



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:

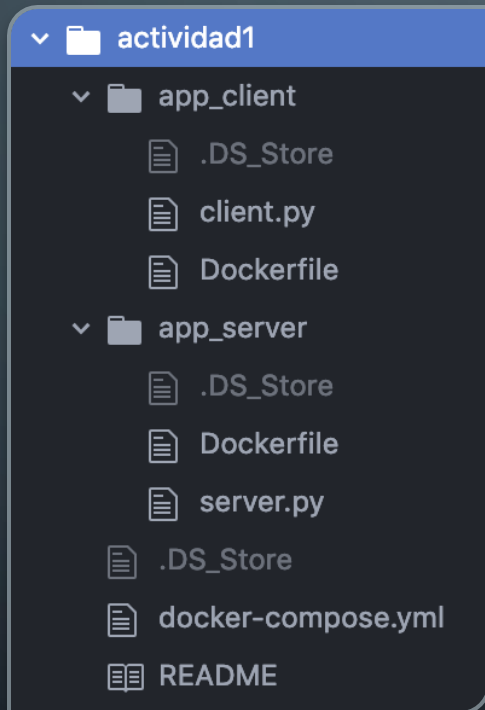
Un cliente y un servidor se comunican vía mensajes, registrando cada uno la respuesta del otro.

Actividad 2:

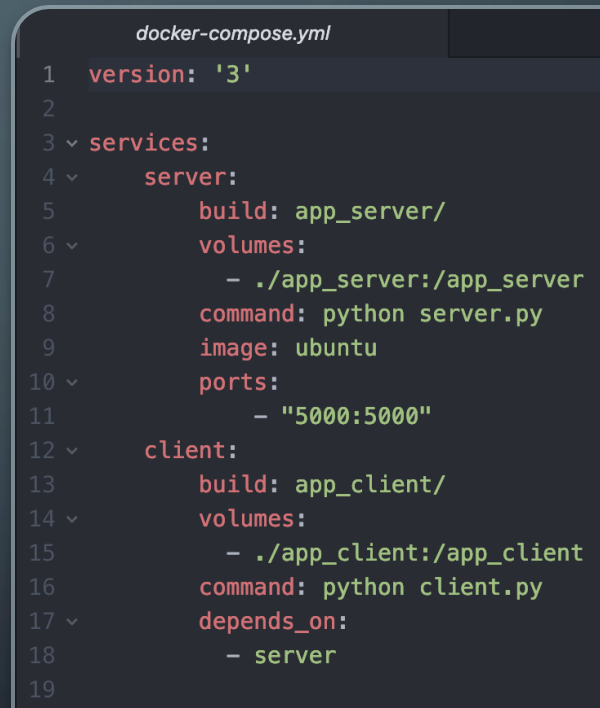
Un cliente envía mensajes a un servidor y estos se registran en uno de los 3 nodos de almacenamiento interno que están conectados directamente al servidor.

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.

```
server.py
1 import socket
2
3 print("Servidor iniciado.")
4 print("Esperando consultas por parte del cliente...\n")
5
6 server_file_name = 'log.txt'
7 server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8 server.bind(('', 5000))
9 server.listen(10)
10
11 while True:
12     conn, addr = server.accept()
13     while True:
14         data = conn.recv(4096)
15         if not data: break
16         print("Se ha establecido conexión con el cliente.\n")
17         file = open(server_file_name, "a+")
18         file.write(str(addr[0]) + ":" + str(addr[1]) + " -> " + data.decode() + "\n")
19         file.close()
20         print("Mensaje recibido y registrado correctamente.\n")
21         conn.send(("El servidor ha recibido satisfactoriamente su mensaje '" + data.decode() + "'.").encode())
22         print("Se ha perdido la conexión con el cliente.\n")
23         conn.close()
24     server.close()
25
```


