PRESENTACIÓN TAREA 1 SISTEMAS DISTRIBUIDOS

YERKO CUZMAR

CRISTIAN VALLEJOS

OBJETIVOS

1. Crear nodos de manera virtual utilizando Docker.
2. Conexión entre nodos a través de sus sockets utilizando Python.
3. Envío y registro de mensajes entre nodos.

¿CÓMO?

Actividad 1:

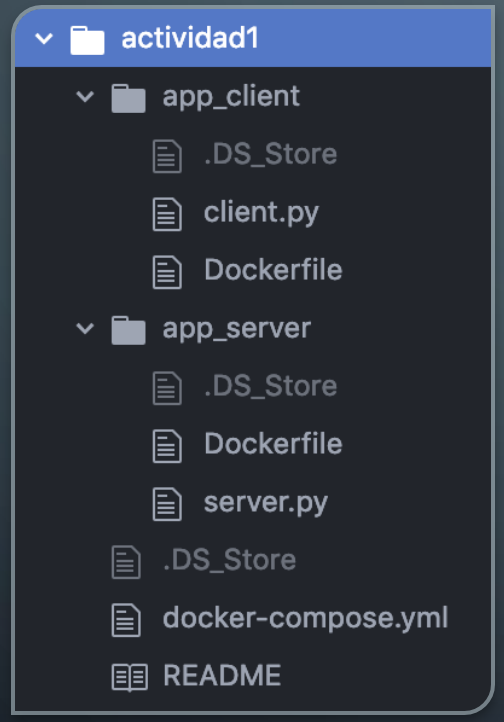
Un cliente y un servidor se comunican vía mensajes, registrando cada uno la respuesta del otro.

Actividad 2:

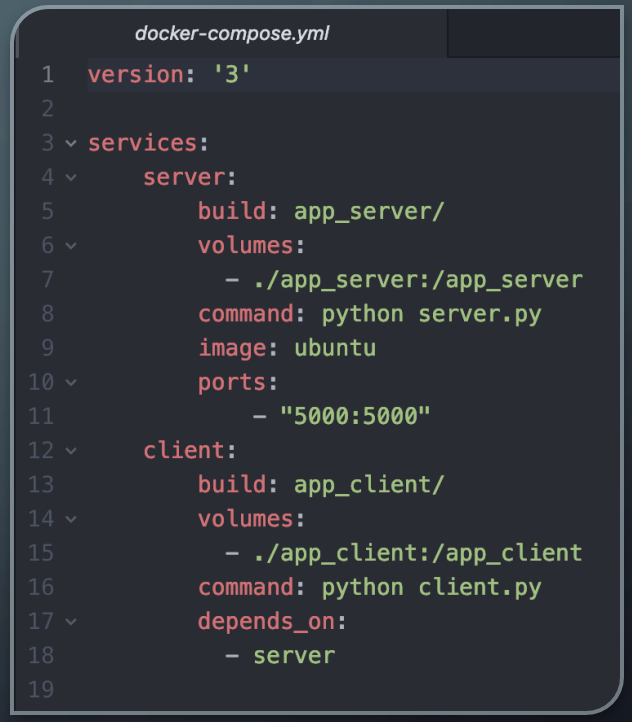
Un cliente envía mensajes a un servidor y estos se registran en uno de los 3 nodos de almacenamiento interno que están conectados directamente al servidor.

ACTIVIDAD 1

Árbol de archivos



docker-compose.yml



ACTIVIDAD 1

Dockerfile / Servidor

Dockerfile

```
1  FROM ubuntu:latest
2
3  RUN apt-get update \
4      && apt-get -y install iputils-ping \
5      && apt-get -y install iproute2 \
6      && apt-get -y install dnsutils \
7      && apt-get -y install python3 \
8      && apt-get -y install python3-pip \
9      && apt-get -y install vim \
10     && apt-get -y install sqlite
11
12 ADD server.py /app_server/
13
14 WORKDIR /app_server/
15
```

Dockerfile / Cliente

Dockerfile

```
1  FROM ubuntu:latest
2
3  RUN apt-get update \
4      && apt-get -y install iputils-ping \
5      && apt-get -y install iproute2 \
6      && apt-get -y install dnsutils \
7      && apt-get -y install python3 \
8      && apt-get -y install python3-pip \
9      && apt-get -y install vim \
10     && apt-get -y install sqlite
11
12 ADD client.py /app_client/
13
14 WORKDIR /app_client/
15
```

ACTIVIDAD 1

Servidor:

