Лабораторна робота №2

Золотов Іван ФБ-31мп

Посилання на GitHub

https://github.com/SideOctopus/Lab2

1. Встановлення Hazelcast:

sudo docker run -d --name hazelcast1 -p 5704:5701 hazelcast/hazelcast:latest

sudo docker run -d --name hazelcast2 -p 5705:5701 hazelcast/hazelcast:latest

sudo docker run -d --name hazelcast3 -p 5706:5701 hazelcast/hazelcast:lates

2. Розгортання Management Center:

sudo docker run -d -p 8081:8080 hazelcast/management-center

3. Підключаємо кластер до Management Center:

4. Distributed Map

```python
import hazelcast

def main():
    # Create a Hazelcast client and connect to the cluster
    client = hazelcast.HazelcastClient(cluster_members=["183.158.69.103:4704", "183.158.69.103:4705", "183.158.69.103:4706"])

    # Get a distributed map named "capitals"
    distributed_map = client.get_map("capitals").blocking()

    # Insert 1000 key-value pairs into the map
    for i in range(1000):
        distributed_map.put(i, f"City {i}")

    # Print the size of the map
    print(f"Map size: {distributed_map.size()}")

    # Shutdown the Hazelcast client
    client.shutdown()

if __name__ == "__main__":
    main()
```

Перевіряємо розподіл даних між вузлами в Management Center:

Member	Entries	Gets	Puts	Removals	Sets	Entry Memory	Events
172.17.0.2:5700	351	0	351	0	0	45.25 kB	0
172.17.0.3:5700	333	0	333	0	0	42.85 kB	0
172.17.0.4:5700	316	0	316	0	0	40.70 kB	0
TOTAL	1,000	0	1,000	0	0	128.80 kB	0

Вимикаємо одну ноду

sudo docker stop hazelcast1

Member	Entries	Gets	Puts	Removals	Sets
172.17.0.3:5700	510	0	333	0	0
172.17.0.4:5700	490	0	316	0	0
TOTAL	1,000	0	649	0	0

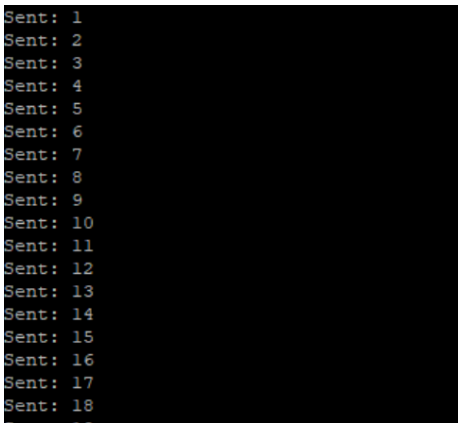
Вимикаємо ще одну ноду

sudo docker stop hazelcast2

4. Topic

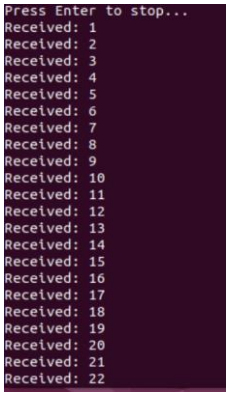
```python
import hazelcast
import time

def task4_1():
    # Create a Hazelcast client and connect to the cluster
    client = hazelcast.HazelcastClient(cluster_members=["183.158.69.103:4704", "183.158.69.103:4705", "183.158.69.103:4706"])

    try:
        # Get a distributed topic named "my-distributed-topic"
        topic = client.get_topic("my-distributed-topic")

        # Publish numbers 1 to 100 to the topic
        for i in range(1, 101):
            topic.publish(i)
            print(f"Sent: {i}")
            time.sleep(0.1)  # Sleep to simulate delay between publishes

    finally:
        # Shutdown the Hazelcast client
        client.shutdown()

if __name__ == "__main__":
    task4_1()
```

Отримуємо:



Клієнт повідомлень без паузи:

```python
import hazelcast

def task4_2():
    # Create a Hazelcast client and connect to the cluster
    client = hazelcast.HazelcastClient(cluster_members=["183.158.69.103:4704", "183.158.69.103:4705", "183.158.69.103:4706"])

    try:
        # Get the distributed topic named "my-distributed-topic"
        topic = client.get_topic("my-distributed-topic")

        # Define a listener function to handle incoming messages
        listener = topic.add_listener(lambda message: print(f"Received: {message.message}"))

        # Wait for user input to stop the listener (typically press Enter)
        input("Press Enter to stop...\n")

    finally:
        # Shutdown the Hazelcast client
        client.shutdown()

if __name__ == "__main__":
    task4_2()
```



Клієнт повідомлень з паузою:

```python
import hazelcast
import time

def task4_2():
    # Create a Hazelcast client and connect to the cluster
    client = hazelcast.HazelcastClient(cluster_members=["183.158.69.103:4704", "183.158.69.103:4705", "183.158.69.103:4706"])

    try:
        # Get the distributed topic named "my-distributed-topic"
        topic = client.get_topic("my-distributed-topic")

        # Define a listener function to handle incoming messages
        listener = topic.add_listener(lambda message: print(f"Received: {message.message}"))

        # Pause execution for 10 seconds to listen for messages
        time.sleep(10)

        # Wait for user input to stop the listener (typically press Enter)
        input("Press Enter to stop...\n")

    finally:
        # Shutdown the Hazelcast client
        client.shutdown()

if __name__ == "__main__":
    task4_2()
```

5. Bounded Queue

```python
import hazelcast
import time

def produce():
    # Create a Hazelcast client and connect to the cluster
    client = hazelcast.HazelcastClient(cluster_members=["183.158.69.103:4704", "183.158.69.103:4705", "183.158.69.103:4706"])

    try:
        # Get the distributed queue named "bounded-queue"
        queue = client.get_queue("bounded-queue").blocking()

        # Produce items (numbers 1 to 100) and enqueue them
        for i in range(1, 101):
            queue.put(i)
            print(f"Produced: {i}")
            time.sleep(0.1)  # Introducing a small delay for demonstration

    finally:
        # Shutdown the Hazelcast client to release resources
        client.shutdown()

if __name__ == "__main__":
    produce()
```

```python
import hazelcast
import time

def consume():
    # Create a Hazelcast client and connect to the cluster
    client = hazelcast.HazelcastClient(cluster_members=["183.158.69.103:4704", "183.158.69.103:4705", "183.158.69.103:4706"])

    try:
        # Get the distributed queue named "bounded-queue"
        queue = client.get_queue("bounded-queue").blocking()

        # Continuously consume items from the queue
        while True:
            item = queue.take()
            print(f"Consumed: {item}")

    finally:
        # Shutdown the Hazelcast client to release resources
        client.shutdown()

if __name__ == "__main__":
    consume()
```

На виході отримуємо 2 термінали з парними і непарними новерами.