CLOUD DEVELOPER

Udacity Nanodegrees

Author: Pham Tat Dat - DatPT45

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 modules, tương ứng với 4 project bao gồm:

- Cloud Fundamentals: Giới thiệu cơ bản về AWS
- Full Stack Apps on AWS: Giới thiệu về cách lập trình, viết test case, stora data, build &deploy một Full stack app trên AWS
- Monolith to Microservices at Scale: Giới thiệu về kiến trúc Microservice, docker, k8s và cách deploy 1 ứng dụng Microservice với docker và K8s. Ngoài ra, bạn cũng sẽ làm quen với khái niệm CI/CD trong quá trình phát triển phần mềm (basic)
- Develop & Deploy Serverless App: Làm quen với Rest API, deploy ứng dụng sử dụng serverless framework trên AWS, các best practices khi deploy ứng dụng với serverless framework

Mỗi project sẽ là một bước nhỏ hướng tới việc giúp cho lập trình viên có thể phát triển ứng dụng trên Cloud, cụ thể là AWS, từ ứng dụng Monolith tới Microservice. Với 2 project đầu khá là basic, sẽ cần nhiều kiến thức về AWS hơn 1 chút, nhưng 2 project sau sẽ cần khá nhiều về kiến thức lập trình Typescript. Lưu ý, cần bám sát từng bài giảng và exercise, các phần này sẽ giúp làm project nhanh hơn, đặc biệt là Project 04.

Ngoài ra, cuối khóa sẽ có 1 Capstone project. Capstone project là project tổng hợp, yêu cầu bạn phải sử dụng hoặc Project 3 hoặc Project 4 để làm, chú ý follow theo Project Rubric.

Prerequisite

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

- AWS Skills (chủ yếu về các service phổ biến như IAM, S3, Lambda, API Gateway, CloudFormation)

- Programing: Nodejs
- IDE: Visual studio code
- AWS CLI
- Serverless
- Auth0 account
- Có kiến thức về security là một lợi thế

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

Attention

Mỗi người sẽ được cấp 1 Account AWS với Budget $75 cho toàn khóa học, và 1 lần xin extend Budget $25. Nếu vượt quá số này bạn sẽ phải tự sử dụng môi trường riêng để làm. Nên mọi người cần chú ý một số vấn đề:

- Cố gắng làm project gói gọn trong 1-3 ngày, tránh kéo dài vì resource đã tạo để lâu sẽ càng mất nhiều chi phí, toàn bộ resource đang tính tiền theo model PAY AS YOU GO, nghĩa là cứ tạo là phải trả phí, dùng bao nhiêu trả bấy nhiêu
- Luôn luôn xóa hết resource sau khi làm xong 1 project, có thể vào từng service trên AWS Console để xóa.
- Project 03 - Monolith to Microservices at Scale là project dễ làm mọi người tốn nhiều chi phí nhất do tạo rất nhiều resource EC2. Đặc biết lưu ý thời gian làm project này. Cụ thể cách để giảm chi phí cho project này mình sẽ nói ở phần sau.
- Về quyền được cấp trong AWS, nên dùng luôn credential của udacty generate ra và add vào AWS CLI dùng thôi, không vướng gì cả. Và vì nó là STS token thì có expired, gen lại và add lại vào AWS configure thôi, mất có vài giây mà, còn đỡ hơn mọi người vào tạo IAM rồi access key các thứ để rồi bị dính permission denied.
- Mỗi account cấp cho mọi người là nó thuộc 1 OU trong AWS org rồi, nó dc design để có đủ permision cho course đó, trừ trường hợp là bị cấp nhầm aws account của course khác (OU khác) thì mới dính lỗi permision -> Nhưng cái này rất hiếm

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.

Project 01: Cloud Fundamentals

Trong học phần này bạn sẽ được học về các khái niệm cơ bản về Cloud Computing, một số service gắn với domain quan trong trong cloud như:

- EC2, S3, CloudFront, DynamoDB
- Security (AWS Firewall, AWS Shield và IAM)
- Networking (Route53, Autoscaling, ALB)
- Messaging & Containers: SNS, SQS, Docker, ECS
- AWS management: Cloutrail, Cloudformation, AWS CL

Project01: Public 1 ứng dụng cơ bản lên Cloud, sử dụng S3, CloudFront và IAM policy để secure S3 bucket:

Với project này, mình đánh giá khá là cơ bản, mọi người chỉ cần follow theo từng bước trong project là hoàn thành. *Estimated time*: 2h

Project 02: Full Stack Apps on AWS

Trong học phần này bạn sẽ tìm hiểu các nguyên tắc cơ bản về thiết kế, triển khai và cung cấp dịch vụ cho các ứng dụng đơn giản trên Cloud bao gồm:

- Design app logic và database
- Viết các function và test case
- Implement authentication, implement JWT
- Deploy ứng dụng với Elastic Beanstalk
- Làm quen và hiểu cách hoạt động của DNS, CDN

Một số kiến thức cần:

- Python
- Nodejs
- Github
- AWS CLI
- Postman

Project02: Chỉnh sửa một project có sẵn "Image Filter" cho phép download Image với URL. User nhập URL image, hệ thống trả về ảnh.

Một số lưu ý:

- Cần verify URL user nhập vào (http validator)
- Sử dụng trực tiếp credential của IAM user mà cấp mà không cần tạo IAM user mới, dễ phát sinh lỗi thiếu quyền
- Code cần lưu trên github để tracking và submit project

Đối với phần standout suggestion:

- Add basic authen cho api
- Sử dụng Route53 để add custom domain, tuy nhiên bạn phải có sẵn domain riêng của mình và add vào Route53, nếu không có thì bỏ qua cũng được.

Nhìn chung project này không yêu cầu code nhiều, khoảng 20 dòng là có thể pass, tập trung vào phần deploy ứng dụng với Elastic Beanstalk, làm theo hướng dẫn là có thể pass.

