**AWS Cloud Developer**

Udacity Nanodegree

**Thach Quoc Cuong - CuongTQ17**

Giới thiệu tổng quan và một số thông tin cần chú ý khi học và tham gia làm project

# **Overview**

Khóa học bao gồm 4 nội dung chính và 5 project tương ứng:

* Cloud Fundamentals
  + Project 1: Deploy Static Website
* Full Stack App on AWS
  + Project 2: Udagram: Your own Instagram on AWS
* Monolith to Microservices at Scale
  + Project 3: Refactor Monolith to Microservices
* Develop & Develop Serverless App
  + Project 4: Serverless Application
* Capstone
  + Project 5: Capstone

Với những project trên bạn sẽ được tìm hiểu và tham gia:

* Triển khai ứng dụng trên đám mây: Hướng dẫn triển khai ứng dụng web trên các nền tảng đám mây.
* Xử lý dữ liệu: Học cách lưu trữ và quản lý dữ liệu trong môi trường đám mây.
* Điều khiển phiên bản và quản lý mã nguồn: Sử dụng các công cụ như Git để quản lý mã nguồn và triển khai các ứng dụng.
* Microservices và Kubernetes: Hiểu về kiến trúc microservices và cách triển khai chúng bằng Kubernetes.
* Xác thực và bảo mật: Tìm hiểu về cơ chế xác thực và bảo mật trong các ứng dụng đám mây.
* Serverless computing: Học cách xây dựng và triển khai các ứng dụng không cần quản lý máy chủ (serverless) trên đám mây.
* CI/CD: Áp dụng quy trình liên tục tích hợp và liên tục triển khai (CI/CD) để cải thiện quá trình phát triển và triển khai ứng dụng.

# **Prerequisite**

Khóa học sẽ cần một số kiến thức về các mảng sau:

* Căn bản về cloud, cài đặt aws cli



* Lập trình cơ bản (javascript)
* IDE: Visual studio code
* Git,Github

Tất cả các project sẽ làm lab thực tế trên AWS Cloud, môi trường do Udacity cung cấp.

# **Attention**

Để có thể sử dụng các dịch vụ AWS chúng ta có thể sử dụng tài khoản lab do Udacity cung cấp để Launch AWS Web Console.

**Lưu ý**: Khi sử dụng tài khoản lab AWS Web Console do Udacity cung cấp thì chỉ có giới hạn chi phí thực hành trên các service của AWS trong mỗi project là **25$**, nếu sử dụng hết thì phải sử dụng tài khoản AWS cá nhân, để thực hành tiếp những phần còn lại và những tài nguyên trước đó đã build thì sẽ không truy cập được nữa. Chỉ cho phép truy cập vào các region **us-east-1, us-east-2, us-west-1, us-west-2**. Nên hãy cố gắng tận dụng và hoàn thành nhanh nhất. Sau khi sử dụng xong, luôn luôn deleted hoặc stopped hết tài nguyên.

Khi chuẩn bị submit project để review, hãy dành thời gian đọc **Project Rubric** và xem xem mình đã làm đủ các yêu cầu trong Rubric chưa. Trừ phần **Standout Suggestion** thì các phần còn lại trong Rubric phải đáp ứng hết thì mới có thể **PASS** được.

Trường hợp sử dụng hết budget mà Udacity cung cấp cho mỗi project, học viên cần:

* Sử dụng budget ở các project còn lại
* Có thể sử dụng tài khoản được cấp của học viên khác còn credit
* Tạo ticket để Udacity support cung cấp thêm budget: <https://udacityenterprise.zendesk.com/hc/en-us/requests/new>

# **Project 01: Deploy Static Website**

Trong project này bạn sẽ Cấu hình S3 Bucket cho việc lưu trữ trang web tĩnh và Sử dụng CloudFront để phân phối nội dung: Bạn sẽ tìm hiểu về dịch vụ CloudFront của AWS, một dịch vụ phân phối nội dung (CDN) giúp tăng tốc độ tải trang web bằng cách sử dụng các máy chủ gần người dùng và bộ nhớ cache

**Một số service dùng trong project:**

CloudFront, S3

* **Chi tiết project:**

**Step 1:** Truy cập vào s3 service và tạo một bucket, nhớ bỏ tick **Block all public access**

**Step 2:** Tải [folder](https://drive.google.com/file/d/15vQ7-utH7wBJzdAX3eDmO9ls35J5_sEQ/view) này về, giải nén và upload toàn bộ nội dung của folder vào bucket vừa mới tạo

**Step 3:** Trong phần permissions của bucket, sửa lại bucket policy theo bên dưới

{ "Version":"2012-10-17", "Statement":

[ {

"Sid":"AddPerm",

"Effect":"Allow",

"Principal": "\*",

"Action":["s3:GetObject"],

"Resource":["arn:aws:s3:::your-website/\*"