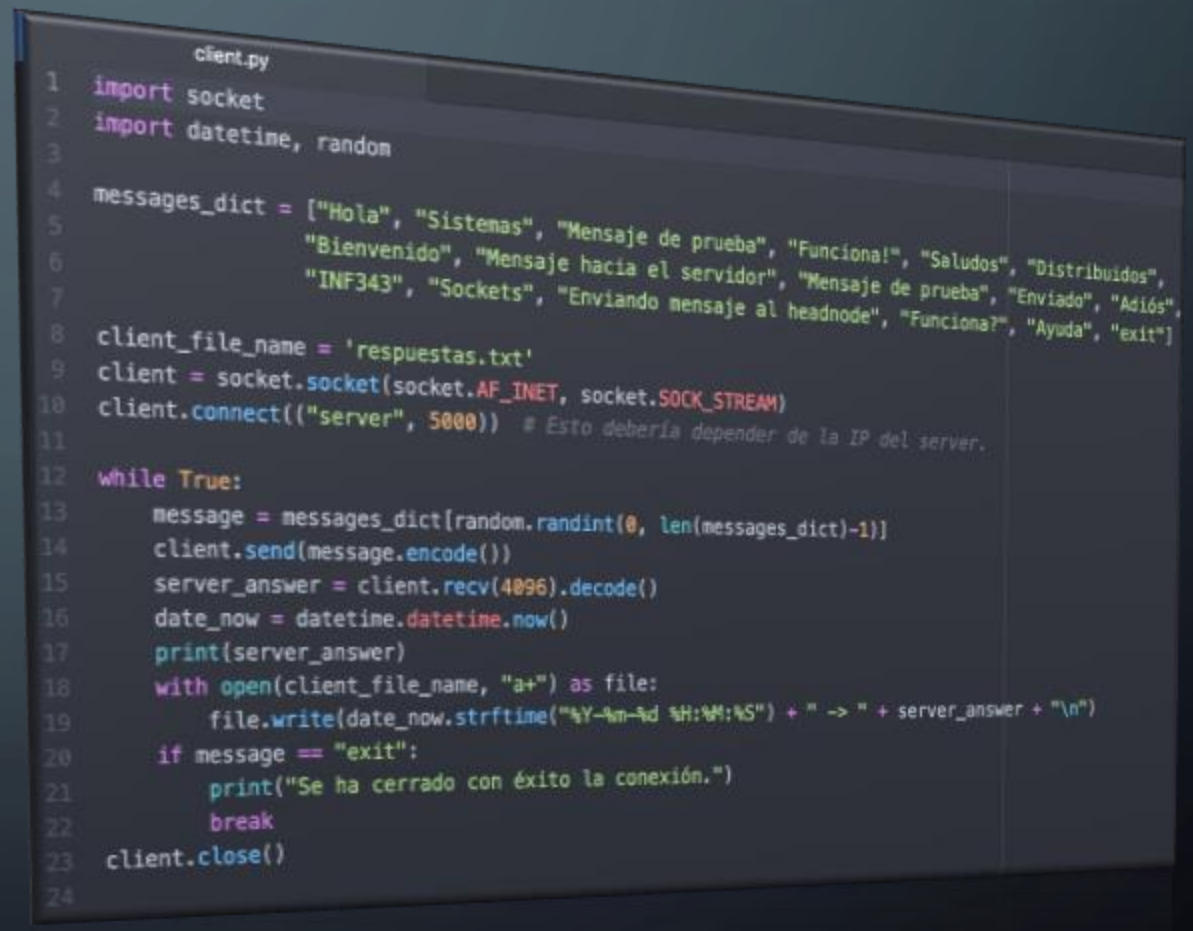
1. Se mantiene activo esperando conexión de cliente en el puerto 5000.

2. Al recibir un mensaje del cliente lo registra en un archivo de texto y responde al cliente que el mensaje fue recibido.

