# BÀI TẬP TRÊN LỚP MÔN HỌC: HỆ PHÂN TÁN CHƯƠNG 1: TỔNG QUAN VÀ KIẾN TRÚC HỆ PHÂN TÁN

# BÁO CÁO THỰC HÀNH

# Sinh Viên : Vũ Văn Trọng Mã số sinh viên : 20184206

# 1. Web server apache2

## 1.1 Cài đặt web server apache2

**Câu 1:** Đường dẫn đến file html chứa nội dung mặc định của trang web Apache2 là:

/var/www/html/index.html

**Câu 2:** Cổng mặc định của dịch vụ www là 80

## 1.2 Cài đặt virtual hosts cho apache2

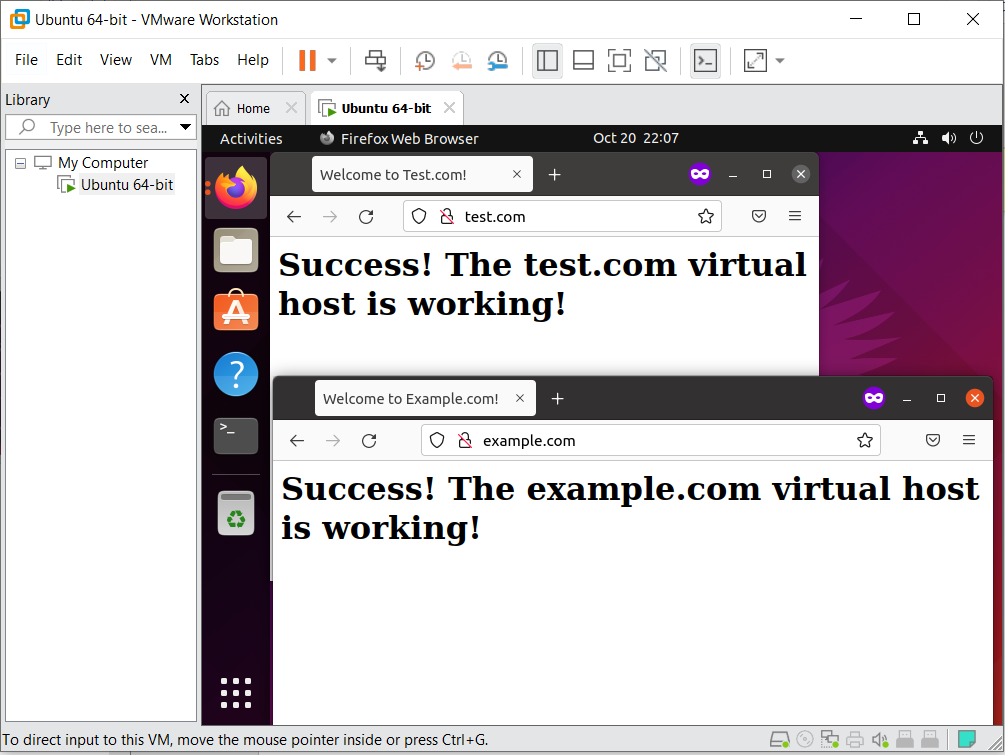
**Câu 3:** Với thự mục và file:

Chuỗi 755 cho biết thông tin phần quyền:

+ Owner: Đọc, Ghi, Thực thi ( 7 = 111(2) )

+ Group, Others: Đọc, Thực thi ( 5 = 101(2) )

**Câu 4:** Bạn quan sát thấy nội dung gì sau khi gõ 2 địa chỉ trên?



Do ta đã cho phép cung cấp địa chỉ test.com và example.com cho host nên khi ta nhập 2 tên miền ,host sẽ dẫn ra ta đến localhost nơi ta đã cài đặt trong file /etc/hosts

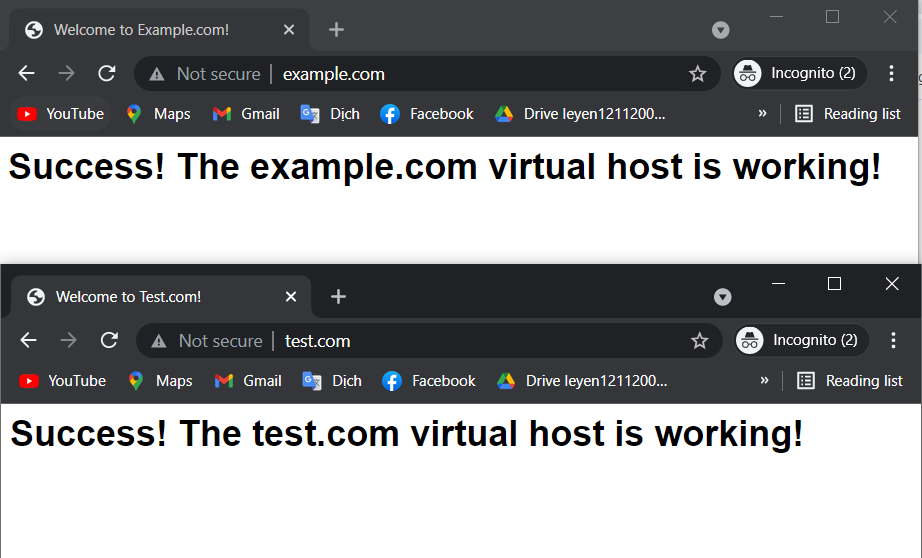
**Câu hỏi 5:** Thử truy cập từ các máy tính khác trong cùng mạng LAN vào 2 trang web đó.

