BÀI THỰC HÀNH

HỌC PHẦN: HỆ PHÂN TÁN

CHƯƠNG 1: TỔNG QUAN VÀ KIẾN TRÚC HPT

1. Web server apache2

B1: Cài đặt web server apache2

```
root@DESKTOP-CKFEBUT:~# sudo service apache2 start
 * Starting Apache httpd web server apache2
 *
root@DESKTOP-CKFEBUT:~# ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 1500
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0xfe<compat,link,site,host>
        loop  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Hình 1: Bật web server apache2 và lấy địa chỉ IP

Câu hỏi 1: Đường dẫn mặc định là `/var/www/html/index.html`

Câu hỏi 2: Cổng ứng dụng mặt định của dịch vụ www là: 80

B2: Cài đặt virtual hosts cho apache2

```
root@DESKTOP-CKFEBUT:~# sudo mkdir /var/www/example.com/public_html
mkdir: cannot create directory ‘/var/www/example.com/public_html’: No such file or directory
root@DESKTOP-CKFEBUT:~# sudo mkdir -p /var/www/example.com/public_html
root@DESKTOP-CKFEBUT:~# sudo mkdir -p /var/www/test.com/public_html
root@DESKTOP-CKFEBUT:~# example.com
example.com: command not found
root@DESKTOP-CKFEBUT:~# sudo chmod -rr 775 /var/www
chmod: cannot access '775': No such file or directory
root@DESKTOP-CKFEBUT:~# sudo chmod -rr 755 /var/www
chmod: cannot access '755': No such file or directory
root@DESKTOP-CKFEBUT:~# sudo chmod -R 755 /var/www
root@DESKTOP-CKFEBUT:~#
```

Hình 2: thay đổi tiên miền nội tuyến và thiết lập quyền cho thư mục

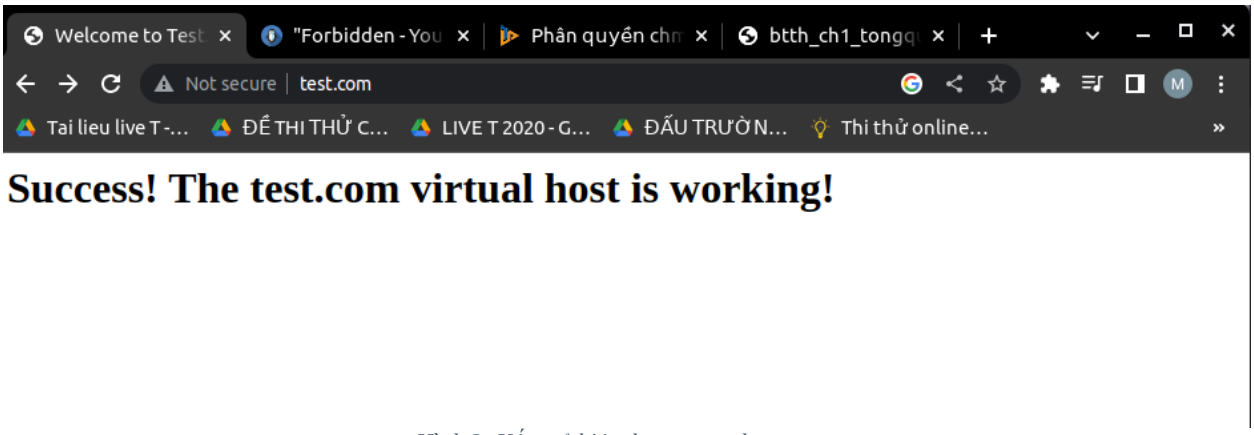
Câu hỏi 3: thiết lập quyền Chmod 775:

7: Người sở hữu thư mục có quyền đọc, chỉnh sửa, liệt kê và thực thi

