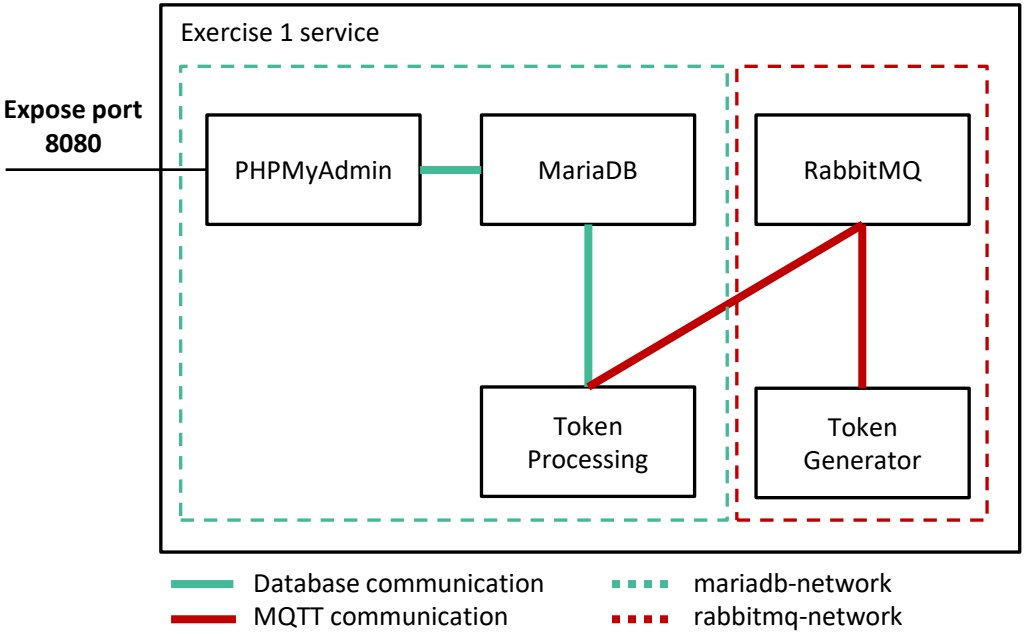
Assignment 1

COMPUTING SYSTEMS AND INFRASTRUCTURES

(SISTEMAS E INFRAESTRUTURAS DE COMPUTAÇÃO)

Assignment 1

- The objective of this exercise is to implement a proof of concept for an application that will generate secret tokens for authentication, similar to the well-known app "chave movil"
- The secret token will be generated by the Token Generator (app-pub.py) and published in a message broker (RabbitMQ). An application, Token Processing (app-sub.py), will consume the secret token and storage in a DB for auditing purposes
- The database can be browsed through a web browser through PHPMyAdmin
- Create a **Docker compose file** that is able to instantiate the required goals by instantiating and interconnecting the required services:
  - Message broker (RabbitMQ)
  - Database (MariaDB)
  - PHPMySQL
  - Token Generator – app-pub.py
  - Token Processing – app-sub.py



Token Generator

- The token generator application has to be implemented in Python and fulfill the following requirements:
  - Generate a hexadecimal token using the library **`secrets`**
  - The length of the tokens is determined by an environmental variable in the container (**`LENGTHTOKEN`**)
  - The number of tokens that should generated is also determined by an environmental variable in the container (**`NUMTOKENS`**)
  - The tokens generated must be published in the channel **/sic/tokens**

Token Processing

- The token processing application has to be implemented in Python and fulfill the following requirements:
  - Connect to the MariaDB and use the DB “**sic**". This DB must be created when the MariaDB container is run
  - Create, if required, the table “**sic**” with the structure:
    - id (int, auto increment)
    - date (timestamp)
    - channel (varchar)
    - value (varchar)
  - Subscribe to the MQTT channel **/sic/#** (subscribe to /sic and all child channels)
  - Insert into the database a new record with the current date and time, the channel name and the token received