ACTIVIDAD 1

Cliente:

1. Conexión a un servidor en puerto 5000.
2. Envía mensajes predeterminados de manera aleatoria al servidor.
3. Registra respuestas del servidor en archivo de texto.
4. Cliente termina la conexión al enviar “exit” y termina su ejecución.



client.py

```python
import socket
import datetime, random

messages_dict = ["Hola", "Sistemas", "Mensaje de prueba", "Funciona!", "Saludos", "Distribuidos",
                 "Bienvenido", "Mensaje hacia el servidor", "Mensaje de prueba", "Enviado", "Adiós",
                 "INF343", "Sockets", "Enviando mensaje al headnode", "Funciona?", "Ayuda", "exit"]

client_file_name = 'respuestas.txt'
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(("server", 5000))  # Esto debería depender de la IP del server.

while True:
    message = messages_dict[random.randint(0, len(messages_dict)-1)]
    client.send(message.encode())
    server_answer = client.recv(4096).decode()
    date_now = datetime.datetime.now()
    print(server_answer)
    with open(client_file_name, "a+") as file:
        file.write(date_now.strftime("%Y-%m-%d %H:%M:%S") + " -> " + server_answer + "\n")
    if message == "exit":
        print("Se ha cerrado con éxito la conexión.")
        break
client.close()
```

ACTIVIDAD 1

log.txt

```
1  172.19.0.3:46274 -> Hola
2  172.19.0.3:46274 -> Adiós
3  172.19.0.3:46274 -> Saludos
4  172.19.0.3:46274 -> Mensaje de prueba
5  172.19.0.3:46274 -> INF343
6  172.19.0.3:46274 -> Saludos
7  172.19.0.3:46274 -> Sistemas
8  172.19.0.3:46274 -> exit
```

Servidor

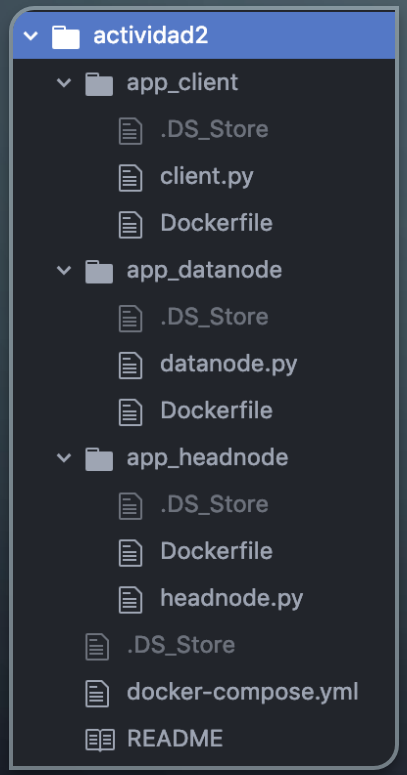
Cliente

respuestas.txt

```
1  2019-10-05 01:57:38 -> El servidor ha recibido satisfactoriamente su mensaje 'Hola'.
2  2019-10-05 01:57:38 -> El servidor ha recibido satisfactoriamente su mensaje 'Adiós'.
3  2019-10-05 01:57:38 -> El servidor ha recibido satisfactoriamente su mensaje 'Saludos'.
4  2019-10-05 01:57:38 -> El servidor ha recibido satisfactoriamente su mensaje 'Mensaje de prueba'.
5  2019-10-05 01:57:39 -> El servidor ha recibido satisfactoriamente su mensaje 'INF343'.
6  2019-10-05 01:57:39 -> El servidor ha recibido satisfactoriamente su mensaje 'Saludos'.
7  2019-10-05 01:57:39 -> El servidor ha recibido satisfactoriamente su mensaje 'Sistemas'.
8  2019-10-05 01:57:39 -> El servidor ha recibido satisfactoriamente su mensaje 'exit'.
```

ACTIVIDAD 2

Árbol de archivos



docker-compose.yml

docker-compose.yml

```yaml
version: '3'

services:
    headnode:
        build: app_headnode/
        volumes:
            - ./app_headnode:/app_headnode
        command: python headnode.py
        image: ubuntu
        ports:
            - "10000:10000"
    datanode1:
        build: app_datanode/
        volumes:
            - ./app_datanode:/app_datanode
        command: python datanode.py
        depends_on:
            - headnode
    datanode2:
        build: app_datanode/
        volumes:
            - ./app_datanode:/app_datanode
        command: python datanode.py
        depends_on:
            - datanode1
    datanode3:
        build: app_datanode/
        volumes:
            - ./app_datanode:/app_datanode
        command: python datanode.py
        depends_on:
            - datanode2
    client:
        build: app_client/
        volumes:
            - ./app_client:/app_client
        command: python client.py
        depends_on:
            - datanode3
```

ACTIVIDAD 2

Dockerfile / Servidor

Dockerfile

```
FROM ubuntu:latest

RUN apt-get update \
    && apt-get -y install iputils-ping \
    && apt-get -y install iproute2 \
    && apt-get -y install dnsutils \
    && apt-get -y install python3 \
    && apt-get -y install python3-pip \
    && apt-get -y install vim \
    && apt-get -y install sqlite

ADD headnode.py /app_headnode/

WORKDIR /app_headnode/
```