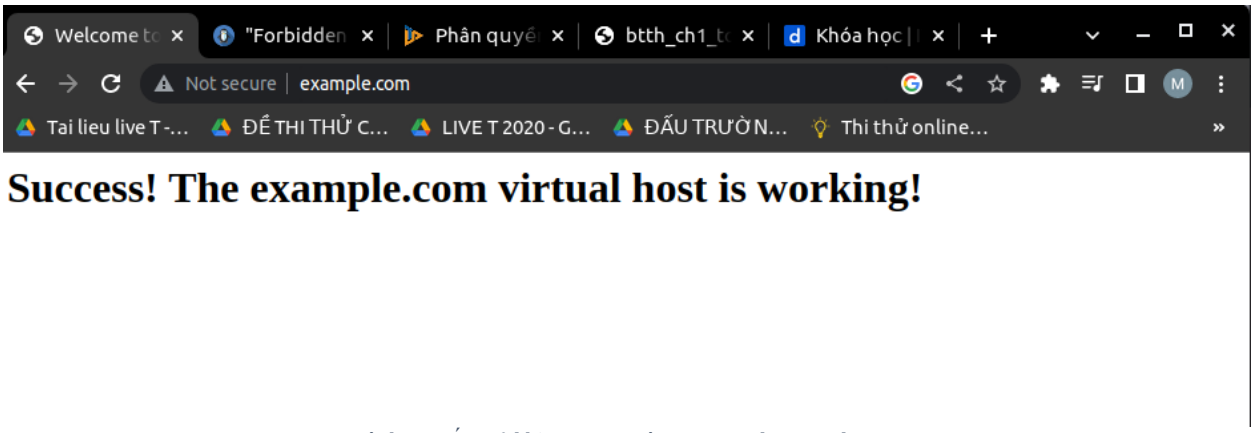
5: Người cùng nhóm chỉ có quyền đọc, liệt kê và thực thi

5: Người còn lại chỉ có quyền đọc, liệt kê và thực thi các thư mục bên trong.

B3: Tạo file index.html 2 trang web và cấu hình cho 2 máy ảo apahce2.



Hình 3: Kết quả hiện thị trang web test.com



Hình 4: Kết quả khi truy cập vào trang web example.com

Câu hỏi 4: Khi truy nhập vào trong web có tên miền như trên, thì trình duyệt sẽ gửi yêu cầu lên sever apache2. Sau đó, apache2 sẽ truy cập vào public_html và lấy nội dung từ file index.html, chạy và hiện thị nội dung lên màn hình.

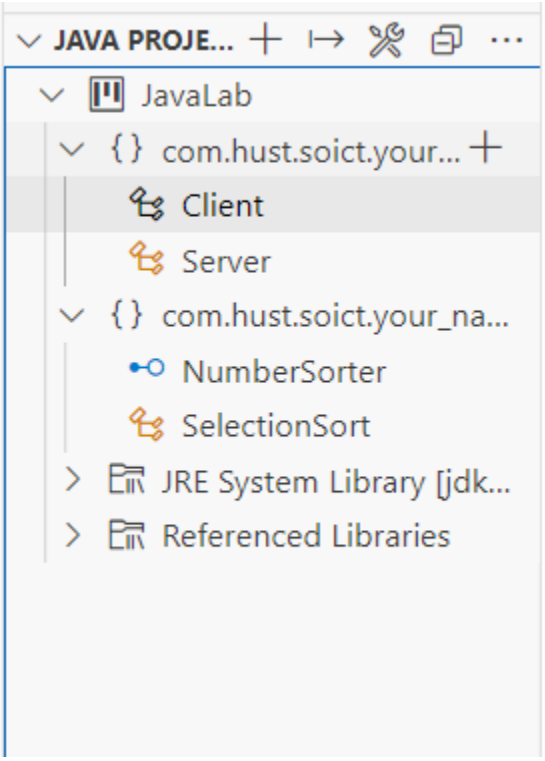
Câu hỏi 5: Khi truy cập từ máy khác mạng thì không truy cập được bởi vì địa chỉ ip là nội vùng trong láy local.

2. Interface trong Java

B1: Cài đặt môi trường

- Trong bài thực hành này em sử dụng môi trường là JDK – 19 và em dùng IDE lập trình là Visual Studio Code với công cụ quản lý dự án Maven.

B2: Xây dựng chương trình



Hình 5: Cấu trúc chương trình

- Code cho class Client:

```
package com.hust.soict.your_name.client_server;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Scanner;

public class Client {
    public static void main(String[] args) throws UnknownHostException, IOExcep-
tion {
        // TODO Auto-generated method stub
        Socket socket = new Socket("localhost", 9898);
        BufferedReader in = new BufferedReader(new InputStreamReader(socket.get-
InputStream()));
        PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
        System.out.println(in.readLine());
```

