INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

BANGALORE

CS 816- SOFTWARE PRODUCTION ENGINEERING

BOOKFLIX

Team Members:

Anoushka Chouksey MT2022165

Satvika Shrivastava MT2022105

Under Guidance of:

Prof. B. Thangaraju

Teaching Assistant:

Suchi Bhargav

1. Abstract

BookFlix is a web application that can keep track of a user's reading list. Using this application, a user can login and add books to their read, currently reading and TBR shelves. The user can also add a starting and finished date to their completed book and add a rating.

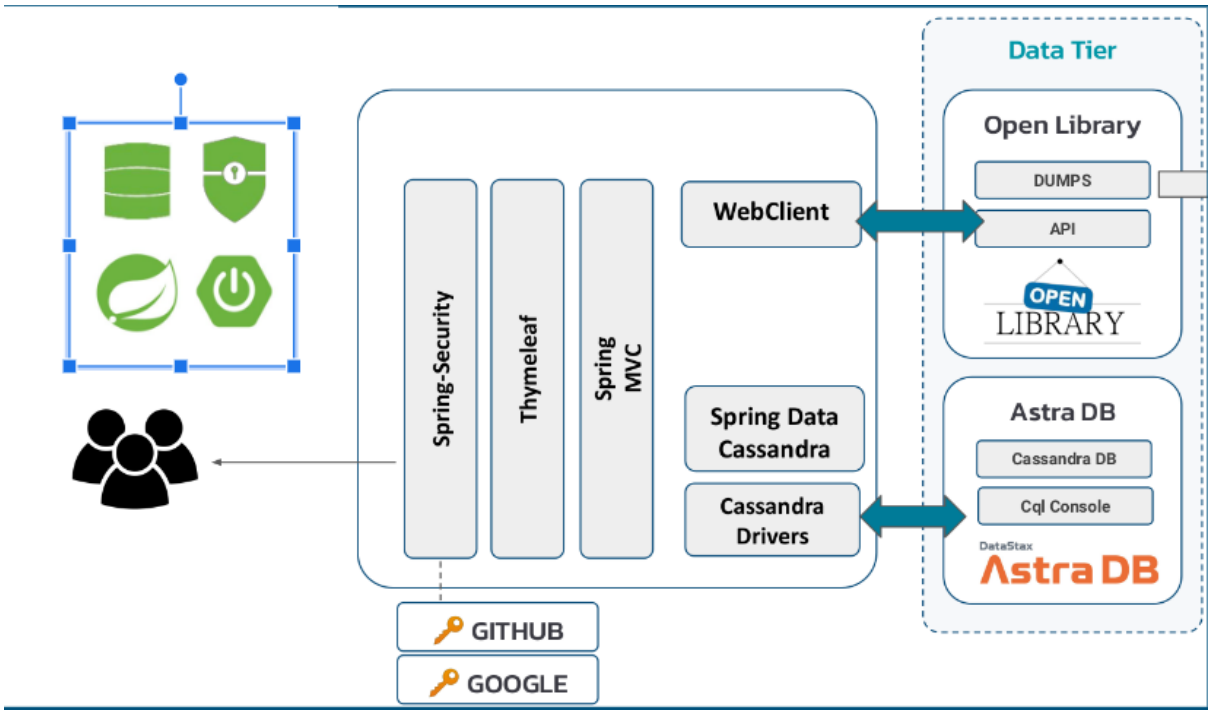
The architecture of our project demands three layers.

- Front end
- Middle layer
- Back end

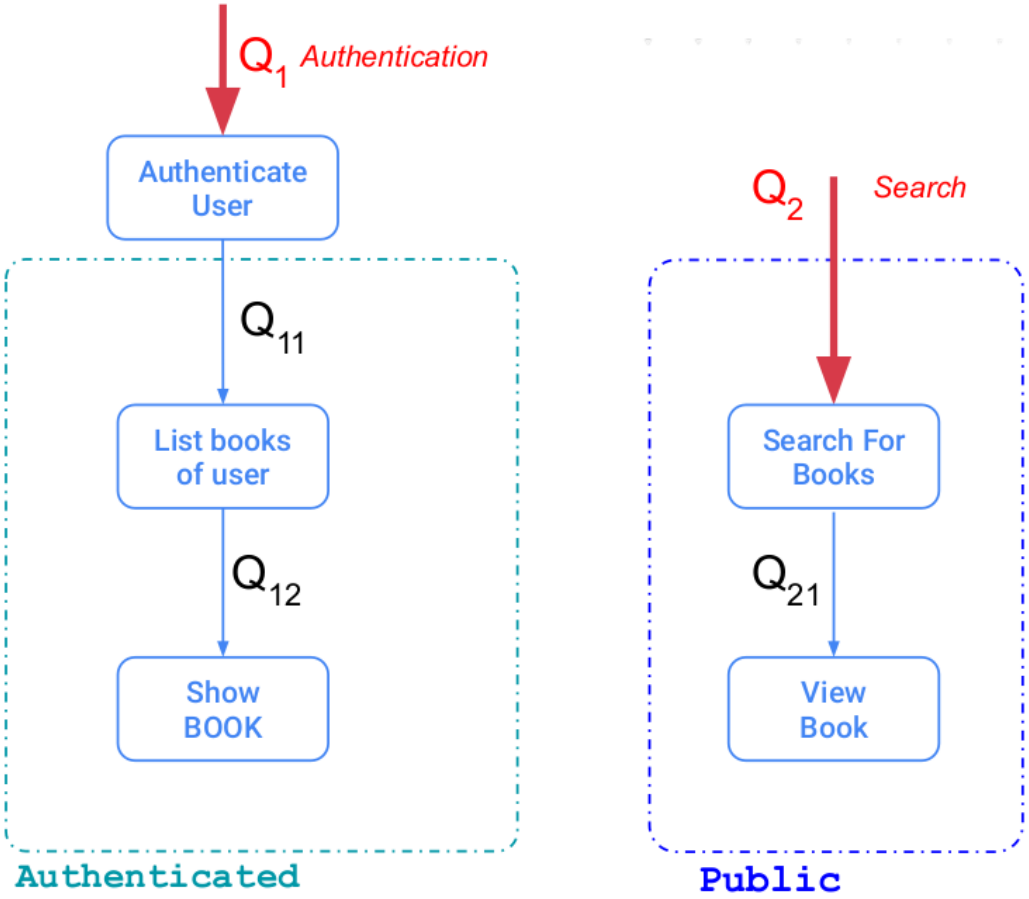
The front end of the project is handled by HTML and Thymeleaf framework. The middle layer is built on SpringBoot Framework and communicates with Cassandra database to show the content on the front end via Rest API.

2. Introduction

2.1 Overview



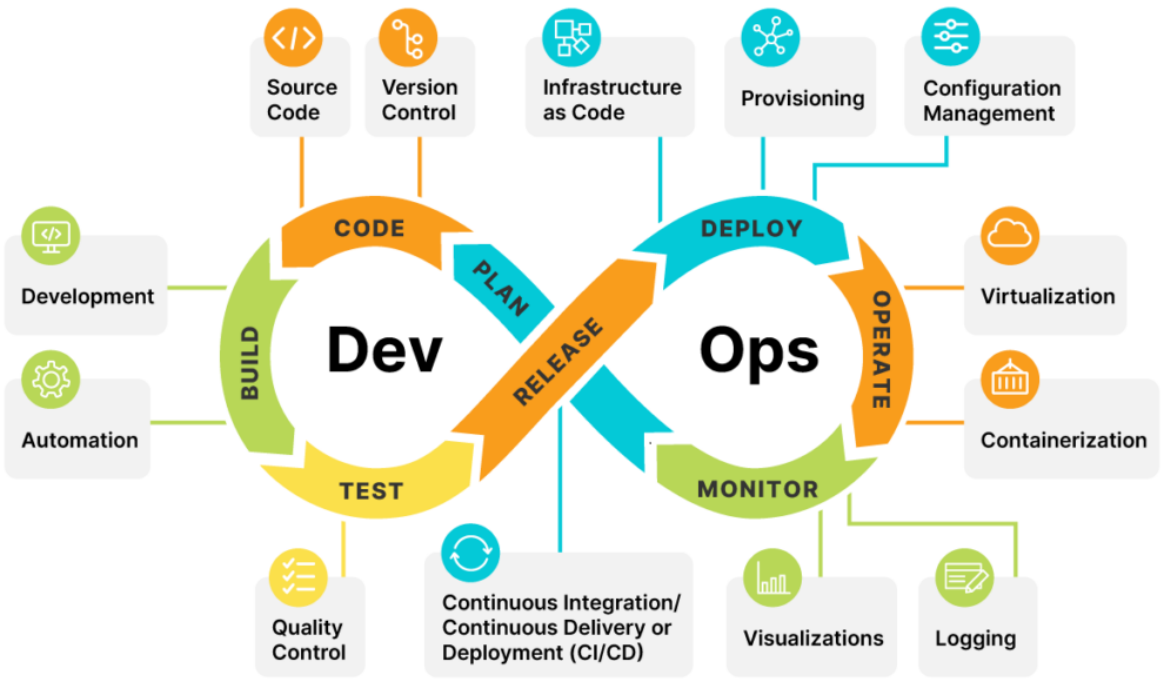
2.2 Features


Q1 Authentication
Authenticate User
Q11
List books of user
Q12
Show BOOK
Authenticated
Q2 Search
Search For Books
Q21
View Book
Public

2.3 Why DevOps

DevOps is a set of practices that automates the processes between software development and IT teams, in order that they can build, test, and release software faster and more reliably. The concept of DevOps is founded on building a culture of collaboration between teams that historically functioned in relative siloes. The promised benefits include increased trust, faster software releases, ability to solve critical issues quickly, and better manage unplanned work.

It unites agile, continuous delivery, automation, and much more, to help development and operations teams be more efficient, innovate faster, and deliver higher value to businesses and customers. Thus, DevOps is preferred as it helps in faster release time and better management of unplanned work.



3. System Configuration

3.1 Operating System

Ubuntu 20.04 LTS

3.2 CPU and RAM

4 core processor and RAM 8 GB

3.3 Language

Cassandra Query Language (CQL), HTML, CSS, Thymeleaf

Java, Spring framework

3.4 Kernel Version

Linux Machine 5.13.0-40-generic

3.5 Database

Datastax Astra DB, based on Apache Cassandra

3.6 Building Tools

Maven to build java application

3.7 DevOps Tools

- Source Control Management- GitHub
- Continuous Integration- Jenkins
- Containerization – Docker
- Continuous deployment- Ansible
- Monitoring- ELK Stack (Elastic Search, Logstash, Kibana)

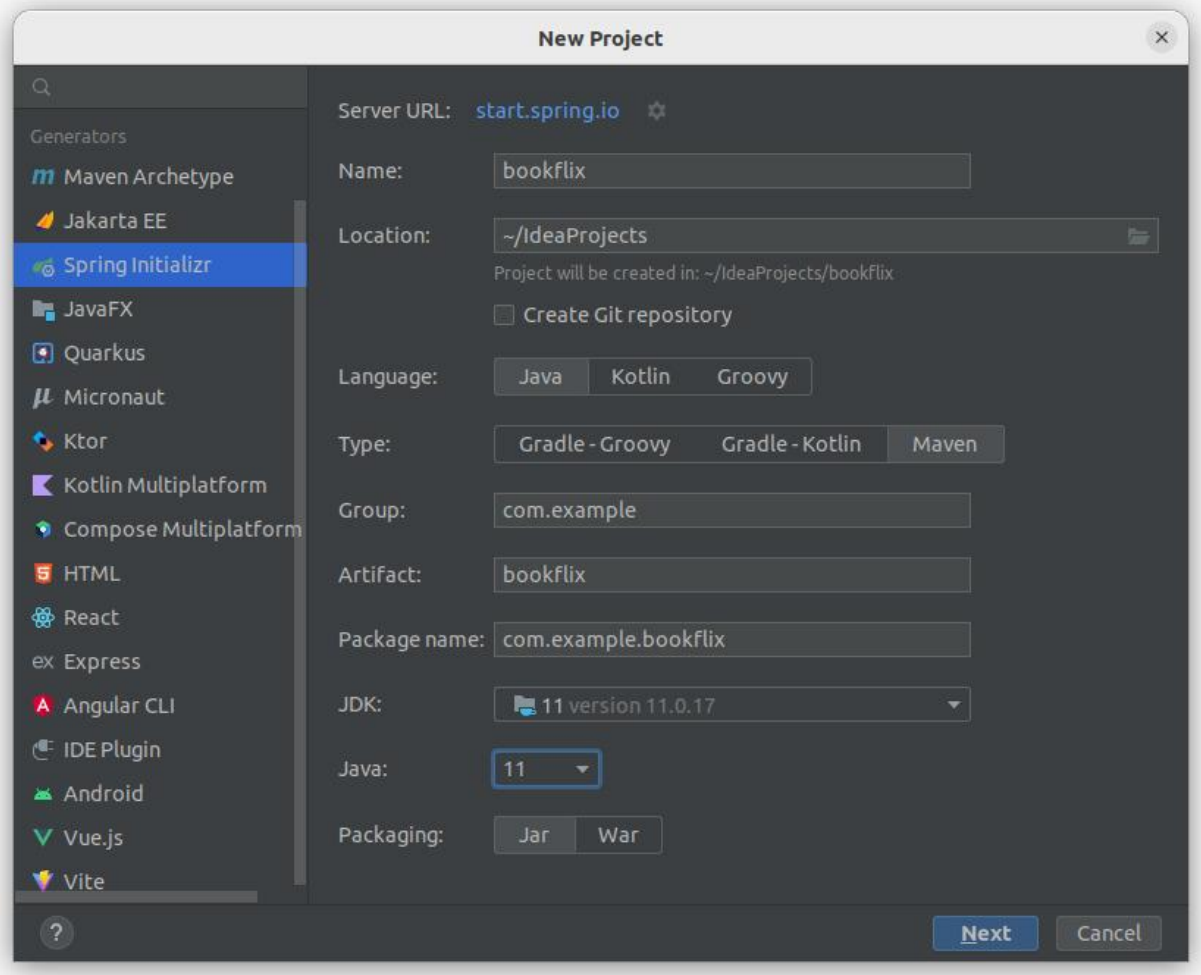
4. Software Development Life Cycle

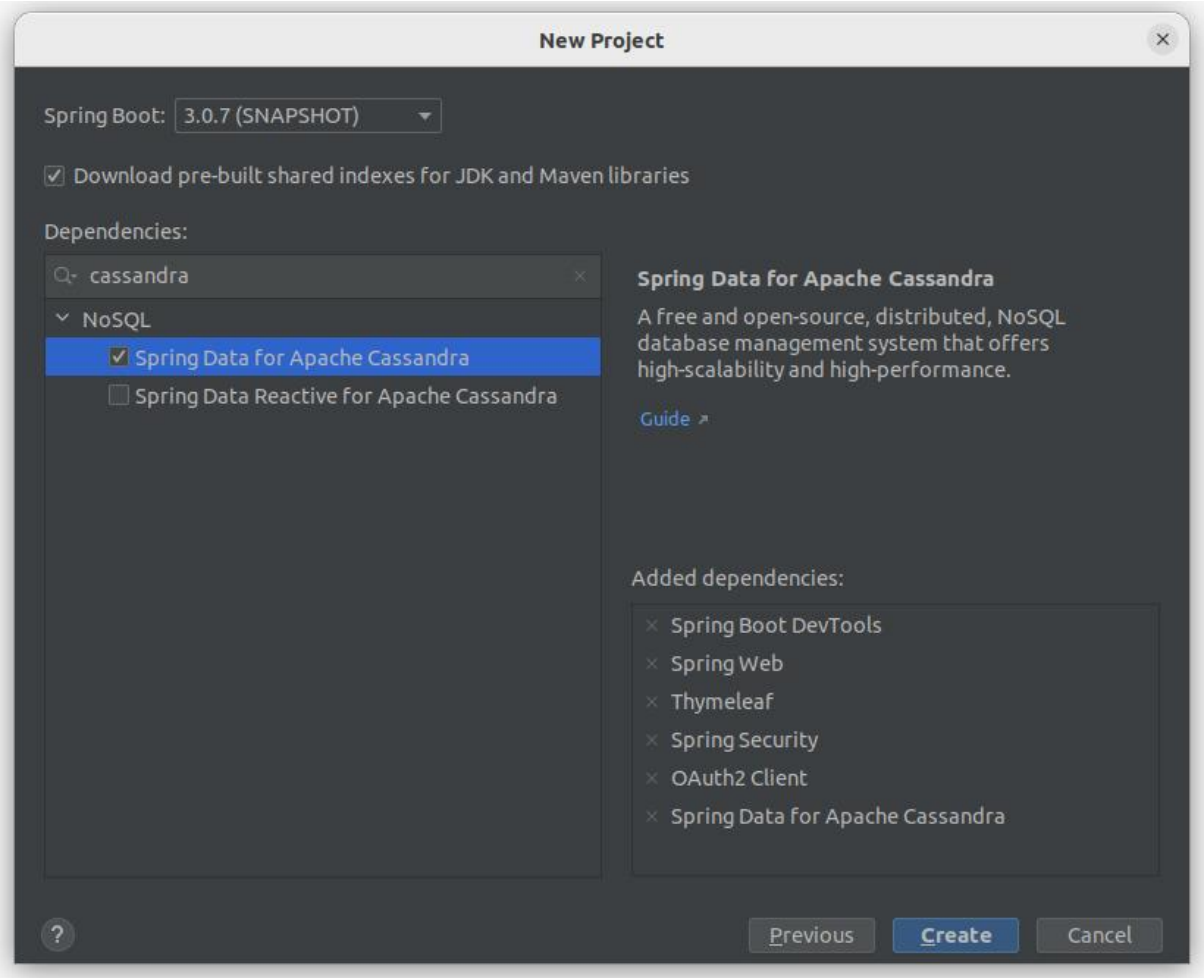
Software development life cycle is a systematic process for building software that ensures the quality and correctness of the software built. SDLC process aims to produce high-quality software that meets customer expectations. The system development should be complete in the pre-defined time frame and cost. SDLC consists of a detailed plan which explains how to plan, build, and maintain specific software. Every phase of the SDLC life cycle has its own process and deliverables that feed into the next phase.

4.1 Installation

4.1.1 SpringBoot

Start Project. Visit (https://start.spring.io/) → Set requirements → Download Project.

