**e-Claim POC Setup Guide**

This document provides detailed steps for setting up, testing, and troubleshooting the e-Claim Proof of Concept (POC) application.

**1. Clone or Unzip the Project**

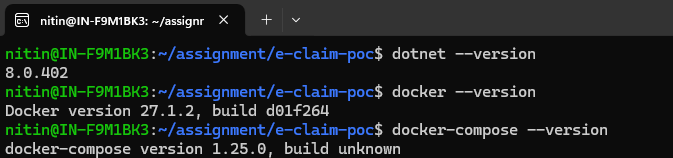
* **Step**: Obtain the project from the repository or as a compressed file.
* **Command**:   
  **git clone** [**https://github.com/nitinrahane/e-claim-poc.git**](https://github.com/nitinrahane/e-claim-poc.git)

or **unzip** the provided archive of **e-claim-poc**.

**2. Navigate to the Project Directory**

* **Command**: **cd e-claim-poc**

**3. Install Prerequisites**

* Ensure the following are installed:
  + **.NET 8 Runtime**: Verify with dotnet --version.
  + **Docker** and **Docker-Compose**: Verify with docker --version and docker-compose --version.
  + 

**4. Run Docker-Compose**

* **Command**: **docker-compose up –build**
* A computer screen with white text

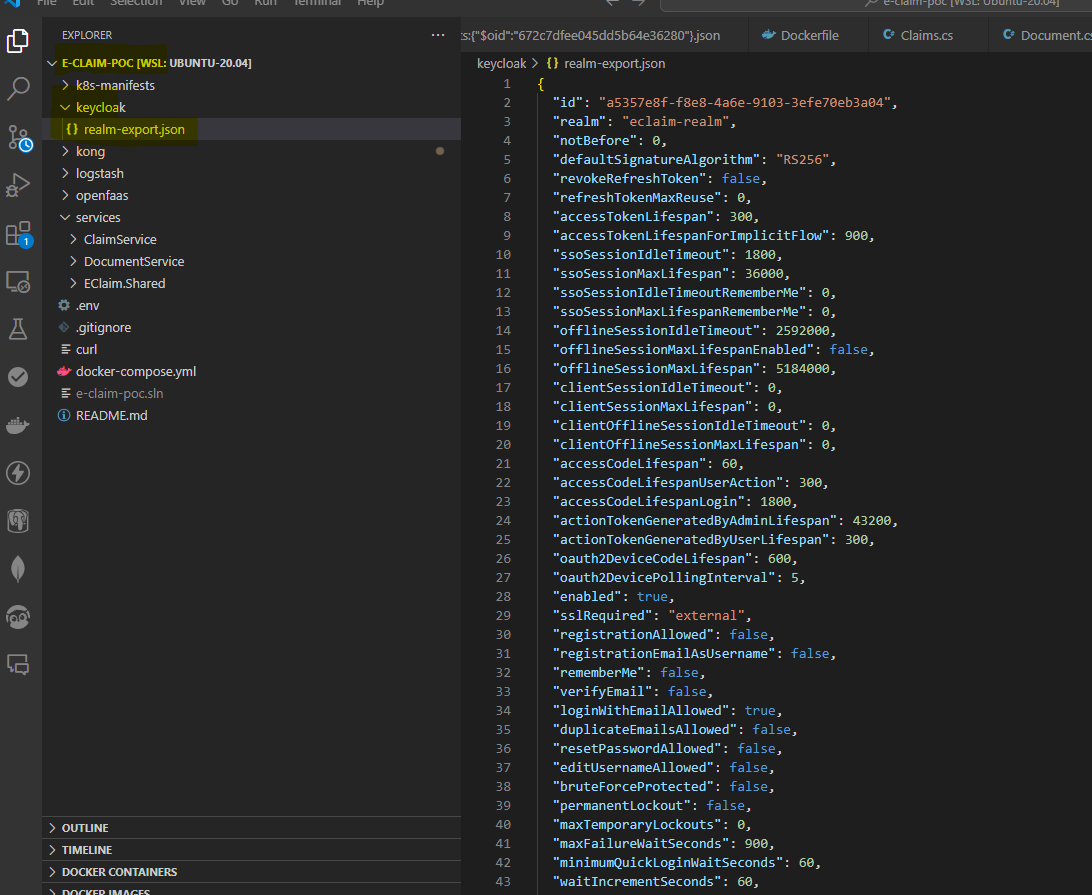
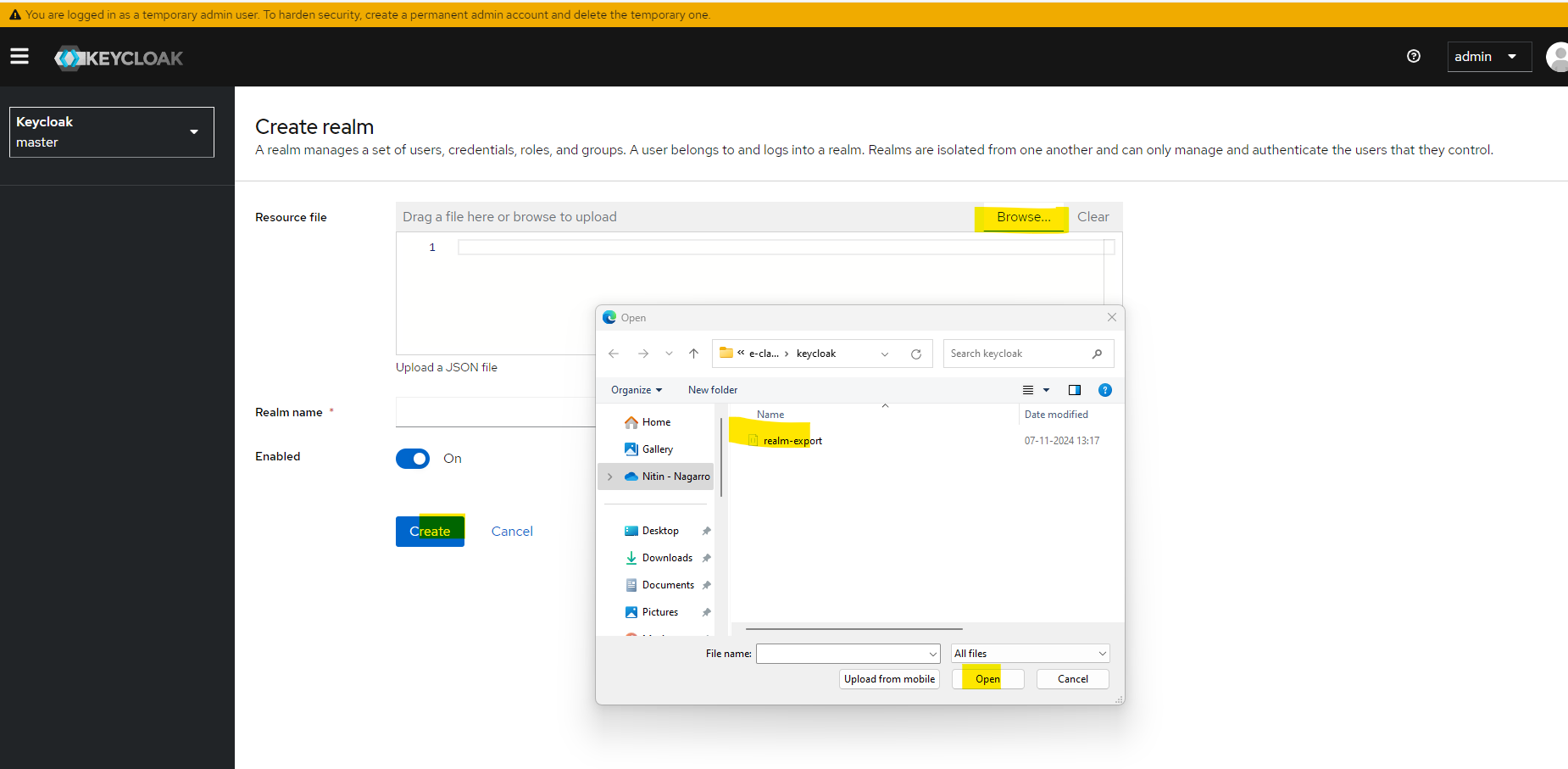
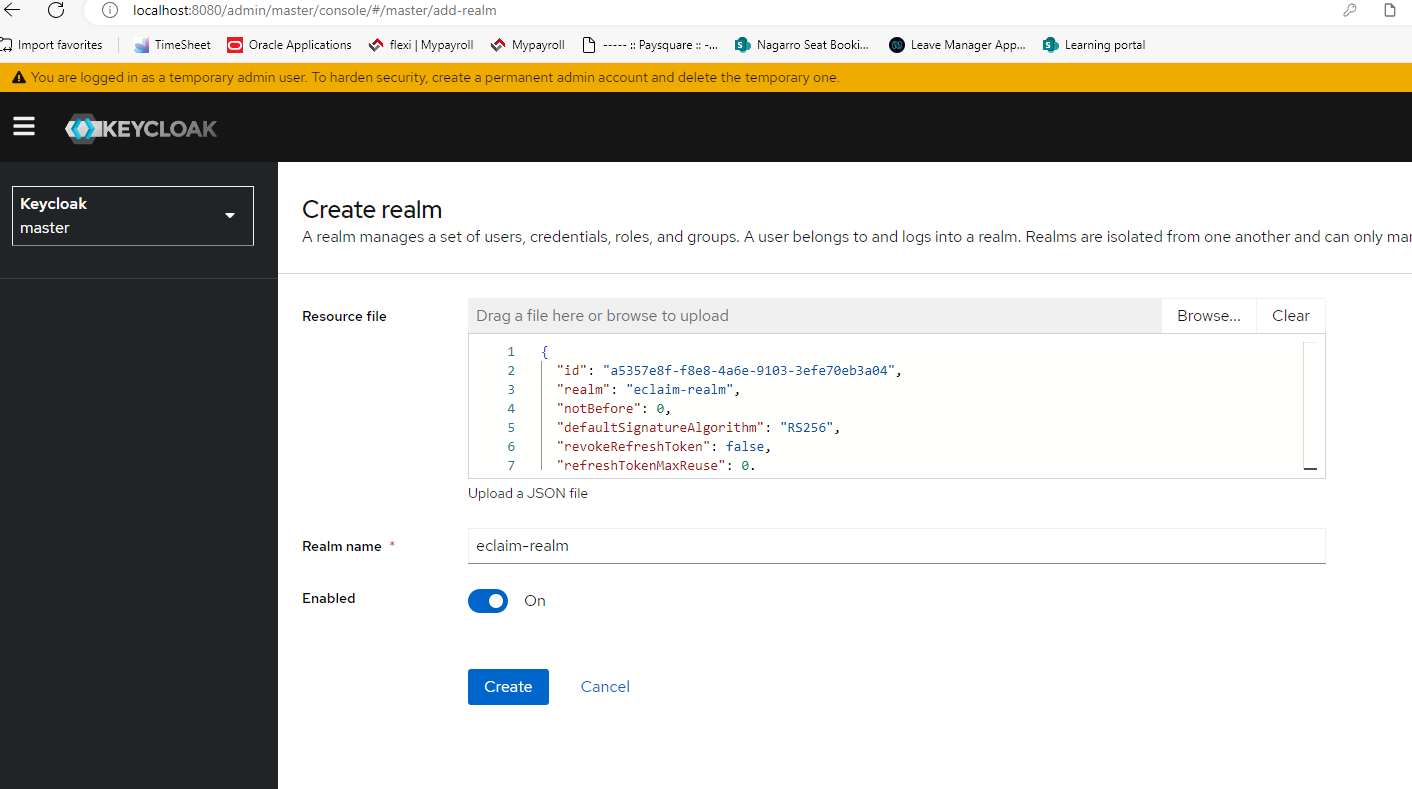
  Description automatically generated
* Wait till images are downloaded and containers are up and running. It will take some time to pull the images from the internet.
* Once containers are up and running, move to the next step.
* A screenshot of a computer