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
1 import socket
2 import datetime, random
3
4 messages_dict = ["Hola", "Sistemas", "Mensaje de prueba", "Funciona!", "Saludos", "Distribuidos",
5                 "Bienvenido", "Mensaje hacia el servidor", "Mensaje de prueba", "Enviado", "Adiós",
6                 "INF343", "Sockets", "Enviando mensaje al headnode", "Funciona?", "Ayuda", "exit"]
7
8 client_file_name = 'respuestas.txt'
9 client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10 client.connect(("server", 5000)) # Esto debería depender de la IP del server.
11
12 while True:
13     message = messages_dict[random.randint(0, len(messages_dict)-1)]
14     client.send(message.encode())
15     server_answer = client.recv(4096).decode()
16     date_now = datetime.datetime.now()
17     print(server_answer)
18     with open(client_file_name, "a+") as file:
19         file.write(date_now.strftime("%Y-%m-%d %H:%M:%S") + " -> " + server_answer + "\n")
20     if message == "exit":
21         print("Se ha cerrado con éxito la conexión.")
22         break
23 client.close()
24
```

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

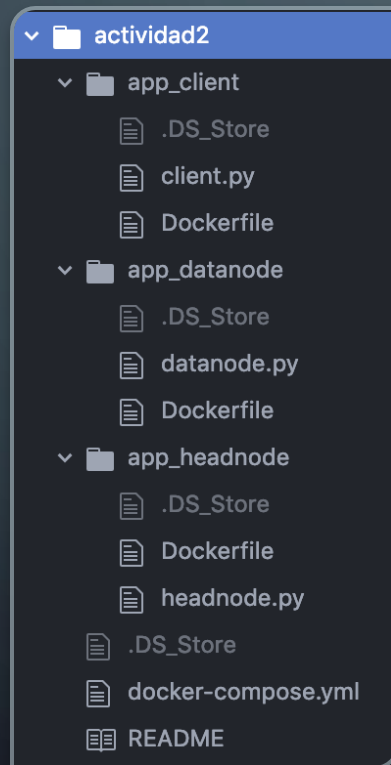
Cliente

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
1 version: '3'
2
3 services:
4   headnode:
5     build: app_headnode/
6     volumes:
7       - ./app_headnode:/app_headnode
8     command: python headnode.py
9     image: ubuntu
10    ports:
11      - "10000:10000"
12
13   datanode1:
14     build: app_datanode/
15     volumes:
16       - ./app_datanode:/app_datanode
17     command: python datanode.py
18     depends_on:
19       - headnode
20
21   datanode2:
22     build: app_datanode/
23     volumes:
24       - ./app_datanode:/app_datanode
25     command: python datanode.py
26     depends_on:
27       - datanode1
28
29   datanode3:
30     build: app_datanode/
31     volumes:
32       - ./app_datanode:/app_datanode
33     command: python datanode.py
34     depends_on:
35       - datanode2
36
37   client:
38     build: app_client/
39     volumes:
40       - ./app_client:/app_client
41     command: python client.py
42     depends_on:
43       - datanode3
```

ACTIVIDAD 2

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 headnode.py /app_headnode/
13
14 WORKDIR /app_headnode/
```

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/
```

Dockerfile / datanode

```
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 datanode.py /app_datanode/
13
14 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
1 import socket, threading, struct, os
2 from _thread import *
3 import random, datetime
4 from time import sleep
5
6 print("Servidor iniciado.")
7 print("Esperando consultas por parte del cliente...\n")
8
9 server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10 server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
11 server.bind(("", 10000))
12
13 connections = []
14 addresses = []
15
16 while True:
17     server.listen(5)
18     sock, address = server.accept()
19
20     if len(connections) <= 3:
21         start_new_thread(threaded, (sock,))
22         connections.append(sock)
23         if len(connections) <= 3:
24             addresses.append("datanode" + str(len(connections)))
25             print("Se ha creado satisfactoriamente un datanode.\n")
26         elif len(connections) == 4:
27             addresses.append("client")
28             print("Se ha establecido conexión con el cliente.\n")
29
30     while True:
31         sleep(5)
32         for i in addresses[0:3]:
33             with open("heartbeat_server.txt", "a+") as file:
34                 response = os.system("ping -c 1 " + i)
35                 date_now = datetime.datetime.now()
36                 if response == 0:
37                     file.write(date_now.strftime("%Y-%m-%d %H:%M:%S") + " -> El " + i + " aun está activo.\n")
38                 else:
39                     file.write(date_now.strftime("%Y-%m-%d %H:%M:%S") + " -> El " + i + " se encuentra inactivo.\n")
40     else:
41         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]) + ".").encode())
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
1 import socket, struct
2 import datetime, os
3
4 client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5 client.connect(("headnode", 10000)) # Esto debería depender de la IP del server.
6
7 while True:
8     data = client.recv(2048)
9     headnode_message = data.decode().split(" | ")
10    if (headnode_message[0] == "0") or (headnode_message[0] == "1") or (headnode_message[0] == "2"):
11        try:
12            os.mkdir('datanode' + headnode_message[0])
13        except FileExistsError:
14            print("Directory " , 'datanode' + str(headnode_message[0]) , " already exists")
15
16        date_now = datetime.datetime.now()
17        answer = ("datanode | El datanode " + headnode_message[0] + " ha registrado satisfactoriamente el mensaje: '" + headnode_message[1] + "'.").encode()
18        client.send(answer)
19        with open("datanode" + headnode_message[0] + "/data.txt", "a+") as file:
20            file.write(date_now.strftime("%Y-%m-%d %H:%M:%S") + " -> " + headnode_message[1] + "\n")
21    elif headnode_message[0] == "close":
22        #break;
23    client.close()
```

