PROJECT REPORT

CS 816 SOFTWARE PRODUCTION ENGINEERING

SHARING HAND

HELPS IN SHARING RESOURCES

Shree Sharan Garodia — **MT2021128**

Shwetank Singh — **MT2021133**

ABSTRACT

This application SharingHand is a “Community Item Sharing Application”. This application provides the users a platform for sharing items that are too costly to be bought by every individual or are needed rarely. This platform is flexible enough to accommodate itself to meet the needs of any type of community, be it a school/college group, hostel group, family, friends, societies etc. The main purpose of this application is to promote sharing and create a goodwill among the society.

Apart from that the application has been completely automated using DevOps model with a single click deploy from development to deployment in production. Configuration management for servers has also been automated.

WHY DEVOPS?

We have integrated DevOps Approach because of the following advantages:

1. **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.

2. **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.

3. **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.

4. **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.

5. **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:

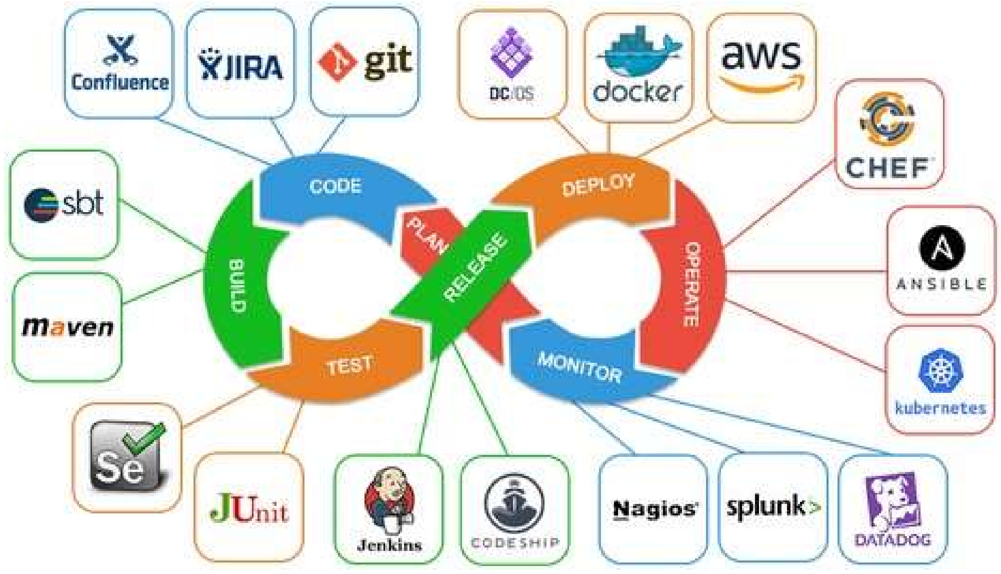


Figure: Example of tools used in DevOps

6. **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.

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

8. **SOFTWARE STABILITY AND QUALITY**
   DevOps yields greater stability and reliability because faster and more frequent release cycles allows 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.

9. **REDUCED RISK OF CHANGE**

Traditional IT have 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:

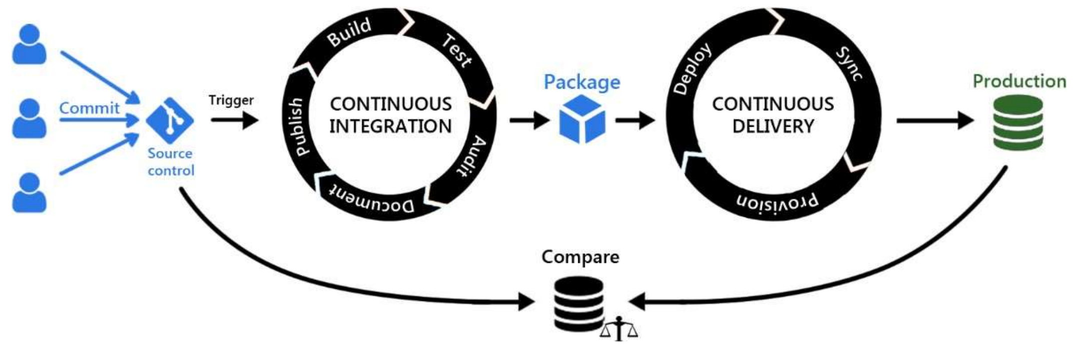


Figure: DevOps Workflow

10. **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.

11. **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.

12. **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.

13. **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.

14. **COMPETITIVE ADVANTAGE**

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

15. **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.

SCOPE OF THE PROJECT

1. Project Goals

Goal of this project is to develop a “Community/Group Sharing Application” (named SharingHand) to provide a platform to facilitate easy and smooth sharing of tools, equipment, sport gear and everything that can be shared among groups and communities such as Friends, Family, Relatives, Societies, Clubs, Study Circles, Office Groups etc.

Secondary objective of this project is to automate every step from development to deployment i.e., to incorporate the complete DevOps framework to facilitate one click deployment to production.

2. Functionality

Version 1 of the software will provide the following basic functionality to the end user:

- Allow users to register and login into the application
- Allow users to create groups and add members that are already registered
- Post request for items in single or multiple groups at the same time
- View and accept the item requests posted by the other members of his groups
- View all the item requests posted by him
- View all the item requests accepted by him
- On accepting a request open a private chat between him and the requestor
- View all the users who have accepted his request
- Option to start a private chat with the user who has graciously accepted the item request
- The entire chat and request history is saved for future reference of the user

A simple user interface has been also provided to facilitate all the above functionality.

3. Tasks

The application needs to be implemented from scratch using Java's Spring Boot framework.

Various automation tools also needs to be integrated some when the development starts and some when the first draft version of the application is ready for deployment to production environment.

4. Deadlines

This project is expected to be completed on or before May 12th, 2022, to keep upwith the semester schedule.

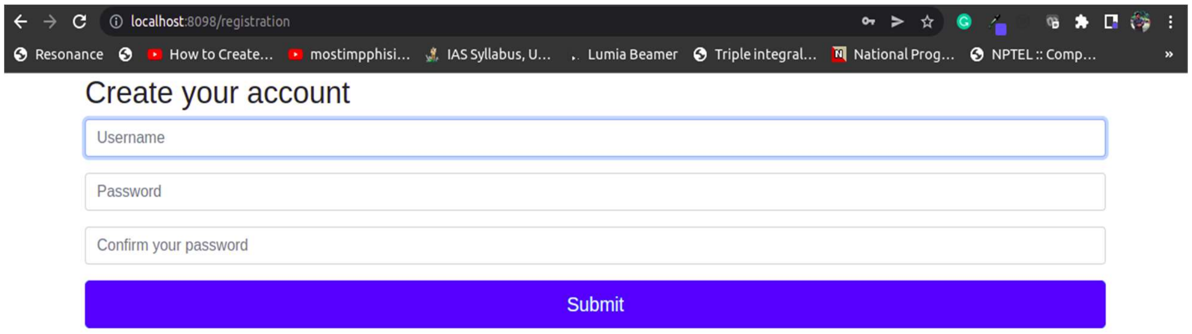
5. Deliverables

Deliverables include the entire project source code, documentation, automation scripts and documentation for setting up the automation tools.

APPLICATION SCREENSHOTS

Following screenshots show the application functionality:

1. Registration Page



Create your account

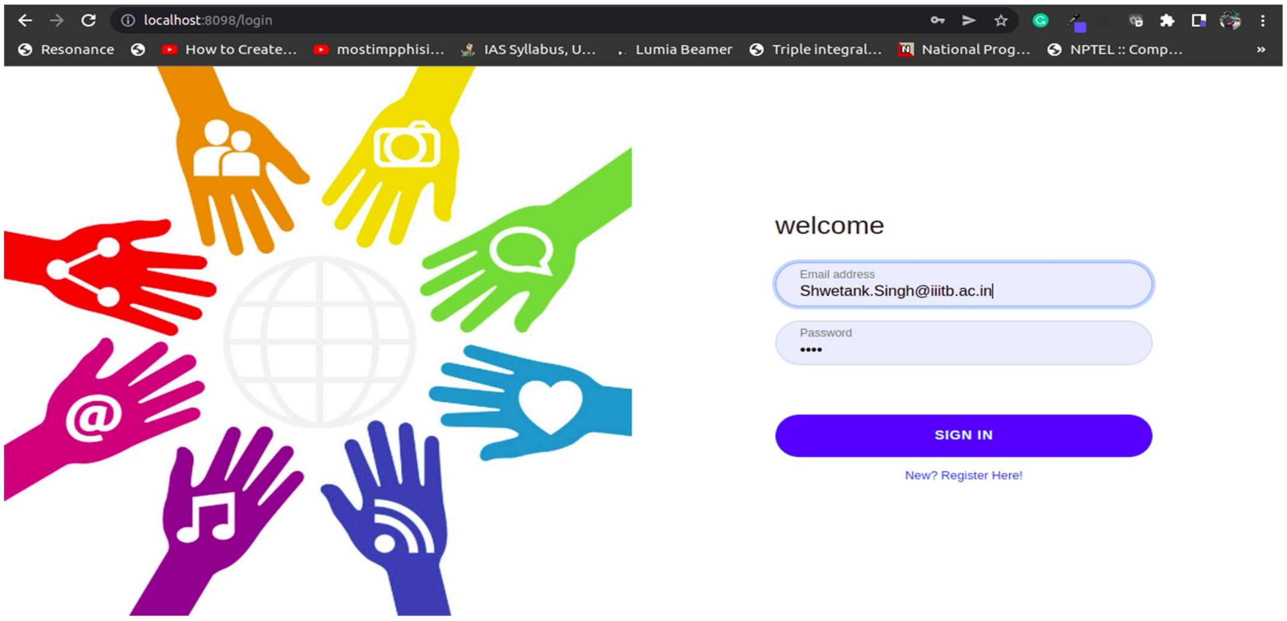
Username

Password

Confirm your password

Figure: Registration Page

2. Login Page



welcome

Email address: Shwetank.Singh@iiitb.ac.in

Password: ••••

Figure: Login Page

3. Home Page

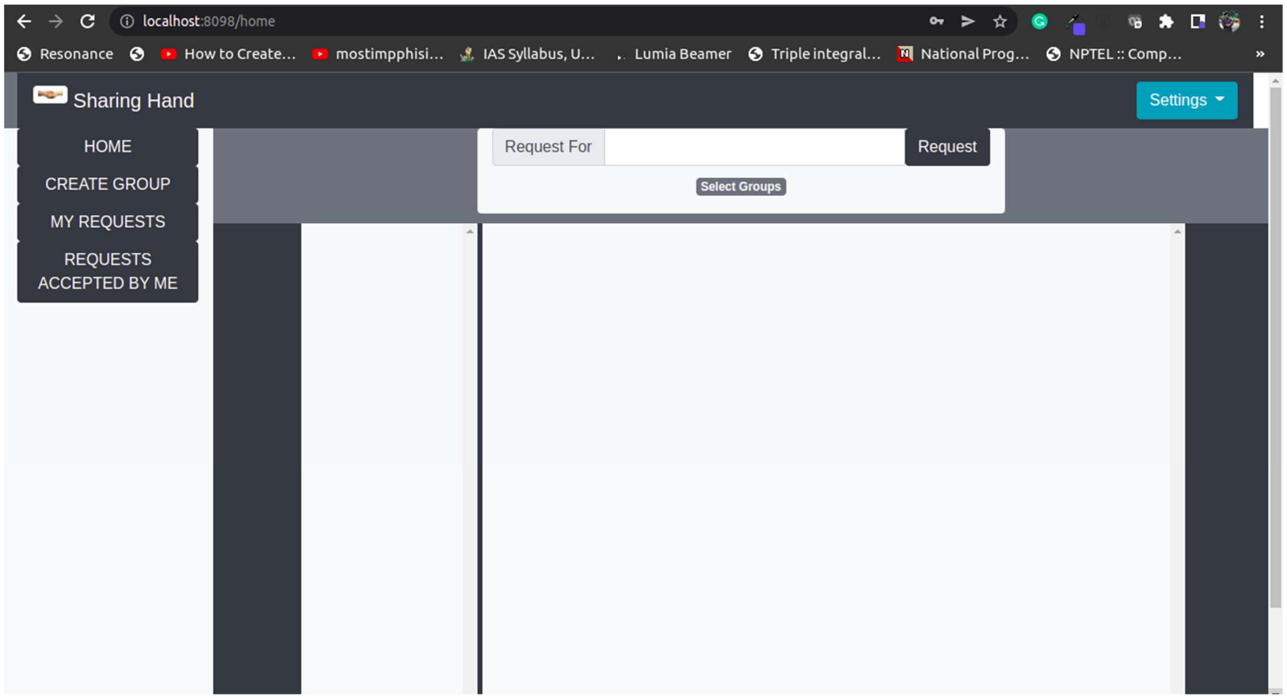


Figure: Home Page

