TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Vân Duy Quang
Lê Thái Như Quỳnh
Lê Trần Thiện Thắng
Trần Minh Toàn
Võ Duy Trường
Nguyễn Lê Công Quý

HƯỚNG DẪN TRIỂN KHAI RELEASE

HỆ THỐNG QUẢN LÝ KHÁCH HÀNG CHO DOANH NGHIỆP NHỎ

THỰC TẬP DỰ ÁN TỐT NGHIỆP
CHƯƠNG TRÌNH CHÍNH QUY

Thành phố Hồ Chí Minh, tháng 01/2024

Mục lục

Danh sách hình

Tóm tắt

Tài liệu này được ra đời với mục đích hướng dẫn người đọc cách triển khai các kết quả biên dịch và release của đề tài Thực tập dự án tốt nghiệp "Hệ thống quản lý khách hàng cho doanh nghiệp nhỏ" lên các máy chủ, hoàn thiện quá trình xuất bản ứng dụng đến người dùng.

Đề tài được thực hiện theo kiến trúc microservice, do đó tất cả service hoàn toàn có thể được triển khai độc lập. Sau khi kết thúc quá trình release như được hướng dẫn ở file `SOURCE/ReleasenGuide.pdf` , ta cần tiến hành đưa những sản phẩm sau đây lên một hoặc một số server khác nhau. Cụ thể, những sản phẩm cần được triển khai bao gồm:

1. Server `KeyCloak` (Docker Container): Dùng cho việc lưu trữ dữ liệu xác thực, thông tin cá nhân, quyền hạn của người dùng, thông tin doanh nghiệp.

2. Service `authenticaion` (Java Spring Boot): Dùng để tương tác với máy chủ KeyCloak.

3. Service `type-service` (Java Spring Boot): Giúp quản lý, khởi tạo, xây dựng và cấu hình cấu trúc các đối tượng dữ liệu cho công ty.

4. Service `record-service` (Java Spring Boot): Lưu trữ thông tin chi tiết các bản ghi của những đối tượng dữ liệu đã cấu hình.

5. Service `notification-service` (Java Spring Boot): Tiếp nhận và trao đổi thông báo giữa các service đến phía người dùng.

6. Service `service-registry` (Java Spring Boot): Cổng đăng kí các server khả dụng cho từng service.

7. Server `RabbitMQ` (Docker Container): Dùng cho việc giao tiếp bất đồng bộ giữa các service trong hệ thống Microservice.

8. Server `PostgreSQL` (Database): Các database PostgreSQL riêng cho các service `notification`, `type-ervice`, `record-service`.

9. **Storage Service** để lưu các tệp tin, hình ảnh của ứng dụng.

10. Service `api-gateway` (Java Spring Boot): Cổng giao tiếp giữa client và các service khả dụng.

11. Ứng dụng `Frontend` (Vite, React, TypeScript, TailwindCSS): Ứng dụng giao tiếp với người dùng cuối.

Tất cả nội dung và sản phẩm thực hiện đồ án kể trên đều được đi kèm trong file `SourceCode.zip`. Đối với những database, hệ thống sẽ sử dụng dịch vụ **AWS RDS** (Free Tier). Đối với front-end, hệ thống sẽ sử dụng **Vercel**. Đối với tất cả những service còn lại, hệ thống sẽ sử dụng những máy chủ **AWS EC2** (Free Tier) hoặc **GCP Computer Engine** để triển khai những Docker image đã khởi tạo. Mọi hướng dẫn được thực hiện trên hệ điều hành CentOS 7.0 – hệ điều hành phổ biến cho các Server chạy nhân Linux.

Chương 1

Tạo tài khoản Cloud Service

1.1. Tạo tài khoản và đăng nhập AWS/GCP

1.1.1. AWS – Amazon Web Services

1. Truy cập vào liên kết
   `https://portal.aws.amazon.com/billing/signup?`
   để đến trang đăng kí tài khoản của AWS. Thực hiện theo hướng dẫn của trang web để tạo tài khoản.

