Lemay AI Assessment

Farid Faraji

01/10/2023

Building the images:

Using **docker-compose build** to build the images of the **producer_app** and **consumer_app** services

Running the containers:

Using the **docker-compose up** to run the docker containers.

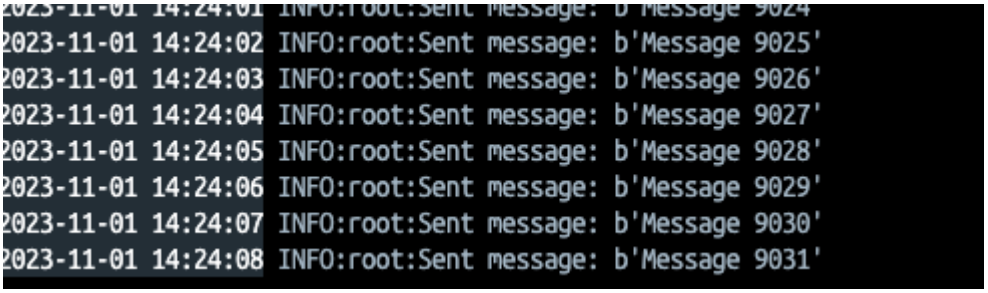
```
devops on  main [!] on  v20.10.23 using  default/iron-burner-389219 via  base
➜ docker-compose up -d

[+] Running 6/6
 ⠿ Container devops-neo4j-1          Started
 ⠿ Container zookeeper               Started
 ⠿ Container elasticsearch           Started
 ⠿ Container kafka                   Started
 ⠿ Container devops-consumer_app-1   Started
 ⠿ Container devops-producer_app-1   Started
```

Here are the running containers

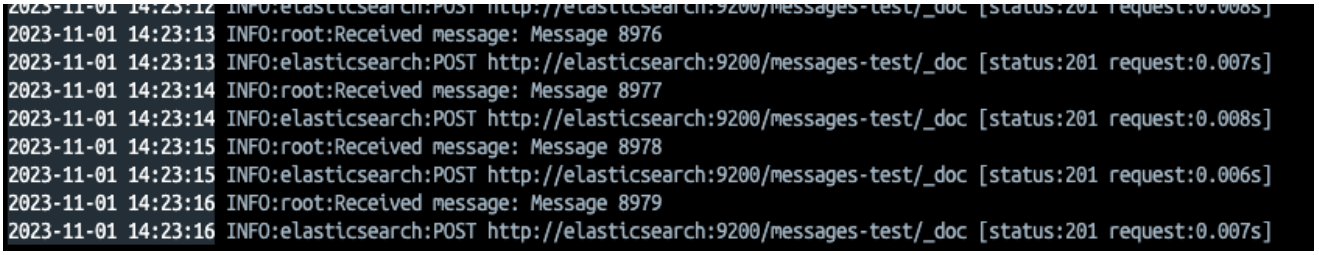
```
devops on  main [!] on  v20.10.23 runs  DDDEKZ using  default/iron-burner-389219 via  base
➜ docker container ls
CONTAINER ID   IMAGE                                              COMMAND                  CREATED          STATUS          PORTS                                                  NAMES
281425b0bb60   devops-producer_app                                "python producer_app…"   35 minutes ago   Up 20 seconds                                                          devops-producer_app-1
da0e94ff01d4   devops-consumer_app                                "python consumer_app…"   35 minutes ago   Up 20 seconds                                                          devops-consumer_app-1
0f0280b889ce   neo4j:4.2.3                                        "/sbin/tini -g -- /d…"   35 minutes ago   Up 20 seconds   0.0.0.0:7474->7474/tcp, 7473/tcp, 0.0.0.0:7687->7687/tcp   devops-neo4j-1
9e88fa7af03b   docker.elastic.co/elasticsearch/elasticsearch:7.10.2   "/tini -- /usr/local…"   39 minutes ago   Up 20 seconds   0.0.0.0:9200->9200/tcp, 9300/tcp                       elasticsearch
8b1d18e605c5   confluentinc/cp-kafka:latest                       "/etc/confluent/dock…"   40 minutes ago   Up 20 seconds   9092/tcp, 0.0.0.0:9093->9093/tcp                       kafka
3fb07511957d   confluentinc/cp-zookeeper:latest                   "/etc/confluent/dock…"   41 minutes ago   Up 20 seconds   2888/tcp, 0.0.0.0:2181->2181/tcp, 3888/tcp             zookeeper
```

Name	Image	Status	Port(s)	Last started	Actions
devops	-	Running (6/6)		2 minutes ago	
zookeeper 3fb07511957d	confluentinc/cp-zookeeper:latest	Running	2181:2181	2 minutes ago	
kafka 8b1d18e605c5	confluentinc/cp-kafka:latest	Running	9093:9093	2 minutes ago	
elasticsearch 9e88fa7af03b	docker.elastic.co/elasticsearch/elasticsearch:7.10.2	Running	9200:9200	2 minutes ago	
neo4j-1 0f0280b889ce AMD64	neo4j:4.2.3	Running	7474:7474 Show all ports (2)	2 minutes ago	
consumer_app-1 da0e94ff01d4	devops-consumer_app	Running		2 minutes ago	
producer_app-1 281425b0bb60	devops-producer_app	Running		2 minutes ago	

The producer service creates the message sending it through Kafka to be consumed by the consumer service.
Here are the logs from the producer service.



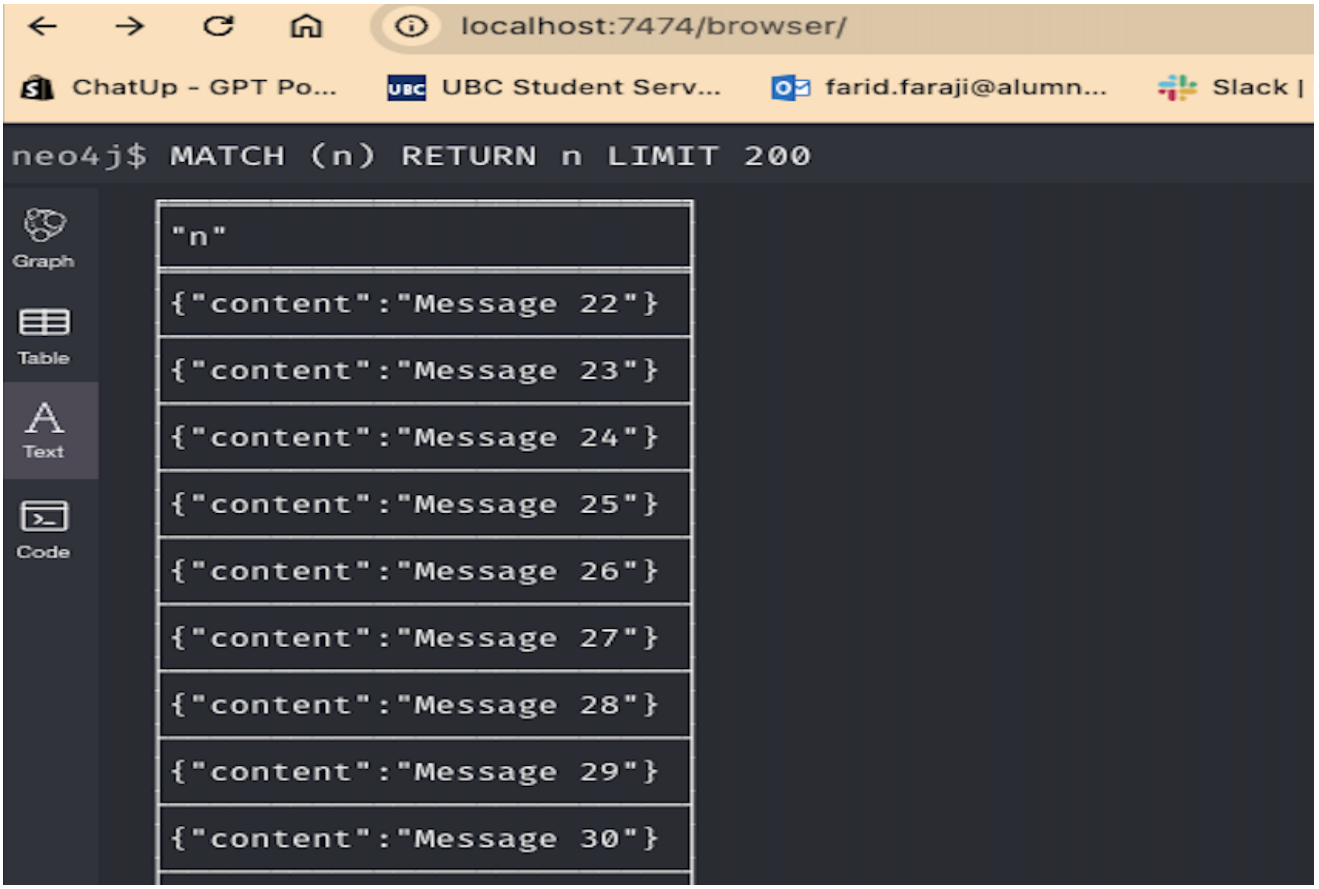
And here are the logs from the consumer service.