```
        Scanner scanner = new Scanner(System.in);
        String message;
        // Câu 6: viết đoạn gửi dữ liệu lên sever
        while (true){
            message = scanner.nextLine();
            out.write(message +"\n");
            // out.newLine();
            out.flush();
            if (message.equals("")){
                // out.write("");
                // out.newLine();
                // out.flush();
                break;
            }
        }
        // do {
        //     message = scanner.nextLine();
        // } while (message != "");
        System.out.println(in);
        socket.close();
        scanner.close();
    }
}
```

- Code cho lớp server

```
package com.hust.soict.your_name.client_server;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Arrays;

import com.hust.soict.your_name.helper.*;

public class Server {
    private static class Sorter extends Thread {
        private Socket socket;
        private int clientNumber;
```

```
        public Sorter(Socket socket, int clientNumber) {
            this.socket = socket;
            this.clientNumber = clientNumber;
            System.out.println("New client #" + clientNumber + " connected at
" + socket);
        }

        public void run() {
            try {
                BufferedReader in = new BufferedReader(new In-
putStreamReader(socket.getInputStream()));
                PrintWriter out = new PrintWriter(socket.getOutputStream(),
true);
                out.println("Hello, you are client #" + clientNumber);
                // Get messages from the client, line by line; Each line has
several numbers
                // separated by a space
                while (true) {
                    String input = in.readLine();
                    if (input == null || input.isEmpty()) {
                        break;
                    }
                    // Put it in a string array
                    String[] nums = input.split(" ");
                    // Convert this string array to an int array
                    int[] intarr = new int[nums.length];
                    int i = 0;
                    for (String textValue : nums) {
                        intarr[i] = Integer.parseInt(textValue);
                        i++;
                    }
                    // Sort the numbers in this int array
                    new SelectionSort().sort(intarr);
                    // Convert the int array to String
                    String strArray[] = Arrays.stream(intarr).map-
ToObj(String::valueOf).toArray(String[]::new);
                    // Send the result to Client
                    String sendClient = new String();
                    System.out.println("Sen to Client: ");
                    for (int j = 0; j < strArray.length ; j++) {
                        // System.out.print(strArray[j] + " ");
                        sendClient += strArray[j] + " ";
                    }
                    System.out.println(sendClient);
```

```
                    out.println(sendClient);
                }
            } catch (IOException e) {
                System.out.println("Error handling client #" + clientNumber);
            } finally {
                try {
                    socket.close();
                } catch (IOException e) {
                }
                System.out.println("Connection with client # " + clientNumber
+ " closed");
            }
        }
    }

    public static void main(String[] args) throws IOException {
        // TODO Auto-generated method stub
        System.out.println("The Sorter Server is running!");
        int clientNumber = 0;
        try (ServerSocket listener = new ServerSocket(9898)) {
            while (true) {
                new Sorter(listener.accept(), clientNumber++).start();
            }
        }
    }

}
```

Câu hỏi 7: Vai trò của phương thức run(): lấy một chuỗi số từ client gửi lên cách nhau bằng khoảng trống, chuyển chuỗi thành mảng integer sau đó sắp xếp theo thứ tự tăng dần và gửi lại về cho client .

Nó được gọi khi: Có yêu cầu gửi lên từ phía client có số hiệu cổng trùng với số hiệu cổng của server.

3. Microservices – Docker an Kubernetes

```
mxn11@DESKTOP-CKFEBUT MINGW64 /c/Program Files/Docker Toolbox
$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://h
ub.docker.com to create one.
Username: rymmxn
Password:
WARNING! Your password will be stored unencrypted in C:\Users\mxn11\.docker\config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