2. Đăng nhập vào tài khoản AWS đã tạo tại
   `https://console.aws.amazon.com/console/home`.

1.1.2. GCP – Google Cloud Platform

1. Truy cập vào liên kết `https://cloud.google.com/` để đến trang chính thức của GCP.

2. Chọn `Get started for free` để tạo tài khoản mới.

3. Điền thông tin cần thiết và thực hiện theo hướng dẫn của trang web để tạo tài khoản.

4. Đăng nhập vào tài khoản GCP đã tạo tại `https://console.cloud.google.com/`.

1.2. Cài đặt CLI cho AWS/GCP

Việc cài đặt CLI giúp người dùng có thể tương tác với các dịch vụ của AWS/GCP thông qua dòng lệnh. Đồng thời, giúp việc trao đổi, quản lý thư mục và tệp tin dễ dàng hơn.

1.2.1. AWS

1. Truy cập vào liên kết này để cài đặt AWS CLI.

2. Chạy lệnh sau để kiểm tra việc cài đặt CLI đã thành công hay chưa:

3. `aws -version`

4. Đăng nhập vào tài khoản AWS bằng CLI bằng cách chạy lệnh sau:

5. `aws configure`

6. Nhập thông tin cần thiết như `Access Key ID`, `Secret Access Key`, `Default region name`, `Default output format`.

7. Kiểm tra việc đăng nhập thành công bằng cách chạy lệnh sau:

8. `aws s3 ls`

9. Nếu không có lỗi nào xuất hiện, việc đăng nhập đã thành công.

1.2.2. GCP

1. Truy cập vào liên kết `https://cloud.google.com/sdk/docs/install` để cài đặt Google Cloud SDK.
2. Chạy lệnh sau để kiểm tra việc cài đặt CLI đã thành công hay chưa:
3. `gcloud -version`
4. Đăng nhập vào tài khoản GCP bằng CLI bằng cách chạy lệnh sau:
5. `gcloud auth login`
6. Nhập thông tin cần thiết để đăng nhập vào tài khoản GCP.
7. Kiểm tra việc đăng nhập thành công bằng cách chạy lệnh sau:
8. `gcloud projects list`
9. Nếu không có lỗi nào xuất hiện, việc đăng nhập đã thành công.

1.3. Tạo máy chủ AWS EC2 hoặc GPC Computer Engine

1.3.1. AWS EC2

1. Truy cập vào `https://console.aws.amazon.com/ec2/` để tạo máy chủ EC2.
2. Chọn `Launch instance` để bắt đầu quá trình tạo máy chủ.

3. Chọn loại máy chủ, cấu hình máy chủ, lưu trữ, bảo mật, và tạo key pair để truy cập máy chủ từ CLI. (Khuyến khích sử dụng hệ điều hành CentOS)

4. Chọn `Review and Launch` để xem lại thông tin và chọn `Launch` để tạo máy chủ.

5. Chọn key pair đã tạo để mở máy chủ và truy cập vào máy chủ.

6. Đăng nhập vào máy chủ bằng cách chạy lệnh sau:

```
ssh ec2-user@ec2-public-ip -v -i "path/to/authentication/key.pem
    "
```

1.3.2. GCP Computer Engine

1. Truy cập vào `https://console.cloud.google.com/compute/instances` để tạo máy chủ GCP.

2. Chọn `Create instance` để bắt đầu quá trình tạo máy chủ.

3. Điền thông tin cần thiết như tên máy chủ, loại máy chủ, cấu hình máy chủ, lưu trữ, bảo mật. (Khuyến khích sử dụng hệ điều hành CentOS)

