
Cloud Computing A3

Site URL's:

Elastic Beanstalk: cloudfitness.click

API Gateway: api.cloudfitness.click

ECS/EC2: admin.cloudfitness.click

Student Names:

Tomas Haddad: s3811596

Alicia Hallal: s3811836

Introduction

Usage of fitness and health technology, platforms and applications are steadily on the rise, with 39% of adult Australians confirming that they used digital technology to assist in meeting their fitness goals during lockdowns[1]. As such, there is an increasing demand for applications which help users create, plan, and manage their exercises and workouts. Many existing web or mobile applications aim to do just that, but often either provide a broad array of services not useful to these specific needs or are too restrictive and lack any meaningful user interaction.

CloudFitness is a web application that is designed to help users quickly find exercises appropriate for their experience level and interests. Users can select from exercises contained within the public database or can create their own custom exercises if they do not already exist. The motivation for this application is to help users engage with particular exercises so that other like-minded users can benefit from the same exercises by the following: reading their comments, viewing the likes on the exercises, and building their exercise profile pages. This way they can build more efficient and enjoyable exercise routines with the exposure of different exercises and user comments.

CloudFitness is made up of three separate applications which all provide specific functionalities.

Main User Application:

- Users can search exercises by their names
- Allow users to view exercise view and like counts
- Allow users to comment on exercises
- Users can add exercises to their profiles
- Users can submit their own exercises via upload functionality, these must be approved before being published, or upload personal exercises to their profile.

Admin Application:

- Load exercises from API Gateway
- Edit exercises on API Gateway
- Set approved exercises to pending
- Set pending exercises to approved
- View all exercises available from API gateway

REST API

- Get all exercises in the database
- Get exercises by type and name
- Get exercises by approved status
- Update exercise approved status (this requires an API key)

Related work

exrx.net


Exercise Prescription (ExRx) is an online provider of fitness-related information. Their available content includes an exercise library containing over 1,900 entries, fitness calculators, aggregated information covering a diverse range of topics such as weight loss, nutrition, and more.

Of particular interest in the context of this report is their exercise library, which is organised into muscle groups and the type of equipment needed, if any. Each entry provides a short summary of the exercise, including its classification, instructions on how to execute the exercise, a general comments section, and a list of the muscles involved.

A major downside of the website's simple, minimalist structure is that it does not provide any user interaction beyond its public forum; the content of each exercise is controlled entirely by the staff's administration.

Barbell Squat

ExRx.net > Directory > Quads > Exercise



Comments

Keep head facing forward, back straight and feet flat on floor; equal distribution of weight throughout forefoot and heel. Knees should point same direction as feet throughout movement. See:

- Squat mount and dismount
- Spotting technique
 - Spotting assistance
- Side view A
 - Sideview B

Also see Squat for Targeting Glutes and Squat Analysis.

Muscles

Target

- Quadriceps

Synergists

- Gluteus Maximus
- Adductor Magnus
- Soleus

Dynamic Stabilizers

- Hamstrings
- Gastrocnemius

Stabilizers

- Erector Spinae

Antagonist Stabilizers

- Rectus Abdominis
- Obliques

Classification

Utility: Basic

Mechanics: Compound

Force: Push

Instructions

Preparation

From rack with barbell at upperchest height, position bar high on back of shoulders and grasp barbell to sides. Dismount bar from rack and stand with shoulder width stance.

Execution

Squat down by bending hips back while allowing knees to bend forward, keeping back straight and knees pointed same direction as feet. Descend until thighs are just past parallel to floor. Extend knees and hips until legs are straight. Return and repeat.

wger.de

wger provides a public database of exercises in addition to providing tools for managing meals, nutrition, and weight tracking. Users can use their website to search for exercises based on equipment or muscles used. The defining feature of this website is that it provides a REST API for users to retrieve information such as equipment, exercise information, exercise category, meals, ingredients, and more.