About the dockers (1/2)

- Rely only on official and latest images to develop this exercise (**RabbitMQ, MariaDB, Python, PHPMyAdmin**)
- RabbitMQ requires the MQTT plugin to be enabled
- Python requires the Paho MQTT (**PIP install to support the MQTT message broker**)
- Python MySQL Connector Python (**PIP install to support MySQL connections**)
- Create the required Dockerfile to build custom images
- Evaluate the usage of volumes and networks
- Address the dependencies requirements of the containers
- The compose file must be self-contained → **Do not depend on existing custom images, volumes, or networks**

About the dockers (2/2)

- The DB must be persistent by using a volume **mariadb-data-sic**
- The implementation of your solution must consider network isolation, for this consider two different networks for your containers
  - **rabbitmq-network**
  - **mariadb-network**
- A container in your solution must be executed if and only if its dependencies are already satisfied
- Be careful with the syntax in your files and remember to follow the instructions described here
- All the debug information generated by your applications must be published in the channel **/sic/log**

Example of python MQTT connection (publish & subscribe)

```
import paho.mqtt.client as mqtt

def on_disconnect(client, userdata, rc):
    print("Disconnected with result code "+str(rc))

def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))

def on_publish(client, userdata, result):
    print("Message published: "+str(result))


if __name__ == '__main__':
    client= mqtt.Client("sic-pub")
    client.on_publish = on_publish
    client.on_connect = on_connect
    client.on_disconnect = on_disconnect

    client.connect("MQTT_SERVER",1883)

    client.publish("/sic", "MSG")
```

```
import paho.mqtt.client as mqtt

def on_disconnect(client, userdata, rc):
    print("Disconnected with result code "+str(rc))

def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe("/sic/#")

def on_subscribe(client, userdata, mid, granted_qos):
    print("Subscribed")

def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.payload))

if __name__ == '__main__':
    client= mqtt.Client("sic-sub")
    client.on_message = on_message
    client.on_subscribe = on_subscribe
    client.on_connect = on_connect
    client.on_disconnect = on_disconnect

    client.connect("MQTT_SERVER",1883)

    client.loop_forever()
```