  Description automatically generated

**5. Keycloak Setup**

1. **Access Keycloak**:
   * **URL**: <http://localhost:8080>
   * **Credentials**: admin / admin  
       
     A screenshot of a computer

     Description automatically generated
2. **Import the Realm**:
   * Navigate to **Administration Console** → **Create Realm**.
   * A screenshot of a computer

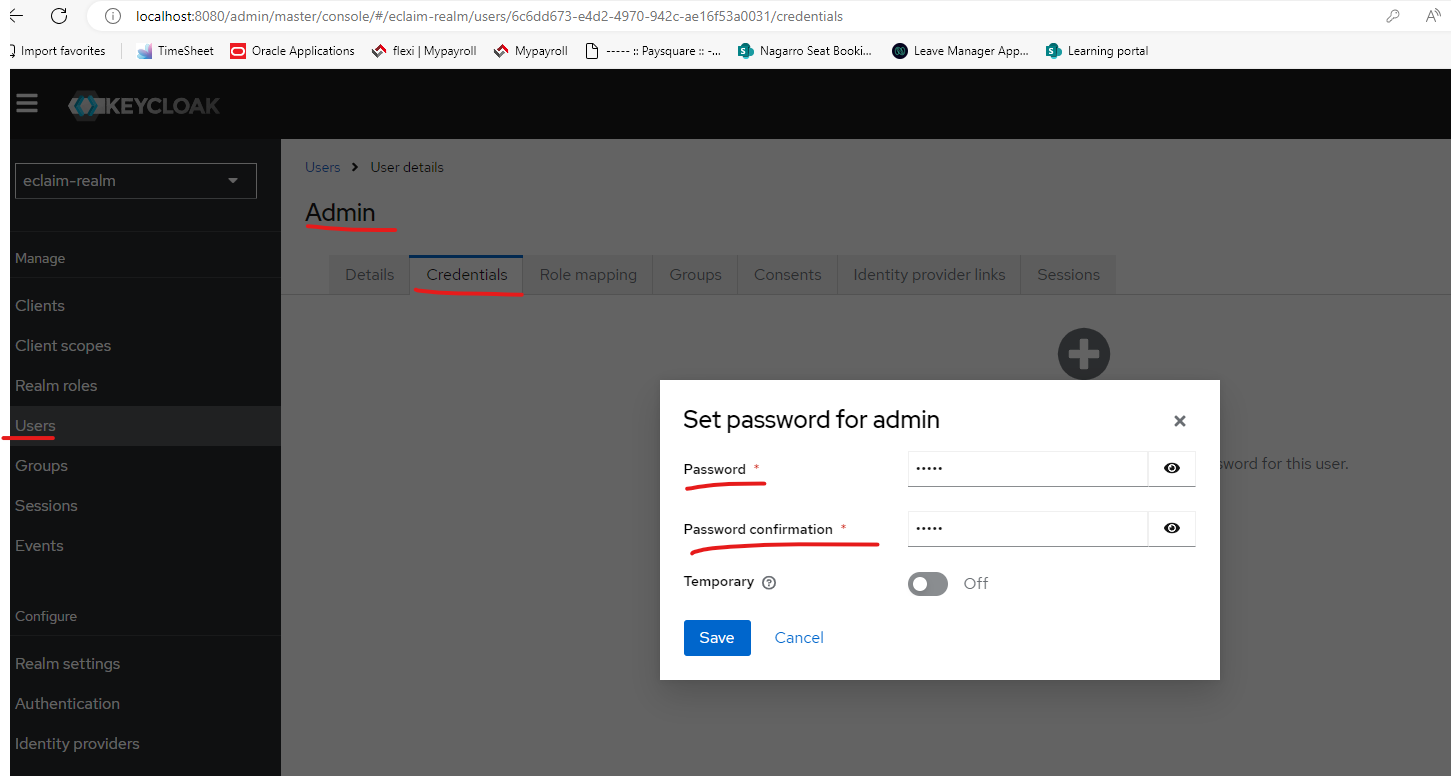
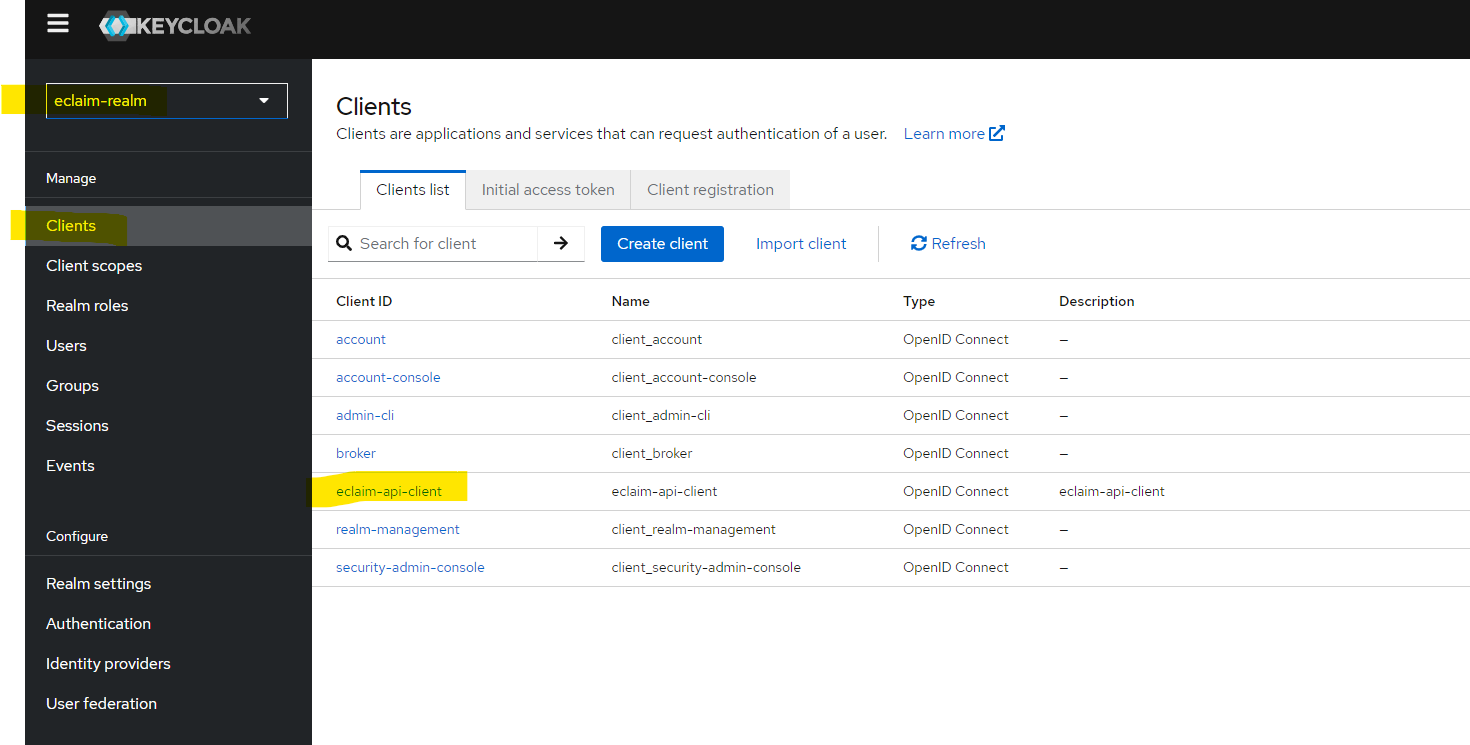
     Description automatically generated
   * Upload the provided **realm-export.json** file in the **e-claim-poc/ keycloak** folder.
   * 
   * 
   * Once you upload the JSON file it will look like following image and click on create button.
   * 
3. **Add Admin User**:
   * Create an admin user with an admin password and assign the **Admin** role.
   * A screenshot of a computer