Unlike exrx.net, users can submit exercises to the website which are added to the public database subject to administrator approval. Furthermore, users can create accounts to create and manage custom workout routines. However, a major limitation of the website is that users cannot create or manage custom exercises and are limited to the public database of approved exercises.

```
GET /api/v2/equipment/

HTTP 200 OK

Allow:
GET, HEAD, OPTIONS

Content-Type:
application/json

Vary:
Accept

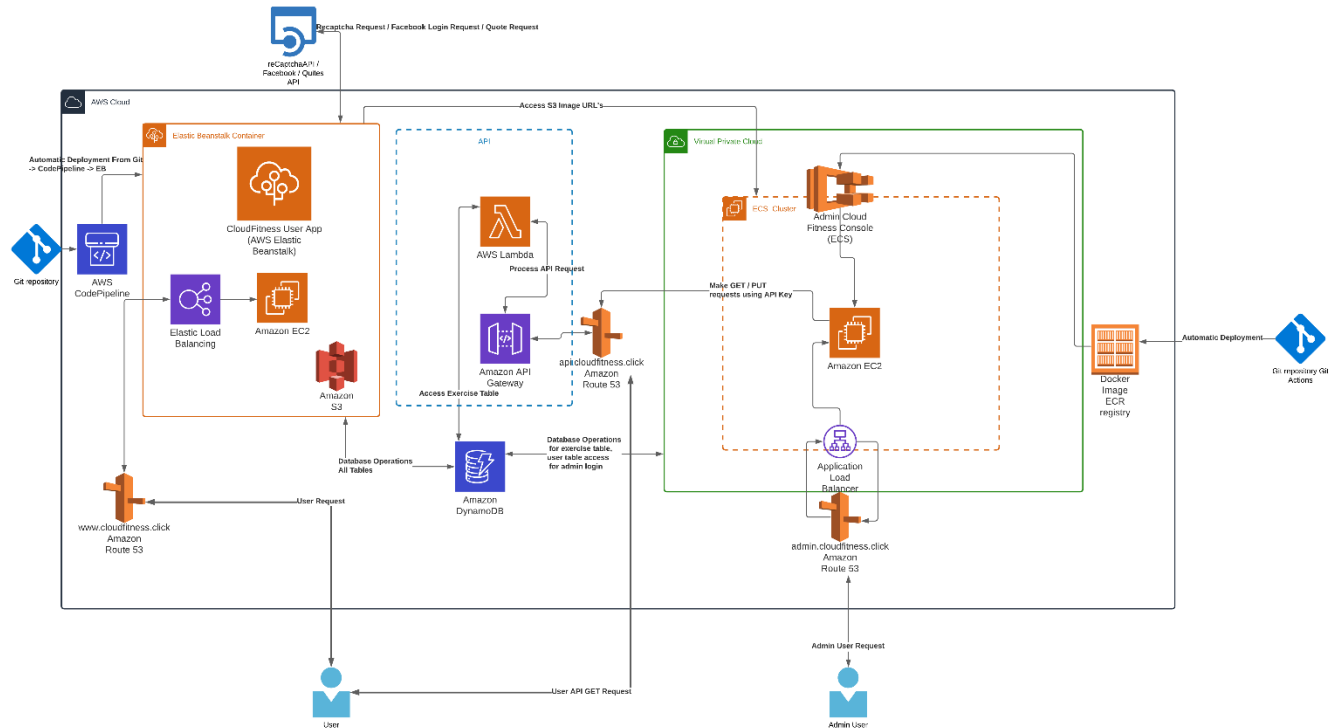
{
  "count": 10,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 1,
      "name": "Barbell"
    },
    {
      "id": 8,
      "name": "Bench"
    },
    {
      "id": 3,
      "name": "Dumbbell"
    }
  ]
}
```

bodybuilding.com

This website is the world's largest online fitness product and information provider, but also provides registered users with the ability to build custom workouts under their "BodySpace" workout builder. Users can build custom workout routines using exercises available in the public database. However, again, users cannot create their own exercises without administrator approval. Furthermore, the majority of exercise information is locked behind paywalls, and the broad scope of the website will not likely attract many users for this specific functionality.