4. Chọn `Create` để tạo máy chủ.

5. Chọn `SSH` để mở máy chủ và truy cập vào máy chủ.

1.3.3. Cấu hình cơ bản cho máy chủ CentOS

1. Cập nhật hệ thống bằng cách chạy lệnh sau:
2. `sudo yum update`
3. Theo dõi hướng dẫn tại đây để cài đặt Docker cho hệ điều hành CentOS.
4. Đăng nhập vào Docker Hub (hoặc AWS ECR) bằng cách chạy lệnh: `docker login`

Chú ý: Có thể sẽ phải thay đổi cấu hình bảo mật của máy chủ để cho phép truy cập từ xa. (VPC Network Inbounce và Outbounce Rules đối với AWS, Firewall Rules đối với GCP)

1.4. RDS – Relational Database Service

1.4.1. AWS RDS

1. Truy cập vào `https://console.aws.amazon.com/rds/` để tạo RDS.
2. Chọn `Create database` để bắt đầu quá trình tạo RDS.
3. Chọn loại database cần sử dụng, cấu hình database, bảo mật, và tạo database. (Khuyến khích sử dụng PostgreSQL)
4. Cài đặt network để cho phép database được kết nối đến từ xa bởi nhiều máy chủ khác nhau.

Chương 2

Triển khai KeyCloak

Phần này sẽ hướng dẫn cách triển khai một server KeyCloak trên máy chủ CentOS. Đây là một trong những công đoạn phức tạp nhất của việc triển khai.

2.1. Tạo keystore certificate

Để KeyCloak có thể được chạy trong môi trường production. KeyCloak cần được kết nối và giao tiếp thông qua phương thức HTTPS. Do đó, ta cần tạo một keystore certificate trỏ từ một tên miền đến địa chỉ IP của máy chủ. Để tạo keystore certificate, ta cần thực hiện các bước sau:

1. Cài đặt `certbot` bằng cách chạy lệnh sau:

```
sudo yum install certbot
```

2. Chạy lệnh sau để tạo keystore certificate:

```
sudo certbot certonly --standalone -d example.com
```

3. Keystore certificate sẽ được lưu tại `/etc/letsencrypt/live/example.com/`

4. Chuyển keystore certificate về thư mục `/etc/ssl/certs/` bằng cách chạy lệnh sau:

5.
```
 sudo cp /etc/letsencrypt/live/example.com/fullchain.pem /etc
/ssl/certs/keycloak.crt
 sudo cp /etc/letsencrypt/live/example.com/privkey.pem /etc/
ssl/certs/keycloak.key
 sudo chown keycloak:keycloak /etc/ssl/certs/keycloak.crt
 sudo chown keycloak:keycloak /etc/ssl/certs/keycloak.key
 sudo chmod 600 /etc/ssl/certs/keycloak.crt
 sudo chmod 600 /etc/ssl/certs/keycloak.key

```

6. Tạo keystore từ certificate bằng cách chạy lệnh sau:

7.
```
 sudo keytool -importkeystore -srckeystore /etc/ssl/certs/
keycloak.jks -destkeystore /etc/ssl/certs/keycloak.p12 -
deststoretype PKCS12

```

8. Chuyển keystore về thư mục `/etc/keycloak/` bằng cách chạy lệnh sau:

9.
```
 sudo cp /etc/ssl/certs/keycloak.p12 /etc/keycloak/
 sudo chown keycloak:keycloak /etc/keycloak/keycloak.p12
 sudo chmod 600 /etc/keycloak/keycloak.p12

```

2.2. Chạy Container Keycloak

Sau khi đã tạo keystore certificate, ta sẽ chạy container KeyCloak trên máy chủ.Việc chạy KeyCloak đòi hỏi một số biến môi trường nhất định Để chạy container KeyCloak, ta cần thực hiện các bước sau:

1. Đăng nhập vào Docker Hub hoặc AWS ECR dựa vào hướng dẫn tương ứng trong `ReleaseGuide.pdf`.

