1. **Web server apache2**

**Tạo file docker-compose.yaml**

services:

  apache-server:

    image: ubuntu/apache2

    container\_name: apache-server

    ports:

      - "80:80"

    volumes:

      - ./example.com/public\_html:/var/www/example.com/public\_html

      - ./test.com/public\_html:/var/www/test.com/public\_html

      - ./example.com/example.com.conf:/etc/apache2/sites-available/example.com.conf  # Mount file cấu hình example.com

      - ./test.com/test.com.conf:/etc/apache2/sites-available/test.com.conf

      - ./logs:/var/log/apache2

    command: >

      /bin/bash -c "a2ensite example.com.conf &&

                   a2ensite test.com.conf &&

                   service apache2 restart &&

                   tail -f /var/log/apache2/error.log"

**Run docker-compose up -d**

*- Câu hỏi 1: Đường dẫn đến file html chữa nội dung mặc định của trang web các bạn vừa xem là gì?*

/var/www/html/index.html

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

*- Câu hỏi 2: Cổng mặc định của dịch vụ www là gì?*

Cổng 80

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

Nội dung do AI tạo ra có thể không chính xác.

*- Câu hỏi 3: Hãy giải thích quyền mang số 755 là gì?*

Cho phép chủ sở hữu truy cập, sửa file theo cách họ muốn. Mọi người khác chỉ có thể truy cập mà không có quyền sửa nó. Tất cả mọi người đều có quyền thực thi.

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

- “Welcome to Example.com!”, “Success! The example.com virtual host is working!” trong phần nội dung.

Ảnh có chứa ảnh chụp màn hình, phần mềm, máy tính, Phần mềm đa phương tiện

Nội dung do AI tạo ra có thể không chính xác.

- “Welcome to Test.com!”, “Success! The test.com virtual host is working!” trong phần nội dung.

Ảnh có chứa ảnh chụp màn hình, phần mềm, văn bản, máy tính

Nội dung do AI tạo ra có thể không chính xác.

- Giải thích: Trang web đã được thực thi và chủ sở hữu có thể nhìn thấy giao diện web.

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

- Phải cấu hình file hosts và proxy mới có thể truy cập được.

- Kết quả tại máy chính (Windows) bật kết nối Host-only hơi máy ảo (Ubuntu) (cùng 1 LAN).

Ảnh có chứa ảnh chụp màn hình, phần mềm, máy tính, Phần mềm đa phương tiện

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa ảnh chụp màn hình, phần mềm, văn bản, máy tính

Nội dung do AI tạo ra có thể không chính xác.

**2. Interface trong Java**

*- Câu hỏi 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.*

Gợi ý: hãy dùng lệnh sau để nhận xâu ký tự người dùng gõ vào: String message = scanner.nextLine();

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Nội dung do AI tạo ra có thể không chính xác.

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

Phương thức “run” thực thi luồng tương ứng nếu phương thức “start” được gọi. Khi có một kết nối từ Client đến Server, một thread mới tại server được tạo ra, phương thức “start” được gọi để bắt đầu thực thi phương thức “run”.

**3. Kiến trúc Microservices**

Build/rebuild 3 dịch vụ:

**Ảnh có chứa văn bản, phần mềm, Phần mềm đa phương tiện, máy tính

Nội dung do AI tạo ra có thể không chính xác.**

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

*docker build --tag=microservice-kubernetes-demo-apache apache*

Ảnh có chứa văn bản, ảnh chụp màn hình

Nội dung do AI tạo ra có thể không chính xác.

docker tag microservice-kubernetes-demo- apache thangit67/microservice-kubernetes-demo- apache:latest

docker push thangit67/microservice-kubernetes-demo-apache Ảnh có chứa văn bản, ảnh chụp màn hình, thực đơn, thiết kế

Nội dung do AI tạo ra có thể không chính xác.

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

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

docker tag microservice-kubernetes-demo-order thangit67/microservice-kubernetes-demo-order:latest

docker push thangit67/microservice-kubernetes-demo-order

Ảnh có chứa văn bản, phần mềm, Phần mềm đa phương tiện, ảnh chụp màn hình

Nội dung do AI tạo ra có thể không chính xác.

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

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Nội dung do AI tạo ra có thể không chính xác.

docker tag microservice-kubernetes-demo-catalog thangit67/microservice-kubernetes-demo-catalog:latest

docker push thangit67/microservice-kubernetes-demo-catalog

Ảnh có chứa văn bản, phần mềm, Phần mềm đa phương tiện, ảnh chụp màn hình

Nội dung do AI tạo ra có thể không chính xác.