     Description automatically generated
   * A screenshot of a computer

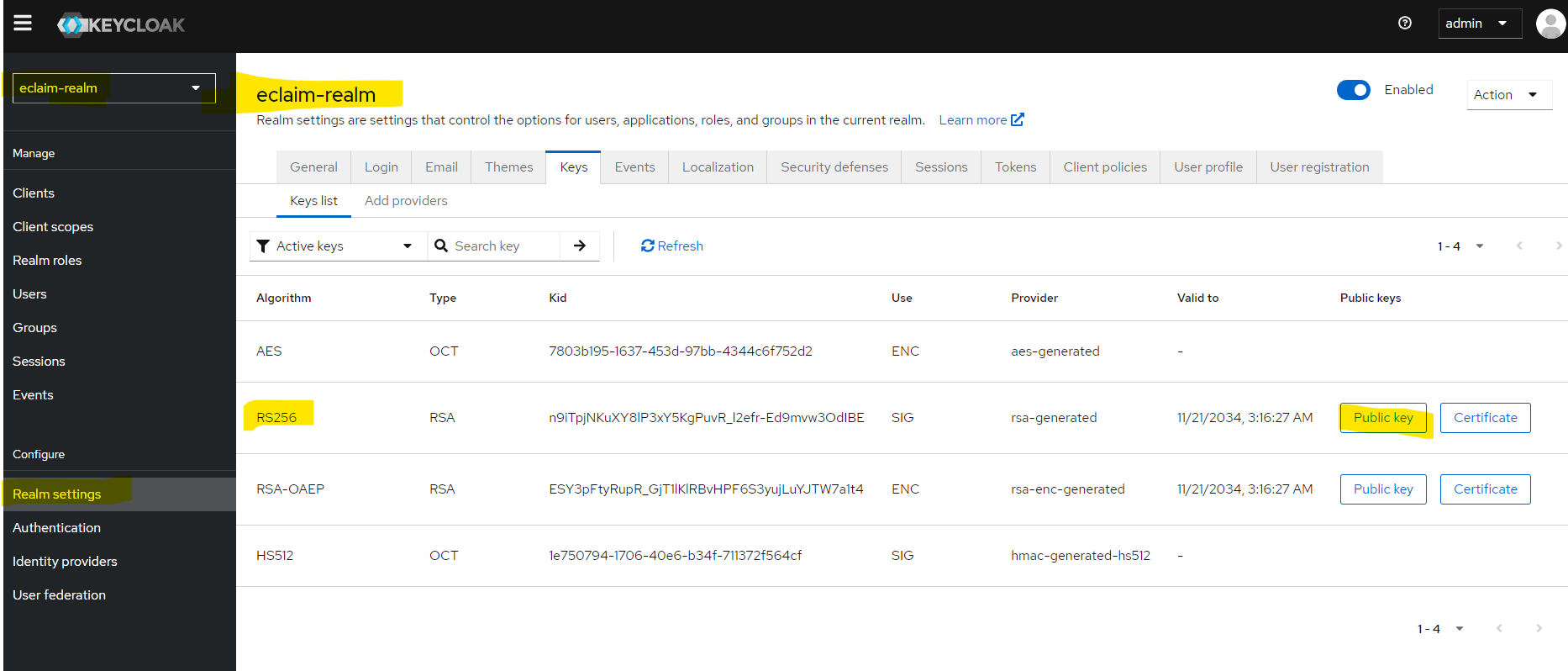
     Description automatically generated
   * Once use created, go to the Role mapping tab
   * A screenshot of a computer

     Description automatically generated
   * Click on Assign role
   * A screenshot of a computer

     Description automatically generated
   * Filter by realm roles, select Admin role and assign
   * A screenshot of a computer