Hình 6: Login vào docker

```
MINGW64:/c/Users/mxn11/microservices-demo

mxn11@DESKTOP-CKFEBUT MINGW64 ~ (master)
$ git clone https://github.com/anhth318/microservices-demo.git
fatal: destination path 'microservices-demo' already exists and is not an empty
directory.

mxn11@DESKTOP-CKFEBUT MINGW64 ~ (master)
$ https://github.com/anhth318/microservices-demo.git
bash: https://github.com/anhth318/microservices-demo.git: No such file or direct
ory

mxn11@DESKTOP-CKFEBUT MINGW64 ~ (master)
$ git clone https://github.com/anhth318/microservices-demo.git
fatal: destination path 'microservices-demo' already exists and is not an empty
directory.

mxn11@DESKTOP-CKFEBUT MINGW64 ~ (master)
$ mvnw.cmd clean package -Dmaven.test.skip=true
bash: mvnw.cmd: command not found

mxn11@DESKTOP-CKFEBUT MINGW64 ~ (master)
$ cd microservices-demo/

mxn11@DESKTOP-CKFEBUT MINGW64 ~/microservices-demo (master)
$ mvnw.cmd clean package -Dmaven.test.skip=true
bash: mvnw.cmd: command not found

mxn11@DESKTOP-CKFEBUT MINGW64 ~/microservices-demo (master)
$ ./mvnw clean package -Dmaven.test.skip=true
Warning: JAVA_HOME environment variable is not set.
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
```

Hình 7: Clone code về máy

```
mxn11@DESKTOP-CKFEBUT MINGW64 ~/microservices-demo (master)
$ docker tag microservice-kubernetes-demo-apache rymmxn/microservice-kubernetes-demo-apache:late
st

mxn11@DESKTOP-CKFEBUT MINGW64 ~/microservices-demo (master)
$ docker push ngoc/microservice-kubernetes-demo-apache
The push refers to repository [docker.io/ngoc/microservice-kubernetes-demo-apache]
An image does not exist locally with the tag: ngoc/microservice-kubernetes-demo-apache
```

Hình 8: push lên Docker Hub

Câu hỏi 1: thực hiện với 3 dịch vụ còn lại.
docker build --tag=microservice-kubernetes-demo-apache apache

docker tag microservice-kubernetes-demo-apache rymmxn/microservice-kubernetes-demo-apache:latest

docker push rymmxn/microservice-kubernetes-demo-apache

docker build --tag=microservice-kubernetes-demo-order microservice-kubernetes-demo-order

docker tag microservice-kubernetes-demo-order rymmxn/microservice-kubernetes-demo-order:latest

docker push rymmxn/microservice-kubernetes-demo-order

docker build --tag=microservice-kubernetes-demo-catalog microservice-kubernetes-demo-catalog

docker tag microservice-kubernetes-demo-catalog rymmxn/microservice-kubernetes-demo-catalog:latest

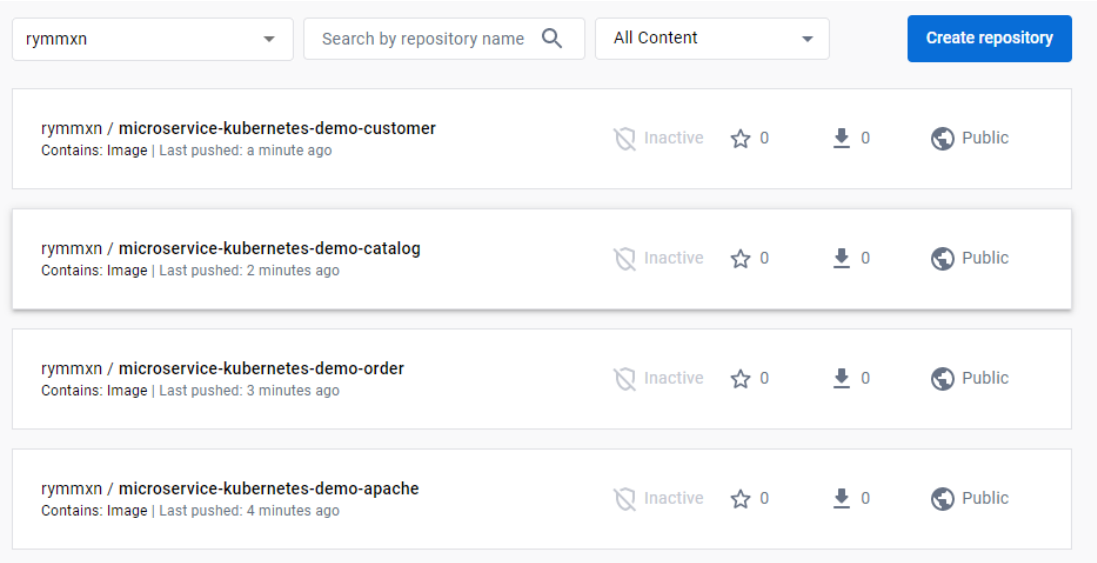
docker push rymmxn/microservice-kubernetes-demo-catalog