Không truy cập được từ 1 máy tính trong mạng LAN

Graphical user interface, application

Description automatically generated

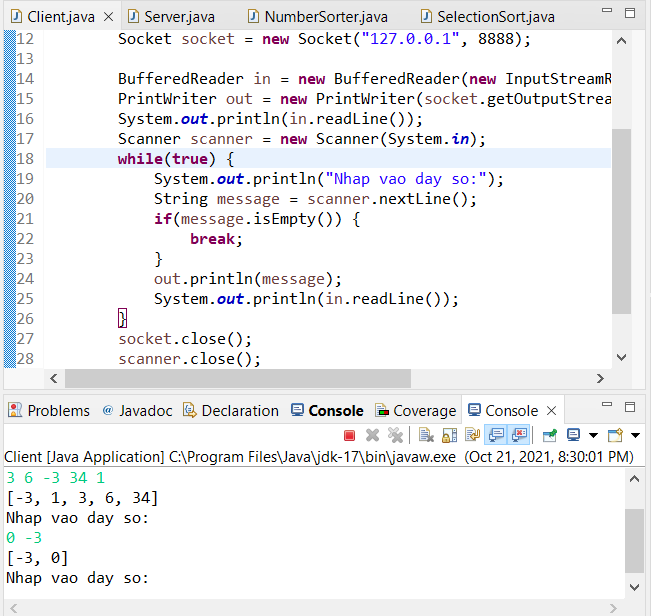
Sửa file hosts với địa chỉ máy ảo và tên miền thì mới có thể truy cập vào được:

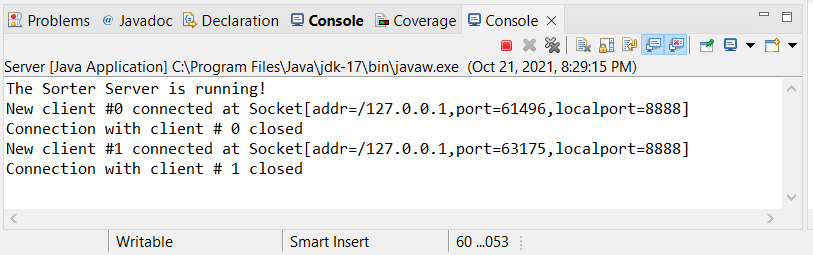


# 2. Interface trong Java

## 2.1 Xây dựng chương trình

**Câu 6:** Hãy tự viết một đoạn code để thực hiện 1 vòng lặp while sao cho nó sẽ nhận các số mà người dùng gõ và gửi về server, cho đến khi nào người dùng gõ ký tự rỗng rồi ấn enter.





**Câu 7:** Vai trò của phương thức run là gì? Khi nào thì nó được gọi?

Vai trò của phương thức run là nhận dữ liệu từ clients, xử lý rồi gửi lại kết quả cho clients. Nó sẽ được gọi khi có 1 message từ client gửi đến server

# 3. Kiến trúc Microservices

**Câu 1:** Hãy thực hiện gõ những lệnh tương tự như trên với 3 dịch vụ còn lại.

**Câu 2:** Vào trang web DockerHub và đăng nhập vào tài khoản của bạn. Bạn thấy những gì mới xuất hiện trên docker hub repository của bạn?

