

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.

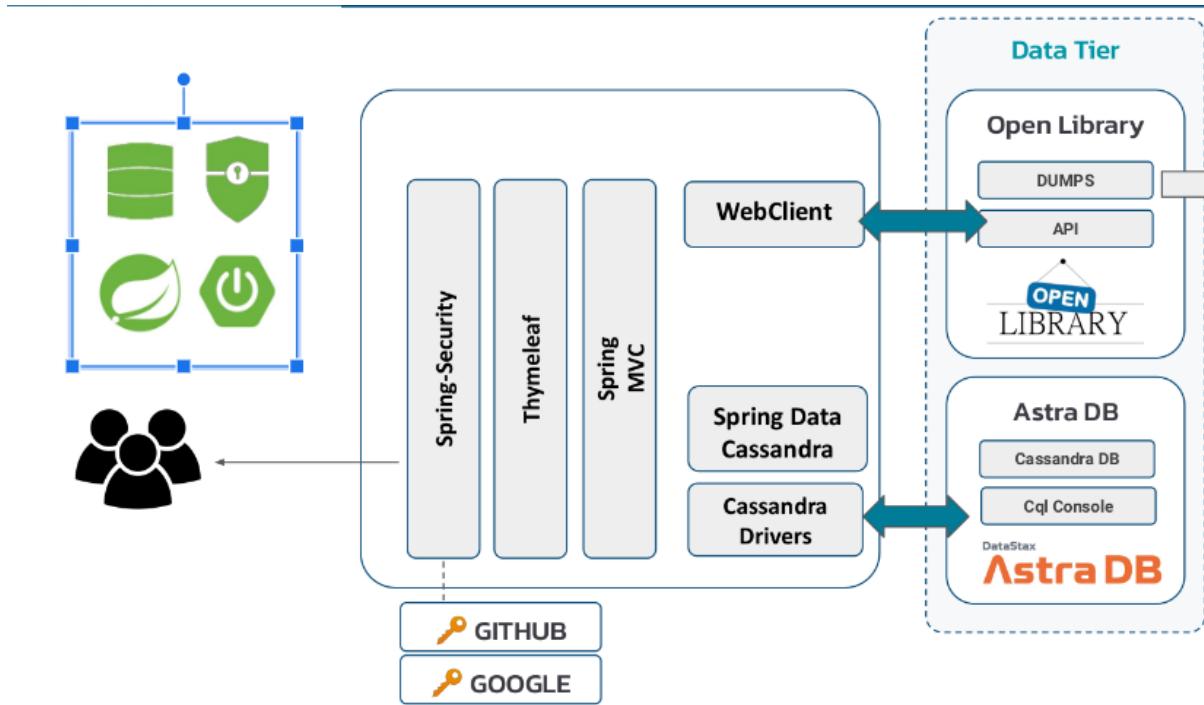
The architecture of our project demands three layers.

- Front end
- Middle layer
- Back end

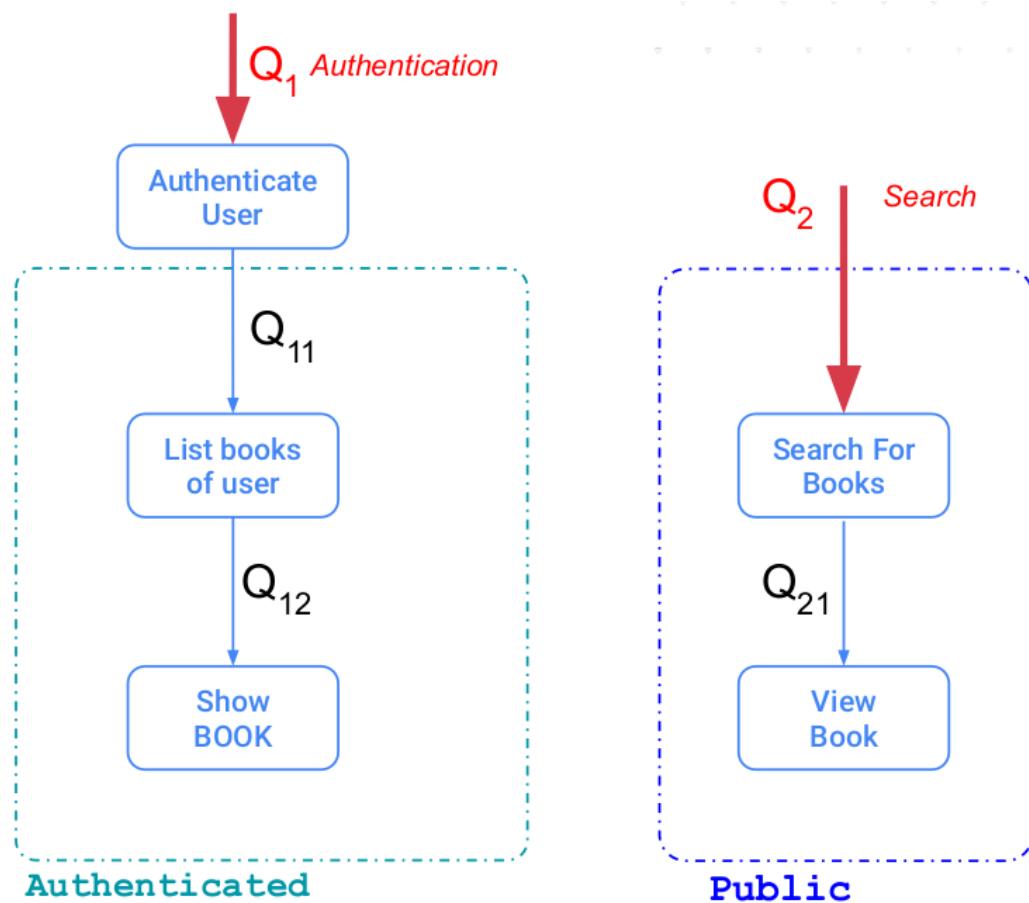
The front end of the project is handled by HTML and Thymeleaf framework. The middle layer is built on SpringBoot Framework and communicates with Cassandra database to show the content on the front end via Rest API.

2. Introduction

2.1 Overview



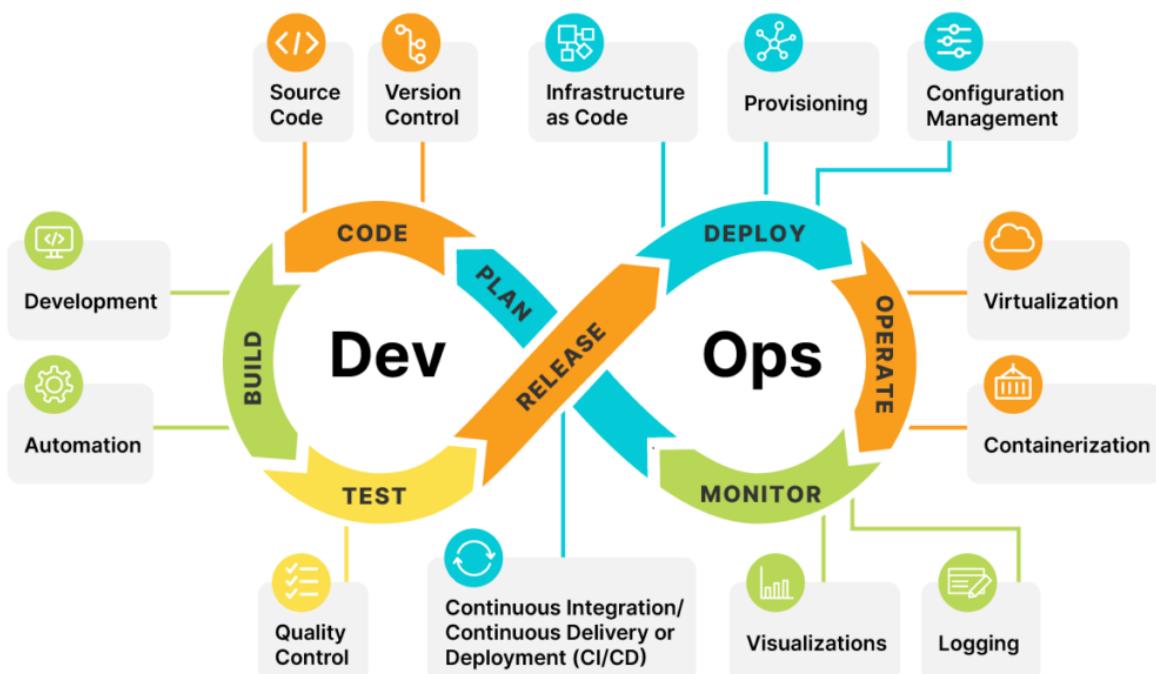
2.2 Features



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)