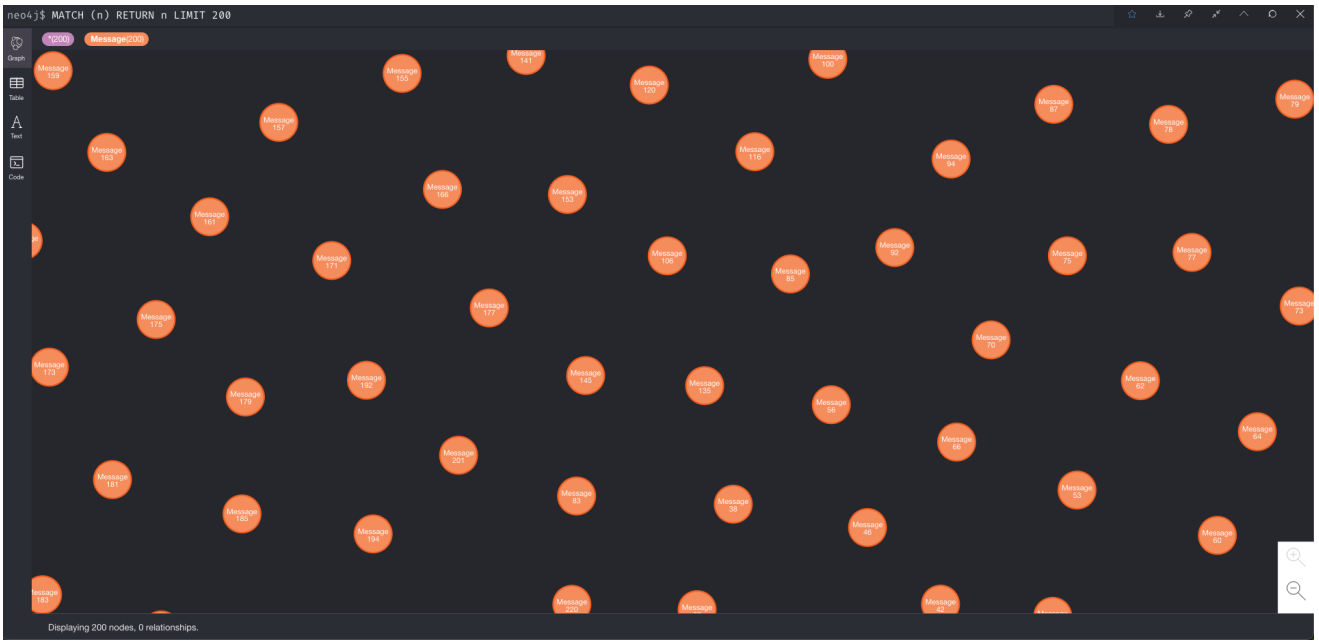


ElasticSearch and Neo4j

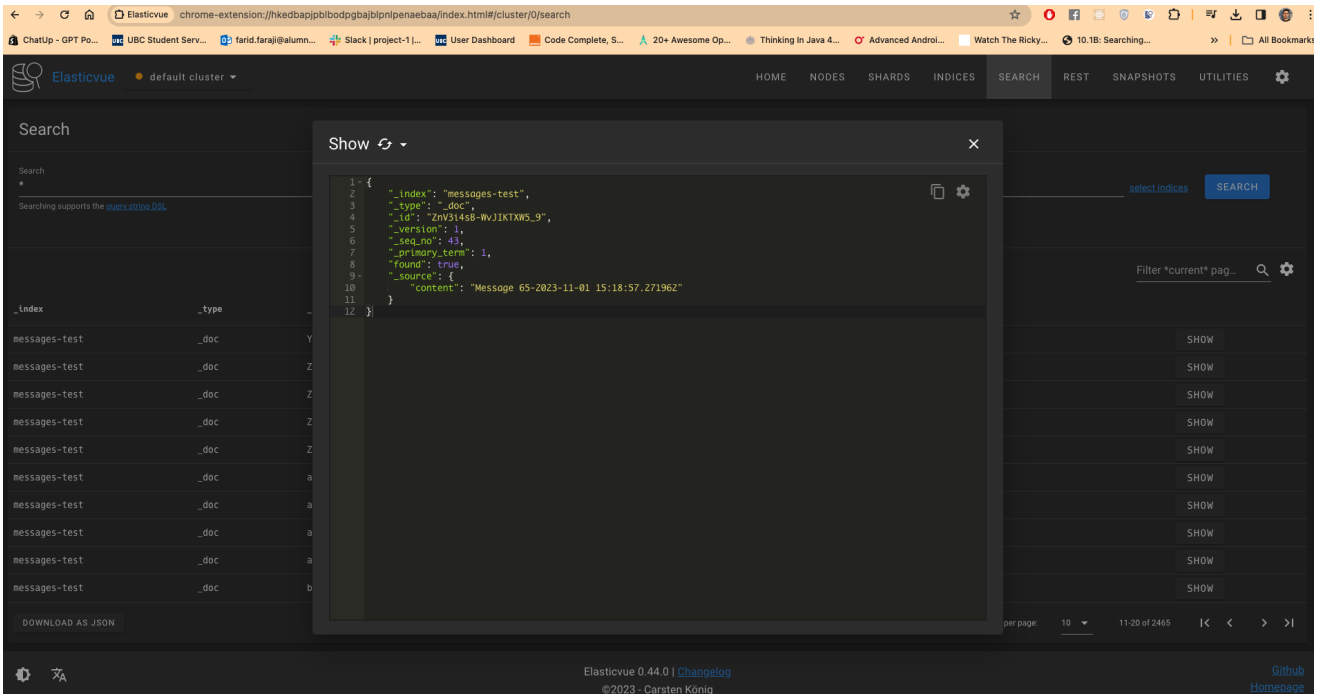
The consumer service then inserts the messages into the ElasticSearch and the Neo4j services.