Note: Một số bạn sử dụng máy Windows để làm sẽ gặp lỗi khi build tương tự như sau:

```
> npm run clean && tsc && copy package.json www/package.json && mkdir www/tmp/ && cd www && zip -r Archive.zip . && cd ..

> udacity-c2-image-filter@1.0.0 clean C:\Users\                                \cloud-developer-master\course-02\project\image-filter-starter-code
> rimraf www/ || true

The syntax of the command is incorrect.
npm ERR! code ELIFECYCLE
npm ERR! errno 1
npm ERR! udacity-c2-image-filter@1.0.0 build: `npm run clean && tsc && copy package.json www/package.json && mkdir www/tmp/ && cd www && zip -r Archive.zip .
npm ERR! Exit status 1
npm ERR!
npm ERR! Failed at the udacity-c2-image-filter@1.0.0 build script.
npm ERR! This is probably not a problem with npm. There is likely additional logging output above.

npm ERR! A complete log of this run can be found in:
```

Lỗi này xuất hiện do Windows không có lệnh **ZIP.** Có thể tham khảo cách sau:

- Hướng dẫn fix từ Udacity trong course material: Building and Deploying (udacity.com) – Note for Windows Users
- Hoặc thay lệnh Build trong file package.json bằng lệnh sau. Lệnh này mình tự chỉnh sửa và thay thế dựa trên mục đích của lệnh trên linux.

```
npm run clean && tsc && copy package.json www\\ && cd www && tar -cvzf Archive.zip * && cd ..
```

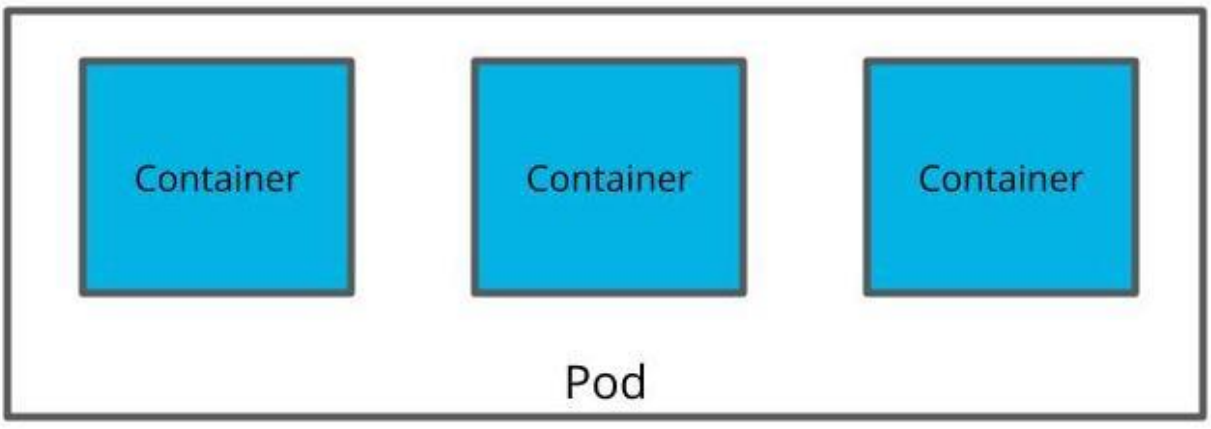
```
tatda@Luffy MINGW64 ~/Documents/Project02
$ npm run build

> udacity-c2-image-filter@1.0.0 build
> npm run clean && tsc && copy package.json www\ && cd www && tar -cvzf Archive.zip * && cd ..


> udacity-c2-image-filter@1.0.0 clean
> rimraf www/ || true

        1 file(s) copied.
package.json
server.js
server.js.map
util/
util/util.js
util/util.js.map

tatda@Luffy MINGW64 ~/Documents/Project02
$ dir www/
Archive.zip  package.json  server.js  server.js.map  util
```

Project 03: Monolith to Microservices at Scale

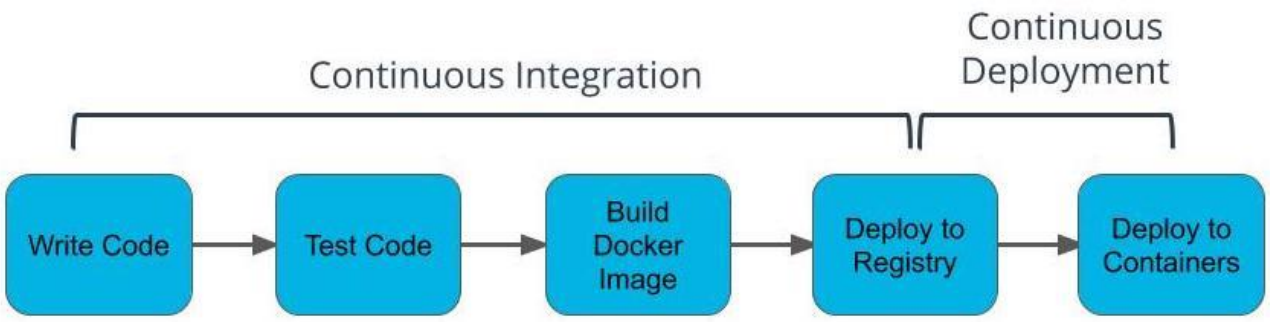
Học phần này tiếp nối Project 2 tuy nhiên sẽ có sử thay đổi lớn trong kiến trúc ứng dụng và kiến trúc triển khai. Nếu như project 2 là ứng dụng Monolith thì trong phần này sẽ hướng dẫn chuyển từ ứng dụng Monolith sáng microservice. Bạn sẽ được giới thiệu và hiểu thế nào là Microservice, ưu nhược điểm, các deploy một ứng dụng Microservice sao cho hiệu quả.

