#Part 1: initializes a single-node Docker Swarm, creates a network, and deploys a service

```
import docker

# Connect to the Docker daemon
client = docker.from_env()

# Initialize a single-node Docker Swarm
client.swarm.init(advertise_addr='eth0')

# Create a network
network = client.networks.create(
    "se443_test_net",
    driver="overlay",
    scope="global",
    ipam={"driver": "default", "config": [{"subnet": "10.10.10.0/24"}]},
)
# Print out the ID and creation date of the network
print(f"ID: {network.id}, Creation date: {network.attrs['CreatedAt']}")
```

```
# Deploy a service
service = client.services.create(
    "broker",
    image="eclipse-mosquitto",
    mode=docker.types.ServiceMode("replicated", replicas=3),
    restart_policy=docker.types.RestartPolicy("always"),
    networks=[network],
)
# Print out the ID and creation date of the service
print(f"ID: {service.id}, Creation date: {service.attrs['CreatedAt']}")
```

#Part 2: deploys services for a publisher and a subscriber, and publishes/subscribes to the MQTT broker

```
import docker
import time

# Connect to the Docker daemon
client = docker.from_env()

# Deploy a service for the subscriber
subscriber_service = client.services.create(
    "subscriber",
    image="efrecon/mqtt-client",
    mode=docker.types.ServiceMode("replicated", replicas=3),
    restart_policy=docker.types.RestartPolicy("always"),
)
# Deploy a service for the publisher
publisher_service = client.services.create(
    "publisher",
    image="efrecon/mqtt-client",
    mode=docker.types.ServiceMode("replicated", replicas=3),
```

```
    restart_policy=docker.types.RestartPolicy("always"),
)
# Print out the names, IDs, and number of running replicas of the services
print(f"Subscriber service: {subscriber_service.name}, ID: {subscriber_service.id}, Running replicas: {subscriber_service.attrs['Spec']['Mode']['Replicated']['Replicas']}")
print(f"Publisher service: {publisher_service.name}, ID: {publisher_service.id}, Running replicas: {publisher_service.attrs['Spec']['Mode']['Replicated']['Replicas']}")
# Start publishing messages to the broker
counter = 1
while True:
    # Publish a message to the "Alfaisal_uni" topic
    client.containers.run(
        "efrecon/mqtt-client",
        "pub -h host.docker.internal:1883 -t Alfaisal_uni -m '<-Abdulrahman-Hassan-No-{}>'".format(counter),
        network_mode="host",
    )
    # Increment the counter
    counter += 1

    # Sleep for 1 second before publishing the next message
```

time.sleep(1)

#Part 3: runs the publisher and subscriber services for a set amount of time, sends a number of messages, and then cleans up after itself

```
import docker

import time


# Connect to the Docker daemon

client = docker.from_env()


# Set the running time (in seconds)

running_time = 300


# Get the start time

start_time = time.time()


# Run the publisher and subscriber services for the set amount of time

while time.time() - start_time < running_time:

    time.sleep(1)


# Stop the publisher and subscriber services

client.services.get("publisher").update(mode={"Replicated": {"Replicas": 0}})
```

```
client.services.get("subscriber").update(mode={"Replicated": {"Replicas": 0}})


# Wait for the services to stop

time.sleep(5)


# Remove the services

client.services.get("publisher").remove()

client.services.get("subscriber").remove()


# Disconnect from the network

client.networks.get("se443_test_net").disconnect(client.services.get("broker"))


# Remove the network

client.networks.get("se443_test_net").remove()


# Leave the Swarm

client.swarm.leave(force=True)
```