Spring Boot + MongoDB Application Documentation

This document provides a detailed guide to setting up, running, and deploying a Spring Boot application that interacts with a MongoDB database. The application exposes a single GET "/books" endpoint that retrieves data from the MongoDB database.

Table of Contents

---

1. Overview

The Spring Boot application integrates with a MongoDB backend. The MongoDB database stores book records, which are retrieved and displayed through the `/books` API endpoint. This document covers:

- Steps to build and run the application locally.
- Deployment instructions on Minikube.
- CI/CD pipeline setup using GitHub Actions.

---

2. Local Setup Using Docker

Prerequisites

- Install Docker on your local machine.
- Clone the project repository from GitHub.

Instructions

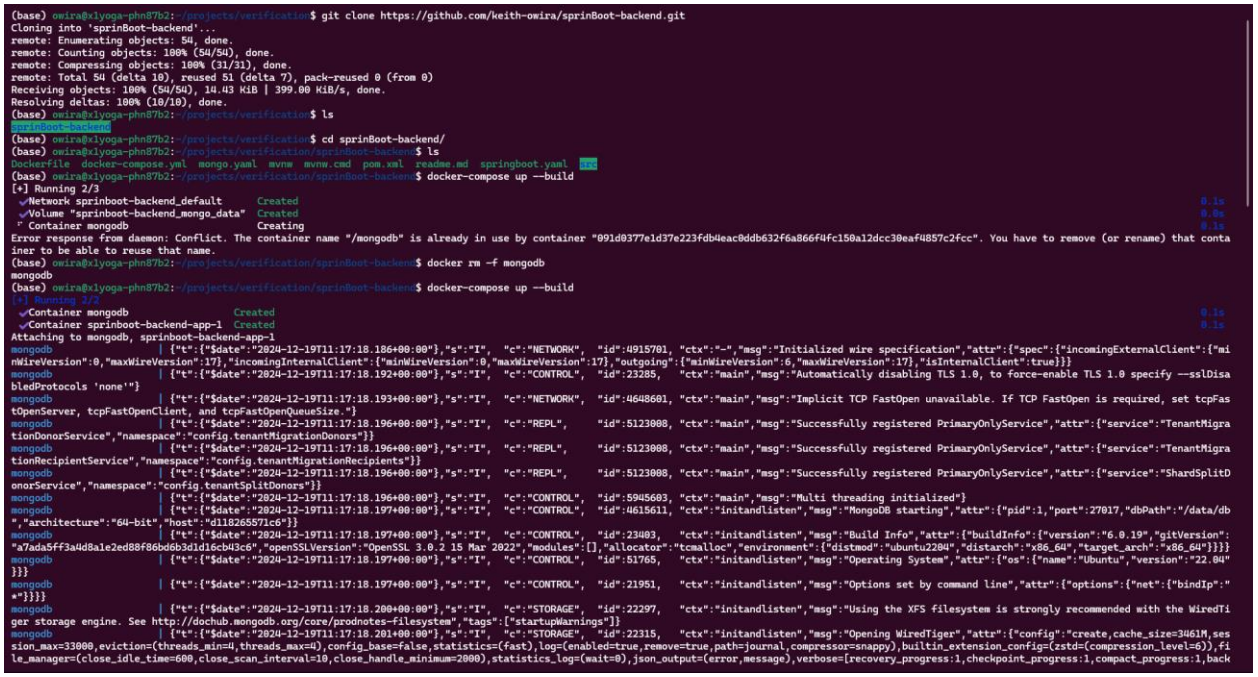
1. Build and Start the Application

Navigate to the project's root directory and execute:

```
docker-compose up --build
```

This command will:

- Build the Docker image for the Spring Boot application using the `Dockerfile`.
- Start the Spring Boot and MongoDB containers defined in `docker-compose.yml`.



