Alfaisal University - College of Engineering

Software Engineering Department

Subject: SE 443 – Cloud Computing for Software Engineers
Final Project (Fall 2022)

Instructor	Dr. George Violettas		
Date	Dec 23 2022 (2 days before the final exam)	**Time**	11:59 PM
Room	Online		
Grade Percentage	15%		

Student Name: **Rayan Taha Almudawah**

Student Number: **200203**

Student Signature: _____

Information and Instructions

- *This is an open book, open notes project. The University's code of ethics applies.*
- *Allocate your time wisely.*
- *Answer each of the exam questions to the best of your knowledge.*
- *Clearly state any assumptions made that might be needed to understand your solution.*
- *Show your work – any sign of serious effort will be considered.*

Question	Mark	Full
1		**5**
1.a		**5**
1.b		**5**
2.a		**5**
2.b		**5**
2.b.i		**5**
2.b.ii		**5**
Total		**35**

Course Learning Outcomes (CLO)		Questions
S01	An ability to identify, formulate, and solve complex engineering problems by applying principles of engineering, science, and mathematics.	1-2
S02	An ability to apply engineering design to produce solutions that meet specified needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors	1-2
S03	An ability to communicate effectively with a range of audiences	N/A
S04	An ability to recognize ethical and professional responsibilities in engineering situations and make informed judgments, which must consider the impact of engineering solutions in global, economic, environmental, and societal contexts	N/A
S05	An ability to function effectively on a team whose members together provide leadership, create a collaborative and inclusive environment, establish goals, plan tasks, and meet objectives	N/A
S06	An ability to develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgment to draw conclusions.	2
S07	An ability to acquire and apply new knowledge as needed, using appropriate learning strategies	N/A

Project Delivery

For all the following, you have to upload ONE SINGLE FILE (pdf) with all the links for the code created (GitHub), plus as many screenshots as you consider necessary. Please, keep it minimum). You have to deliver the file TWO DAYS before the ORIGINAL final exam date (December 22).

Please write as may scripts as necessary in Python 3 to do the following. You are allowed to use any 3rd libraries and/or frameworks. The easiest Python library to use for such purpose is here.

1. The first part is an mqtt broker running locally into your computer (localhost) by using the image eclipse-mosquitto. Then do the following:
    a. Initialize a single-node Docker Swarm. Try to name the swarm. Is it possible?
    b. Print out the ID, name, and creation date of the Swarm.
    c. Creates a network named "se443_test_net", using the "overlay" driver, "global" scope, and with an IP CIDR range of 10.10.10.0/24.
    d. Print out the ID, name, and creation date of the network.
    e. Deploys a service named "broker" with three (3) replicas of the docker image.
        i. You will also have to configure the service to automatically restart (*always*).

2. The second part has a subscriber and a publisher again running from your local computer with the docker image efrecon/mqtt-client. By using as many as necessary Python scripts you have to publish/subscribe to the above mqtt broker. The topic has to be "alfaisal_uni" and the messages have to be <ID-Name-Surname-No-XX> where XX will be an increasing counter. Send and receive as many messages as you need to exhibit the above implementation.
    a. Deploy a service named "subscriber" with three (3) replicas of the efrecon/mqtt-client docker image
    b. Deploy a service named "publisher" with three (3) replicas of the efrecon/mqtt-client docker image
    c. You will also have to configure the services to automatically restart (*always*)
    d. Your code should print out the names, IDs, and number of running replicas of both server and client services.
    e. Start publishing messages to the broker, so your code can show how multiple subscriber replicas subscribe to the same topic of multiple broker replicas and multiple publisher replicas send multiple messages. Do craft your messages as you wish (e.g., 1.1.1) to exhibit the above random behaviours.

3. Make all your code run for a set amount of time (e.g., 5 minutes), sends a number of messages, and then shut down cleanly, cleaning up after itself to bring down services, remove the overlay network, and finally tear down the Docker Swarm(s).

Part1-Code:

Broker_Part1.py

```
import docker
import time
import json
client = docker.from_env()

#Reinitialize the swarm To make sure the swarm is empty and new
client.swarm.leave(force = True)
client.swarm.init()

#Print the swarm ID, name, and creation date
print("Swarm's ID: ", client.swarm.attrs['ID'])
print("Swarm's Name: ", client.swarm.attrs['Spec']['Name'])
print("Swarm's Creation Date: ", client.swarm.attrs['CreatedAt'])

print("------------------------------------------------------------------------------
-------------------------------------")
#Creating the network with the name se443_test_net and the subnet and driver and scope
client.networks.create("se443_test_net", driver = "overlay", scope ="global", ipam =
docker.types.IPAMConfig(pool_configs = [docker.types.IPAMPool(subnet =
"10.10.10.0/24")]))
for net in client.networks.list():
    if net.name == "se443_test_net":
        print("Network's ID: ", net.id)
        print("Network's Name: ", net.name)
        print("Network's Creation Date: ", net.attrs['Created'])

print("------------------------------------------------------------------------------
-------------------------------------")
#Creating the broker service with scale of 3 (replicas) with the name broker and the
restart policy of 'any' (which is always)
client.services.create("eclipse-mosquitto",name = "broker", restart_policy =
docker.types.RestartPolicy(condition = "any")).scale(3)


print("Leave it running for 10 minutes...")
time.sleep(600)
print("------------------------------------------------------------------------------
-------------------------------------")


#Cleanup and termination
print("\nTermination Broker service...")
client.services.get("broker").remove()
print("Termination is complete")
```