2. Tải image của KeyCloak từ Docker Hub hoặc AWS ECR bằng cách chạy lệnh sau:

3.
```
    docker pull repository-of-keycloak:latest

```

4. Build container KeyCloak với những environment variable và volume cần thiết bằng cách chạy lệnh sau:

5.
```
    sudo docker run -d -p 8443:8443 -p 8080:8083 -e
KEYCLOAK_ADMIN=admin-account -e KEYCLOAK_ADMIN_PASSWORD=admin
-password \
    \item -e KC_DB=database-name -e KC_DB_URL=jdbc:postgresql://
database-host:port/database-name \
    -e KC_DB_USERNAME=database-account -e KC_DB_PASSWORD=
database-password \
    -e KC_HOSTNAME=host -e KC_HOSTNAME_ADMIN_URL=https://host \
    -v /etc/keycloak/:/opt/keycloak/conf \
    repository-of-keycloak start --hostname-debug=true --https-
key-store-file=/conf/keycloak.p12 --https-key-store-password=
password-cert

```

Chú ý: Có thể sẽ cần phải cài đặt thêm ngnix cho máy chủ để truy cập thông qua giao thức HTTPS đến máy chủ KeyCloak.

2.3. Kiểm tra kết quả

Sau khi đã cài đặt thành công, vào trình duyệt và truy cập vào địa chỉ `https://tên-miền:8443` để truy cập vào giao diện quản trị của KeyCloak.

Chương 3

Triển khai các service back-end

Phần này sẽ hướng dẫn cách triển khai tất cả những service phía back-end còn lại như:

1. Service `authentication`
2. Service `api-gateway`
3. Service `type-service`
4. Service `record-service`
5. Service `notification-service`
6. Service `service-registry`
7. Service `rabbitmq`

3.1. Tải image từ AWS ECR hoặc Docker Hub

3.1.1. Đối với RabbitMQ

Ta sử dụng cài đặt mặc định của RabbitMQ, có thể được tìm thấy tại `salesync/back-end/message-queue/docker-compose.yml`.
Hoặc đơn giản hơn, có thể pull RabbitMQ mặc định tại `rabbitmq:3-management`, cụ thể bằng lệnh:

```
docker pull rabbitmq:3-management
```

3.1.2. Đối với các service còn lại

Ta cần tải image từ AWS ECR hoặc Docker Hub, thực hiện các bước tương tự với KeyCloak như sau:

1. Đăng nhập vào AWS ECR hoặc Docker Hub dựa vào hướng dẫn tương ứng trong `ReleaseGuide.pdf`.

2. Chạy lệnh `sudo docker pull tên-repository:latest`

3.2. Triển khai service `authentication`

3.2.1. Chuẩn bị keystore validation

Đây là một file keystore validation được sử dụng để xác thực các request từ service `api-gateway` đến service `authentication`, giúp cho việc giao tiếp từ `api-gateway` đến các service khác được bảo mật. Keystore có thể được tạo bởi nhiều cách khác nhau, theo thuật toán RSA256. Kết quả của việc tạo keystore là một file nhị phân có đuôi `.jks`. Kèm theo đó là một

public key có thể được sử dụng tại các service khác để validate.
Xem thêm về hướng dẫn tạo một file keystore bằng `keytool` tại đây.

3.2.2. Chạy container service `authentication`

Đối với tất cả các service, ta cần cung cấp các biến môi trường, các thư mục và tệp tin cần thiết để chạy service. Để chạy service `authentication`, Các biến môi trường sau cần được bổ sung:

1. `KEYSTORE_PATH`: `/app/keys/auth.jks` – Vị trí lưu file keystore vừa tạo trong môi trường Docker.
2. `KEYSTORE_PASSWORD`: Password đã đặt lúc tạo file keystore.
3. `KEYSTORE_ALIAS`: Keystore alias đã đặt lúc tạo keystore.
4. `KEYCLOAK_HOST`: Tên miền của KeyCloak server đã tạo ở chương trước.
5. `KEYCLOAK_ADMIN_USER`: Tài khoản admin KeyCloak đã đặt ở chương trước.
6. `KEYCLOAK_ADMIN_PASSWORD`: Mật khẩu admin KeyCloak đã đặt.
7. `MAIL_SERVER_USER`: Tên tài khoản xác thực của mail server (dùng để gửi email xác thực, khuyến khích sử dụng `AWS SES`).
8. `MAIL_SERVER_PASSWORD`: Mật khẩu tài khoản xác thực của mail server.
9. `DB_HOST`: Tên host của Database Instance đã sử dụng để tạo KeyCloak database.

10. `DB_PORT`: Port của Database Instance đã sử dụng để tạo KeyCloak database.

11. `DB_NAME`: Tên của database tạo riêng cho service authentication (nên là database khác với KeyCloak)

12. `DB_USER`: Tên tài khoản truy cập database

13. `DB_PASSWORD`: Mật khẩu truy cập database của tài khoản trên.

Ngoài ra, ta cần phải chuẩn bị file keystore đã tạo ở một vị trí nào đó trên máy chủ, và tạo volume cho container để truy cập file này.

Kết hợp, ta sẽ có một lệnh chạy container như sau:

```
1  sudo docker run -d -e KEYSTORE_PATH=/app/keys/auth.jks -e
   KEYSTORE_PASSWORD=example  -e ...
2  -v /path/of/keystore/folder:/app/keys -p 8082:8082 --name the-
   repository-image-name:latest
```

3.3. Chạy container `RabbitMQ`

Để chạy container `RabbitMQ`, ta cần thực hiện lệnh đơn giản sau:

```
1  docker run -p 5672:5672 -p 15672:15672 -e RABBITMQ_DEFAULT_USER=
   user -e RABBITMQ_DEFAULT_PASS=password rabbitmq:3-management
```

Trong đó, 5672*lportmcØnhcaRabbitMQ*,15672 là port để truy cập vào giao diện quản trị của RabbitMQ, *user*vpassword là tài khoản và mật khẩu để truy cập vào RabbitMQ.

3.4. Chạy service `service-registry`

Chạy Eureka bằng lệnh

```
docker run -p 8761:8761 repository-service-registry-name:latest
```

3.5. Chạy service `type-service`

Tương tự, ta cần các biến môi trường sau:

1. `DB_HOST`: Tên host của Database Instance sẽ được sử dụng cho `type-service`.
2. `DB_PORT`: Port của Database Instance.
3. `DB_USER`: Tên tài khoản truy cập database
4. `DB_PASSWORD`: Mật khẩu truy cập database của tài khoản trên.
5. `RABBITMQ_HOST`: Tên host của RabbitMQ server.
6. `RABBITMQ_PORT`: Port của RabbitMQ server.
7. `RABBITMQ_USER`: Tên tài khoản truy cập server
8. `RABBITMQ_PASSWORD`: Mật khẩu truy cập của tài khoản trên.
9. `EUREKA_SERVER_URL`: Địa chỉ của server Eureka.

3.6. Chạy service `record-service`

Tương tự, ta cần các biến môi trường sau:

1. `DB_HOST`: Tên host của Database Instance sẽ được sử dụng cho `record-service`.

2. `DB_PORT`: Port của Database Instance.

3. `DB_USER`: Tên tài khoản truy cập database

4. `DB_PASSWORD`: Mật khẩu truy cập database của tài khoản trên.

5. `RABBITMQ_HOST`: Tên host của RabbitMQ server.

6. `RABBITMQ_PORT`: Port của RabbitMQ server.

7. `RABBITMQ_USER`: Tên tài khoản truy cập server

8. `RABBITMQ_PASSWORD`: Mật khẩu truy cập của tài khoản trên.

9. `EUREKA_SERVER_URL`: Địa chỉ của server Eureka.