2. Access the Application

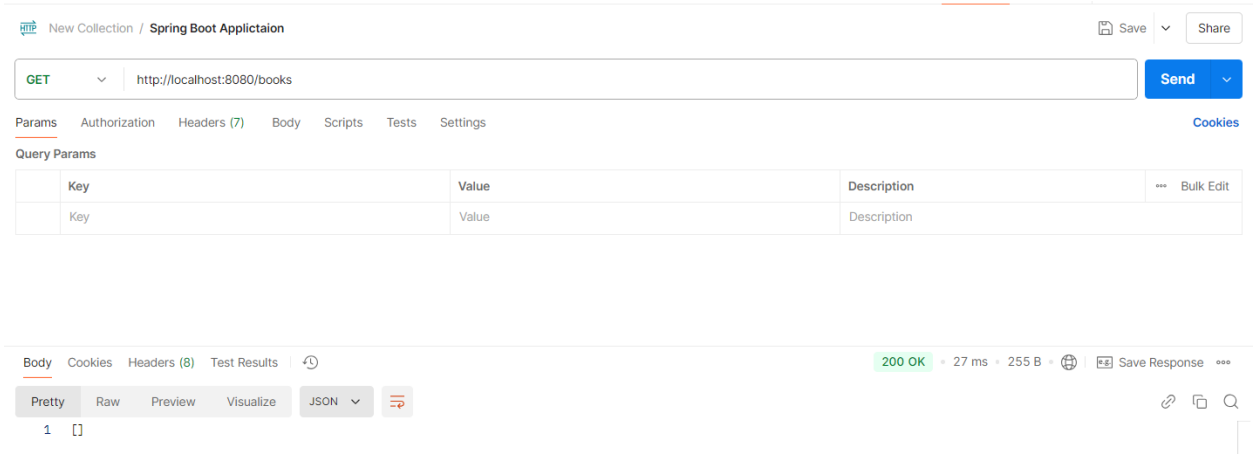
Open a browser or use a curl command:

`http://localhost:8080/books`

Expected Response:

`[]`

This indicates an empty database.



3. Add Sample Data to MongoDB

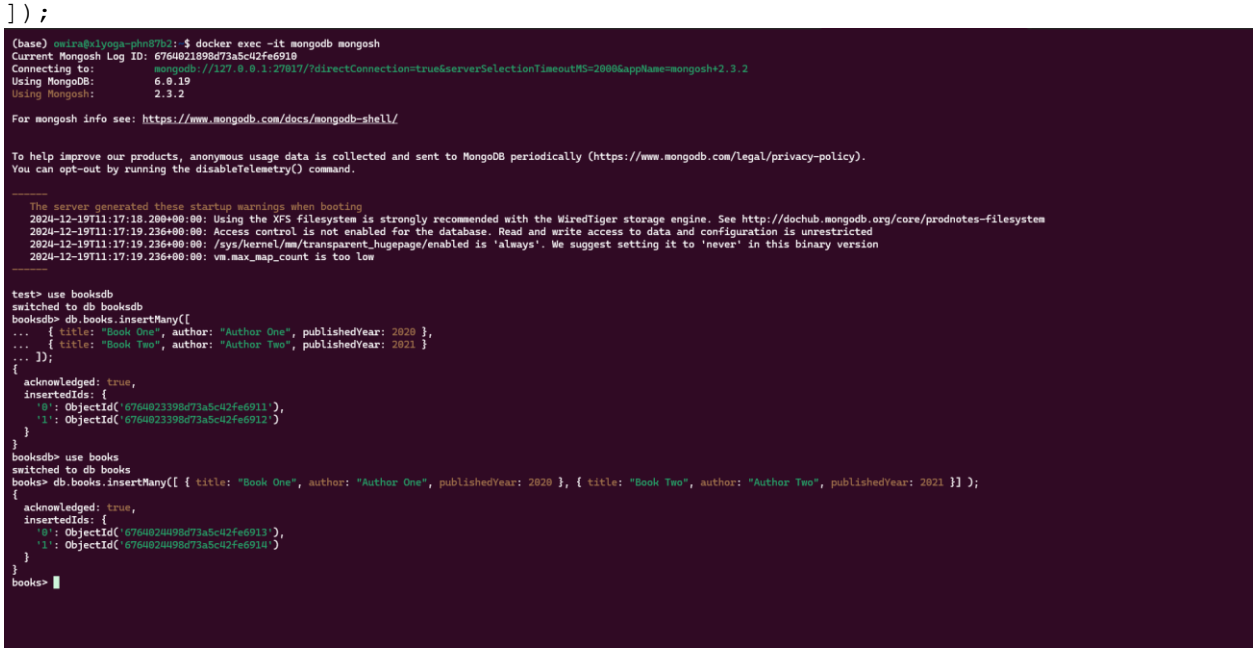
To see meaningful API responses, add sample data to MongoDB:

- Access the MongoDB terminal:

```
docker exec -it mongodb mongosh
```

- Switch to the `books` database and insert sample data:

```
use books;
db.books.insertMany([
  { title: "Book One", author: "Author One", publishedYear: 2020 },
  { title: "Book Two", author: "Author Two", publishedYear: 2021 }
]);
```

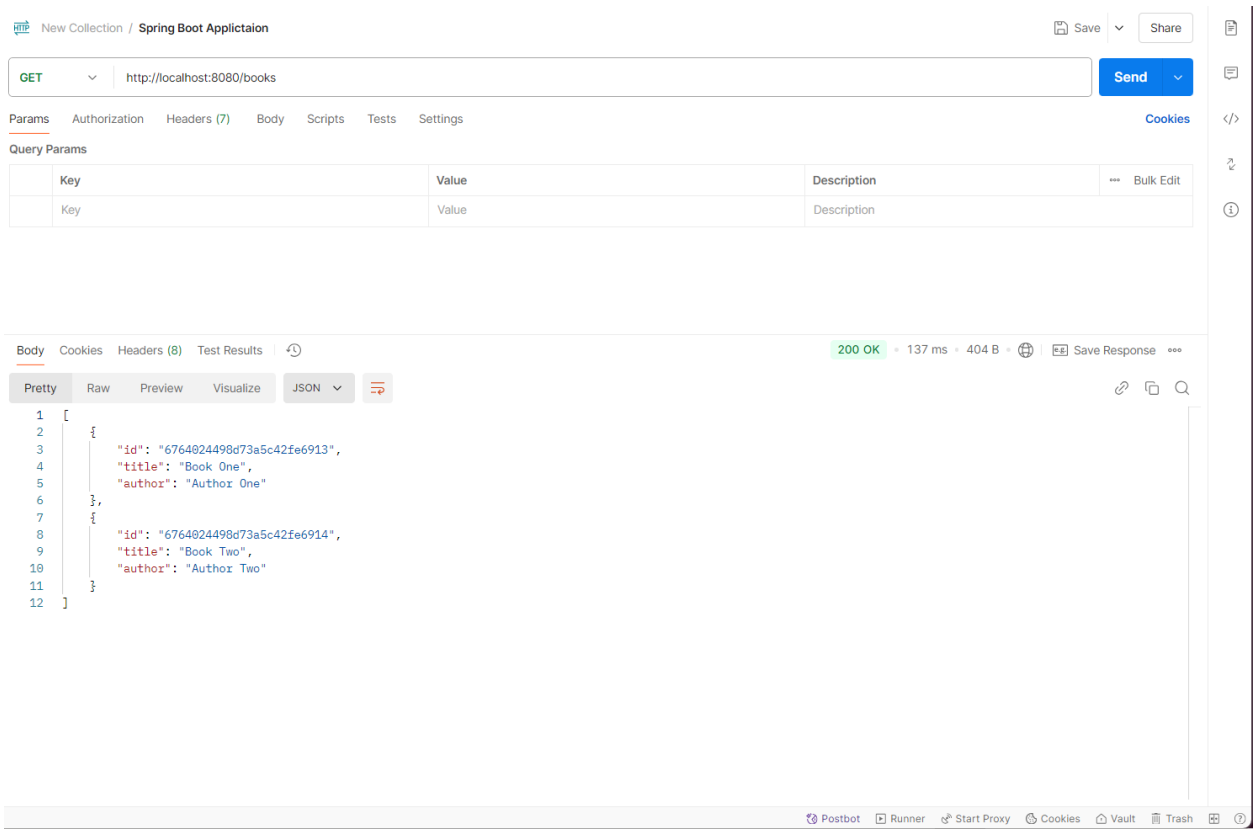


- Test the endpoint again:

```
curl http://localhost:8080/books
```

Expected Response:

```
[
  { "title": "Book One", "author": "Author One", "publishedYear": 2020 },
  { "title": "Book Two", "author": "Author Two", "publishedYear": 2021 }
]
```



3. Deployment on Minikube

Prerequisites

- Install kubectl.
- Install Minikube.

