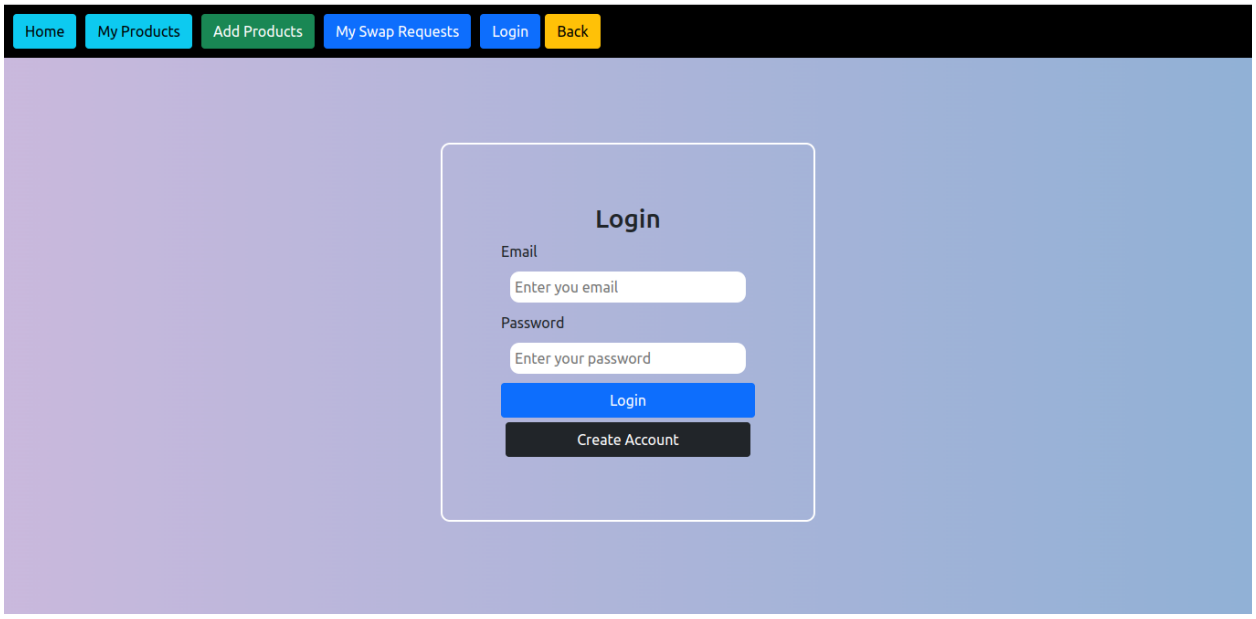
Product Info Page(Owner) -



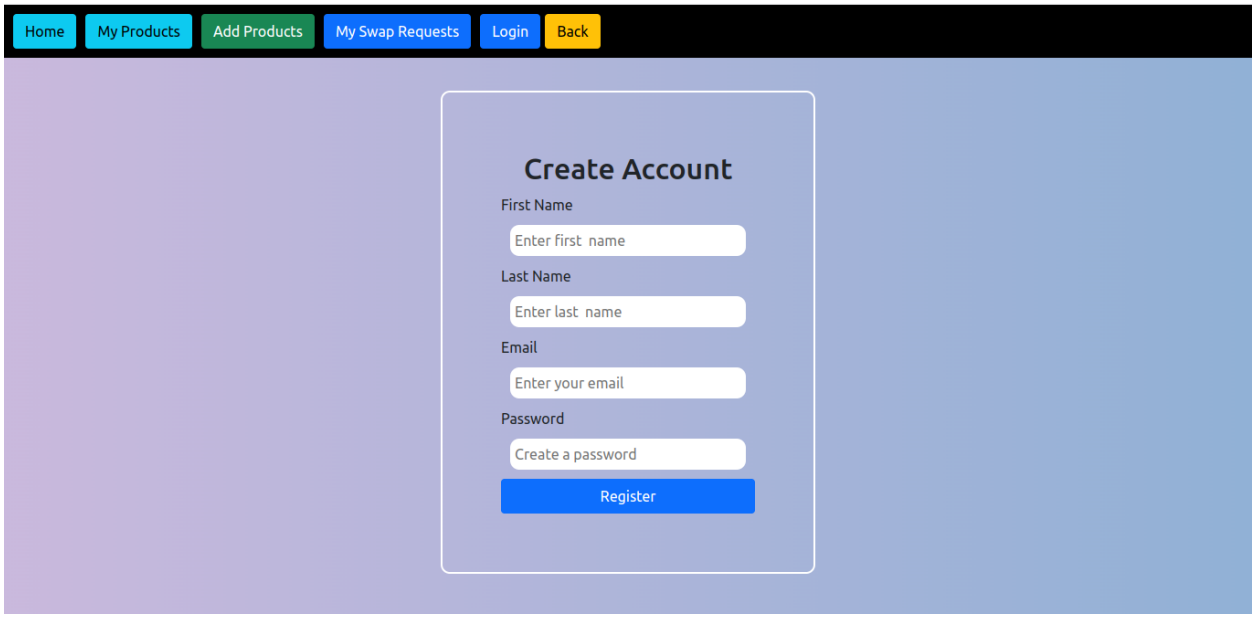
My Products Page -



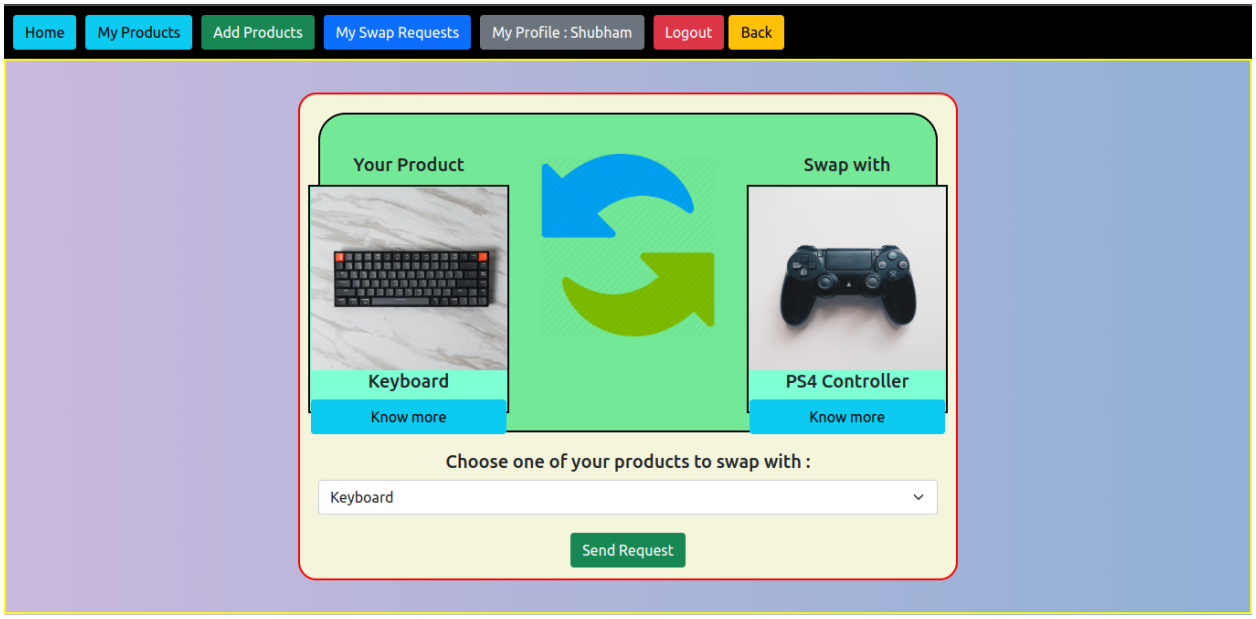
Login Page -



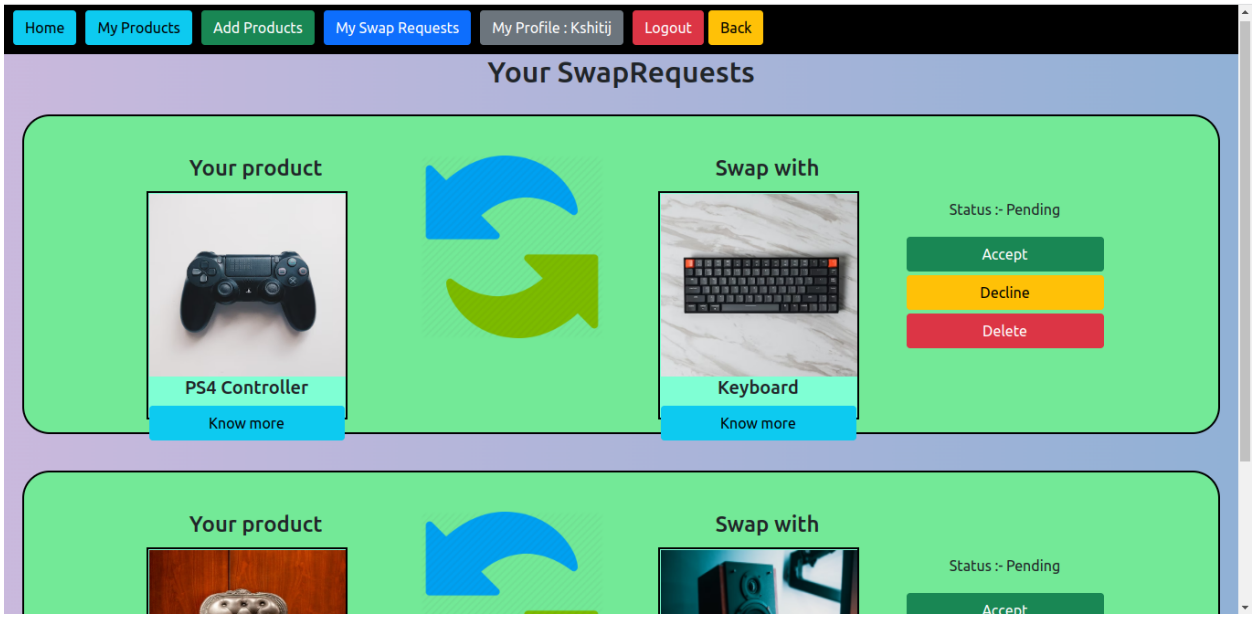
Registration Page -



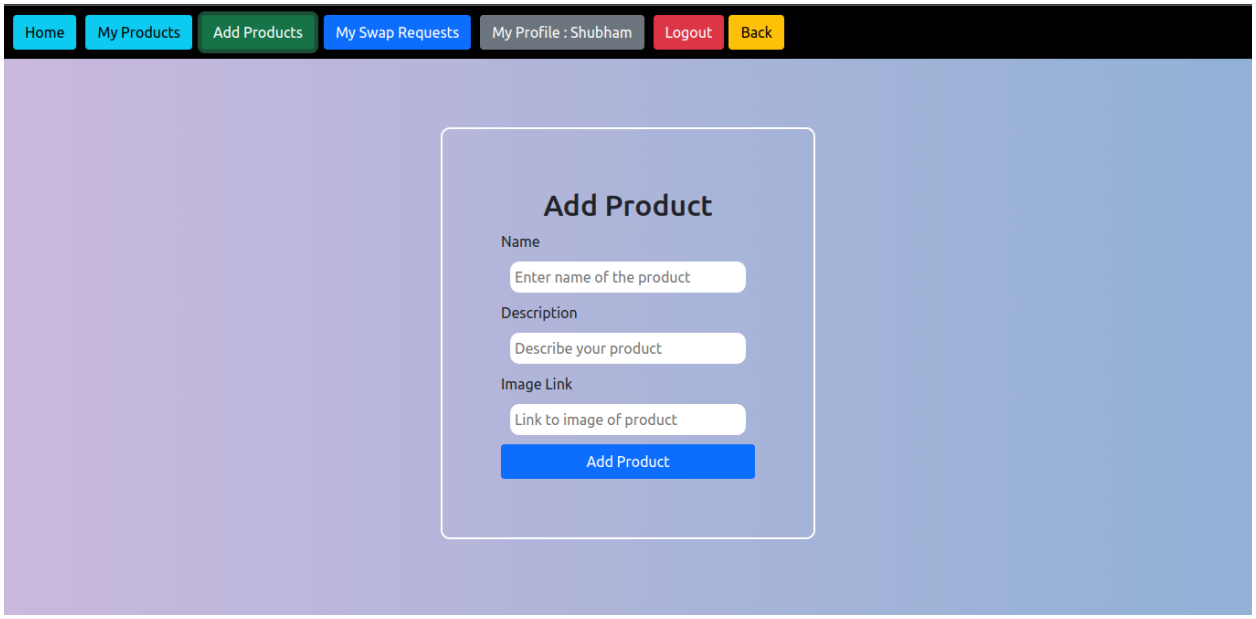
Request Swap -



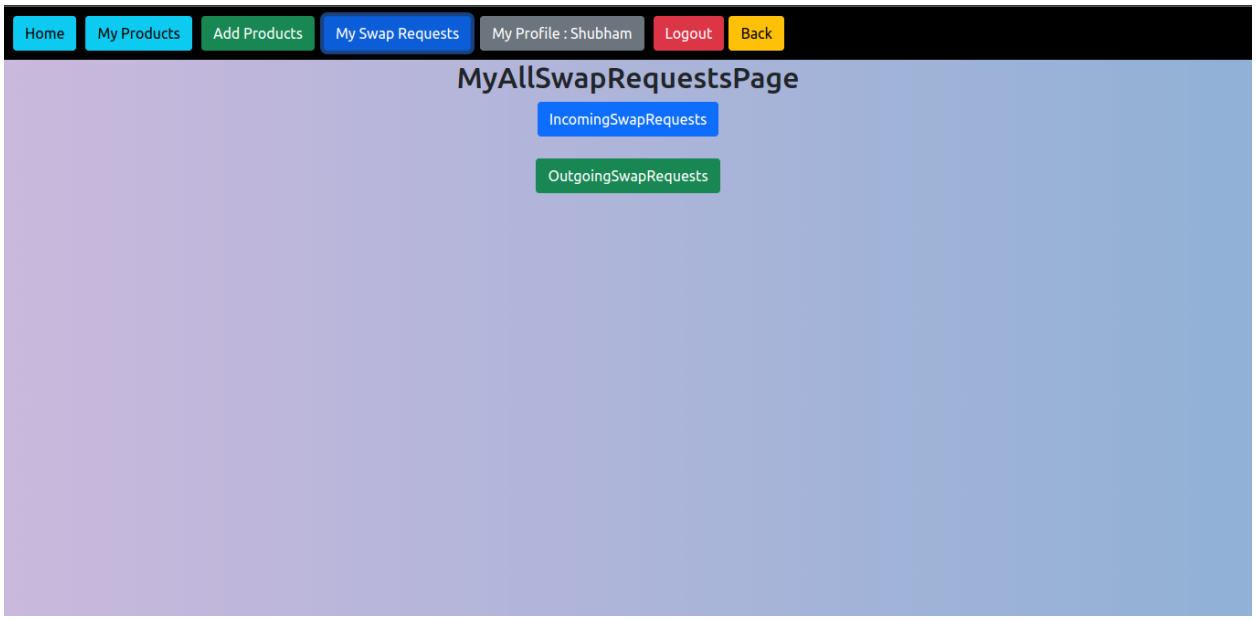
Incoming Request Page -



Add Product Page-



All Request Page -



DEVOPS

WHY DEVOPS?

We have integrated DevOps Approach because of the following advantages:

IMPROVED CUSTOMER EXPERIENCE AND SATISFACTION

The primary goal of DevOps is to deliver higher quality software to end users at a faster pace, driving topline benefits around improved customer experience and increased revenue opportunity.

BREAKING DOWN THE SILOS between Development and Operations

It has ended the old linear process of one team completing all tasks associated with a discrete project before passing it over to another team to work on and so on. The result is a much more flexible and dynamic approach to systems development and -deployment.

ALIGNING IT AND BUSINESS

The biggest business problem in the new fully digitalized world is not the cost of IT operations or the DevOps team – it is the lost opportunity on not executing your business service delivery with enough quality and speed compared with the new breed of competitors attacking your business segment.