4. Create Group Page

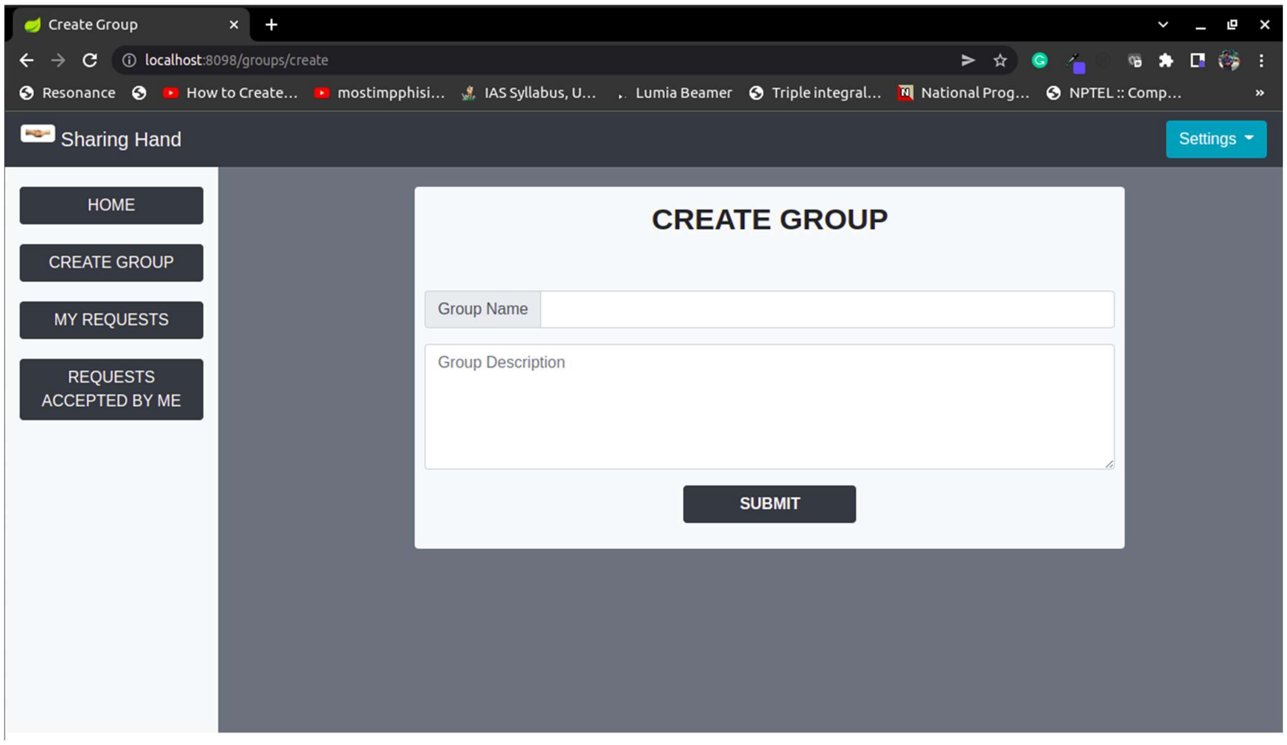


Figure: Create Group Page

5. Create Requests Page

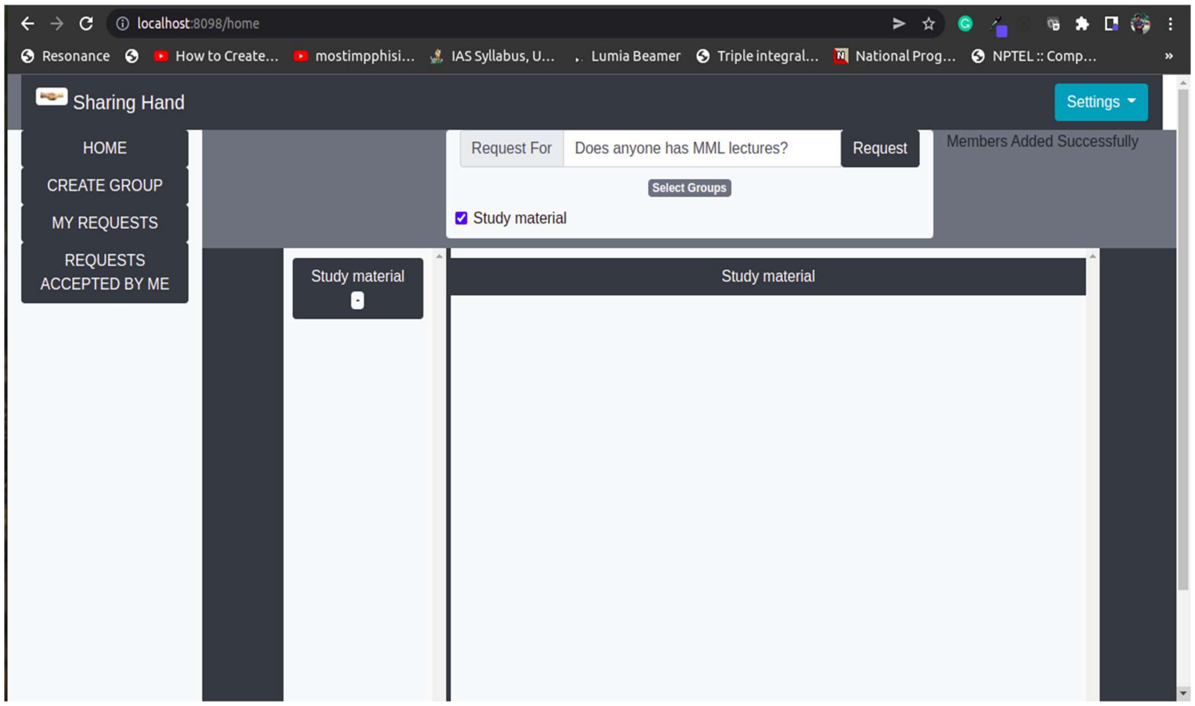
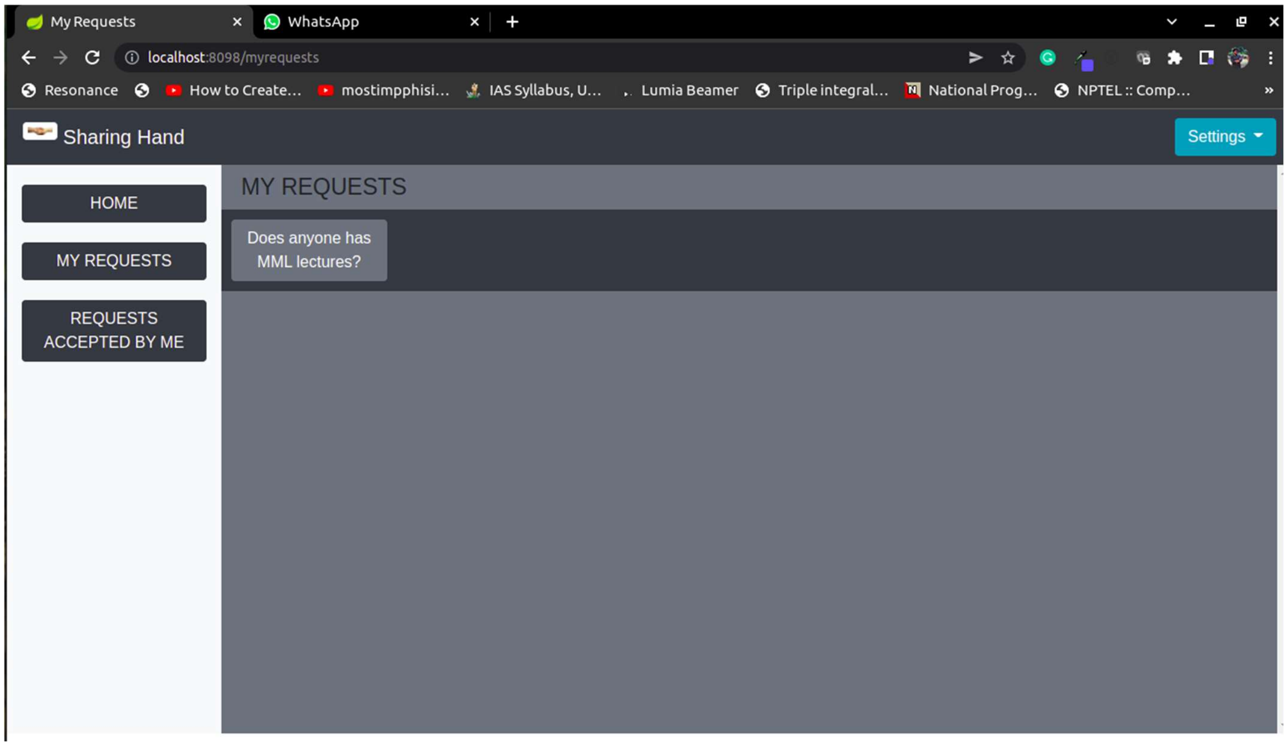


Figure: Create Requests Page

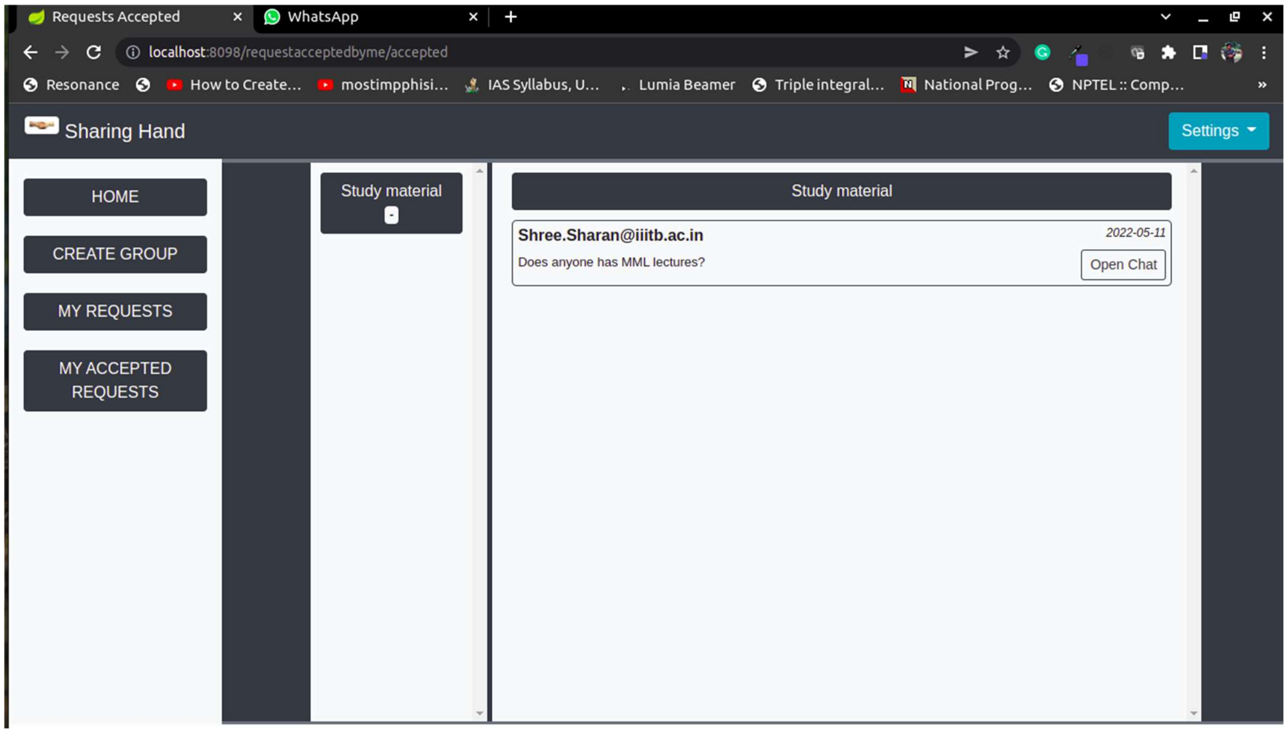
6. My Requests Page



Does anyone has MML lectures?