Here are examples of the data being populated in Neo4j





Below shows an example of content in the elasticsearch index being populated by the consumer service.

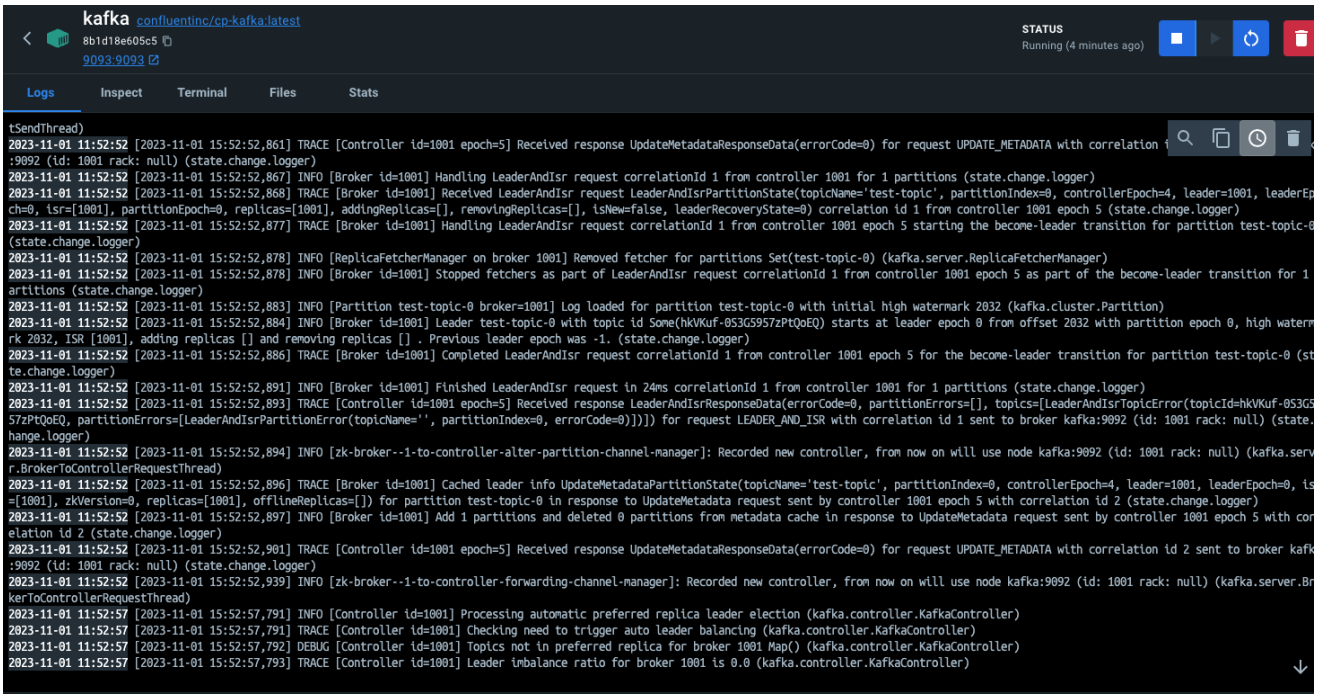


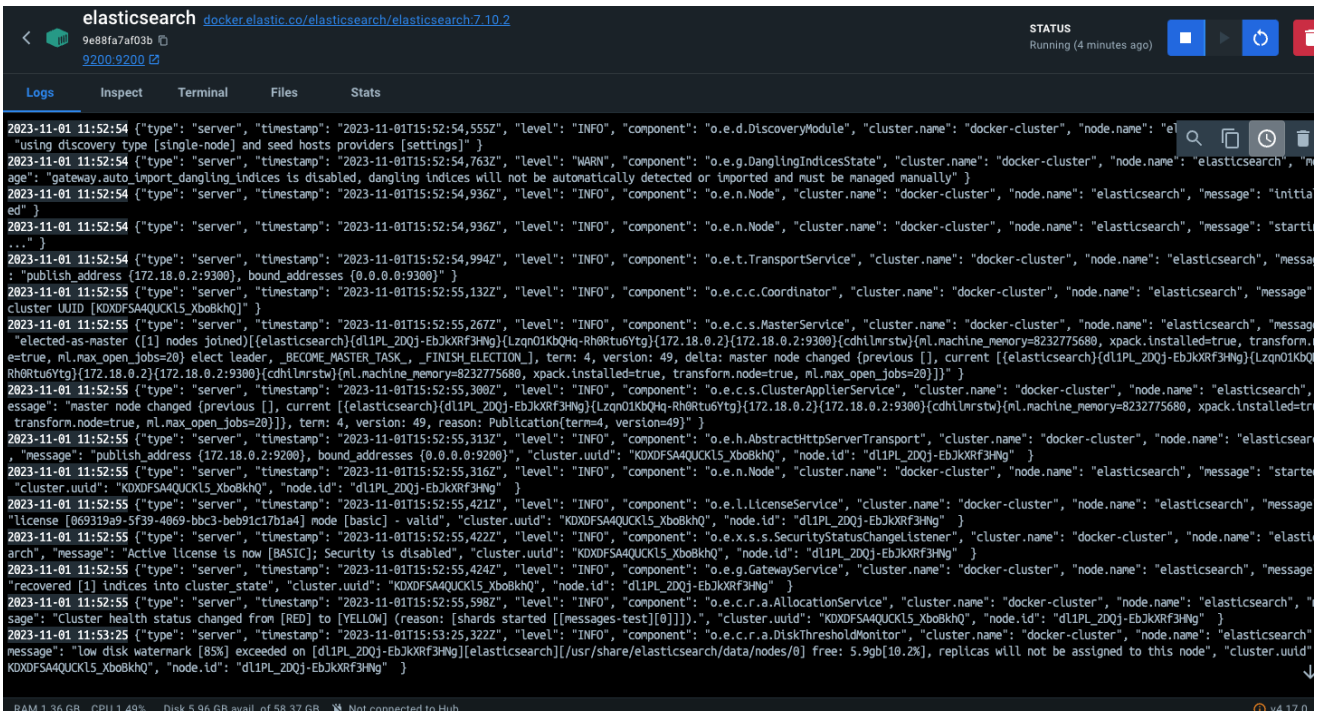
Logs Usage:

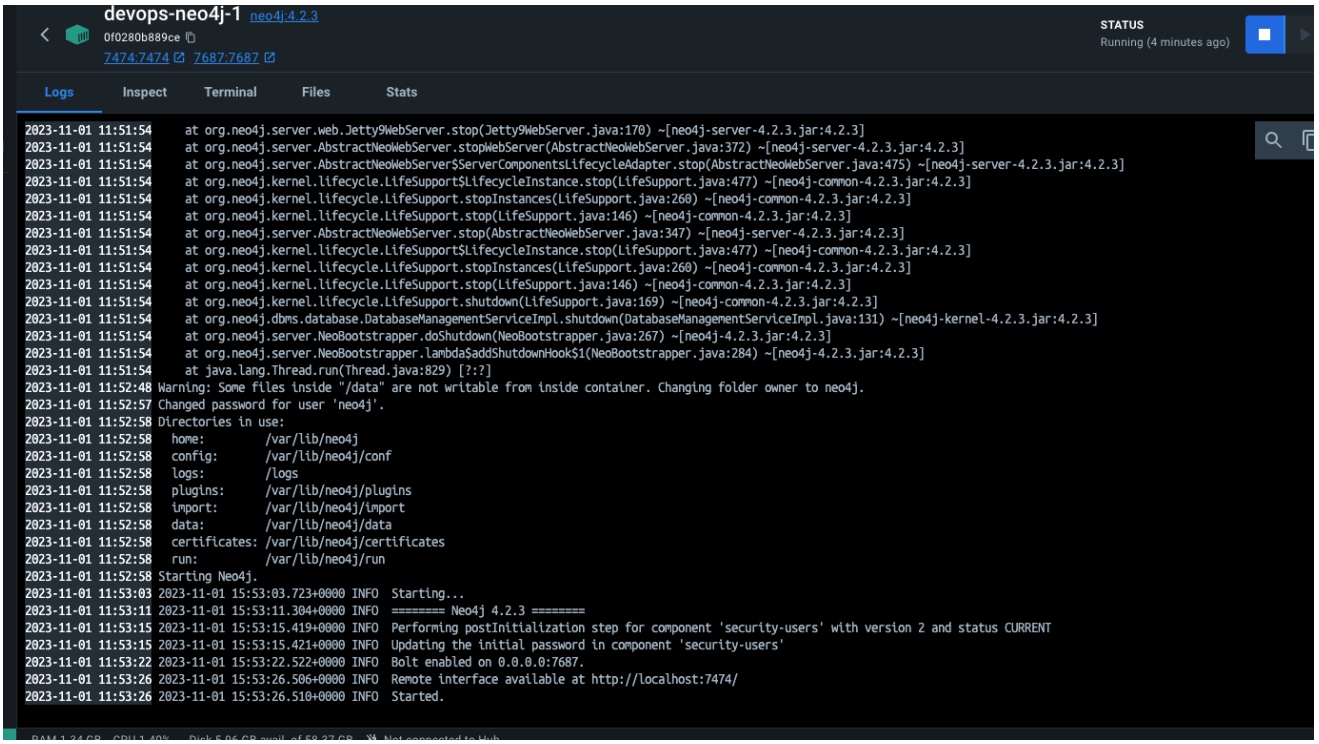
The log files from each container play a critical role in troubleshooting issues within the pipeline. Consider an instance where unexpected data is detected in the Neo4j graph.

In such a case, it is standard procedure to examine the consumer service's log files to confirm the integrity of the messages being recorded. If the messages are not as anticipated, the investigation would proceed to review the producer's log files to determine the content being generated.

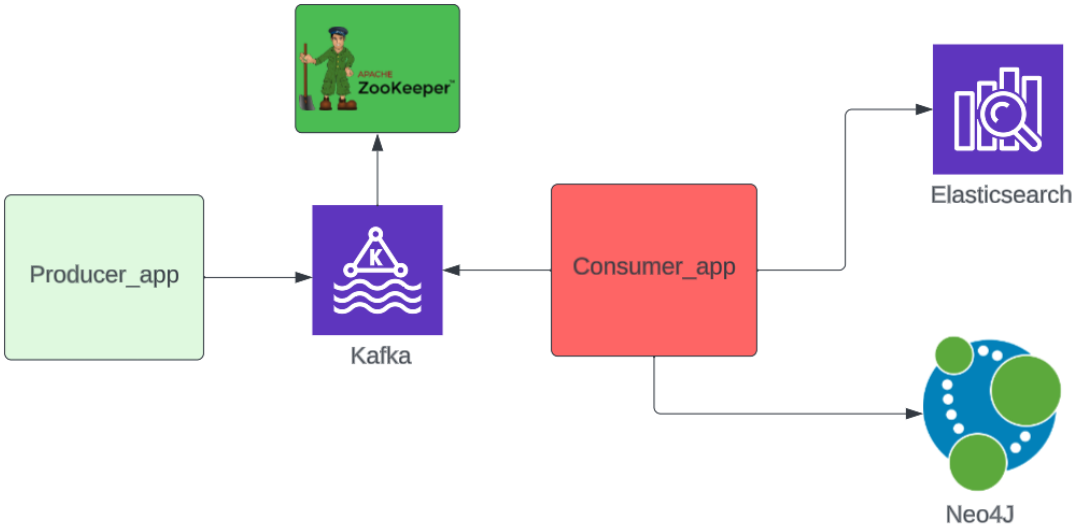
Should an error or complication arise within Kafka or any ancillary service, the logs provide essential diagnostics. By analyzing these logs, one can identify the issue and take appropriate measures to address it.






devops-neo4j-1
neo4j:4.2.3
0f0280b889ce
7474:7474
7687:7687
STATUS
Running (4 minutes ago)
Logs
Inspect
Terminal
Files
Stats
2023-11-01 11:51:54 at org.neo4j.server.web.Jetty9WebServer.stop(Jetty9WebServer.java:170) ~[neo4j-server-4.2.3.jar:4.2.3]
2023-11-01 11:51:54 at org.neo4j.server.AbstractNeoWebServer.stopWebServer(AbstractNeoWebServer.java:372) ~[neo4j-server-4.2.3.jar:4.2.3]
2023-11-01 11:51:54 at org.neo4j.server.AbstractNeoWebServer$ServerComponentsLifecycleAdapter.stop(AbstractNeoWebServer.java:475) ~[neo4j-server-4.2.3.jar:4.2.3]
2023-11-01 11:51:54 at org.neo4j.kernel.lifecycle.LifeSupport$LifecycleInstance.stop(LifeSupport.java:477) ~[neo4j-common-4.2.3.jar:4.2.3]
2023-11-01 11:51:54 at org.neo4j.kernel.lifecycle.LifeSupport.stopInstances(LifeSupport.java:260) ~[neo4j-common-4.2.3.jar:4.2.3]
2023-11-01 11:51:54 at org.neo4j.kernel.lifecycle.LifeSupport.stop(LifeSupport.java:146) ~[neo4j-common-4.2.3.jar:4.2.3]
2023-11-01 11:51:54 at org.neo4j.server.AbstractNeoWebServer.stop(AbstractNeoWebServer.java:347) ~[neo4j-server-4.2.3.jar:4.2.3]
2023-11-01 11:51:54 at org.neo4j.kernel.lifecycle.LifeSupport$LifecycleInstance.stop(LifeSupport.java:477) ~[neo4j-common-4.2.3.jar:4.2.3]
2023-11-01 11:51:54 at org.neo4j.kernel.lifecycle.LifeSupport.stopInstances(LifeSupport.java:260) ~[neo4j-common-4.2.3.jar:4.2.3]
2023-11-01 11:51:54 at org.neo4j.kernel.lifecycle.LifeSupport.stop(LifeSupport.java:146) ~[neo4j-common-4.2.3.jar:4.2.3]
2023-11-01 11:51:54 at org.neo4j.kernel.lifecycle.LifeSupport.shutdown(LifeSupport.java:169) ~[neo4j-common-4.2.3.jar:4.2.3]
2023-11-01 11:51:54 at org.neo4j.dbms.database.DatabaseManagementServiceImpl.shutdown(DatabaseManagementServiceImpl.java:131) ~[neo4j-kernel-4.2.3.jar:4.2.3]
2023-11-01 11:51:54 at org.neo4j.server.NeoBootstrapper.doShutdown(NeoBootstrapper.java:267) ~[neo4j-4.2.3.jar:4.2.3]
2023-11-01 11:51:54 at org.neo4j.server.NeoBootstrapper.lambda$addShutdownHook$1(NeoBootstrapper.java:284) ~[neo4j-4.2.3.jar:4.2.3]
2023-11-01 11:51:54 at java.lang.Thread.run(Thread.java:829) [?:?]
2023-11-01 11:52:48 Warning: Some files inside "/data" are not writable from inside container. Changing folder owner to neo4j.
2023-11-01 11:52:57 Changed password for user 'neo4j'.
2023-11-01 11:52:58 Directories in use:
2023-11-01 11:52:58 home: /var/lib/neo4j
2023-11-01 11:52:58 config: /var/lib/neo4j/conf
2023-11-01 11:52:58 logs: /logs
2023-11-01 11:52:58 plugins: /var/lib/neo4j/plugins
2023-11-01 11:52:58 import: /var/lib/neo4j/import
2023-11-01 11:52:58 data: /var/lib/neo4j/data
2023-11-01 11:52:58 certificates: /var/lib/neo4j/certificates
2023-11-01 11:52:58 run: /var/lib/neo4j/run
2023-11-01 11:52:58 Starting Neo4j.
2023-11-01 11:53:03 2023-11-01 15:53:03.723+0000 INFO Starting...
2023-11-01 11:53:11 2023-11-01 15:53:11.304+0000 INFO ======== Neo4j 4.2.3 ========
2023-11-01 11:53:15 2023-11-01 15:53:15.419+0000 INFO Performing postInitialization step for component 'security-users' with version 2 and status CURRENT
2023-11-01 11:53:15 2023-11-01 15:53:15.421+0000 INFO Updating the initial password in component 'security-users'
2023-11-01 11:53:22 2023-11-01 15:53:22.522+0000 INFO Bolt enabled on 0.0.0.0:7687.
2023-11-01 11:53:26 2023-11-01 15:53:26.506+0000 INFO Remote interface available at http://localhost:7474/
2023-11-01 11:53:26 2023-11-01 15:53:26.510+0000 INFO Started.