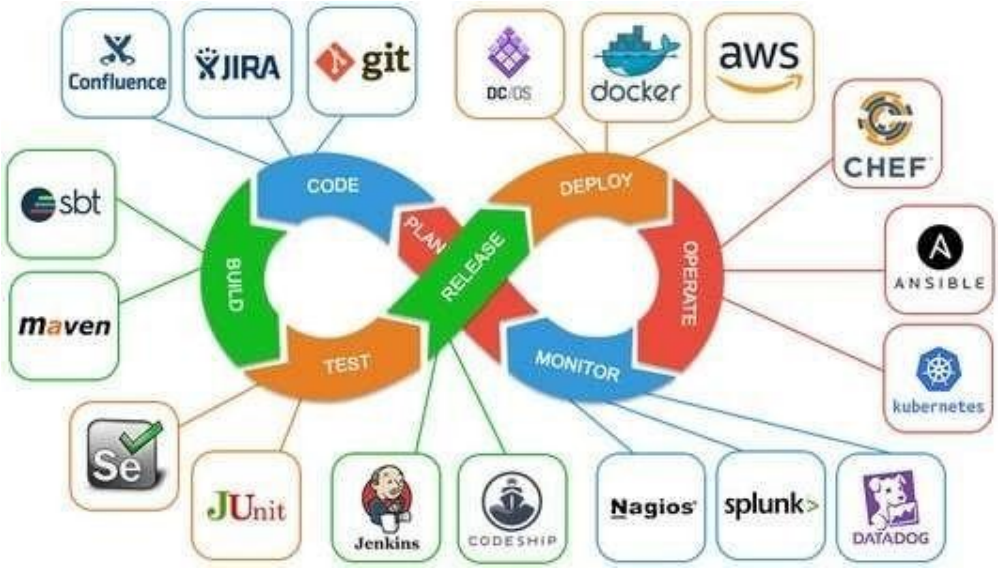
AGILITY

The key advantage for adopting DevOps in the current business environment is business agility. As the rate of change for business accelerates, companies are less able to predict where business is heading. The top strategic imperative becomes responding rapidly via agility and modularity through DevOps and adaptive IT.

VELOCITY: INCORPORATING FEEDBACK

The most important bottom-line advantage is the ability to obtain continuous feedback and incorporate it more expeditiously into application development. This leads to increased revenue and customer satisfaction. In the digital business era, customers' needs are evolving so rapidly that their needs can no longer wait a year or more to incorporate suggestions. The turnaround needs to be much more quick and seamless, which requires many organizations to change their tools, processes and culture.

Following figure shows various tools used in the DevOps approach:



VISIBILITY TO BUILD, RUN AND SECURE MODERN APPLICATIONS

More and more demands are being placed on application teams. They are being asked to get to market quicker by driving faster delivery of software. The quickest way to get there is by leveraging centralized log management and real-time machine analytics throughout the software delivery lifecycle to enable real-time application and business insights.

INNOVATION

DevOps provides the ability for teams to deliver innovation rapidly (multiple times a day)

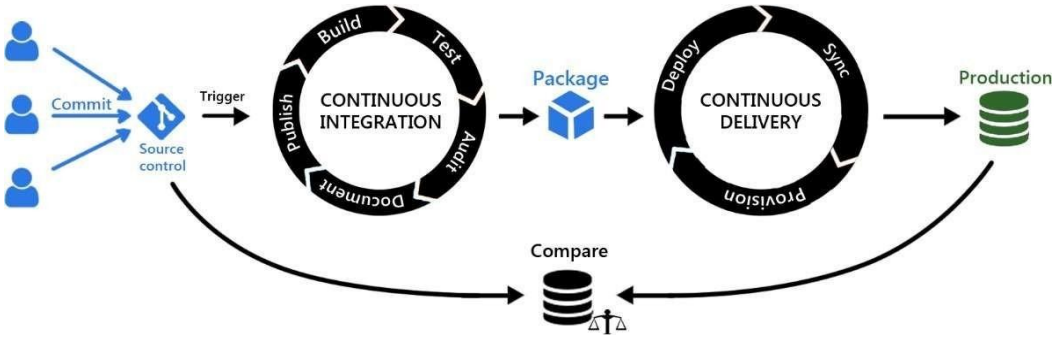
SOFTWARE STABILITY AND QUALITY

DevOps yields greater stability and reliability because faster and more frequent release cycles allow us to identify and resolve issues immediately. This means that developers and others throughout the company have more time to focus on improvements and innovations that contribute to the bottom line.

REDUCED RISK OF CHANGE

Traditional IT has always feared change, which is the main root cause for most of operational issues. A way to minimize change was to slow down the delivery processes with numerous review, assessment, and approval workflows. However, today change is not only inevitable but necessary in order to deliver the speed and agility expected from IT by business. Transition to DevOps and automation of the entire change lifecycle from build to run as a single integrated end-to-end process should minimize the risk of introduced changes while accelerating pace of changes.

Following figure illustrates the DevOps workflow -



TEAM EMPOWERMENT

A critical component in executing DevOps is trust in everyone that contributes to the continuous delivery chain. A culture of trust means empowerment of every individual, ultimately resulting in more motivated employees producing better, timely output.

OPTIMIZING AND STREAMLINING PROCESSES

DevOps brings the confidence teams across the entire enterprise need to move forward quickly. By replacing the loosely coupled and error-prone handoffs used in the traditional waterfall process with a continuous pipeline that leads from development to operations, teams learn from experience that they can move quickly, deploy small changes often, detect issues and opportunities in near real-time and as often as possible and finally, react by rolling forward, not by reverting.

IMPROVING PRODUCTIVITY

Just as traditional manufacturing firms such as car production lines have pioneered and applied lean manufacturing techniques to the assembly line to cut out waste, cut down defect rates, and produce higher quality cars quicker, DevOps helps bring in the same discipline and efficiency to the software development lifecycle.

SAVING TIME AND MONEY

By focusing on performance throughout the lifecycle – not just the testing phase – DevOps teams can prevent bugs from getting deeply baked into products, when they become harder to fix. Excessive, costly rework is avoided, while paving the way for highly satisfying and engaging user experiences. This equates to improved profitability and competitive edge.

COMPETITIVE ADVANTAGE

With faster time to market and continuous incorporation of user feedback businesses can maintain a competitive advantage.

GREATER BUSINESS VALUATION

As DevOps teams become more proficient at recovering from failures and implement changes more frequently their efficiency improves dramatically. This in turn positions them to widen the performance gap between themselves and their sharing(common) group.

PROJECT WORKFLOW

We divided our project workflows into two parts:

1. Initial Setup

This is the one time setup for setting up the deployment environment and deploying the application.

This workflow follows the following steps:

- Starting up servers - Ubuntu docker image
- Adding them as Ansible Node
- Adding playbook for the node
- Pulling the configuration
- Build Step: Jenkins pulls the updated source code from the GitHub repository and then invokes maven to build and package the application as a .jar file
- Copying the .jar file on the server containers using ansible
- Before deploying the updated application we stop the currently running instance using ansible
- The application is now deployed using ansible

2. Continuous Deployment “Single Click Deployment”

This workflow is designed to push any infrastructure or code changes directly to production on a single click. This workflow follows the following steps:

This workflow follows the following steps:

- Pulling the configuration
- Build Step: Jenkins pulls the updated source code from the GitHub repository and then invokes maven to build and package the application as a .jar file
- Copying the .jar file on the server containers using ansible
- Before deploying the updated application we stop the currently running instance using ansible
- The application is now deployed using ansible

SOFTWARE DEVELOPMENT LIFECYCLE

1. SOURCE CODE MANAGEMENT (SCM)

We used Git as the Version Control System integrated with GitHub for online repository management.

The repository for the project is available as a public repository at the following link https://github.com/Shubhamp194/Swapsie-frontend and https://github.com/Shubhamp194/Swapsie-backend.

We choose Git as the VCS because of following benefits that are offered:

1. PERFORMANCE

Git performs very strongly and reliably when compared to other version control systems. New code changes can be easily committed, version branches can be effortlessly compared and merged, and code can also be optimized to perform better.

2. SECURITY

Git is designed specially to maintain the integrity of source code. File contents as well as the relationship between file and directories, tags, commits, versions etc. are secured cryptographically using an algorithm called SHA1 which protects the code and change history against accidental as well as malicious damage.