ACTIVIDAD 2

```
client.py
1 import socket
2 import random, datetime
3 from time import sleep
4
5 messages_dict = ["Hola", "Sistemas", "Mensaje de prueba", "Funciona!", "Saludos", "Distribuidos",
6                 "Bienvenido", "Mensaje hacia el servidor", "Mensaje de prueba", "Enviado", "Adiós",
7                 "INF343", "Sockets", "Enviando mensaje al headnode", "Funciona?", "exit"]
8
9 positions = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
10 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]
11
12 client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
13 client.connect(("headnode", 10000)) # Esto debería depender de la IP del server.
14
15 while True:
16     message = messages_dict[random.choices(positions, probs)[0]]
17     client.send(("client | " + message).encode())
18     server_answer = client.recv(2048).decode()
19     print(server_answer)
20     date_now = datetime.datetime.now()
21     with open("registro_cliente.txt", "a+") as file:
22         file.write(date_now.strftime("%Y-%m-%d %H:%M:%S") + " -> " + server_answer + "\n")
23     if message == "exit":
24         print("Se ha cerrado con éxito la conexión.")
25         client.send(("close").encode())
26         break
27 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
1 2019-10-05 02:50:14 -> El datanode 2 ha registrado satisfactoriamente el mensaje: 'Distribuidos'.
2 2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Bienvenido'.
3 2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Hola'.
4 2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Funciona!'.
5 2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Enviado'.
6 2019-10-05 02:50:15 -> El datanode 2 ha registrado satisfactoriamente el mensaje: 'Funciona?'.
7 2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Enviado'.
8 2019-10-05 02:50:15 -> El datanode 1 ha registrado satisfactoriamente el mensaje: 'Sockets'.
9 2019-10-05 02:50:15 -> El datanode 1 ha registrado satisfactoriamente el mensaje: 'Sockets'.
10 2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Enviando mensaje al headnode'.
11 2019-10-05 02:50:15 -> El datanode 2 ha registrado satisfactoriamente el mensaje: 'Enviado'.
12 2019-10-05 02:50:15 -> El datanode 2 ha registrado satisfactoriamente el mensaje: 'Sockets'.
13 2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'INF343'.
14 2019-10-05 02:50:15 -> El datanode 1 ha registrado satisfactoriamente el mensaje: 'Bienvenido'.
15 2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Enviando mensaje al headnode'.
16 2019-10-05 02:50:15 -> El datanode 2 ha registrado satisfactoriamente el mensaje: 'Enviando mensaje al headnode'.
17 2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Mensaje hacia el servidor'.
18 2019-10-05 02:50:15 -> El datanode 2 ha registrado satisfactoriamente el mensaje: 'Saludos'.
19 2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Funciona?'.
20 2019-10-05 02:50:15 -> El datanode 2 ha registrado satisfactoriamente el mensaje: 'Funciona!'.
21 2019-10-05 02:50:15 -> El datanode 2 ha registrado satisfactoriamente el mensaje: 'Mensaje de prueba'.
22 2019-10-05 02:50:15 -> El datanode 1 ha registrado satisfactoriamente el mensaje: 'Sockets'.
23 2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Mensaje de prueba'.
24 2019-10-05 02:50:15 -> El datanode 1 ha registrado satisfactoriamente el mensaje: 'Sockets'.
25 2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Enviado'.
26 2019-10-05 02:50:15 -> El datanode 1 ha registrado satisfactoriamente el mensaje: 'Saludos'.
27 2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Mensaje de prueba'.
28 2019-10-05 02:50:15 -> El datanode 2 ha registrado satisfactoriamente el mensaje: 'Hola'.
29 2019-10-05 02:50:15 -> El datanode 0 ha registrado satisfactoriamente el mensaje: 'Saludos'.
30 2019-10-05 02:50:15 -> El datanode 1 ha registrado satisfactoriamente el mensaje: 'Sistemas'.
```