System Architecture

System architecture diagram



System descriptions

Elastic beanstalk (5)

Elastic Beanstalk gives the developer a simple, low configuration solution that is also scalable during higher demand situations. It was the perfect option when trying to decide what was the best way to deploy the main site. This is because it is easy to configure with automated AWS deployment systems such as CodePipeline, and also sets up its own S3 bucket and EC2 instances. This made it quick and easy to deploy the flask application main site once the code for it had been developed. It also had the ability to run container commands before deployment, meaning we could have more control over what happened on the EC2 instances before the application started up, in this case we used it to ensure that the DynamoDB tables were created before the site deployed. Furthermore, access to DynamoDB was easy to configure by giving permissions to the Elastic Beanstalk Role in the IAM AWS console. Elastic Beanstalk has its own load balancer automatically set up upon deployment, which was extremely easy to hook up to Route 53. It has two connection points `www.cloudfitness.click` and `cloudfitness.click`. The application load balancer was easily configured using `.ebextension` config files. Which means that each time the system is deployed, we can be confident that HTTP and HTTPS will work correctly.

ECS (5)

Elastic Container Service allows more control over how the admin flask site was deployed, whilst also giving us a fully managed platform solution. If one of the EC2 instances which hosted the admin site failed, ECS would automatically deploy another instance with the

last working image on it. It also gave us a secure way to deploy our application using github actions and Elastic Container Registry. It does take more time to create the Dockerfile and deployment files initially when using ECS, however, it does pay off in the long run. Environment variables are set as secrets and then automatically built into the application meaning that nothing secure is uploaded to any public/private instance. After the initial set up future deployments are all completely automatic using GitHub actions. Furthermore, ECS gives more options in terms of scalability in the future, we do not expect too much demand currently from the admin site but this may change in the future, with more submissions/uploads on the main site. Using ECS scaling the admin application becomes extremely simple and there would be almost no downtime in launching a new instance that was larger and could handle more requests.

DynamoDB (2)

Separate DynamoDB tables are used to store a number of different types of records: the user table stores login information; the exercise table stores information about each exercise needed to generate URLs, facilitate searching, and render pages; the comments table stores user-submitted comments on exercise pages; the metadata table contains static information, such as exercise types, accessible by the REST API; and existing and custom made exercises are stored in two separate tables to allow users to maintain their own custom exercises separate from exercises submitted to the main website.

API Gateway (5)

API gateway acts as a single-entry point for our backend REST API. Using API gateway's proxy functionality, the request body generated when users visit a valid API URL is effortlessly passed to a lambda function, which then processes the URL based on the parameters given.

Lambda (5)

Amazon Lambda's serverless architecture allows backend functions to be run only when required. Like wger.de, we expose our database to developers as a REST API via *api.cloudfitness.click*. Since this is not a service that needs to be online at all times, a lambda function is instead triggered by API gateway when a valid API call is made.

Route 53 (2)

Amazon Route 53 is used for domain registration and routing. A domain is needed in order to utilise the Facebook login API, which only works with HTTPS-authenticated websites. Furthermore, adjacent services such as the REST API can be mapped to their own subdomains, which allows for changes to be made to application business logic without affecting the frontend web application.

ReCAPTCHA API

reCAPTCHA is used to protect websites from spam or automated abuse (such as bots). The API here is used for login on the main site www.cloudfitness.click and uses reCAPTCHA v2 Checkbox. An API key pair was specifically generated for this site so that it can only be used on this domain, information about requests on the reCAPTCHA api are available to the logged in developer who owns the API key pair.

Facebook login API

Facebook login API allows users to log into the main site www.cloudfitness.click with their facebook logins. This facebook login has an developer account attached to its credentials with cloudfitness as the authorised website domain. Facebook API login only allows users to use the website if it is HTTPS/SSL secure. Facebook user details are automatically sent to the dynamodb database upon their first login to generate them a localised account for the site, users will not be able to log into the site via normal means using these accounts, only via the facebook login API.

Zenquotes API

This simple API is presented to users when they view the main site home page www.cloudfitness.click. It is an API which randomly selects an inspiring quote and presents it to users. This is meant to motivate exercise users of the site so they can get the best out of the exercise routines each time they view the home page.