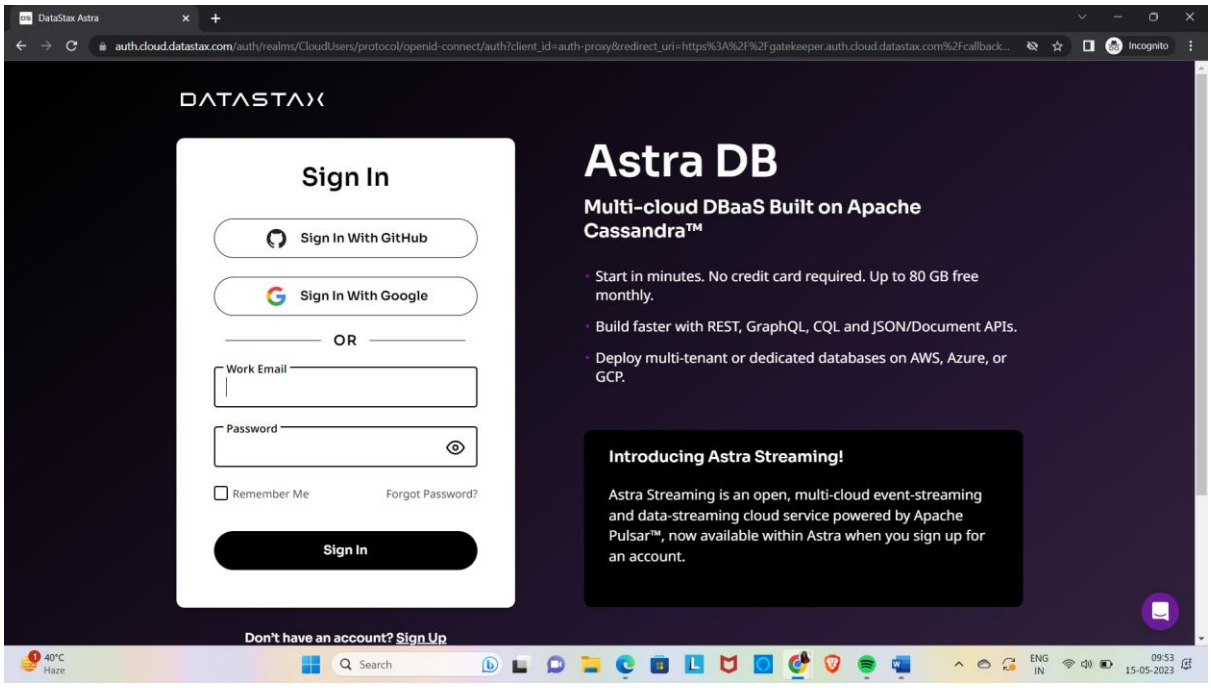



New Project
Spring Boot: 3.0.7 (SNAPSHOT)
Download pre-built shared indexes for JDK and Maven libraries
Dependencies:
cassandra
NoSQL
Spring Data for Apache Cassandra
Spring Data Reactive for Apache Cassandra
Spring Data for Apache Cassandra
A free and open-source, distributed, NoSQL database management system that offers high-scalability and high-performance.
Guide
Added dependencies:
Spring Boot DevTools
Spring Web
Thymeleaf
Spring Security
OAuth2 Client
Spring Data for Apache Cassandra
Previous
Create
Cancel

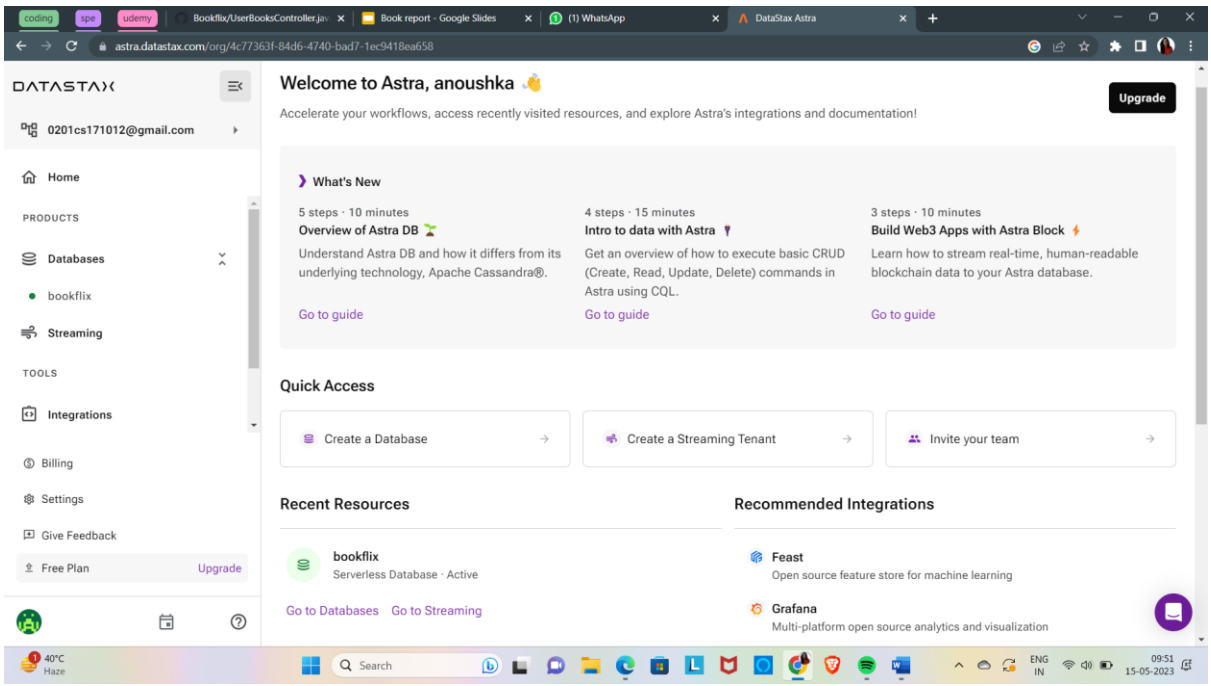
4.1.2 Cassandra

CQL query language is a NoSQL interface that is intentionally similar to SQL, providing users who are comfortable with relational databases a familiar language that ultimately lowers the barrier of entry to Apache Cassandra.

- Sign in to Astra DB



- Create a Database named 'BookFlix'



- Populate the database using data dumps from https://openlibrary.org/

Works.txt

```
1 /type/work      /works/OL10000355W      4       2020-12-07T23:42:52.481900      {"title": "Le verbe en action. grammaire contrastive des temps verbaux (fran\u00e7ais, allemand, anglais, espagnol)",
  "covers": [3140607], "key": "/works/OL10000355W", "authors": [{"type": {"key": "/type/author_role"}, "author": {"key": "/authors/OL3965376A"}}], "type": {"key": "/type/work"}, "subjects": ["Comparative
  and general Grammar", "Verb", "Tense", "French language", "German language", "English language", "Spanish language"], "latest_revision": 4, "revision": 4, "created": {"type": "/type/datetime", "value":
  "2009-12-11T01:57:19.964652"}, "last_modified": {"type": "/type/datetime", "value": "2020-12-07T23:42:52.481900"}}
2 /type/work      /works/OL10000411W      3       2010-04-28T06:54:19.472104      {"title": "Je marche sous un ciel de tra\u00eene", "created": {"type": "/type/datetime", "value":
  "2009-12-11T01:57:19.964652"}, "covers": [3140776], "last_modified": {"type": "/type/datetime", "value": "2010-04-28T06:54:19.472104"}, "latest_revision": 3, "key": "/works/OL10000411W", "authors":
  [{"type": "/type/author_role", "author": {"key": "/authors/OL3965457A"}}], "type": {"key": "/type/work"}, "revision": 3}
3 /type/work      /works/OL10000514W      3       2010-04-28T06:54:19.472104      {"title": "AMIR", "created": {"type": "/type/datetime", "value": "2009-12-11T01:57:19.964652"}, "covers": [3140971],
  "last_modified": {"type": "/type/datetime", "value": "2010-04-28T06:54:19.472104"}, "latest_revision": 3, "key": "/works/OL10000514W", "authors": [{"type": "/type/author_role", "author": {"key": "/
  authors/OL3965554A"}}], "type": {"key": "/type/work"}, "revision": 3}
4 /type/work      /works/OL10000820W      4       2020-12-07T03:18:53.261225      {"title": "Antoine Pevsner - Dans Les Collections Du Mnam", "covers": [3141623], "key": "/works/OL10000820W", "authors":
  [{"type": {"key": "/type/author_role"}, "author": {"key": "/authors/OL3965920A"}}], "type": {"key": "/type/work"}, "subjects": ["Exhibitions", "Constructivism (Art)", "Art", "Centre Georges Pompidou"],
  "latest_revision": 4, "revision": 4, "created": {"type": "/type/datetime", "value": "2009-12-11T01:57:19.964652"}, "last_modified": {"type": "/type/datetime", "value": "2020-12-07T03:18:53.261225"}}
5 /type/work      /works/OL10001517W      3       2010-04-28T06:54:19.472104      {"title": "Liberez-vous de votre pass\u00e9", "created": {"type": "/type/datetime", "value":
  "2009-12-11T01:57:29.804644"}, "covers": [3143262], "last_modified": {"type": "/type/datetime", "value": "2010-04-28T06:54:19.472104"}, "latest_revision": 3, "key": "/works/OL10001517W", "authors":
  [{"type": "/type/author_role", "author": {"key": "/authors/OL3966691A"}}], "type": {"key": "/type/work"}, "revision": 3}
6 /type/work      /works/OL10001746W      3       2010-04-28T06:54:19.472104      {"title": "R\u00e9ussir le commentaire de texte \u00e0 l'examen profressionnel de r\u00e9dacteur-chef", "created":
  {"type": "/type/datetime", "value": "2009-12-11T01:57:29.804644"}, "covers": [3143833], "last_modified": {"type": "/type/datetime", "value": "2010-04-28T06:54:19.472104"}, "latest_revision": 3, "key":
  "/works/OL10001746W", "authors": [{"type": "/type/author_role", "author": {"key": "/authors/OL3966927A"}}], "type": {"key": "/type/work"}, "revision": 3}
7 /type/work      /works/OL10001885W      3       2010-04-28T06:54:19.472104      {"title": "Sardaigne 2002", "created": {"type": "/type/datetime", "value": "2009-12-11T01:57:29.804644"}, "covers":
  [3144175], "last_modified": {"type": "/type/datetime", "value": "2010-04-28T06:54:19.472104"}, "latest_revision": 3, "key": "/works/OL10001885W", "authors": [{"type": "/type/author_role", "author":
  {"key": "/authors/OL3967107A"}}], "type": {"key": "/type/work"}, "revision": 3}
8 /type/work      /works/OL10002168W      3       2010-04-28T06:54:19.472104      {"title": "Manuel m\u00e9thodique de pr\u00e9paration. Examens et concours administratifs, concours d'entr\u00e9e en
  Institut de formation des cadres de sant\u00e9", "created": {"type": "/type/datetime", "value": "2009-12-11T01:57:38.254267"}, "covers": [3144806], "last_modified": {"type": "/type/datetime", "value":
  "2010-04-28T06:54:19.472104"}, "latest_revision": 3, "key": "/works/OL10002168W", "authors": [{"type": "/type/author_role", "author": {"key": "/authors/OL3967457A"}}], "type": {"key": "/type/work"},
  "revision": 3}
9 /type/work      /works/OL10002187W      3       2010-04-28T06:54:19.472104      {"title": "L'entr\u00e9e en IFMK dans la poche", "created": {"type": "/type/datetime", "value":
  "2009-12-11T01:57:38.254267"}, "covers": [3144820], "last_modified": {"type": "/type/datetime", "value": "2010-04-28T06:54:19.472104"}, "latest_revision": 3, "key": "/works/OL10002187W", "authors":
  [{"type": "/type/author_role", "author": {"key": "/authors/OL3967473A"}}], "type": {"key": "/type/work"}, "revision": 3}
10 /type/work     /works/OL10002236W      3       2010-04-28T06:54:19.472104      {"title": "R\u00e9ussir le DPAP 2002", "created": {"type": "/type/datetime", "value": "2009-12-11T01:57:38.254267"},
  "covers": [3144911], "last_modified": {"type": "/type/datetime", "value": "2010-04-28T06:54:19.472104"}, "latest_revision": 3, "key": "/works/OL10002236W", "authors": [{"type": "/type/author_role",
  "author": {"key": "/authors/OL3967514A"}}], "type": {"key": "/type/work"}, "revision": 3}
11 /type/work     /works/OL10002277W      3       2010-04-28T06:54:19.472104      {"title": "Des lc\u00f4nes pour prier", "created": {"type": "/type/datetime", "value": "2009-12-11T01:57:38.254267"},
  "covers": [3145019], "last_modified": {"type": "/type/datetime", "value": "2010-04-28T06:54:19.472104"}, "latest_revision": 3, "key": "/works/OL10002277W", "authors": [{"type": "/type/author_role",
  "author": {"key": "/authors/OL3967591A"}}], "type": {"key": "/type/work"}, "revision": 3}
12 /type/work     /works/OL10002825W      4       2014-05-24T01:57:52.484771      {"title": "Fragmentarium", "created": {"type": "/type/datetime", "value": "2009-12-11T01:57:38.254267"}, "covers":
  [3146154], "last_modified": {"type": "/type/datetime", "value": "2014-05-24T01:57:52.484771"}, "latest_revision": 4, "key": "/works/OL10002825W", "authors": [{"type": {"key": "/type/author_role"},
  "author": {"key": "/authors/OL4954686A"}}], "type": {"key": "/type/work"}, "revision": 4}
13 /type/work     /works/OL10002844W      1       2009-12-11T01:57:38.254267      {"title": "L'Annuel Des Arts 1998", "created": {"type": "/type/datetime", "value": "2009-12-11T01:57:38.254267"},
  "last_modified": {"type": "/type/datetime", "value": "2009-12-11T01:57:38.254267"}, "latest_revision": 1, "key": "/works/OL10002844W", "authors": [{"type": "/type/author_role", "author": {"key": "/
  authors/OL3968285A"}}], "type": {"key": "/type/work"}, "revision": 1}
14 /type/work     /works/OL10003095W      3       2010-04-28T06:54:19.472104      {"title": "Th\u00e9s, infusions, sant\u00e9", "created": {"type": "/type/datetime", "value":
  "2009-12-11T01:57:52.839053"}, "covers": [3146455], "last_modified": {"type": "/type/datetime", "value": "2010-04-28T06:54:19.472104"}, "latest_revision": 3, "key": "/works/OL10003095W", "authors":
  [{"type": "/type/author_role", "author": {"key": "/authors/OL3968483A"}}], "type": {"key": "/type/work"}, "revision": 3}
15 /type/work     /works/OL10003205W      3       2010-04-28T06:54:19.472104      {"title": "Manuel de pilotage de montgolfi\u00e8res", "created": {"type": "/type/datetime", "value":
  "2009-12-11T01:57:52.839053"}, "covers": [3146614], "last_modified": {"type": "/type/datetime", "value": "2010-04-28T06:54:19.472104"}, "latest_revision": 3, "key": "/works/OL10003205W", "authors":
  [{"type": "/type/author_role", "author": {"key": "/authors/OL3968648A"}}], "type": {"key": "/type/work"}, "revision": 3}
16 /type/work     /works/OL10003382W      3       2010-04-28T06:54:19.472104      {"title": "Le Tour du monde en famille", "created": {"type": "/type/datetime", "value": "2009-12-11T01:57:52.839053"},
  "covers": [3146852], "last_modified": {"type": "/type/datetime", "value": "2010-04-28T06:54:19.472104"}, "latest_revision": 3, "key": "/works/OL10003382W", "authors": [{"type": "/type/author_role",
  "author": {"key": "/authors/OL3968834A"}}], "type": {"key": "/type/work"}, "revision": 3}
17 /type/work     /works/OL10003418W      3       2010-04-28T06:54:19.472104      {"title": "La r\u00e9volution", "created": {"type": "/type/datetime", "value": "2009-12-11T01:57:52.839053"}, "covers":
  [5203780], "last_modified": {"type": "/type/datetime", "value": "2010-04-28T06:54:19.472104"}, "latest_revision": 3, "key": "/works/OL10003418W", "authors": [{"type": "/type/author_role", "author":
```

Authors.txt

```
1 /type/author    /authors/OL3980739A     1       2008-04-30T20:50:18.033121      {"name": "Colombe Plante", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/
  authors/OL3980739A", "type": {"key": "/type/author"}, "revision": 1}
2 /type/author    /authors/OL3971384A     1       2008-04-30T20:50:18.033121      {"name": "Cabanes/", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/
  authors/OL3971384A", "type": {"key": "/type/author"}, "revision": 1}
3 /type/author    /authors/OL3968874A     1       2008-04-30T20:50:18.033121      {"name": "Damenie Louis", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/
  authors/OL3968874A", "type": {"key": "/type/author"}, "revision": 1}
4 /type/author    /authors/OL3968874A     1       2008-04-30T20:50:18.033121      {"name": "Damenie Louis", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/
  authors/OL3968874A", "type": {"key": "/type/author"}, "revision": 1}
5 /type/author    /authors/OL3979572A     1       2008-04-30T20:50:18.033121      {"name": "Jerome Maran", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/
  authors/OL3979572A", "type": {"key": "/type/author"}, "revision": 1}
6 /type/author    /authors/OL3980496A     1       2008-04-30T20:50:18.033121      {"name": "Alain Renk", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/
  authors/OL3980496A", "type": {"key": "/type/author"}, "revision": 1}
7 /type/author    /authors/OL3966691A     1       2008-04-30T20:50:18.033121      {"name": "Francois G. Ricaud", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key":
  "/authors/OL3966691A", "type": {"key": "/type/author"}, "revision": 1}
8 /type/author    /authors/OL3967514A     1       2008-04-30T20:50:18.033121      {"name": "Jacquelline Bregetzer", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"},
  "key": "/authors/OL3967514A", "type": {"key": "/type/author"}, "revision": 1}
9 /type/author    /authors/OL3967457A     1       2008-04-30T20:50:18.033121      {"name": "Jacky Loufrani", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/
  authors/OL3967457A", "type": {"key": "/type/author"}, "revision": 1}
10 /type/author   /authors/OL3971853A     1       2008-04-30T20:50:18.033121      {"name": "Brigitte Engammare", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key":
  "/authors/OL3971853A", "type": {"key": "/type/author"}, "revision": 1}
11 /type/author   /authors/OL3972162A     1       2008-04-30T20:50:18.033121      {"name": "Mouret", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/authors/
  OL3972162A", "type": {"key": "/type/author"}, "revision": 1}
12 /type/author   /authors/OL3965554A     1       2008-04-30T20:50:18.033121      {"name": "Bruno Meshaka", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/
  authors/OL3965554A", "type": {"key": "/type/author"}, "revision": 1}
13 /type/author   /authors/OL3977715A     1       2008-04-30T20:50:18.033121      {"name": "D. Cuthbert", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/
  authors/OL3977715A", "type": {"key": "/type/author"}, "revision": 1}
14 /type/author   /authors/OL3968483A     1       2008-04-30T20:50:18.033121      {"name": "Marie-Nadine Antol", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key":
  "/authors/OL3968483A", "type": {"key": "/type/author"}, "revision": 1}
15 /type/author   /authors/OL3965376A     1       2008-04-30T20:50:18.033121      {"name": "Marie-Helene Perennec", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"},
  "key": "/authors/OL3965376A", "type": {"key": "/type/author"}, "revision": 1}
16 /type/author   /authors/OL3968834A     1       2008-04-30T20:50:18.033121      {"name": "Jerome Bourgine", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/
  authors/OL3968834A", "type": {"key": "/type/author"}, "revision": 1}
17 /type/author   /authors/OL3971678A     1       2008-04-30T20:50:18.033121      {"name": "S. Bolsselier", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/
  authors/OL3971678A", "type": {"key": "/type/author"}, "revision": 1}
18 /type/author   /authors/OL3978461A     1       2008-04-30T20:50:18.033121      {"name": "Albrecht /Guyard", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key":
  "/authors/OL3978461A", "type": {"key": "/type/author"}, "revision": 1}
19 /type/author   /authors/OL3972397A     1       2008-04-30T20:50:18.033121      {"name": "Tallemand du Reaux", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key":
  "/authors/OL3972397A", "type": {"key": "/type/author"}, "revision": 1}
20 /type/author   /authors/OL3967591A     1       2008-04-30T20:50:18.033121      {"name": "Fr\u00e8re Luc de Taiz\u00e9", "last_modified": {"type": "/type/datetime", "value":
  "2008-04-30T20:50:18.033121"}, "key": "/authors/OL3967591A", "type": {"key": "/type/author"}, "revision": 1}
21 /type/author   /authors/OL769457A      11      2020-09-30T12:21:52.547543      {"bio": {"type": "/type/text", "value": "Abraham Grace Merritt was born in Beverly, New Jersey. In 1894, he moved with
  his family to Philadelphia, Pennsylvania. He worked to become a lawyer, but instead became a journalist. In 1912 he became assistant editor of The American Weekly until 1937, when he became its editor.
  He also wrote fiction, including eight novels."}, "name": "A. Merritt", "links": [{"url": "http://en.wikipedia.org/wiki/A._Merritt", "type": {"key": "/type/link"}, "title": "Wikipedia page"}],
  "personal_name": "Abraham Merritt", "wikipedia": "http://en.wikipedia.org/wiki/A._Merritt", "death_date": "21 August 1943", "alternate_names": ["Abraham Merritt", "Abraham Grace Merritt"], "created":
  {"type": "/type/datetime", "value": "2008-04-01T03:28:50.625462"}, "photos": [6254096], "last_modified": {"type": "/type/datetime", "value": "2020-09-30T12:21:52.547543"}, "latest_revision": 11, "key":
  "/authors/OL769457A", "birth_date": "20 January 1882", "revision": 11, "type": {"key": "/type/author"}, "remote_ids": {"viaf": "9850578", "wikidata": "Q3303765", "isni": "0000000114675787"}}
22 /type/author   /authors/OL3973794A     1       2008-04-30T20:50:18.033121      {"name": "Serge Beninati", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/
  authors/OL3973794A", "type": {"key": "/type/author"}, "revision": 1}
23 /type/author   /authors/OL93018A       2       2008-09-08T03:23:05.850541      {"name": "Alan Hess", "personal_name": "Alan Hess", "last_modified": {"type": "/type/datetime", "value":
  "2008-09-08T03:23:05.850541"}, "key": "/authors/OL93018A", "type": {"key": "/type/author"}, "revision": 2}
24 /type/author   /authors/OL3979174A     1       2008-04-30T20:50:18.033121      {"name": "Bela Marko", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/
  authors/OL3979174A", "type": {"key": "/type/author"}, "revision": 1}
```