Explaining What the code does:



This project is composed of a suite of six microservices, exemplifying the quintessential consumer-producer paradigm. Within this architecture, one microservice is responsible for generating messages, while another is tasked with processing those messages. The system leverages a robust messaging broker—specifically, Kafka, coupled with ZooKeeper—for efficient message handling, ensuring the system's scalability and its capability to manage substantial message sizes. This configuration enables the system to proficiently process a high volume of sizeable requests.

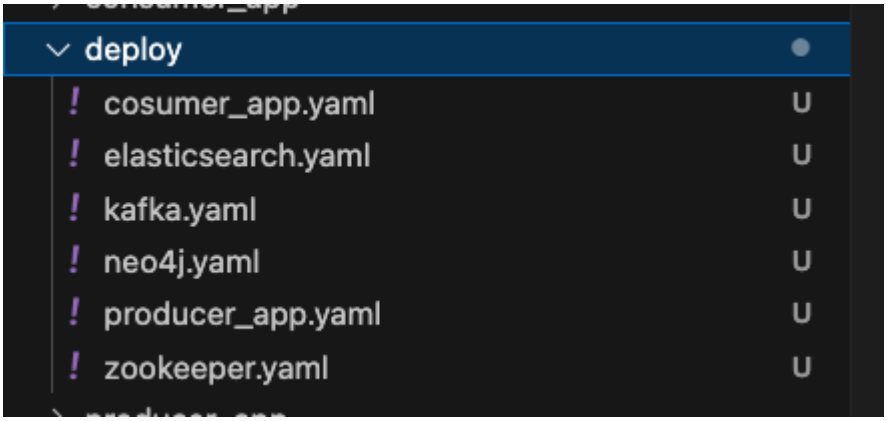
Upon reception, the consumer microservice archives the incoming messages from Kafka into a Neo4j graph database, which allows for sophisticated relationship-driven data insights. Concurrently, Elasticsearch is employed to facilitate comprehensive full-text search capabilities and to perform complex data analytics.

Using GKE and GCR instead of docker-compose:

In order to run the services in a kubernetes cluster (using GKE in my example)
We will still need to build the docker images and in our case we tag the images with the correct tag to be pushed into GCR then kubernetes will pull the images from the GCR.

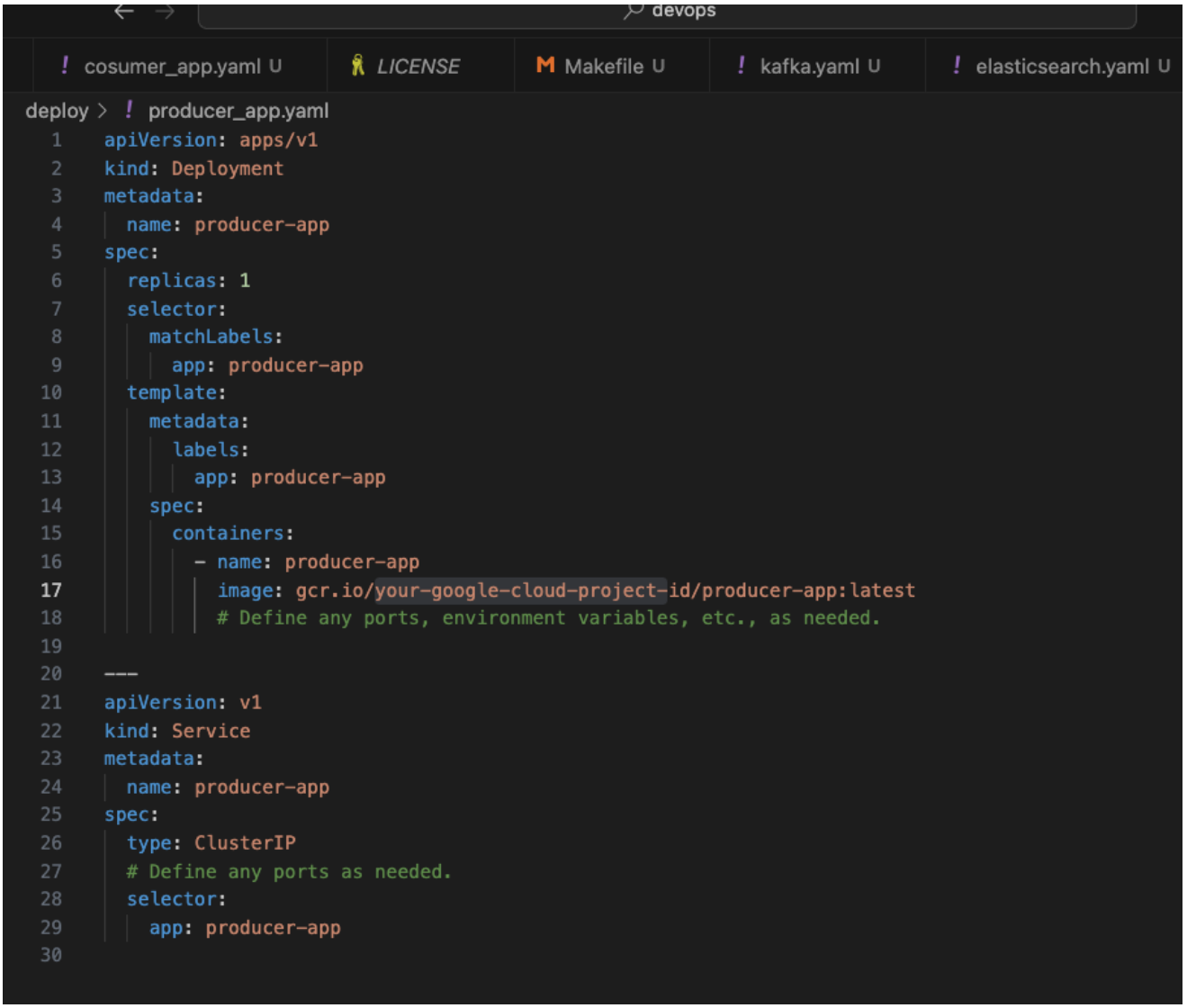
We will have to write a kubernetes manifest for each service. You can see screenshots of these manifests shown in the next page