Dockerfile / Cliente

Dockerfile

```
FROM ubuntu:latest

RUN apt-get update \
    && apt-get -y install iputils-ping \
    && apt-get -y install iproute2 \
    && apt-get -y install dnsutils \
    && apt-get -y install python3 \
    && apt-get -y install python3-pip \
    && apt-get -y install vim \
    && apt-get -y install sqlite

ADD client.py /app_client/

WORKDIR /app_client/
```

Dockerfile / datanode

Dockerfile

```
FROM ubuntu:latest

RUN apt-get update \
    && apt-get -y install iputils-ping \
    && apt-get -y install iproute2 \
    && apt-get -y install dnsutils \
    && apt-get -y install python3 \
    && apt-get -y install python3-pip \
    && apt-get -y install vim \
    && apt-get -y install sqlite

ADD datanode.py /app_datanode/

WORKDIR /app_datanode/
```

ACTIVIDAD 2

Headnode:

1. Servidor se mantiene activo esperando conexión de cliente en el puerto 10000.
2. Maneja conexión de datanodes a través de threads.
3. Chequea conexión con datanodes a través de pings a estos.

headnode.py

```python
import socket, threading, struct, os
from _thread import *
import random, datetime
from time import sleep

print("Servidor iniciado.")
print("Esperando consultas por parte del cliente...\n")

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server.bind(("", 10000))

connections = []
addresses = []

while True:
    server.listen(5)
    sock, address = server.accept()

    if len(connections) <= 3:
        start_new_thread(threaded, (sock,))
        connections.append(sock)
        if len(connections) <= 3:
            addresses.append("datanode" + str(len(connections)))
            print("Se ha creado satisfactoriamente un datanode.\n")
        elif len(connections) == 4:
            addresses.append("client")
            print("Se ha establecido conexión con el cliente.\n")

            while True:
                sleep(5)
                for i in addresses[0:3]:
                    with open("heartbeat_server.txt", "a+") as file:
                        response = os.system("ping -c 1 " + i)
                        date_now = datetime.datetime.now()
                        if response == 0:
                            file.write(date_now.strftime("%Y-%m-%d %H:%M:%S") + " -> El " + i + " aun está activo.\n")
                        else:
                            file.write(date_now.strftime("%Y-%m-%d %H:%M:%S") + " -> El " + i + " se encuentra inactivo.\n")
    else:
        print("El servidor no soporta más conexiones.\n")
```

ACTIVIDAD 2

Headnode:

1. Al recibir un mensaje del cliente elije uno de los datanodes al azar y envía el mensaje del cliente a este datanode.

2. Al recibir un mensaje del datanode envía un mensaje al cliente avisándole que su mensaje fue registrado con éxito.

```
43  def threaded(c):
44      while True:
45          data = c.recv(2048).decode().split(" | ")
46          if data[0] == 'client':
47              node = random.randint(0, 2)
48              connection = connections[node]
49              connection.send((str(node) + " | " + data[1]).encode())
50          elif data[0] == 'datanode':
51              date_now = datetime.datetime.now()
52              connections[3].send(("Su mensaje '" + data[1].split("'")[1] +"' ha sido registrado en el datanode " + str(data[1].split(" ")[2]) + ".").enco
53              with open("registro_server.txt", "a+") as file:
54                  file.write(date_now.strftime("%Y-%m-%d %H:%M:%S") + " -> " + data[1] + "\n")
55          elif data[0] == "close":
56              del connections[-1]
57              print("Se ha perdido la conexión con el cliente.\n")
58              break
59      c.close()
```

ACTIVIDAD 2

Datanode:

1. Al recibir un mensaje del headnode, registra el mensaje y responde que fue registrado con éxito.

datanode.py

```python
import socket, struct
import datetime, os

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(("headnode", 10000)) # Esto debería depender de la IP del server.

while True:
    data = client.recv(2048)
    headnode_message = data.decode().split(" | ")
    if (headnode_message[0] == "0") or (headnode_message[0] == "1") or (headnode_message[0] == "2"):
        try:
            os.mkdir('datanode' + headnode_message[0])
        except FileExistsError:
            print("Directory " , 'datanode' + str(headnode_message[0]) ,  " already exists")

        date_now = datetime.datetime.now()
        answer = ("datanode | El datanode " + headnode_message[0] + " ha registrado satisfactoriamente el mensaje: '" + headnode_message[1] + "'.").encode()
        client.send(answer)
        with open("datanode" + headnode_message[0] + "/data.txt", "a+") as file:
            file.write(date_now.strftime("%Y-%m-%d %H:%M:%S") + " -> " + headnode_message[1] + "\n")
    #elif headnode_message[0] == "close":
        #break;
client.close()
```