docker build --tag=microservice-kubernetes-demo-customer microservice-kubernetes-demo-customer

docker tag microservice-kubernetes-demo-customer rymmxn/microservice-kubernetes-demo-customer:latest

docker push rymmxn/microservice-kubernetes-demo-customer

Câu hỏi 2: Những gì hiện ra trên dockerhub



Kuberneted

Lưu ý: minikube cần 2 processor

Câu hỏi 3:

```
mxn11@DESKTOP-CKFEBUT MINGW64 ~/microservices-demo (master)
$ kubectl get all
NAME                              READY   STATUS              RESTARTS
 AGE
pod/apache-7764bdd97-fxw5d        0/1     ContainerCreating   0
 16s
pod/catalog-6b4d6c4fb4-jmqdh      0/1     ContainerCreating   0
 16s
pod/customer-7ffd9c5b7f-h7mrk     0/1     ContainerCreating   0
 16s
pod/order-6c44788969-ztb5l        0/1     ContainerCreating   0
 16s

NAME                 TYPE           CLUSTER-IP       EXTERNAL-IP   POR
T(S)          AGE
service/apache       LoadBalancer   10.108.72.138    <pending>     80:
30188/TCP     16s
service/catalog      LoadBalancer   10.97.176.86     <pending>     808
0:32707/TCP   16s
service/customer     LoadBalancer   10.110.208.185   <pending>     808
0:31746/TCP   16s
service/kubernetes   ClusterIP      10.96.0.1        <none>        443
/TCP          62s
service/order        LoadBalancer   10.97.193.113    <pending>     808
0:30513/TCP   16s

NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/apache       0/1     1            0           16s
deployment.apps/catalog      0/1     1            0           16s
deployment.apps/customer     0/1     1            0           16s
deployment.apps/order        0/1     1            0           16s

NAME                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/apache-7764bdd97        1         1         0       16s
replicaset.apps/catalog-6b4d6c4fb4      1         1         0       16s
replicaset.apps/customer-7ffd9c5b7f     1         1         0       16s
replicaset.apps/order-6c44788969        1         1         0       16s

mxn11@DESKTOP-CKFEBUT MINGW64 ~/microservices-demo (master)
$
```