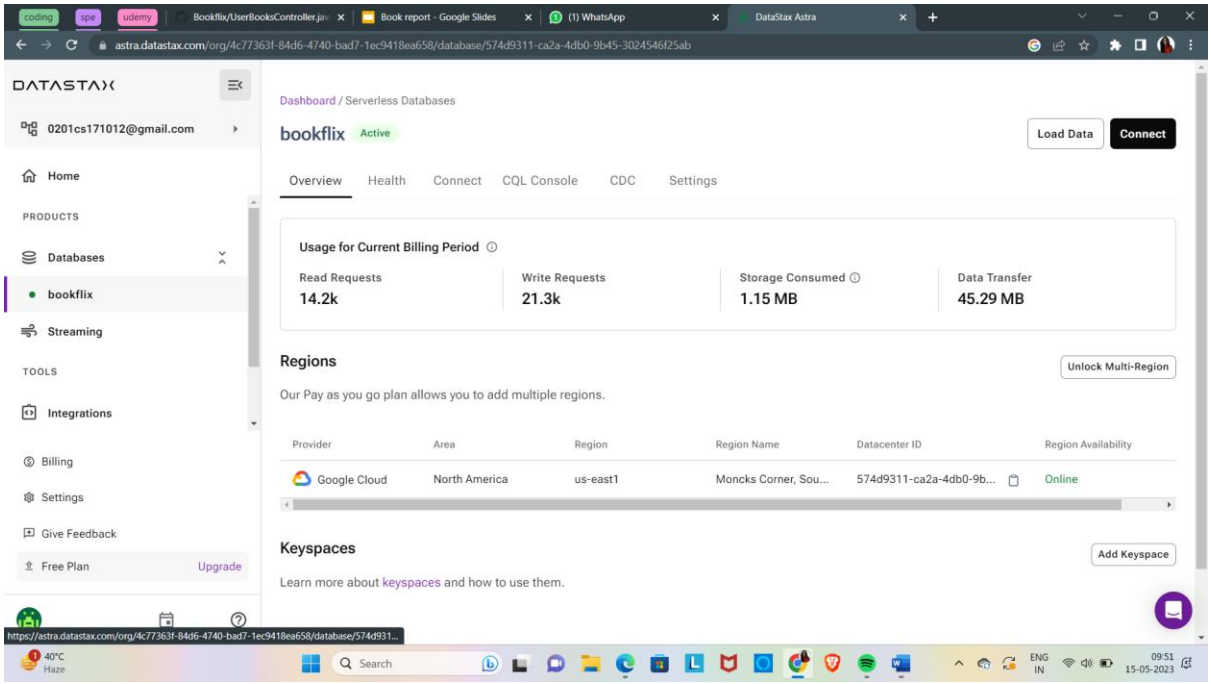
backend > src > main > java > com > example > bookflix > BookflixApplication.java > Language Support for Java(TM) by Red Hat > BookflixApplication > loadBooks()

```
113
114     public void loadBooks() throws IOException {
115         DateTimeFormatter dt = DateTimeFormatter.ofPattern("yyyy-MM-dd'T'HH:mm:ss.SSSSSS");
116         Files.lines(Paths.get(workFile)).forEach(l -> {
117             try {
118                 String json = l.substring(l.indexOf("{"));
119                 JSONObject jsonObj = new JSONObject(json);
120                 Book b = new Book();
121                 b.setId(jsonObj.getString(name:"key").replace("/works/", ""));
122                 b.setName(jsonObj.optString(name:"title"));
123
124                 JSONObject jsonDesc = jsonObj.optJSONObject(name:"decription");
125                 if (jsonDesc != null) {
126                     b.setDescription(jsonDesc.optString(name:"value"));
127                 }
128
129                 JSONObject pubdate = jsonObj.optJSONObject(name:"created");
130                 if (pubdate != null) {
131                     String srVal = pubdate.optString(name:"value");
132                     b.setPublishedDate(LocalDate.parse(srVal, dt));
133                 }
134
135                 JSONArray coversArray = jsonObj.optJSONArray(name:"covers");
136                 if (coversArray != null) {
137                     List<String> coverIds = new ArrayList<>();
138                     for (int i = 0; i < coversArray.length(); i++) {
139                         coverIds.add(String.valueOf(coversArray.getLong(i)));
140                     }
141                     b.setCoverIds(coverIds);
142                 }
143
144                 JSONArray authorsArray = jsonObj.optJSONArray(name:"authors");
145                 if (authorsArray != null) {
146                     List<String> authorIds = new ArrayList<>();
147                     for (int i = 0; i < authorsArray.length(); i++) {
148                         String authorid = authorsArray.getJSONObject(i).getJSONObject(name:"author").getString(name:"key")
149                                 .replace("/authors/", "");
150                         authorIds.add(authorid);
151                     }
```

backend > src > main > java > com > example > bookflix > BookflixApplication.java > Language Support for Java(TM) by Red Hat > BookflixAp

```
91
92
93      public void loadAuthors(int startline) throws IOException {
94
95          final AtomicInteger currentLine = new AtomicInteger(0);
96          Files.lines(Paths.get(authorFile)).forEach(l -> {
97              try {
98                  if (currentLine.incrementAndGet() > startline) {
99                      String json = l.substring(l.indexOf("{"));
100                     JSONObject jsonObj = new JSONObject(json);
101                     Author a = new Author();
102                     a.setName(jsonObj.optString(name:"name"));
103                     a.setPersonalName(jsonObj.optString(name:"personal_name"));
104                     a.setId(jsonObj.optString(name:"key").replaceAll("/authors/", ""));
105                     authorRepository.save(a);
106                     System.out.println(currentLine.get() + ": Saving author: " + a.getName());
107                 }
108             } catch (Exception e) {
109                 e.printStackTrace();
110             }
111         });
112     }
113
114     public void loadBooks() throws IOException {
115         DateTimeFormatter dt = DateTimeFormatter.ofPattern("yyyy-MM-dd'T'HH:mm:ss.SSSSSS");
116         Files.lines(Paths.get(workFile)).forEach(l -> {
117             try {
118                 String json = l.substring(l.indexOf("{"));
119                 JSONObject jsonObj = new JSONObject(json);
120                 Book b = new Book();
121                 b.setId(jsonObj.getString(name:"key").replace("/works/", ""));
122                 b.setName(jsonObj.optString(name:"title"));
```

- After Populating the Database.




```
CREATE TABLE main.author_by_id (
    id text PRIMARY KEY,
    author_name text,
    personal_name text
) WITH additional_write_policy = '99PERCENTILE'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.UnifiedCompactionStrategy'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99PERCENTILE';

CREATE TABLE main.book_by_id (
    id text PRIMARY KEY,
    author_id list<text>,
    author_names list<text>,
    book_description text,
    book_name text,
    cover_ids list<text>,
    published_date date
) WITH additional_write_policy = '99PERCENTILE'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.UnifiedCompactionStrategy'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99PERCENTILE';
```

```
CREATE TABLE main.book_by_user_and_bookid (
    user_id text,
    book_id text,
    completed_date date,
    rating int,
    reading_status text,
    started_date date,
    PRIMARY KEY ((user_id, book_id))
) WITH additional_write_policy = '99PERCENTILE'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.UnifiedCompactionStrategy'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99PERCENTILE';

CREATE TABLE main.books_by_user (
    user_id text,
    readingstatus text,
    timeuuid timeuuid,
    book_id text,
    author_names list<text>,
    book_name text,
    cover_ids list<text>,
    rating int,
    PRIMARY KEY (user_id, readingstatus, timeuuid, book_id)
) WITH CLUSTERING ORDER BY (readingstatus ASC, timeuuid DESC, book_id ASC)
    AND additional_write_policy = '99PERCENTILE'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.UnifiedCompactionStrategy'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
```

4.1.3 Docker

Docker is a continuous delivery tool. In a business enterprise, requirements keep changing and new updates keep waiting for deployment. The process of deployment has various prerequisite steps that need to be followed. After installing some prerequisite packages, we can install docker. Installing Docker

- apt install docker-ce
- usermod-aG docker ${USER}
- su- ${USER}

4.2 Source Control Management

We have used Git for source code management. A git repository is created on the Github with the team members as the collaborators. Hence, any of the members can push the changed code to the repository. Github helped us to work in a synchronous manner so we can change or access the code at the same time. IntelliJ provides VCS (version control system) to handle git commands and connects the local repository to the remote Github repository.

1. git init: This command is used to create a new blank repository. It is used to make an existing project as a Git project.

2. git clone: This command is used to clone the remote repository to local system.

3. git add: The git add command is used to add file contents to the Index (Staging Area). This command updates the current content of the working tree to the staging area.

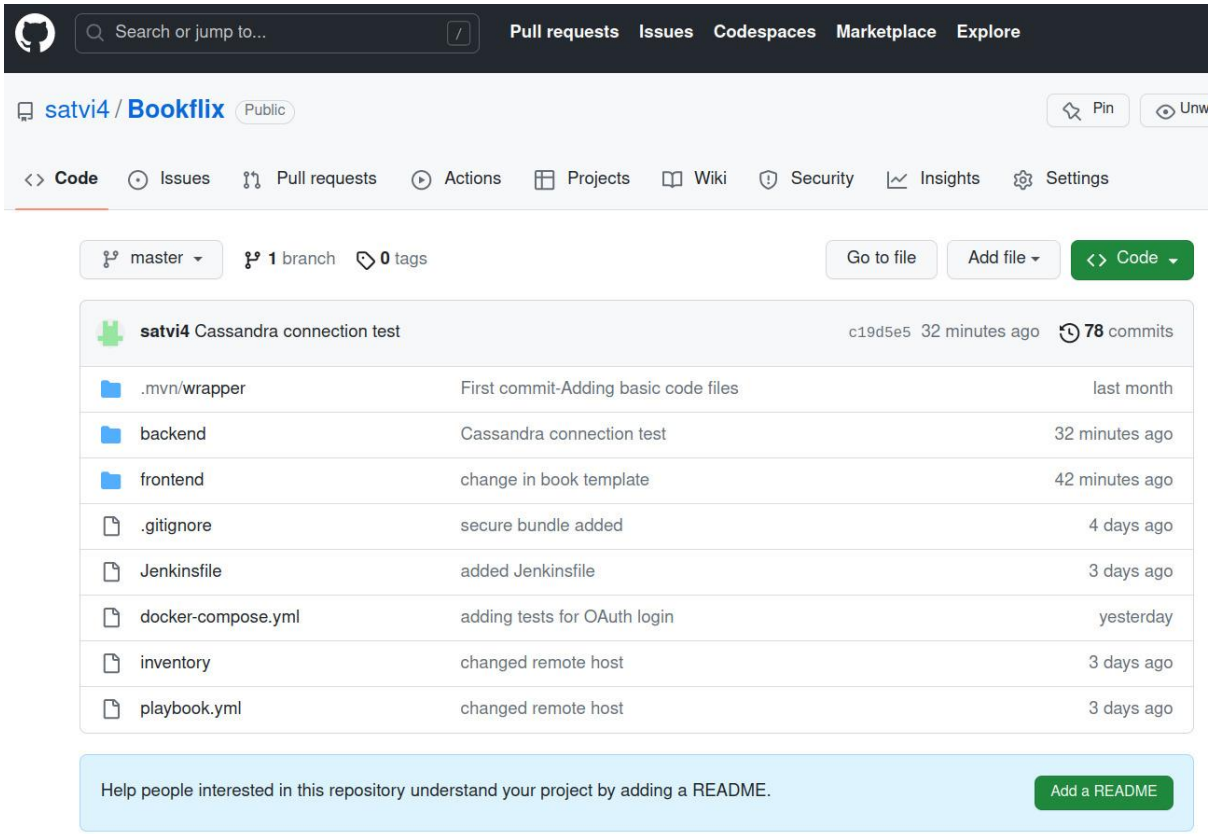
4. git commit-m: This creates a snapshot of the files and commits them by creating a SHA id.

5. git remote add origin: This creates a new remote called origin located at given URL.

6. git push-u origin: This command is used for pushing the local changes to repository on server on specified branch.

Github repository link for the project:

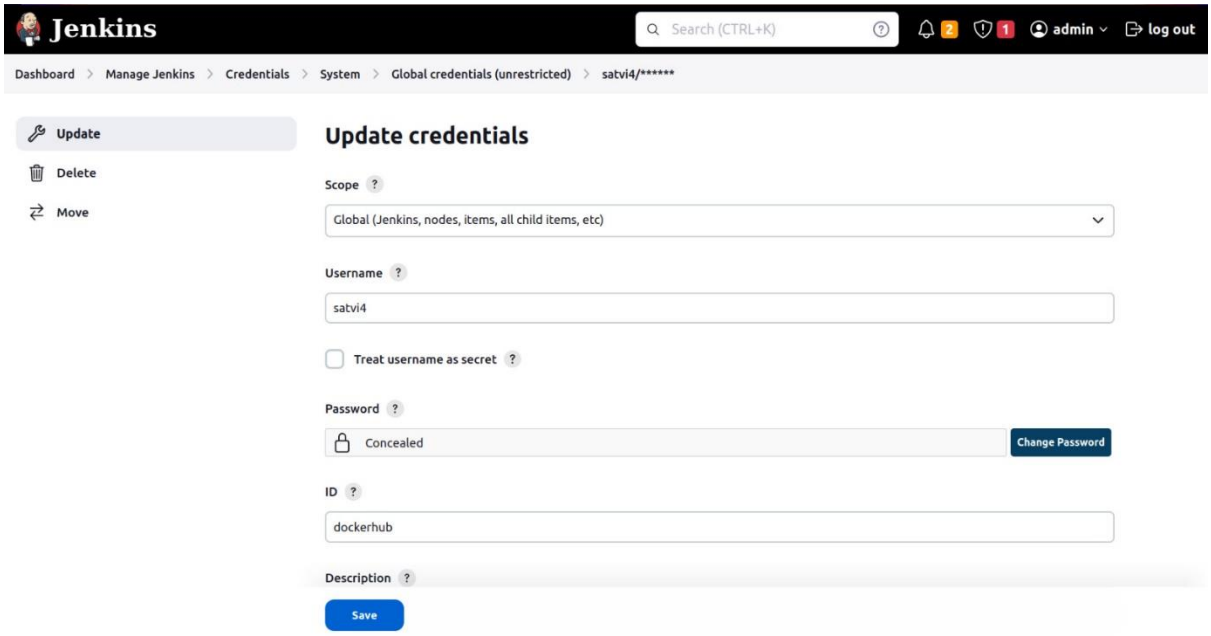
https://github.com/satvi4/Bookflix


Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
satvi4 / Bookflix
Public
Pin
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
master
1 branch
0 tags
Go to file
Add file
Code
satvi4 Cassandra connection test
c19d5e5 32 minutes ago
78 commits
.mvn/wrapper
First commit-Adding basic code files
last month
backend
Cassandra connection test
32 minutes ago
frontend
change in book template
42 minutes ago
.gitignore
secure bundle added
4 days ago
Jenkinsfile
added Jenkinsfile
3 days ago
docker-compose.yml
adding tests for OAuth login
yesterday
inventory
changed remote host
3 days ago
playbook.yml
changed remote host
3 days ago
Help people interested in this repository understand your project by adding a README.
Add a README

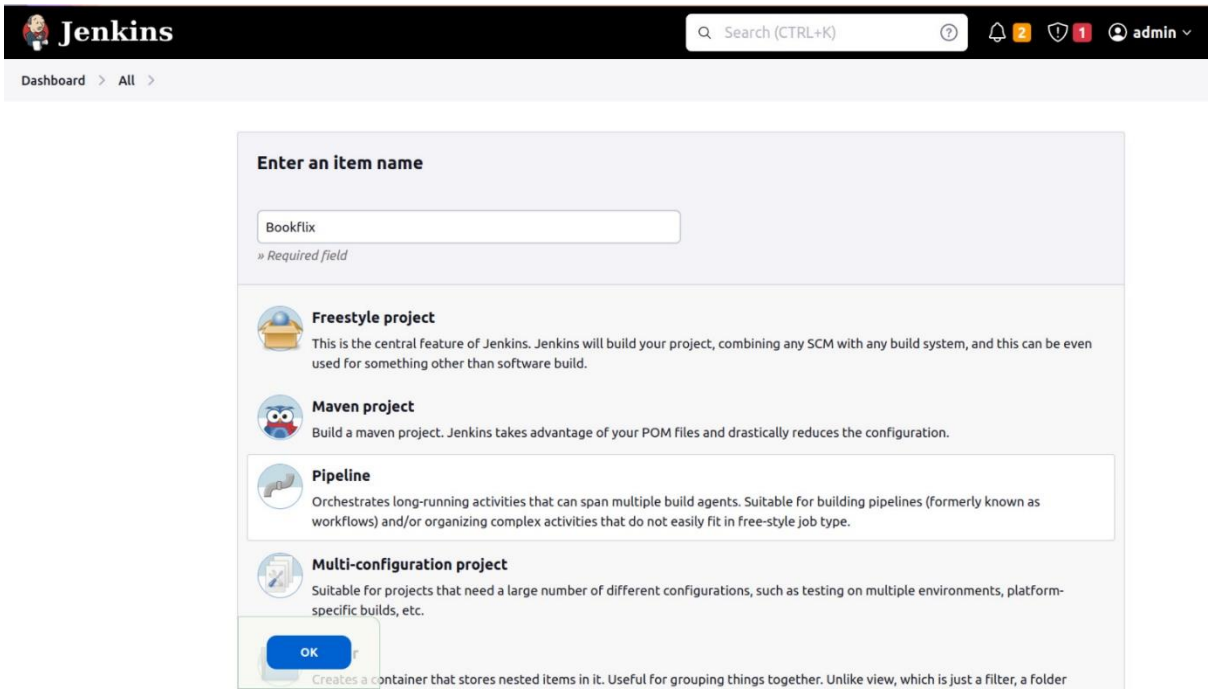
4.2.1 CI Jenkins Pipeline

We used Jenkins Pipeline to clone these repositories from git.