     Description automatically generated
   * Set password for the created user (admin)
   * 
4. **Update Kong Configuration**:
   * Regenerate the client secret in **Keycloak** for **eclaim-api-client**.
   * 
   * Go to the eclaim-api-client and select the Credentials Tab.
   * A screenshot of a computer

     Description automatically generated
   * Click on the Regenerate button and copy and keep the client secrete (we will use it for REST API request)
   * A screenshot of a computer

     Description automatically generated
   * Update the kong.yml file with:
     + Keycloak public key
     + Go to the Realm Setting and copy RS256 public key
     + 
     + A screenshot of a computer

       Description automatically generated
     + Open kong/kong.yml file and paste copied RS256 public key for jwr\_secrets rsa\_public\_key. Make sure you just replace the key and do not change the format.
     + A screen shot of a computer program

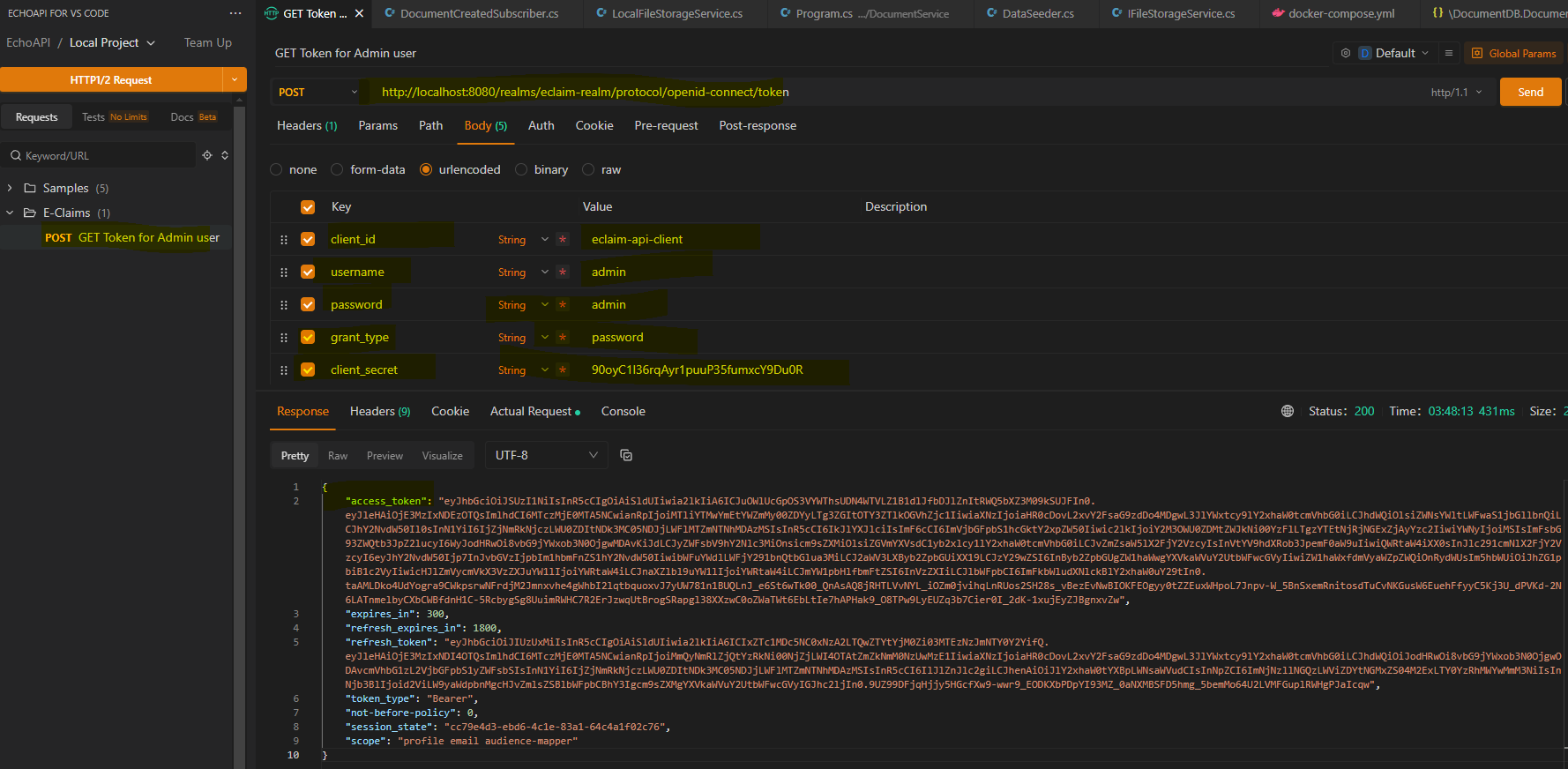
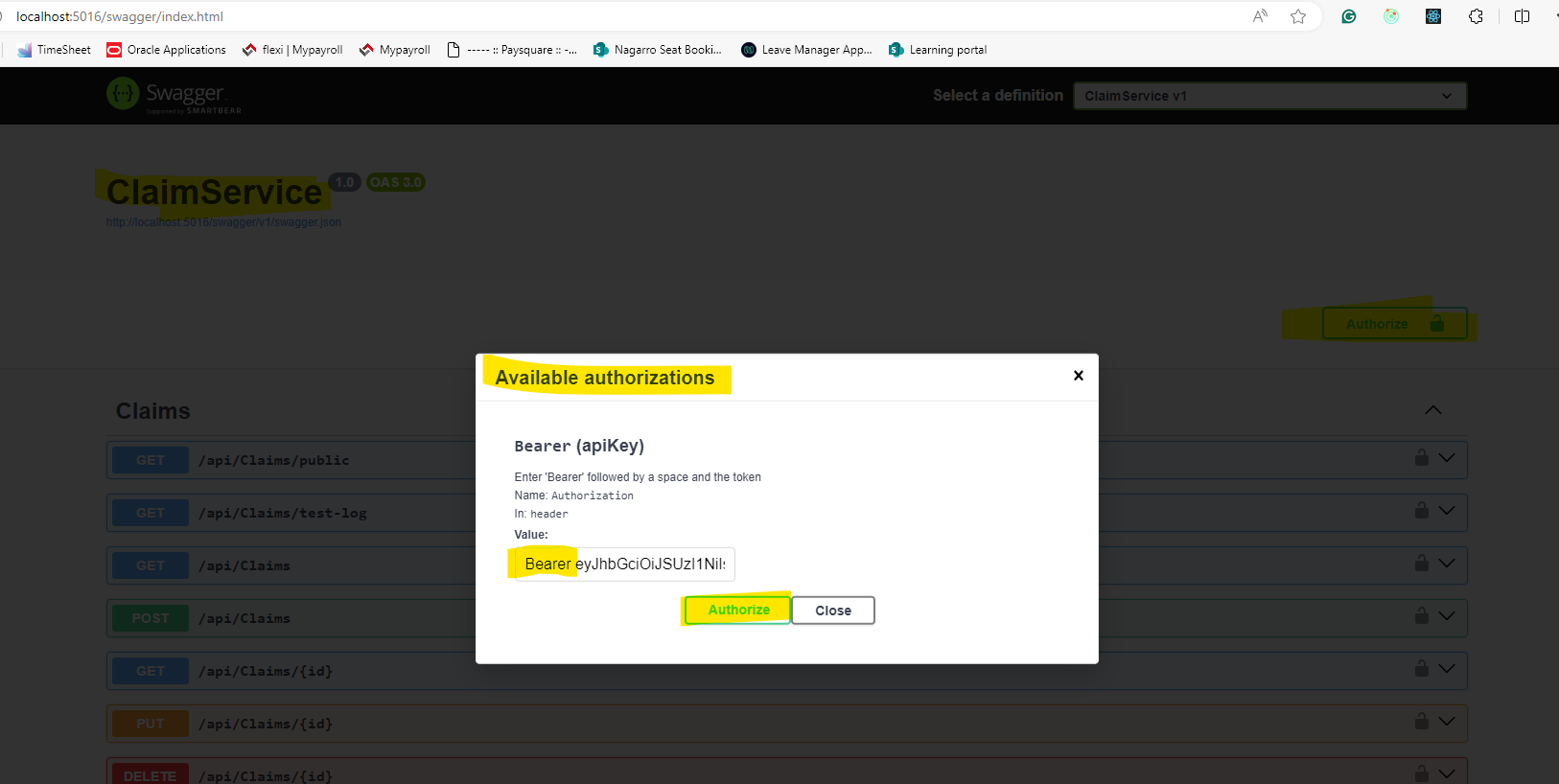
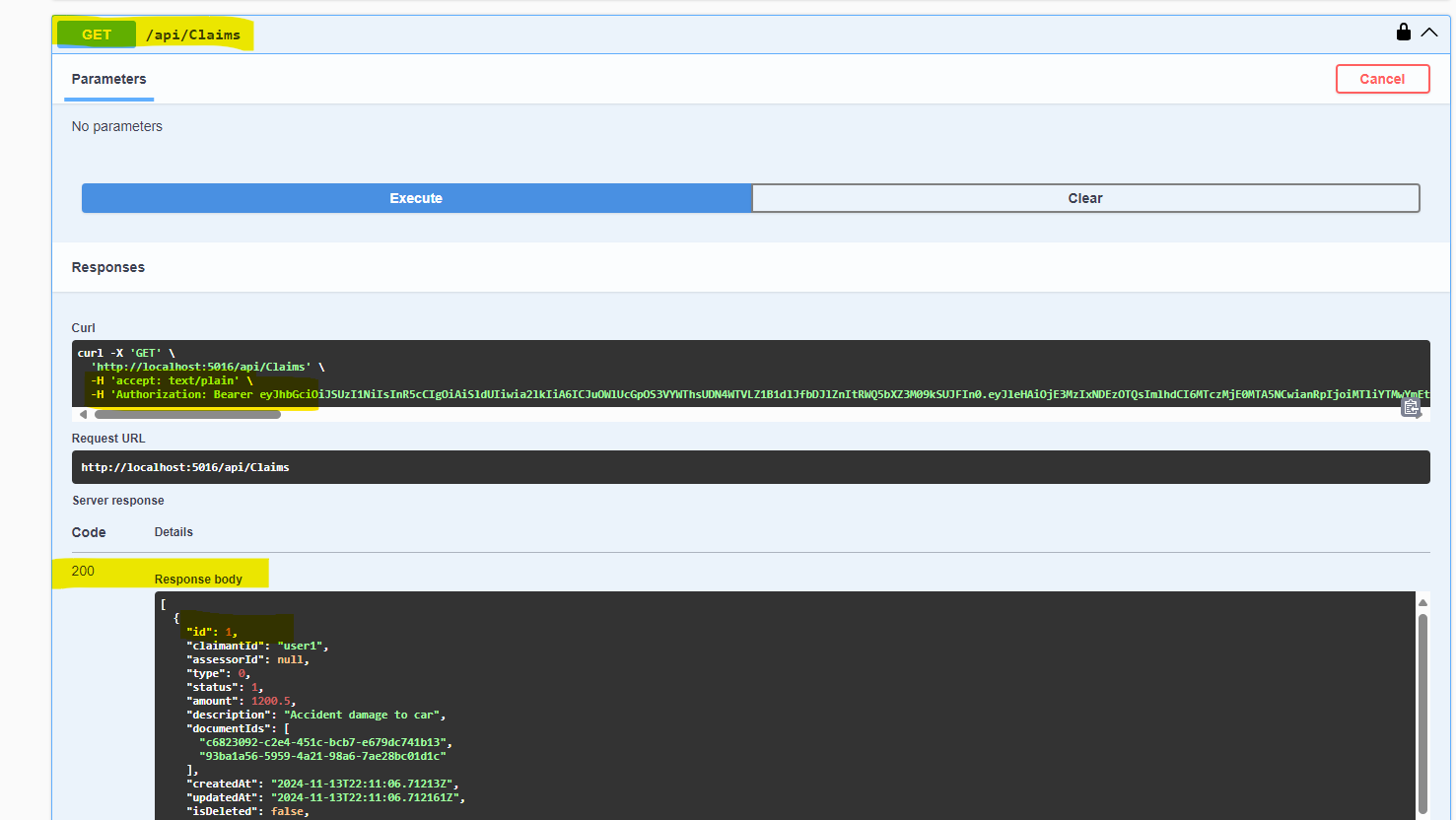
       Description automatically generated
     + Stop docker containers ( to reflect the changes we need to restart the containers)
     + Command – **docker-compose down**
     + And then again start the containers using **docker-compose up –build** cmd
     + A screenshot of a computer program