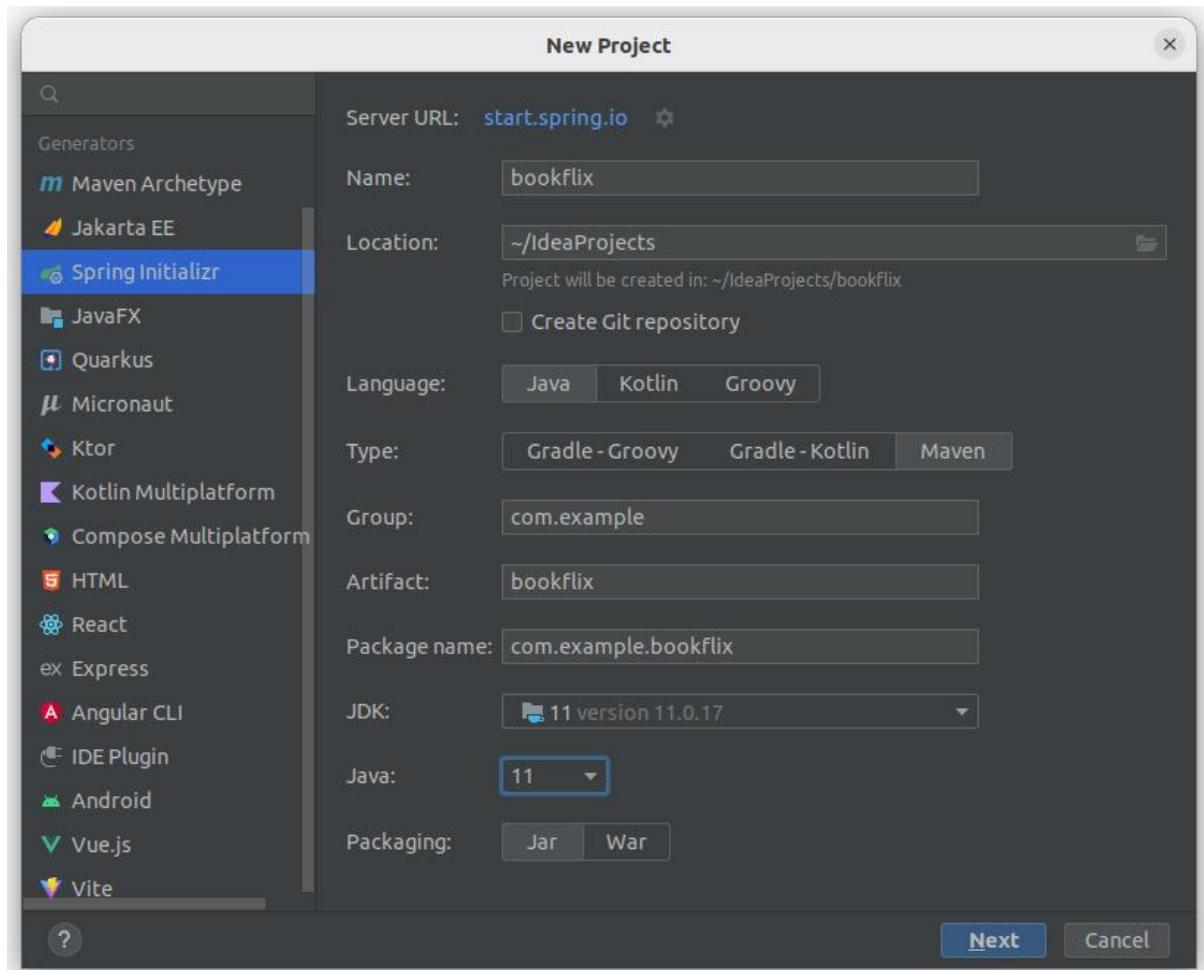
4. Software Development Life Cycle

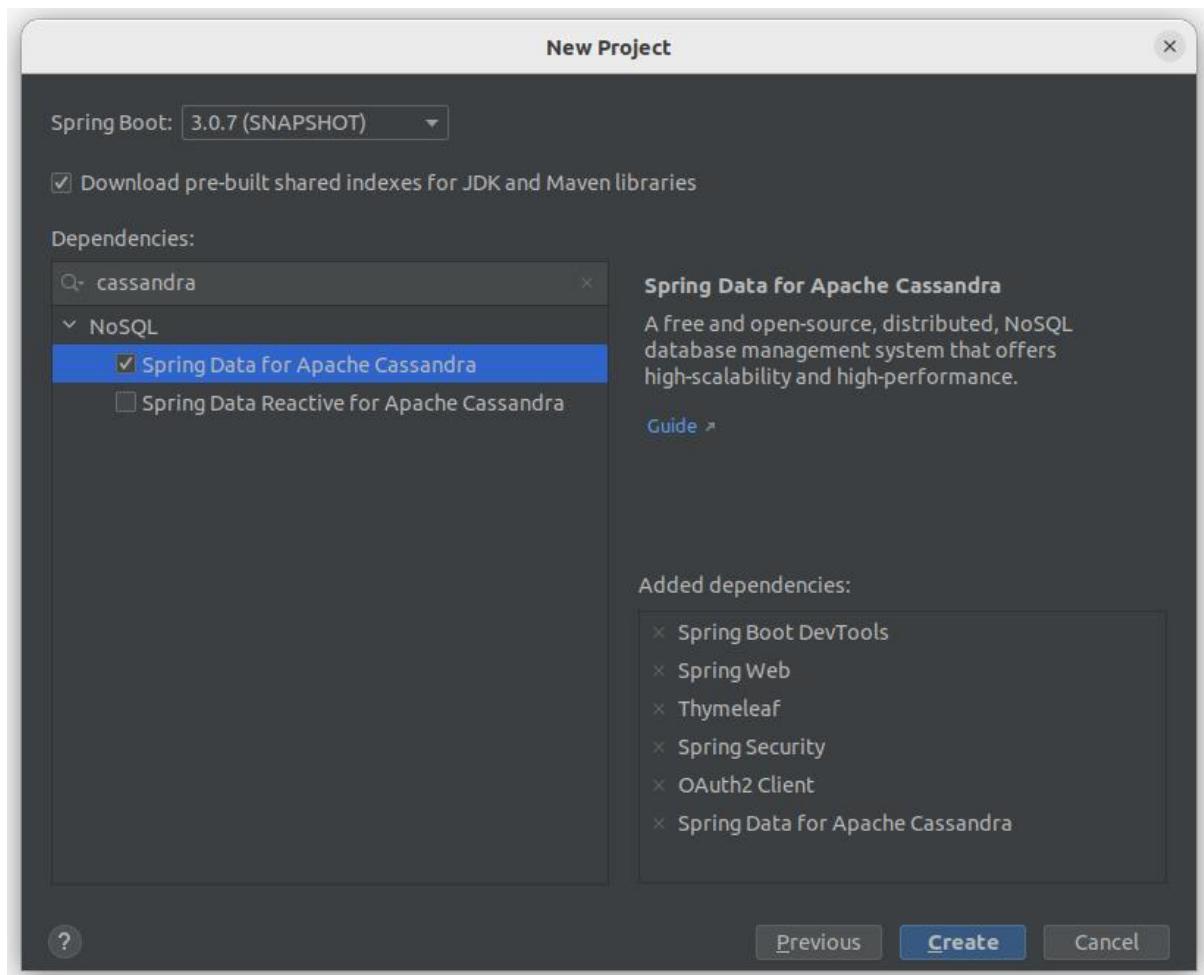
Software development life cycle is a systematic process for building software that ensures the quality and correctness of the software built. SDLC process aims to produce high-quality software that meets customer expectations. The system development should be complete in the pre-defined time frame and 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.