ACTIVIDAD 2

client.py

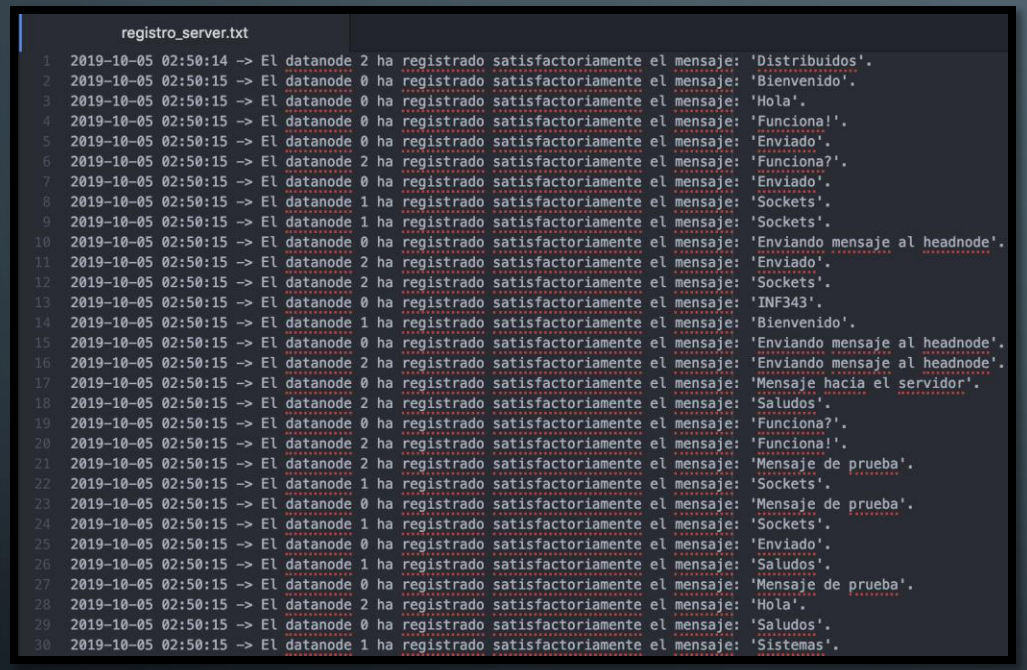
```python
import socket
import random, datetime
from time import sleep

messages_dict = ["Hola", "Sistemas", "Mensaje de prueba", "Funciona!", "Saludos", "Distribuidos",
                 "Bienvenido", "Mensaje hacia el servidor", "Mensaje de prueba", "Enviado", "Adiós",
                 "INF343", "Sockets", "Enviando mensaje al headnode", "Funciona?", "exit"]

positions = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
probs = [0.066, 0.066, 0.066, 0.066, 0.066, 0.066, 0.066, 0.066, 0.066, 0.066, 0.066, 0.066, 0.066, 0.066, 0.066, 0.01]

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(("headnode", 10000)) # Esto debería depender de la IP del server.

while True:
    message = messages_dict[random.choices(positions, probs)[0]]
    client.send(("client | " + message).encode())
    server_answer = client.recv(2048).decode()
    print(server_answer)
    date_now = datetime.datetime.now()
    with open("registro_cliente.txt", "a+") as file:
        file.write(date_now.strftime("%Y-%m-%d %H:%M:%S") + " -> " + server_answer + "\n")
    if message == "exit":
        print("Se ha cerrado con éxito la conexión.")
        client.send(("close").encode())
        break
client.close()
```

Cliente:

1. Conexión a un servidor en puerto 10000.
2. Envía mensajes predeterminados de manera aleatoria al servidor.
3. Registra respuestas del servidor en archivo de texto.
4. Cliente termina la conexión al enviar “exit” y termina su ejecución.