Developer Manual

Section 1: Admin Site (*Elastic Container Service (EC2) / Elastic Container Registry / GitHub Actions*) [2]

Before beginning this step of development, a functioning admin application is needed, complete with a Dockerfile so that it can be used to upload to the EC2 instance. This application needs to be associated with github.

1. **Create the repository on Elastic Container Registry, and set visibility to private so that it can not be accessed and give it a meaningful name:**

The screenshot shows the ECR console interface. At the top, there are tabs for 'Private' and 'Public'. Below this, a section titled 'Private repositories (1)' contains a search bar and a table of repositories. The table has columns for 'Repository name', 'URI', 'Created at', 'Tag immutability', 'Scan on push', and 'Encryption type'. One repository is listed: 'aws-ecs-clouda3' with URI '059411200951.dkr.ecr.ap-southeast-2.amazonaws.com/aws-ecs-clouda3', created on 'Jun 02, 2021 2:17:14 PM', with 'Tag immutability' set to 'Disabled', 'Scan on push' set to 'Disabled', and 'Encryption type' set to 'AES-256'. Below the table, the 'General settings' section is visible. It includes 'Visibility settings' with 'Private' selected (radio button), 'Repository name' with the text '059411200951.dkr.ecr.ap-southeast-2.amazonaws.com/' and a text input field containing 'aws-ecs-clouda3', and 'Tag immutability' with 'Disabled' selected. A blue information box at the bottom states: 'Once a repository is created, the visibility setting of the repository can't be changed.'

2. **Create the Elastic Container Service Task Definition:**

This will specify how many containers should run as part of the task, what type of server they will run on (in this application we are using an EC2 instance) and how the instances will link together. Click create new task definition.

[Create new Task Definition](#)
[Create new revision](#)
[Actions](#)
Last updated on June 9, 2021 12:18:42 PM (0m ago)

Status: **ACTIVE** INACTIVE

Filter in this page < 1-1 > Page size 50


Task Definition	Latest revision status
<input type="checkbox"/> aws-ecs-clouda3	ACTIVE

Select EC2 instance for launch type compatibility

Select launch type compatibility


Select which launch type you want your task definition to be compatible with based on where you want to launch your task.

FARGATE




Price based on task size
Requires network mode awsvpc
AWS-managed infrastructure, no Amazon EC2 instances to manage

EC2



Price based on resource usage
Multiple network modes available
Self-managed infrastructure using Amazon EC2 instances

EXTERNAL



Price based on instance-hours and additional charges for other AWS services used
Self-managed on-premise infrastructure with ECS Anywhere

*Required

[Cancel](#)

[Next step](#)

Give the task a meaningful name and set the Task role to `ecsTaskExecutionRole`, Network mode to default, IAM role should be `ecsTaskExecutionRole`

Configure task and container definitions

A task definition specifies which containers are included in your task and how they interact with each other. You can also specify data volumes for your containers to use. [Learn more](#)

Task Definition Name* ⓘ

Requires Compatibilities* EC2

Task Role ⓘ
Optional IAM role that tasks can use to make API requests to authorized AWS services. Create an Amazon Elastic Container Service Task Role in the [IAM Console](#) ⓘ

Network Mode ⓘ
If you choose <default>, ECS will start your container using Docker's default networking mode, which is Bridge on Linux and NAT on Windows. <default> is the only supported mode on Windows.

Task execution IAM role
This role is required by tasks to pull container images and publish container logs to Amazon CloudWatch on your behalf. If you do not have the `ecsTaskExecutionRole` already, we can create one for you.

Task execution role ⓘ

Click 'add container' in container definitions, give the container a meaningful name and add the URI of ECR repository created in the previous steps. A memory limit of 128mb is set here as the application does not need to use much memory, and it will be hosted on port 5000, so port 5000 is set as the container port.