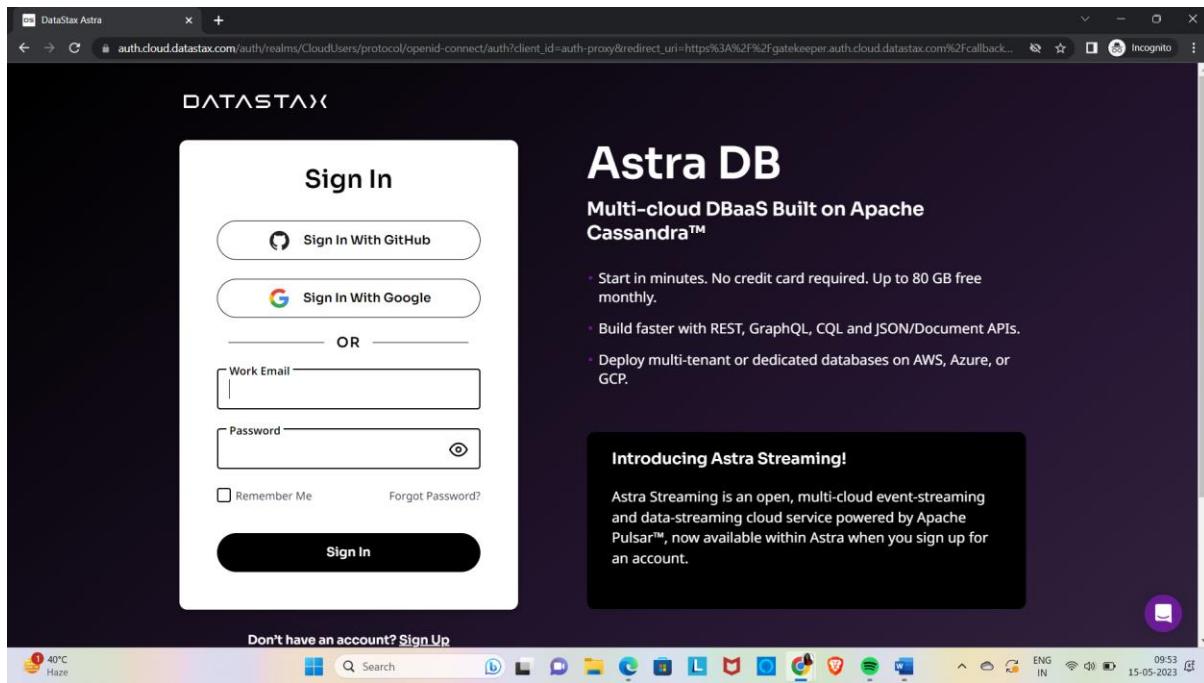




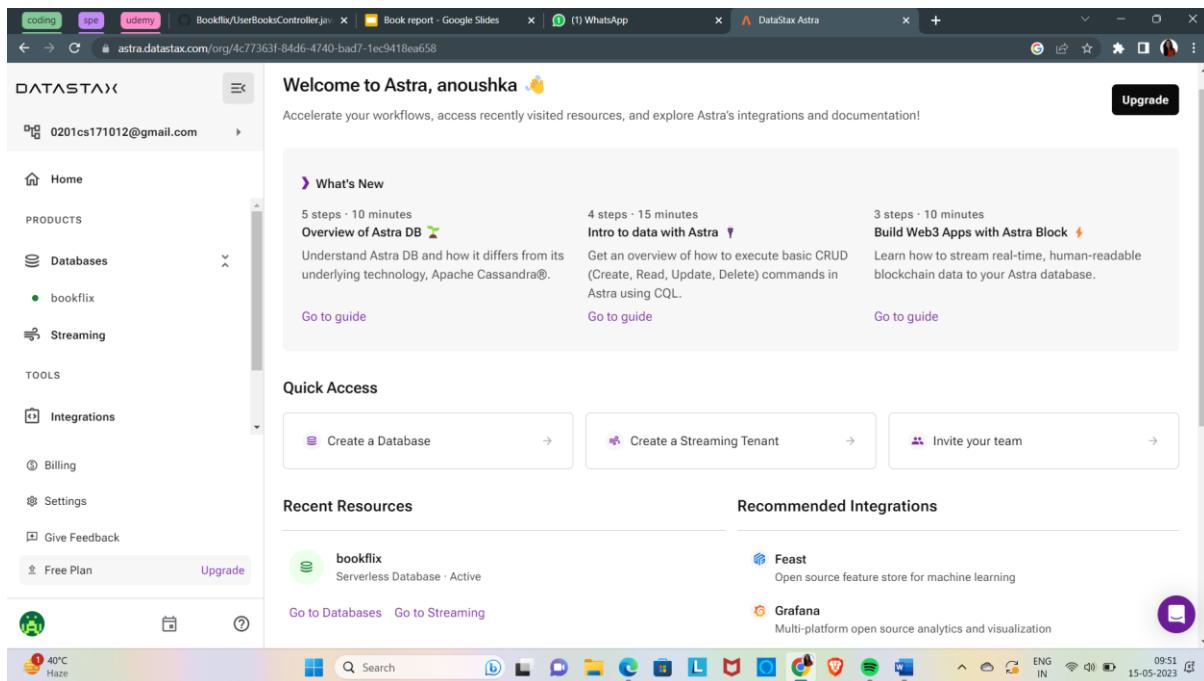
4.1.2 Cassandra

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)u00e9tais, allemand, anglais, espagnol", "covers": [3140607], "key": "/works/OL10000355W", "authors": [{"type": "type/author_role"}, {"author": {"key": "/authors/OL3965376A"}]}, {"type": "/type/work"}, {"subjects": ["Comparative and general Grammar", "Verb", "Tense", "French language", "German language", "English language", "Spanish language"], "latest_revision": 4, "revision": 4, "created": "/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\u00e7eene", "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/OL3965376A"}]}, {"type": "/type/work"}, {"revision": 3}
3 /type/work   /works/OL10000514W   3   2010-04-28T06:54:19.472104   {"title": "MIR", "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": "/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": "type/author_role"}, {"author": {"key": "/authors/OL3965920A"}]}, {"type": "/type/work"}, {"subjects": ["Exhibitions", "Constructivism (Art)", "Art", "Centre Georges Pompidou"], "latest_revision": 4, "revision": 4, "created": "/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": "/type/work"}, {"revision": 3}
5 /type/work   /works/OL10000821W   3   2010-04-28T06:54:19.472104   {"title": "L'art de la photographie dans les mus\u00e9es", "covers": [3141623], "key": "/works/OL10000821W", "authors": [{"type": "type/author_role"}, {"author": {"key": "/authors/OL3965554A"}]}, {"type": "/type/work"}, {"revision": 3}
6 /type/work   /works/OL10001746W   3   2010-04-28T06:54:19.472104   {"title": "R\u00e9alisez le commentaire de texte", "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": "/type/work"}, {"revision": 3}
7 /type/work   /works/OL10001805W   3   2010-04-28T06:54:19.472104   {"title": "Sardaigne 2002", "covers": [3144175], "last_modified": {"type": "/type/datetime", "value": "2010-04-28T06:54:19.472104"}, "latest_revision": 3, "key": "/works/OL10001805W", "authors": [{"type": "type/author_role"}, {"author": {"key": "/authors/OL3967107A"}]}, {"type": "/type/work"}, {"revision": 3}
8 /type/work   /works/OL10002168W   3   2010-04-28T06:54:19.472104   {"title": "Manuel mu00e9thodique de pr\u00e9paration. Examens et concours administratifs, concours d'entr\u00e9e en Institut de formation des cadres de sant\u00e9 (u00e9eep)", "covers": [3144806], "last_modified": {"type": "/type/datetime", "value": "2009-12-11T01:57:38.254267"}, "covers": [3144806], "last_revision": 3, "key": "/works/OL10002168W", "authors": [{"type": "type/author_role"}, {"author": {"key": "/authors/OL3967457A"}]}, {"type": "/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", "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/OL3966691A"}]}, {"type": "/type/work"}, {"revision": 3}
10 /type/work   /works/OL10002236W   3   2010-04-28T06:54:19.472104   {"title": "R\u00e9alisez l'DPAP 2002", "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/OL3966927A"}]}, {"type": "/type/work"}, {"revision": 3}
11 /type/work   /works/OL10002277W   3   2010-04-28T06:54:19.472104   {"title": "Ouvrez l'u00f4rancne pour prier", "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": "/type/work"}, {"revision": 3}
12 /type/work   /works/OL10002825W   3   2014-05-24T07:57:52.484771   {"title": "Fragmentarium", "covers": [3146154], "last_modified": {"type": "/type/datetime", "value": "2014-05-24T07:57:52.484771"}, "latest_revision": 4, "key": "/works/OL10002825W", "authors": [{"type": "type/author_role"}, {"author": {"key": "/authors/OL3966686A"}]}, {"type": "/type/work"}, {"revision": 4}
13 /type/work   /works/OL10002844W   3   2009-12-11T01:57:38.254267   {"title": "L'Annuel Des Arts 1998", "covers": [3146455], "last_modified": {"type": "/type/datetime", "value": "2009-12-11T01:57:38.254267"}, "latest_revision": 3, "key": "/works/OL10002844W", "authors": [{"type": "type/author_role"}, {"author": {"key": "/authors/OL3962858A"}]}, {"type": "/type/work"}, {"revision": 3}
14 /type/work   /works/OL10003095W   3   2010-04-28T06:54:19.472104   {"title": "Th\u00e0u00e9se, infusions, sant\u00e8s", "covers": [3146545], "last_modified": {"type": "/type/datetime", "value": "2009-12-11T01:57:38.254267"}, "covers": [3146545], "last_revision": 3, "key": "/works/OL10003095W", "authors": [{"type": "type/author_role"}, {"author": {"key": "/authors/OL3966684A"}]}, {"type": "/type/work"}, {"revision": 3}
15 /type/work   /works/OL10003205W   3   2010-04-28T06:54:19.472104   {"title": "M\u00e9thode de pilotage de montgolfi\u00e8re u00e9e", "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/OL3966684A"}]}, {"type": "/type/work"}, {"revision": 3}
16 /type/work   /works/OL10003382W   3   2010-04-28T06:54:19.472104   {"title": "Le Tous le monde en famille", "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/OL3965554A"}]}, {"type": "/type/work"}, {"revision": 3}
17 /type/work   /works/OL10003418W   3   2010-04-28T06:54:19.472104   {"title": "La r\u00e9volution", "covers": [3146920], "last_modified": {"type": "/type/datetime", "value": "2010-04-28T06:54:19.472104"}, "latest_revision": 3, "key": "/works/OL10003418W", "authors": [{"type": "type/author_role"}, {"author": {"key": "/authors/OL397380"}]}, {"type": "/type/work"}, {"revision": 3}

```

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": "type/author", "revision": 1}
2 /type/author   /authors/OL3971384A   1   2008-04-30T20:50:18.033121   {"name": "Caban\u00e8s", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/authors/OL3971384A", "type": "type/author", "revision": 1}
3 /type/author   /authors/OL3968744A   1   2008-04-30T20:50:18.033121   {"name": "Damenie Louis", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/authors/OL3968744A", "type": "type/author", "revision": 1}
4 /type/author   /authors/OL3968784A   1   2008-04-30T20:50:18.033121   {"name": "Damenie Louis", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/authors/OL3968784A", "type": "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": "type/author", "revision": 1}
6 /type/author   /authors/OL3980494A   1   2008-04-30T20:50:18.033121   {"name": "Alain Renk", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/authors/OL3980494A", "type": "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": "type/author", "revision": 1}
8 /type/author   /authors/OL3967514A   1   2008-04-30T20:50:18.033121   {"name": "Jacqueline Bregetzer", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/authors/OL3967514A", "type": "type/author", "revision": 1}
9 /type/author   /authors/OL3968483A   1   2008-04-30T20:50:18.033121   {"name": "Jacky Loufrani", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/authors/OL3968483A", "type": "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": "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": "type/author", "revision": 1}
12 /type/author   /authors/OL3972162A   1   2008-04-30T20:50:18.033121   {"name": "Bruno Meshaka", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/authors/OL3972162A", "type": "type/author", "revision": 1}
13 /type/author   /authors/OL3965554A   1   2008-04-30T20:50:18.033121   {"name": "D. Cuthbert", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/authors/OL3965554A", "type": "type/author", "revision": 1}
14 /type/author   /authors/OL3977715A   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/OL3977715A", "type": "type/author", "revision": 1}
15 /type/author   /authors/OL3968483A   1   2008-04-30T20:50:18.033121   {"name": "Marie-H\u00e8le Perennec", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/authors/OL3968483A", "type": "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": "type/author", "revision": 1}
17 /type/author   /authors/OL3971678A   1   2008-04-30T20:50:18.033121   {"name": "S. Bolsseller", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/authors/OL3971678A", "type": "type/author", "revision": 1}
18 /type/author   /authors/OL3978814A   1   2008-04-30T20:50:18.033121   {"name": "Albrecht G\u00f4uard", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/authors/OL3978814A", "type": "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": "type/author", "revision": 1}
20 /type/author   /authors/OL3967591A   1   2008-04-30T20:50:18.033121   {"name": "Fr\u00e9d\u00e9ric Luc de Taliz\u00e8re", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/authors/OL3967591A", "type": "type/author", "revision": 1}
21 /type/author   /authors/OL769457A   1   2020-09-30T12:21:52.547947   {"name": "Abraham Grace Merritt", "last_modified": {"type": "/type/datetime", "value": "1994-09-30T12:21:52.547947"}, "key": "/authors/OL769457A", "type": "type/author", "revision": 1}
He also wrote fiction. Including eight novels.: "name": "A. Merritt", "links": [{"url": "http://en.wikipedia.org/wiki/A._Merritt"}, {"key": "/type/link"}, {"title": "Wikpedia page"}], "personal_name": "Abraham Merritt", "wikidata": "http://en.wikipedia.org/wikidata/Wikidata", "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", "type": "type/author", "revision": 11, "remote_ids": [{"viaf": "9850578", "wikidata": "Q338376", "isbn": "0000000114675787"}]
22 /type/author   /authors/OL3973794A   1   2008-04-30T20:50:18.033121   {"name": "Serge Beaulnat", "last_modified": {"type": "/type/datetime", "value": "2008-04-30T20:50:18.033121"}, "key": "/authors/OL3973794A", "type": "type/author", "revision": 1}
23 /type/author   /authors/OL93018A   1   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": "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": "type/author", "revision": 1}