3. FLEXIBILITY

A key design objective of Git is the kind of flexibility it offers to support several kinds of nonlinear development workflows and its efficiency in handling both small scale and large-scale projects as well as protocols.

4. WIDE ACCEPTANCE

Git offers the type of performance, functionality, security and flexibility that most developers and teams need to develop their projects. When compared to other VCS Git is most widely accepted system owing to its universally accepted usability and performance standards.

5. DISTRIBUTED DEVELOPMENT

Since Git is a distributed VCS it offers a local repository to each developer with its own history of commits. Therefore, you don't require a network connection to create commits, inspect previous file versions, or check differences between two or more commits. Also, it’s much easier to scale the team. With Git, users can be easily added or

removed as and when required. Other team members can continue working using their local
repositories and are not dependent upon whether a new branch is added or an older one closed.

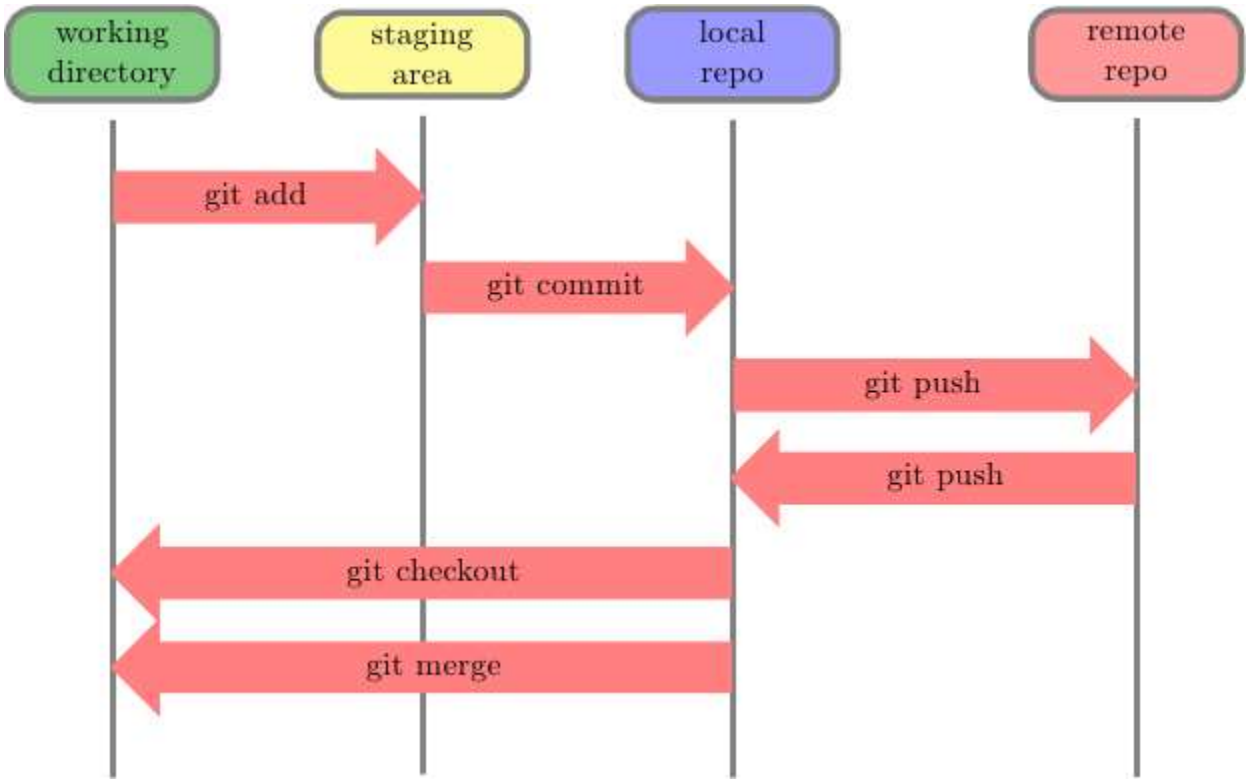
6. PULL REQUESTS

A developer calls a pull request to ask another developer to merge one of his/her branches into the other's repository. Besides making it a lot easier for project leaders to monitor and track code changes, "pulling" also facilitates other developers to discuss their work before integrating the code with the codebase.

7. BRANCH WORKFLOW

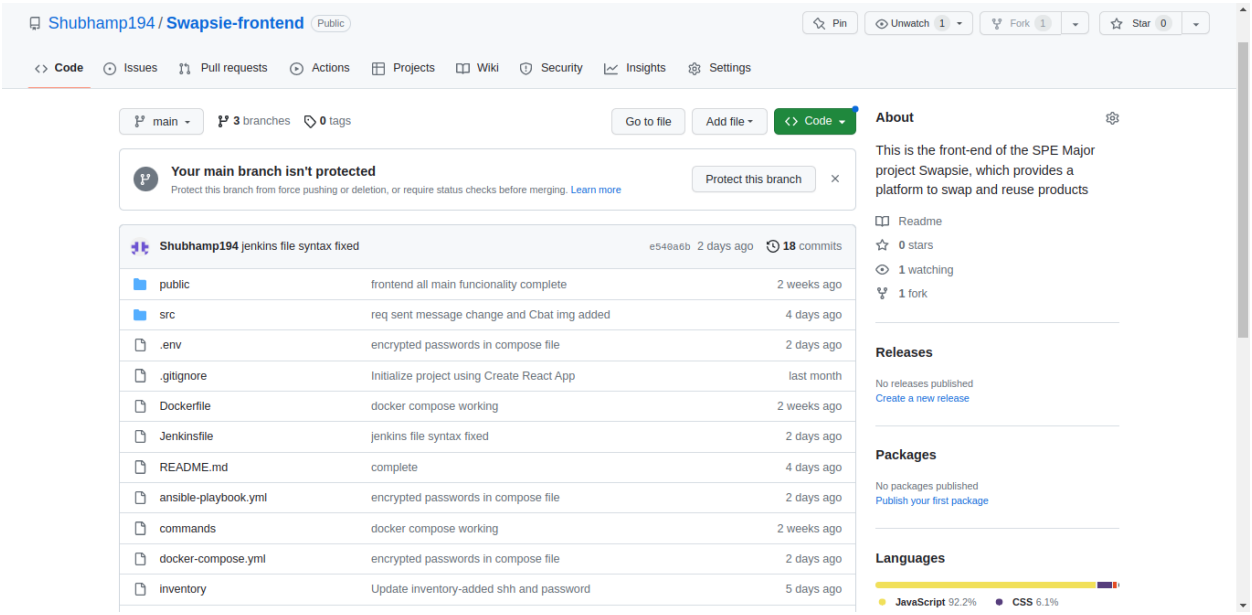
Git has powerful branching capabilities. To start work, developers have to first create a unique branch. Each branch functions in an isolated environment while changes are carried out in the codebase. This ensures that the master branch always supports production-quality code.

Following screenshot shows the workflow followed by Git:

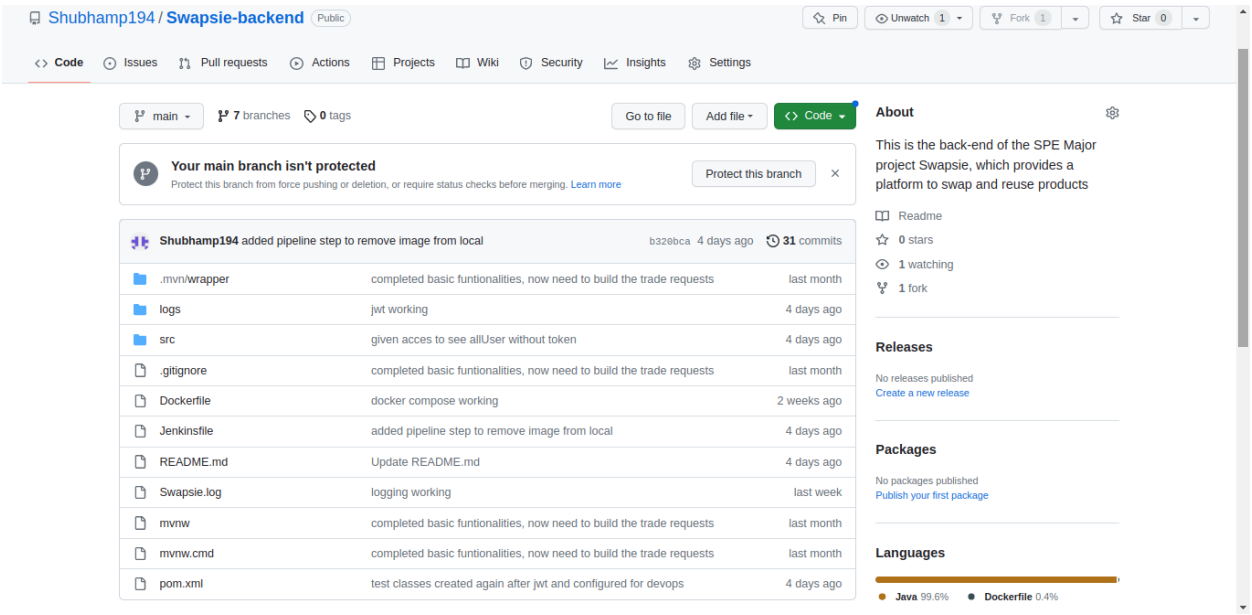


Following screenshot shows the github home page for the application:

FRONTEND-



BACKEND-

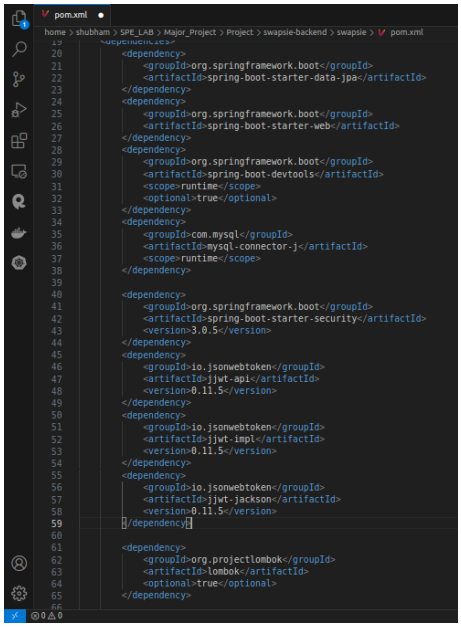


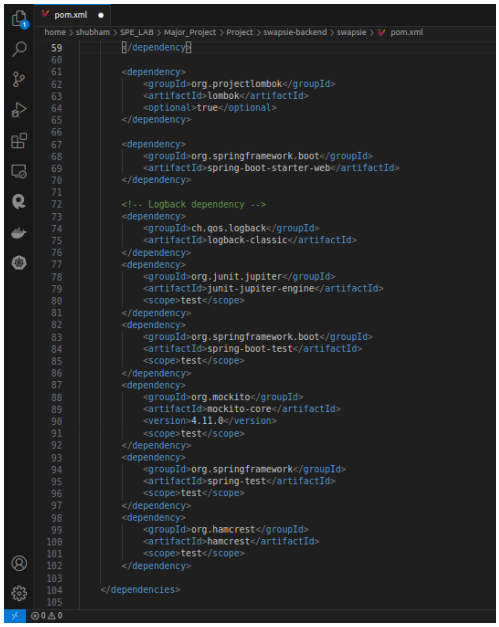
2. BUILD

We used Maven integrated with Jenkins for building the application as well as for dependency management. Maven made the dependency management task super easy.

For the build step we integrated Maven with Jenkins so that whenever the build step is triggered Jenkins first pulls the updated code from the Github repo and then invokes maven to build the application.

Following is the pom.xml that we defined for our project.



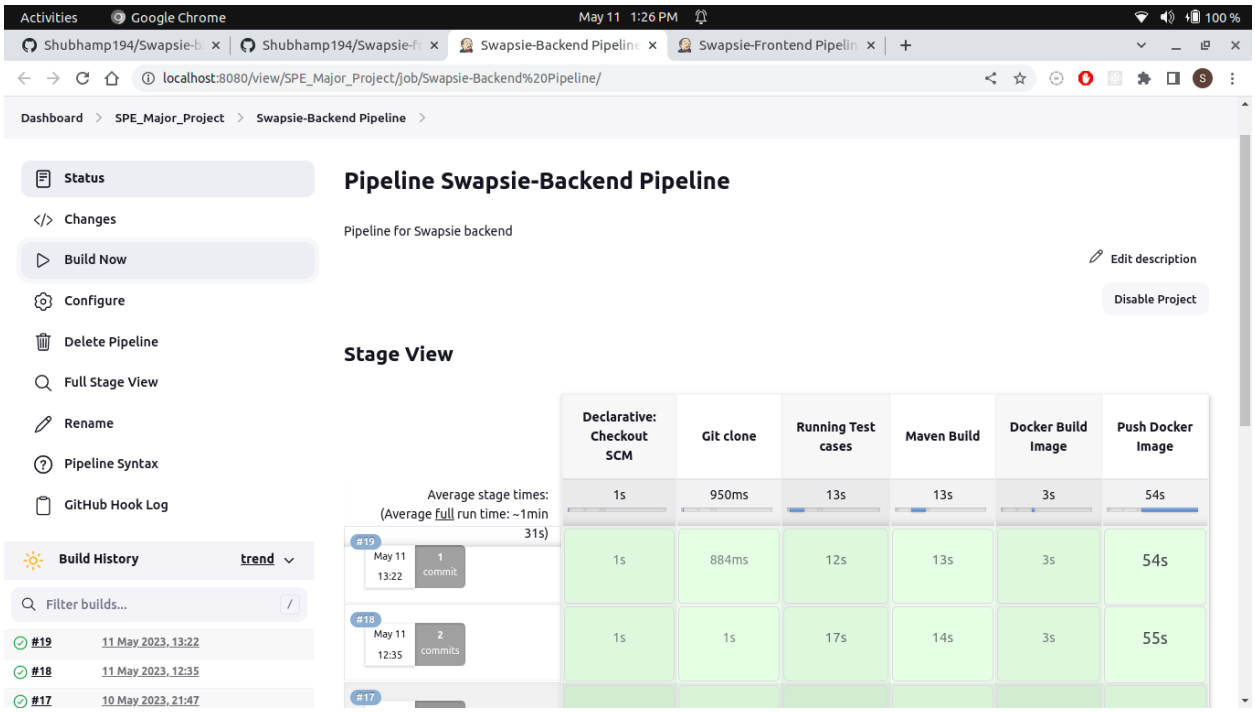


Maven offers the following benefits:

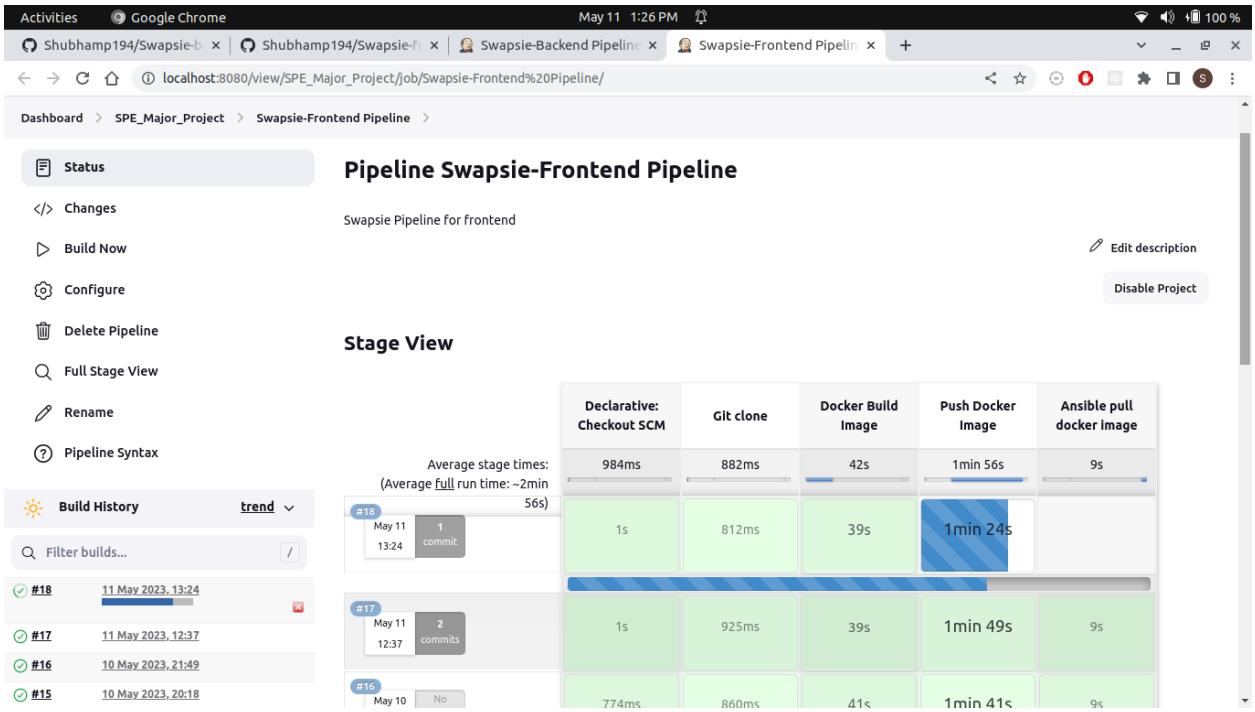
- All artifacts are versioned and store in a repository
- Build process is standardized for all projects
- Provide quality project information with auto generated site
- Easy to learn and use
- Promotes and enforces modular design of code