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

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

docker tag microservice-kubernetes-demo-customer thangit67/microservice-kubernetes-demo-customer:latest

docker push thangit67/microservice-kubernetes-demo-customer

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

*Câu hỏi 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?*

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Nội dung do AI tạo ra có thể không chính xác.

Các image mới

thangit67/microservice-kubernetes-demo-customer

thangit67/microservice-kubernetes-demo-catalog

thangit67/microservice-kubernetes-demo-order

[thangit67/microservice-kubernetes-demo-apache](https://hub.docker.com/repository/docker/thangit67/microservice-kubernetes-demo-apache)

kubectl apply -f microservices.yaml

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

kubectl get all

Ảnh có chứa văn bản, ảnh chụp màn hình

Nội dung do AI tạo ra có thể không chính xác.

*Câu hỏi 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?*

Ban đầu các pods mới tạo ở trạng thái “ContainerCreating”, sau đố chuyển sang trạng thái “Running”.

**4. Kiến trúc JMS và DDS**

**Tạo fill docker-compose/yaml**

services:

  payara:

    image: payara/server-full

    container\_name: payara-server

    ports:

      - "8080:8080"

      - "4848:4848"

    environment:

      - PAYARA\_DOMAIN=domain1

    restart: always

**run docker-compose up -d**

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

Do hiện tại image của glassfish đã hỏng sử dụng một server khác dựa trên glassfish để thay thế tên payara.

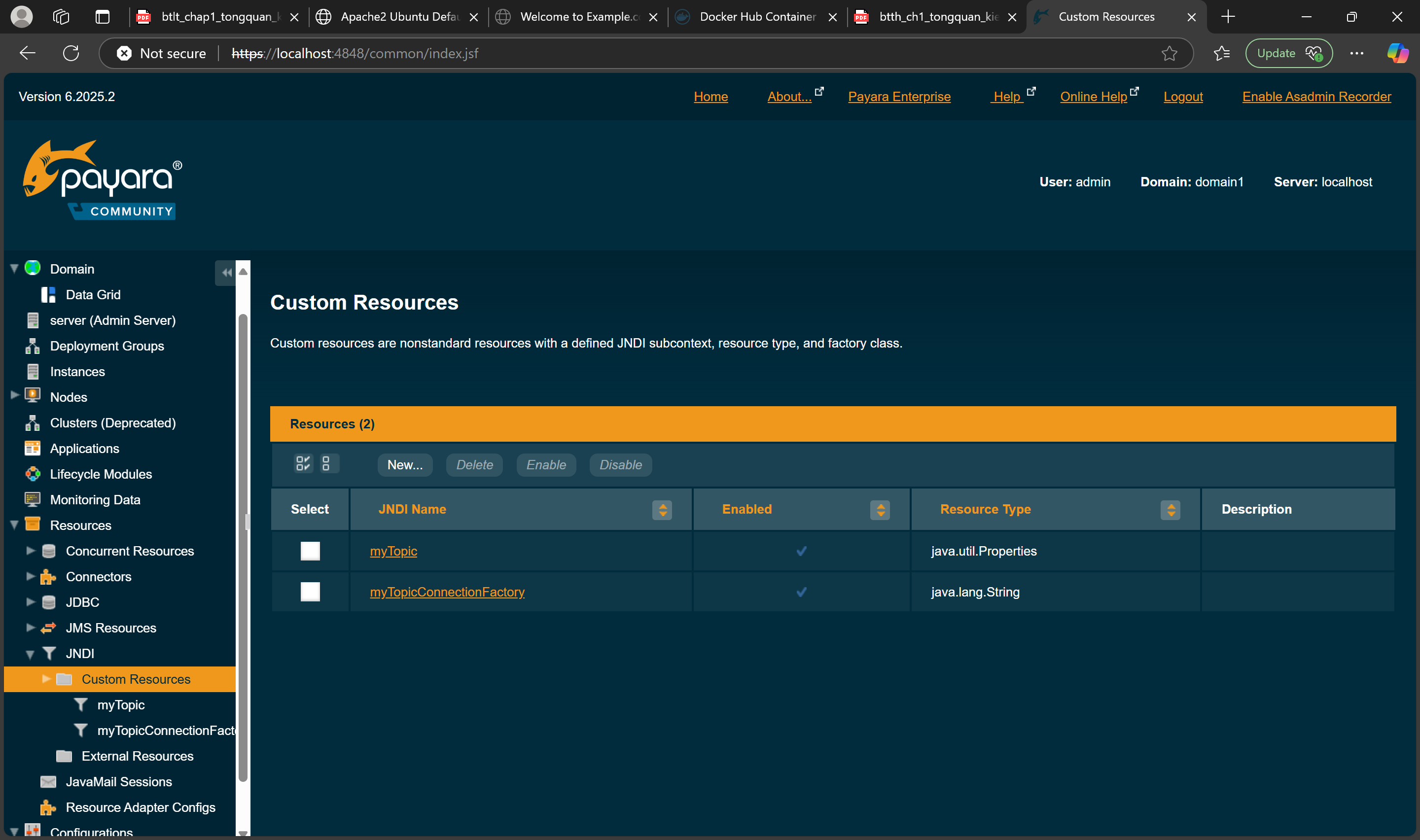
Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Nội dung do AI tạo ra có thể không chính xác.