Microservice vs. Monolith Architecture

Bạn sẽ được học về container, Docker và Kubenestes, những công nghệ về Container Orchestration đang phổ biến và sử dụng rộng rãi trên thế giới.



Bạn sẽ được học về Automating the Application Development Lifecycle – CI/CD



Một số kiến thức cần:

- Frontend develop: JS + HTML
- Backend: Nodejs
- AWS RDS, S3, AWS EKS
- Git
- Travis / Github Action
- Kubectl

Setup tool

Ngoài các công cụ lập trình, cần phải cài các công cụ sau (link cài đặt mọi người tự search GG):

- AWS CLI: để giao tiếp với AWS
- Kubectl: Để manage k8s

Project03: Chia làm 3 phần chính

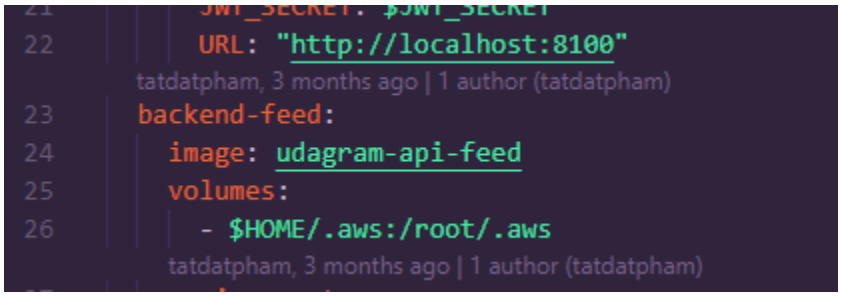
- Phần 1: Chuyển ứng dụng Monolithic sang Microservices
- Phần 2: Continuous Integration: Tự động build image và push image lên docker hub sử dụng Travis
- Phần 3: Container Orchestration with K8s: Deploy ứng dụng trên K8s với AWS EKS

Phần 1: Chuyển ứng dụng Monolithic sang Microservices

Phần này không có gì quá phức tạp, mọi người làm theo hướng dẫn là chắc chắc xong. Một lưu ý khi build docker sẽ có sự khác biệt giữa việc bạn dùng máy Windows hoặc Linux/MacOS ở docker command trong file docker-compose.yml.

Cụ thể là một số người sau khi build nhưng run image thì app không chạy được do không nhận AWS credential.

Lý do là có thể trên máy Windows biến $HOME không được set hoặc nhận không đúng. Sẽ cần set lại



Để set $HOME thì chạy lệnh sau:

- Trên Powershell: $env:HOME="C:/Users/{YOUR_USERNAME}"
- Nếu dụng git windows thì chạy export HOME=C:/Users/{YOUR_USERNAME}

Trong đó YOUR_USERNAME là tên user trên máy windows của bạn.

Phần 2: Continuous Integration

Sử dụng Travis để tự động build docker image và Push lên Docker Hub. Bạn sẽ cần phải tạo tài khoản của Travis và Docker Hub, sau đó link Travis với Github.

Tuy nhiên một số bạn gặp vấn đề với Travis do không có thẻ tín dụng để Trial hoặc hết hạn Trial thì có thể dùng Github Action thay thế, không vấn đề gì cả, chỉ cần giải thích cho Reviewer biết là được. Mình cũng dùng Github Action thay thế.

Phần này cũng khá đơn giản. Có nhiều sample bạn search sample và chỉnh sửa theo ý là được.

Phần 3: Container Orchestration with K8s

Phần này mình đánh giá tương đối khoai nếu anh em lần đầu tiếp xúc với K8s và AWS EKS.

Đầu tiên là tạo EKS cluster, mình khuyên là nên tạo từ AWS Console thay vì tạo từ EKSCTL. Việc tạo này chỉ cần làm theo hướng dẫn từ AWS và gán một số Policy (theo note từ Udacity).

Bạn sẽ cần tạo 2 Role cho EKS Cluster và EKS Node Group trước rồi mới có thể tạo EKS Cluster

Step 1. Create the EKS Cluster

Follow the instructions provided by AWS on *Creating an Amazon EKS Cluster*. Make sure that you are following the steps for *AWS Management Console* and not *eksctl* or *AWS CLI*.

During the creation process, the EKS console will provide dropdown menus for selecting options such as IAM roles and VPCs. If none exist for you, follow the documentation that is linked in the EKS console. Here are some tips for you:

- For the *Cluster Service Role* in the creation process, create an AWS role for EKS. Make sure you attach the policies for `AmazonEKSClusterPolicy`, `AmazonEC2ContainerServiceFullAccess`, and `AmazonEKSServicePolicy`.
- If you don't have a VPC, create one with the `IPv4 CIDR block` value `10.0.0.0/16`. Make sure you select `No IPv6 CIDR block`.
- Your *Cluster endpoint access* should be set to *Public*
- Your cluster may take ~20 minutes to be created. Once it's ready, it should be marked with an *Active* status.

Step 2. Create the EKS Node Groups

Once your cluster is created, we will need to add Node Groups so that the cluster has EC2 instances to process the workloads.