] }

] }

Trong đó:

* "Version": Phiên bản của policy. Trong trường hợp này, nó là phiên bản "2012-10-17".
* "Statement": Khai báo các quy tắc truy cập trong policy. Đây có thể là một mảng (array) chứa nhiều quy tắc truy cập.
* "Sid": Một định danh (ID) cho quy tắc, giúp bạn nhận ra quy tắc cụ thể nếu bạn có nhiều quy tắc trong policy.
* "Effect": Xác định hiệu lực của quy tắc. Trong trường hợp này, "Allow" cho phép truy cập.
* "Principal": Người dùng hoặc đối tượng được áp dụng quy tắc. Trong trường hợp này, "\*" cho tất cả mọi người có quyền truy cập.
* "Action": Xác định hành động được phép thực hiện. Trong trường hợp này, chỉ cho phép "s3:GetObject", tức là tải về các đối tượng từ bucket S3.
* "Resource": Xác định các tài nguyên cụ thể (ARNs) mà quy tắc áp dụng. Trong trường hợp này, "arn:aws:s3:::your-website/\*" áp dụng cho tất cả các đối tượng (objects) trong bucket S3 có tên là "your-website".

**Step 4**: Trong phần properties của bucket, bật Static website hosting

**Step 5**: Sử dụng Cloudfront cho website được host trên s3, lưu ý phần Origin domain sử dụng endpoint theo dạng <http://my-1265158-bucket.s3-website.us-east-2.amazonaws.com>

**Step 6**: Đợi website được deploy, truy cập vào website thông qua domain name (ví dụ: dgf7z6g067r6d.cloudfront.net) và chụp lại ảnh và hoàn thành project

# **Project 02: Udagram: Your Own Instagram on AWS**

Trong project này, bạn sẽ xây dựng một ứng dụng web tương tự Instagram, cho phép người dùng tải lên và chia sẻ hình ảnh. Ứng dụng của bạn sẽ chạy trên AWS (Amazon Web Services) và sử dụng các dịch vụ như Amazon S3 (Simple Storage Service) và Amazon RDS (Relational Database Service). Bạn sẽ sử dụng dịch vụ AWS Elastic Beanstalk để triển khai ứng dụng của mình.

* **Một số service dùng trong project:**

Elastic Beanstalk, S3, RDS.

**Chi tiết về project:**

**Step 1:** Clone starter repo về máy

**Step 2**: Nếu bạn đang dùng nodejs 16 đổ lên thì hãy đổi lại version của các pkg như hình. Nếu để mặc định như khi clone về sẽ bị lỗi type



**Step 3**: Tạo api get filtered image, sử dụng typescript nhiều nhất có thể

**Step 4:** Chạy npm run build để tạo file build để upload lên elastic beanstalk

**Step 5**: Chạy eb init

Lệnh "eb init" trong AWS Elastic Beanstalk (EB) được sử dụng để khởi tạo một ứng dụng mới hoặc kết nối với một ứng dụng đã tồn tại trên EB. Khi chạy lệnh này, bạn sẽ được yêu cầu cung cấp một số thông tin để cấu hình ứng dụng của mình trên EB.

Cụ thể, khi chạy "eb init", bạn sẽ được yêu cầu nhập các thông tin sau:

* Chọn một hệ điều hành mà bạn muốn sử dụng cho môi trường EB của mình (Linux, Windows, hoặc Docker).
* Chọn một ngôn ngữ lập trình mà bạn muốn sử dụng (Java, .NET, Node.js, PHP, Python, Ruby, hoặc Go).
* Chọn một nền tảng máy chủ (64-bit Amazon Linux, 64-bit Amazon Linux 2, 64-bit Windows Server 2019, hoặc Docker).
* Chọn một môi trường EB (như "default" hoặc tên môi trường tùy chỉnh).

Sau khi bạn cung cấp các thông tin này, EB sẽ tạo một tệp cấu hình (.elasticbeanstalk/config.yml) trong thư mục gốc của ứng dụng của bạn. Tệp cấu hình này chứa thông tin về môi trường EB của bạn, bao gồm cả thông tin về hệ điều hành, ngôn ngữ lập trình và nền tảng máy chủ.