**Payara Server** là một máy chủ ứng dụng mã nguồn mở, dựa trên **GlassFish**, được thiết kế để chạy các ứng dụng **Java EE/Jakarta EE**. Vai trò chính của nó bao gồm:

1. **Triển khai ứng dụng doanh nghiệp**: Hỗ trợ các ứng dụng web, EJB, REST, SOAP.
2. **Hỗ trợ microservices và cloud**: Tích hợp với Docker, Kubernetes, và các nền tảng đám mây.
3. **Quản lý và giám sát**: Cung cấp giao diện quản trị (Admin Console) và công cụ dòng lệnh (asadmin).
4. **Bảo mật và hiệu suất**: Cung cấp tính năng bảo mật nâng cao và hỗ trợ clustering.

*Câu hỏi 2: Tại sao lại phải tạo 2 JNDI như trên?*



- Phải tạo 2 JNDI vì

myTopicConnectionFactory để payara tạo một connector connection pool và connector connector resource.

myTopic để payara tạo connector admin object resource.

*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 chuyền và nhận thông điệp của Sender và Receiver:

- Cơ chế chuyền (Sender):

Khi có một nguyên nhân, kích thích làm thay đổi một sự kiện nào đó, bên Sender sẽ gửi một thông điệp (message) cho bên nhận. Thông điệp ấy được nằm trong bộ nhớ tạm, nếu có tín hiệu chuyển phát (commit) mới thì thông điệp mới được chuyển sang cho người nhận.

- Cơ chế nhận (Receiver):

Khi nhận được một thông điệp được gửi đến. Nếu chưa kịp xử lý thì bên nhận đưa tạm vào bộ nhớ đệm, nếu bộ nhớ đầy thì báo lỗi và sử dụng chơ chế roolback gửi lại cho người gửi. Trong quá trình xử lý gặp lỗi thì phía bên nhận cũng phải báo lại cho bên gửi

*Câu hỏi 4: So sánh JMS và DDS.*

Điểm giống:

|  |  |  |
| --- | --- | --- |
| Đặc điểm | JMS – Java Messaging | DDS – Data distributed service |
| Cấu trúc | Publish / Subscribe | Publish / Subscribe (đa hướng) |
| Nền tàng độc lập | Cùng một API được hiển thị cho tất cả HW, OS và các ngôn ngữ được hỗ trợ | Cùng một API được hiển thị cho tất cả HW, OS và các ngôn ngữ được hỗ trợ |

Điểm khác:

|  |  |  |
| --- | --- | --- |
| Đặc điểm | JMS – Java Messaging | DDS – Data distributed service |
| Khám phá các điểm cuối | Máy chủ JNDI và JMS phải được chỉ định và cấu hình | Khám phá động, không cần chỉ định nơi các điểm cuối cư trú |
| Loại an toàn | Đối tượng Tổng quát (Generic) và XML không phải là loại an toàn | An toàn kiểu cách mạnh mẽ, các cuộc gọi ứng dụng write () và read () với một kiểu dữ liệu cụ thể |
| Điều chỉnh hành vi giao tiếp | Khả năng điều chỉnh thông tin liên lạc hạn chế | Các chính sách QoS cho phép dễ dàng điều chỉnh các hành vi ứng xử giao tiếp |
| Khả năng tương tác | Không có | Tiêu chuẩn mở với khả năng tương tác đã được chứng minh |