Example of python MySQL connection

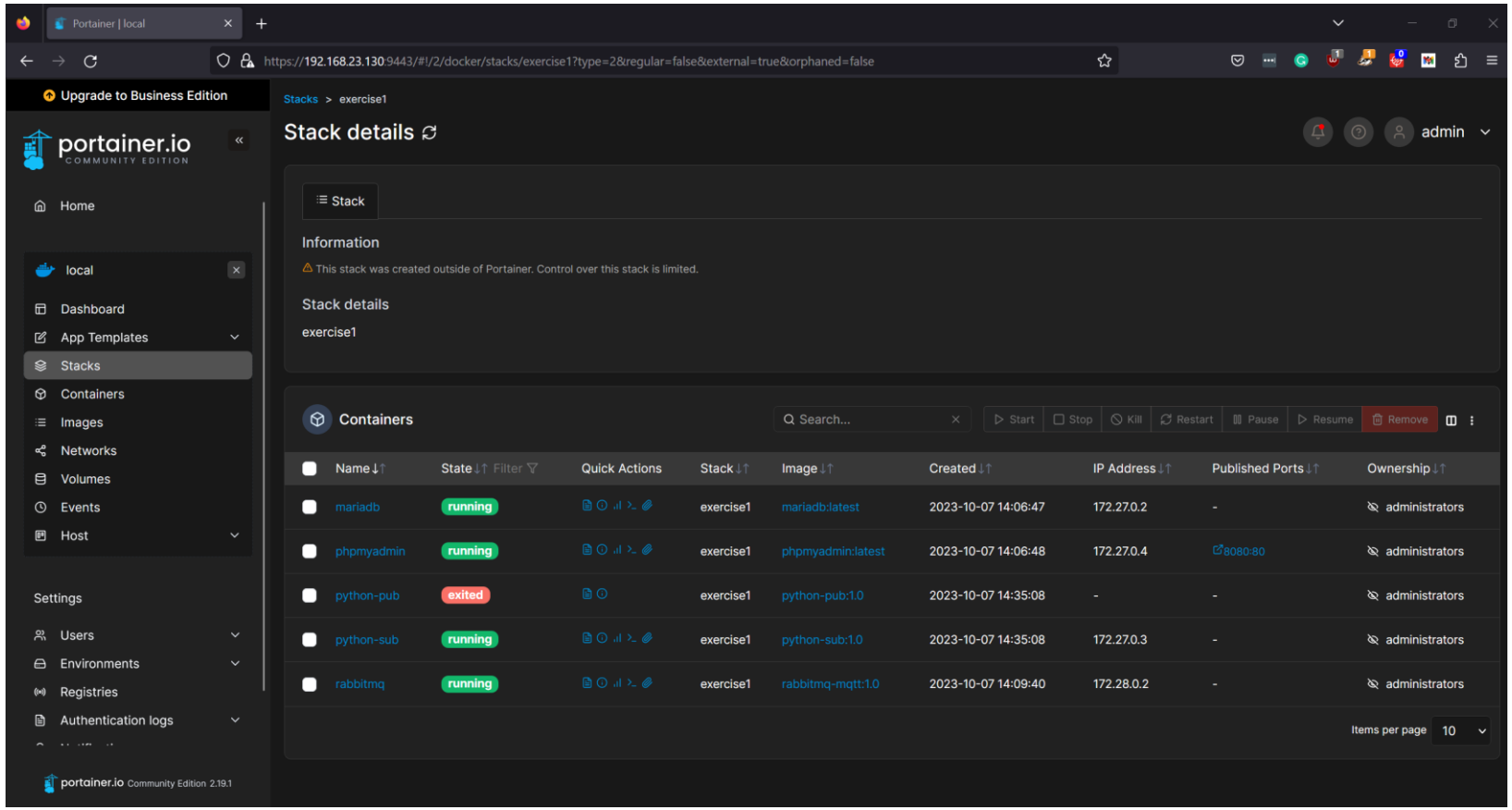
```
import mysql.connector
from mysql.connector import errorcode

if __name__ == '__main__':
    try:
        cnx = mysql.connector.connect(user='root',
password='my-secret-pw', host=MARIADB_SERVER',
database='sic')
        print("Connected to database")

        cursor = cnx.cursor()
    except mysql.connector.Error as err:
        print("Error connecting to database")
        exit(0)
```

```
try:
  TABLE = (
    "CREATE TABLE `sic` ("
    "  `id` int(11) NOT NULL AUTO_INCREMENT,"
    "  `date` TIMESTAMP NOT NULL,"
    "  `client` varchar(64) NOT NULL,"
    "  `value` int(11) NOT NULL,"
    "  PRIMARY KEY (`id`)"
    ") ENGINE=InnoDB")
  cursor.execute(TABLE)
except mysql.connector.Error as err:
  if err.errno == errorcode.ER_TABLE_EXISTS_ERROR:
    print("Table already exists")
  else:
    print("Error creating table")
```