Hình 9: Trạng thái của các pods vừa mới tạo

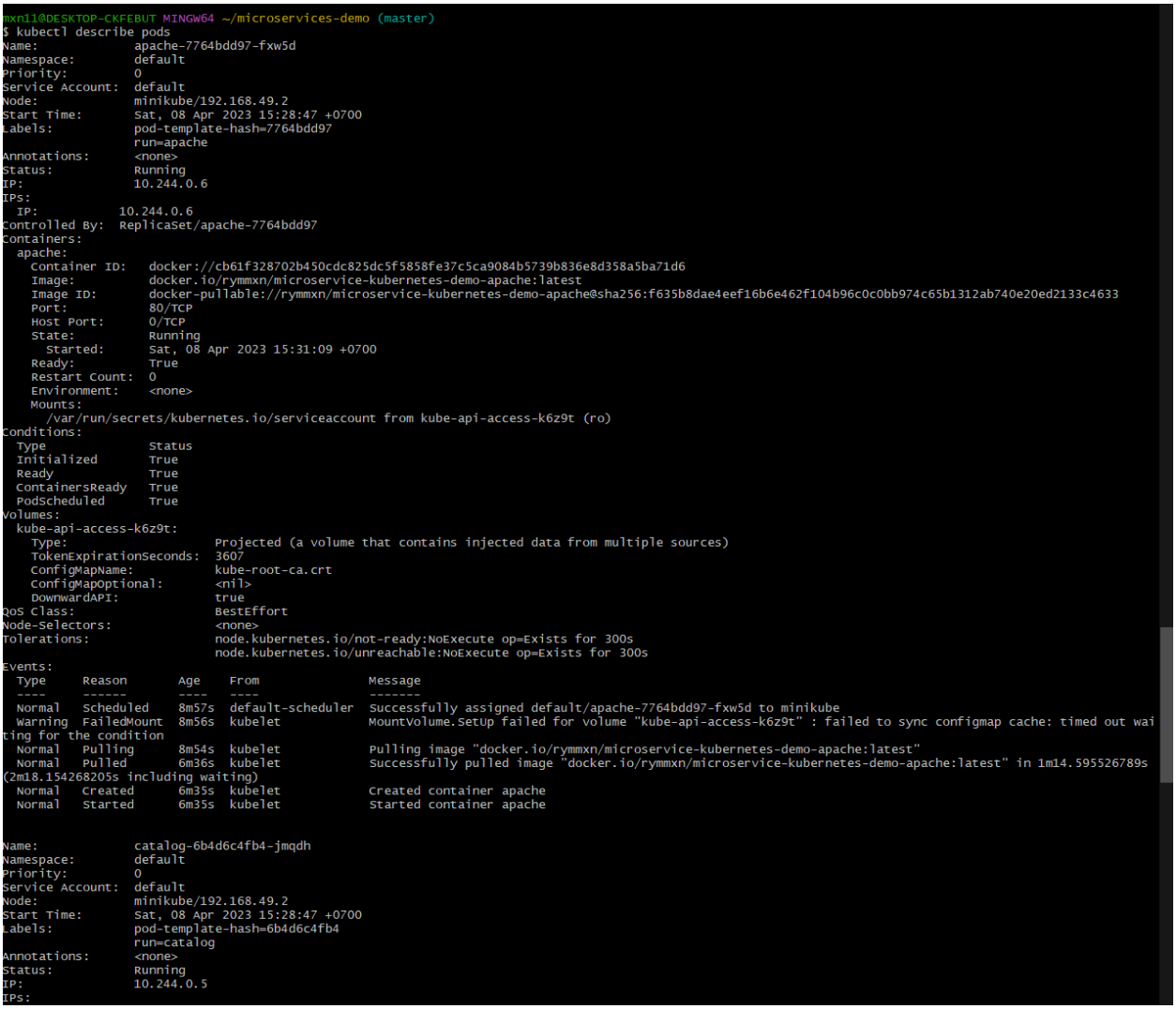
```
mxn11@DESKTOP-CKFEBUT MINGW64 ~/microservices-demo (master)
$ kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/apache-7764bdd97-fxw5d          1/1     Running   0          3m31s
pod/catalog-6b4d6c4fb4-jmqdh        1/1     Running   0          3m31s
pod/customer-7ffd9c5b7f-h7mrk       1/1     Running   0          3m31s
pod/order-6c44788969-ztb5l          1/1     Running   0          3m31s

NAME                 TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
service/apache       LoadBalancer   10.108.72.138    <pending>     80:30188/TCP     3m33s
service/catalog      LoadBalancer   10.97.176.86     <pending>     8080:32707/TCP   3m33s
service/customer     LoadBalancer   10.110.208.185   <pending>     8080:31746/TCP   3m33s
service/kubernetes   ClusterIP      10.96.0.1        <none>        443/TCP          4m19s
service/order        LoadBalancer   10.97.193.113    <pending>     8080:30513/TCP   3m33s

NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/apache       1/1     1            1           3m33s
deployment.apps/catalog      1/1     1            1           3m33s
deployment.apps/customer     1/1     1            1           3m33s
deployment.apps/order        1/1     1            1           3m33s

NAME                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/apache-7764bdd97        1         1         1       3m33s
replicaset.apps/catalog-6b4d6c4fb4      1         1         1       3m33s
replicaset.apps/customer-7ffd9c5b7f     1         1         1       3m33s
replicaset.apps/order-6c44788969        1         1         1       3m33s
```