       Description automatically generated

**6. Swagger API Documentation**

1. **Access Swagger**:
   * **URL**: <http://localhost:5016/swagger>
   * A screenshot of a computer

     Description automatically generated
   * <http://localhost:5020/swagge>
   * A screenshot of a computer

     Description automatically generated
2. **Authenticate**:
   * Obtain a token from Keycloak (details in the next section).
   * 
   * Use **Authorize** in Swagger with **Bearer <token>.  
       
     **
3. **API Endpoints**:
   * Provide a list of API endpoints and payloads. Example:
     + **GET Claims**:
       - URL: http://localhost:8000/api/Claims
       - Headers: Authorization: Bearer <token>
   * 

**7. Token Acquisition**

* **Method**: POST
* **URL**: <http://localhost:8080/realms/eclaim-realm/protocol/openid-connect/token>
* **Parameters**:

grant\_type=password

client\_id=eclaim-api-client

username=admin

password=admin

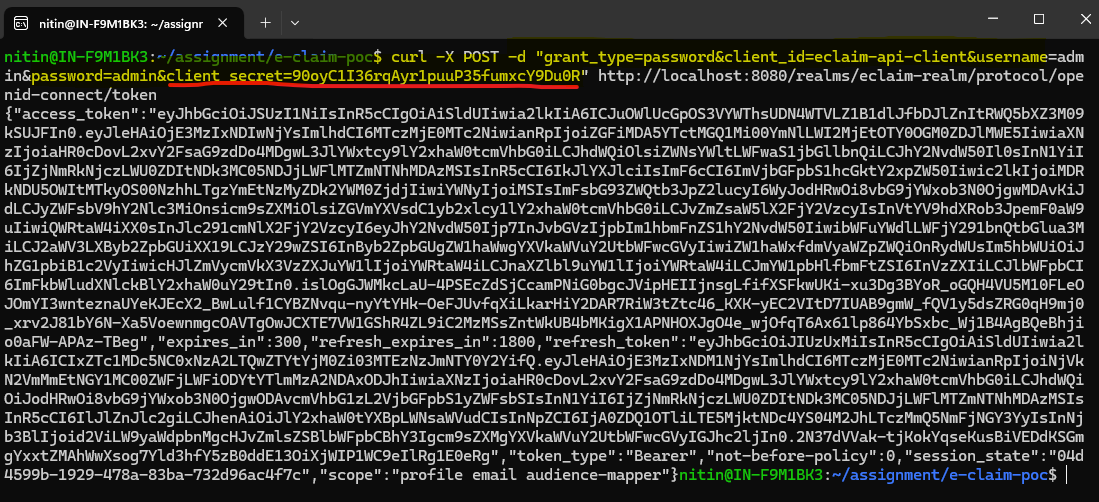
client\_secret=<your-client-secret>

A screenshot of a computer

Description automatically generated

* **Command for Testing**:

curl -X POST -d "grant\_type=password&client\_id=eclaim-api-client&username=admin&password=admin&client\_secret**=<your-secrete-key>**" <http://localhost:8080/realms/eclaim-realm/protocol/openid-connect/token>



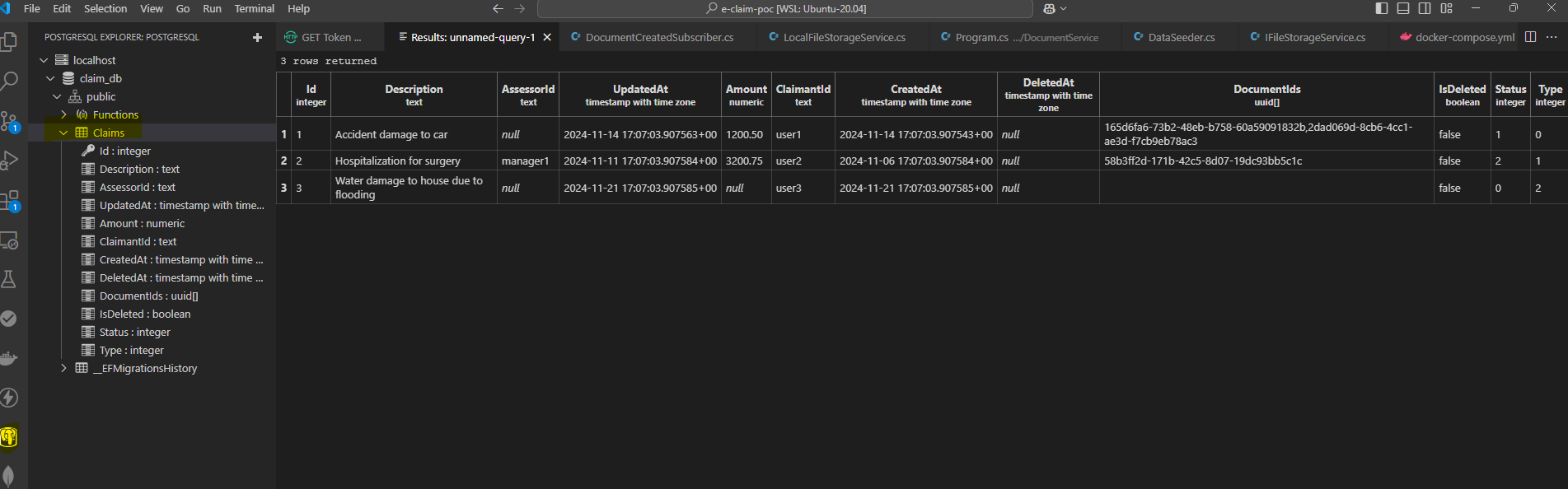
**8. Database Verification**

**PostgreSQL**

1. **Connect**: **docker-compose exec postgres psql -U claim\_user -d claim\_db**
2. **Check Data**: SELECT \* FROM “Claims”;