3.7. Chạy service `notification-service`

Tương tự, ta cần các biến môi trường sau:

1. `DB_HOST`: Tên host của Database Instance sẽ được sử dụng cho `notification-service`.

2. `DB_PORT`: Port của Database Instance.

3. `DB_USER`: Tên tài khoản truy cập database

4. `DB_PASSWORD`: Mật khẩu truy cập database của tài khoản trên.

5. `RABBITMQ_HOST`: Tên host của RabbitMQ server.

6. `RABBITMQ_PORT`: Port của RabbitMQ server.

7. `RABBITMQ_USER`: Tên tài khoản truy cập server

8. `RABBITMQ_PASSWORD`: Mật khẩu truy cập của tài khoản trên.

3.8. Chạy service `api-gateway`

Tương tự, ta cần các biến môi trường sau:

1. `AUTH_SERVER_URL`: Tên miền của server `authentication`.

2. `EUREKA_SERVER_URL`: Địa chỉ của server Eureka.

3.9. Kiểm tra kết quả

Sau khi đã cài đặt thành công, ta đã có được địa chỉ IP, tên miền và các thông tin port cần thiết để truy cập được từ gateway và front-end.

Chương 4

Triển khai AWS S3 và API liên quan đến storage

Trước khi triển khai front-end, ta cần xây dựng service `AWS S3` (hoặc tương tự) để lưu trữ các file ảnh của trang web và của người dùng.
Sau đó, ta xây dựng một AWS API Gateway để thực hiện lệnh upload ảnh đến `AWS S3`.

4.1. Cây tài nguyên AWS S3

Xây dựng và tổ chức thư mục tài nguyên của `AWS S3` như sau:

1. salesync
    (a) `avatars`: Chứa toàn bộ avatar (sau khi đã processed) của người dùng.
    (b) `copmanies`: Chứa toàn bộ tài nguyên hình ảnh của tất cả doanh

nghiệp.

(c) `preproccessed_avatars`: Nơi lưu trữ những avatars do người dùng upload, chưa được processed.

(d) `system`: Những hình ẩnh mà ứng dụng cần sử dụng

Chú ý: Tài nguyên này nên được đặt ở chế độ **Public**.

4.2. Tạo hàm Lambda để process avatar

Mỗi khi người dùng upload avatar, ta cần kích hoạt hàm lambda để xử lý ảnh sang nhiều kích thước khác nhau, sau đó lưu vào thư mục `avatars`. Chi tiết về cách thiết lập quyền cho hàm lambda có thể được xem qua ở đây.

4.3. Xây dựng API Gateway cho Storage

Dựa vào hướng dẫn tại đây. Ta xây dựng một API Gateway resource có 2 API dành riêng cho việc upload avatars vào thư mục `preproccessed_avatars` cũng như vào thư mục `companies`.

4.4. Kết quả

Sau khi đã cài đặt thành công, ta có tổng cộng 3 api, 1 để đọc tài nguyên `AWS S3`, 2 để ghi vào các thư mục con.