PIPELINE SCREENSHOTS -

Backend Pipeline-



Frontend Pipeline-

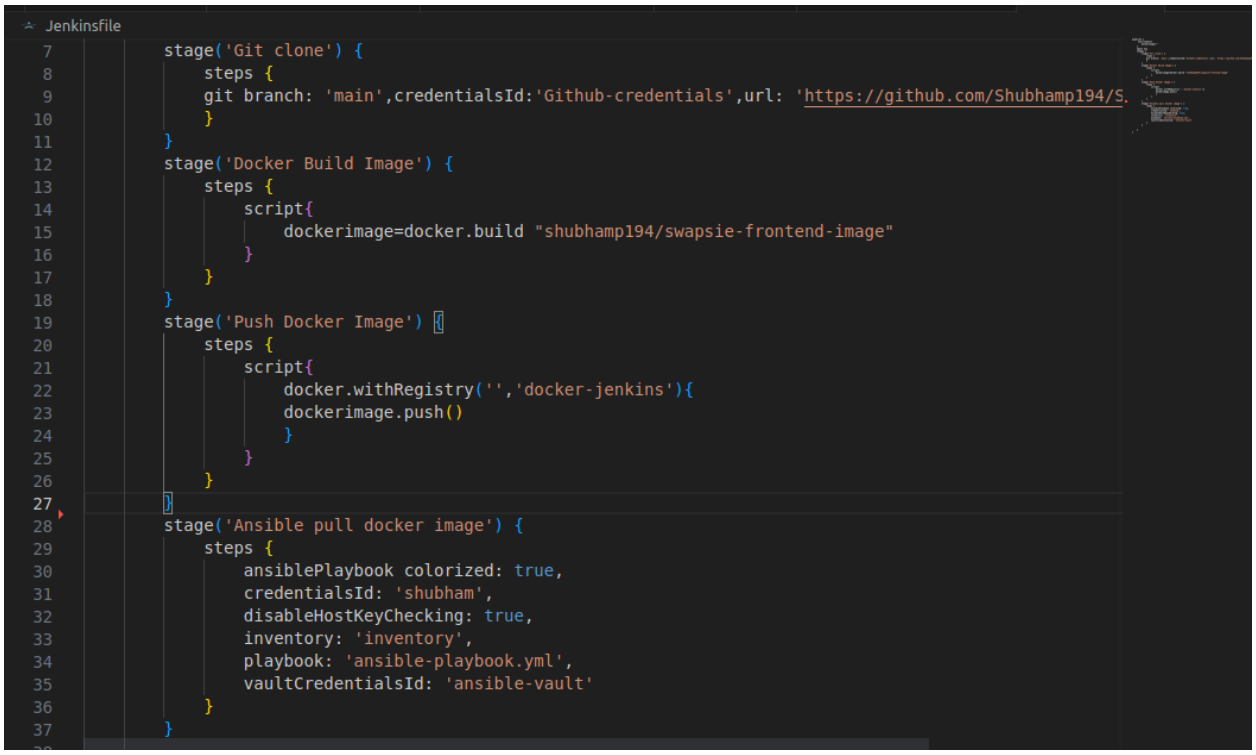


Backend Jenkins file -

```
stages {
    stage('Git clone') {
        steps {
        git branch: 'main',credentialsId:'Github-credentials',url: 'https://github.com/Shubhamp194/Swapsie-backer
        }
    }

    stage("Running Test cases"){
        steps{
            sh "mvn clean test"
        }
    }

    stage("Maven Build"){
        steps{
            sh "mvn clean install"
        }
    }

    stage('Docker Build Image') {
        steps {
            script{
                dockerimage=docker.build "shubhamp194/swapsie-backend-image"
            }
        }
    }
```

```
        steps{
            sh "mvn clean install"
        }
    }

    stage('Docker Build Image') {
        steps {
            script{
                dockerimage=docker.build "shubhamp194/swapsie-backend-image"
            }
        }
    }
    stage('Push Docker Image') {
        steps {
            script{
                docker.withRegistry('','docker-jenkins'){
                dockerimage.push()
                }
            }
        }
    }
}
}
```

Frontend Jenkins file -



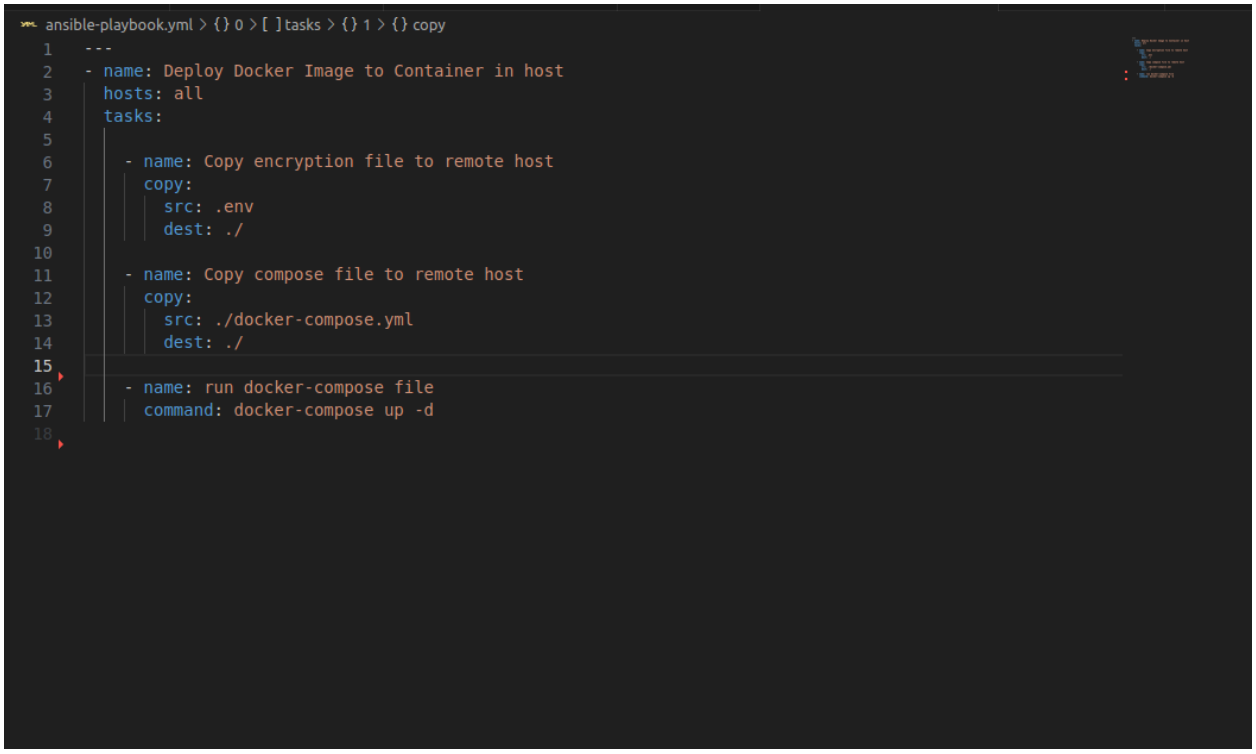
```
Jenkinsfile
7           stage('Git clone') {
8               steps {
9               git branch: 'main',credentialsId:'Github-credentials',url: 'https://github.com/Shubhamp194/S.
10              }
11          }
12          stage('Docker Build Image') {
13              steps {
14                  script{
15                      dockerimage=docker.build "shubhamp194/swapsie-frontend-image"
16                  }
17              }
18          }
19          stage('Push Docker Image') {
20              steps {
21                  script{
22                      docker.withRegistry('','docker-jenkins'){
23                      dockerimage.push()
24                      }
25                  }
26              }
27          }
28          stage('Ansible pull docker image') {
29              steps {
30                  ansiblePlaybook colorized: true,
31                  credentialsId: 'shubham',
32                  disableHostKeyChecking: true,
33                  inventory: 'inventory',
34                  playbook: 'ansible-playbook.yml',
35                  vaultCredentialsId: 'ansible-vault'
36              }
37          }
```