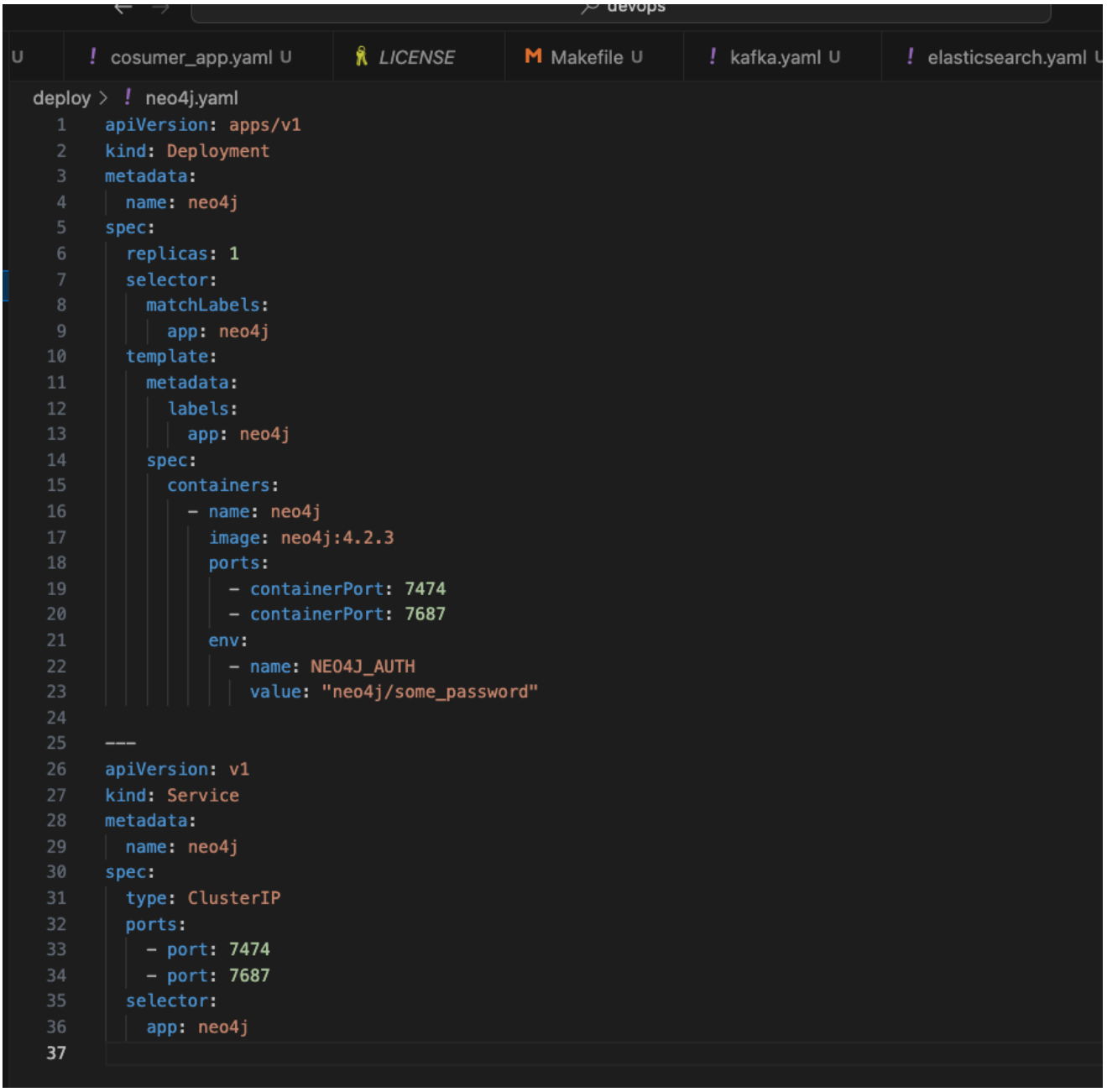
```
# Project configuration
PROJECT_NAME =

ifeq ($(shell uname -p), arm)
DOCKER_PLATFORM = --platform linux/amd64
else
DOCKER_PLATFORM =
endif

default: help

help: # Display help
    @awk -F ':|##' \
        '/^[^\t].+?:.*?##/ {\
            printf "\033[36m%-30s\033[0m %s\n", $$1, $$NF \
        }' $(MAKEFILE_LIST) | sort

build-docker: ## Build the docker images
    docker build $(DOCKER_PLATFORM) -t producer-app -f Dockerfile-producer .
    docker build $(DOCKER_PLATFORM) -t consumer-app -f Dockerfile-consumer .

tag-docker: ## Tag the docker images
    docker tag producer-app gcr.io/lemay-project/producer-app:latest
    docker tag consumer-app gcr.io/lemay-project/consumer-app:latest

push-docker: ## push the images to registry
    docker push gcr.io/lemay-project/producer-app:latest
    docker push gcr.io/lemay-project/consumer-app:latest

k8-cluster: ## deploy service to cluster
    kubectl apply -f deploy/zookeeper.yaml
    kubectl apply -f deploy/kafka.yaml
    kubectl apply -f deploy/elastcsearch.yaml
    kubectl apply -f deploy/Neo4j.yaml
    kubectl apply -f deploy/producer.yaml
    kubectl apply -f deploy/consumer.yaml
```