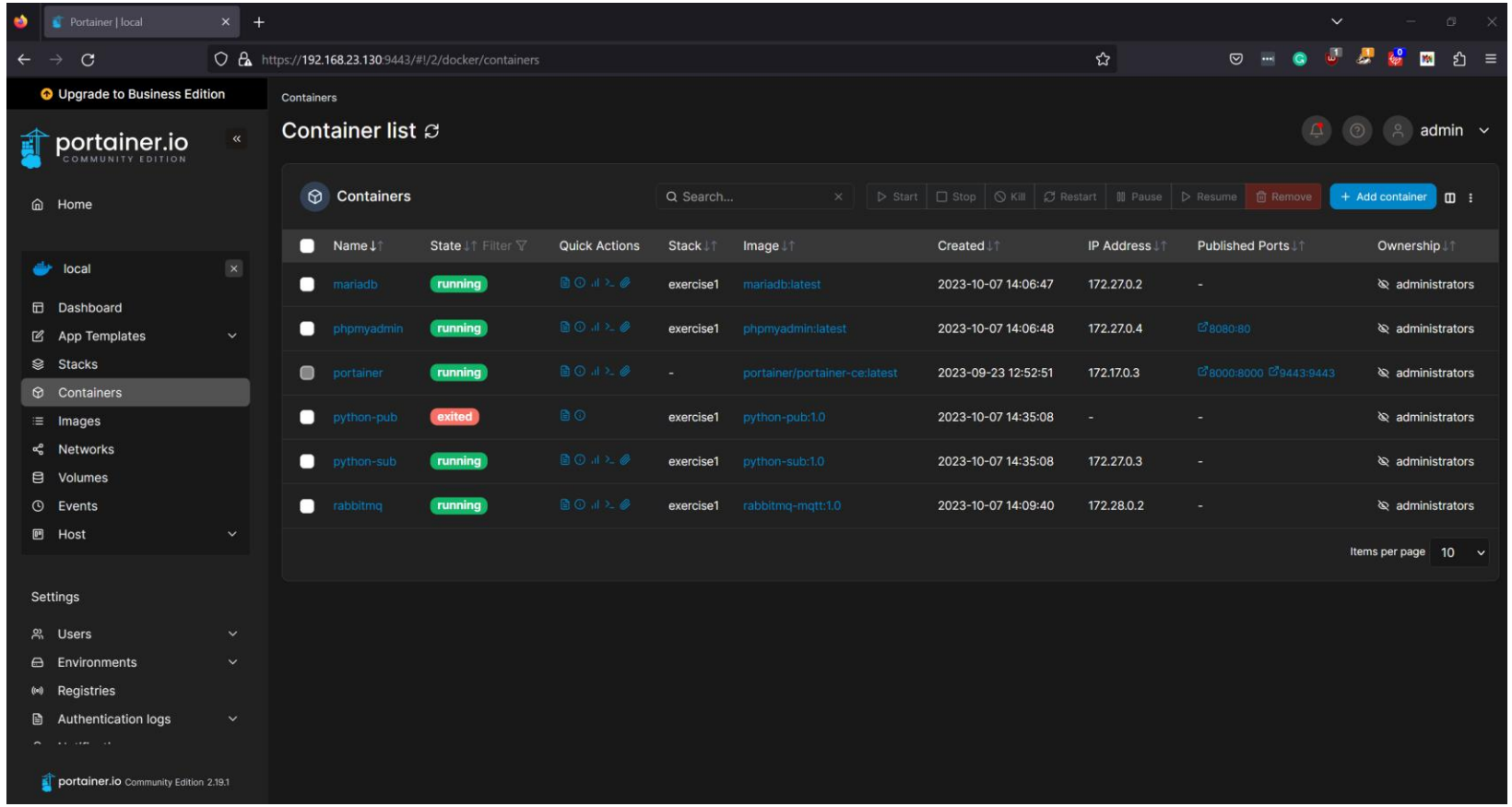
Examples of the expected outcome (1/7)

- View of the directory to be submitted to the InforEstudante (.zip)
  - exercise1
    - compose.yaml
    - Dockerfile
    - pub
      - app-pub.py
      - Dockerfile
    - sub
      - app-sub.py
      - Dockerfile