CONFIGURATION MANAGEMENT

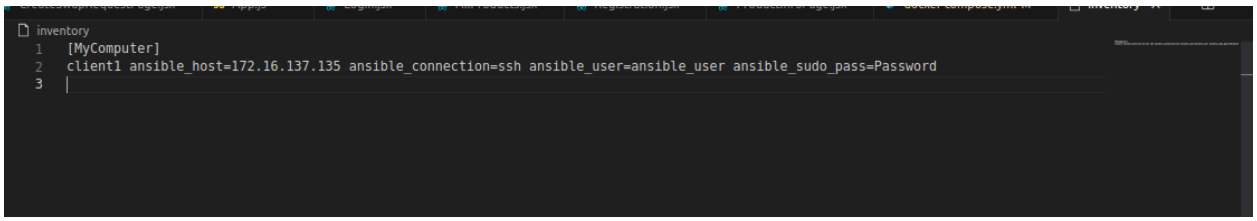
We used ansible for Configuration Management. For application deployment we need java 17 to be installed on the servers and ansible handles this installation automatically in a scalable manner. Also, any future update to the configuration can be as easily done as just changing a single playbook.

We used ansible “java” for Swapsie. This playbook was taken from what we used in the mini-project, and we modified it according to our requirement.

Following code snippet shows the changes made to the default playbook:



Inventory file -



All the step of adding servers as ansible node, adding playbook lists and pulling configuration has been automated using ansible. This entire process has been explained in the deployment section.

Ansible offers the following benefits:

1. Accelerating Software Delivery

High-performing IT organizations are the ones that are able to deploy their software on demand and within one hour of a new commit. Automating your IT infrastructure – building new environments, testing and reviewing changes to the code base, deploying new software versions and more – is one of the most important shifts you can make to bring your business up to speed.

2. Increasing Service Resiliency

An application's efficiency and resiliency are measured by two metrics: the rate that new changes break the system and the mean time to recover from downtime. Efficient IT organizations typically have a change failure rate of less than 15 percent, and take less than an hour to recover.

3. Improving Risk Management

Ansible's infrastructure automation capabilities are able to lower risk and improve compliance at all stages of development. Your compliance and security policies can be encoded as part of a ansible playbook and tested automatically before deployment.

4. Accelerating Cloud Adoption

By automating your cloud infrastructure and handling routine manual actions, Ansible frees up time for your DevOps team to be more agile and efficient. Your applications and workloads can be moved quickly and smoothly, while your servers and infrastructure are automatically installed, configured, and provisioned according to the Ansible playbook that you write ahead of time.

5. Managing Both Data Center and Cloud Environments

Under the Ansible umbrella, you can manage all your cloud and on-premises environments at once, including servers running the Windows, Linux, IBM AIX, and Solaris operating systems. Ansible is also a"cloud agnostic" solution, allowing you to keep using it even as you change cloud providers.

6. Delivering All Your Infrastructure –Any App, Everywhere, Continuously

Ansible can cut through this complexity to streamline your IT operations and workflow. From building and testing all the way through delivery, monitoring, and troubleshooting, Ansible provides a pipeline for continuous deployment that you can use to achieve more and make better decisions.

DEPLOYMENT

We use Ansible along with Jenkins for continuous deployment. The workflow for deployment is as follows:

1. Pull docker image
We pull the created and pushed docker image from docker hub for the application.

2. Running the docker container
We run the container using a pulled docker image on the deployment node.

Backend Dockerfile-

```
# Fetching latest version of Java
FROM openjdk:17

# Setting up work directory
WORKDIR /app

# Copy the jar file into our app
COPY ./target/swapsie-0.0.1-SNAPSHOT.jar /app

# Exposing port 8080
EXPOSE 8081

# Starting the application
CMD ["java", "-jar", "swapsie-0.0.1-SNAPSHOT.jar"]
```

Frontend Dockerfile-

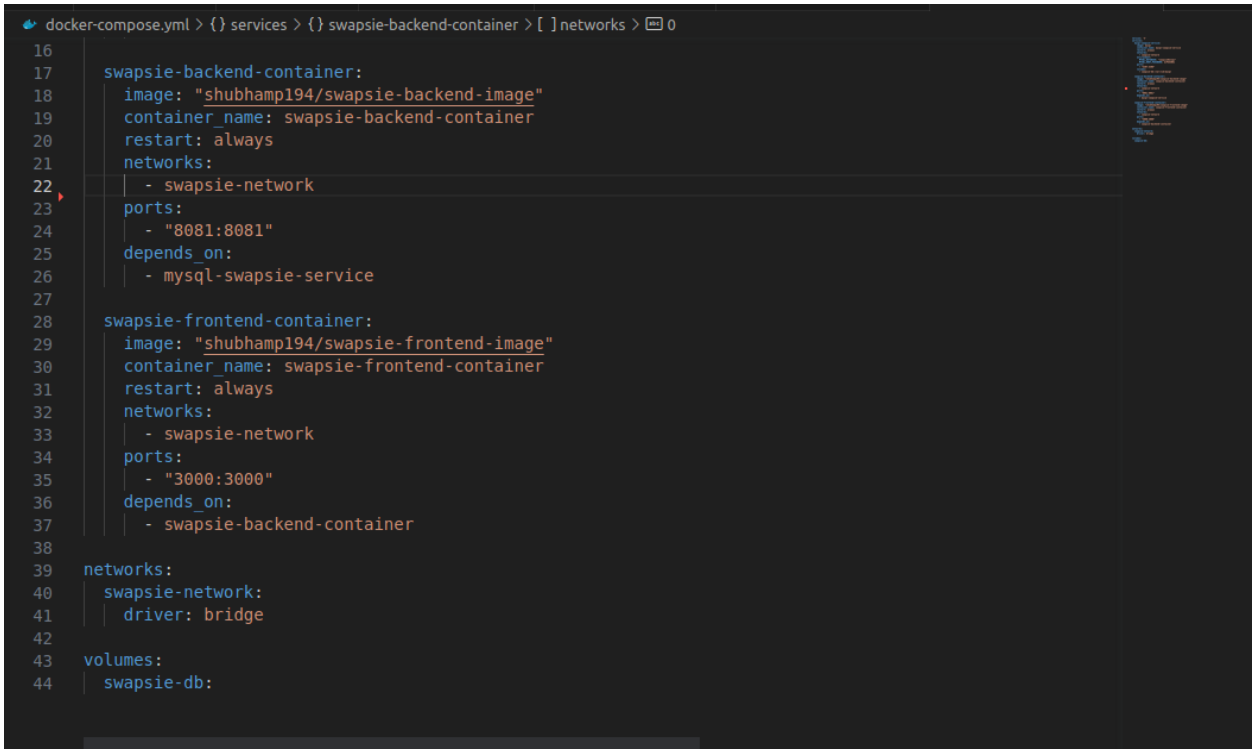
```
Dockerfile
1  FROM node:alpine
2  WORKDIR /app
3  COPY package.json ./
4  COPY package-lock.json ./
5  COPY ./ ./
6  RUN npm i
7  EXPOSE 3000
8  CMD ["npm", "run", "start"]
```

Docker-compose file -

docker-compose.yml > {} services > {} swapsie-backend-container > [ ] networks > 0

docker-compose.yml - The Compose specification establishes a standard for the definition of multi-container platform-agnostic applications. (compose-spec.json)