ACTIVIDAD 2

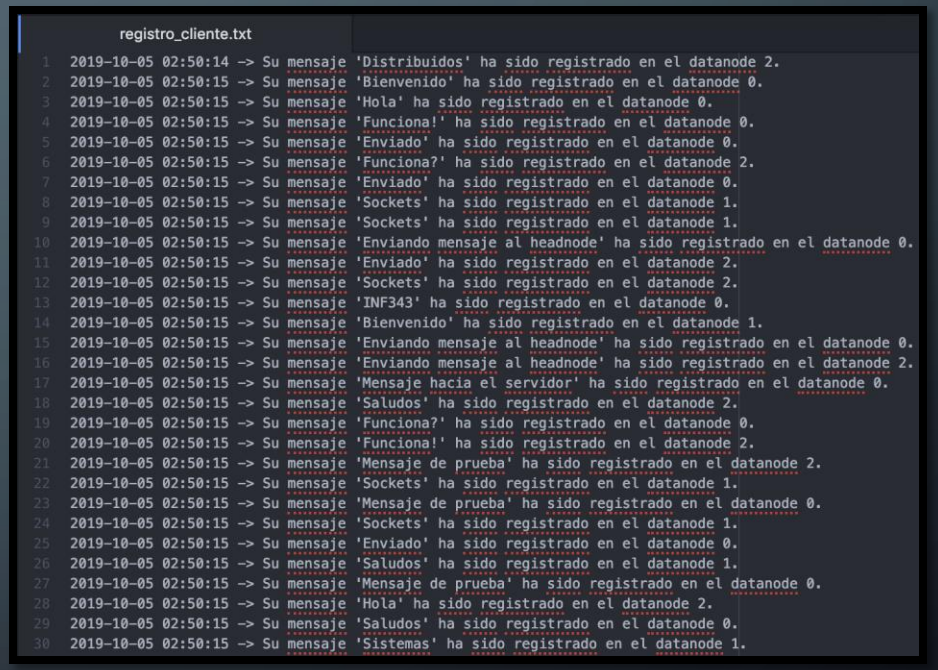
Registro_server.txt



registro_server.txt

```
2019-10-05 02:50:14 -> El datanode 2 ha registrado satisfactoriamente el mensaje: 'Distribuidos'.
2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Bienvenido'.
2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Hola'.
2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Funciona!'.
2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Enviado'.
2019-10-05 02:50:15 -> El datanode 2 ha registrado satisfactoriamente el mensaje: 'Funciona?'.
2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Enviado'.
2019-10-05 02:50:15 -> El datanode 1 ha registrado satisfactoriamente el mensaje: 'Sockets'.
2019-10-05 02:50:15 -> El datanode 1 ha registrado satisfactoriamente el mensaje: 'Sockets'.
2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Enviando mensaje al headnode'.
2019-10-05 02:50:15 -> El datanode 2 ha registrado satisfactoriamente el mensaje: 'Enviado'.
2019-10-05 02:50:15 -> El datanode 2 ha registrado satisfactoriamente el mensaje: 'Sockets'.
2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'INF343'.
2019-10-05 02:50:15 -> El datanode 1 ha registrado satisfactoriamente el mensaje: 'Bienvenido'.
2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Enviando mensaje al headnode'.
2019-10-05 02:50:15 -> El datanode 2 ha registrado satisfactoriamente el mensaje: 'Enviando mensaje al headnode'.
2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Mensaje hacia el servidor'.
2019-10-05 02:50:15 -> El datanode 2 ha registrado satisfactoriamente el mensaje: 'Saludos'.
2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Funciona?'.
2019-10-05 02:50:15 -> El datanode 2 ha registrado satisfactoriamente el mensaje: 'Funciona!'.
2019-10-05 02:50:15 -> El datanode 2 ha registrado satisfactoriamente el mensaje: 'Mensaje de prueba'.
2019-10-05 02:50:15 -> El datanode 1 ha registrado satisfactoriamente el mensaje: 'Sockets'.
2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Mensaje de prueba'.
2019-10-05 02:50:15 -> El datanode 1 ha registrado satisfactoriamente el mensaje: 'Sockets'.
2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Enviado'.
2019-10-05 02:50:15 -> El datanode 1 ha registrado satisfactoriamente el mensaje: 'Saludos'.
2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Mensaje de prueba'.
2019-10-05 02:50:15 -> El datanode 2 ha registrado satisfactoriamente el mensaje: 'Hola'.
2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Saludos'.
2019-10-05 02:50:15 -> El datanode 1 ha registrado satisfactoriamente el mensaje: 'Sistemas'.
```