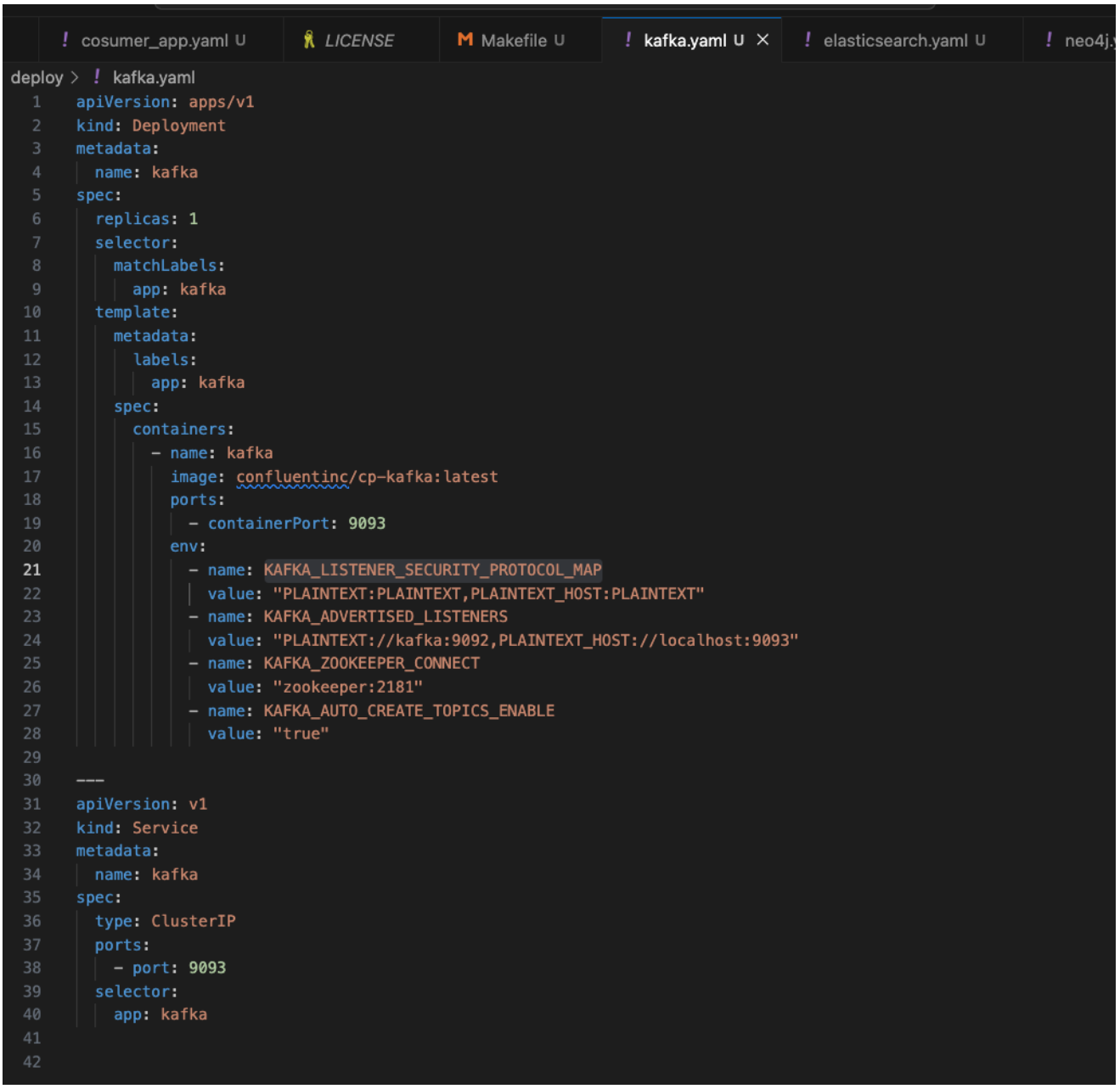


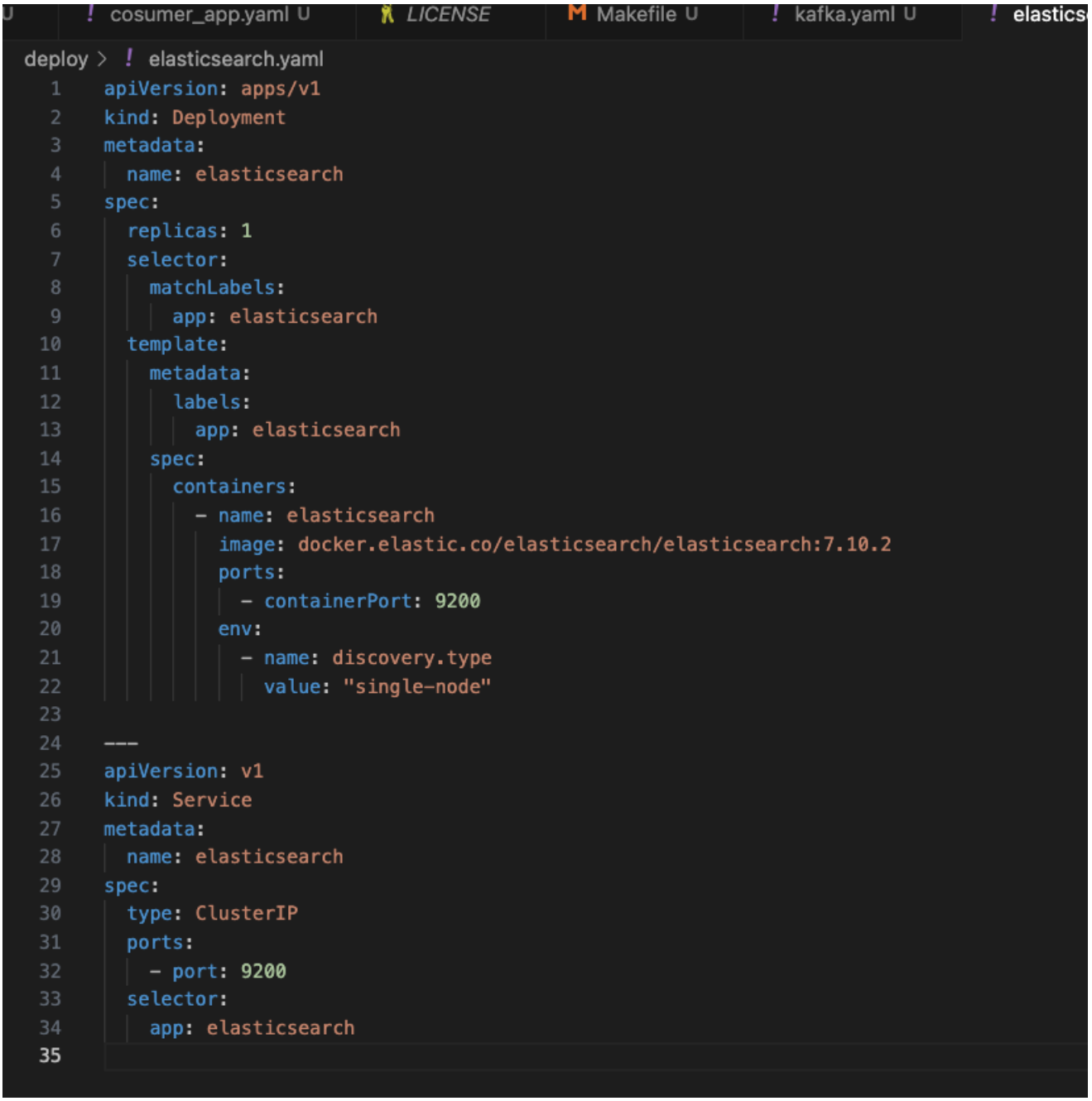
s.