Setup and Deployment

1. Start Minikube

```
minikube start
```

2. Apply MongoDB Configuration

Deploy MongoDB using `mongo.yaml`:

```
kubectl apply -f mongo.yaml
```

3. Apply Spring Boot Backend Configuration

Deploy the Spring Boot backend using `springboot.yaml`:

```
kubectl apply -f springboot.yaml
```

4. Verify Deployments

Check the status of running pods:

```
kubectl get pods
```

```
(base) owira@x1yoga-phn87b2:~/projects/verification/sprinBoot-backend$ minikube start
😄  minikube v1.34.0 on Ubuntu 22.04 (amd64)
✨  Using the docker driver based on existing profile
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.45 ...
🔄  Restarting existing docker container for "minikube" ...
🐳  Preparing Kubernetes v1.31.0 on Docker 27.2.0 ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
(base) owira@x1yoga-phn87b2:~/projects/verification/sprinBoot-backend$ kubectl apply -f mongo.yaml
deployment.apps/mongodb created
service/mongodb-service created
(base) owira@x1yoga-phn87b2:~/projects/verification/sprinBoot-backend$ kubectl apply -f springboot.yaml
deployment.apps/springboot-backend created
service/springboot-backend-service created
(base) owira@x1yoga-phn87b2:~/projects/verification/sprinBoot-backend$ kubectl get all
NAME                                       READY   STATUS    RESTARTS   AGE
pod/mongodb-9cc4d488f-k4gcd                1/1     Running   0          32s
pod/springboot-backend-788f668d7c-99xms    1/1     Running   0          27s

NAME                                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/kubernetes                    ClusterIP   10.96.0.1       <none>        443/TCP          14h
service/mongodb-service               ClusterIP   10.99.47.234    <none>        27017/TCP        32s
service/springboot-backend-service    NodePort    10.99.50.160    <none>        8080:31637/TCP   27s

NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mongodb               1/1     1            1           32s
deployment.apps/springboot-backend    1/1     1            1           27s

NAME                                             DESIRED   CURRENT   READY   AGE
replicaset.apps/mongodb-9cc4d488f                1         1         1       32s
replicaset.apps/springboot-backend-788f668d7c    1         1         1       27s
(base) owira@x1yoga-phn87b2:~/projects/verification/sprinBoot-backend$
```

```
(base) owira@x1yoga-phn87b2:~/projects/verification/sprinBoot-backend$ kubectl logs springboot-backend-788f668d7c-99xms

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.4.0)

2024-12-19T11:29:32.514Z  INFO 1 --- [getBookAPI] [           main] c.k.getBook.GetBookApiApplication        : Starting GetBookApiApplication v0.0.1-SNAPSHOT using Java 17.0.2 with PID 1 (/app/app.jar started
 by root in /app)
2024-12-19T11:29:32.517Z  INFO 1 --- [getBookAPI] [           main] c.k.getBook.GetBookApiApplication        : No active profile set, falling back to 1 default profile: "default"
2024-12-19T11:29:33.522Z  INFO 1 --- [getBookAPI] [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data MongoDB repositories in DEFAULT mode.
2024-12-19T11:29:33.654Z  INFO 1 --- [getBookAPI] [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 124 ms. Found 1 MongoDB repository interface.
2024-12-19T11:29:34.435Z  INFO 1 --- [getBookAPI] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2024-12-19T11:29:34.451Z  INFO 1 --- [getBookAPI] [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-12-19T11:29:34.451Z  INFO 1 --- [getBookAPI] [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.33]