Hình 10: Sau vài phút

- Hiển thị thông tin cổng giao dịch vụ

```
mxn11@DESKTOP-CKFEBUT MINGW64 ~/microservices-demo (master)
$ kubectl describe services
Name:                     apache
Namespace:                default
Labels:                   run=apache
Annotations:              <none>
Selector:                 run=apache
Type:                     LoadBalancer
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.108.72.138
IPs:                      10.108.72.138
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  30188/TCP
Endpoints:                10.244.0.6:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>


Name:                     catalog
Namespace:                default
Labels:                   run=catalog
Annotations:              <none>
Selector:                 run=catalog
Type:                     LoadBalancer
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.97.176.86
IPs:                      10.97.176.86
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  32707/TCP
Endpoints:                10.244.0.5:8080
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>


Name:                     customer
Namespace:                default
Labels:                   run=customer
Annotations:              <none>
Selector:                 run=customer
Type:                     LoadBalancer
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.110.208.185
IPs:                      10.110.208.185
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  31746/TCP
Endpoints:                10.244.0.4:8080
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>


Name:              kubernetes
Namespace:         default
Labels:            component=apiserver
                   provider=kubernetes
Annotations:       <none>
Selector:          <none>
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                10.96.0.1
IPs:               10.96.0.1
Port:              https  443/TCP
TargetPort:        8443/TCP
Endpoints:         192.168.49.2:8443
Session Affinity:  None
Events:            <none>


Name:                     order
Namespace:                default
Labels:                   run=order
Annotations:              <none>
Selector:                 run=order
Type:                     LoadBalancer
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.97.193.113
IPs:                      10.97.193.113
Port:                     <unset>  8080/TCP
```

Hình 11: kubectl describe services.



Hình 12: describe pods

```
mxn11@DESKTOP-CKFEBUT MINGW64 ~/microservices-demo (master)
$ kubectl delete service apache catalog customer orde
service "apache" deleted
service "catalog" deleted
service "customer" deleted
Error from server (NotFound): services "orde" not found

mxn11@DESKTOP-CKFEBUT MINGW64 ~/microservices-demo (master)
$ kubectl delete deployments apache catalog customer order
deployment.apps "apache" deleted
deployment.apps "catalog" deleted
deployment.apps "customer" deleted
deployment.apps "order" deleted

mxn11@DESKTOP-CKFEBUT MINGW64 ~/microservices-demo (master)
$ >minikube stop
bash: stop: command not found

mxn11@DESKTOP-CKFEBUT MINGW64 ~/microservices-demo (master)
$ minikube stop
* Stopping node "minikube"  ...
* Powering off "minikube" via SSH ...
* 1 node stopped.
```

Hình 13: Giải phóng

4. Kiến trúc JMS và DDS

Câu hỏi 1: Giải thích vai trò của application server glassfish. GlassFish cung cấp một môi trường chạy ứng dụng để triển khai và chạy các ứng dụng Java Enterprise Edition (Java EE).

- Vai trò của GlassFish trong việc triển khai các ứng dụng Java EE là cung cấp các tính năng như:
- Quản lý vòng đời ứng dụng: GlassFish quản lý vòng đời của các ứng dụng Java EE, bao gồm việc khởi động, phân phối tài nguyên, giám sát hoạt động và dừng ứng dụng.
- Cung cấp các dịch vụ hạ tầng: GlassFish cung cấp các dịch vụ hạ tầng như giao thức HTTP, JDBC, JMS, JNDI, JavaMail, và các dịch vụ web khác để hỗ trợ các ứng dụng Java EE.
- Quản lý tài nguyên: GlassFish quản lý các tài nguyên được sử dụng bởi các ứng dụng Java EE như các kết nối cơ sở dữ liệu, các tài nguyên tập tin và các tài nguyên liên quan đến web.
- Quản lý bảo mật: GlassFish hỗ trợ các tính năng bảo mật của Java EE như xác thực, ủy quyền, mã hóa và xác thực SSL.
- Quản lý phân phối: GlassFish hỗ trợ các tính năng phân phối và mở rộng của Java EE như cân bằng tải, phân phối phiên và phân phối tài nguyên.