Follow the instructions provided by AWS on Creating a Managed Node Group. Similar to before, make sure you're following the steps for *AWS Management Console*. Here are some tips for you:

- For the *Node IAM Role* in the creation process, create an AWS role for EKS Node Groups. Make sure you attach the policies for `AmazonEKSWorkerNodePolicy`, `AmazonEC2ContainerRegistryReadOnly`, and `AmazonEKS_CNI_Policy`.
- We recommend using `m5.large` instance types
- We recommend setting 2 minimum nodes, 3 maximum nodes

Tới bước deployment, việc đầu tiên là create **kubeconfig** để có thể dung kubectl connect tới EKS cluster. Chạy lệnh

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

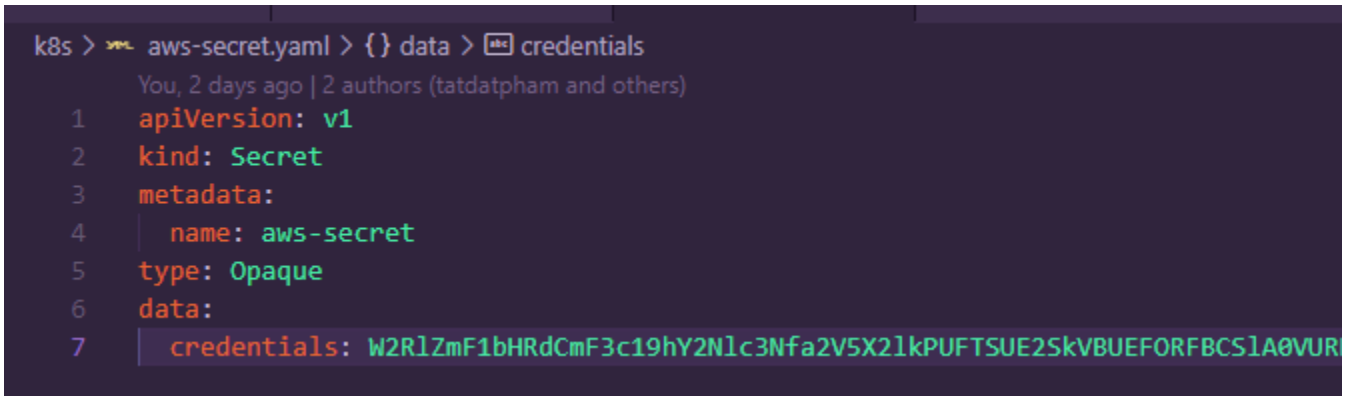
Thay thế region-code và tên EKS cluster

Tiếp theo, tạo file deployment bao gồm:

- aws-secret.yaml
- env-secret.yaml
- env-configmap.yaml
- backend-feed-deployment.yaml
- backend-feed-service.yaml
- backend-user-deployment.yaml
- backend-user-service.yaml
- frontend-deployment.yaml
- frontend-service.yaml
- reverseproxy-deployment.yaml
- reverseproxy-service.yaml
-

a. **aws-secret.yaml**

File chứa aws credential. Bạn phải encode base64 nội dùng của file **~/.aws/credentials**. Có thể copy nội dùng file và vào trang Base64 Encode and Decode - Online để encode nếu bạn không quen dùng lệnh. Nếu file **~/.aws/credentials** có chứa nhiều AWS Credential, bạn chỉ cần copy profile chứa cặp AWS access key và secret key của project.



b. env-secret.yaml

File chứa database credential (trong project này)

Sample config

```
apiVersion: v1
kind: Secret
metadata:
  name: env-secret
type: Opaque
data:
  POSTGRES_USERNAME: DB_USERNAME_ENDCODE_BASE64
  POSTGRES_PASSWORD: DB_PASSWORD_ENDCODE_BASE64
```

c. env-configmap.yaml

File chứa các environment variable sẽ được sử dụng trong các container. Sample config

```
apiVersion: v1
kind: ConfigMap
data:
  AWS_BUCKET: YOUR_S3_BUCKET
  AWS_PROFILE: default
  AWS_REGION: eu-east-1
  JWT_SECRET: Th1s1sR4nd0mS3cr3t
  POSTGRES_DB: postgres
  POSTGRES_HOST: YOUR__HOST
  URL: YOUR BACKEND API
  NODE_OPTIONS: --max-old-space-size=1024
metadata:
  name: env-config
```

d. backend-feed-deployment.yaml

File k8s deployment**,** sample config

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    service: backend-feed
  name: backend-feed
spec:
  replicas: 1
  selector:
    matchLabels:
      service: backend-feed
  template:
    metadata:
      labels:
        service: backend-feed
    spec:
      containers:
      - image: <YOUR_IMAGE>
        name: backend-feed
        imagePullPolicy: Always
        resources:
          requests:
            memory: "512Mi"
            cpu: "150m"
          limits:
            memory: "1024Mi"
            cpu: "300m"
        env:
        - name: URL
          valueFrom:
            configMapKeyRef:
              name: env-config
              key: URL
        - name: AWS_BUCKET
          valueFrom:
            configMapKeyRef:
              name: env-config
              key: AWS_BUCKET
        - name: AWS_PROFILE
          valueFrom:
            configMapKeyRef:
              name: env-config
              key: AWS_PROFILE
        - name: AWS_REGION
          valueFrom:
            configMapKeyRef:
              name: env-config
              key: AWS_REGION
        - name: JWT_SECRET
```

```
          valueFrom:
            configMapKeyRef:
              name: env-config
              key: JWT_SECRET
        - name: POSTGRES_DB
          valueFrom:
            configMapKeyRef:
              name: env-config
              key: POSTGRES_DB
        - name: POSTGRES_HOST
          valueFrom:
            configMapKeyRef:
              name: env-config
              key: POSTGRES_HOST
        - name: POSTGRES_PASSWORD
          valueFrom:
            secretKeyRef:
              name: env-secret
              key: POSTGRES_PASSWORD
        - name: POSTGRES_USERNAME
          valueFrom:
            secretKeyRef:
              name: env-secret
              key: POSTGRES_USERNAME
        - name: NODE_OPTIONS
          valueFrom:
            configMapKeyRef:
              name: env-config
              key: NODE_OPTIONS
        volumeMounts:
        - name: aws-secret
          mountPath: "/root/.aws/"
          readOnly: true
      restartPolicy: Always
      volumes:
      - name: aws-secret
        secret:
          secretName: aws-secret
```

e. backend-feed-service.yaml

Sample config:

```
apiVersion: v1
kind: Service
metadata:
  labels:
    service: backend-feed
  name: backend-feed
spec:
```

```
  ports:
  - name: "8080"
    port: 8080
    targetPort: 8080
  selector:
    service: backend-feed
```

Các file deployment và service còn lại làm tương tự. Sau khi tạo xong thì deploy lên K8s bằng lệnh kubectl apply -f <yaml_file>

Enable Horizontal Pod Autoscaling (HPA)

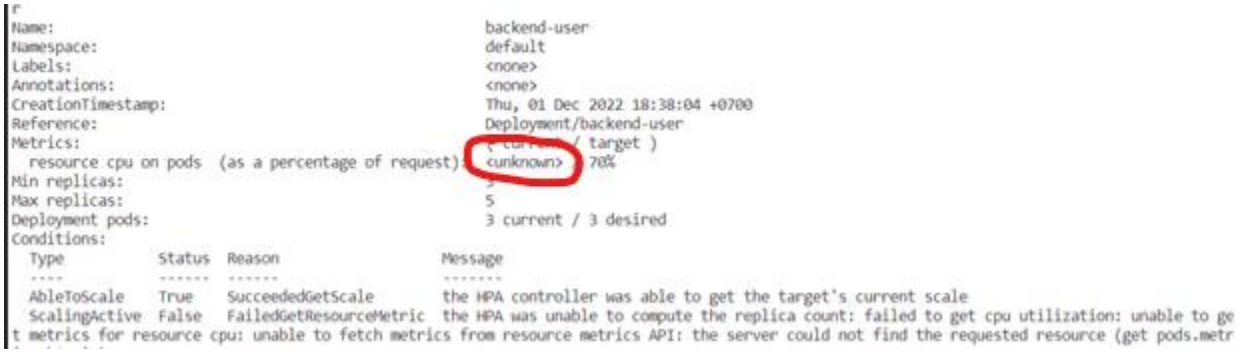
Chạy lệnh sau để enable HPA

```
kubectl autoscale deployment <deployment_name> --cpu-percent=70 --min=2 --max=3
```

Trong đó:

- cpu-percent : Ngưỡng để scale
- min: Num of minimum pod
- max: Num of maximum pod

Nếu status CPU là Unknow như này, bạn install metric server vào k8s để có thể show CPU metrics



Để cài đặt chạy lệnh

```
kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
```

Nếu gặp lỗi trong quá trình deploy, check mục Troubleshoot trong bài, có một số lỗi phổ biến và câu lệnh hay dùng để tìm lỗi.

Một số lệnh phổ biến:

STT	Command	Description
1	kubectl get deployment	Get all deployment
2	kubectl get pods	Get all pods
3	kubectl get svc	Get all service

4	kubectl get hpa	Get all HPA
5	kubectl describe deployment <deployment_name>	Xem chi tiết về 1 deployment
6	kubectl describe pod <pod_name>	Xem chi tiết về 1 pod
7	kubectl exec --stdin --tty <pod_name> -- /bin/bash	SSH vào 1 pod

Project 04: Develop & Deploy Serverless App

Project 04 là học phần cuối trong khóa học, tập trung dạy về việc phát triển và triển khai ứng dụng thông qua serverless.

Nội dung khóa học bao gồm:

- Giới thiệu về serverless
- Event processing
- Xác thực phân quyền trên serverless
- Best practice
- Deploy ứng dụng với serverless trên cloud sử dụng API Gateway, Lambda

Phần này theo mình thì khá nặng và nhiều kiến thức, từ việc sử dụng serverless framework, cách sử dụng với DynamoDB, indexing; phân quyền, validator .... Với các phần trước có thể học lướt bỏ một số nội dung vẫn có thể làm được project nhưng phần này bắt buộc phải học kỹ.

Một số lỗi mình gặp trong quá trình làm project:

1. **Serverless account và Serverless.yaml**

Khi bạn tạo serverless account, có 2 option là personal hoặc org, nếu bạn chọn personal thì phải xóa dòng "Org" (line 01) trong file serverless.yaml. Còn nếu tạo account type là org thì điền org name của bạn vào.

2. **Change package.json trong backend**

Khi mình build backend thì phát hiện file package.json ban đầu họ đưa bị lỗi, sẽ không cài được 1 số package đã không tồn tại để mà cài nữa, việc cần làm là bạn load file package json mới ở đây về : https://knowledge.udacity.com/questions/912446 sau đó thì "npm install" như bình thường.

3. **Error when import aws-xray-sdk**

Trong code starter có import sẵn XRAY module (file helper/attachmentUtils.ts và helper/todosAccess.ts) nhưng báo lỗi import. Nếu bạn gặp lỗi này thì có thể đổi thành như sau:

```
import * as AWS from 'aws-sdk'
const AWSXRay = require('aws-xray-sdk')
import { createLogger } from '../utils/logger'
const XAWS = AWSXRay.captureAWS(AWS)
```

Link tham khảo https://knowledge.udacity.com/questions/70893

4. **Change S3 bucket name trong serverless.yml**

Trên AWS, S3 bucket name là global name nghĩa là tên không thể trùng nhau được. Default trong serverless.yml file, S3 bucket đã được setname.