Part1-Output:

```
PS D:\4th-Year\SE443-Cloud Computing\Docker_Project> & C:/Users/rayan/AppData/Local/Programs/Python/Python311/python.exe
Swarm's ID:  vicxxjclwu8gixd3o2bbfzske
Swarm's Name:  default
Swarm's Creation Date:  2022-12-20T19:57:45.187138352Z
------------------------------------------------------------------------------------------------------------------------
Network's ID:  8v0ulb32dl50n5nso7uxomzb4
Network's Name:  se443_test_net
Network's Creation Date:  2022-12-20T19:57:45.715605156Z
------------------------------------------------------------------------------------------------------------------------
Leave it running for 10 minutes...
------------------------------------------------------------------------------------------------------------------------

Termination Broker service...
Termination is complete
PS D:\4th-Year\SE443-Cloud Computing\Docker_Project>
```

Part2-Code:

```
import docker
import time
import json
client = docker.from_env()

#Reinitialize the swarm To make sure the swarm is empty and new
client.swarm.leave(force=True)
client.swarm.init()
#Creating the network with the name se443_test_net and the subnet and driver and scope
client.networks.create("se443_test_net", driver="overlay", scope="global",
ipam=docker.types.IPAMConfig(pool_configs=[docker.types.IPAMPool(subnet='10.10.10.0/24'
)]))

#Printing the network details required
for net in client.networks.list():
    if net.name == "se443_test_net":
        print("Network ID: ", net.id)
        print("Network Name: ", net.name)
        print("Network Creation Date: ", net.attrs['Created'])



print("------------------------------------------------------------------------------------
-------------------------------------")

#Creating subscriber service with scale of 3 (replicas) with the name Subscriber and
the restart policy of 'any' (which is always) with the image efrecon/mqtt-
client
client.services.create("efrecon/mqtt-client",
name="Subscriber",  restart_policy=docker.types.RestartPolicy(condition="any"),
networks=["se443_test_net"],
                       command='sub -h host.docker.internal:1888 -t alfaisal_uni -
v').scale(3)
```

```
print("Susbcriber's ID:" , client.services.list()[0].id)
print("Susbcriber's Name:" , client.services.list()[0].name)
print("Susbcriber's Creation Date:" , client.services.list()[0].attrs['CreatedAt'])
print("Susbcriber's Number Of Replicas:" ,
client.services.list()[0].attrs['Spec']['Mode']['Replicated']['Replicas'])

print("------------------------------------------------------------------------------
-------------------------------------")

#Creating publisher service with scale of 3 (replicas) with the name Publisher and the
restart policy of 'any' (which is always) with the image efrecon/mqtt-client
client.services.create("efrecon/mqtt-client",
name="Publisher",  restart_policy=docker.types.RestartPolicy(condition="any"),
networks=["se443_test_net"],
                       command='pub -h 172.17.0.1 -t alfaisal_uni -m "<200203 - Rayan -
Almudawah - 0564086839>"').scale(3)
print("Publisher's  ID:" , client.services.list()[0].id)
print("Publisher's  Name:" , client.services.list()[0].name)
print("Publisher's  Creation Date:" , client.services.list()[0].attrs['CreatedAt'])
print("Publisher's  Number Of Replicas:" ,
client.services.list()[0].attrs['Spec']['Mode']['Replicated']['Replicas'])

print("------------------------------------------------------------------------------
-------------------------------------")
print("Leave it running for 10 minutes...")
time.sleep(600)
print("------------------------------------------------------------------------------
-------------------------------------\n")

#Cleanup and termination

print("Terminating Publisher, Subscriber, and Network services and finally leaving the
swarm")

client.services.get("Publisher").remove()
print("Publisher Terminated....")

client.services.get("Subscriber").remove()
print("Subscriber Terminated....")

client.networks.get("se443_test_net").remove()
print("Network Terminated....")

client.swarm.leave(force=True)
print("Swarm Left Forcefully....")
```

Part2-Output:

```
PS D:\4th-Year\SE443-Cloud Computing\Docker_Project> & C:/Users/rayan/AppData/Local/Programs/Python/Python311/python.exe "
Network ID:  hk3m78tenanyngl6uukehh98i
Network Name:  se443_test_net
Network Creation Date:  2022-12-20T19:59:30.999968658Z
---------------------------------------------------------------------------------------------------------------------------
Susbcriber's ID: xcg798pszjyne2qe19nj1ch3x
Susbcriber's Name: Subscriber
Susbcriber's Creation Date: 2022-12-20T19:59:31.096258015Z
Susbcriber's Number Of Replicas: 3
---------------------------------------------------------------------------------------------------------------------------
Publisher's  ID: axrb9ozc1nrpnzpl025ath35f
Publisher's  Name: Publisher
Publisher's  Creation Date: 2022-12-20T19:59:31.249018076Z
Publisher's  Number Of Replicas: 3
---------------------------------------------------------------------------------------------------------------------------
Leave it running for 10 minutes...
---------------------------------------------------------------------------------------------------------------------------

Terminating Publisher, Subscriber, and Network services and finally leaving the swarm
Publisher Terminated....
Subscriber Terminated....
Network Terminated....
Swarm Left Forcefully....
PS D:\4th-Year\SE443-Cloud Computing\Docker_Project>
```