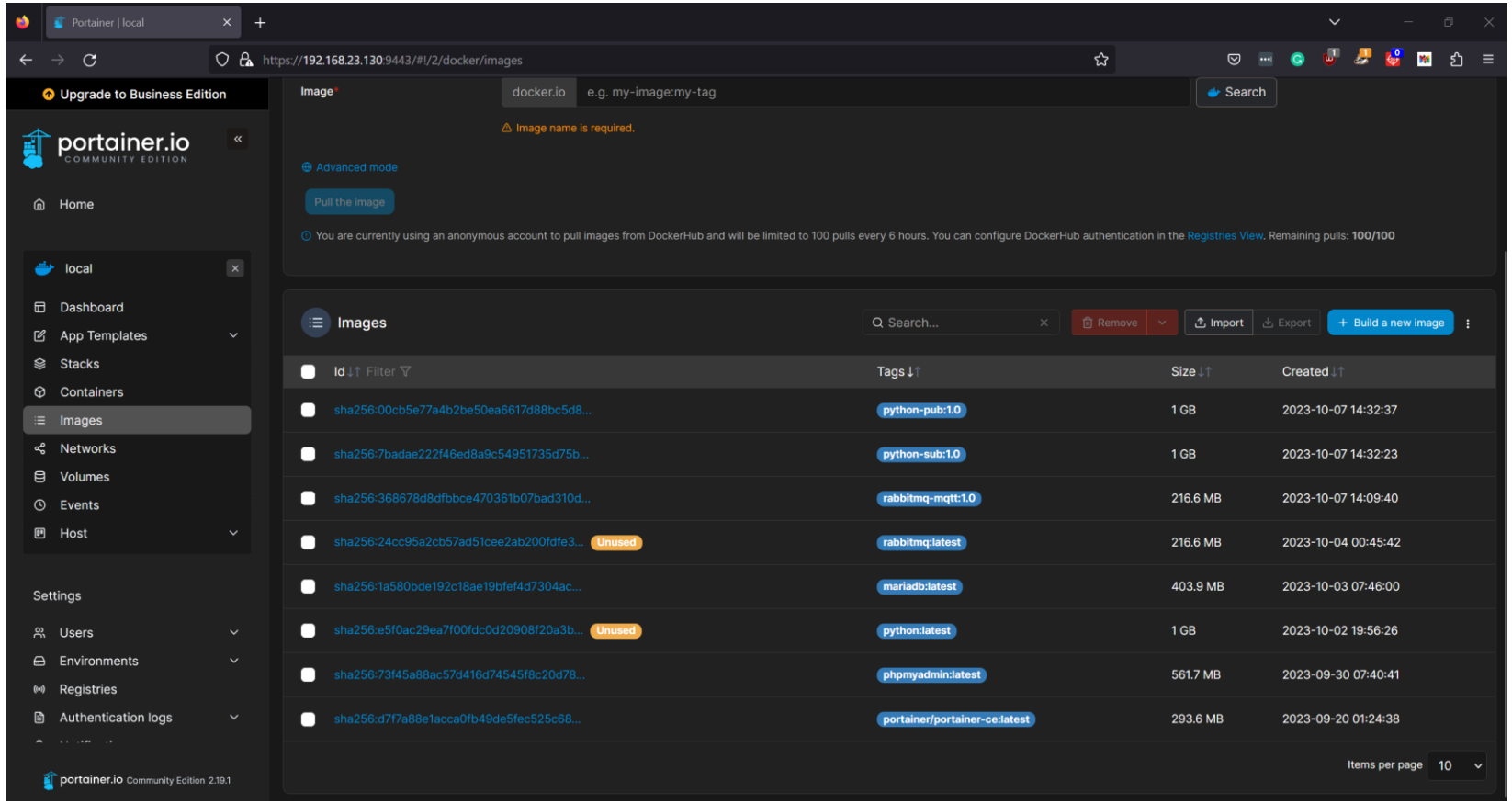
Examples of the expected outcome (2/7)