```
  environment:
    ATTACHMENT_S3_BUCKET: serverless-c4-todo-images-${self:provider.stage}
```

Điều này dẫn tới việc nếu mọi người để nguyên giá trị của "ATTACHMENT_S3_BUCKET" thì lúc deploy sẽ báo lỗi ở S3 bucketname, có thể đã tồn tại do cũng có người dùng S3 bucketname này rồi.

Để fix lỗi này thì bạn chỉ cần thêm 1 random string trước S3 bucket name là xong, đảm bảo không bị trùng với ai khác nữa:

```
  environment:
              ATTACHMENT_S3_BUCKET:      asweq1d13aa-datpt45-serverless-c4-todo-images-
${self:provider.stage}
```

5. **Validator request**

Có 2 cách để validate request tới API:

- Dùng validator plugin trong serverless famework
- Dùng schema validate

Mình khuyến khích dùng schema validate vì nó đơn giản và config nhanh. Schema sẽ dự vào json model để validate request.

Ví dụ về validate request trong function create todo:

```
  CreateTodo:
    handler: src/lambda/http/createTodo.handler
    events:
      - http:
          method: post
          path: todos
          cors: true
          authorizer: Auth
          request:
            schemas:
```

```
            application/json: ${file(models/create-todo-model.json)}
```

6. Check empty todo name

Trong Project rubric không có nói tới việc phải check TodoName empty nhưng khi review thì reviewer luôn nhắc tới phần này và yêu cầu mình làm. Việc này khá đơn giản, dùng schema request validator để xử lý. Mở file models/create-todo-model.json và sửa như sau:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "group",
  "type": "object",
  "properties": {
    "name": {
      "type": "string",
      "minLength": 1
    },
    "dueDate": {
      "type": "string"
    }
  },
  "required": [
    "name",
    "dueDate"
  ],
  "additionalProperties": false
}
```

7. Enable CORS

Để enable CORS sẽ cần thực hiện 2 chỗ là trong response trả về của Lambda func và trên API Gateway thông qua serverless.yml config:

Với serverless.yml thì bạn thêm **cosr: true** như dưới:

```
  CreateTodo:
    handler: src/lambda/http/createTodo.handler
    events:
      - http:
          method: post
          path: todos
          cors: true
```

Với response của lambda function thì thêm **Handler** như dưới

```
export const handler = middy(
  async (event: APIGatewayProxyEvent): Promise<APIGatewayProxyResult> => {
```

```
    const todoId = event.pathParameters.todoId
    // TODO: Remove a TODO item by id

  }
)

handler
  .use(httpErrorHandler())
  .use(
    cors({
      credentials: true
    })
  )
```

8. **Thêm `iamRoleStatements` trong từng lambda function thông qua serverless.yml**

Lưu ý: Việc thêm IAM role này phải đảm bảo permission tối thiểu cần để func có thể chạy không được thêm thừa, vì reviewer cũng sẽ yêu cầu bạn sửa lại nếu thừa:

Ví dụ về iamRoleStatement cho một lambda function

```
  # Provide property for setting up CORS, Authorizer, iamRoleStatements
  DeleteTodo:
    handler: src/lambda/http/deleteTodo.handler
    events:
      - http:
          method: delete
          path: todos/{todoId}
          cors: true
          authorizer: Auth
    iamRoleStatementsName: delete-todo-role-function
    iamRoleStatementsInherit: true
    iamRoleStatements:
      - Effect: Allow
        Action:
          - dynamodb:Query
          - dynamodb:DeleteItem
                                                                    Resource:
arn:aws:dynamodb:${self:provider.region}:*:table/${self:provider.environment.TODOS_TABLE
}
      - Effect: Allow
        Action:
          - s3:DeleteObject
        Resource: 'arn:aws:s3:::${self:provider.environment.ATTACHMENT_S3_BUCKET}/*'
      - Effect: Allow
        Action:
          - xray:PutTraceSegments
          - xray:PutTelemetryRecords
        Resource:
          - "*"
```

Trong config cho Delete Todo function trên minh đang define permission cho 3 resource. Do function là function delete nên không cần tới quyền update hay create gì ở đây cả:

- S3: Delete object
- Dynamodb: Có quyền query và delete item
- Xray: Put segment và record

Cuối cùng, bạn có thể tham khảo code của các bạn đã làm trước đó và check lại 1 lượt cái yêu cầu trong Project rubric trước khi submit.

Project 05: CAPSTONE

Project 05 là là project cuối khóa, trong project này cho phép bạn lựa chọn 1 trong 2 project đã làm để phát triển thêm tính năng dựa trên các kiến thức bạn đã học từ đầu tới giờ.

Tham khảo project rubric: Udacity Reviews

- Project 03: Bạn chọn project này nếu kiến thức về K8s của bạn tương đối và có kiến thức về mảng Operation vì với project 03 này, yêu cầu sẽ thiên về việc cấu hình, deploy ứng dụng trên K8S
- Project 04: Chọn project này nếu bạn mạnh về coding Nodejs hơn.

Mình thấy tới 98% anh em đã đang học khóa này chọn Project 04 để làm Project 05 vì có thể là hầu hết là Developer và Project 04 vừa làm nóng hổi xong vẫn còn nắm trong lòng bàn tay =)).

Với mình thì mình chọn Project 03 để làm, 1 phần là thích thử thách chút, phần vì mình có kinh nghiệm với k8s hơn là Nodejs, nói thật là mới học code NodeJS khi học khóa này, từ Project tới Projecr 03 mình làm mất 2 ngày là xong vì nó cũng không nhiều code tới P4 thì tắc tịt vì Code hơi nhiều =)) và Job Title của mình cũng không hẳn là Developer :D.

Với Project 04, thấy anh em bảo chỉ cần viết thêm test case và 1 function delete attachment là có thể pass. Mọi người hỏi thêm kinh nghiệm anh em khác nhé.