Figure: My Requests Page

7. Request Accepted By Me Page



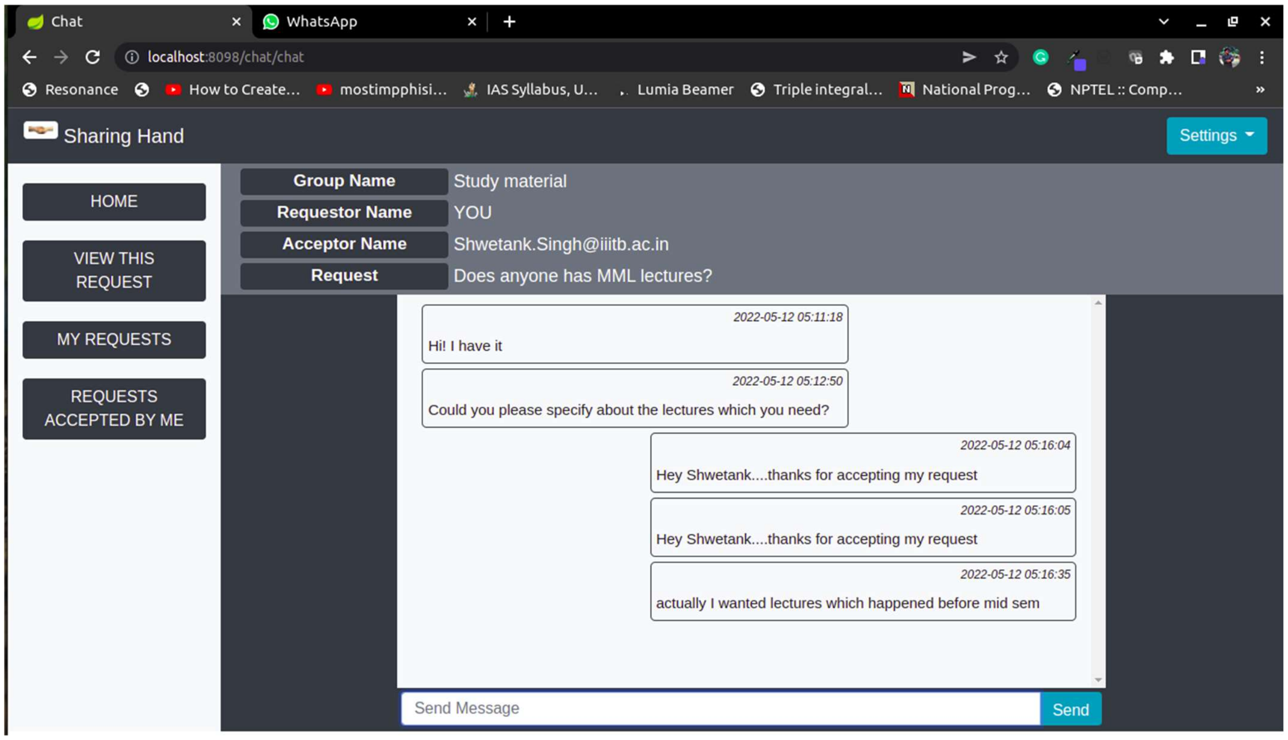
Study material

Shree.Sharan@iiitb.ac.in 2022-05-11

Does anyone has MML lectures?

Figure: Requests Accepted Page

8. Private Chat Page



Group Name	Study material
Requestor Name	YOU
Acceptor Name	Shwetank.Singh@iiitb.ac.in
Request	Does anyone has MML lectures?

2022-05-12 05:11:18
Hi! I have it

2022-05-12 05:12:50
Could you please specify about the lectures which you need?

2022-05-12 05:16:04
Hey Shwetank....thanks for accepting my request

2022-05-12 05:16:05
Hey Shwetank....thanks for accepting my request

2022-05-12 05:16:35
actually I wanted lectures which happened before mid sem

Figure: Private Chat Page

PROJECT ARCHITECTURE

Project Architecture is summarized as follows:

Overview

Framework	Java Spring Boot Framework
Database	MySQL (db4free.net)
Deployment Server	Tomcat
Repository Hosting	Github
Artifact	JAR file
Server	Docker Containers
Github Repo	https://github.com/shwetanknaveen/SharingHands

Automation Tools

SCM	Git
Build	Maven integrated with Jenkins
Testing	Junit, Mockito
Configuration	Ansible
Deployment	Jenkins, Docker
Monitoring	ELK Stack (Elastic Search, Logstash, Kibana)

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/shwetanknaveen/SharingHands ’.

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:

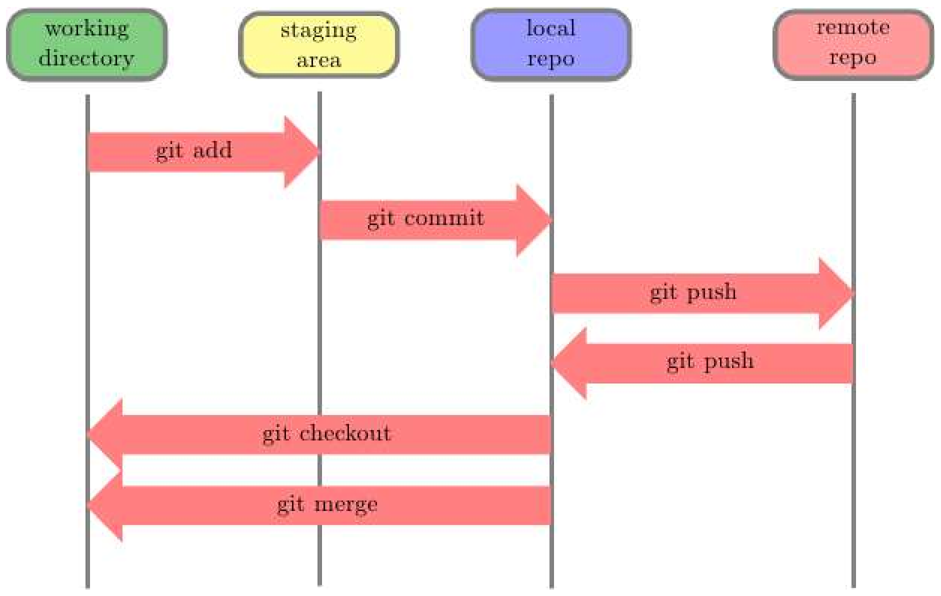
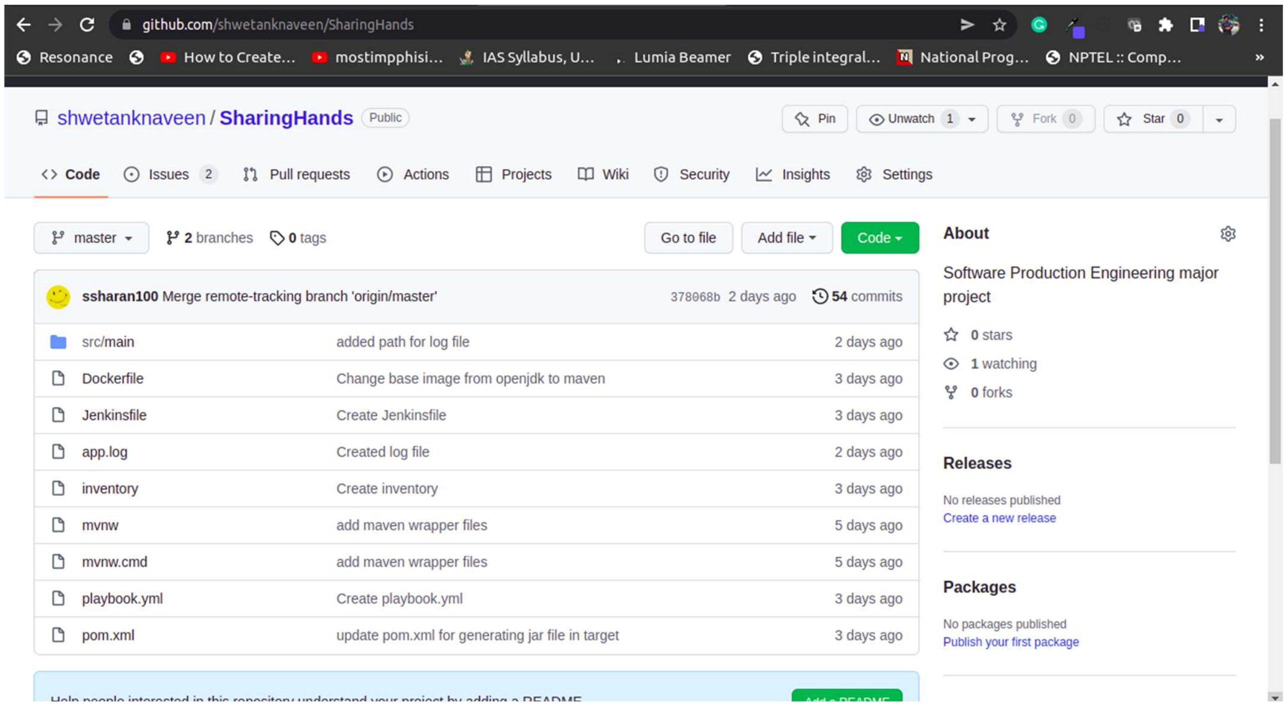


Figure: Git Workflow

Following screenshot shows the github home page for the application:



github.com/shwetanknaveen/SharingHands

shwetanknaveen / SharingHands Public

master · 2 branches · 0 tags

ssharan100 Merge remote-tracking branch 'origin/master' 378068b 2 days ago 54 commits

src/main	added path for log file	2 days ago
Dockerfile	Change base image from openjdk to maven	3 days ago
Jenkinsfile	Create Jenkinsfile	3 days ago
app.log	Created log file	2 days ago
inventory	Create inventory	3 days ago
mvnw	add maven wrapper files	5 days ago
mvnw.cmd	add maven wrapper files	5 days ago
playbook.yml	Create playbook.yml	3 days ago
pom.xml	update pom.xml for generating jar file in target	3 days ago