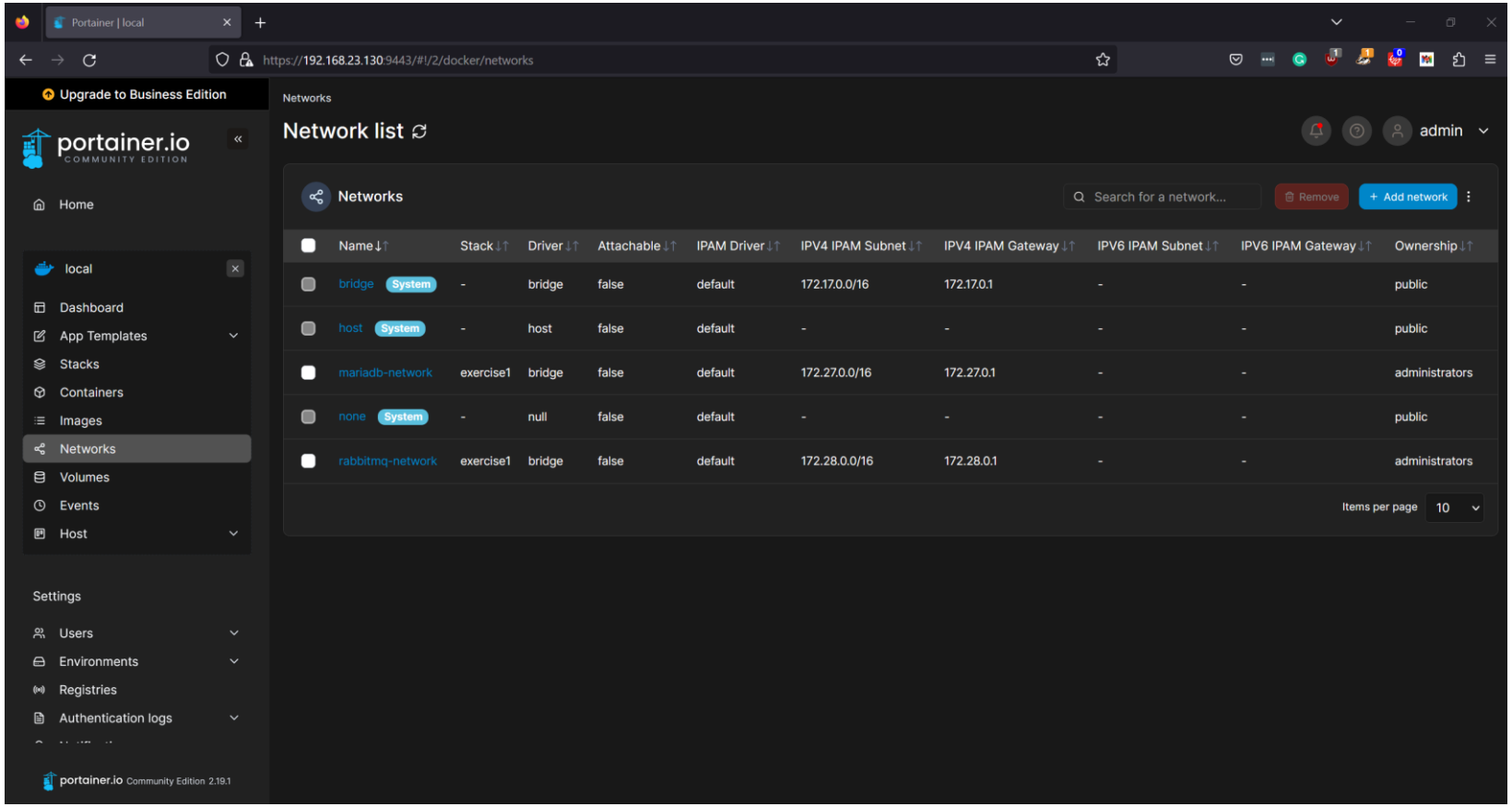
Examples of the expected outcome (3/7)