➢ First we add credentials of Dockerhub repository



➢ Then we create a Jenkins pipeline.



➢ Git Script with Jenkins pipeline

Jenkinsfile

```
pipeline {
    environment{
        dockerimg = ''
        dockerimg2 = ''
        DOCKERHUB_CREDENTIALS = credentials('dockerhub')
    }
    agent any

    stages {
        stage('Git pull') {
            steps {
                git 'https://github.com/satvi4/Bookflix'
            }
        }
        stage('Maven Build'){
            steps {
                dir('backend'){
                    sh 'mvn clean install'
                }
            }
        }
```

4.3 Build

4.3.1 Maven Apache

Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting, and documentation from a central piece of information. It handles dependencies and plugins so we don't need to add jar files for the library explicitly. First, it checks for project dependencies in pom.xml and then compiles the code for any errors.

We have to install maven in our system so we can build what we write. To install Maven, we want to first install its dependency JAVA, here we will be installing java version 8.

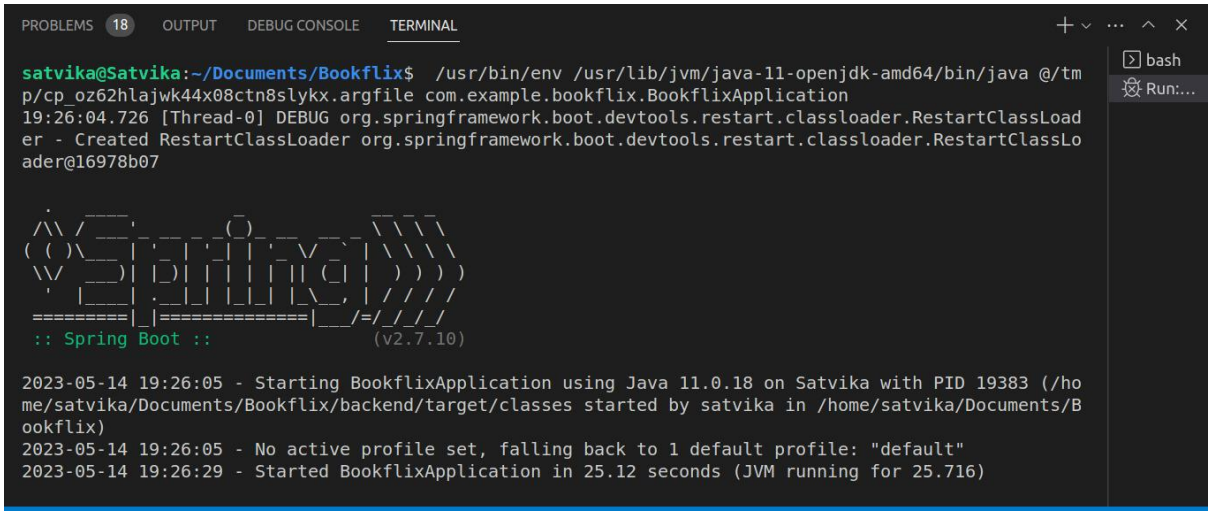
sudo apt update

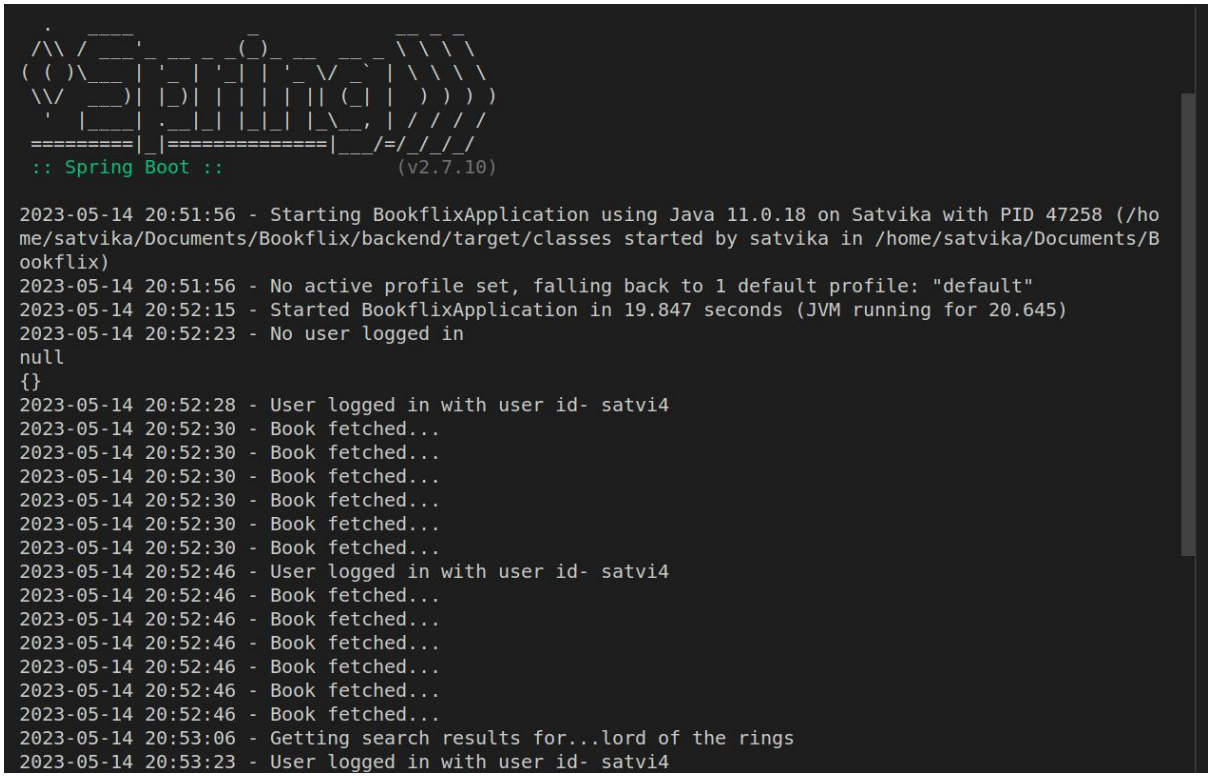
sudo apt install openjdk-8-jdk

We will now install maven on our setup

sudo apt update

sudo apt install maven


PROBLEMS 18
OUTPUT
DEBUG CONSOLE
TERMINAL
bash
Run:...
satvika@Satvika:~/Documents/Bookflix$ /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java @/tmp/cp_oz62hlajwk44x08ctn8slykx.argfile com.example.bookflix.BookflixApplication
19:26:04.726 [Thread-0] DEBUG org.springframework.boot.devtools.restart.classloader.RestartClassLoader - Created RestartClassLoader org.springframework.boot.devtools.restart.classloader.RestartClassLoader@16978b07
:: Spring Boot :: (v2.7.10)
2023-05-14 19:26:05 - Starting BookflixApplication using Java 11.0.18 on Satvika with PID 19383 (/home/satvika/Documents/Bookflix/backend/target/classes started by satvika in /home/satvika/Documents/Bookflix)
2023-05-14 19:26:05 - No active profile set, falling back to 1 default profile: "default"
2023-05-14 19:26:29 - Started BookflixApplication in 25.12 seconds (JVM running for 25.716)


:: Spring Boot :: (v2.7.10)
2023-05-14 20:51:56 - Starting BookflixApplication using Java 11.0.18 on Satvika with PID 47258 (/home/satvika/Documents/Bookflix/backend/target/classes started by satvika in /home/satvika/Documents/Bookflix)
2023-05-14 20:51:56 - No active profile set, falling back to 1 default profile: "default"
2023-05-14 20:52:15 - Started BookflixApplication in 19.847 seconds (JVM running for 20.645)
2023-05-14 20:52:23 - No user logged in
null
{}
2023-05-14 20:52:28 - User logged in with user id- satvi4
2023-05-14 20:52:30 - Book fetched...
2023-05-14 20:52:30 - Book fetched...
2023-05-14 20:52:30 - Book fetched...
2023-05-14 20:52:30 - Book fetched...
2023-05-14 20:52:30 - Book fetched...
2023-05-14 20:52:30 - Book fetched...
2023-05-14 20:52:46 - User logged in with user id- satvi4
2023-05-14 20:52:46 - Book fetched...
2023-05-14 20:52:46 - Book fetched...
2023-05-14 20:52:46 - Book fetched...
2023-05-14 20:52:46 - Book fetched...
2023-05-14 20:52:46 - Book fetched...
2023-05-14 20:52:46 - Book fetched...
2023-05-14 20:53:06 - Getting search results for...lord of the rings
2023-05-14 20:53:23 - User logged in with user id- satvi4

4.3.2 CI Pipeline

Jenkins uses the system's maven to build the maven project, so we must have the maven integration plugin installed on our Jenkins.

- Install Maven integration plugin
- We add another stage to our pipeline.
- In mvn clean install, it also runs all the test cases for the project and creates the executable jar file in the target folder in the current working directory.

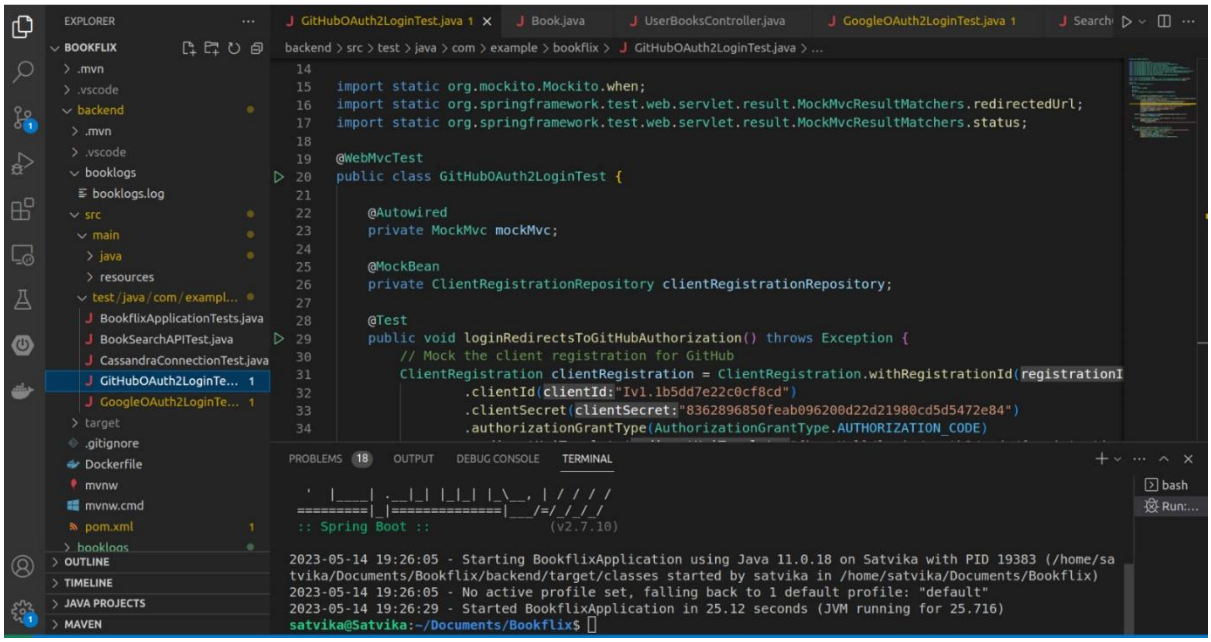
Jenkinsfile

```
pipeline {
    environment{
        dockerimg = ''
        dockerimg2 = ''
        DOCKERHUB_CREDENTIALS = credentials('dockerhub')
    }
    agent any

    stages {
        stage('Git pull') {
            steps {
                git 'https://github.com/satvi4/Bookflix'
            }
        }
        stage('Maven Build'){
            steps {
                dir('backend'){
                    sh 'mvn clean install'
                }
            }
        }
```

4.4 Testing

Jenkins provides us with continuous integration which includes integrated testing, so every time we push code to GitHub it integrates all the different modules of the project and tests their proper functioning. JUnit is a simple, open-source framework to write and run repeatable tests. It is an instance of the xUnit architecture for unit testing frameworks. JUnit promotes the idea of "first testing then coding", which emphasises on setting up the test data for a piece of code that can be tested first and then implemented.

This approach is like "test a little, code a little, test a little, code a little." It increases the productivity of the programmer and the stability of program code, which in turn reduces the stress on the programmer and the time spent on debugging.



For testing within java code, we use MockMvc. MockMvc test cases are written in covering the scope of the project. Also, MockMvc dependency is added in pom.xml for maven to resolve the dependency.

4.5 Docker Artifact

Docker is an application that makes it simple and easy to run application processes in a container, which are like virtual machines, only more portable, more resource-friendly, and more dependent on the host operating system.

Artifact repository tools store all of the bulk binary artifact files that rarely, if ever, need to be altered or changed. Docker offers a tool like this. First, we need to install docker on our system. Once it is installed, we can run it using following command:

$ sudo systemctl start docker

Here, our aim is to build a docker image out of the jar file that will be created in build phase and push the latest image to Docker Hub, which will then be pulled by Ansible to deploy it to other machines. All this will be automated in Jenkins pipeline. To push the docker image, use the following command: docker push /:tagname

To pull the docker image, use the following command:

docker pull

To run the docker image, use the following command:

docker run –it

Also, make sure to add Jenkins to docker group, so that Jenkins can use docker for build docker image. This can be done by following command:

$ sudo usermod-aG docker Jenkins

Docker hub links for the project:-

https://hub.docker.com/r/anoushkachouksey/bookflix-backend

https://hub.docker.com/r/anoushkachouksey/bookflix-frontend

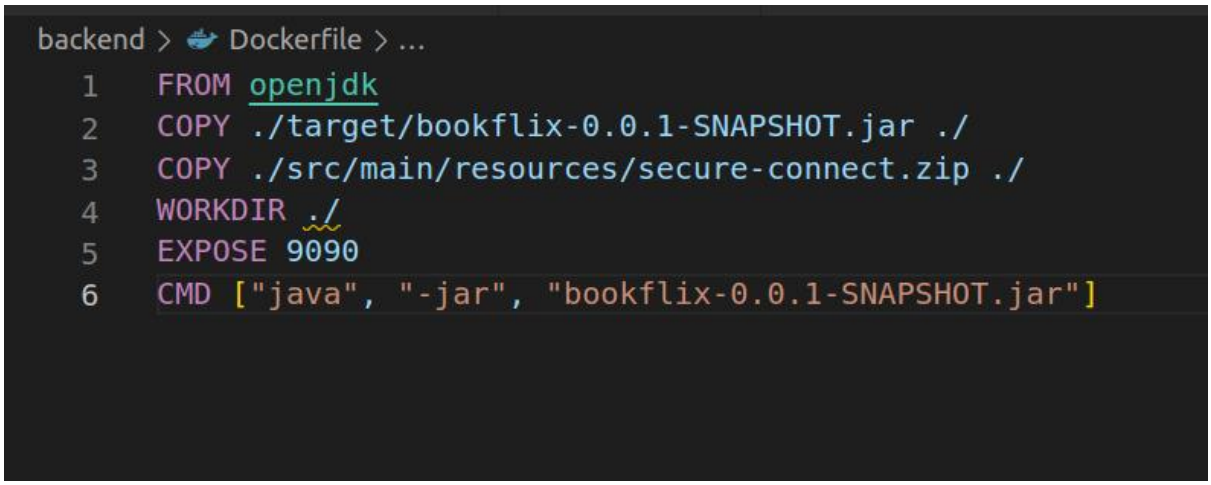
https://hub.docker.com/r/satvi4/bookflix-backend

https://hub.docker.com/r/satvi4/bookflix-frontend

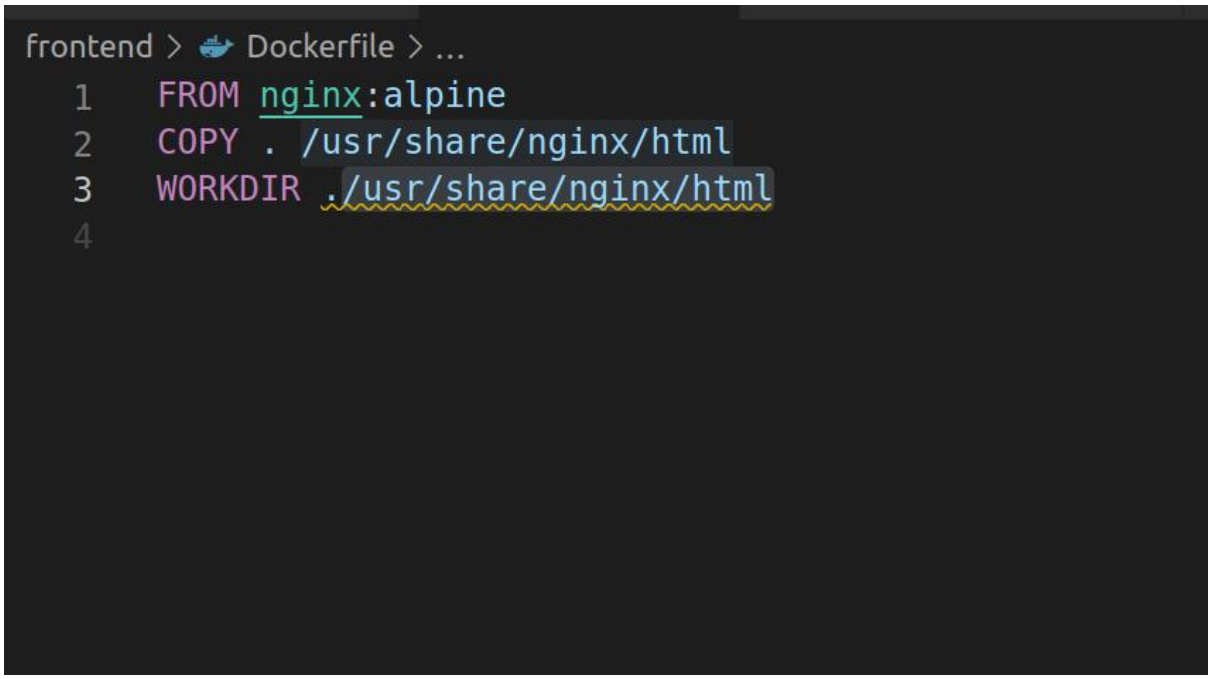
4.5.1 CI Pipeline

➢ Add docker integration plugin for docker in Jenkins under Jenkins → Plugin Manager.