Add container

Standard

Container name*

ecs-container-name

Image*

059411200951.dkr.ecr.ap-southeast-2.amazonaws.com/aws-ecs-conda3

Private repository authentication*

☐

Memory Limits (MiB)*

Soft limit

128

Add Hard limit

Define hard and/or soft memory limits in MiB for your container. Hard and soft limits correspond to the 'memory' and 'memoryReservation' parameters, respectively, in task definitions. ECS recommends 300-500 MiB as a starting point for web applications.

Port mappings

Host port	Container port	Protocol
0	5000	tcp

Add port mapping

Advanced container configuration

No other configurations are needed, so click add the container and then create the task so that it shows up in the task definitions screen.

3. Creating the Elastic Container service Cluster:

The cluster for ECS is a grouping of the tasks/instances

Clusters

An Amazon ECS cluster is a regional grouping of one or more container instances on which you can run task requests. Each account receives a default cluster the first time you use the Amazon ECS service. Clusters may contain more than one Amazon EC2 instance type.

For more information, see the [ECS documentation](#).

Create Cluster

Get Started

View

list

card

view all

1 - 1 of 1

ecs-conda3-cluster >

CloudWatch monitoring

Default Monitoring

FARGATE

To create the cluster, click 'create cluster'. Select the cluster template EC2 linux + networking.

Select cluster template

The following cluster templates are available to simplify cluster creation. Additional configuration and integrations can be added later.

Networking only ⓘ
Resources to be created:
Cluster
VPC (optional)
Subnets (optional)
 ⓘ For use with either AWS Fargate or External instance capacity.

EC2 Linux + Networking
Resources to be created:
Cluster
VPC
Subnets
Auto Scaling group with Linux AMI

EC2 Windows + Networking
Resources to be created:
Cluster
VPC
Subnets
Auto Scaling group with Windows AMI

*Required

Cancel

Next step

Give the cluster a meaningful name, set the ec2 instance type to t2.nano and the number of instances to 1. As the task we are running here is a web application that does not need much memory.

Configure cluster

Cluster name* ⓘ

☐ Create an empty cluster

Instance configuration

Provisioning Model ☒ On-Demand Instance

With On-Demand Instances, you pay for compute capacity by the hour, with no long-term commitments or upfront payments.

☐ Spot

Amazon EC2 Spot Instances let you take advantage of unused EC2 capacity in the AWS cloud. Spot instances are available at up to a 90% discount compared to On-Demand prices. [Learn more](#)

EC2 instance type* ⓘ ⓘ

☐ Manually enter desired instance type

Number of instances* ⓘ

EC2 AMI ID* ⓘ

Root EBS Volume Size (GiB) ⓘ

Key pair ⓘ ⓘ



You will not be able to SSH into your EC2 instances without a key pair. You can create a


Ensure that the key pair is set so that the instance can be accessed with SSH as this cannot be changed after creation.



For networking within this cluster a new VPC will be created so that all components within the cluster can talk to each other but will not be accessed by outside components not part of the system.


Networking

Configure the VPC for your container instances to use. A VPC is an isolated portion of the AWS cloud populated by AWS objects, such as Amazon EC2 instances. You can choose an existing VPC, or create a new one with this wizard.



VPC  


CIDR block 

Subnet 1  

Subnet 2 

[+ Add more subnets.](#)

Security group  

Security group inbound rules 

CIDR block


Port range

Protocol

Create the cluster with these settings.

4. Create an application load balancer for our application.

Before moving to the next step, we need to create a load balancer so that we can distribute the incoming traffic across the tasks that will run in our ECS service. We can do this in the **EC2 Console** and select **Load Balancers** in the side menu. Click 'Create Load Balancer' and select 'HTTP HTTPS' Application Load Balancer

Create Load Balancer 


Filter by tags and attributes or search by keyword << 1 to 2 of 2 >>

Name	DNS name	State	VPC ID	Availability Zones	Type	Created At	Monitoring
aws-ecs-clouda3-loadbalance	aws-ecs-clouda3-loadbalanc...	Active	vpc-0d04fed61e074aa57	ap-southeast-2b, ap-so...	application	June 2, 2021 at 2:49:38 PM ...	
awseb-AWSEB-17MU5ALZNIIBBS	awseb-AWSEB-17MU5ALZN...	Active	vpc-fb11099c	ap-southeast-2b, ap-so...	application	May 30, 2021 at 7:12:44 PM...	