A black screen with white lines

Description automatically generated

Alternatively, you can connect and verify the claims data in PostgreSQL using the VS Code PostgreSQL extension.  
  


**MongoDB**

**Install the mongo CLI in the Container.**

1. Check if the mongo CLI Tool is Installed in the Container
   1. Run the following command to confirm:
      1. docker exec -it e-claim-poc-mongodb-1 ls /usr/bin/mongo
      2. If the output shows ls: cannot access '/usr/bin/mongo': No such file or directory, it confirms the tool is not installed.
   2. Install the mongo CLI in the Container
      1. docker exec -it e-claim-poc-mongodb-1 bash
      2. apt-get update
   3. Connect to MongoDB Using a Separate MongoDB Client Container
      1. docker run -it --rm --network=host mongo:latest mongosh mongodb://root:password@localhost:27017
2. Query Your MongoDB Data
   1. Show databases:
      1. show dbs
   2. Switch to a specific database:
      1. use DocumentDB
   3. Show collections:
      1. show collections
   4. Query the documents collection:
      1. db.Documents.find().pretty()

A screenshot of a computer

Description automatically generated

Optionally, you can use **MongoDB for VS Code** extension to check the data.

A screenshot of a computer program

Description automatically generated

**9. Elasticsearch Setup**

1. **Run Password Setup**: docker compose exec elasticsearch bin/elasticsearch-setup-passwords interactive
   1. A screenshot of a computer

      Description automatically generated
2. **Verify Elasticsearch**: curl -u elastic:admin@123 <http://localhost:9200>
   1. A computer screen shot of white text

      Description automatically generated
3. Define the logstash\_writer role with appropriate privileges::

curl -X POST -u elastic:your\_elastic\_password http://localhost:9200/\_security/role/logstash\_writer \

-H "Content-Type: application/json" \

-d '{

"cluster": ["monitor", "manage\_index\_templates"],

"indices": [

{

"names": ["logstash-\*", "claims-api-logs-\*"],

"privileges": "create", "write", "create\_index", "manage", "auto\_configure"]

}

]

}'

1. Create the logstash\_internal user and assign the logstash\_writer role:

curl -X POST -u elastic:your\_elastic\_password http://localhost:9200/\_security/user/logstash\_internal \

-H "Content-Type: application/json" \

-d '{

"password": "admin@123",

"roles": ["logstash\_writer"],

"full\_name": "Logstash Internal User",

"email": "logstash\_internal@example.com"

}'

1. Verify the User and Role:
   1. Check the newly created user:
      1. curl -X GET -u elastic:your\_elastic\_password <http://localhost:9200/_security/user/logstash_internal>
   2. Check the newly created role:
      1. curl -X GET -u elastic:your\_elastic\_password <http://localhost:9200/_security/role/logstash_writer>
2. Restart the containers: docker-compose down

Docker-compose up --build

1. **Test the Connection:**
   1. curl -u logstash\_internal:admin@123 http://localhost:9200/

A computer screen shot of a program

Description automatically generated

**Query Logs**:  
  
curl -u elastic:admin@123 -X POST \

http://localhost:9200/claims-api-logs-\*/\_search?pretty \

-H "Content-Type: application/json" \

-d '{

"size": 1,

"sort": [

{

"@timestamp": {

"order": "desc"

}

}

],

"query": {

"match\_all": {}

}

}'

A screenshot of a computer program

Description automatically generated

You can also log in to kibana dashboard and check the index:  
http://localhost:5601/  
user name : elastic

Password:admin@123

A screenshot of a computer

Description automatically generated

**Ignore the following errors in the docker logs**  
  
A screen shot of a computer

Description automatically generated

**10. RabbitMQ Event Processing**

1. **Access RabbitMQ**:
   * **URL**: http://localhost:15672
   * **Credentials**: guest / guest
2. **Publish Test Event**:

curl -u guest:guest -X POST \

http://localhost:15672/api/exchanges/%2F/claims\_exchange/publish \

-H "Content-Type: application/json" \

-d '{

"properties": {},

"routing\_key": "document.processed",

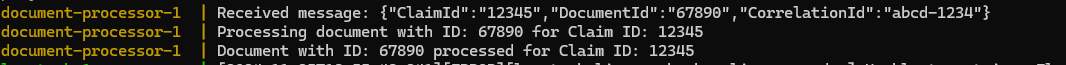
"payload": "{\"ClaimId\":\"12345\",\"DocumentId\":\"67890\",\"CorrelationId\":\"abcd-1234\"}",

"payload\_encoding": "string"

}'

A screenshot of a computer program

Description automatically generated



**11. Troubleshooting**

**Common Issues**

1. **Keycloak Login Failure**:
   * Verify Keycloak is running and the correct credentials are used.
2. **Elasticsearch Query Errors**:
   * Ensure passwords and roles are configured properly.
   * Restart Elasticsearch and Logstash containers if needed.
3. **RabbitMQ Event Not Triggering**:
   * Verify the routing key and check RabbitMQ logs for errors.

**12. Final Checklist**

* Ensure all services are running (docker ps).
* Test all APIs using Swagger or Postman.
* Verify logs in Elasticsearch and RabbitMQ processing.