```

```

backend > src > main > java > com > example > bookflix > J 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:"description");
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                             .replaceAll("/authors/", "");
150                         authorIds.add(authorid);
151                     }
152                 }
153             }
154
155             final AtomicInteger currentLine = new AtomicInteger(0);
156             Files.lines(Paths.get(authorFile)).forEach(l -> {
157                 try {
158                     if (currentLine.incrementAndGet() > startline) {
159                         String json = l.substring(l.indexOf("{"));
160                         JSONObject jsonObj = new JSONObject(json);
161                         Author a = new Author();
162                         a.setName(jsonObj.optString(name:"name"));
163                         a.setPersonalName(jsonObj.optString(name:"personal_name"));
164                         a.setId(jsonObj.optString(name:"key").replaceAll("/authors/", ""));
165                         authorRepository.save(a);
166                         System.out.println(currentLine.get() + ": Saving author: " + a.getName());
167                     }
168                 } catch (Exception e) {
169                     e.printStackTrace();
170                 }
171             });
172         }
173     }

174     public void loadAuthors(int startline) throws IOException {
175
176         final AtomicInteger currentLine = new AtomicInteger(0);
177         Files.lines(Paths.get(authorFile)).forEach(l -> {
178             try {
179                 if (currentLine.incrementAndGet() > startline) {
180                     String json = l.substring(l.indexOf("{"));
181                     JSONObject jsonObj = new JSONObject(json);
182                     Author a = new Author();
183                     a.setName(jsonObj.optString(name:"name"));
184                     a.setPersonalName(jsonObj.optString(name:"personal_name"));
185                     a.setId(jsonObj.optString(name:"key").replaceAll("/authors/", ""));
186                     authorRepository.save(a);
187                     System.out.println(currentLine.get() + ": Saving author: " + a.getName());
188                 }
189             } catch (Exception e) {
190                 e.printStackTrace();
191             }
192         });
193     }

194     public void loadBooks() throws IOException {
195         DateTimeFormatter dt = DateTimeFormatter.ofPattern("yyyy-MM-dd'T'HH:mm:ss.SSSSSS");
196         Files.lines(Paths.get(workFile)).forEach(l -> {
197             try {
198                 String json = l.substring(l.indexOf("{"));
199                 JSONObject jsonObj = new JSONObject(json);
200                 Book b = new Book();
201                 b.setId(jsonObj.getString(name:"key").replace("/works/", ""));
202                 b.setName(jsonObj.optString(name:"title"));
203             }
204         });
205     }

```

- After Populating the Database.

The screenshot shows the DataStax Astra dashboard for the 'bookflix' database. On the left sidebar, under 'Databases', 'bookflix' is selected. The main panel displays 'Usage for Current Billing Period' with metrics: Read Requests (14.2k), Write Requests (21.3k), Storage Consumed (1.15 MB), and Data Transfer (45.29 MB). Below this, the 'Regions' section lists a single region: Google Cloud in North America (us-east1), Moncks Corner, South Carolina, Datacenter ID 574d9311-ca2a-4db0-9b..., and Region Availability Online. The status bar at the bottom shows system information like temperature (40°C Haze), battery level (40%), and system date (15-05-2023).

```

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>

The screenshot shows a GitHub repository page for 'satvi4 / Bookflix'. The repository is public and has 1 branch and 0 tags. The master branch was last updated 32 minutes ago. There are 78 commits in total. The commits are listed below:

Commit	Message	Time Ago
.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

At the bottom, there is a call to action to '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

The screenshot shows the Jenkins 'Update credentials' page. The URL is `Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) > satvi4/*****`. The page has a header with a Jenkins logo, search bar, notifications, and user info. On the left, there are 'Update', 'Delete', and 'Move' buttons. The main form has a 'Scope' dropdown set to 'Global (Jenkins, nodes, items, all child items, etc)'. It contains fields for 'Username' (satvi4), 'Password' (Concealed), 'ID' (dockerhub), and 'Description'. A 'Treat username as secret' checkbox is unchecked. A 'Change Password' button is visible next to the password field. A 'Save' button is at the bottom.

➤ Then we create a Jenkins pipeline.

The screenshot shows the Jenkins 'Enter an item name' dialog. The URL is `Dashboard > All >`. The page has a header with a Jenkins logo, search bar, notifications, and user info. The main area has a title 'Enter an item name' and an input field containing 'Bookflix'. Below the input field, there are four project type options: 'Freestyle project' (with a build icon), 'Maven project' (with a pom icon), 'Pipeline' (with a gear icon), and 'Multi-configuration project' (with a gear icon). Descriptions for each are provided. At the bottom, there is an 'OK' button and a note: 'Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder'.

➤ Git Script with Jenkins pipeline

```

❶ Jenkinsfile
1  pipeline {
2      environment{
3          dockerimg = ''
4          dockerimg2 = ''
5          DOCKERHUB_CREDENTIALS = credentials('dockerhub')
6      }
7      agent any
8
9      stages {
10         stage('Git pull') {
11             steps {
12                 git 'https://github.com/satvi4/Bookflix'
13             }
14         }
15         stage('Maven Build'){
16             steps {
17                 dir('backend'){
18                     sh 'mvn clean install'
19                 }
20             }
21         }
}

```

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 + v ... ^ x

satvika@Satvika:~/Documents/Bookflix\$ /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java @tm
 p/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 20:52:46 - User logged in with user id- satvi4
 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.

```

❶ Jenkinsfile
1  pipeline {
2      environment{
3          dockerimg = ''
4          dockerimg2 = ''
5          DOCKERHUB_CREDENTIALS = credentials('dockerhub')
6      }
7      agent any
8
9      stages {
10         stage('Git pull') {
11             steps {
12                 git 'https://github.com/satvi4/Bookflix'
13             }
14         }
15         stage('Maven Build'){
16             steps {
17                 dir('backend'){
18                     sh 'mvn clean install'
19                 }
20             }
21         }
}

```

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.

The screenshot shows the Eclipse IDE interface. On the left is the Explorer view, displaying the project structure for 'BOOKFLIX' with 'backend' selected. Inside 'backend', there are 'src', 'main', 'resources', and 'test' folders containing various Java files like 'BookFlixApplicationTests.java', 'BookSearchAPITest.java', 'CassandraConnectionTest.java', 'GitHubOAuth2LoginTest.java', and 'GoogleOAuth2LoginTest.java'. Below the project tree are icons for '.gitignore', 'Dockerfile', 'mvnw', 'mvnw.cmd', and 'pom.xml'. The central area is the code editor, showing the content of 'GitHubOAuth2LoginTest.java'. The code uses Mockito and Spring MockMvc to test GitHub OAuth2 login redirects. The right side of the interface includes a Problems view (18 issues), an Output view, a Debug Console, and a Terminal view. The Terminal view shows the application's startup logs, indicating it was started successfully on port 19383 at 2023-05-14 19:26:05.

```

14 import static org.mockito.Mockito.when;
15 import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.redirectedUrl;
16 import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
17
18 @WebMvcTest
19 public class GitHubOAuth2LoginTest {
20
21     @Autowired
22     private MockMvc mockMvc;
23
24     @MockBean
25     private ClientRegistrationRepository clientRegistrationRepository;
26
27     @Test
28     public void loginRedirectsToGitHubAuthorization() throws Exception {
29         // Mock the client registration for GitHub
30         ClientRegistration clientRegistration = ClientRegistration.withRegistrationId(registrationId)
31             .clientId("IV1.1b5dd7e2c0cf8cd")
32             .clientSecret(clientSecret:"8362896850feab096200d22d21980cd5d5472e84")
33             .authorizationGrantType(AuthorizationGrantType.AUTHORIZATION_CODE)
34     }
35
36     :: Spring Boot ::
37
38     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)
39     2023-05-14 19:26:05 - No active profile set, falling back to 1 default profile: "default"
40     2023-05-14 19:26:29 - Started BookflixApplication in 25.12 seconds (JVM running for 25.716)
41 satvika@Satvika:~/Documents/Bookflix$ 

```

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

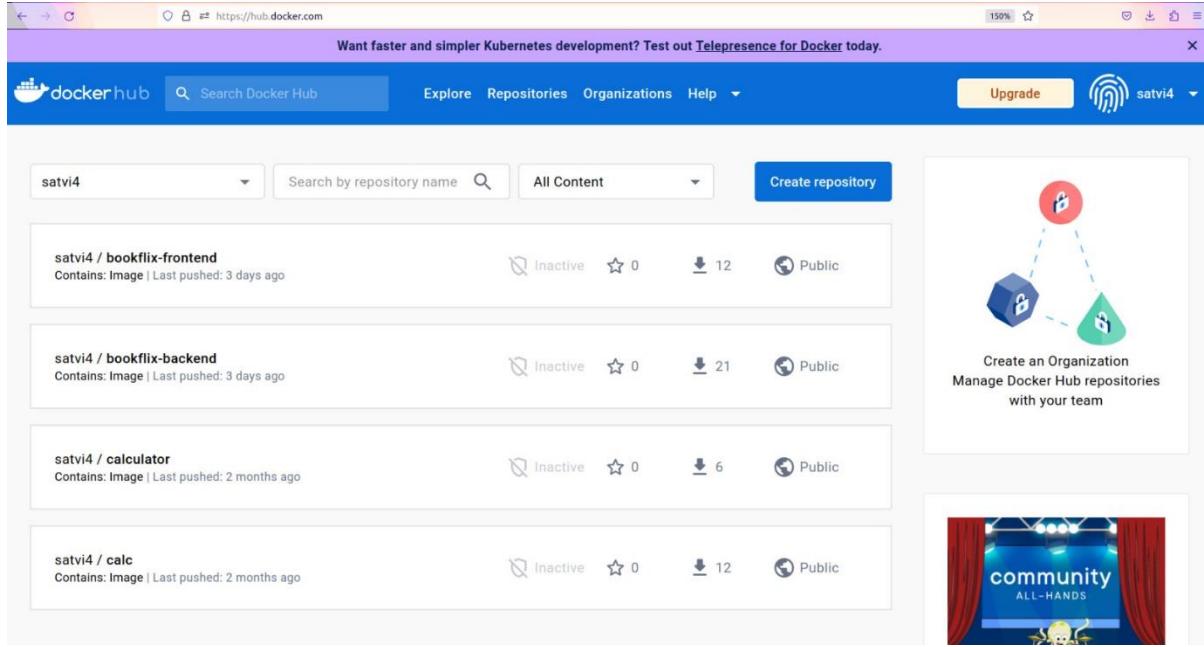
- Add docker integration plugin for docker in Jenkins → Plugin Manager.
- Build and Push Spring Boot Image.