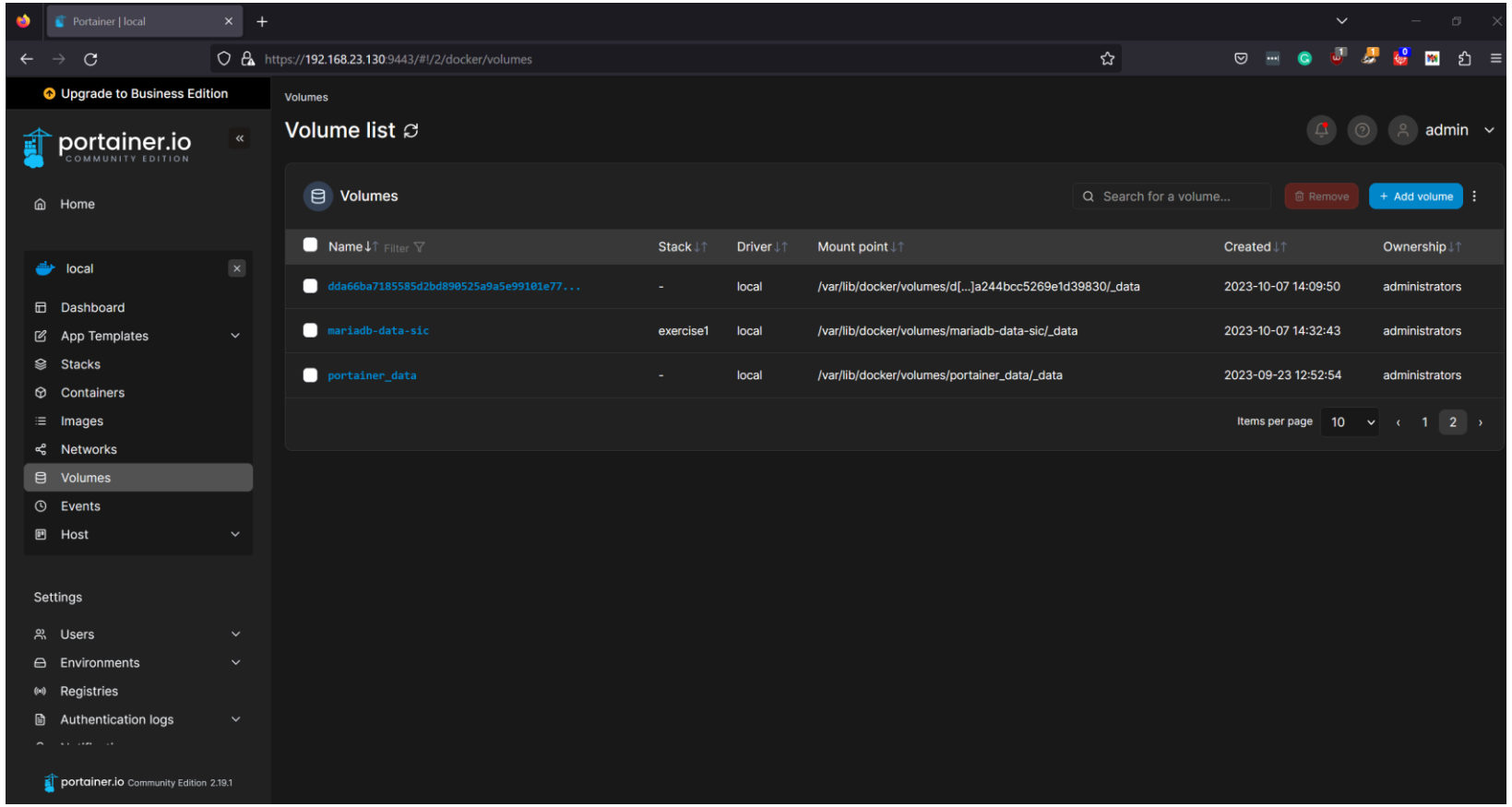
Examples of the expected outcome (4/7)