Chương 5

Deploy Frontend

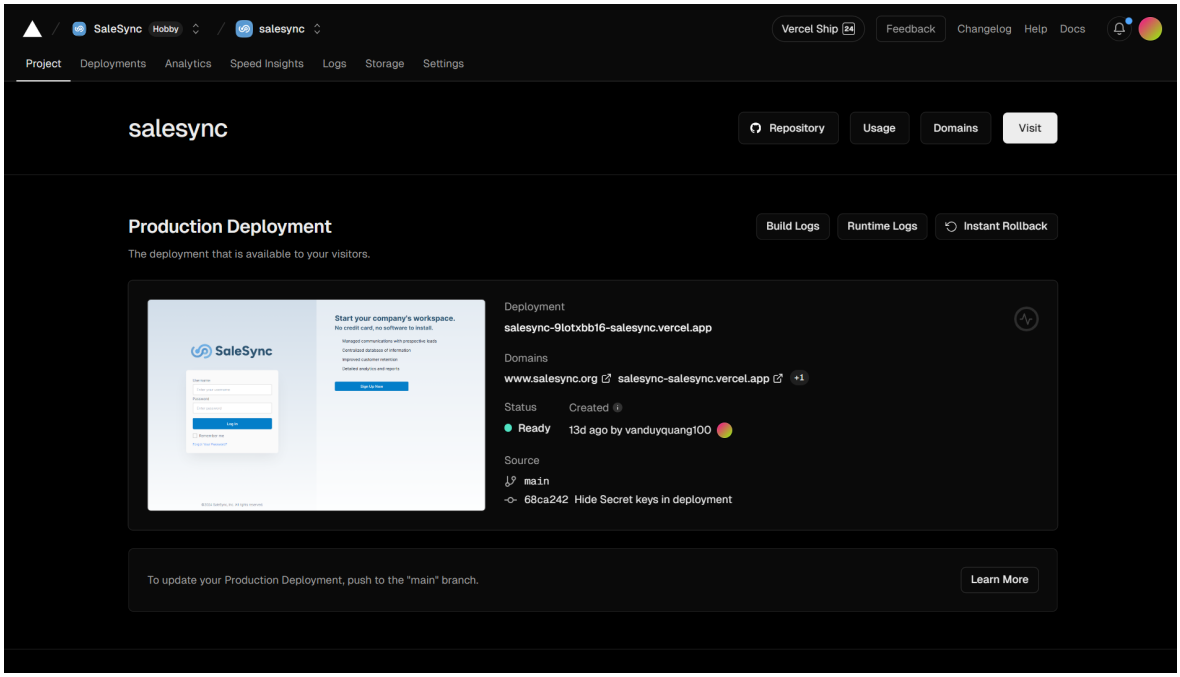
Sau khi tất cả service đã được chuẩn bị, ta tiến hành deploy `front-end` bằng **Vercel**.

5.1. Deploy `front-end`

Truy cập và tạo tài khoản **Vercel**. Sau đó, tạo một project mới và deploy từ GitHub repository của `front-end` đã được chuẩn bị. Ta cũng có thể cấu hình thêm tên miền và domain riêng để deploy **Vercel**.

Tại cài đặt của project, truy cập vào mục Environment và cấu hình các biến môi trường sau:

1. `VITE_GATEWAY_HOST`: Địa chỉ của tên miền API Gateway.
2. `VITE_STORAGE_SERVICE_HOST`: Địa chỉ truy cập xem storage của `AWS S3` đã cấu hình ở chương trước.
3. `VITE_UPLOAD_API_ENDPOINT`: Địa chỉ API Gateway dùng để upload avatar người dùng lên `AWS S3` đã cấu hình ở chương trước.



Hình 5.2.0.1: Màn hình chính của Vercel

4. `VITE_UPLOAD_COMPANY_API_ENDPOINT`: Địa chỉ API Gateway dùng để upload tài nguyên doanh nghiệp lên `AWS S3` đã cấu hình ở chương trước.

5.2. Kết quả

Sau khi đã cài đặt thành công, ta có thể truy cập vào trang web của mình thông qua domain đã được cấu hình.

Ta cũng có thể nhìn thấy kết quả deploy tại trang quản lý của Vercel (Xem Hình 5.2.0.1.

Chương 6

Kết quả Triển khai

Sau khi kết thúc quá trình triển khai máy chủ. Ta đã có thể truy cập và thao tác được với toàn bộ hệ thống thông qua các service đã được triển khai. Dưới đây là danh sách các service đã được triển khai.

Tham khảo trang web mẫu của đề tài tại
`https://www.salesync.org`.

Tài liệu `UserGuide.pdf` đi kèm sẽ tiếp tục hướng dẫn cách sử dụng ứng dụng đã deploy này.