```
backend > 🚤 Dockerfile > ...
1  FROM openjdk
2  COPY ./target/bookflix-0.0.1-SNAPSHOT.jar .
3  COPY ./src/main/resources/secure-connect.zip .
4  WORKDIR ./
5  EXPOSE 9090
6  CMD ["java", "-jar", "bookflix-0.0.1-SNAPSHOT.jar"]
```

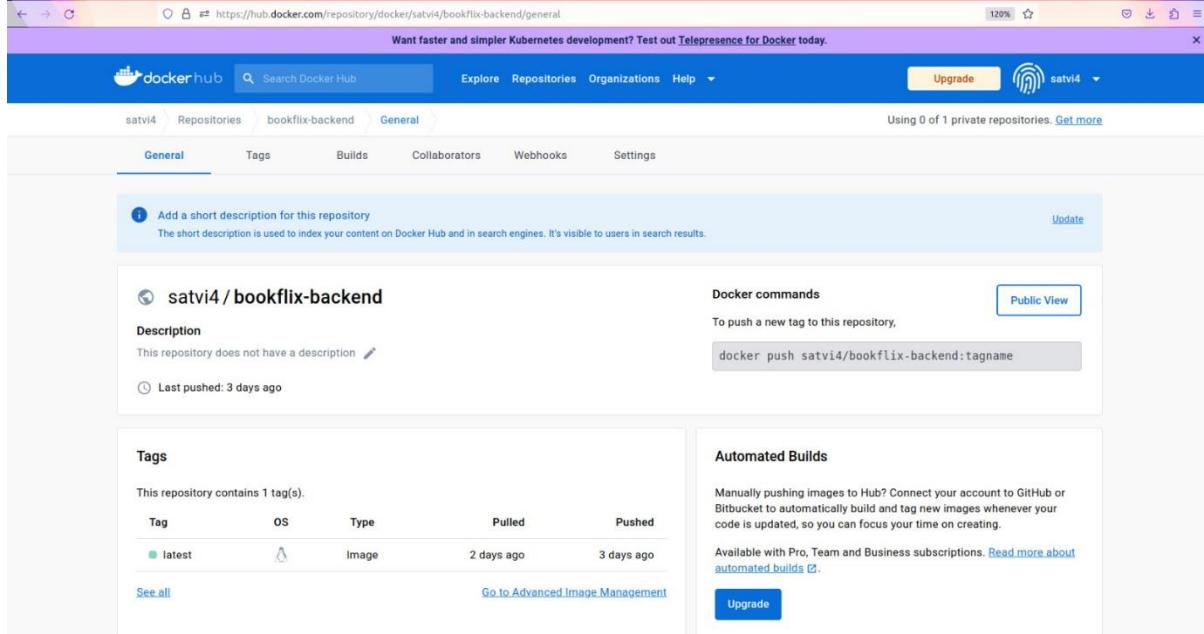
- Build and Push Frontend Image

```
frontend > 🚤 Dockerfile > ...
1  FROM nginx:alpine
2  COPY . /usr/share/nginx/html
3  WORKDIR ./usr/share/nginx/html
4
```

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



The screenshot shows the Docker Hub interface for the user 'satvi4'. It displays four public repositories: 'bookflix-frontend', 'bookflix-backend', 'calculator', and 'calc'. Each repository card includes details like status (Inactive), stars (0), downloads (12, 21, 6, 12 respectively), and visibility (Public). To the right of the repositories is a sidebar with options to 'Create an Organization' and 'Manage Docker Hub repositories with your team'. Below the repositories is a banner for the 'community ALL-HANDS' event.



The screenshot shows the detailed view for the 'bookflix-backend' repository under the 'satvi4' organization. The 'General' tab is selected. The repository has no description. The 'Docker commands' section shows the command 'docker push satvi4/bookflix-backend:tagname'. The 'Tags' section lists one tag, 'latest', which was pushed 3 days ago. The 'Automated Builds' section is available with Pro, Team, and Business subscriptions.

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
1  services:
2    bookflix-frontend:
3      image: satvi4/bookflix-frontend:latest
4      container_name: bookflix-frontend
5      stdin_open: true
6      ports:
7        - "3000:3000"
8      networks:
9        my-net:
10     restart: always
11     volumes:
12       - /frontend:/usr/share/nginx/html/templates
13
14   bookflix-backend:
15     image: satvi4/bookflix-backend:latest
16     container_name: bookflix-backend
17     ports:
18       - "9090:9090"
19     expose:
20       - 9090
21     networks:
22       my-net:
23     # Using Docker volume for persisting logs
24     volumes:
25       - /logs:/booklogs:rw
26       - /frontend:/templates
27
28   networks:
29     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.



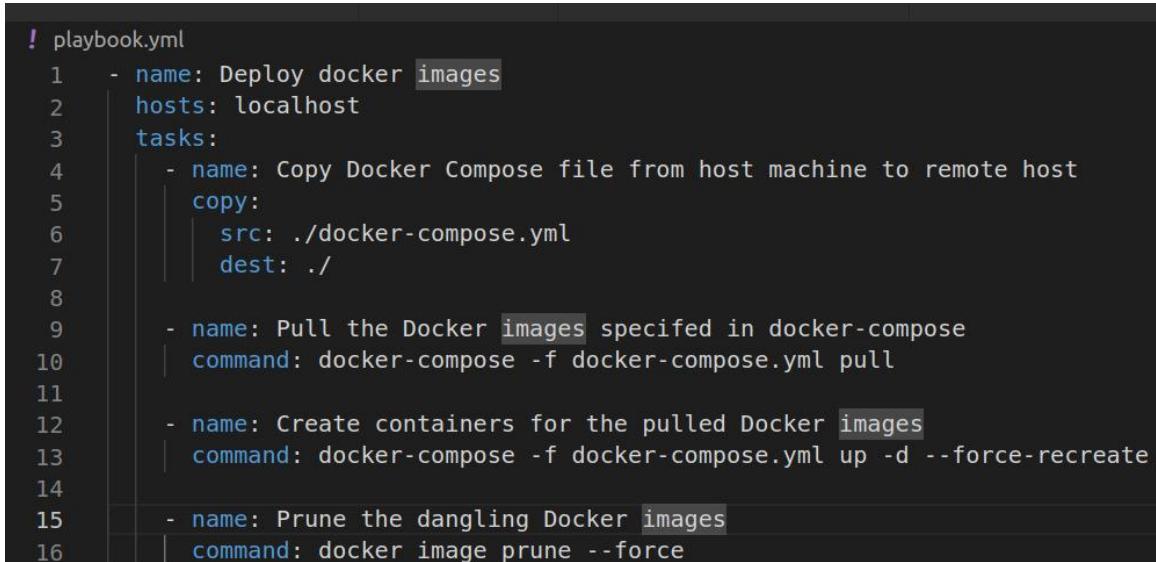
A screenshot of a Jenkins pipeline script. The code shows a Jenkins pipeline block with a 'stage' block named 'Ansible Deploy'. Inside the stage, there are 'steps' including a shell command to set the LANG environment variable and an Ansible playbook execution. The Ansible command includes options like 'becomeUser: null', 'colorized: true', 'disableHostKeyChecking: true', and 'installations'.