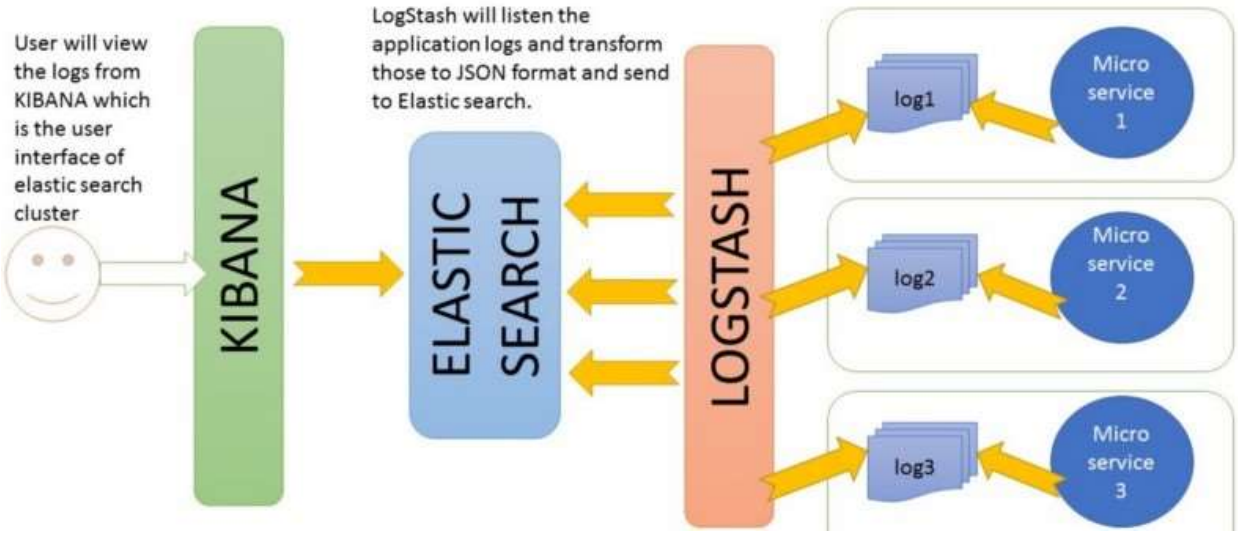
```yaml
version: '3'
services:
  mysql-swapsie-service:
    image: mysql
    container_name: mysql-swapsie-service
    restart: always
    networks:
      - swapsie-network
    environment:
      MYSQL_DATABASE: "swapsieDevops"
      MYSQL_ROOT_PASSWORD: ${PASSWD}
    ports:
      - "3307:3306"
    volumes:
      - swapsie-db:/var/lib/mysql

  swapsie-backend-container:
    image: "shubhamp194/swapsie-backend-image"
    container_name: swapsie-backend-container
    restart: always
    networks:
      - swapsie-network
    ports:
      - "8081:8081"
    depends_on:
      - mysql-swapsie-service

  swapsie-frontend-container:
    image: "shubhamp194/swapsie-frontend-image"
    container_name: swapsie-frontend-container
    restart: always
```



```yaml
  swapsie-backend-container:
    image: "shubhamp194/swapsie-backend-image"
    container_name: swapsie-backend-container
    restart: always
    networks:
      - swapsie-network
    ports:
      - "8081:8081"
    depends_on:
      - mysql-swapsie-service

  swapsie-frontend-container:
    image: "shubhamp194/swapsie-frontend-image"
    container_name: swapsie-frontend-container
    restart: always
    networks:
      - swapsie-network
    ports:
      - "3000:3000"
    depends_on:
      - swapsie-backend-container

networks:
  swapsie-network:
    driver: bridge

volumes:
  swapsie-db:
```

MONITORING

ELK Stack -



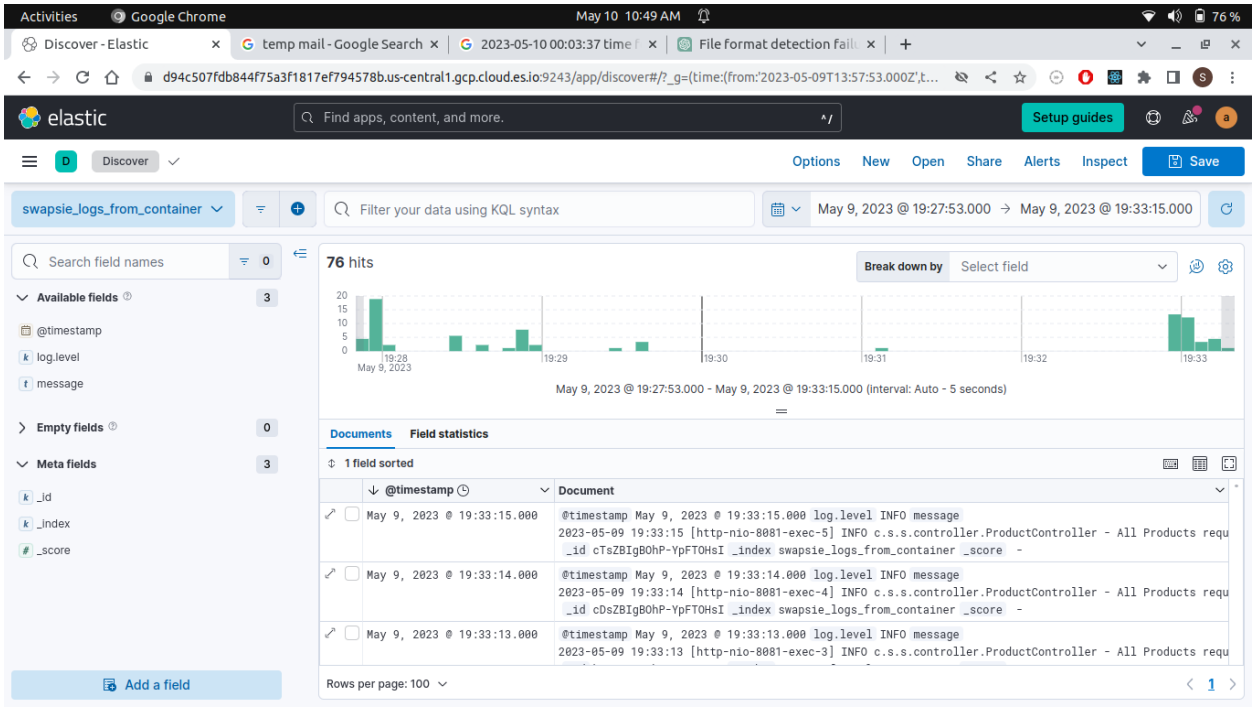
We used Elasticsearch with Logstash and Kibana for monitoring.

Elasticsearch is a substantial REST HTTP service that enables scaling of complex query & search operations even up to thousands of queries per second. So integration of Elasticsearch with mysql database can be proved a powerful value addition to the application.

Kibana is used for analysis and visualization of data stored in Elasticsearch indices in a variety of charts, tables, and graphs. It's a simple, browser-based interface that enables you to quickly create and share dynamic dashboards that display changes to Elasticsearch queries in real time.

Logstash enables the application to collect data from different systems. Moreover, it normalizes different schemas. It enables you to keep the data gathered from various systems into a common & custom format. Before starting Logstash, a Logstash configuration file is created in which the details of input file, output location, and filter methods are specified.

Following image show the logs in ELK -



REFERENCES

[1].https://guides.github.com/

[2].https://spring.io/guides/gs/spring-boot/

[3].https://dev.mysql.com/doc/refman/8.0/en/tutorial.html/

[4].https://tomcat.apache.org/tomcat-8.0-doc/config/valve.html

[5].https://stackoverflow.com/ [6].https://docs.docker.com/get-started/

[7].https://jenkins.io/doc/tutorials/

[8].https://maven.apache.org/guides/getting-started

[9].https://docs.spring.io/spring-boot/docs/current/reference/html/boot-featu res-testing.html [10].https://docs.chef.io/chef_overview.html

[11].https://docs.rundeck.com/docs/tutorials/

[12].https://www.elastic.co/guide/en/elasticsearch/reference/current/gettingstarted.html