2024-12-19T11:29:34.627Z  INFO 1 --- [getBookAPI] [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2024-12-19T11:29:34.630Z  INFO 1 --- [getBookAPI] [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2010 ms
2024-12-19T11:29:35.348Z  INFO 1 --- [getBookAPI] [           main] org.mongodb.driver.client                : MongoClient with metadata {"driver": {"name": "mongo-java-driver|sync|spring-boot", "version": "5
.2.1"}, "os": {"type": "Linux", "name": "Linux", "architecture": "amd64", "version": "5.15.167.4-microsoft-standard-WSL2"}, "platform": "Java/Oracle Corporation/17.0.2+8-86", "env": {"container": {"runtime":
"docker", "orchestrator": "kubernetes"}}} created with settings MongoClientSettings{readPreference=primary, writeConcern=WriteConcern{w=null, wTimeout=null ms, journal=null}, retryWrites=true, retryReads=true
, readConcern=ReadConcern{level=null}, credential=null, transportSettings=null, commandListeners=[], codecRegistry=ProvidersCodecRegistry{codecProviders=[ValueCodecProvider{}, BsonValueCodecProvider{}, DBRefC
odecProvider{}, DBObjectCodecProvider{}, DocumentCodecProvider{}, CollectionCodecProvider{}, IterableCodecProvider{}, MapCodecProvider{}, GeoJsonCodecProvider{}, GridFSFileCodecProvider{}, Jsr310CodecProvider
{}, JsonObjectCodecProvider{}, BsonCodecProvider{}, EnumCodecProvider{}, com.mongodb.client.model.mql.ExpressionCodecProvider@1eb9a3ef, com.mongodb.Jep395RecordCodecProvider@5d318e91, com.mongodb.KotlinCodecP
rovider@298b64f7]}, loggerSettings=LoggerSettings{maxDocumentLength=1000}, clusterSettings={hosts=[mongodb-service:27017], srvServiceName=mongodb, mode=SINGLE, requiredClusterType=UNKNOWN, requiredReplicaSetN
ame='null', serverSelector='null', clusterListeners='[]', serverSelectionTimeout='30000 ms', localThreshold='15 ms'}, socketSettings=SocketSettings{connectTimeoutMS=10000, readTimeoutMS=0, receiveBufferSize=0
, proxySettings=ProxySettings{host=null, port=null, username=null, password=null}}, heartbeatSocketSettings=SocketSettings{connectTimeoutMS=10000, readTimeoutMS=10000, receiveBufferSize=0, proxySettings=Proxy
Settings{host=null, port=null, username=null, password=null}}, connectionPoolSettings=ConnectionPoolSettings{maxSize=100, minSize=0, maxWaitTimeMS=120000, maxConnectionLifeTimeMS=0, maxConnectionIdleTimeMS=0,
 maintenanceInitialDelayMS=0, maintenanceFrequencyMS=60000, connectionPoolListeners=[], maxConnecting=2}, serverSettings=ServerSettings{heartbeatFrequencyMS=10000, minHeartbeatFrequencyMS=500, serverMonitorin
gMode=AUTO, serverListeners='[]', serverMonitorListeners='[]'}, sslSettings=SslSettings{enabled=false, invalidHostNameAllowed=false, context=null}, applicationName='null', compressorList=[], uuidRepresentatio
n=JAVA_LEGACY, serverApi=null, autoEncryptionSettings=null, dnsClient=null, inetAddressResolver=null, contextProvider=null, timeoutMS=null}
2024-12-19T11:29:35.419Z  INFO 1 --- [getBookAPI] [b-service:27017] org.mongodb.driver.cluster       : Monitor thread successfully connected to server with description ServerDescription{address=mongod
b-service:27017, type=STANDALONE, cryptd=false, state=CONNECTED, ok=true, minWireVersion=0, maxWireVersion=17, maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30, roundTripTimeNanos=62568665, minRoundT
ripTimeNanos=0}
2024-12-19T11:29:36.525Z  INFO 1 --- [getBookAPI] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path '/'
2024-12-19T11:29:36.556Z  INFO 1 --- [getBookAPI] [           main] c.k.getBook.GetBookApiApplication        : Started GetBookApiApplication in 4.768 seconds (process running for 5.587)
(base) owira@x1yoga-phn87b2:~/projects/verification/sprinBoot-backend$
```

5. Port Forwarding

To access the services locally:

- MongoDB:

```
kubectl port-forward service/mongodb-service 27017:27017
```

```
mongodb                 | {"t":{"$date":"2024-12-19T11:36:04.661+00:00"},"s":"I",  "c":"-",        "id":20883,   "ctx":"conn1","msg":"Interrupted operation as its client disconnected","attr":{"opId":14137}}
mongodb                 | {"t":{"$date":"2024-12-19T11:36:04.661+00:00"},"s":"I",  "c":"NETWORK",  "id":22944,   "ctx":"conn2","msg":"Connection ended","attr":{"remote":"172.21.0.3:34176","uuid":"5a41fb62-6
30-417b-89c4-35b770232c84","connectionId":2,"connectionCount":1}}
mongodb                 | {"t":{"$date":"2024-12-19T11:36:04.661+00:00"},"s":"I",  "c":"NETWORK",  "id":22944,   "ctx":"conn3","msg":"Connection ended","attr":{"remote":"172.21.0.3:53080","uuid":"d854234b-2
22-4ae5-8247-154d65d63422","connectionId":3,"connectionCount":2}}
mongodb                 | {"t":{"$date":"2024-12-19T11:36:04.663+00:00"},"s":"I",  "c":"NETWORK",  "id":22944,   "ctx":"conn1","msg":"Connection ended","attr":{"remote":"172.21.0.3:34174","uuid":"eef5ab6f-6
87-4b28-be79-93d3a15b2f6d","connectionId":1,"connectionCount":0}}
sprinboot-backend-app-1 exited with code 137
mongodb                 | {"t":{"$date":"2024-12-19T11:36:19.601+00:00"},"s":"I",  "c":"WTCHKPT",  "id":22430,   "ctx":"Checkpointer","msg":"WiredTiger message","attr":{"message":{"ts_sec":1734608179,"ts_us
c":601363,"thread":"1:0x7fe5faaef640","session_name":"WT_SESSION.checkpoint","category":"WT_VERB_CHECKPOINT_PROGRESS","category_id":6,"verbose_level":"DEBUG","verbose_level_id":1,"msg":"saving checkpoint sna
shot min: 91, snapshot max: 91 snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 1"}}}
mongodb exited with code 137
(base) owira@x1yoga-phn87b2:~/projects/verification/sprinBoot-backend$ kubectl port-forward service/mongodb-service 27017:27017
Forwarding from 127.0.0.1:27017 -> 27017
Forwarding from [::1]:27017 -> 27017
Handling connection for 27017
Handling connection for 27017
Handling connection for 27017
Handling connection for 27017
Handling connection for 27017
Handling connection for 27017
Handling connection for 27017
Handling connection for 27017
```

- Spring Boot:

```
kubectl port-forward service/springboot-backend-service 8080:8080
```

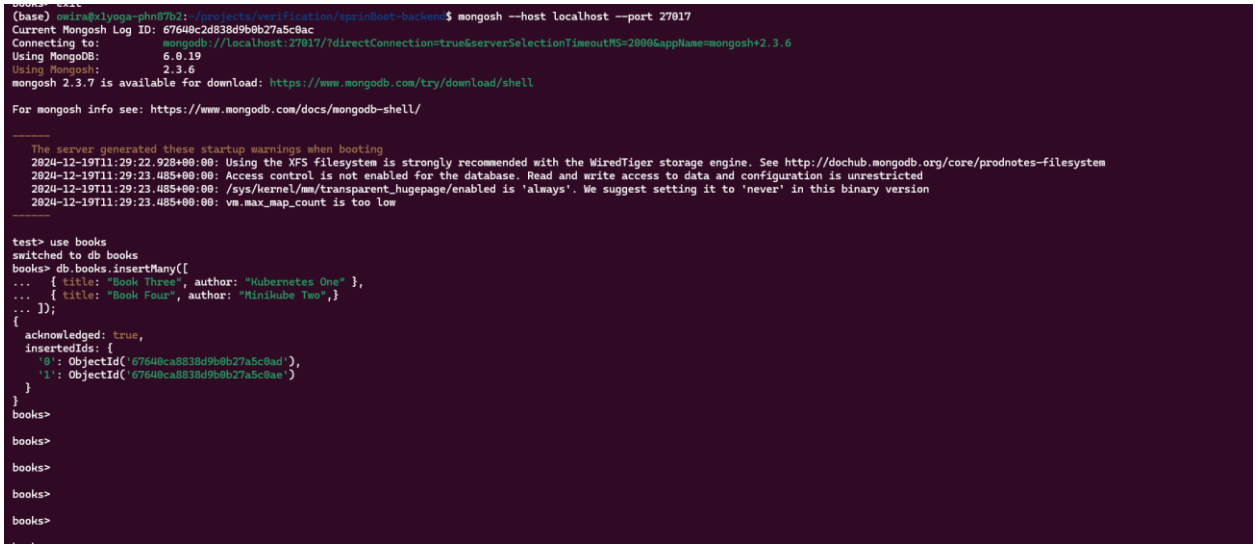
```
(base) owira@x1yoga-phn87b2:~/projects/verification/sprinBoot-backend$ minikube ip
192.168.49.2
(base) owira@x1yoga-phn87b2:~/projects/verification/sprinBoot-backend$ kubectl port-forward service/springboot-backend-service 8080:8080
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
Handling connection for 8080
Handling connection for 8080
```