```

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

```

The dictionary is represented in key: value pair.



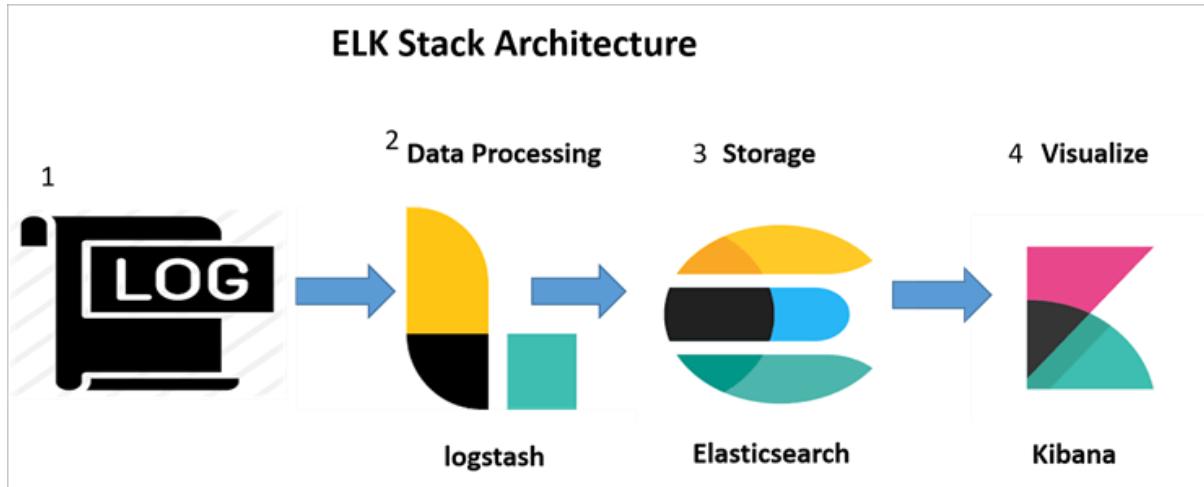
A screenshot of an Ansible playbook named 'playbook.yml'. The playbook defines a single job with the following tasks:

- 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 specified 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%

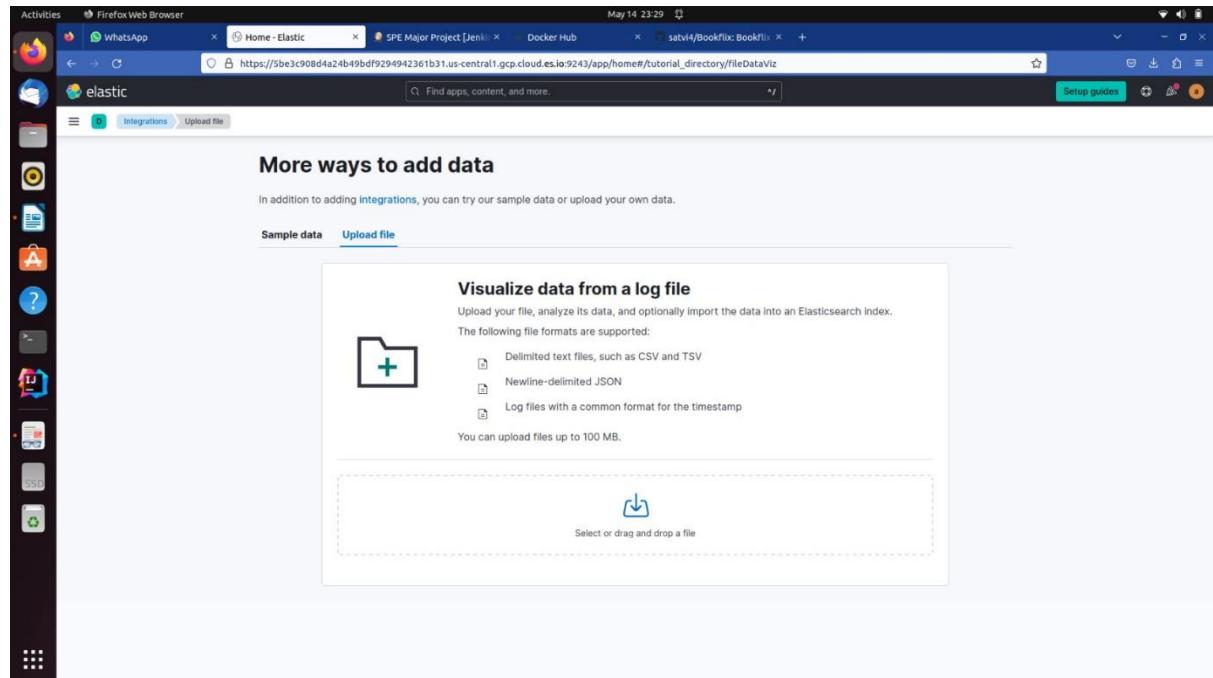
- **%d** – Returns the time when the log message occurred.
- **%thread** – Returns the name of the thread in which the log message occurred.
- **%-5level** – Returns the logging level of the log message (ERROR, WARN, INFO, DEBUG, and TRACE).
- **%logger{64}** – Returns the package with the package and 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.

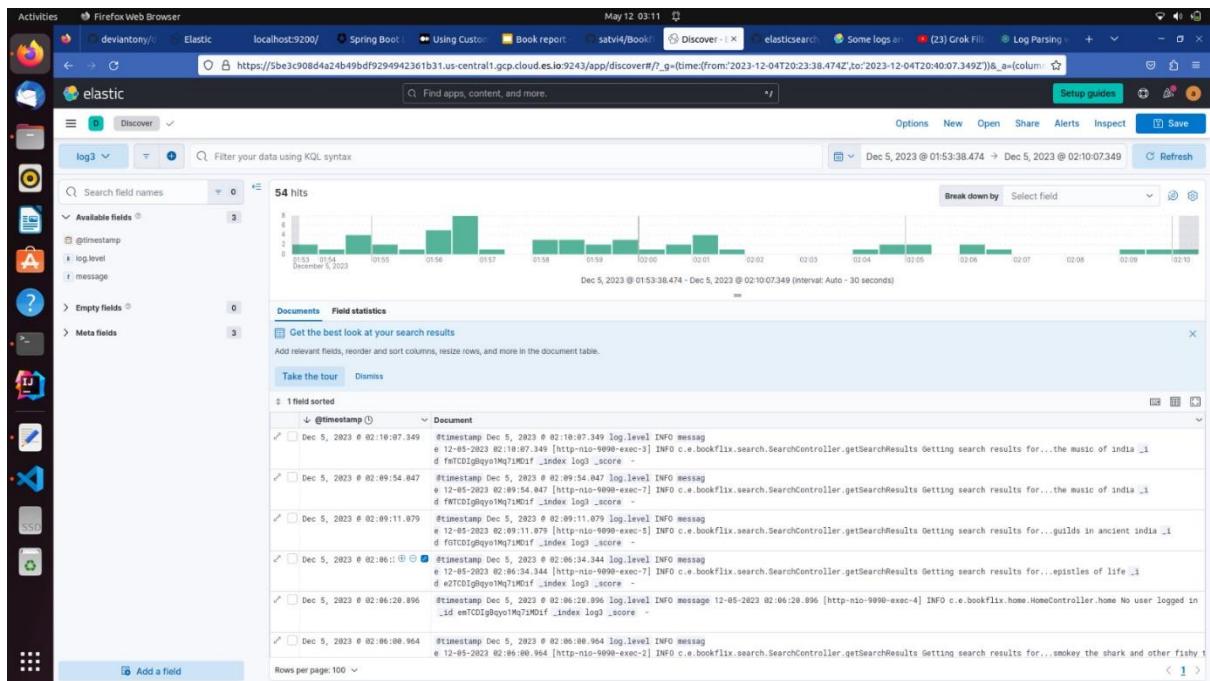
SAMPLE LOG

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}





4.8 Working with Webhooks

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

The screenshots illustrate the configuration of a GitHub webhook. In the first screenshot, the 'Webhooks / Add webhook' page is shown, where a payload URL is specified as `http://2a14-103-156-19-229.ngrok.io/github-webhook/`. In the second screenshot, the 'Webhooks' list page shows the newly created webhook with the URL `http://794e-2409-4081-ac17-8329... (push)`.

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.

The screenshot shows the Jenkins Pipeline Stage View for the 'SPE Major Project'. On the left, there's a sidebar with various Jenkins management icons. The main area displays a table titled 'Stage View' showing the execution times for different stages across multiple builds. The columns represent 'Git pull', 'Maven Build', 'Docker image build', 'Push Docker image', and 'Ansible Deploy'. The table includes a header row with average times and a summary row for the total pipeline run time.

	Git pull	Maven Build	Docker image build	Push Docker image	Ansible Deploy
Average stage times: (Average full run time: - 1min 48s)	1s	14s	3s	1min 11s	14s
#15 May 12, 2023, 12:19 PM May 12, 2023, 11:20 AM No Changes	1s	18s	3s	39s	16s
#16 May 12, 2023, 11:20 AM May 12, 2023, 1 commit	2s	13s	4s	1min 15s	11s
#17 May 12, 2023, 03:17 AM May 12, 2023, 1:41 AM 1 commit	1s	15s	4s	1min 26s	10s
#18 May 12, 2023, 12:40 AM May 12, 2023, 12:37 AM No Changes	1s	14s	4s	58s	12s
#19 May 12, 2023, 12:09 AM May 12, 2023, 11:50 PM 1 commit	1s	13s	3s	1min 24s	14s
#20 May 11, 2023, 9:59 PM May 11, 2023, 4:11 PM May 12, 2023, 00:40 No Changes	1s	11s	2s	46s	13s

- Deployed images on the target machine.

```

satvika@Satvika: ~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND           CREATED          STATUS          PORTS          NAMES
ffd704475eeb   satvi4/bookflix-backend:latest   "java -jar bookflix-..."   2 days ago       Exited (143) 2 days ago
07db37db4858   satvi4/bookflix-frontend:latest  "/docker-entrypoint..."  2 days ago       Exited (0) 3 seconds ago
c7f05eb62645   e9e754064dde   "/docker-entrypoint..."  3 days ago       Exited (0) 3 days ago
ba82cd932638   0e429962ca00   "/java -jar bookflix-..."  3 days ago       Exited (130) 3 days ago
d1241a6773d7   frontend-html    "/docker-entrypoint..."  4 days ago       Exited (0) 4 days ago
390a062a9445   frontend-html    "/docker-entrypoint..."  4 days ago       Exited (127) 4 days ago
984eb49bc2fd   frontend-html    "/java -jar bookflix-..."  4 days ago       Exited (130) 3 days ago
d3d2daec5aa0   85990e45244d   "/java -jar bookflix-..."  4 days ago       Exited (130) 4 days ago
23a567d4aa44   85990e45244d   "/java -jar bookflix-..."  4 days ago       Exited (130) 4 days ago
da8f05a13d47   gcr.io/k8s-minikube/kicbase:v0.0.37  "/usr/local/bin/entr..."  7 weeks ago     Exited (137) 5 weeks ago
1b6e8392e664   satvi4/calculator  "/java -jar Calculato..."  7 weeks ago     Exited (0) 7 weeks ago
9c6b16749725   gaurangi99/spe-mini-project  "/java -jar SPE_MiniP..."  8 weeks ago     Exited (0) 8 weeks ago
9f2298fe0cc7   eb44f51e82c6   "/bin/bash"          8 weeks ago     Exited (255) 8 weeks ago
4192cf437e7c  ubuntu          "/bin/bash"          2 months ago    Exited (255) 8 weeks ago
f32bbc9a94ea  ubuntu          "/bin/bash"          2 months ago    Exited (0) 2 months ago
efdfaf00341a1  copy_add:latest  "/bin/bash"          3 months ago    Exited (0) 3 months ago
71482bf9f1d6  ubuntu          "/bin/bash"          3 months ago    Exited (255) 8 weeks ago
c79632b63f16  ubuntu          "/bin/bash"          3 months ago    Exited (0) 3 months ago
satvika@Satvika: ~$
```