Registro_cliente.txt

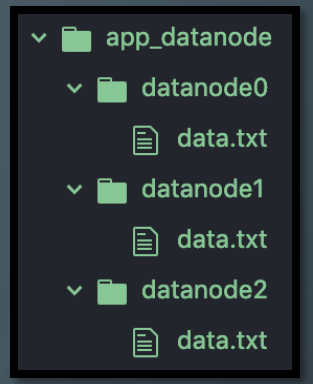


registro_cliente.txt

```
2019-10-05 02:50:14 -> Su mensaje 'Distribuidos' ha sido registrado en el datanode 2.
2019-10-05 02:50:15 -> Su mensaje 'Bienvenido' ha sido registrado en el datanode 0.
2019-10-05 02:50:15 -> Su mensaje 'Hola' ha sido registrado en el datanode 0.
2019-10-05 02:50:15 -> Su mensaje 'Funciona!' ha sido registrado en el datanode 0.
2019-10-05 02:50:15 -> Su mensaje 'Enviado' ha sido registrado en el datanode 0.
2019-10-05 02:50:15 -> Su mensaje 'Funciona?' ha sido registrado en el datanode 2.
2019-10-05 02:50:15 -> Su mensaje 'Enviado' ha sido registrado en el datanode 0.
2019-10-05 02:50:15 -> Su mensaje 'Sockets' ha sido registrado en el datanode 1.
2019-10-05 02:50:15 -> Su mensaje 'Sockets' ha sido registrado en el datanode 1.
2019-10-05 02:50:15 -> Su mensaje 'Enviando mensaje al headnode' ha sido registrado en el datanode 0.
2019-10-05 02:50:15 -> Su mensaje 'Enviado' ha sido registrado en el datanode 2.
2019-10-05 02:50:15 -> Su mensaje 'Sockets' ha sido registrado en el datanode 2.
2019-10-05 02:50:15 -> Su mensaje 'INF343' ha sido registrado en el datanode 0.
2019-10-05 02:50:15 -> Su mensaje 'Bienvenido' ha sido registrado en el datanode 1.
2019-10-05 02:50:15 -> Su mensaje 'Enviando mensaje al headnode' ha sido registrado en el datanode 0.
2019-10-05 02:50:15 -> Su mensaje 'Enviando mensaje al headnode' ha sido registrado en el datanode 2.
2019-10-05 02:50:15 -> Su mensaje 'Mensaje hacia el servidor' ha sido registrado en el datanode 0.
2019-10-05 02:50:15 -> Su mensaje 'Saludos' ha sido registrado en el datanode 2.
2019-10-05 02:50:15 -> Su mensaje 'Funciona?' ha sido registrado en el datanode 0.
2019-10-05 02:50:15 -> Su mensaje 'Funciona!' ha sido registrado en el datanode 2.
2019-10-05 02:50:15 -> Su mensaje 'Mensaje de prueba' ha sido registrado en el datanode 2.
2019-10-05 02:50:15 -> Su mensaje 'Sockets' ha sido registrado en el datanode 1.
2019-10-05 02:50:15 -> Su mensaje 'Mensaje de prueba' ha sido registrado en el datanode 0.
2019-10-05 02:50:15 -> Su mensaje 'Sockets' ha sido registrado en el datanode 1.
2019-10-05 02:50:15 -> Su mensaje 'Enviado' ha sido registrado en el datanode 0.
2019-10-05 02:50:15 -> Su mensaje 'Saludos' ha sido registrado en el datanode 1.
2019-10-05 02:50:15 -> Su mensaje 'Mensaje de prueba' ha sido registrado en el datanode 0.
2019-10-05 02:50:15 -> Su mensaje 'Hola' ha sido registrado en el datanode 2.
2019-10-05 02:50:15 -> Su mensaje 'Saludos' ha sido registrado en el datanode 0.
2019-10-05 02:50:15 -> Su mensaje 'Sistemas' ha sido registrado en el datanode 1.
```

ACTIVIDAD 2

Directorios de almacenamiento



data.txt

data.txt

```
2019-10-05 02:50:15 -> Bienvenido
2019-10-05 02:50:15 -> Hola
2019-10-05 02:50:15 -> Funciona!
2019-10-05 02:50:15 -> Enviado
2019-10-05 02:50:15 -> Enviado
2019-10-05 02:50:15 -> Enviando mensaje al headnode
2019-10-05 02:50:15 -> INF343
2019-10-05 02:50:15 -> Enviando mensaje al headnode
2019-10-05 02:50:15 -> Mensaje hacia el servidor
2019-10-05 02:50:15 -> Funciona?
2019-10-05 02:50:15 -> Mensaje de prueba
2019-10-05 02:50:15 -> Enviado
2019-10-05 02:50:15 -> Mensaje de prueba
2019-10-05 02:50:15 -> Saludos
2019-10-05 02:50:15 -> Bienvenido
2019-10-05 02:50:15 -> Funciona!
2019-10-05 02:50:15 -> Funciona!
2019-10-05 02:50:15 -> Mensaje hacia el servidor
2019-10-05 02:50:15 -> Adiós
2019-10-05 02:50:15 -> Distribuidos
2019-10-05 02:50:15 -> Mensaje de prueba
2019-10-05 02:50:15 -> INF343
2019-10-05 02:50:15 -> INF343
2019-10-05 02:50:15 -> Mensaje de prueba
2019-10-05 02:50:15 -> Sockets
2019-10-05 02:50:15 -> Mensaje de prueba
2019-10-05 02:50:15 -> Mensaje hacia el servidor
2019-10-05 02:50:15 -> Distribuidos
2019-10-05 02:50:15 -> Enviado
2019-10-05 02:50:15 -> Enviando mensaje al headnode
```