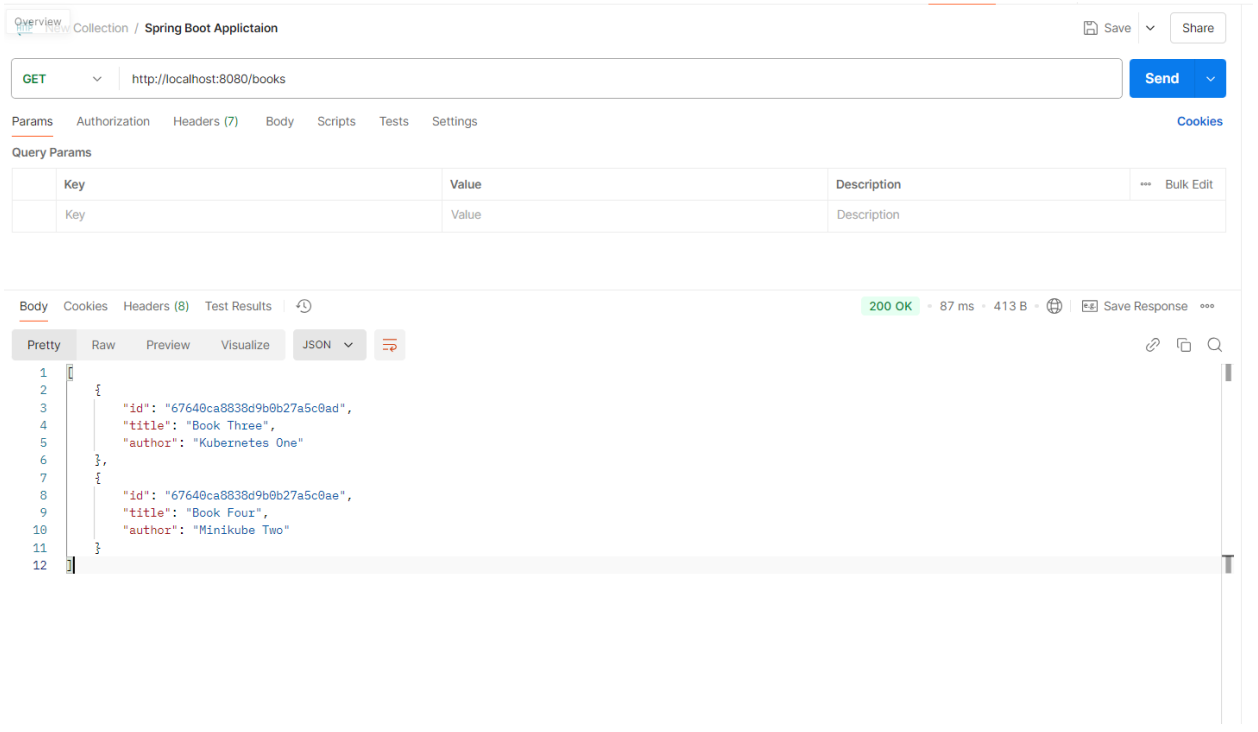
6. Add Sample Data to MongoDB

Follow the steps outlined in the Local Setup section to add sample data to MongoDB via port-forwarded access.

```
use books;
db.books.insertMany([
  { title: "Book Three", author: "Kubernetes One" },

  { title: "Book Four", author: "Minikube Two",}
]);
```





---

4. CI/CD Pipeline with GitHub Actions

Overview

The CI/CD pipeline automates building and deploying the Docker image for the Spring Boot application.