```

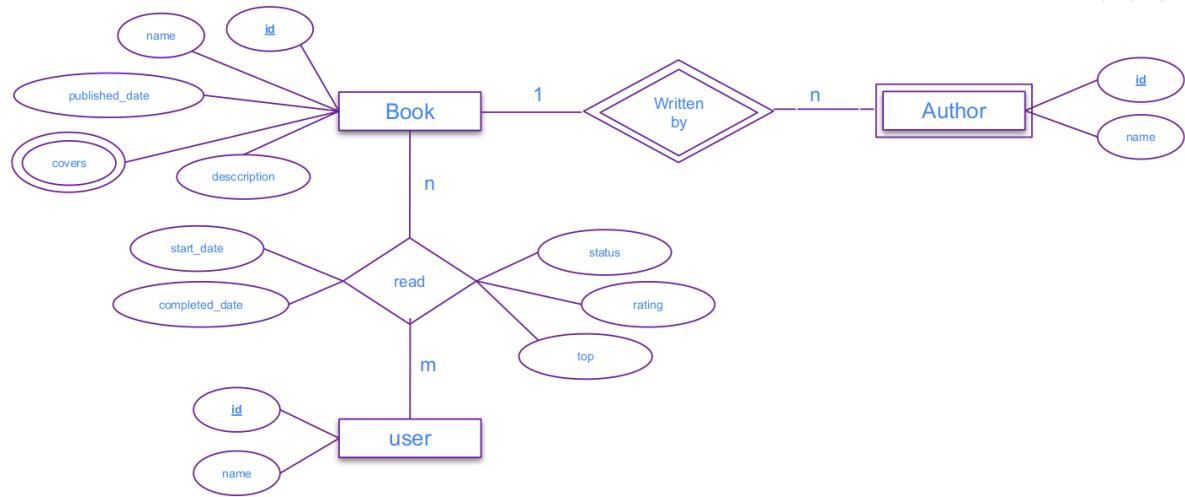
satvika@Satvika: ~/Documents/Bookflix  x      satvika@Satvika: ~
satvika@Satvika: ~$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
satvi4/bookflix-frontend  latest  a86afaf6de8e  2 days ago    41MB
satvi4/bookflix-backend  latest  d35e41376fb5  2 days ago    520MB
satvi4/bookflix-backend <none>  edb2e801bf32  3 days ago    519MB
satvi4/bookflix-frontend <none>  bcebb19bd5ae6  3 days ago    41MB
frontend-html         latest  9157dc824b1f  4 days ago    41MB
satvi4/bookflix        <none>  85990e45244d  4 days ago    519MB
satvi4/bookflix        6       67c59695a37d  2 weeks ago   519MB
satvi4/bookflix        5       320e2f7862ec  2 weeks ago   519MB
satvi4/bookflix        4       9eed4526fdeb  2 weeks ago   519MB
nginx                alpine   8e75cbc5b25c  6 weeks ago   41MB
satvi4/calculator      latest  cdb43cf94c25  7 weeks ago   472MB
satvi4/calculator      1       568781433f18  7 weeks ago   472MB
satvi4/calc             latest  4c22741b5fda  8 weeks ago   472MB
gaurangi99/spe-mini-project  latest  84197255758e  8 weeks ago   472MB
copy_add              latest  96f7c4fcf226  3 months ago  77.8MB
openjdk               latest  71260f256d19  3 months ago  470MB
gcr.io/k8s-minikube/kicbase v0.0.37  01c0ce65fff7  3 months ago  1.15GB
ubuntu                latest  58db3edaf2be  3 months ago  77.8MB
satvika@Satvika: ~$
```

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



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 ... ior.java U ● J Book.java .../entity U ● J BooksByUser.java U ● J SearchResult.java U ● J SearchResultBook.java U ● D v
BOOKFLIX D+ ⌂ ⌂ ⌂ @ backend > src > main > java > com > example > bookflix > entity > J SearchResultBook.java > {} com.example.bookflix.entity

src
  main
    java/com/example/bookflix
      connection
        J DataStaxAstraProperties.java
      controller
        J BookController.java
        J HomeController.java
        J SearchController.java
        J UserBooksController.java
      entity
        J Author.java
        J Book.java
        J BooksByUser.java
        J SearchResult.java
      J SearchResultBook.java U
      J UserBooks.java
      J UserBooksPrimaryKey.java
      repository
        J AuthorRepository.java 1,U
        J BookRepository.java 1,U
        J BooksByUserRepository.java U
        J UserBooksRepository.java U
      I BookflixAnnlication.java 7
    > OUTLINE
    > TIMELINE
    > JAVA PROJECTS
    > MAVEN

EXPLORER ... J SearchResultBook.java U ● J Book.java .../entity U ● J BooksByUser.java U ● J SearchResult.java U ● J SearchResultBook.java U ● D v
BOOKFLIX D+ ⌂ ⌂ ⌂ @ backend > src > main > java > com > example > bookflix > controller > J SearchController.java > Language Support for Java(TM) by Red Hat >
  20
  21 @Controller
  22 public class SearchController {
  23
    private final String COVER_IMAGE_ROOT = "http://covers.openlibrary.org/b/id/";
  24
    private final WebClient webClient;
  25
    private static final Logger logger = LogManager.getLogger(SearchController.class);
  26
  27
    public SearchController(WebClient.Builder webClientBuilder) {
  28      this.webClient = webClientBuilder.exchangeStrategies(ExchangeStrategies.builder()
  29        .codecs(configurer -> configurer
  30          .defaultCodecs()
  31            .maxInMemorySize(16 * 1024 * 1024))
  32          .build()).baseUrl("http://openlibrary.org/search.json").build();
  33
  34
  35
  36
  37
  38 @GetMapping(value = "/search")
  39 public String getSearchResults(@RequestParam String query, Model model) {
  40   logger.info("Getting search results for..." + query);
  41   Mono<SearchResult> resultsMono = this.webClient.get()
  42     .uri(uri:"?q={query}", query)
  43     .retrieve().bodyToMono(elementClass:SearchResult.class);
  44   SearchResult result = resultsMono.block();
  45   List<SearchResultBook> books = result.getDocs()
  46     .stream()
  47     .limit(10)
  48     .map(bookResult -> {
  49       bookResult.setKey(bookResult.getKey().replace("/works/", ""));
  50       String coverId = bookResult.getCover_i();
  51       if (StringUtils.hasText(coverId)) {
```

EXPLORER

```

backend > src > main > java > com > example > bookflix > entity > J BooksByUser.java > Language Support for Java(TM) by Red Hat > {} com.example.bookflix
  < src
    < main
      < java/com/example/bookflix
        < connection
          J DataStaxAstraProperties.java
        < controller
          J BookController.java
          J HomeController.java
          J SearchController.java
          J UserBooksController.java
        < entity
          J Author.java
          J Book.java
          J BooksByUser.java
          J SearchResult.java
          J SearchResultBook.java
          J UserBooks.java
          J UserBooksPrimaryKey.java
        < repository
          J AuthorRepository.java
          J BookRepository.java
          J BooksByUserRepository.java
          J UserBooksRepository.java
        < application
          L BookflixApplication.java
    > OUTLINE
    > TIMELINE
    > JAVA PROJECTS
    > MAVEN
  > EXPLORER
  > a .../entity U
  > J BooksByUser.java U
  > J SearchResult.java U
  > J SearchResultBook.java U
  > J UserBooks.java U
  > D ...

```

```

17 /**
18 * Model that represents the books read by a user.
19 * Helps with showing user's recent books in the dashboard page.
20 */
21 @Table(value = "books_by_user")
22 public class BooksByUser {
23
24     @PrimaryKeyColumn(name = "user_id", ordinal = 0, type = PrimaryKeyType.PARTITIONED)
25     private String id;
26
27     @PrimaryKeyColumn(name = "book_id", ordinal = 1, type = PrimaryKeyType.CLUSTERED)
28     @CassandraType(type = Name.TEXT)
29     private String bookId;
30
31     @PrimaryKeyColumn(type = PrimaryKeyType.CLUSTERED, ordering = Ordering.DESCENDING)
32     @CassandraType(type = Name.TIMESTAMP)
33     private UUID timeUuid;
34
35     @PrimaryKeyColumn(type = PrimaryKeyType.CLUSTERED, ordering = Ordering.ASCENDING)
36     @CassandraType(type = Name.TEXT)
37     private String readingStatus;
38
39     @Column("book_name")
40     @CassandraType(type = Name.TEXT)
41     private String bookName;
42
43     @Column("author_names")
44     @CassandraType(type = Name.LIST, typeArguments = Name.TEXT)
45     private List<String> authorNames;
46
47     @Column("cover_ids")
48     @CassandraType(type = Name.LIST, typeArguments = Name.TEXT)

```

EXPLORER

```

backend > src > main > java > com > example > bookflix > entity > J UserBooks.java > Language Support for Java(TM) by Red Hat > {} com.example.bookflix
  < src
    < main
      < java/com/example/bookflix
        < connection
          J DataStaxAstraProperties.java
        < controller
          J BookController.java
          J HomeController.java
          J SearchController.java
          J UserBooksController.java
        < entity
          J Author.java
          J Book.java
          J BooksByUser.java
          J SearchResult.java
          J SearchResultBook.java
          J UserBooks.java
          J UserBooksPrimaryKey.java
        < repository
          J AuthorRepository.java
          J BookRepository.java
          J BooksByUserRepository.java
          J UserBooksRepository.java
        < application
          L BookflixApplication.java
    > OUTLINE
    > TIMELINE
    > JAVA PROJECTS
    > MAVEN
  > EXPLORER
  > a .../entity U
  > J BooksByUser.java U
  > J SearchResult.java U
  > J SearchResultBook.java U
  > J UserBooks.java U
  > D ...

```

```

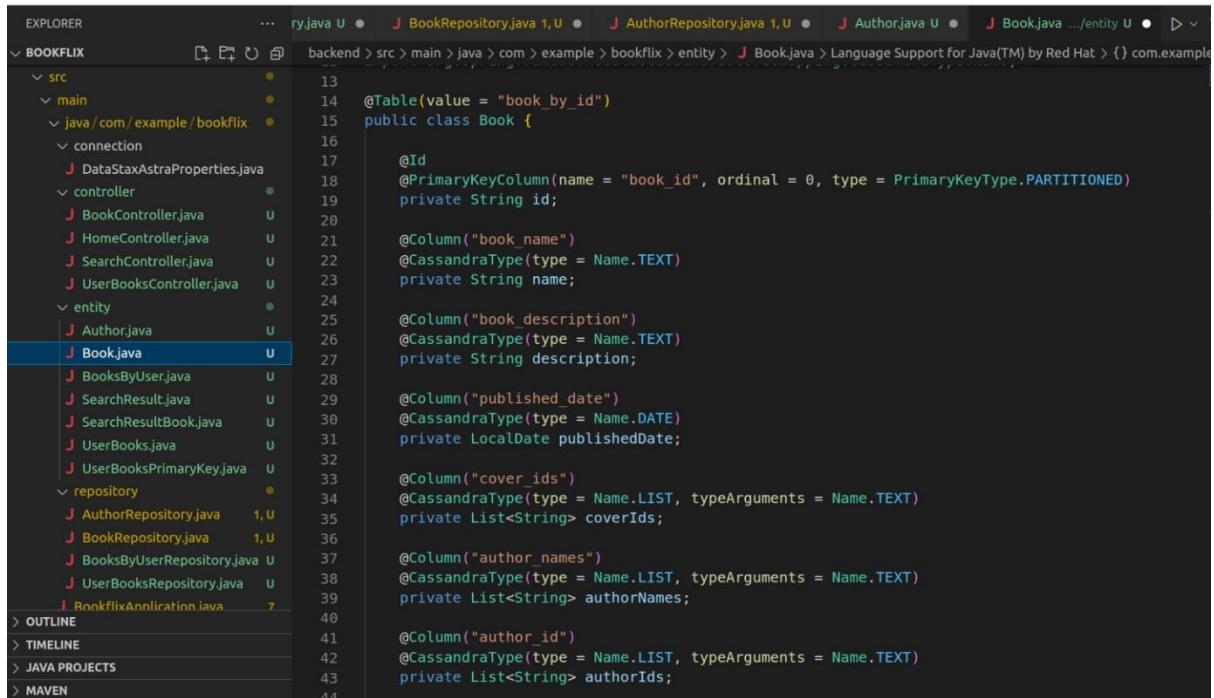
10
11     @Table(value = "book_by_user_and_bookid")
12     public class UserBooks {
13
14         @PrimaryKey
15         private UserBooksPrimaryKey key;
16
17         @Column("started_date")
18         @CassandraType(type = Name.DATE)
19         private LocalDate startedDate;
20
21         @Column("completed_date")
22         @CassandraType(type = Name.DATE)
23         private LocalDate completedDate;
24
25         @Column("reading_status")
26         @CassandraType(type = Name.TEXT)
27         private String readingStatus;
28
29         @Column("rating")
30         @CassandraType(type = Name.INT)
31         private int rating;
32
33         public UserBooksPrimaryKey getKey() {
34             return key;
35         }
36
37         public void setKey(UserBooksPrimaryKey key) {
38             this.key = key;
39         }
40
41         public LocalDate getStartedDate() {