Với Project 03, bạn cần Implement Blue/Green deployment và enable Cloudwatch metric, nó sẽ fit với 3. requirement này:

The app can be upgraded via rolling-update	The students can deploy a new version of the application without downtime
A/B deployment of the application	Two versions of the same app can run at the same and service traffic
Monitoring	The application is monitored by Amazon CloudWatch

Mình sẽ guide từng phần:

1. **Enable Cloudwatch metrics**

Tạo service account trên cluster

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cwagent/cwagent-serviceaccount.yaml
```

Tải configmap sample

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cwagent/cwagent-configmap.yaml
```

Mở file và thay thế giá trị **{{cluster_name}}** bằng tên EKS cluster (ekycbpo-prod) và sau đó áp dụng config

```
kubectl apply -f cwagent-configmap.yaml
```

Deploy Cloudwatch agent với DaemonSet

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cwagent/cwagent-daemonset.yaml
```

Kiểm tra việc deploy

```
kubectl get pods -n amazon-cloudwatch
```

Kết quả như dưới là thành công (**running**)

```
NAME                     READY   STATUS    RESTARTS   AGE
cloudwatch-agent-dhhvc   1/1     Running   0          4m37s
```

```
cloudwatch-agent-dtrr5   1/1     Running   0          4m37s
```

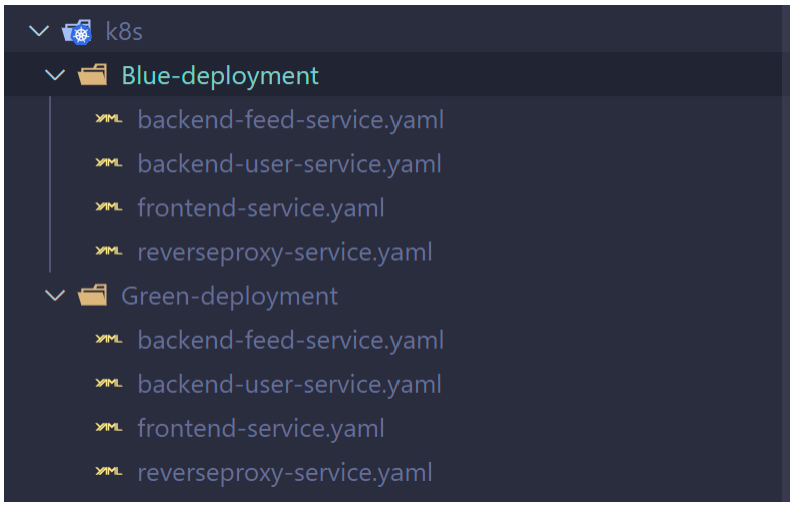
2. Enable Blue/Green deployment

Blue/Green deployment là một thuật ngữ chỉ kỹ thuật duy trình cùng lúc 2 version của app và cho phép route request từ version cũ sang version mới mà không bị downtime đồng thời cũng có thể test app trên version mới với traffic thật.

Tham khảo: [Using AWS Load Balancer Controller for blue/green deployment, canary deployment and A/B testing | Containers (amazon.com)]

Bạn sẽ cần tạo 2 bộ config cho deployment và service cho các app và deploy lên K8s:

- Backend-feed
- Backend-user
- Frontend
- Reverseproxy



Về cơ bản thì configuration sẽ giữ nguyên như project 03, chỉ khác là bạn sẽ cần thêm selector trong metada và đổi tên service.

Ví dụ cho blue deployment:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    service: backend-feed
  name: backend-feed-v1
spec:
  replicas: 1
  selector:
    matchLabels:
      service: backend-feed
      version: v1.1.0
  template:
    metadata:
      labels:
        service: backend-feed
        version: v1.1.0
    spec:
      containers:      tatdatpham, 3 months ago • add k8s deployment configuration …
      - image: tatdat171/udagram-api-feed:v1.1.0
```

Trong config mình có thêm version vào lable để phân biệt giữa 2 phiên bản ứng dụng. Blue deployment mình đang để version v1.1.0

Service config cũng thêm version selector và tên service thêm "**-v1**" để phân biệt phiên bản:

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    service: backend-feed
  name: backend-feed-v1
spec:
  ports:
  - name: "8080"
    port: 8080
    targetPort: 8080
  selector:
    service: backend-feed
    version: v1.1.0
```

Ví dụ cho Green deployment

```
Dat Pham Tat, 5 days ago | 2 authors (tatdatpham and others)
apiVersion: apps/v1      tatdatpham, 3 months ago • Update k8s deployme
kind: Deployment
metadata:
  labels:
    service: backend-feed
  name: backend-feed-v2
spec:
  replicas: 1
  selector:
    matchLabels:
      service: backend-feed
      version: v1.2.0
  template:
    metadata:
      labels:
        service: backend-feed
        version: v1.2.0
    spec:
      containers:
      - image: tatdat171/udagram-api-feed:v1.2.0
```

Vesion đổi sang v1.2.0. Tương tự với service configuration

```
apiVersion: v1
kind: Service
metadata:
  labels:
    service: backend-feed
  name: backend-feed-v2
spec:
  ports:
  - name: "8080"
    port: 8080
    targetPort: 8080
  selector:
    service: backend-feed
    version: v1.2.0
```

Sau khi chỉnh sửa xong 2 bộ config bạn deploy tất cả lên k8s sẽ được kết quả như dưới

```
tatda@Luffy MINGW64 /d/Tatdat171/Work/github/cd0354-monolith-to-microservices-project (main)
NAME              READY   UP-TO-DATE   AVAILABLE   AGE
backend-feed-v1   1/1     1            1           86m
backend-feed-v2   1/1     1            1           4m10s
backend-user-v1   1/1     1            1           86m
backend-user-v2   1/1     1            1           4m3s
frontend-v1       2/2     2            2           3m2s
frontend-v2       2/2     2            2           18s
reverseproxy-v1   2/2     2            2           85m
reverseproxy-v2   2/2     2            2           7s