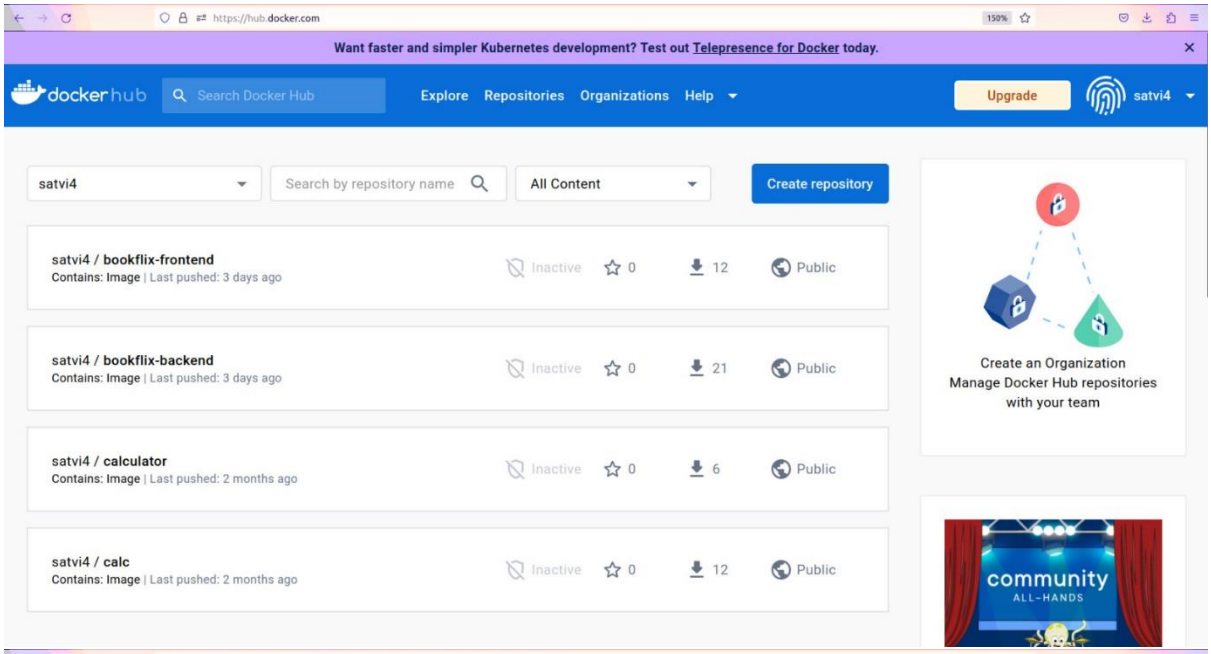
➢ Build and Push Spring Boot Image.

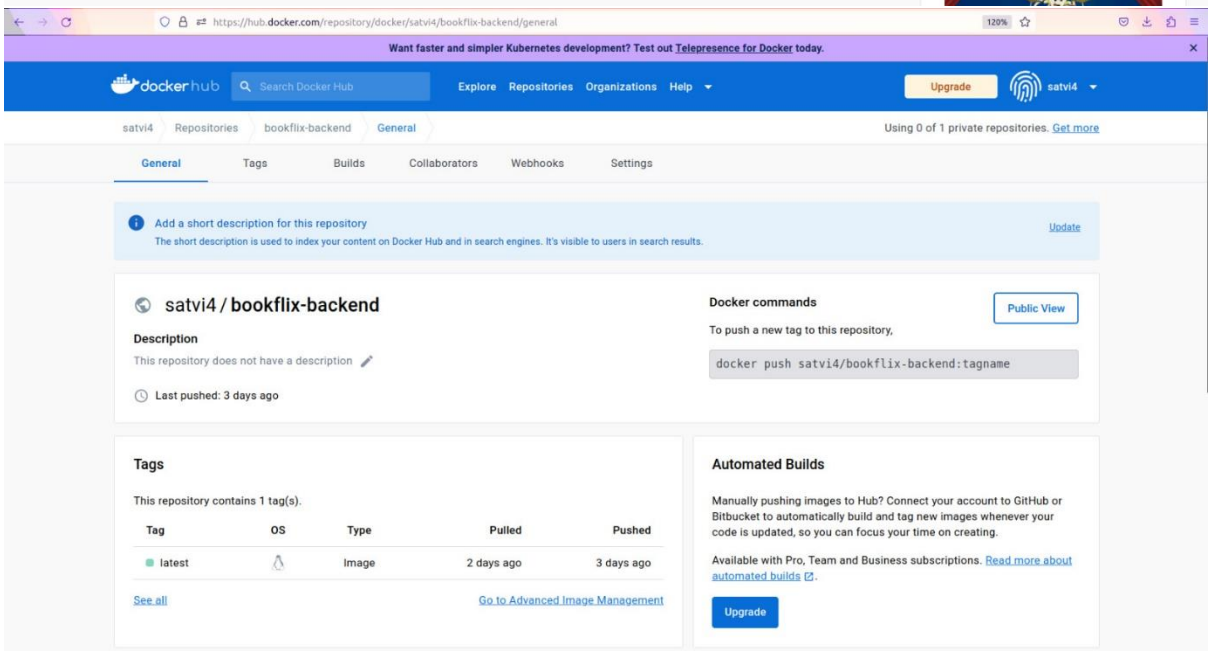


➢ Build and Push Frontend Image



➢ After a successful deployment of docker images, we can see the result on our docker-hub account.





4.5.2 Docker compose

Docker Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration.

Docker Compose commands:

- Docker-compose up: Start all services (it will run docker-compose.yml file)
- Docker-compose build: This command will build the images with the help of docker-compose.yml file.
- Docker-compose down: Stop all the services.

docker-compose.yml

```yaml
services:
  bookflix-frontend:
    image: satvi4/bookflix-frontend:latest
    container_name: bookflix-frontend
    stdin_open: true
    ports:
      - "3000:3000"
    networks:
      my-net:
    restart: always
    volumes:
      - /frontend:/usr/share/nginx/html/templates

  bookflix-backend:
    image: satvi4/bookflix-backend:latest
    container_name: bookflix-backend
    ports:
      - "9090:9090"
    expose:
      - 9090
    networks:
      my-net:
    # Using Docker volume for persisting logs
    volumes:
      - /logs:/booklogs:rw
      - /frontend:/templates

networks:
  my-net:
```

4.6 Continuous Deployment

A Deployment pipeline is the process of taking code from version control and making it readily available to users of your application in an automated fashion. When a team of developers are working on projects or features, they need a reliable and efficient way to build, test and deploy their work.

We are using Ansible for the deployment. It is a system of configuration management written in Python programming language which uses a declarative markup language to describe configurations. It's used for automation of configuration and OS setup. It is often used to manage Linux-nodes, but Windows is also supported.

Ansible allows us to write 'Playbooks' that are descriptions of the desired state of our systems, which are usually kept in source control. Ansible then does the hard work of getting your systems to that state no matter what state they are currently in. Playbooks make your installations, upgrades, and day-to-day management repeatable and reliable.

First, we need to install ansible on our system. Also, Ansible in the backend uses python to run various small codes and uses ssh in linux to connect to the hosts machines. So, we need to make sure that python (generally installed in linux) and ssh server are installed.

We can install them using following commands:

```
$ sudo apt install openssh-server
```

```
$ ssh-keygen-t rsa
```

```
$ sudo apt update
```

```
$ sudo apt install ansible
```

After the ansible is installed, we can verify it by checking its version using

```
$ ansible –version
```

4.6.1 CI Pipeline

➢ Jenkins would call Ansible to deploy the docker image on the deployment node

➢ Ansible uses YAML syntax for expressing Ansible playbooks. YAML uses simple key-value pairs to represent the data.

```
                }
            }
        }
        stage('Ansible Deploy'){
            steps{
                sh 'export LANG=en_US.UTF-8'
                ansiblePlaybook becomeUser: null, colorized: true, disableHostKeyChecking: true, installat
            }
        }
    }
}
```

The dictionary is represented in key: value pair.

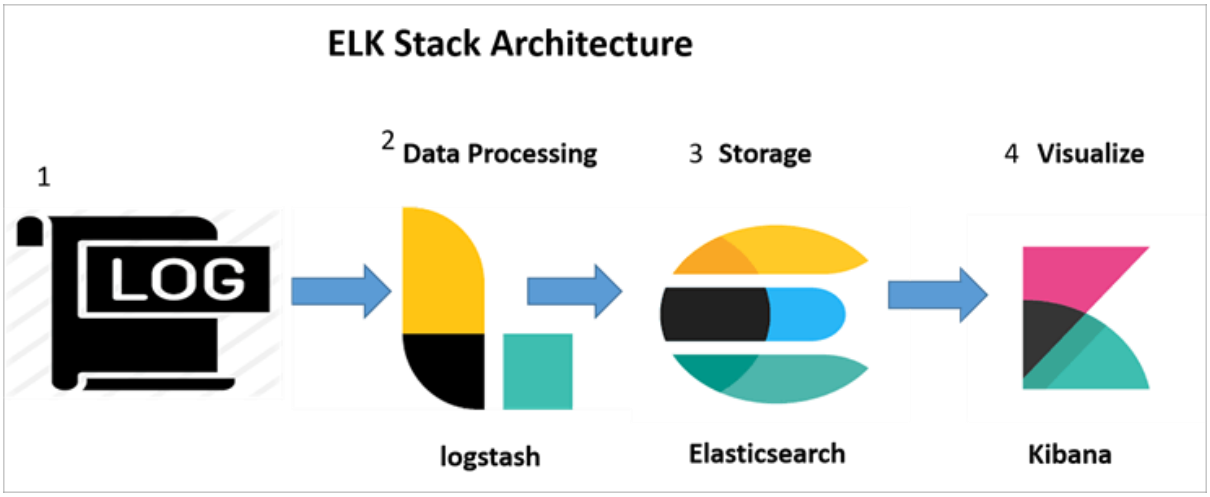
playbook.yml

```yaml
- name: Deploy docker images
  hosts: localhost
  tasks:
    - name: Copy Docker Compose file from host machine to remote host
      copy:
        src: ./docker-compose.yml
        dest: ./

    - name: Pull the Docker images specifed in docker-compose
      command: docker-compose -f docker-compose.yml pull

    - name: Create containers for the pulled Docker images
      command: docker-compose -f docker-compose.yml up -d --force-recreate

    - name: Prune the dangling Docker images
      command: docker image prune --force
```

4.7 Monitor

The final stage in the DevOps paradigm is monitoring.

In this step, developers keep track of user interaction and another usage of the application accordingly. The architecture used for monitoring is the ELK stack. "ELK" is the acronym for three open-source projects: Elasticsearch, Logstash, and Kibana.



- Elasticsearch is a search and analytics engine.
- Logstash is a server-side data processing pipeline that ingests data from multiple sources simultaneously, transforms it, and then sends it to a "stash" like Elasticsearch.
- Kibana lets users visualise data with charts and graphs in Elasticsearch.

The ELK Stack is famous in light of the fact that it satisfies a need in the log, the executives and investigation space.

PATTERN

%d{dd-MM-yyyy HH:mm:ss.SSS} [%thread] %-5level %logger{36}.%M %msg%n

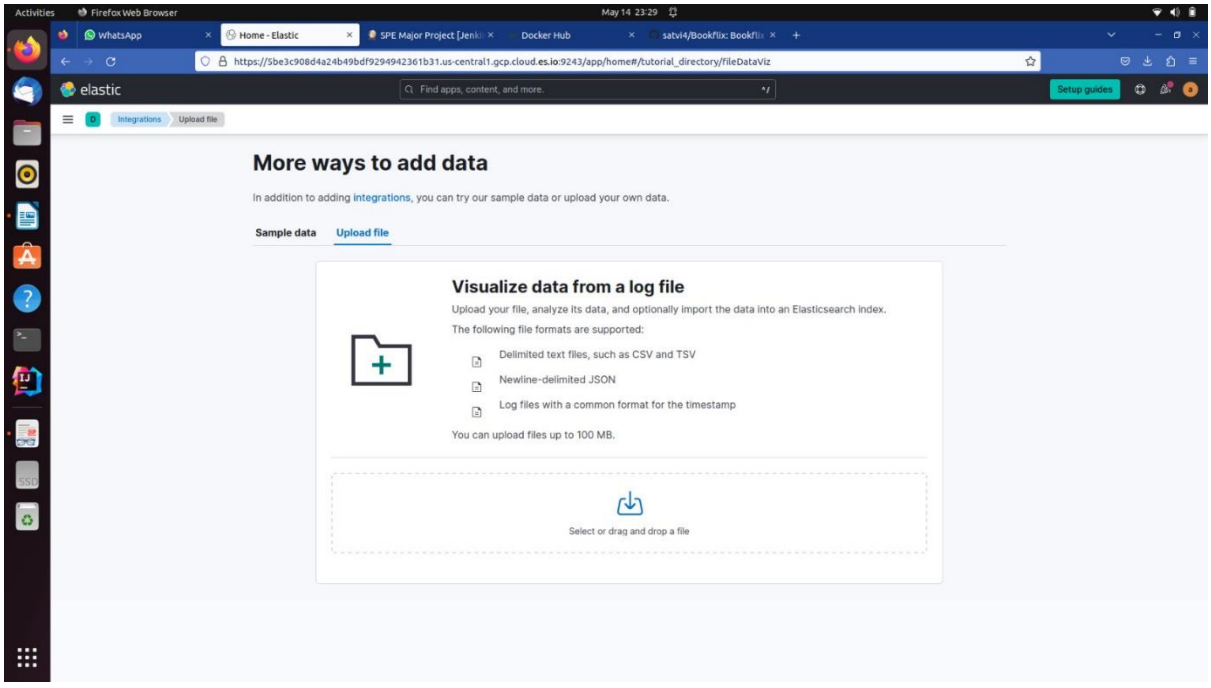
- **%d** – Returns the time when the log message occurred.
- **%thread** – Returns the name of the thread in which the log message occurred.
- **%-5level** – Returns the logging level of the log message (ERROR, WARN, INFO, DEBUG, and TRACE).
- **%logger{64}** – Returns the package with the package and class name where the log message occurred. The number 64 inside the brackets represents the maximum length of the package and class name combined. You can change this number as per your need.
- **%M** – Returns the name of the method where the log message has occurred.
- **%msg** – Returns the actual log message.
- **%n** – Line break.

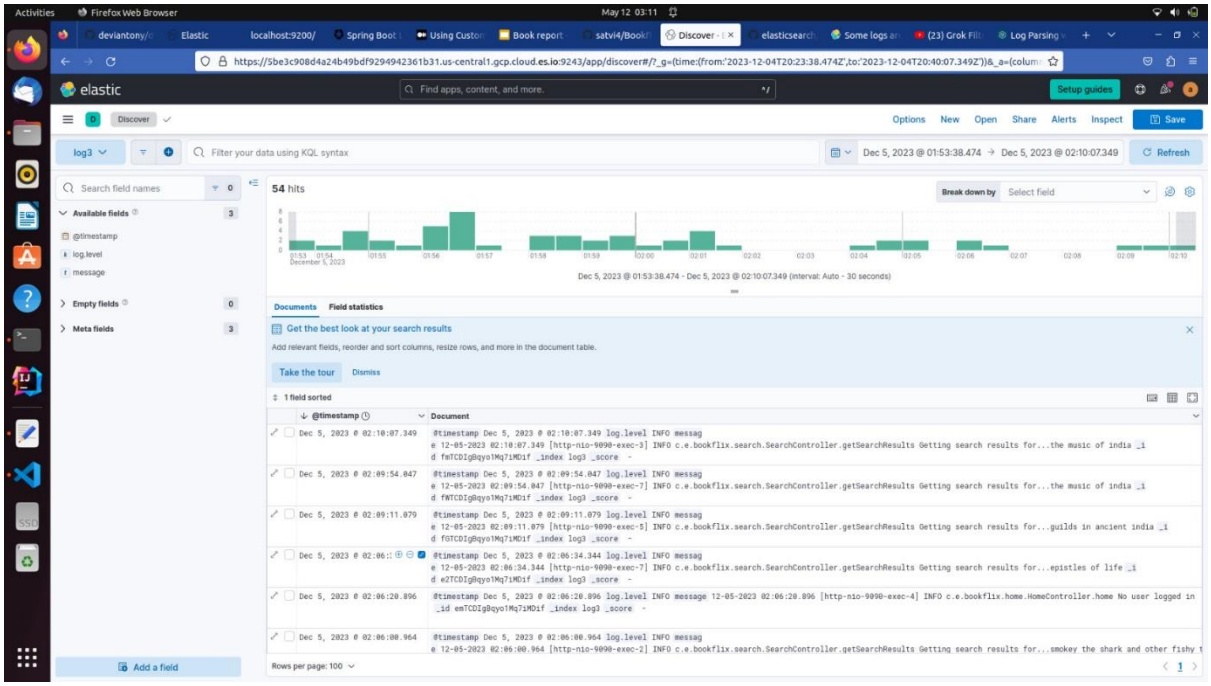
SAMPLE LOG

2023-05-11 12:33:25.395 [http-nio-9090-exec-9] INFO c.e.bookflix.home.HomeController.lambda$0 Book fetched…

GROK PATTERN

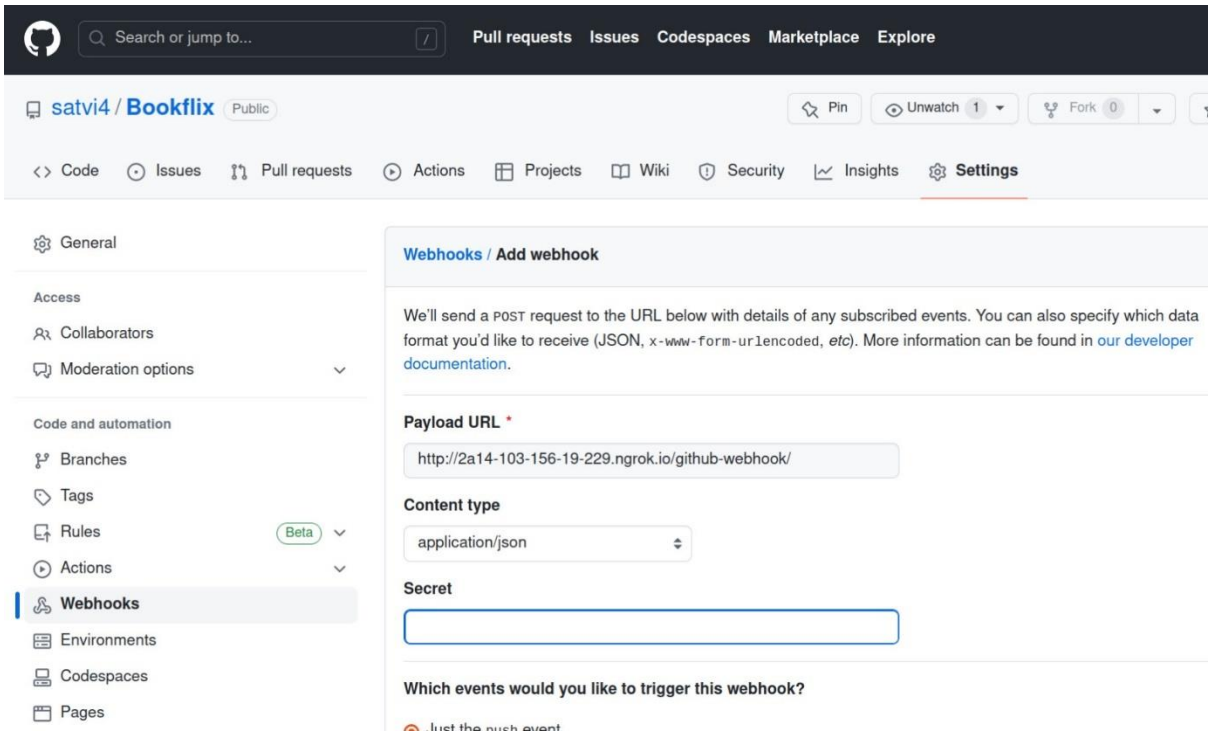
%{DATESTAMP:logDate}%{SPACE}\[%{JAVALOGMESSAGE:thread}\]%{SPACE}%{LOGLEVEL:LOG}%{SPACE}%{JAVACLASS:CLASS}%{SPACE}%{GREEDYDATA:message}