```

```

24
25  @Controller
26  public class UserBooksController {
27
28      @Autowired
29      UserBooksRepository userBooksRepository;
30
31      @Autowired
32      BooksByUserRepository booksByUserRepository;
33
34      @Autowired
35      BookRepository bookRepository;
36
37      private static final Logger logger = LogManager.getLogger(clazz:UserBooksController.class);
38
39      @PostMapping("/addUserBook")
40      public ModelAndView addBookForUser(
41          @RequestBody MultiValueMap<String, String> formData,
42          @AuthenticationPrincipal OAuth2User principal) {
43
44          if (principal == null || principal.getAttribute(name:"login") == null) {
45              logger.info(message:"User not logged in...");
46              return null;
47          }
48
49          String bookId = formData.getFirst(key:"bookId");
50          Optional<Book> optionalBook = bookRepository.findById(bookId);
51
52          if (!optionalBook.isPresent()) {
53              logger.info("Book Data not available for book id =" + bookId);
54              return new ModelAndView(viewName:"redirect:/");
55          }
56
57          Book book = optionalBook.get();

```



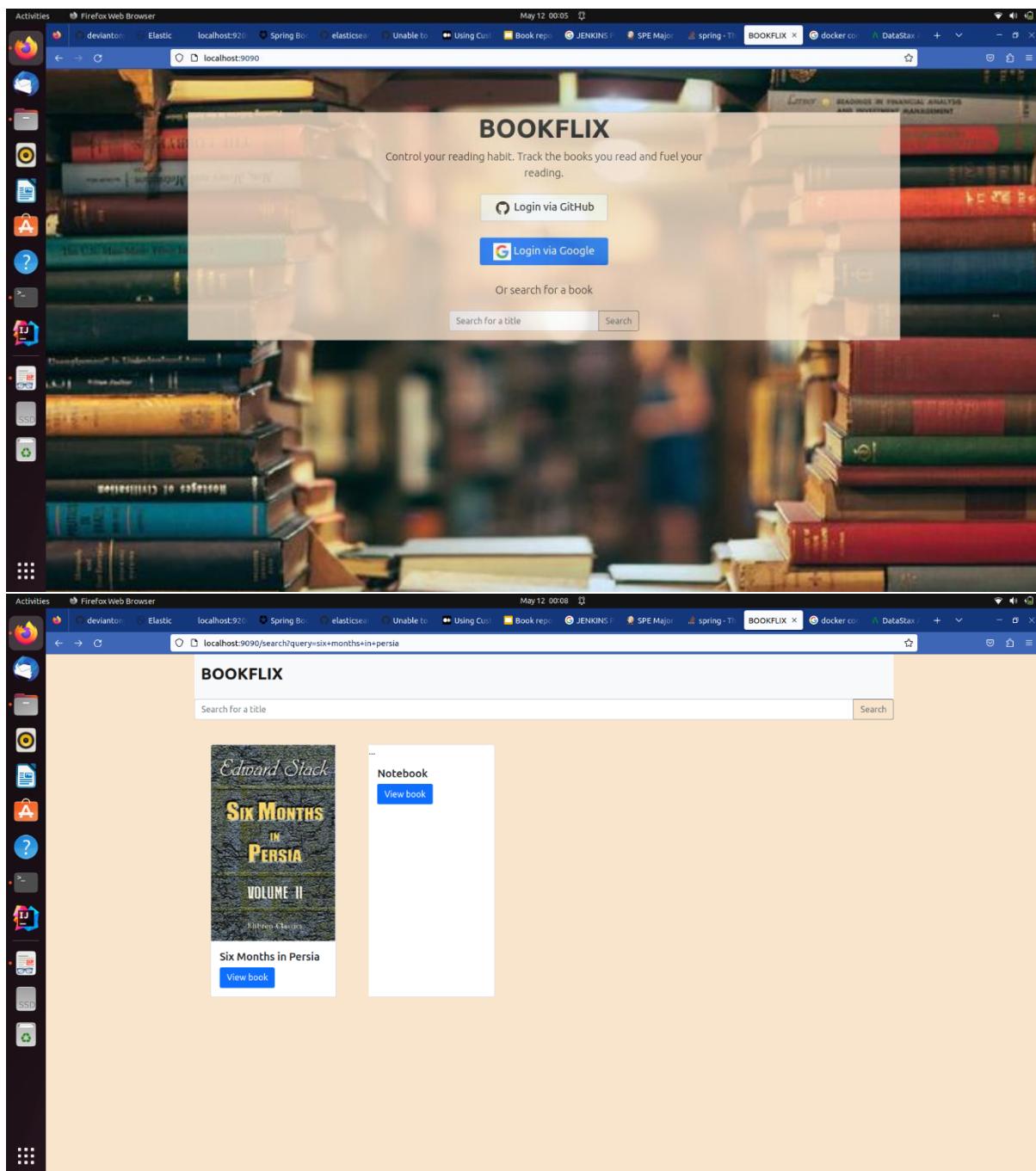
The screenshot shows the Eclipse IDE interface with the 'EXPLORER' view open. The left pane displays the project structure under 'BOOKFLIX'. The 'src' folder contains several packages: 'main', 'connection', 'controller', 'entity', and 'repository'. Under 'entity', the 'Book.java' file is selected and highlighted with a blue background. The right pane shows the code for the 'Book' class. The code defines a table named 'book_by_id' with various columns and their types. The 'Book' class has an ID column, a name column, a description column, a published date column, a list of cover IDs, a list of author names, and a list of author IDs.

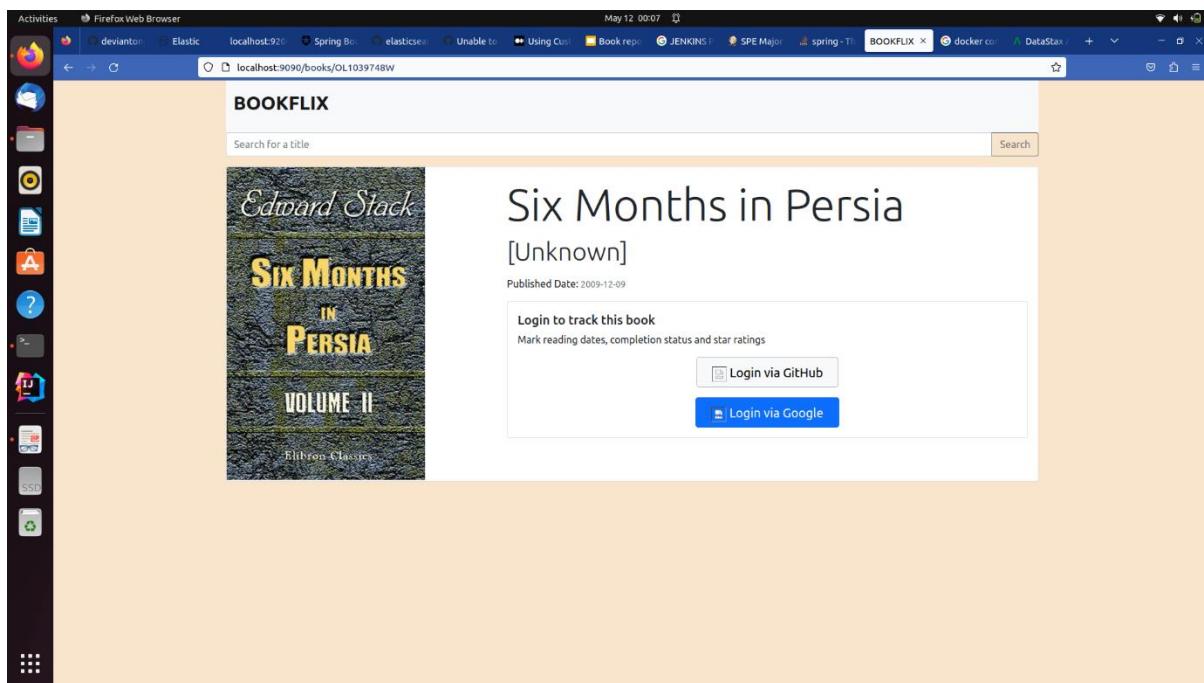
```


13
14     @Table(value = "book_by_id")
15     public class Book {
16
17         @Id
18         @PrimaryKeyColumn(name = "book_id", ordinal = 0, type = PrimaryKeyType.PARTITIONED)
19         private String id;
20
21         @Column("book_name")
22         @CassandraType(type = Name.TEXT)
23         private String name;
24
25         @Column("book_description")
26         @CassandraType(type = Name.TEXT)
27         private String description;
28
29         @Column("published_date")
30         @CassandraType(type = Name.DATE)
31         private LocalDate publishedDate;
32
33         @Column("cover_ids")
34         @CassandraType(type = Name.LIST, typeArguments = Name.TEXT)
35         private List<String> coverIds;
36
37         @Column("author_names")
38         @CassandraType(type = Name.LIST, typeArguments = Name.TEXT)
39         private List<String> authorNames;
40
41         @Column("author_id")
42         @CassandraType(type = Name.LIST, typeArguments = Name.TEXT)
43         private List<String> authorIds;
44


```

6. Results and Discussion







Sign in to GitHub
to continue to SPEMajor

Username or email address

Password [Forgot password?](#)

[Sign in](#)

New to GitHub? [Create an account.](#)

The image shows two screenshots of the BOOKFLIX application running on a Linux desktop environment.

Screenshot 1: My Books

This screenshot shows the user's library. It displays a single book entry for "The Ranch House" by Alan Hess. The book cover features a photograph of a ranch-style house with a large porch. Below the cover, the title "The Ranch House" and author "Alan Hess" are listed, along with the status "Finished" and a rating of three stars.

Screenshot 2: Book Detail Page

This screenshot shows a detailed view of the book "Six Months in Persia" by Edward Stack. The book cover is shown on the left, featuring the title "Six Months in Persia" and "VOLUME - II". To the right of the cover, the book's title "Six Months in Persia" and author "[Unknown]" are displayed. Below this, the published date "2009-12-09" is shown. The page contains several input fields for tracking reading progress: "Start date" (mm / dd / yyyy), "Completed date" (mm / dd / yyyy), "Status" (a dropdown menu), and "Rating" (a dropdown menu). A "Submit" button is located at the bottom of the form.

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>