tatda@Luffy MINGW64 /d/Tatdat171/Work/github/cd0354-monolith-to-microservices-project (main)
$ kubectl.exe get pods
NAME                               READY   STATUS    RESTARTS      AGE
backend-feed-v1-69687c8f58-9rcjk   1/1     Running   0             64m
backend-feed-v2-69bdffb87d-5r4zb   1/1     Running   0             4m20s
backend-user-v1-7b8985dcc6-hv5tf   1/1     Running   0             64m
backend-user-v2-68cc4d456-jggcd    1/1     Running   0             4m13s
frontend-v1-7976b8d74d-2dh6n       1/1     Running   0             3m12s
frontend-v1-7976b8d74d-xxq9r       1/1     Running   0             3m12s
frontend-v2-79f85d5db4-5hdrp       1/1     Running   0             28s
frontend-v2-79f85d5db4-tcf44       1/1     Running   0             28s
reverseproxy-v1-846dc4cd88-6rv7z   1/1     Running   0             64m
reverseproxy-v1-846dc4cd88-gj5mw   1/1     Running   1 (64m ago)   64m
reverseproxy-v2-7c85c776d-5kdvt    1/1     Running   0             17s
reverseproxy-v2-7c85c776d-ppvlm    1/1     Running   0             17s
```

Tiếp theo, tạo ingress để apply Blue/Green deployent

Bạn cũng sẽ cần cài đặt AWS Loadbalancer controller trên K8s để khi bạn tạo Ingress trên K8s, ingress sẽ tự động tạo ALB trên AWS. Tham khảo: [Installing the AWS Load Balancer Controller add-on - Amazon EKS]

Ứng dụng có 2 public url gồm Frontend service và Reverse service (backend).

Tạo ingress config như sau:

Trong đó, “alb.ingress.kubernetes.io/subnets” là public subnets ID, bạn có thể lấy ID trên AWS console, VPC service

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: frontend-ingress
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/scheme: internet-facing
            alb.ingress.kubernetes.io/subnets:     subnet-07698675feef5f3f5,     subnet-
0de8d55ec877c7dca, subnet-0a37626814a280a75, subnet-00ce6c73d69b5b8f9
    alb.ingress.kubernetes.io/target-type: 'ip'
    alb.ingress.kubernetes.io/actions.blue-green-frontend: |
      {
        "type":"forward",
        "forwardConfig":{
```

```
          "targetGroups":[
            {
              "serviceName":"frontend-v1",
              "servicePort":"8100",
              "weight":100
            },
            {
              "serviceName":"frontend-v2",
              "servicePort":"8100",
              "weight":0
            }
          ]
        }
      }
  labels:
    service: frontend-ingress
spec:
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: blue-green-frontend
                port:
                  name: use-annotation
```

Và ingess service cho Reverseproxy service:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: reverseproxy-ingress
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/scheme: internet-facing
            alb.ingress.kubernetes.io/subnets:     subnet-07698675feef5f3f5,     subnet-
0de8d55ec877c7dca, subnet-0a37626814a280a75, subnet-00ce6c73d69b5b8f9
    alb.ingress.kubernetes.io/target-type: 'ip'
    alb.ingress.kubernetes.io/actions.blue-green-reverseproxy: /
      {
        "type":"forward",
        "forwardConfig":{
          "targetGroups":[
            {
              "serviceName":"reverseproxy-v1",
              "servicePort":"8080",
```

```
              "weight":100
            },
            {
              "serviceName":"reverseproxy-v2",
              "servicePort":"8080",
              "weight":0
            }
          ]
        }
      }
  labels:
    service: reverseproxy-ingress
spec:
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: blue-green-reverseproxy
                port:
                  name: use-annotation
```

Sau đó deploy configuration với lênh kubectl apply -f <ingresss_config_file>

Kết quả:

```
tatda@Luffy MINGW64 /d/Tatdat171/Work/github/cd0354-monolith-to-microservices-project (main)
$ kubectl.exe get ingress
NAME               CLASS    HOSTS   ADDRESS                                                                   PORTS   AGE
frontend-ingress   <none>   *       k8s-default-frontend-4af3bd04c1-1578541941.us-east-1.elb.amazonaws.com   80      46s

tatda@Luffy MINGW64 /d/Tatdat171/Work/github/cd0354-monolith-to-microservices-project (main)
$ kubectl.exe get ingress
NAME               CLASS    HOSTS   ADDRESS                                                                   PORTS   AGE
frontend-ingress   <none>   *       k8s-default-frontend-4af3bd04c1-1578541941.us-east-1.elb.amazonaws.com   80      53s

tatda@Luffy MINGW64 /d/Tatdat171/Work/github/cd0354-monolith-to-microservices-project (main)
$ kubectl.exe apply -f k8s/reverseproxy-ingress.yml
ingress.networking.k8s.io/reverseproxy-ingress created

tatda@Luffy MINGW64 /d/Tatdat171/Work/github/cd0354-monolith-to-microservices-project (main)
$ kubectl.exe apply -f k8s/frontend-ingress.yml
ingress.networking.k8s.io/frontend-ingress configured

tatda@Luffy MINGW64 /d/Tatdat171/Work/github/cd0354-monolith-to-microservices-project (main)
$ kubectl.exe get ingress
NAME                  CLASS    HOSTS   ADDRESS                                                                   PORTS   AGE
frontend-ingress      <none>   *       k8s-default-frontend-4af3bd04c1-1578541941.us-east-1.elb.amazonaws.com   80      93s
reverseproxy-ingress  <none>   *       k8s-default-reversep-279d502cbb-1082054293.us-east-1.elb.amazonaws.com   80      16s
```

Bạn test truy cập vào 2 endpoint kia, truy cập được là đã xong, chụp ảnh và submit thôi :D

Support Channel

- **Sử dụng kênh support của Udacity**: Knowledge Portal Udacity/ Mentor Help cho các vấn đề về nội dung
  Account Help cho các vấn đề kĩ thuật: https://udacityenterprise.zendesk.com/hc/en-us/requests/new
- Tham gia group nội bộ FSOFT **Cloud Developer AWS Sep 2022** trên MS Team của FSO
- Contact: AnLT18@fsoft.com.vn