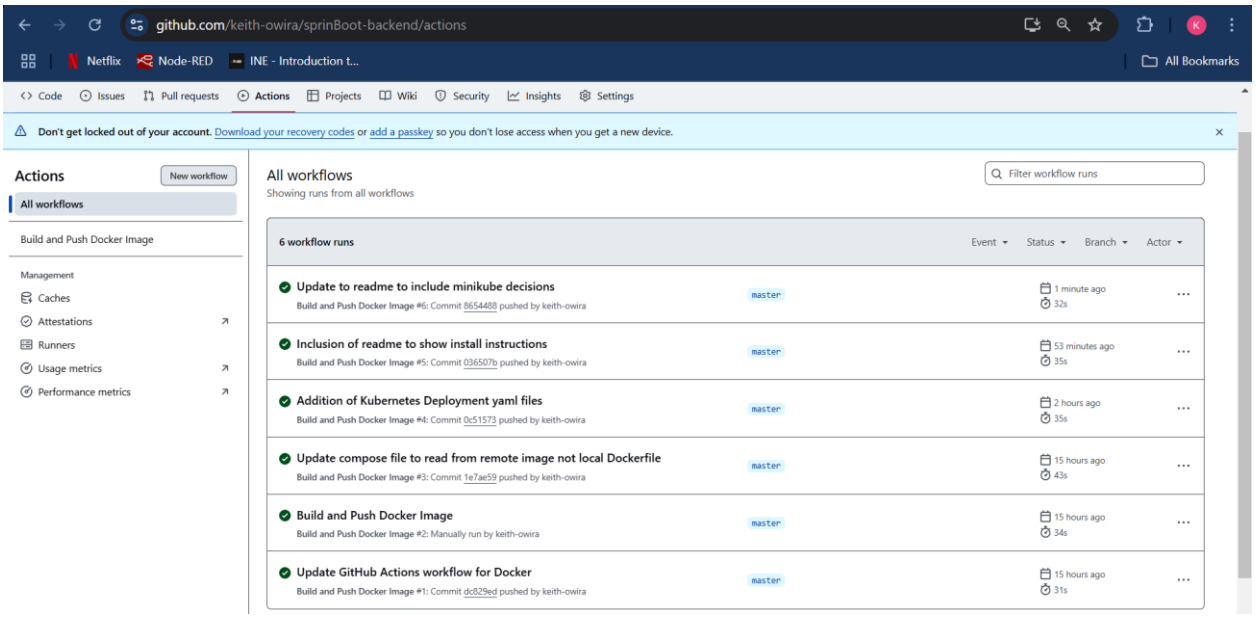
Workflow Steps

1. **Trigger:** Runs on every push or pull request to the `main` branch.
2. **Login to Docker Hub:** Uses GitHub secrets for secure authentication.
3. **Build and Push Docker Image:**

```
- name: Build and push Docker image
      uses: docker/build-push-action@v4
       with:
       push: true
       tags: owira57/springboot-backend-api:latest
```

Benefits

- Automates the image creation process.
- Ensures the latest application version is always available on Docker Hub.



---

6. Decisions, Assumptions, and Challenges

Key Decisions

- **ClusterIP for MongoDB:** Simplified internal communication within the Kubernetes cluster.
- **Separate the dependency and service configurations into distinct YAML files for Spring Boot and MongoDB in Kubernetes**: Deployment Flexibility: Allows independent application of configurations :MongoDB can be deployed and verified first, ensuring the database is ready before starting the Spring Boot application**.**

Assumptions

- MongoDB uses a temporary persistent volume (`emptyDir`).
- Endpoints were pre-tested locally before deployment.

Challenges

1. **Database Connectivity:** Ensuring the MongoDB URI is correctly configured.
2. **Port Mapping:** Avoiding port conflicts during port-forwarding by disabling all local host running instances