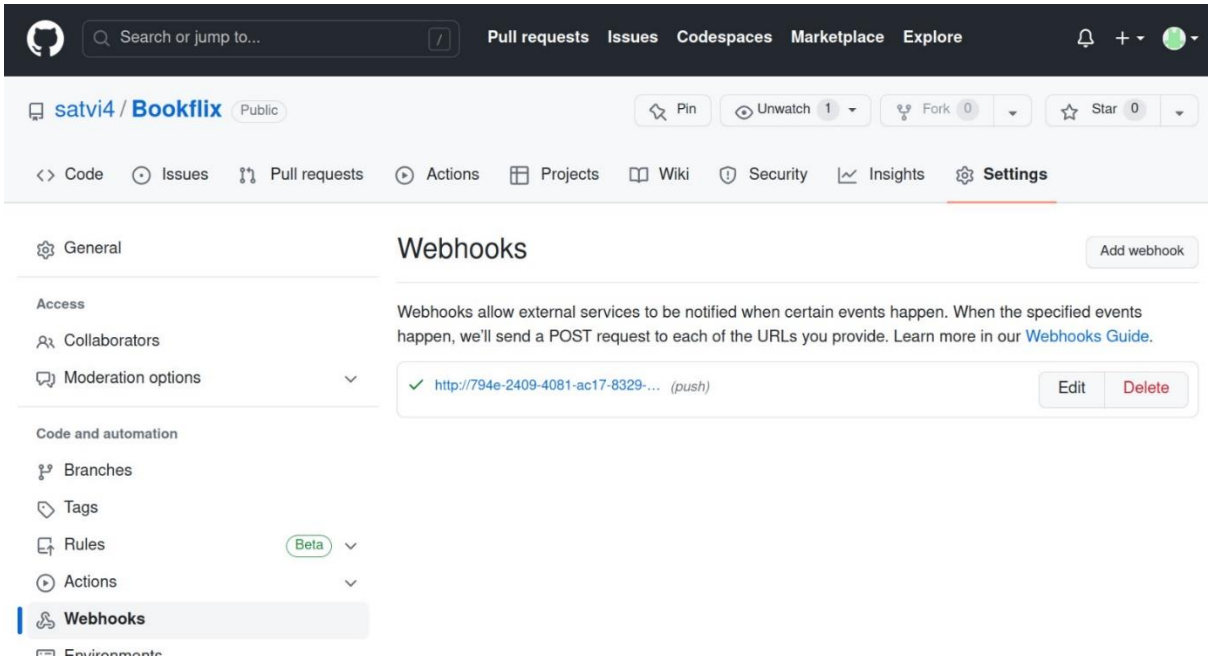



elastic
Discover
log3
54 hits
Documents
Field statistics
Get the best look at your search results
Take the tour
Dismiss

4.8 Working with Webhooks

After committing changes to the GitHub repository, we want Jenkins to trigger build automatically. For this we implement Webhooks.

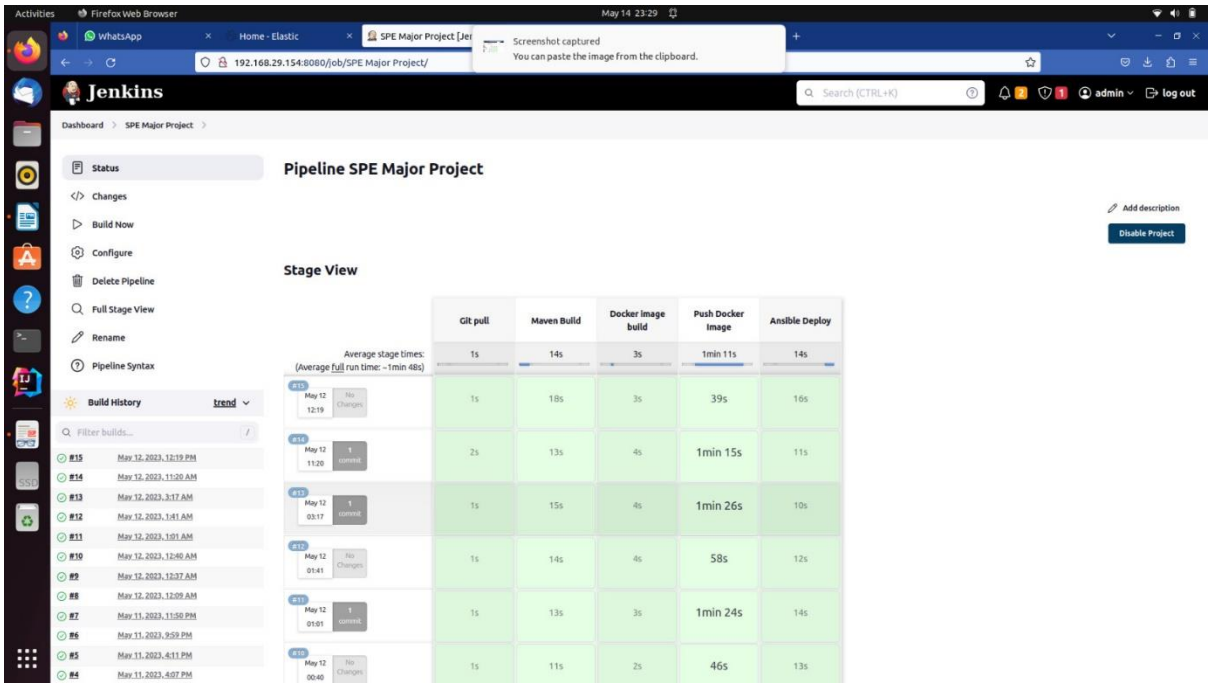




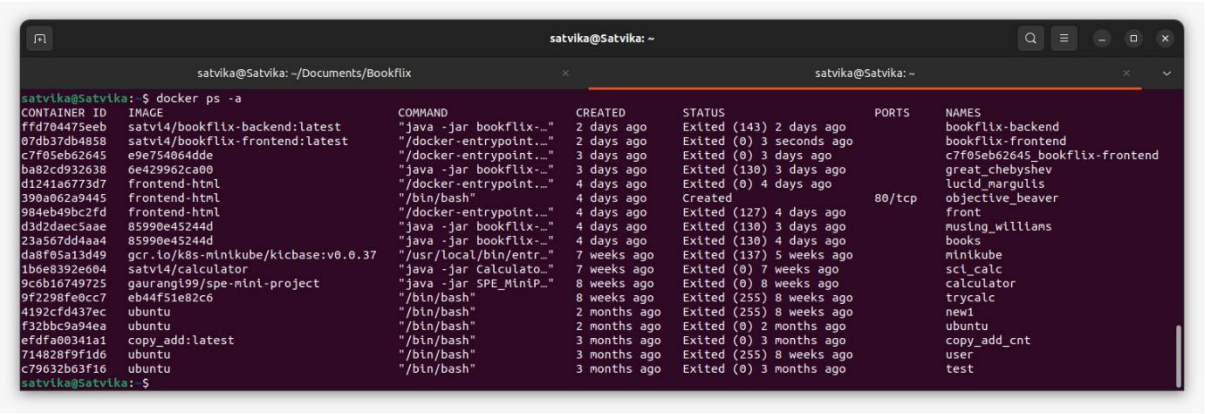
4.9 Building Pipeline and Running on Deployed Machine

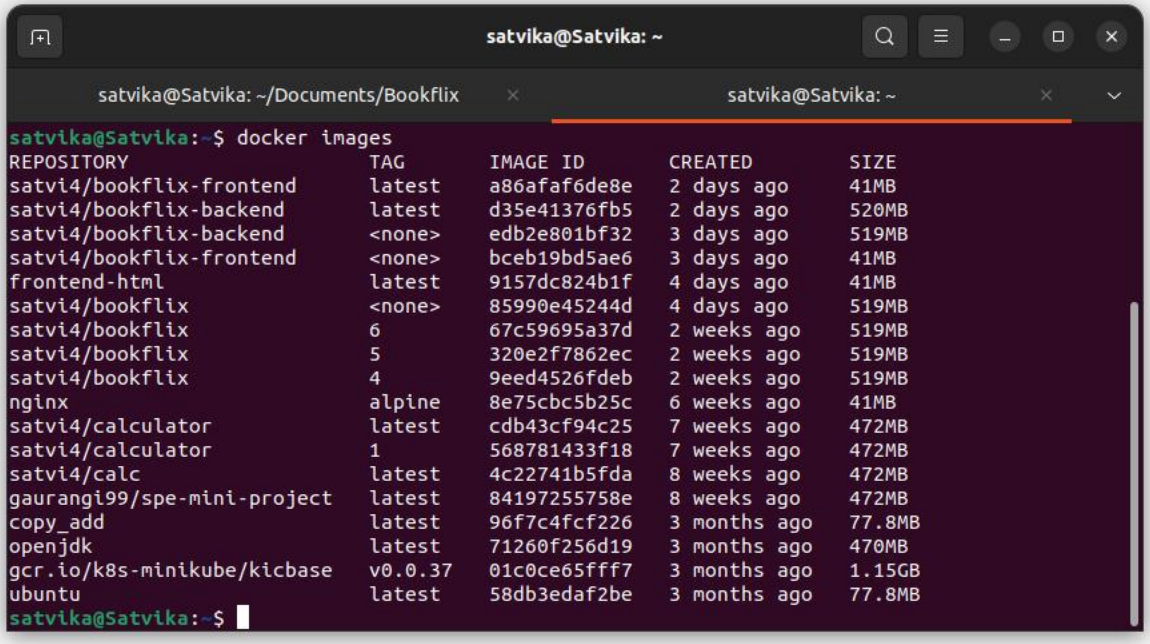
After completing all stages of SDLC we deployed the images successfully to the deployment node. We pulled the repo from git, build from maven, build and deploy the docker images and at last configured Ansible with Jenkins.

- Pipeline stages in Jenkins.



- Deployed images on the target machine.



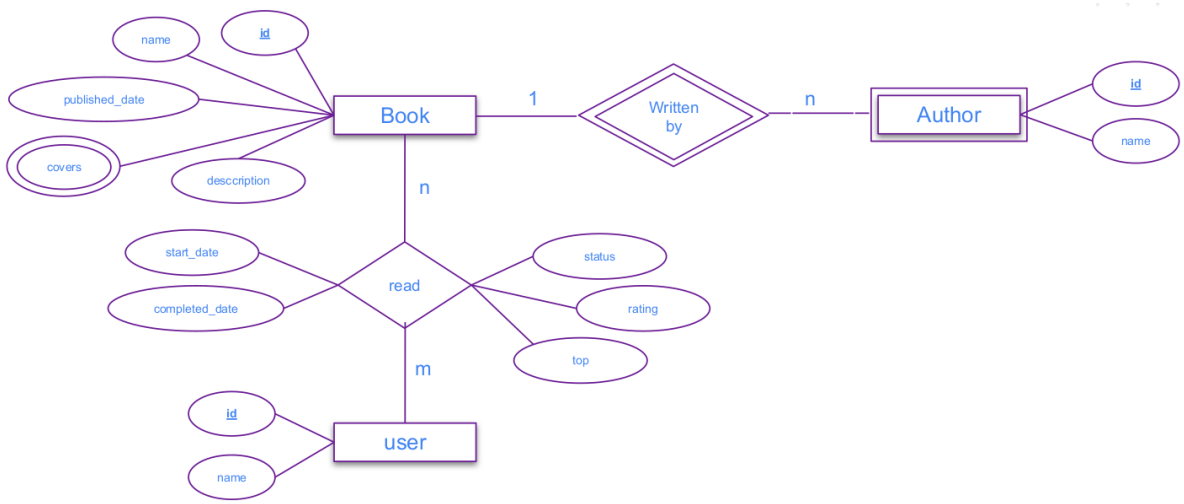


5. Experimental Setup

5.1 Functional Requirements

- Users can sign-in using GitHub/Google.
- The signed-in users can view their books.
- Users can search for any book.
- Only signed-in users can add books to their shelves and add a starting and finished date to the book.
- Give a rating to the book from 1 star to 5 stars.
- Add a tag to the book like – “currently reading”, “finished” or “want to read”,

5.2 Schema

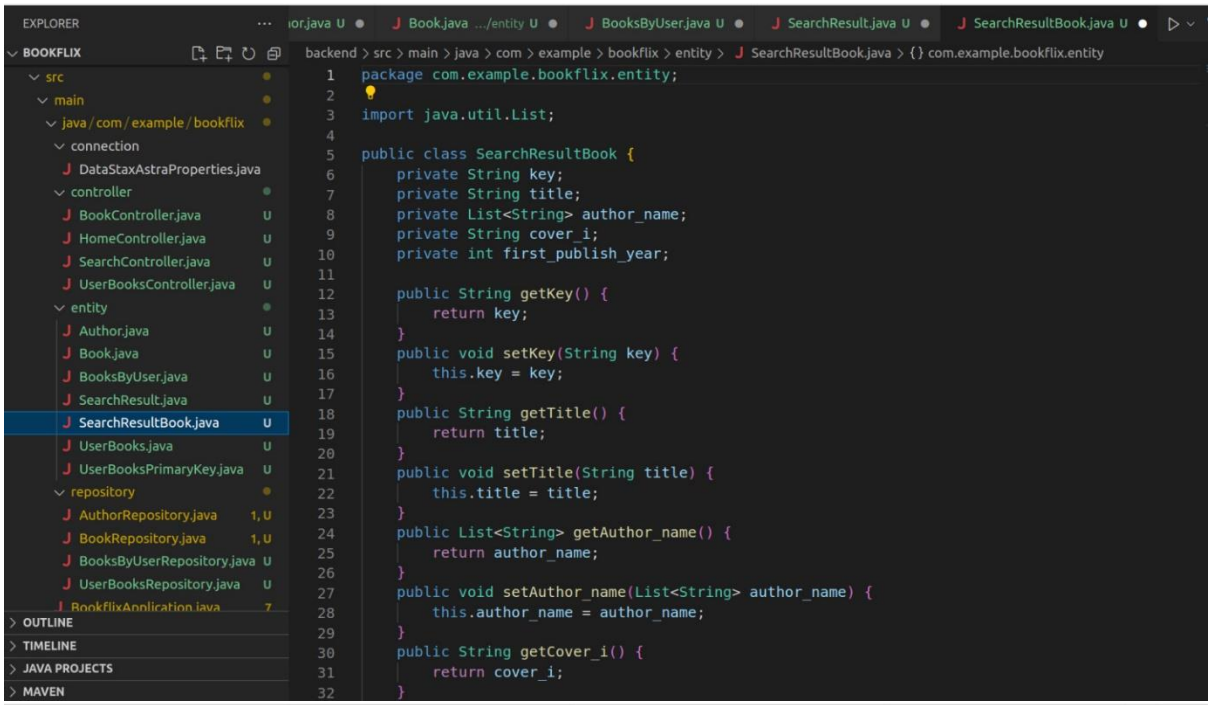

name
id
published_date
Book
1
Written by
n
Author
id
name
covers
desccription
n
start_date
read
status
completed_date
rating
top
m
id
user
name

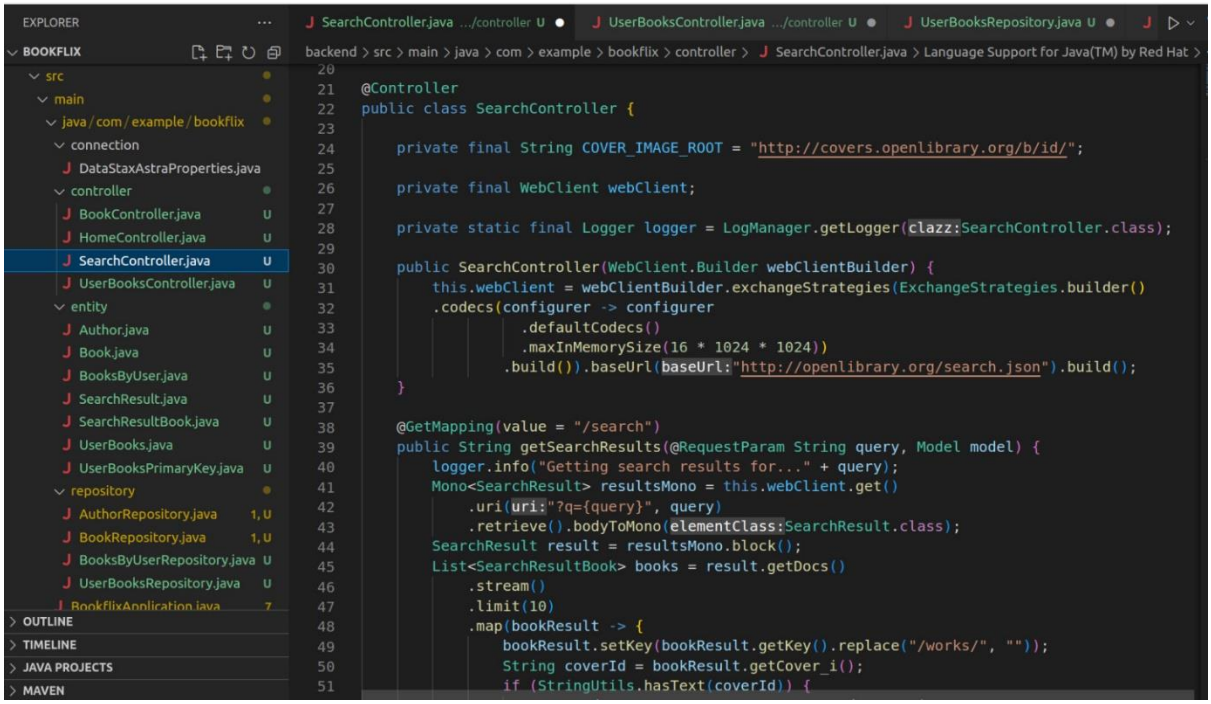
5.3 API Documentation

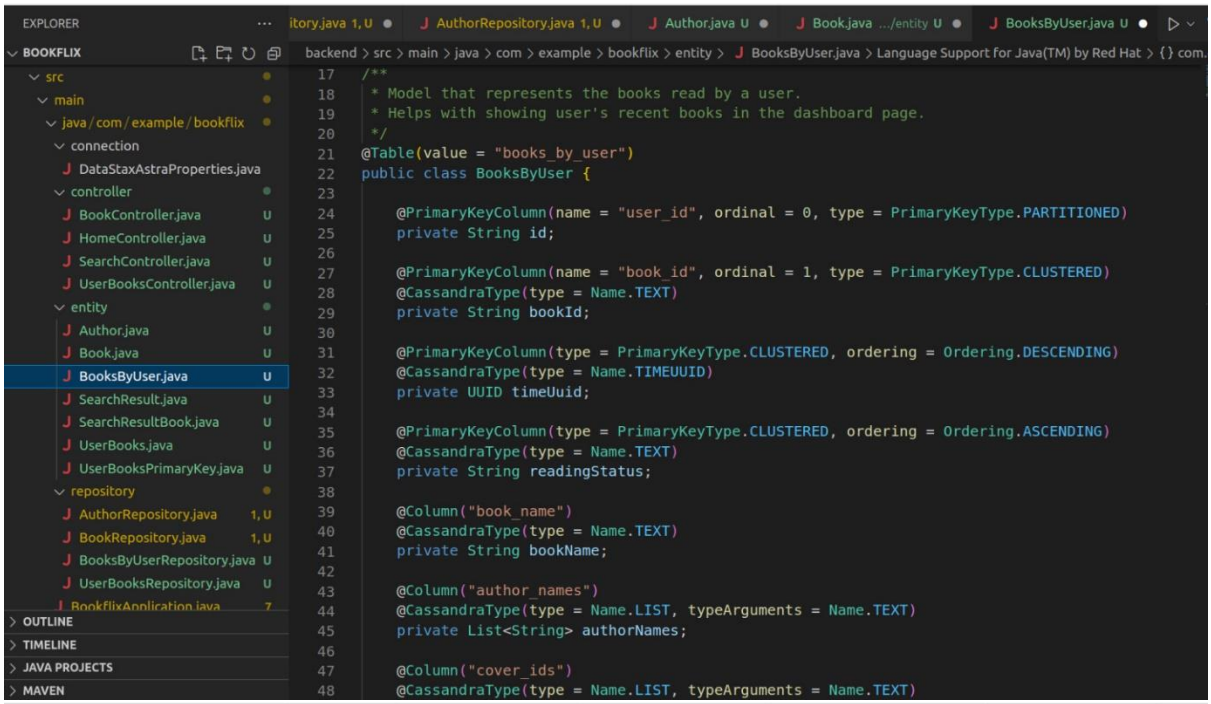
S. No.	Use Case	API	Request Method
1.	Get books of a specific user	/books/{bookId}	GET
2.	Load home page	/	GET
3.	Search for books	/search	GET
4.	Add books by user	/addUserBook	POST