**Step 6**: Mở file .eb/config.yml thêm dòng sau đây

deploy:

artifact: ./www/Archive.zip

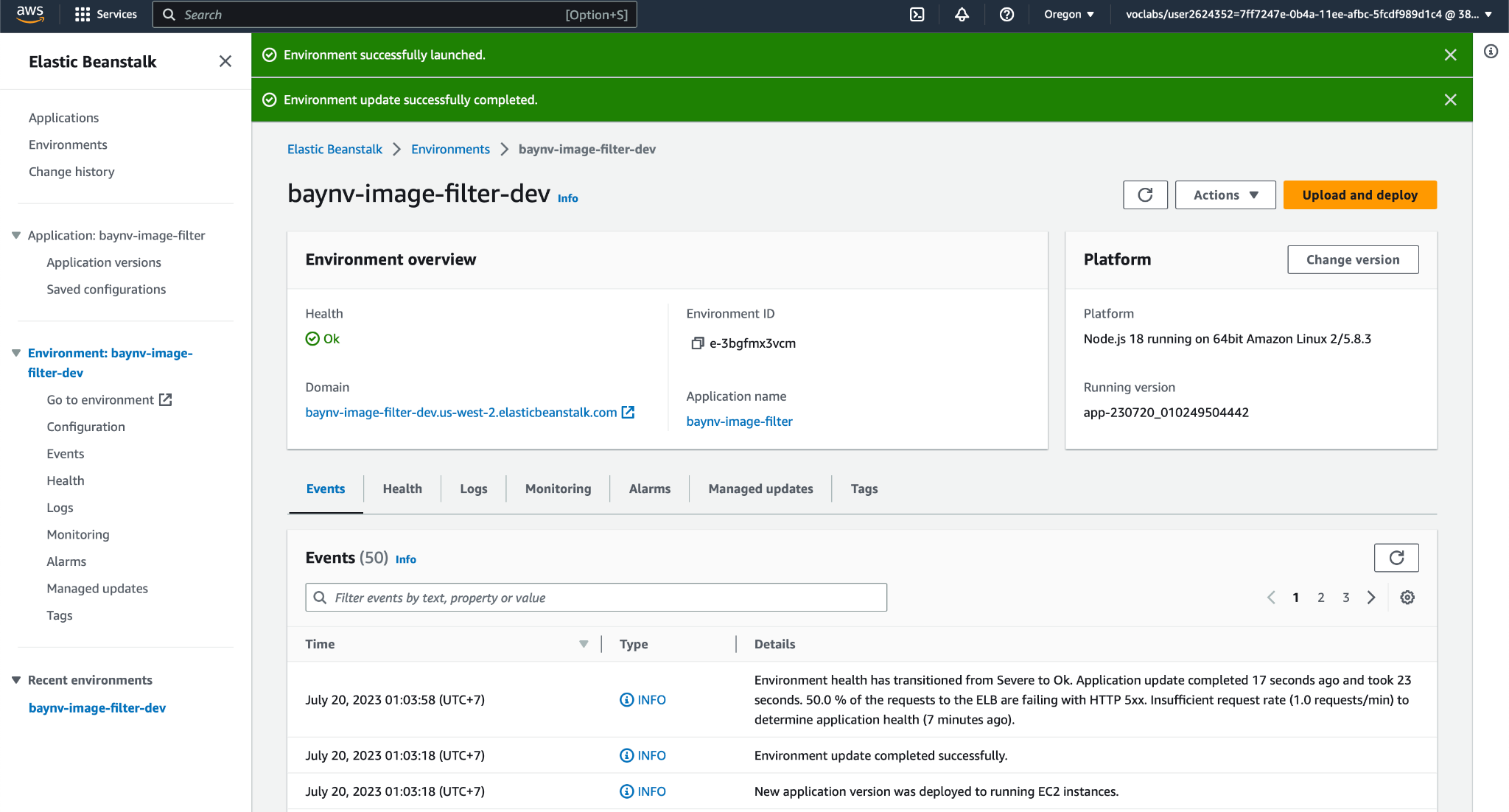
**Step 7**: Chạy eb create

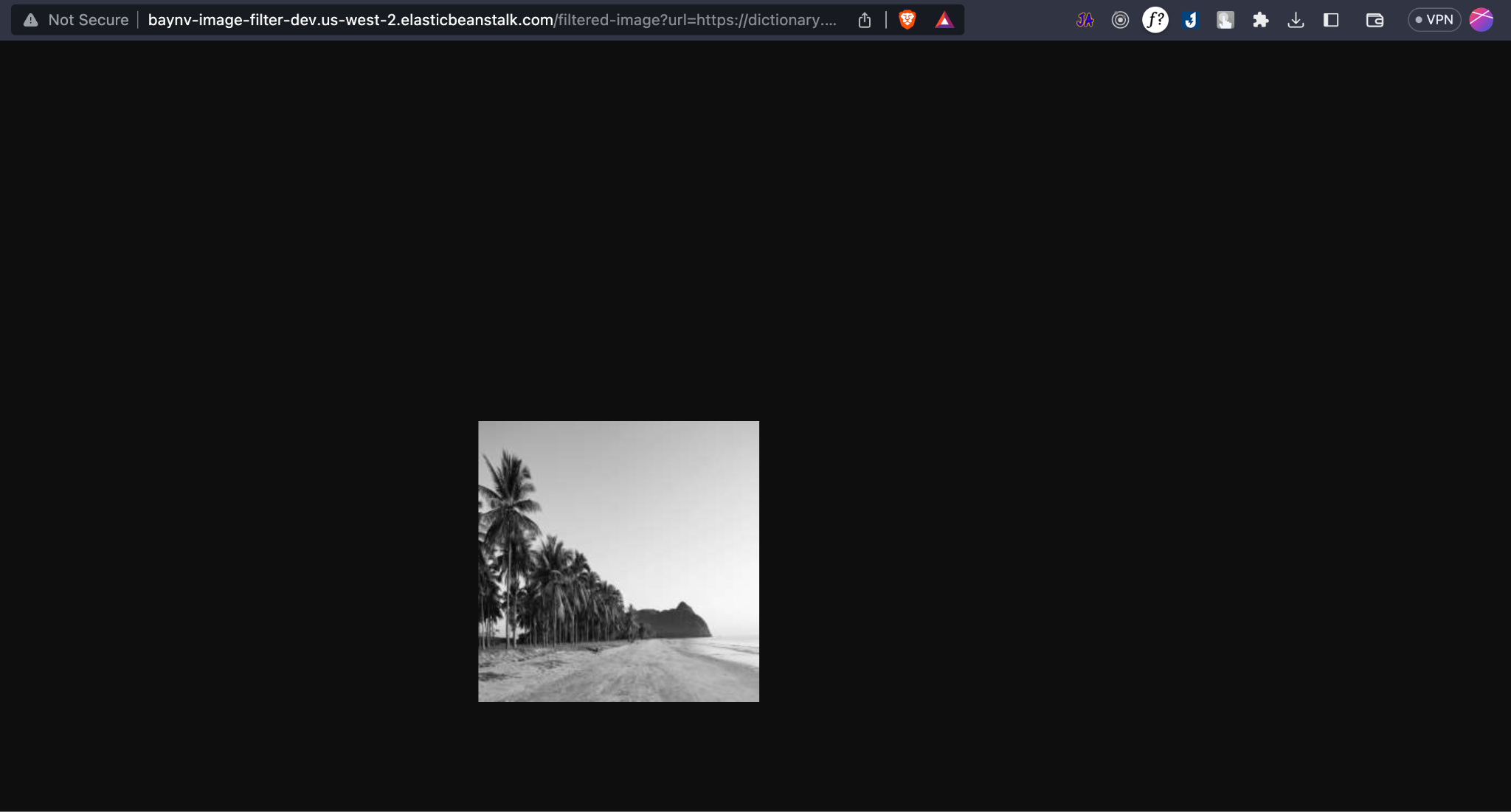
Khi chạy lệnh "eb create" trong AWS Elastic Beanstalk (EB), bạn đang tạo một môi trường mới để chạy ứng dụng của mình trên EB. Môi trường này sẽ bao gồm các tài nguyên cần thiết để chạy ứng dụng của bạn, bao gồm máy chủ, mạng, cơ sở dữ liệu và các tài nguyên khác.

Khi chạy lệnh "eb create", bạn sẽ được yêu cầu cung cấp một số thông tin để tạo môi trường EB, bao gồm:

* Tên môi trường: Đây là tên duy nhất để xác định môi trường EB của bạn.
* Cấu hình môi trường: Bạn có thể chọn một cấu hình môi trường có sẵn hoặc tạo một cấu hình tùy chỉnh.
* Cấu hình máy chủ: Bạn có thể chọn kích thước và loại máy chủ phù hợp với ứng dụng của bạn.
* Cấu hình mạng: Bạn có thể chọn cấu hình mạng như VPC, subnet và nhóm bảo mật.

Sau khi bạn cung cấp các thông tin này, EB sẽ tạo một môi trường mới dựa trên cấu hình và tài nguyên bạn đã chỉ định. EB sẽ tự động triển khai ứng dụng của bạn lên môi trường này và cung cấp cho bạn một URL để truy cập ứng dụng.

**Step 8**: Truy cập vào elastic beanstalk để kiểm tra deployment. Truy cập vào domain được cấp (ví dụ <http://xxxdev.us-west-2.elasticbeanstalk.com>) và thử dán một link ảnh bất kì



**Lưu ý:**

* Bỏ link domain và một link ví dụ hình ảnh đã filtered bởi app vào file README.
* Tạo một repo trên github có 2 nhánh main và beta và liên kết tài khoản github đó với udacity trước khi submit
* Nếu deploy fail bạn hãy kiểm tra lại trong tab Events xem error là gì hoặc có thể check log từ instance để xem chi tiết
* Nếu sửa src, deploy lại bằng lệnh eb deploy

# **Project 03: Refactor Monolith to Microservices**

Mục tiêu của dự án "Refactor Monolith to Microservices" là tách ứng dụng hiện có có tên là Udagram thành một kiến trúc microservices. Bạn sẽ phải xây dựng quy trình CI/CD tự động xây dựng và triển khai các Docker image lên một Kubernetes cluster. Kubernetes cluster sẽ được cấu hình để giải quyết các thách thức liên quan đến scale và bảo mật.