About

Software Production Engineering major project

0 stars
1 watching
0 forks

Releases

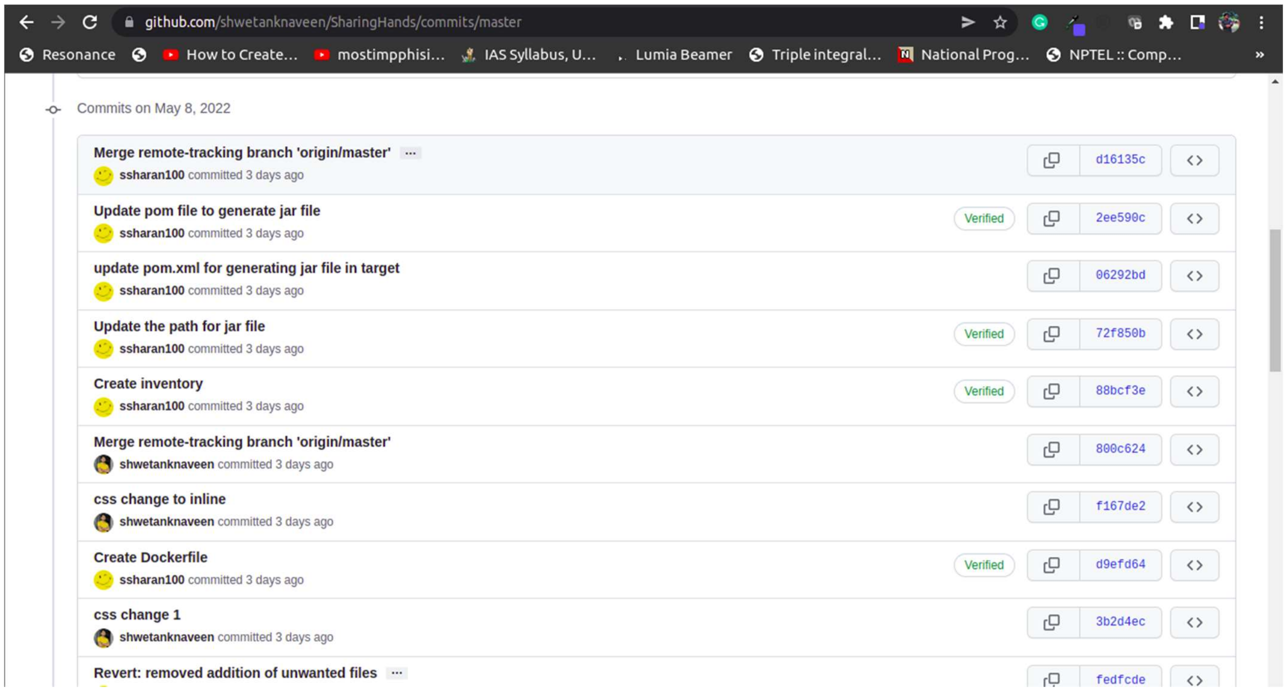
No releases published
Create a new release

Packages

No packages published
Publish your first package

Figure: Github Home SharingHands

Following screenshot shows some of the commits:



github.com/shwetanknaveen/SharingHands/commits/master

Commits on May 8, 2022

Merge remote-tracking branch 'origin/master' — ssharan100 committed 3 days ago		d16135c
Update pom file to generate jar file — ssharan100 committed 3 days ago	Verified	2ee590c
update pom.xml for generating jar file in target — ssharan100 committed 3 days ago		06292bd
Update the path for jar file — ssharan100 committed 3 days ago	Verified	72f850b
Create inventory — ssharan100 committed 3 days ago	Verified	88bcf3e
Merge remote-tracking branch 'origin/master' — shwetanknaveen committed 3 days ago		800c624
css change to inline — shwetanknaveen committed 3 days ago		f167de2
Create Dockerfile — ssharan100 committed 3 days ago	Verified	d9efd64
css change 1 — shwetanknaveen committed 3 days ago		3b2d4ec
Revert: removed addition of unwanted files		fedfcde

Figure: Commit History

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.

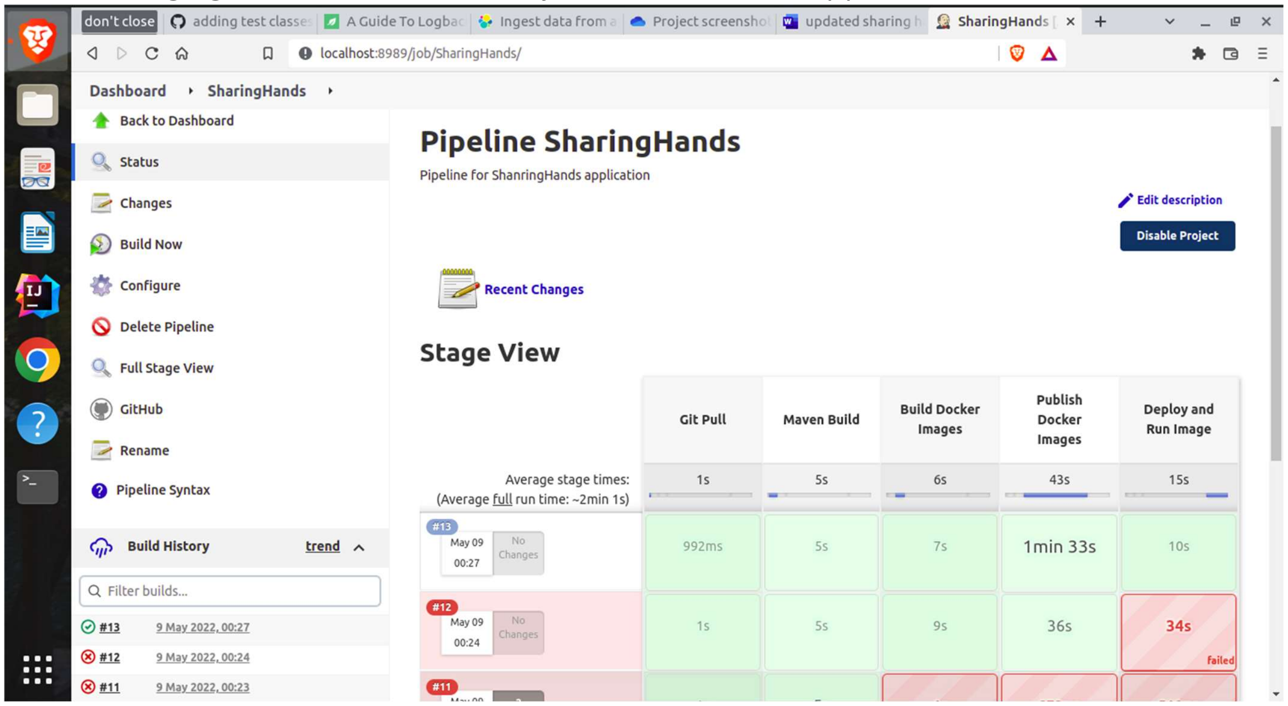
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
     <modelVersion>4.0.0</modelVersion>
     <parent>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-parent</artifactId>
            <version>2.1.3.RELEASE</version>
            <relativePath/> <!-- lookup parent from repository -->
     </parent>
     <groupId>com.sharinghand</groupId>
     <artifactId>SharingHand</artifactId>
     <version>0.0.1-SNAPSHOT</version>

     <name>SharingHand</name>
     <description>Collaborative Sharing Platform - SPE Project</description>

     <properties>
            <java.version>1.8</java.version>
     </properties>

     <dependencies>
            <dependency>
                   <groupId>org.springframework.boot</groupId>
                   <artifactId>spring-boot-starter-data-jpa</artifactId>
            </dependency>
            <dependency>
                   <groupId>org.springframework.boot</groupId>
                   <artifactId>spring-boot-starter-web</artifactId>
            </dependency>
            <dependency>
                   <groupId>org.springframework.boot</groupId>
                   <artifactId>spring-boot-starter-tomcat</artifactId>
                   <scope>provided</scope>
             </dependency>
            <dependency>
                   <groupId>mysql</groupId>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.apache.tomcat.embed</groupId>
            <artifactId>tomcat-embed-jasper</artifactId>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-websocket</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

Code: pom.xml

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

Following figure shows the Jenkins job that builds the application:



Pipeline SharingHands

Pipeline for ShanringHands application

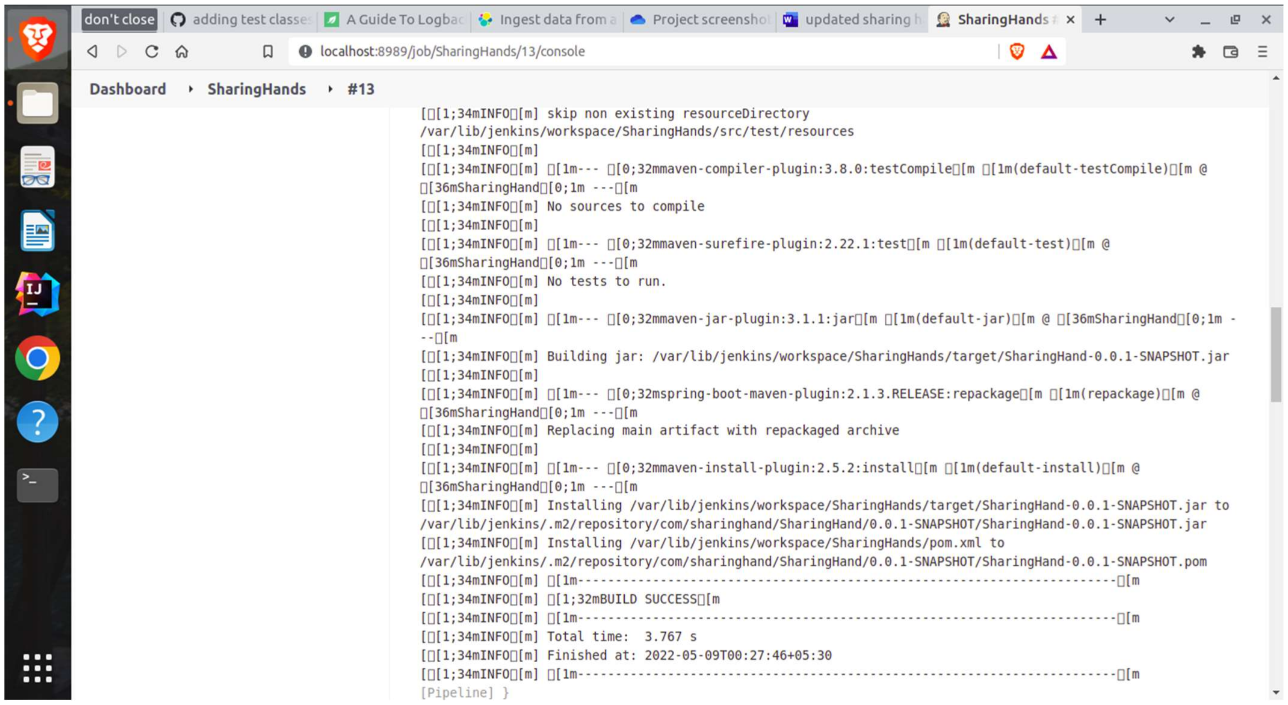
Stage View

	Git Pull	Maven Build	Build Docker Images	Publish Docker Images	Deploy and Run Image
Average stage times: (Average full run time: ~2min 1s)	1s	5s	6s	43s	15s
#13 May 09 00:27 No Changes	992ms	5s	7s	1min 33s	10s
#12 May 09 00:24 No Changes	1s	5s	9s	36s	34s failed

#13	9 May 2022, 00:27
#12	9 May 2022, 00:24
#11	9 May 2022, 00:23

Figure: Maven Application Build Job

Following figure shows the successful build step completed using Maven in Jenkins:



```
[□[1;34mINFO□[m] skip non existing resourceDirectory
/var/lib/jenkins/workspace/SharingHands/src/test/resources
[□[1;34mINFO□[m]
[□[1;34mINFO□[m] □[1m--- □[0;32mmaven-compiler-plugin:3.8.0:testCompile□[m □[1m(default-testCompile)□[m @
□[36mSharingHand□[0;1m ---□[m
[□[1;34mINFO□[m] No sources to compile
[□[1;34mINFO□[m]
[□[1;34mINFO□[m] □[1m--- □[0;32mmaven-surefire-plugin:2.22.1:test□[m □[1m(default-test)□[m @
□[36mSharingHand□[0;1m ---□[m
[□[1;34mINFO□[m] No tests to run.
[□[1;34mINFO□[m]
[□[1;34mINFO□[m] □[1m--- □[0;32mmaven-jar-plugin:3.1.1:jar□[m □[1m(default-jar)□[m @ □[36mSharingHand□[0;1m -
--□[m
[□[1;34mINFO□[m] Building jar: /var/lib/jenkins/workspace/SharingHands/target/SharingHand-0.0.1-SNAPSHOT.jar
[□[1;34mINFO□[m]
[□[1;34mINFO□[m] □[1m--- □[0;32mspring-boot-maven-plugin:2.1.3.RELEASE:repackage□[m □[1m(repackage)□[m @
□[36mSharingHand□[0;1m ---□[m
[□[1;34mINFO□[m] Replacing main artifact with repackaged archive
[□[1;34mINFO□[m]
[□[1;34mINFO□[m] □[1m--- □[0;32mmaven-install-plugin:2.5.2:install□[m □[1m(default-install)□[m @
□[36mSharingHand□[0;1m ---□[m
[□[1;34mINFO□[m] Installing /var/lib/jenkins/workspace/SharingHands/target/SharingHand-0.0.1-SNAPSHOT.jar to
/var/lib/jenkins/.m2/repository/com/sharinghand/SharingHand/0.0.1-SNAPSHOT/SharingHand-0.0.1-SNAPSHOT.jar
[□[1;34mINFO□[m] Installing /var/lib/jenkins/workspace/SharingHands/pom.xml to
/var/lib/jenkins/.m2/repository/com/sharinghand/SharingHand/0.0.1-SNAPSHOT/SharingHand-0.0.1-SNAPSHOT.pom
[□[1;34mINFO□[m] □[1m------------------------------------------------------------------------□[m
[□[1;34mINFO□[m] □[1;32mBUILD SUCCESS□[m
[□[1;34mINFO□[m] □[1m------------------------------------------------------------------------□[m
[□[1;34mINFO□[m] Total time:  3.767 s
[□[1;34mINFO□[m] Finished at: 2022-05-09T00:27:46+05:30
[□[1;34mINFO□[m] □[1m------------------------------------------------------------------------□[m
[Pipeline] }
```

Figure: Maven Successful Build

3. TESTING - JUnit and Mockito

Before building the application we make sure that everything is tried and tested to cause minimal disruption to the end user. To implement this we use JUnit and Mockito.

JUnit is a unit testing framework for Java programming language. Following code snippet demonstrates how the application is tested using JUnit:

```
package com.sharinghand.home;

import com.sharinghand.chat.ChatController;
import com.sharinghand.chat.ChatmapController;
import com.sharinghand.group.GroupstableController;
import com.sharinghand.login.UsertableController;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

import static org.assertj.core.api.Assertions.assertThat;


@SpringBootTest
@RunWith(SpringRunner.class)
public class JUnitControllerInitializationTest {
  @Autowired
  UsertableController usertableController;

  @Test
  public void testUsertableControllerInitialization()   {
        assertThat(usertableController).isNotNull();
  }

  @Autowired
  GroupstableController groupstableController;

  @Test
  public void testGroupstableControllerInitialization() {
        assertThat(groupstableController).isNotNull();
  }

  @Autowired
  ChatController chatController;

  @Test
  public void testchatControllerInitialization()  {
        assertThat(chatController).isNotNull();
  }

  @Autowired
  ChatmapController chatmapController;

  @Test
  public void testChatmapControllerInitialization()  {
        assertThat(chatmapController).isNotNull();
  }
}
```