Select load balancer type

Elastic Load Balancing supports four types of load balancers. Apply [Learn more about which load balancer is right for you](#)

Application Load Balancer



[Create](#)

Choose an Application Load Balancer when you need a flexible feature set for your web applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers.

[Learn more >](#)

Ensure you set a meaningful name to the load balancer. The most important step of creating this load balancer is to ensure that the Availability zone VPC is set to the same one that we previously made when creating the cluster so they can talk to each-other within the VPC.

Step 1: Configure Load Balancer

Basic Configuration

To configure your load balancer, provide a name, select a scheme, specify one or more listeners, and select a network. The default configuration is an Internet-facing load t

Name ⓘ

Scheme ⓘ ☒ Internet-facing
☐ Internal

IP address type ⓘ

Listeners

A listener is a process that checks for connection requests, using the protocol and port that you configured.

Load Balancer Protocol	Load Balancer Port
<input type="text" value="HTTP"/>	<input type="text" value="80"/>
<input type="button" value="Add listener"/>	

Step 1: Configure Load Balancer

Availability Zones

Specify the Availability Zones to enable for your load balancer. The load balancer routes traffic to the targets in these Availa
balancer.

VPC ⓘ

Availability Zones

☒ ap-southeast-2a
IPv4 address ⓘ Assigned by AWS

☒ ap-southeast-2b
IPv4 address ⓘ Assigned by AWS

Click ‘Next’ and ‘Next’ again to ‘Configure Security Groups’

Set the security group ID to the same one created for the EC2 Service.

Step 3: Configure Security Groups

A security group is a set of firewall rules that control the traffic to your load balancer. On this page, you can add rules to allow specific traffic to reach your load balancer. First,

Assign a security group ☐ Create a new security group
☒ Select an existing security group

Security Group ID	Name
<input type="checkbox"/> sg-03f0f3a30bdd727c8	default
<input checked="" type="checkbox"/> sg-01a1af4d72d33a108	EC2ContainerService-ecs-clouda3-cluster-EcsSecurityGroup-1P8CAXGIU4S1G

Click ‘Next’ to the ‘Configure Routing’ screen. Give a name to the target group to create one and set the port to 5000, as this is the port we are hosting the application on.

Step 4: Configure Routing

Your load balancer routes requests to the targets in this target group using the protocol and port that you specify or edit or add listeners after the load balancer is created.

Target group

Target group ⓘ New target group

Name ⓘ aws-ecs-target

Target type

- ☒ Instance
- ☐ IP
- ☐ Lambda function

Protocol ⓘ HTTP

Port ⓘ 5000

Protocol version ⓘ

- ☒ HTTP1
Send requests to targets using HTTP/1.1. Supported when the request uses HTTP/2.
- ☐ HTTP2
Send requests to targets using HTTP/2. Supported when the request protocol-specific features are not available.
- ☐ gRPC
Send requests to targets using gRPC. Supported when the request protocol

Health checks

Protocol ⓘ HTTP

Path ⓘ /

► Advanced health check settings

Click 'Next' to 'Register Targets' select the target and press *Add to registered*.

Step 5: Register Targets

Register targets with your target group. If you register a target in an enabled Availability Zone, the load balancer starts routing requests to the targets as soon as the registration process completes and the target passes the initial health checks.

Registered targets

To deregister instances, select one or more registered instances and then click Remove.

Remove

<input type="checkbox"/>	Instance	Name	Port	State	Security groups	Zone
<input checked="" type="checkbox"/>	i-08d8dd115ea4b9d3d	ECS Instance - EC2Container...	5000	running	EC2ContainerService-ecs-clouda3-cluster-EcsSecurityGroup-1...	ap-southeast-2a

Instances

To register additional instances, select one or more running instances, specify a port, and then click Add. The default port is the port specified for the target group. If the instance is already registered on the specified port, you must specify a different port.