Registro_cliente.txt

```
registro_cliente.txt
1 2019-10-05 02:50:14 -> Su mensaje 'Distribuidos' ha sido registrado en el datanode 2.
2 2019-10-05 02:50:15 -> Su mensaje 'Bienvenido' ha sido registrado en el datanode 0.
3 2019-10-05 02:50:15 -> Su mensaje 'Hola' ha sido registrado en el datanode 0.
4 2019-10-05 02:50:15 -> Su mensaje 'Funciona!' ha sido registrado en el datanode 0.
5 2019-10-05 02:50:15 -> Su mensaje 'Enviado' ha sido registrado en el datanode 0.
6 2019-10-05 02:50:15 -> Su mensaje 'Funciona?' ha sido registrado en el datanode 2.
7 2019-10-05 02:50:15 -> Su mensaje 'Enviado' ha sido registrado en el datanode 0.
8 2019-10-05 02:50:15 -> Su mensaje 'Sockets' ha sido registrado en el datanode 1.
9 2019-10-05 02:50:15 -> Su mensaje 'Sockets' ha sido registrado en el datanode 1.
10 2019-10-05 02:50:15 -> Su mensaje 'Enviando mensaje al headnode' ha sido registrado en el datanode 0.
11 2019-10-05 02:50:15 -> Su mensaje 'Enviado' ha sido registrado en el datanode 2.
12 2019-10-05 02:50:15 -> Su mensaje 'Sockets' ha sido registrado en el datanode 2.
13 2019-10-05 02:50:15 -> Su mensaje 'INF343' ha sido registrado en el datanode 0.
14 2019-10-05 02:50:15 -> Su mensaje 'Bienvenido' ha sido registrado en el datanode 1.
15 2019-10-05 02:50:15 -> Su mensaje 'Enviando mensaje al headnode' ha sido registrado en el datanode 0.
16 2019-10-05 02:50:15 -> Su mensaje 'Enviando mensaje al headnode' ha sido registrado en el datanode 2.
17 2019-10-05 02:50:15 -> Su mensaje 'Mensaje hacia el servidor' ha sido registrado en el datanode 0.
18 2019-10-05 02:50:15 -> Su mensaje 'Saludos' ha sido registrado en el datanode 2.
19 2019-10-05 02:50:15 -> Su mensaje 'Funciona?' ha sido registrado en el datanode 0.
20 2019-10-05 02:50:15 -> Su mensaje 'Funciona!' ha sido registrado en el datanode 2.
21 2019-10-05 02:50:15 -> Su mensaje 'Mensaje de prueba' ha sido registrado en el datanode 2.
22 2019-10-05 02:50:15 -> Su mensaje 'Sockets' ha sido registrado en el datanode 1.
23 2019-10-05 02:50:15 -> Su mensaje 'Mensaje de prueba' ha sido registrado en el datanode 0.
24 2019-10-05 02:50:15 -> Su mensaje 'Sockets' ha sido registrado en el datanode 1.
25 2019-10-05 02:50:15 -> Su mensaje 'Enviado' ha sido registrado en el datanode 0.
26 2019-10-05 02:50:15 -> Su mensaje 'Saludos' ha sido registrado en el datanode 1.
27 2019-10-05 02:50:15 -> Su mensaje 'Mensaje de prueba' ha sido registrado en el datanode 0.
28 2019-10-05 02:50:15 -> Su mensaje 'Hola' ha sido registrado en el datanode 2.
29 2019-10-05 02:50:15 -> Su mensaje 'Saludos' ha sido registrado en el datanode 0.
30 2019-10-05 02:50:15 -> Su mensaje 'Sistemas' ha sido registrado en el datanode 1.
```

ACTIVIDAD 2

Directorios de
almacenamiento

```
▼ app_datanode
  ▼ datanode0
    data.txt
  ▼ datanode1
    data.txt
  ▼ datanode2
    data.txt
```