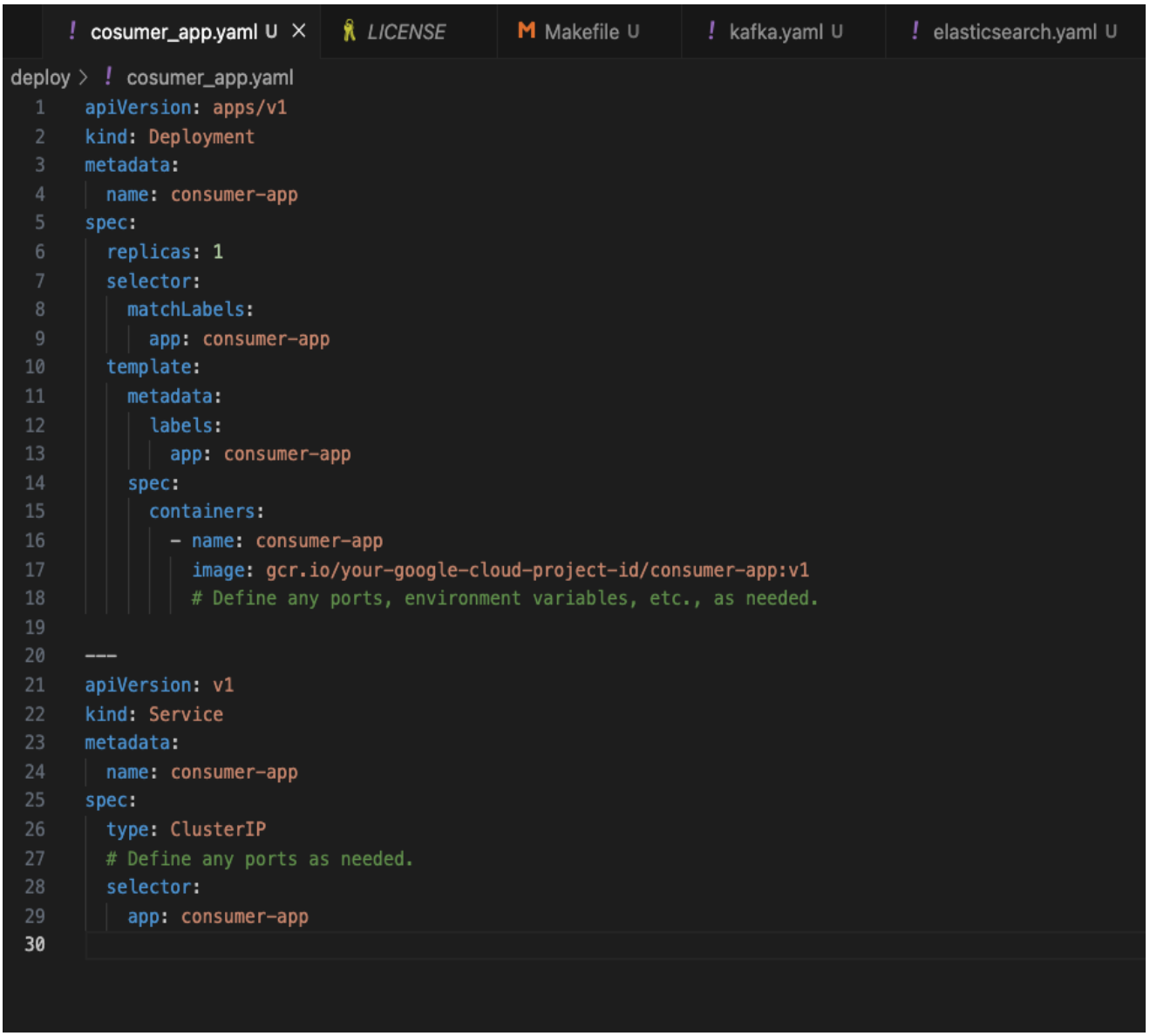

! zookeeper.yaml U
! cosumer_app.yaml U
LICENSE
Makefile U
deploy > ! zookeeper.yaml
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4 name: zookeeper
5 spec:
6 replicas: 1
7 selector:
8 matchLabels:
9 app: zookeeper
10 template:
11 metadata:
12 labels:
13 app: zookeeper
14 spec:
15 containers:
16 - name: zookeeper
17 image: confluentinc/cp-zookeeper:latest
18 ports:
19 - containerPort: 2181
20 env:
21 - name: ZOOKEEPER_CLIENT_PORT
22 value: "2181"
23
24 ---
25 apiVersion: v1
26 kind: Service
27 metadata:
28 name: zookeeper
29 spec:
30 type: ClusterIP
31 ports:
32 - port: 2181
33 selector:
34 app: zookeeper
35


devops
cosumer_app.yaml U
LICENSE
Makefile U
kafka.yaml U
elasticsearch.yaml U
deploy > producer_app.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
name: producer-app
spec:
replicas: 1
selector:
matchLabels:
app: producer-app
template:
metadata:
labels:
app: producer-app
spec:
containers:
- name: producer-app
image: gcr.io/your-google-cloud-project-id/producer-app:latest
Define any ports, environment variables, etc., as needed.
---
apiVersion: v1
kind: Service
metadata:
name: producer-app
spec:
type: ClusterIP
Define any ports as needed.
selector:
app: producer-app


devops
U
cosumer_app.yaml U
LICENSE
Makefile U
kafka.yaml U
elasticsearch.yaml U
deploy > neo4j.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
name: neo4j
spec:
replicas: 1
selector:
matchLabels:
app: neo4j
template:
metadata:
labels:
app: neo4j
spec:
containers:
- name: neo4j
image: neo4j:4.2.3
ports:
- containerPort: 7474
- containerPort: 7687
env:
- name: NEO4J_AUTH
value: "neo4j/some_password"
---
apiVersion: v1
kind: Service
metadata:
name: neo4j
spec:
type: ClusterIP
ports:
- port: 7474
- port: 7687
selector:
app: neo4j


! cosumer_app.yaml U
LICENSE
M Makefile U
! kafka.yaml U
! elasticsearch.yaml U
deploy > ! kafka.yaml
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4 name: kafka
5 spec:
6 replicas: 1
7 selector:
8 matchLabels:
9 app: kafka
10 template:
11 metadata:
12 labels:
13 app: kafka
14 spec:
15 containers:
16 - name: kafka
17 image: confluentinc/cp-kafka:latest
18 ports:
19 - containerPort: 9093
20 env:
21 - name: KAFKA_LISTENER_SECURITY_PROTOCOL_MAP
22 value: "PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT"
23 - name: KAFKA_ADVERTISED_LISTENERS
24 value: "PLAINTEXT://kafka:9092,PLAINTEXT_HOST://localhost:9093"
25 - name: KAFKA_ZOOKEEPER_CONNECT
26 value: "zookeeper:2181"
27 - name: KAFKA_AUTO_CREATE_TOPICS_ENABLE
28 value: "true"
29
30 ---
31 apiVersion: v1
32 kind: Service
33 metadata:
34 name: kafka
35 spec:
36 type: ClusterIP
37 ports:
38 - port: 9093
39 selector:
40 app: kafka
41
42


cosumer_app.yaml U
LICENSE
Makefile U
kafka.yaml U
deploy > ! elasticsearch.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
name: elasticsearch
spec:
replicas: 1
selector:
matchLabels:
app: elasticsearch
template:
metadata:
labels:
app: elasticsearch
spec:
containers:
- name: elasticsearch
image: docker.elastic.co/elasticsearch/elasticsearch:7.10.2
ports:
- containerPort: 9200
env:
- name: discovery.type
value: "single-node"
---
apiVersion: v1
kind: Service
metadata:
name: elasticsearch
spec:
type: ClusterIP
ports:
- port: 9200
selector:
app: elasticsearch


cosumer_app.yaml U
LICENSE
Makefile U
kafka.yaml U
elasticsearch.yaml U
deploy > cosumer_app.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
name: consumer-app
spec:
replicas: 1
selector:
matchLabels:
app: consumer-app
template:
metadata:
labels:
app: consumer-app
spec:
containers:
- name: consumer-app
image: gcr.io/your-google-cloud-project-id/consumer-app:v1
Define any ports, environment variables, etc., as needed.
---
apiVersion: v1
kind: Service
metadata:
name: consumer-app
spec:
type: ClusterIP
Define any ports as needed.
selector:
app: consumer-app