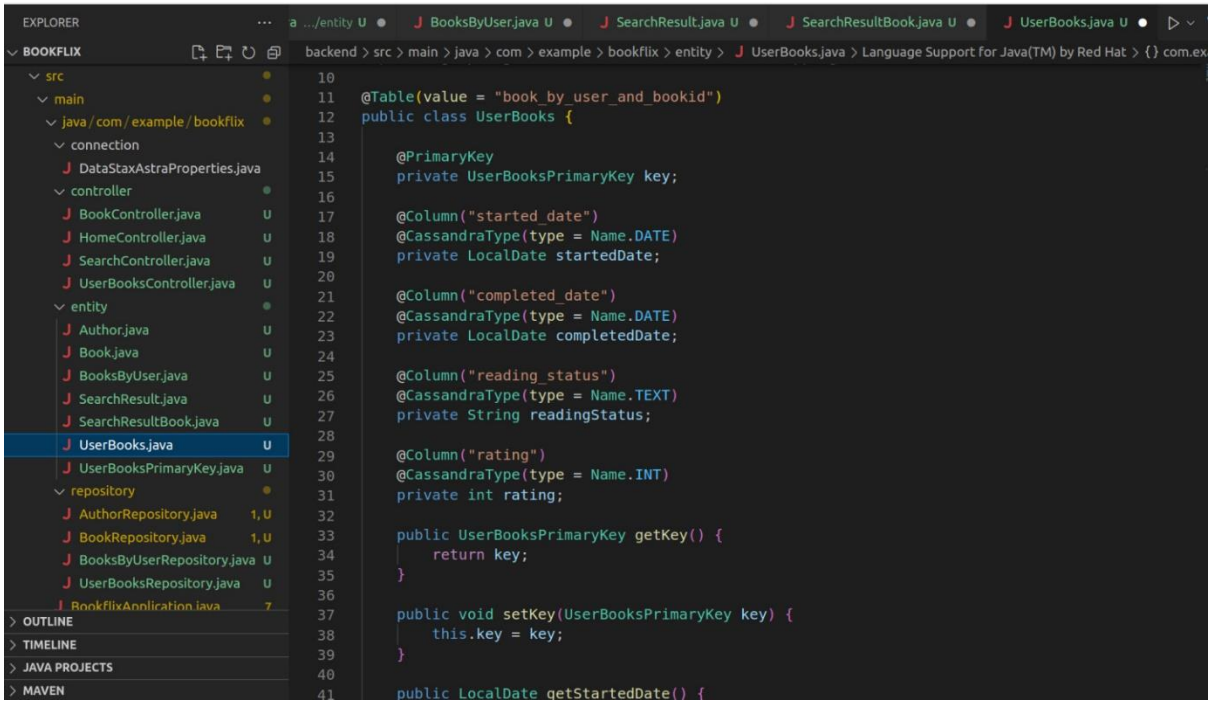
5.4 Code Walkthrough

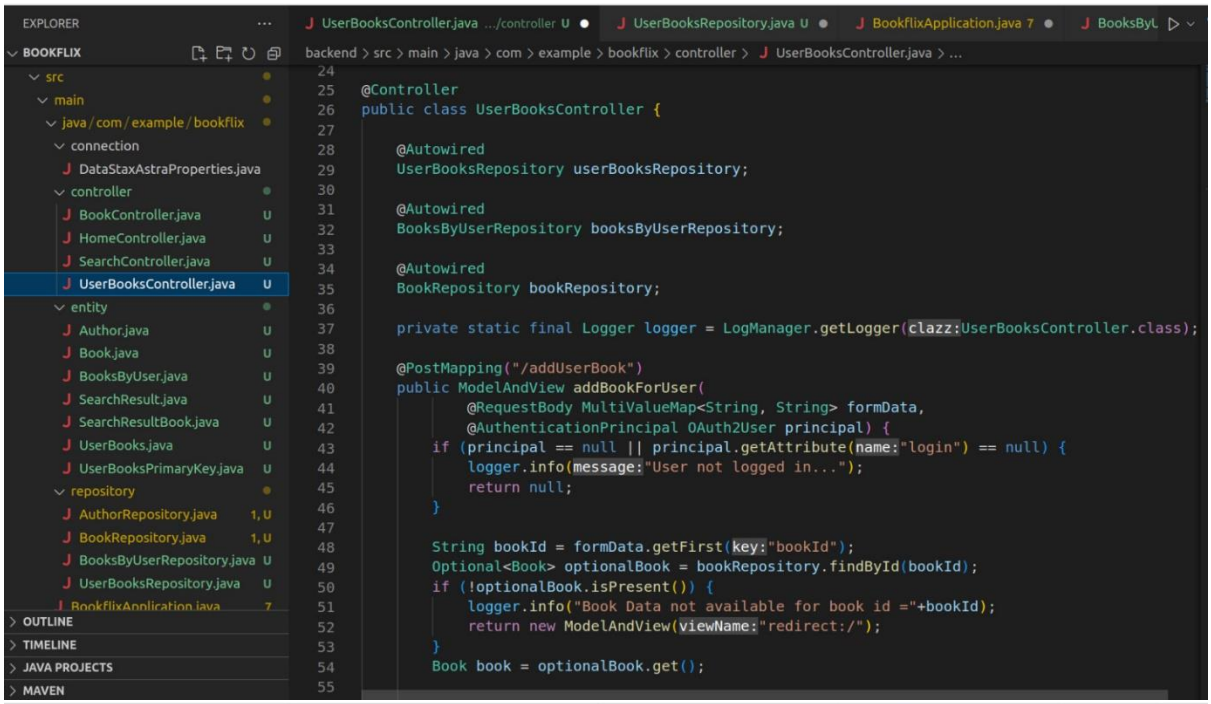
Below code shows the modules that we have used in the front-end and components that we have created to drive our app.

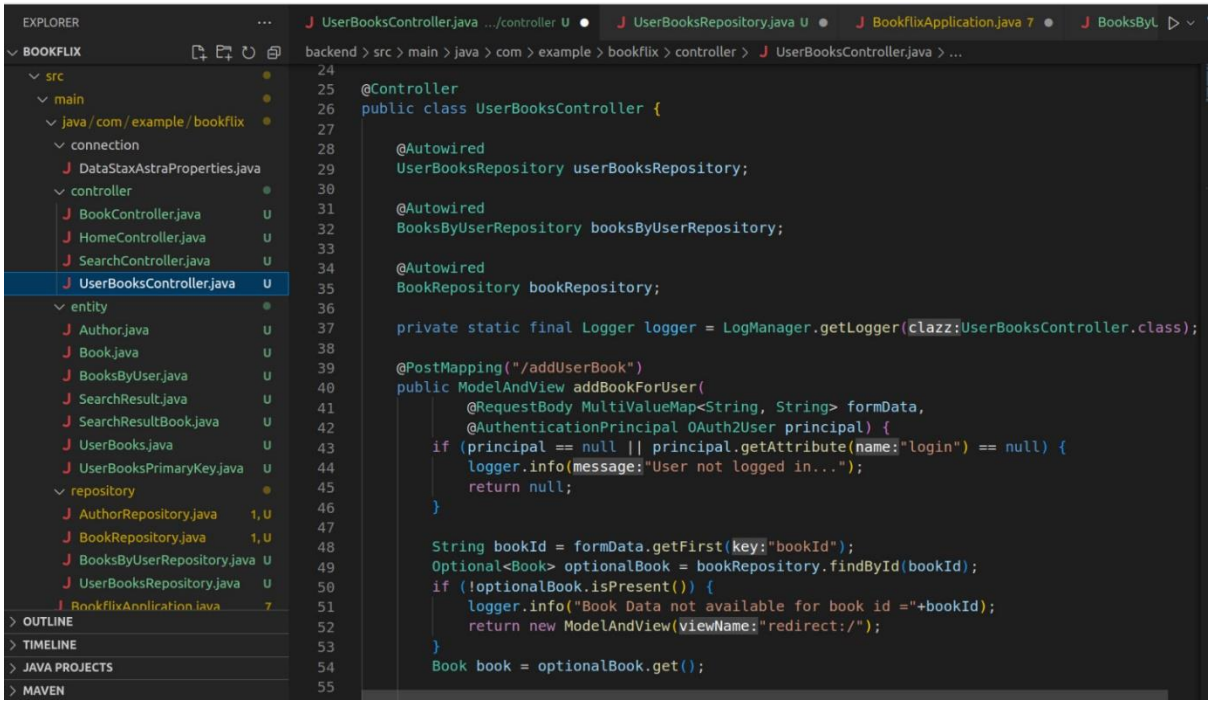


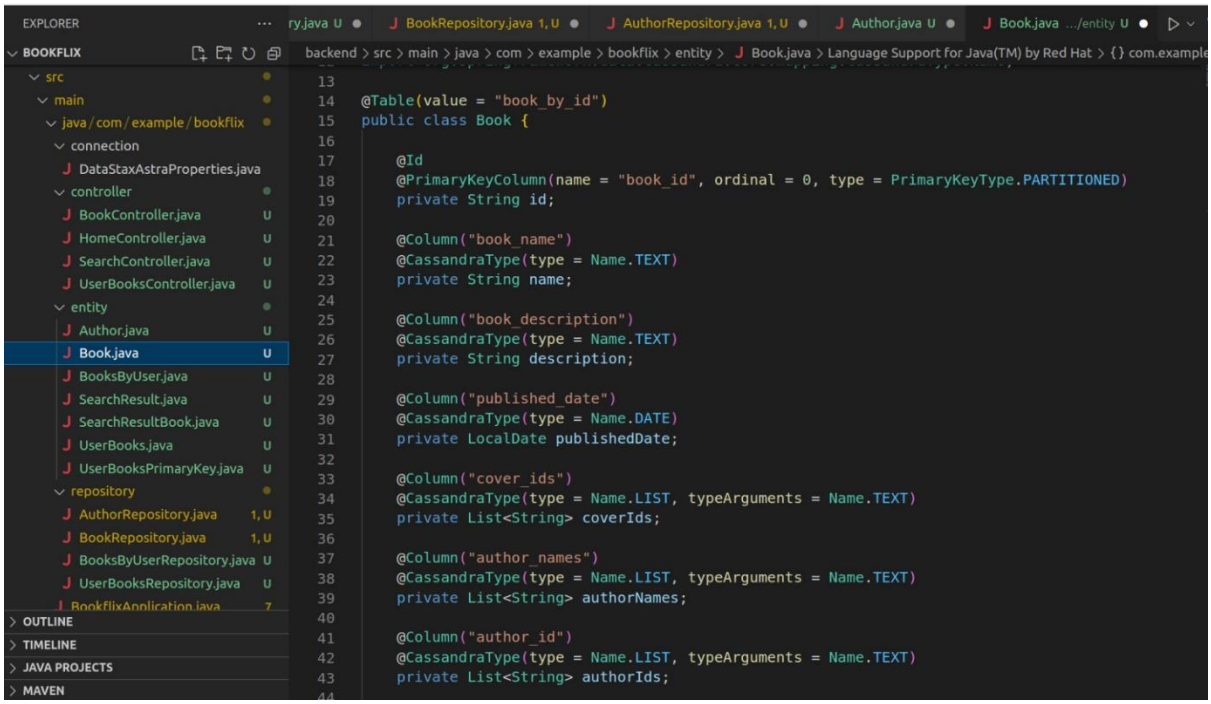



EXPLORER
BOOKFLIX
src
main
java / com / example / bookflix
connection
DataStaxAstraProperties.java
controller
BookController.java
HomeController.java
SearchController.java
UserBooksController.java
entity
Author.java
Book.java
BooksByUser.java
SearchResult.java
SearchResultBook.java
UserBooks.java
UserBooksPrimaryKey.java
repository
AuthorRepository.java
BookRepository.java
BooksByUserRepository.java
UserBooksRepository.java
OUTLINE
TIMELINE
JAVA PROJECTS
MAVEN
AuthorRepository.java 1, U
Author.java U
Book.java .../entity U
BooksByUser.java U
backend > src > main > java > com > example > bookflix > entity > BooksByUser.java > Language Support for Java(TM) by Red Hat > { } com.
/**
* Model that represents the books read by a user.
* Helps with showing user's recent books in the dashboard page.
*/
@Table(value = "books_by_user")
public class BooksByUser {
@PrimaryKeyColumn(name = "user_id", ordinal = 0, type = PrimaryKeyType.PARTITIONED)
private String id;
@PrimaryKeyColumn(name = "book_id", ordinal = 1, type = PrimaryKeyType.CLUSTERED)
@CassandraType(type = Name.TEXT)
private String bookId;
@PrimaryKeyColumn(type = PrimaryKeyType.CLUSTERED, ordering = Ordering.DESCENDING)
@CassandraType(type = Name.TIMEUUID)
private UUID timeUuid;
@PrimaryKeyColumn(type = PrimaryKeyType.CLUSTERED, ordering = Ordering.ASCENDING)
@CassandraType(type = Name.TEXT)
private String readingStatus;
@Column("book_name")
@CassandraType(type = Name.TEXT)
private String bookName;
@Column("author_names")
@CassandraType(type = Name.LIST, typeArguments = Name.TEXT)
private List<String> authorNames;
@Column("cover_ids")
@CassandraType(type = Name.LIST, typeArguments = Name.TEXT)

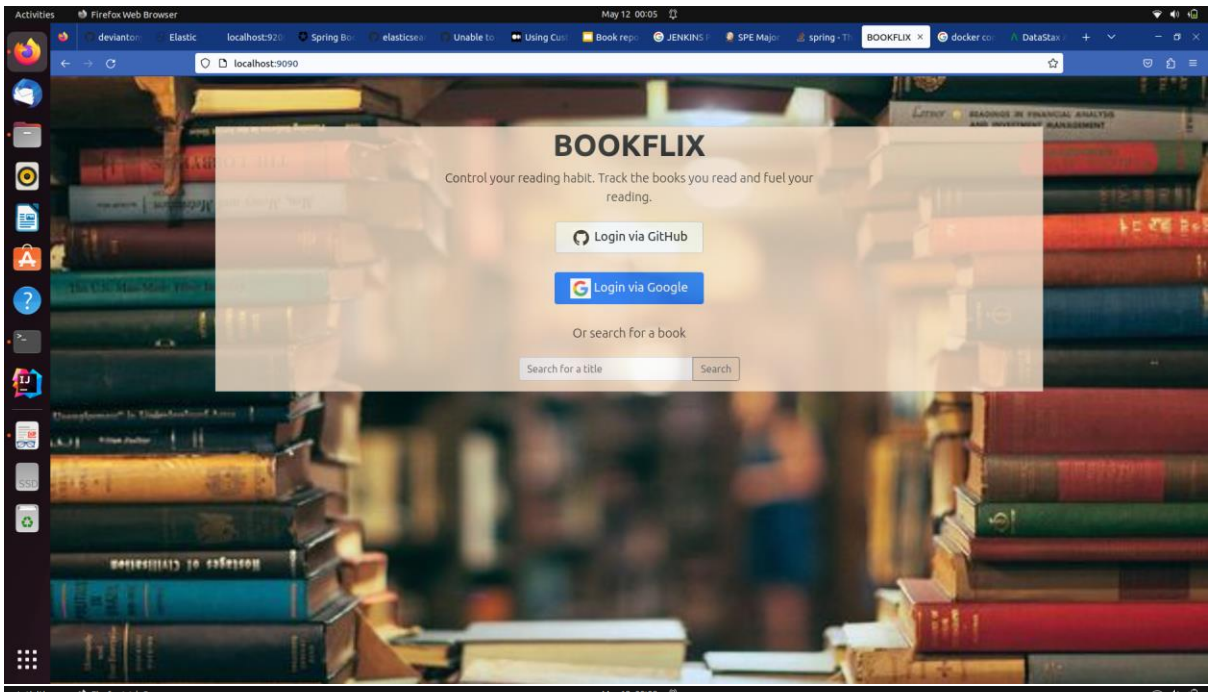

EXPLORER
BOOKFLIX
src
main
java / com / example / bookflix
connection
DataStaxAstraProperties.java
controller
BookController.java
HomeController.java
SearchController.java
UserBooksController.java
entity
Author.java
Book.java
BooksByUser.java
SearchResult.java
SearchResultBook.java
UserBooks.java
UserBooksPrimaryKey.java
repository
AuthorRepository.java
BookRepository.java
BooksByUserRepository.java
UserBooksRepository.java
OUTLINE
TIMELINE
JAVA PROJECTS
MAVEN
BooksByUser.java U
SearchResult.java U
SearchResultBook.java U
UserBooks.java U
backend > src > main > java > com > example > bookflix > entity > UserBooks.java > Language Support for Java(TM) by Red Hat > { } com.ex
@Table(value = "book_by_user_and_bookid")
public class UserBooks {
@PrimaryKey
private UserBooksPrimaryKey key;
@Column("started_date")
@CassandraType(type = Name.DATE)
private LocalDate startedDate;
@Column("completed_date")
@CassandraType(type = Name.DATE)
private LocalDate completedDate;
@Column("reading_status")
@CassandraType(type = Name.TEXT)
private String readingStatus;
@Column("rating")
@CassandraType(type = Name.INT)
private int rating;
public UserBooksPrimaryKey getKey() {
return key;
}
public void setKey(UserBooksPrimaryKey key) {
this.key = key;
}
public LocalDate getStartedDate() {