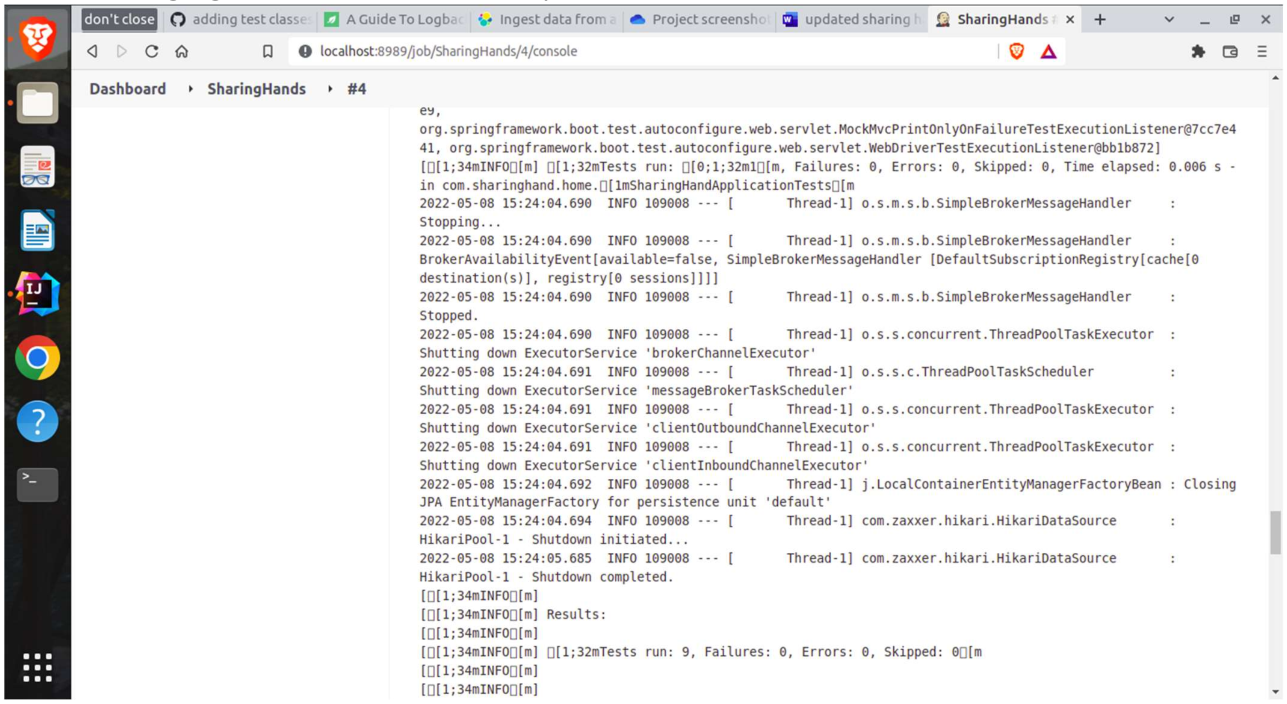
Code: Example JUnit Tester

Mockito is a mocking framework, JAVA-based library that is used for effective unit testing of JAVA applications. Mockito is used to mock interfaces so that a dummy functionality can be added to a mock interface that can be used in unit testing. Following code snippet demonstrates how Mockito is used for testing:

```
package com.sharinghand.home;

import static org.hamcrest.CoreMatchers.instanceOf;
import static org.junit.Assert.assertThat;

import com.sharinghand.group.GroupstableController;
import com.sharinghand.login.UsertableService;
import com.sharinghand.users.UsersSubtableService;
import org.junit.Before;
import org.junit.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;
import org.springframework.web.servlet.ModelAndView;

public class MockitoGroupstableControllerTest {
  @InjectMocks
  GroupstableController groupstableController = new GroupstableController();

  @Mock
  UsertableService userService;

  @Mock
  UsersSubtableService userSubService;

  @Before
    public void init() {
        MockitoAnnotations.initMocks(this);
    }

  @Test
  public void createGroupTester()     {
         assertThat(groupstableController.createGroup(), instanceOf(ModelAndView.class));
  }

  @Test
  public void insertMembersTester()    {
         assertThat(groupstableController.insertMembers(), instanceOf(ModelAndView.class));
  }
}
```

Code: Example Mockito Tester

Following figure shows the test output:



```
ey,
org.springframework.boot.test.autoconfigure.web.servlet.MockMvcPrintOnlyOnFailureTestExecutionListener@7cc7e4
41, org.springframework.boot.test.autoconfigure.web.servlet.WebDriverTestExecutionListener@bb1b872]
[[1;34mINFO[m] [1;32mTests run: [0;1;32m1[m, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.006 s -
in com.sharinghand.home.[1mSharingHandApplicationTests[m
2022-05-08 15:24:04.690  INFO 109008 --- [       Thread-1] o.s.m.s.b.SimpleBrokerMessageHandler     :
Stopping...
2022-05-08 15:24:04.690  INFO 109008 --- [       Thread-1] o.s.m.s.b.SimpleBrokerMessageHandler     :
BrokerAvailabilityEvent[available=false, SimpleBrokerMessageHandler [DefaultSubscriptionRegistry[cache[0
destination(s)], registry[0 sessions]]]]
2022-05-08 15:24:04.690  INFO 109008 --- [       Thread-1] o.s.m.s.b.SimpleBrokerMessageHandler     :
Stopped.
2022-05-08 15:24:04.690  INFO 109008 --- [       Thread-1] o.s.s.concurrent.ThreadPoolTaskExecutor  :
Shutting down ExecutorService 'brokerChannelExecutor'
2022-05-08 15:24:04.691  INFO 109008 --- [       Thread-1] o.s.s.c.ThreadPoolTaskScheduler          :
Shutting down ExecutorService 'messageBrokerTaskScheduler'
2022-05-08 15:24:04.691  INFO 109008 --- [       Thread-1] o.s.s.concurrent.ThreadPoolTaskExecutor  :
Shutting down ExecutorService 'clientOutboundChannelExecutor'
2022-05-08 15:24:04.691  INFO 109008 --- [       Thread-1] o.s.s.concurrent.ThreadPoolTaskExecutor  :
Shutting down ExecutorService 'clientInboundChannelExecutor'
2022-05-08 15:24:04.692  INFO 109008 --- [       Thread-1] j.LocalContainerEntityManagerFactoryBean : Closing
JPA EntityManagerFactory for persistence unit 'default'
2022-05-08 15:24:04.694  INFO 109008 --- [       Thread-1] com.zaxxer.hikari.HikariDataSource       :
HikariPool-1 - Shutdown initiated...
2022-05-08 15:24:05.685  INFO 109008 --- [       Thread-1] com.zaxxer.hikari.HikariDataSource       :
HikariPool-1 - Shutdown completed.
[[1;34mINFO[m]
[[1;34mINFO[m] Results:
[[1;34mINFO[m]
[[1;34mINFO[m] [1;32mTests run: 9, Failures: 0, Errors: 0, Skipped: 0[m
[[1;34mINFO[m]
[[1;34mINFO[m]
```

Figure: Tests successfully completed

4. CONFIGURATION MANAGEMENT

We used ansible for Configuration Management. For application deployment we need java 8 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.

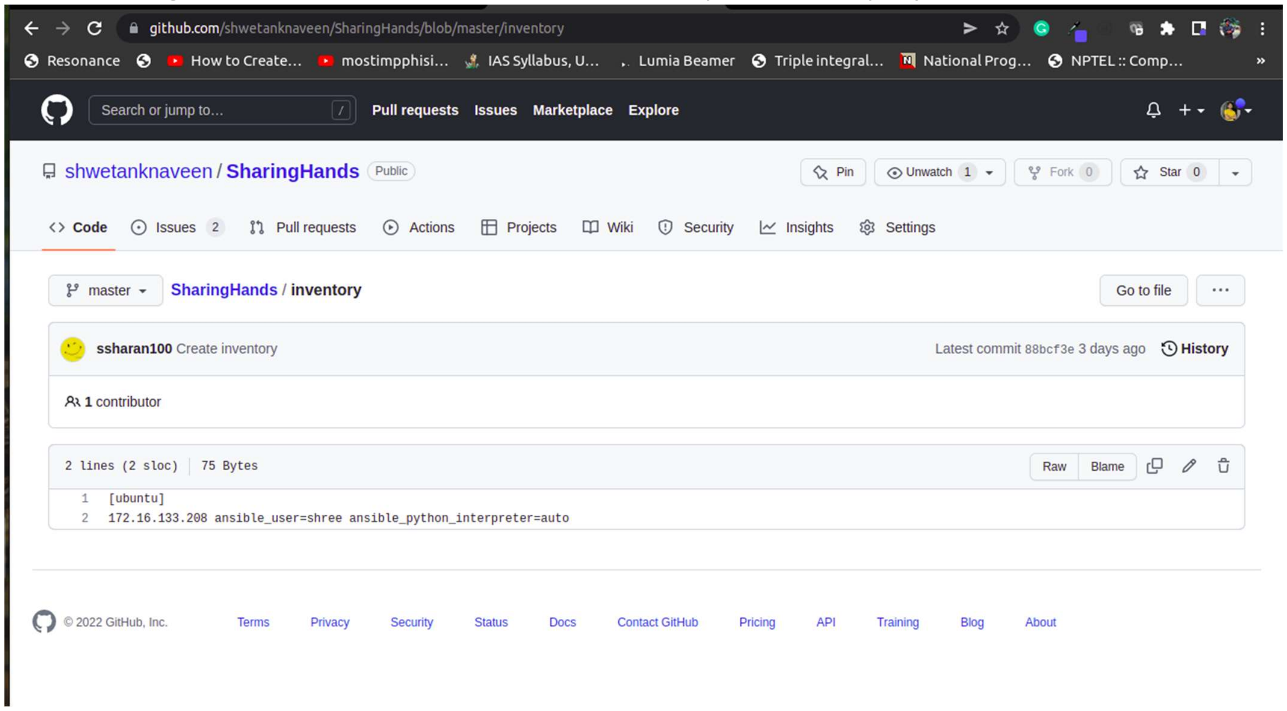
We used ansible "java" for SharingHand. This playbook was taken from what we used in mini-project, and we modified it according to our requirement. Following codesnippet shows the changes made to the default playbook:

```
---
- name: Pull docker image of SharingHands application
  hosts: all
  tasks:
    - name: Start docker service
      service:
        name: docker
        state: started

    - name: pull docker image
      shell: docker pull 1651078/sharing_hands:latest

    - name: running container
      shell: docker run -d --name sharing_hands_app -p 9995:8098
1651078/sharing_hands
```

Code: Playbook.yml

We used inventory file in our project:

Following screenshot shows the used inventory file in the project:


github.com/shwetanknaveen/SharingHands/blob/master/inventory
shwetanknaveen / SharingHands Public
Code Issues 2 Pull requests Actions Projects Wiki Security Insights Settings
master SharingHands / inventory
ssharan100 Create inventory Latest commit 88bcf3e 3 days ago History
1 contributor
2 lines (2 sloc) 75 Bytes

```
[ubuntu]
172.16.133.208 ansible_user=shree ansible_python_interpreter=auto
```

Figure: Inventory file used

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 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 throughdelivery, monitoring, and troubleshooting, Ansible provides a pipeline for continuous deployment that you can use to achieve more and make better decisions.