data.txt

```
2019-10-05 02:50:15 -> Sockets
2019-10-05 02:50:15 -> Sockets
2019-10-05 02:50:15 -> Bienvenido
2019-10-05 02:50:15 -> Sockets
2019-10-05 02:50:15 -> Sockets
2019-10-05 02:50:15 -> Saludos
2019-10-05 02:50:15 -> Sistemas
2019-10-05 02:50:15 -> Enviando mensaje al headnode
2019-10-05 02:50:15 -> Mensaje de prueba
2019-10-05 02:50:15 -> Adiós
2019-10-05 02:50:15 -> Enviando mensaje al headnode
2019-10-05 02:50:15 -> Adiós
2019-10-05 02:50:15 -> Sistemas
2019-10-05 02:50:15 -> Adiós
2019-10-05 02:50:15 -> Bienvenido
2019-10-05 02:50:15 -> Bienvenido
2019-10-05 02:50:15 -> Enviado
2019-10-05 02:50:15 -> Mensaje de prueba
2019-10-05 02:50:15 -> Hola
2019-10-05 02:50:15 -> Mensaje de prueba
2019-10-05 02:50:15 -> Enviando mensaje al headnode
2019-10-05 02:50:15 -> Sistemas
2019-10-05 02:50:15 -> Mensaje de prueba
2019-10-05 02:50:15 -> Funciona?
2019-10-05 02:50:15 -> Saludos
2019-10-05 02:50:15 -> Mensaje de prueba
2019-10-05 02:50:15 -> Enviado
2019-10-05 02:50:15 -> Adiós
2019-10-05 02:50:15 -> Enviando mensaje al headnode
2019-10-05 02:50:15 -> INF343
```

data.txt

```
2019-10-05 02:50:14 -> Distribuidos
2019-10-05 02:50:15 -> Funciona?
2019-10-05 02:50:15 -> Enviado
2019-10-05 02:50:15 -> Sockets
2019-10-05 02:50:15 -> Enviando mensaje al headnode
2019-10-05 02:50:15 -> Saludos
2019-10-05 02:50:15 -> Funciona!
2019-10-05 02:50:15 -> Mensaje de prueba
2019-10-05 02:50:15 -> Hola
2019-10-05 02:50:15 -> Sistemas
2019-10-05 02:50:15 -> Mensaje de prueba
2019-10-05 02:50:15 -> Adiós
2019-10-05 02:50:15 -> Adiós
2019-10-05 02:50:15 -> Funciona?
2019-10-05 02:50:15 -> Saludos
2019-10-05 02:50:15 -> Mensaje de prueba
2019-10-05 02:50:15 -> Saludos
2019-10-05 02:50:15 -> Enviado
2019-10-05 02:50:15 -> Enviando mensaje al headnode
2019-10-05 02:50:15 -> Adiós
2019-10-05 02:50:15 -> INF343
2019-10-05 02:50:15 -> Enviando mensaje al headnode
2019-10-05 02:50:15 -> Mensaje de prueba
2019-10-05 02:50:15 -> Hola
2019-10-05 02:50:15 -> INF343
2019-10-05 02:50:15 -> Enviando mensaje al headnode
2019-10-05 02:50:15 -> Saludos
2019-10-05 02:50:15 -> INF343
2019-10-05 02:50:15 -> INF343
2019-10-05 02:50:15 -> Funciona?
```

PROBLEMAS ENCONTRADOS

1. Familiarización con Docker.
2. Mensajes ingresados por el usuario de manera manual.
3. Uso de multicast con Docker *(comandos build y up)*.
4. Comunicación entre nodos con diferente IP *('hostname')*.
5. Gestión del tiempo entre mensajes y ping *(sleep)*.