data.txt

data.txt	data.txt	data.txt
1 2019-10-05 02:50:15 -> Bienvenido	1 2019-10-05 02:50:15 -> Sockets	1 2019-10-05 02:50:14 -> Distribuidos
2 2019-10-05 02:50:15 -> Hola	2 2019-10-05 02:50:15 -> Sockets	2 2019-10-05 02:50:15 -> Funciona?
3 2019-10-05 02:50:15 -> Funciona!	3 2019-10-05 02:50:15 -> Bienvenido	3 2019-10-05 02:50:15 -> Enviado
4 2019-10-05 02:50:15 -> Enviado	4 2019-10-05 02:50:15 -> Sockets	4 2019-10-05 02:50:15 -> Sockets
5 2019-10-05 02:50:15 -> Enviado	5 2019-10-05 02:50:15 -> Sockets	5 2019-10-05 02:50:15 -> Enviando mensaje al headnode
6 2019-10-05 02:50:15 -> Enviando mensaje al headnode	6 2019-10-05 02:50:15 -> Saludos	6 2019-10-05 02:50:15 -> Saludos
7 2019-10-05 02:50:15 -> INF343	7 2019-10-05 02:50:15 -> Sistemas	7 2019-10-05 02:50:15 -> Funciona!
8 2019-10-05 02:50:15 -> Enviando mensaje al headnode	8 2019-10-05 02:50:15 -> Enviando mensaje al headnode	8 2019-10-05 02:50:15 -> Mensaje de prueba
9 2019-10-05 02:50:15 -> Mensaje hacia el servidor	9 2019-10-05 02:50:15 -> Mensaje de prueba	9 2019-10-05 02:50:15 -> Hola
10 2019-10-05 02:50:15 -> Funciona?	10 2019-10-05 02:50:15 -> Adiós	10 2019-10-05 02:50:15 -> Sistemas
11 2019-10-05 02:50:15 -> Mensaje de prueba	11 2019-10-05 02:50:15 -> Enviando mensaje al headnode	11 2019-10-05 02:50:15 -> Mensaje de prueba
12 2019-10-05 02:50:15 -> Enviado	12 2019-10-05 02:50:15 -> Adiós	12 2019-10-05 02:50:15 -> Adiós
13 2019-10-05 02:50:15 -> Mensaje de prueba	13 2019-10-05 02:50:15 -> Sistemas	13 2019-10-05 02:50:15 -> Adiós
14 2019-10-05 02:50:15 -> Saludos	14 2019-10-05 02:50:15 -> Adiós	14 2019-10-05 02:50:15 -> Funciona?
15 2019-10-05 02:50:15 -> Bienvenido	15 2019-10-05 02:50:15 -> Bienvenido	15 2019-10-05 02:50:15 -> Saludos
16 2019-10-05 02:50:15 -> Funciona!	16 2019-10-05 02:50:15 -> Bienvenido	16 2019-10-05 02:50:15 -> Mensaje de prueba
17 2019-10-05 02:50:15 -> Funciona!	17 2019-10-05 02:50:15 -> Enviado	17 2019-10-05 02:50:15 -> Saludos
18 2019-10-05 02:50:15 -> Mensaje hacia el servidor	18 2019-10-05 02:50:15 -> Mensaje de prueba	18 2019-10-05 02:50:15 -> Enviado
19 2019-10-05 02:50:15 -> Adiós	19 2019-10-05 02:50:15 -> Hola	19 2019-10-05 02:50:15 -> Enviando mensaje al headnode
20 2019-10-05 02:50:15 -> Distribuidos	20 2019-10-05 02:50:15 -> Mensaje de prueba	20 2019-10-05 02:50:15 -> Adiós
21 2019-10-05 02:50:15 -> Mensaje de prueba	21 2019-10-05 02:50:15 -> Enviando mensaje al headnode	21 2019-10-05 02:50:15 -> INF343
22 2019-10-05 02:50:15 -> INF343	22 2019-10-05 02:50:15 -> Sistemas	22 2019-10-05 02:50:15 -> Enviando mensaje al headnode
23 2019-10-05 02:50:15 -> INF343	23 2019-10-05 02:50:15 -> Mensaje de prueba	23 2019-10-05 02:50:15 -> Mensaje de prueba
24 2019-10-05 02:50:15 -> Mensaje de prueba	24 2019-10-05 02:50:15 -> Funciona?	24 2019-10-05 02:50:15 -> Hola
25 2019-10-05 02:50:15 -> Sockets	25 2019-10-05 02:50:15 -> Saludos	25 2019-10-05 02:50:15 -> INF343
26 2019-10-05 02:50:15 -> Mensaje de prueba	26 2019-10-05 02:50:15 -> Mensaje de prueba	26 2019-10-05 02:50:15 -> Enviando mensaje al headnode
27 2019-10-05 02:50:15 -> Mensaje hacia el servidor	27 2019-10-05 02:50:15 -> Enviado	27 2019-10-05 02:50:15 -> Saludos
28 2019-10-05 02:50:15 -> Distribuidos	28 2019-10-05 02:50:15 -> Adiós	28 2019-10-05 02:50:15 -> INF343
29 2019-10-05 02:50:15 -> Enviado	29 2019-10-05 02:50:15 -> Enviando mensaje al headnode	29 2019-10-05 02:50:15 -> INF343
30 2019-10-05 02:50:15 -> Enviando mensaje al headnode	30 2019-10-05 02:50:15 -> INF343	30 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*).