* **Một số service dùng trong project:**

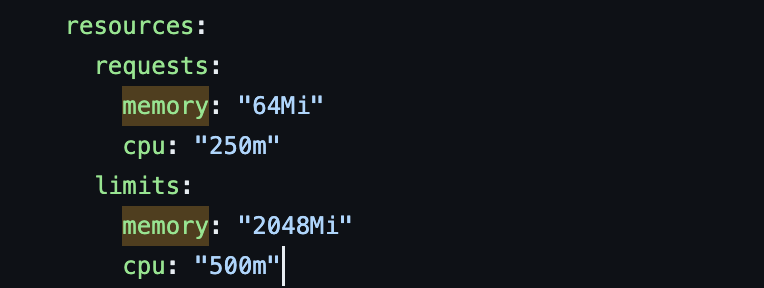
Github action (có thể dùng thay cho circleci nếu không muốn sử dụng thẻ), docker hub, eks

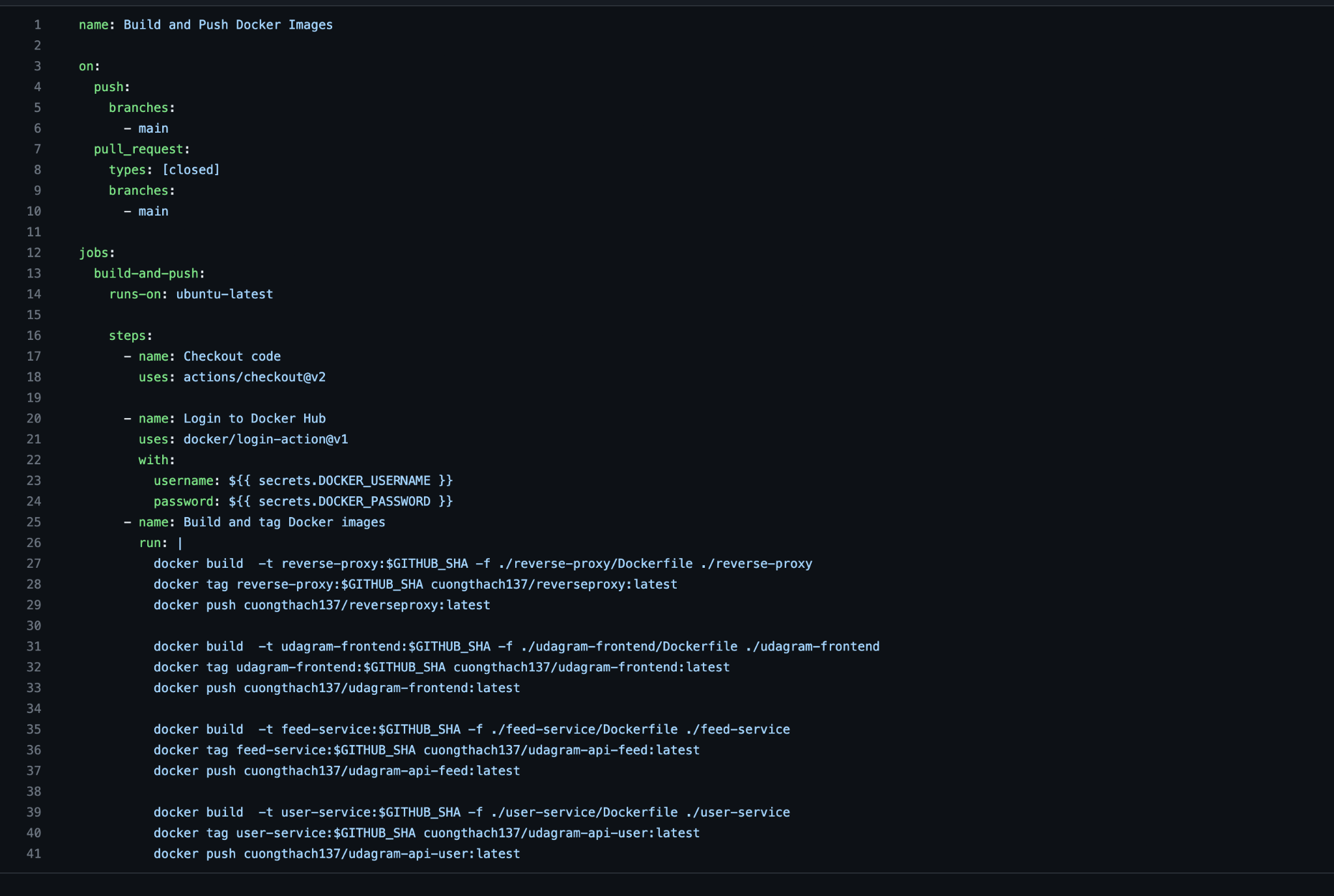
Chi tiết về project:

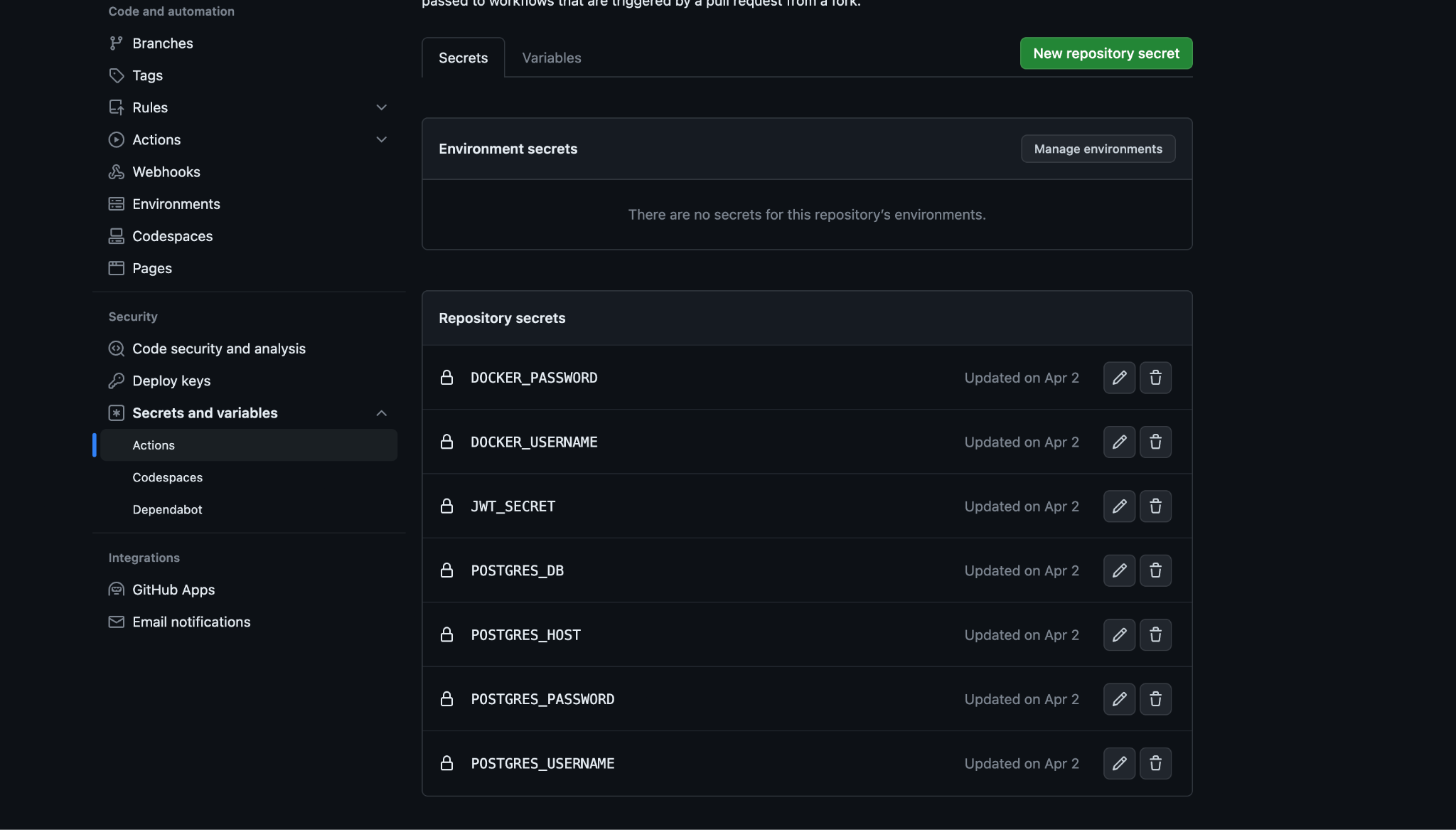
**Step 1**: Tạo một public S3 bucket, config lại policy và cors để bất kì ai cũng có thể truy cập

**Step 2**: Truy cập AWS RDS và tạo postgresql database, config theo hướng dẫn trong khóa học

**Step 3**: Tách backend thành feed-service và user-service.   
Lưu ý trong phần deployment của các service, ở mục resources [config cpu và memory](https://github.com/jamessimonsd/udacity-project/tree/main/deployments) cho instance cho phù hợp nếu không khi chạy app sẽ bị lỗi thiếu tài nguyên