Trên docker hub repository có 4 repository được push:

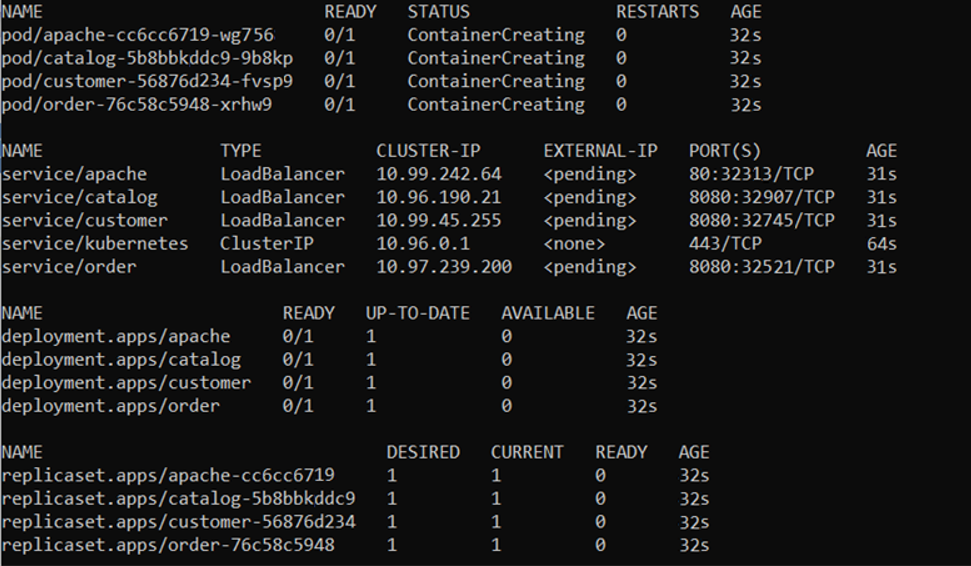
microservice-kubernetes-demo-customer

microservice-kubernetes-demo-order

microservice-kubernetes-demo-catalog

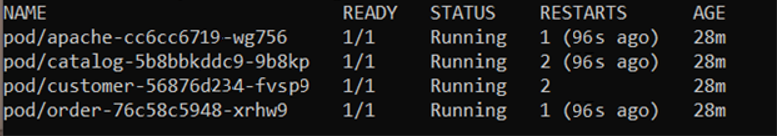
microservice-kubernetes-demo-apache

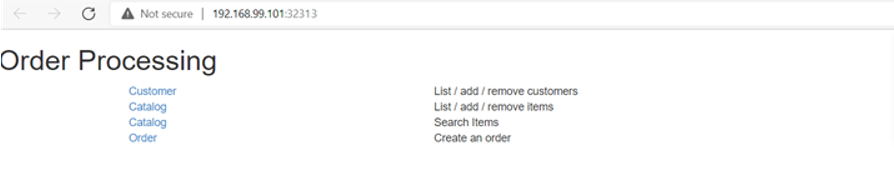
**Câu 3**: Trạng thái (status) của các pods vừa mới tạo được là gì? Bây giờ, hãy chờ vài phút và gõ lại lệnh đó, trạng thái mới của các pods giờ đã chuyển thành gì?



Trạng thái của các pods vừa mới được tạo là ContainerCreating

- Sau vài phút gõ lại lệnh, trạng thái các pods chuyển thành Running





# 4. Kiến trúc JMS và DDS

**Câu 1:** Giải thích vai trò của application server glassfish.

|  |
| --- |
| - Glassfish server là một Webserver, cho phép triển khai các ứng dụng web viết bằng Java lên đó. |
|  |

- Glassfish server cho phép quản lý, cấu hình nhanh và linh hoạt.

**Câu 3**: Sau khi chạy thử chương trình Sender và Receiver, vận dụng lý thuyết kiến trúc hướng sự kiện đã học trên lớp giải thích cơ chế chuyền và nhận thông điệp của Sender và Receiver.

|  |  |
| --- | --- |
| Sender | Receiver |
| 1. Create and start connection 2. create queue session 3. get the Topic object 4. create TopicPublisher object 5. create TextMessage object 6. write message 7. send message 8. connection close | 1. Create and start connection 2. Create topic session 3. Get the Topic object 4. Create TopicSubscriber 5. Create listener object 6. Register the listener object with subscriber |

**Câu 4:** So sánh JMS và DDS.

|  |  |
| --- | --- |
| JMS | DDS |
| * Java Messaging Service * Java Mesage Oriented Middleware API * Used for sending message between two or more clients * Centrailized  |  | | --- | | * Allows the communication between different components of a distributed application * Most commenly used * Many implementations: ActiveMQ, GlassFish, RabbitMQ | |  | | * Data distributed service * Decentralized  |  |  |  |  |  | | --- | --- | --- | --- | --- | | * DDS is networking middleware that simplifies complex network programming  |  |  |  | | --- | --- | --- | | * It implements a publish/subscribe model for sending events and commands among the nodes * Automatically handles all aspects of message delivery  |  | | --- | | * Automatically handles hot-swapping redundant publishers if the primary fails | |  | | |  | | |  | |

|  |
| --- |
|  |