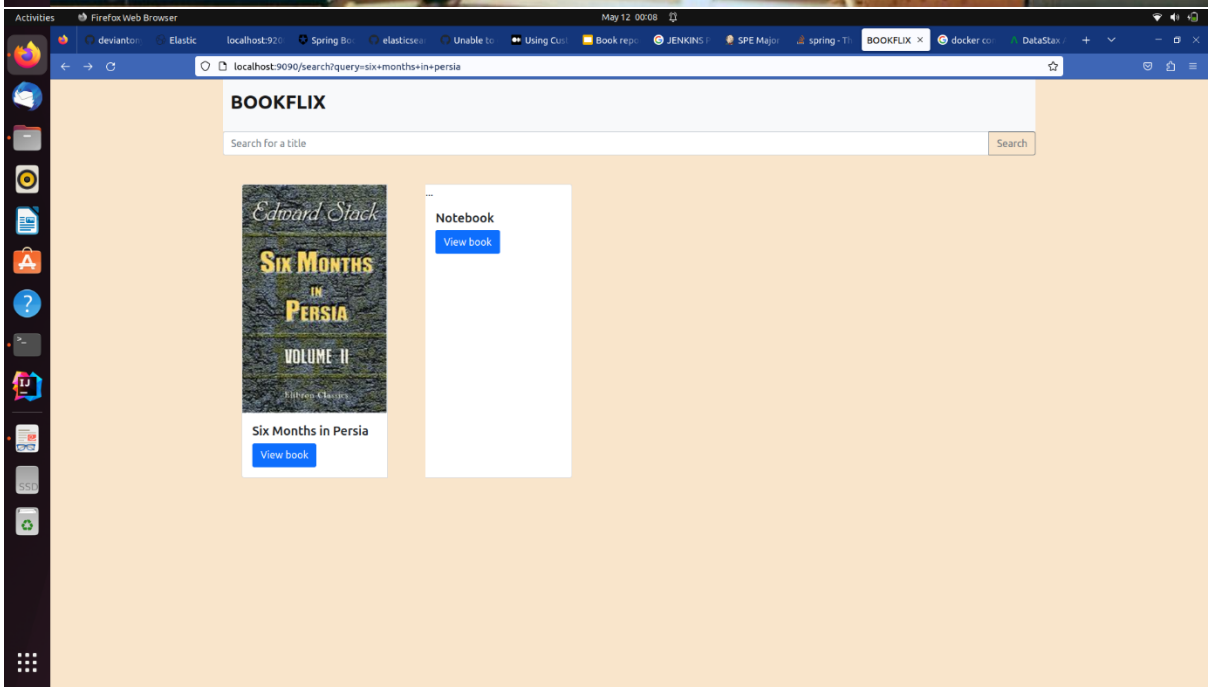

EXPLORER
BOOKFLIX
src
main
java / com / example / bookflix
connection
DataStaxAstraProperties.java
controller
BookController.java U
HomeController.java U
SearchController.java U
UserBooksController.java U
entity
Author.java U
Book.java U
BooksByUser.java U
SearchResult.java U
SearchResultBook.java U
UserBooks.java U
UserBooksPrimaryKey.java U
repository
AuthorRepository.java 1, U
BookRepository.java 1, U
BooksByUserRepository.java U
UserBooksRepository.java U
BookflixApplication.java
OUTLINE
TIMELINE
JAVA PROJECTS
MAVEN
UserBooksController.java .../controller U
UserBooksRepository.java U
BookflixApplication.java 7
backend > src > main > java > com > example > bookflix > controller > UserBooksController.java > ...
@Controller
public class UserBooksController {
@Autowired
UserBooksRepository userBooksRepository;
@Autowired
BooksByUserRepository booksByUserRepository;
@Autowired
BookRepository bookRepository;
private static final Logger logger = LogManager.getLogger(clazz:UserBooksController.class);
@PostMapping("/addUserBook")
public ModelAndView addBookForUser(
@RequestBody MultiValueMap<String, String> formData,
@AuthenticationPrincipal OAuth2User principal) {
if (principal == null || principal.getAttribute(name:"login") == null) {
logger.info(message:"User not logged in...");
return null;
}
String bookId = formData.getFirst(key:"bookId");
Optional<Book> optionalBook = bookRepository.findById(bookId);
if (!optionalBook.isPresent()) {
logger.info("Book Data not available for book id ="+bookId);
return new ModelAndView(viewName:"redirect:/");
}
Book book = optionalBook.get();

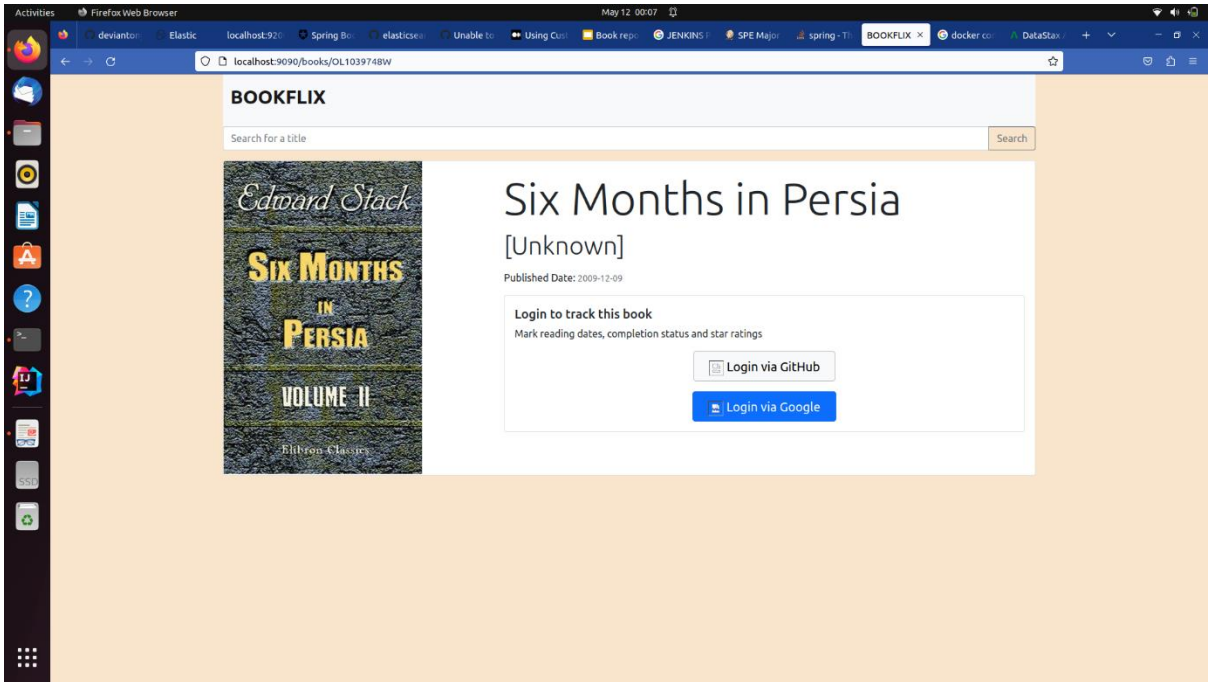

EXPLORER
BOOKFLIX
src
main
java / com / example / bookflix
connection
DataStaxAstraProperties.java
controller
BookController.java U
HomeController.java U
SearchController.java U
UserBooksController.java U
entity
Author.java U
Book.java U
BooksByUser.java U
SearchResult.java U
SearchResultBook.java U
UserBooks.java U
UserBooksPrimaryKey.java U
repository
AuthorRepository.java 1, U
BookRepository.java 1, U
BooksByUserRepository.java U
UserBooksRepository.java U
BookflixApplication.java
OUTLINE
TIMELINE
JAVA PROJECTS
MAVEN
UserBooksController.java .../controller U
UserBooksRepository.java U
BookflixApplication.java 7
backend > src > main > java > com > example > bookflix > controller > UserBooksController.java > ...
@Controller
public class UserBooksController {
@Autowired
UserBooksRepository userBooksRepository;
@Autowired
BooksByUserRepository booksByUserRepository;
@Autowired
BookRepository bookRepository;
private static final Logger logger = LogManager.getLogger(clazz:UserBooksController.class);
@PostMapping("/addUserBook")
public ModelAndView addBookForUser(
@RequestBody MultiValueMap<String, String> formData,
@AuthenticationPrincipal OAuth2User principal) {
if (principal == null || principal.getAttribute(name:"login") == null) {
logger.info(message:"User not logged in...");
return null;
}
String bookId = formData.getFirst(key:"bookId");
Optional<Book> optionalBook = bookRepository.findById(bookId);
if (!optionalBook.isPresent()) {
logger.info("Book Data not available for book id ="+bookId);
return new ModelAndView(viewName:"redirect:/");
}
Book book = optionalBook.get();

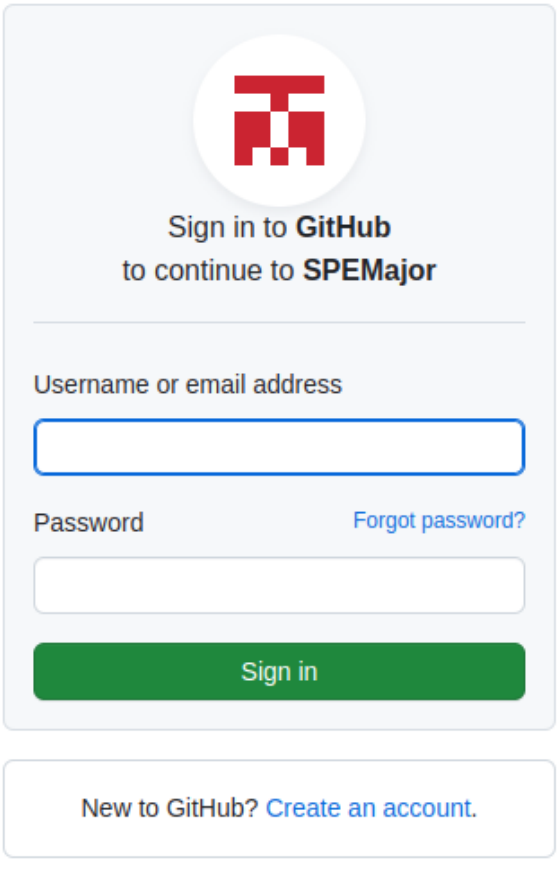

EXPLORER
BOOKFLIX
src
main
java / com / example / bookflix
connection
DataStaxAstraProperties.java
controller
BookController.java
HomeController.java
SearchController.java
UserBooksController.java
entity
Author.java
Book.java
BooksByUser.java
SearchResult.java
SearchResultBook.java
UserBooks.java
UserBooksPrimaryKey.java
repository
AuthorRepository.java
BookRepository.java
BooksByUserRepository.java
UserBooksRepository.java
OUTLINE
TIMELINE
JAVA PROJECTS
MAVEN
BookRepository.java 1, U
AuthorRepository.java 1, U
Author.java U
Book.java .../entity U
backend > src > main > java > com > example > bookflix > entity > Book.java > Language Support for Java(TM) by Red Hat > { } com.example
@Table(value = "book_by_id")
public class Book {
@Id
@PrimaryKeyColumn(name = "book_id", ordinal = 0, type = PrimaryKeyType.PARTITIONED)
private String id;
@Column("book_name")
@CassandraType(type = Name.TEXT)
private String name;
@Column("book_description")
@CassandraType(type = Name.TEXT)
private String description;
@Column("published_date")
@CassandraType(type = Name.DATE)
private LocalDate publishedDate;
@Column("cover_ids")
@CassandraType(type = Name.LIST, typeArguments = Name.TEXT)
private List<String> coverIds;
@Column("author_names")
@CassandraType(type = Name.LIST, typeArguments = Name.TEXT)
private List<String> authorNames;
@Column("author_id")
@CassandraType(type = Name.LIST, typeArguments = Name.TEXT)
private List<String> authorIds;

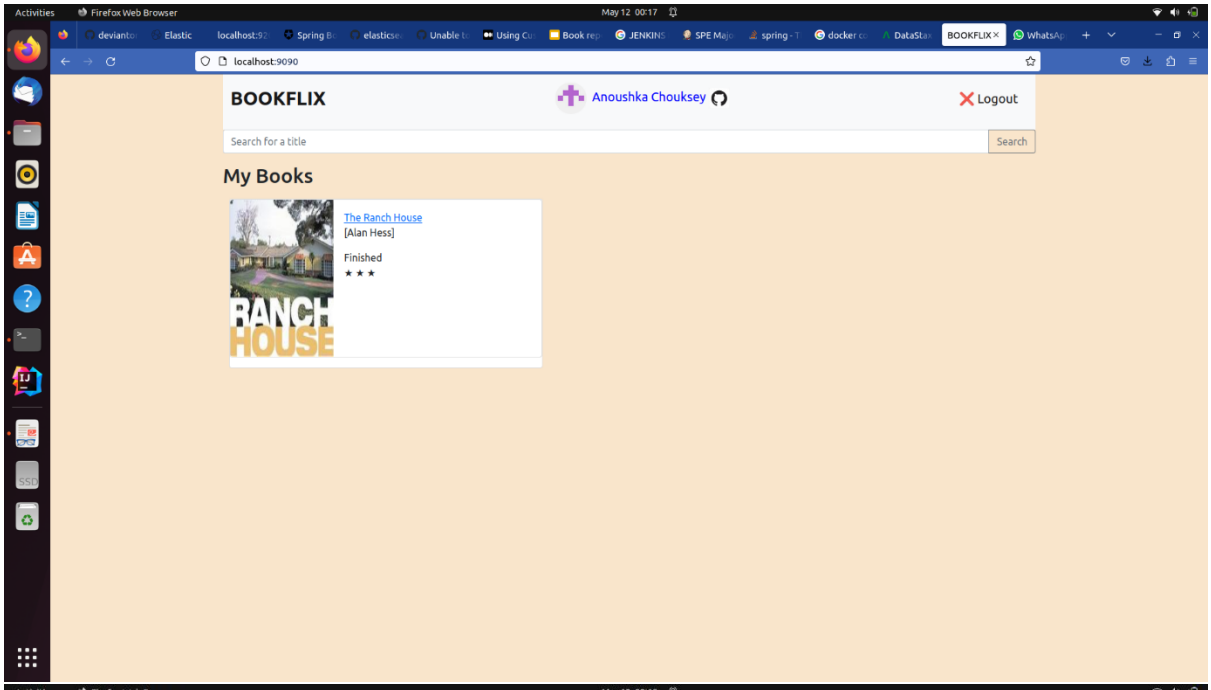
6. Results and Discussion

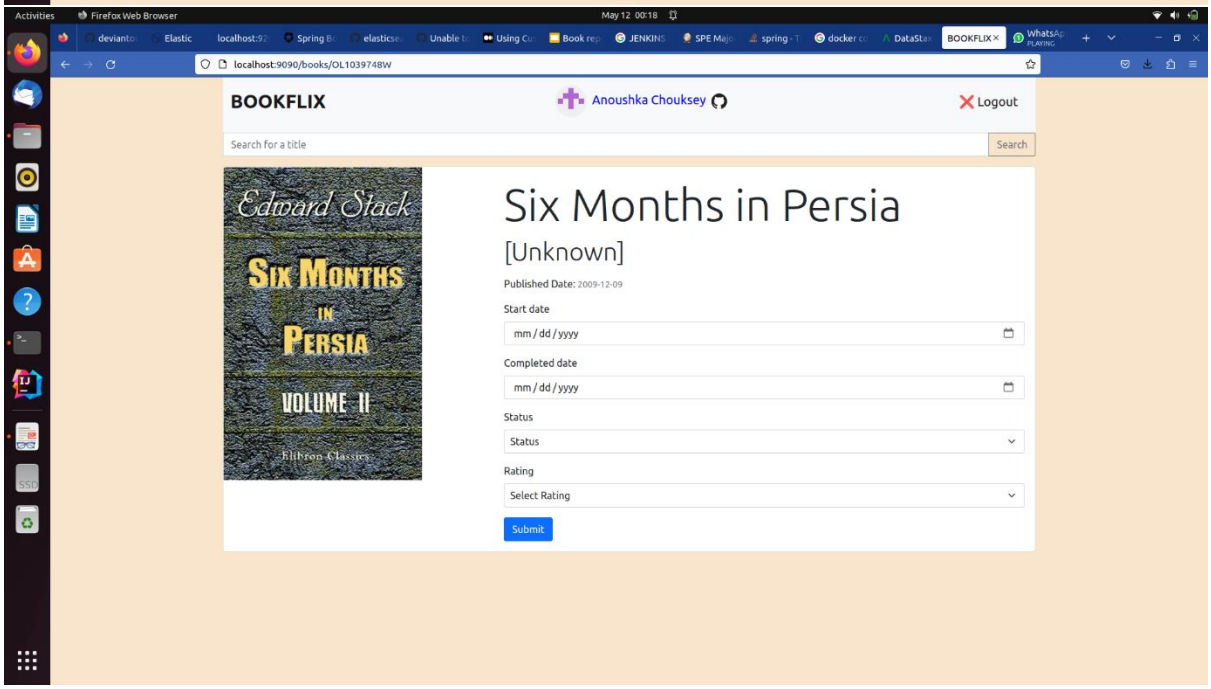





Activities
Firefox Web Browser
May 12 00:07
localhost:9090/books/OL1039748W
BOOKFLIX
Search for a title
Search
Edward Stack
SIX MONTHS IN PERSIA
VOLUME II
Six Months in Persia
[Unknown]
Published Date: 2009-12-09
Login to track this book
Mark reading dates, completion status and star ratings
Login via GitHub
Login via Google


Sign in to GitHub
to continue to SPEMajor
Username or email address
Password
Forgot password?
Sign in
New to GitHub? Create an account.


Activities
Firefox Web Browser
May 12 00:17
localhost:9090
BOOKFLIX
Anoushka Chouksey
Logout
Search for a title
Search
My Books
The Ranch House
[Alan Hess]
Finished
★ ★ ★


Activities
Firefox Web Browser
May 12 00:18
localhost:9090/books/OL1039748W
BOOKFLIX
Anoushka Chouksey
Logout
Search for a title
Search
Six Months in Persia
[Unknown]
Published Date: 2009-12-09
Start date
mm / dd / yyyy
Completed date
mm / dd / yyyy
Status
Status
Rating
Select Rating
Submit

7. Scope for Future Work

- Integrate with social media sites so that friends can follow each other's activity.
- Review a book/add a summary.
- Suggest books based on user's past read books.
- Reminder feature to inculcate the habit of consistent reading

8. Conclusion

We successfully built the web application BookFlix that can be by book nerds to keep track of their favourite books. This application can prove to be very useful to instil the habit of reading.

The project taught us a great deal about all the DevOps tools that we used for our projects. Through usage of these tools, we understood the importance of automation in software development. We learnt about technologies like HTML and Thymeleaf which was used to make the front-end of our website, SpringBoot to make the back-end, SCM tool of Git, CI/CD tools like Jenkins, Deployment platform like Docker, and CM tool like ELK. These tools diminished our work impressively.

9. References

https://www.freecodecamp.org/news/the-apache-cassandra-beginner-tutorial/

https://youtu.be/35EQXmHKZYs

https://youtu.be/nP7O26fFkjI

https://youtu.be/cC4GGJ0JsSE