Ở phần ci-cd thay vì dùng circle ci (theo mình nhớ là phải cấp thông tin thẻ thanh toán thì mới dùng được free plan) có thể dùng luôn github action, chỉ cần ghi chú lại trước khi submit hoặc ở trong file ảnh, [tham khảo](https://github.com/jamessimonsd/udacity-project/blob/main/.github/workflows/deploy-image.yml)

Trong phần settings của repo hãy cung cấp các biến môi trường cần thiết

****

**Project 04: Serverless application**

Mục tiêu của dự án Serverless Application là xây dựng một ứng dụng web cho phép người dùng tạo, cập nhật và xóa các mục TODO. Dự án này sẽ giúp bạn làm quen với việc triển khai ứng dụng serverless trên AWS Lambda và sử dụng các dịch vụ như API Gateway, DynamoDB và S3.

* **Một số service dùng trong project:**

Lambda, API Gateway, DynamoDb, S3, Serverless framework

**Chi tiết về project:**

**Step 1:** Tạo tài khoản Auth0, serverless. Tạo app trên các service này.

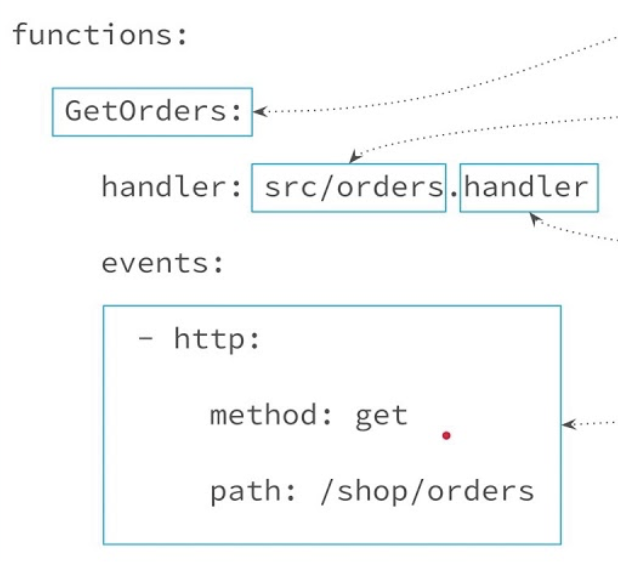
**Step 2:** Cài serverless cli, nên sử dụng phiên bản mới nhất 3.xx

**Step 3:** Clone [starter files](https://github.com/udacity/cloud-developer/tree/master/course-04/project/c4-final-project-starter-code) về máy, có thể tham khảo file [package.json](https://github.com/jamessimonsd/serverless/blob/main/backend/package.json) này của mình đã update version các package

**Step 4**: Vào thư mục backend xem file serverless.yml và bắt đầu thực hiện hoàn thành các mục Todos

Đây là cấu trúc cơ bản của một file serverless.yml mà bạn cần biết:

* service: Thuộc tính này xác định tên của dự án hoặc dịch vụ serverless của bạn. Nó là bắt buộc và phải là một chuỗi.
* provider: Đây là một đối tượng định nghĩa nhà cung cấp dịch vụ cloud mà bạn đang sử dụng, chẳng hạn như AWS, Azure, hay Google Cloud. Bạn cần chỉ định provider và các thông số cấu hình liên quan đến nhà cung cấp trong đối tượng này.
* functions: Đây là một đối tượng chứa tất cả các hàm của ứng dụng serverless của bạn. Mỗi hàm được định nghĩa trong đối tượng này với tên và cấu hình của nó.
* events: Đối tượng events xác định các sự kiện sẽ kích hoạt hàm. Ví dụ: HTTP endpoints, message queues, timers, etc. Mỗi sự kiện được định nghĩa trong một mảng trong đối tượng events.



* resources: Đối tượng resources cho phép bạn định nghĩa các tài nguyên hạ tầng không phải là hàm, chẳng hạn như csdl, message quebe. Bạn có thể sử dụng các nguồn này trong các function của mình.
* custom: Đối tượng custom cho phép bạn định nghĩa các cấu hình tùy chỉnh khác không thuộc vào các thuộc tính khác. Ví dụ: biến môi trường, plugins, etc.