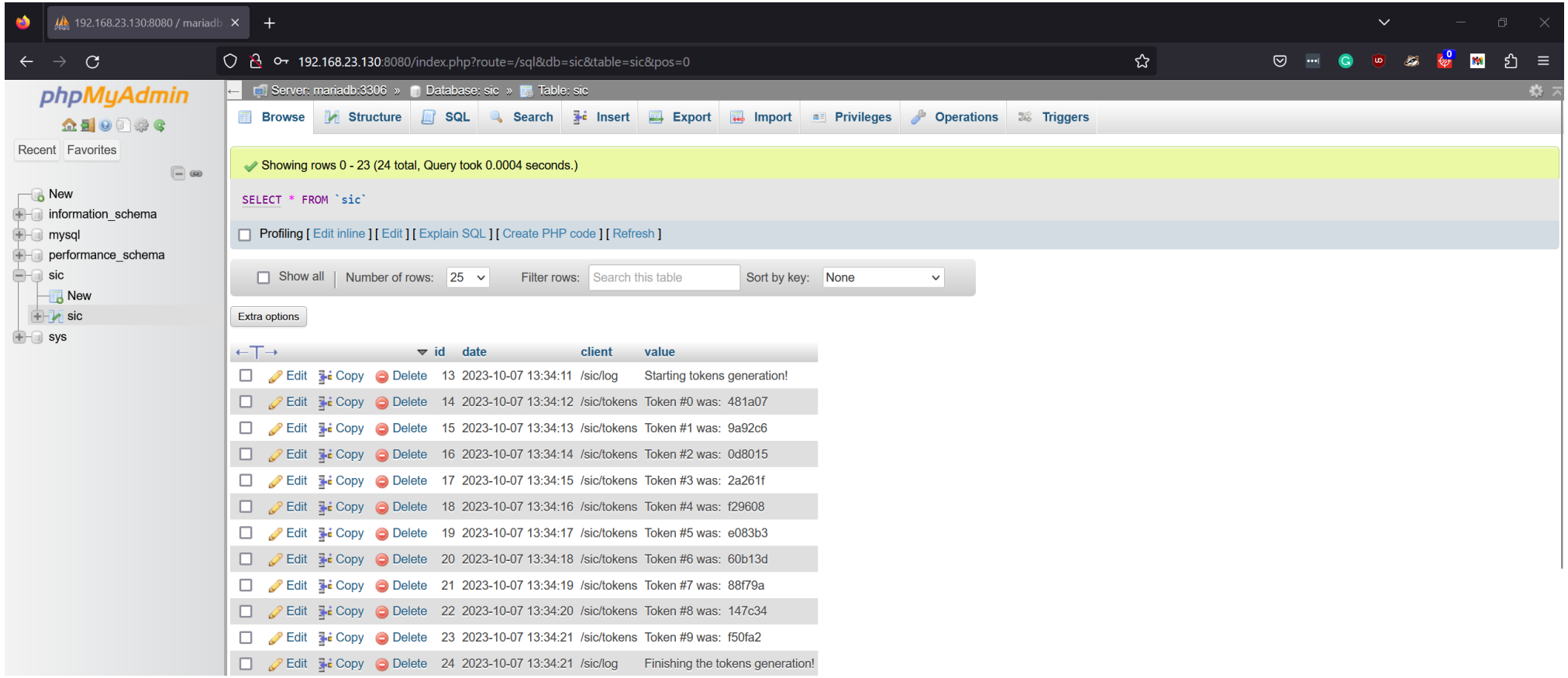
Examples of the expected outcome (5/7)



Examples of the expected outcome (6/7)



Examples of the expected outcome (7/7)



Final remarks

- The assignment must be developed in groups of **2** students
- The assignment must be submitted in InforEstudante in a **.zip** file containing all required files
- Students must enroll in one of the available defense time slots available in InforEstudante

- Submission deadline: **28/10/2023**
- Defence: **30/10/2023**