5. 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 pulled docker image on the deployment node.

6. MONITORING

We used Elasticsearch with Logstash and Kibana for monitoring MySQL database. MySQL database is where the data is safely persisted to disk. The database tends to be the "black box" for most operations people: queries go in, results come out. So Monitoring, as a whole, is aimed to gives clearer and easier understanding of complex query to make predictions. Predictions allow us to provision our environment correctly to ensure the best performance for our customer use case.

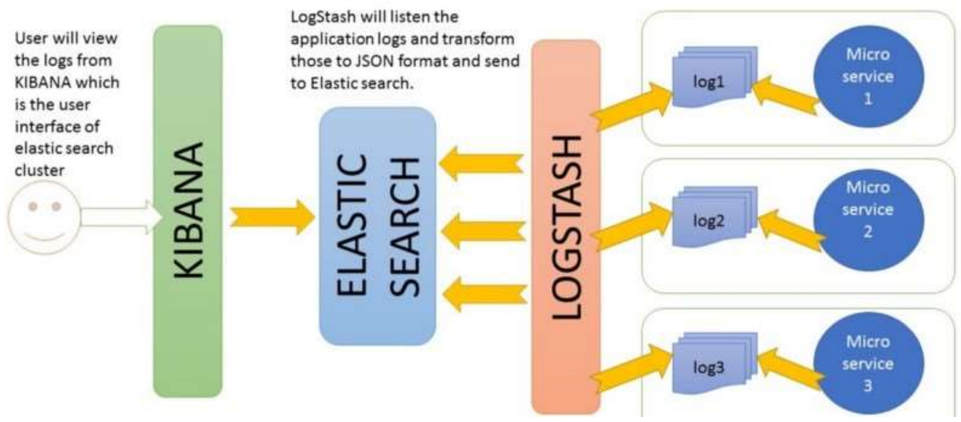

User will view the logs from KIBANA which is the user interface of elastic search cluster
LogStash will listen the application logs and transform those to JSON format and send to Elastic search.
KIBANA
ELASTIC SEARCH
LOGSTASH
log1
log2
log3
Micro service 1
Micro service 2
Micro service 3

Figure: ELK interaction with different application based on log file

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 visualize our data that stored in Elasticsearch indices in a variety of charts, tables, and graphs. It's a simple, browser-based interface 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 code snippet from configuration file demonstrates jdbc mysql connection with Elasticsearch.

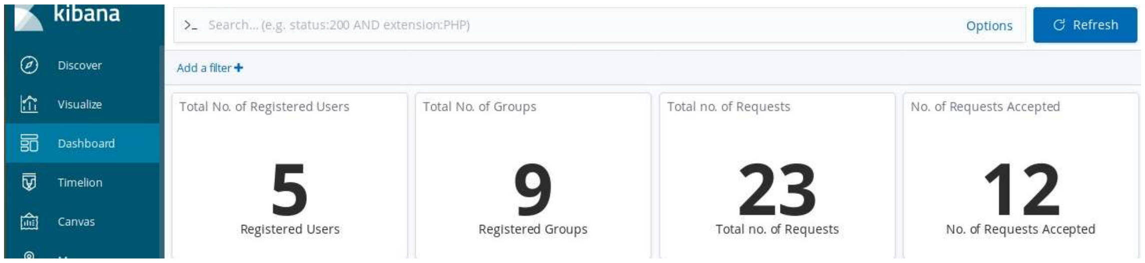
```
input {
      jdbc {
             jdbc_driver_library => "/home/shwetank/Downloads/mysql-connector-java-
8.0.15.jar"
             jdbc_driver_class => "com.mysql.cj.jdbc.Driver"
             jdbc_connection_string => "jdbc:mysql://db4free.net:3306/sharinghand"
             jdbc_user => "ss2022project"
             jdbc_password => "password"
             statement => "SELECT *, DATE_FORMAT(date, '%h %p') FROM chattable"
             schedule => "*/1 * * * *"
             type => "chattrafficbytime"
      }
}
output {
      elasticsearch {
             hosts => [ "https://my-dep-a6b7a4.es.us-central1.gcp.cloud.es.io:9243" ]
             document_id => "%{msgid}"
             index => "%{type}"
      }
      stdout { codec => dots }
}
```

Code: .conf file

We created few metrics, charts & graphs which shows

1. Total no of registered user
2. Total no of registered groups
3. Total no of request raised
4. Total no of request accepted
5. Top 5 group based on request raised
6. Top 5 group based on members count
7. Chat traffic per day
8. Chat traffic during day

Following screenshot shows the dashboards created in Kibana:



Total No. of Registered Users
5
Registered Users

Total No. of Groups
9
Registered Groups

Total no. of Requests
23
Total no. of Requests

No. of Requests Accepted
12
No. of Requests Accepted

Figure: Metrics for Number of Users, Groups and Requests

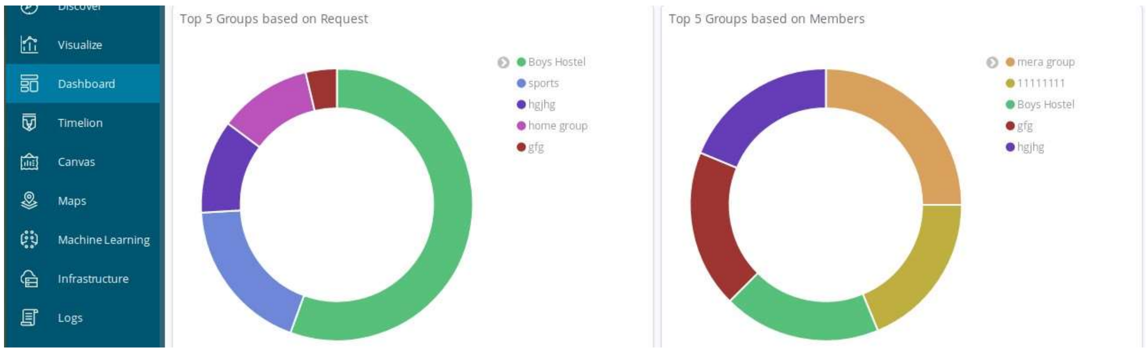


Figure: Top 5 groups based on number of requests and users

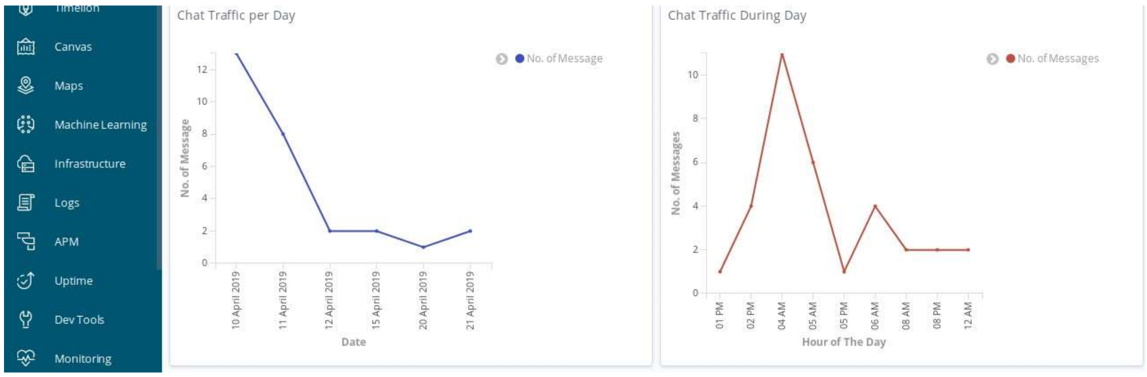


Figure: Reports for website traffic

FUTURE WORK

In terms of application development

This application has tremendous potential for new and interesting features.

Some of the features that we plan to incorporate in the near future are:

1. Display notifications
    a. When someone posts a new request
    b. When someone accepts my request
    c. For a new chat message
2. Add the functionality to track the items borrowed and lended
3. Provide users with credibility and karma points
4. Provide the group admin with management functionality like
    a. Adding new members to the group
    b. Kick rogue/spamming members from the group
    c. Change the group name
5. Provide functionality to the members to crowdfund items

In terms of automation

Right now our application is packaged as a .jar file and even for a small change we need to rebuild and deploy the .jar file.

This will work fine as the application is currently small. As the application will grow we would like to deploy small changes by modifying the already deployed application rather than redeploying the complete application.

We will have two separate pipelines for

1. Complete application redeployment
2. Releasing small changes to the application

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-features-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/getting-started.html