- Các tính năng khác: GlassFish cung cấp nhiều tính năng khác như quản lý log, giám sát hoạt động và quản lý gói. Câu hỏi 2: Tại sao lại phải tạo 2 JNDI như trên? Việc tạo 2 JNDI trên là để định nghĩa các thành phần chính trong một hệ thống JMS, bao gồm Connection Factory và Destination.
- Connection Factory: Đây là thành phần cần thiết để tạo ra các kết nối tới message broker, giúp ứng dụng có thể gửi và nhận message.
- Destination: Là đích đến của các message được gửi và nhận. Nó có thể là một topic hoặc một queue. Việc tạo 2 JNDI như trên sẽ giúp cho ứng dụng có thể truy cập vào Connection Factory và Destination một cách dễ dàng. Connection Factory và Destination này có thể được sử dụng để tạo ra các kết nối và gửi/nhận message tới message broker thông qua các API của JMS. Cơ chế truyền và nhận thông điệp trong ví dụ sử dụng JMS được thực hiện bằng cách sử dụng kiến trúc hướng sự kiện. Trong đó:

Câu hỏi 3: Sau khi chạy thử chương trình Sender và Receiver, vận dụng lý thuyết kiến trúc hướng sự kiện đã học trên lớp để giải thích cơ chế chuyền và nhận thông điệp của Sender và Receiver. Cơ chế truyền và nhận thông điệp trong ví dụ sử dụng JMS được thực hiện bằng cách sử dụng kiến trúc hướng sự kiện.

Trong đó:

- Trong lớp MySender, sử dụng publish method của TopicPublisher để gửi message đến topic tìm được thông qua JNDI lookup tạo một đối tượng TextMessage và gán nội dung thông điệp vào đối tượng này bằng phương thức setText(). Sau đó, đối tượng này được chuyển đến đối tượng TopicPublisher, đại diện cho nguồn phát của thông điệp.
- Trong đối tượng TopicPublisher, phương thức publish() được gọi để gửi thông điệp đến đối tượng Topic, đại diện cho kênh truyền thông điệp.
- Bên phía đối tượng MyReceiver, đối tượng TopicSubscriber được tạo ra để nhận thông điệp từ đối tượng Topic. Đối tượng TopicSubscriber sử dụng phương thức setMessageListener() để đăng ký đối tượng MyListener làm bộ lắng nghe sự kiện nhận thông điệp.
- Khi một thông điệp được đẩy vào đối tượng Topic, đối tượng Topic sẽ phát sinh sự kiện và thông báo cho tất cả các đối tượng TopicSubscriber đã đăng ký với nó. Đối tượng TopicSubscriber sử dụng đối tượng MyListener để xử lý sự kiện này, và phương thức onMessage() của đối

tượng MyListener được gọi để xử lý thông điệp nhận được. Khi một thông điệp được đẩy vào đối tượng Topic, đối tượng Topic phát sinh sự kiện để thông báo cho tất cả các đối tượng TopicSubscriber đã đăng ký với nó. Đối tượng TopicSubscriber sử dụng đối tượng MyListener để xử lý sự kiện này và xử lý thông điệp được nhận.

Câu hỏi 4: So sánh JMS và DDS.

	DDS	JMS
Mục đích sử dụng	Trường được sử dụng để truyền tin nhắn giữa các ứng dụng trong một hệ thống phân tán	Được sử dụng để phân phối dữ liệu giữa các nút trong một mạng máy tính phân tán
Kiến trúc	Là một kiến trúc phần mềm giúp cho các ứng dụng có thể truyền tin nhắn theo mô hình tương tác giữa client và sever	Là một kiến trúc phần mềm phân tán, nó hoạt động dựa trên mô hình publish – subscribe
Tính đồng bộ	Hỗ trợ tính đồng bộ, nghĩa là nhận được tin nhắn sẽ chờ đợi cho đến khi tin nhắn đó đươc xử lý hoàn tất	Không đồng bộ, nghĩa là dữ liệu được phát đi và nhận ở các nút khác nhau trong thời gian thực
Tốc độ truyền dữ liệu	Thường có tốc độ truyền dữ liệu nhanh hơn so với JMS. Do DDS sừ dụng một số kỹ thuật tối ưu hóa như đa luồng, tối ưu hóa bộ nhớ và sử dụng gao thức UDP	Có tốc độ truyền chậm hơn so với DDS
Độ tin cậy	Có khả năng đảm bảo đọ tin cậy bởi vì cung cấp tính năng phục hồi	Có linh hoạt hơn trong việc xử lý các tính huống đặc biệt