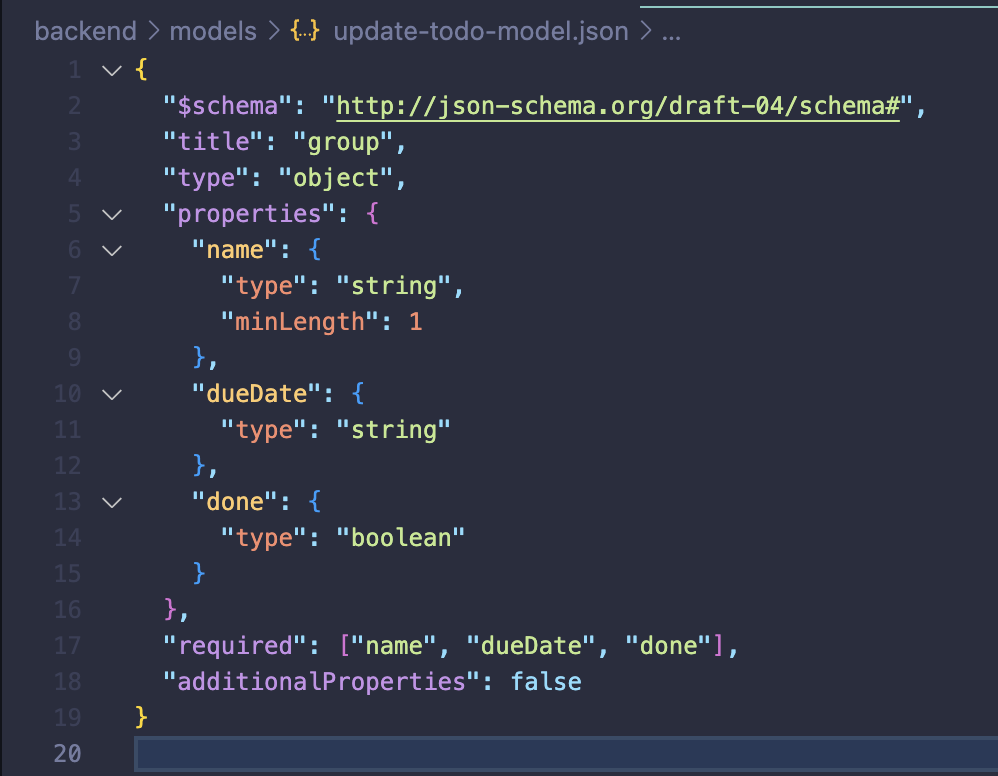
Một số lưu ý:

* Deploy bằng lệnh sls deploy (bỏ -v), nếu không dùng default profile thì thêm tên của profile sẽ sử dụng bằng lệnh --aws-profile serverless
* Trong phần runtime chọn nodejs18.x hoặc cao hơn nếu dùng cli bản 3.x
* [**auth0Authorizer.ts**](https://github.com/jamessimonsd/serverless/blob/main/backend/src/lambda/auth/auth0Authorizer.ts)

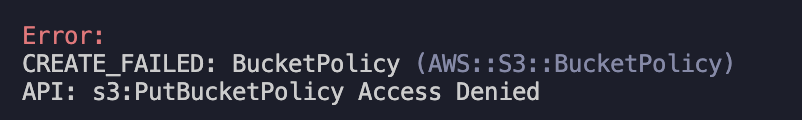
Để hoàn thành được phần này bạn cần xem thật kĩ phần Authentication trong bài học

Truy vập vào auth0, trong phần settings của apps chọn advanced settings và vào tab endpoints để lấy url tải certificate để verify JWT signature

* Lưu ý bạn sẽ phải thêm validation vào các file trong folder models. Mentor sẽ test validation ở client app. Ví dụ:



* Nếu gặp lỗi dưới đây

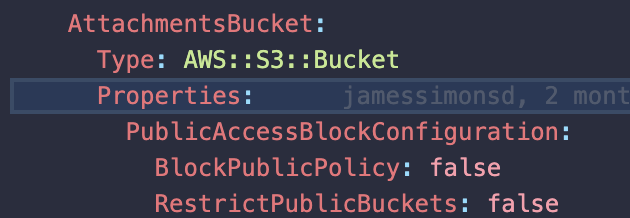


Bạn thêm mấy dòng này vào properties của resource

PublicAccessBlockConfiguration:

BlockPublicPolicy: false

RestrictPublicBuckets: false



# **Project 05: Capstone**

Mục tiêu của project này như để tổng hợp lại kiến thức mà bạn đã học về triển khai microservices hay một serverless app. Bạn sẽ được chọn làm 1 trong 2.

Cá nhân mình chọn serverless app vì phần backend triển khai rất nhanh, bạn chỉ cần dùng nguyên của project 4 sang và thêm 1 2 API. Phần Frontend chỉ cần thêm tính năng tương ứng với api đó.

Tham khảo [repo](https://github.com/jamessimonsd/final)

# **Support Channel**

* **Sử dụng kênh support của Udacity**:
* Knowledge Portal Udacity/ Mentor Help/ Udacity GPT cho các vấn đề về nội dung
* Account Help cho các vấn đề kĩ thuật: <https://udacityenterprise.zendesk.com/hc/en-us/requests/new>
* AWS Cloud Architect Teams Groups
* **Contact Point**: DanPNL (HN+OB)/ ThaoNT39 (ĐN+ Hue +QN)/ VyTT4 (HCM+CT)