Add to registered on port 5000

Search Instances

<input type="checkbox"/>	Instance	Name	State	Security groups	Zone	Subnet ID	Subnet CIDR
<input checked="" type="checkbox"/>	i-08d8dd115ea4b9d3d	ECS Instance - EC2Con...	running	EC2ContainerService-ec...	ap-southeast-2a	subnet-0cea763f53df36d5c	10.0.0.0/24

5. Create the ECS Cluster Service:

Now that we have a cluster, you can click the created cluster to add a service to it.

Services | Tasks | ECS Instances | Metrics | Scheduled Tasks | Tags | Capacity Providers

Create | Update | Delete | Actions

Last updated on June 9, 2021 12:36:23 PM (0m ago)

Filter in this page | Launch type: ALL | Service type: ALL

<input type="checkbox"/>	Service Name	Status	Service type	Task Definition	Desired tasks	Running tasks	Launch type	Platform version
<input type="checkbox"/>	aws-ecs-clouda3-service	ACTIVE	REPLICA	aws-ecs-clouda3-7	1	1	EC2	--

The service allows us to better manage our tasks inside the cluster, specifically how many copies of the task we will run.

Click 'Create' and set the launch type to EC2, ensure the task definition is the one previously created above and the cluster is the previously created cluster. Set the service type to REPLICa and for this application we will be using 1 task.

of tasks in your service.

Launch type ☐ FARGATE ☒ EC2 ☐ EXTERNAL ⓘ

[Switch to capacity provider strategy](#) ⓘ

Task Definition Family ⓘ Enter a value

Revision

Cluster ⓘ

Service name ⓘ

Service type* ☒ REPLICa ☐ DAEMON ⓘ

Number of tasks ⓘ

Minimum healthy percent ⓘ

Maximum percent ⓘ

Deployment circuit breaker ⓘ

Deployments

Choose a deployment option for the service

Click 'Next' to set the load balancing settings of the application to 'Application Load Balancer' and select the load balancer we created in the previous step. Click 'Add to load balancer' set the 'Production listener port' to port 80 and set the 'target group name' to the target previously created. Click 'Next step' then click 'Create service'.

6. Set the EC2 Instance Security settings to the following:

Inbound Rules: Note that 'All traffic' is set to the security group of the EC2 instance itself.

Inbound rules				
Inbound rules (7)				
Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	0.0.0.0/0	-
HTTP	TCP	80	:::0	-
All traffic	All	All	sg-01a1af4d72d33a108 / EC2ContainerService-ecs-conda3-cluster-EcsSecurityGroup-1P8CAXGIU451G	-
SSH	TCP	22	0.0.0.0/0	-
SSH	TCP	22	:::0	-
HTTPS	TCP	443	0.0.0.0/0	-
HTTPS	TCP	443	:::0	-

Outbound Rules:

Inbound rules

Outbound rules

Tags

Outbound rules (1)

Edit outbound rules

Type	Protocol	Port range	Destination	Description - optional
All traffic	All	All	0.0.0.0/0	-

7. Set the Domain name for the admin site in Route 53

See Section n-3a: Route 53 (*Routing traffic – Admin site*).

8. **Ensure the Dockerfile is created before moving onto the steps below:** As this admin ECS site uses a Docker Image to create the application, a Dockerfile of how to set up the environment before launching the application is needed and should look exactly like below.

Dockerfile

```
# get base image
FROM python:3.7

RUN apt-get update
RUN apt-get install -y gunicorn
RUN apt-get install -y python-gevent

# set ENV variables
ENV FLASK_ENV=production

ARG AWS_ACCESS_KEY_ID
ARG AWS_SECRET_ACCESS_KEY
ARG AWS_DEFAULT_REGION

ARG API_GATEWAY_KEY

ENV AWS_ACCESS_KEY_ID $AWS_ACCESS_KEY_ID
ENV AWS_SECRET_ACCESS_KEY $AWS_SECRET_ACCESS_KEY
ENV AWS_DEFAULT_REGION $AWS_DEFAULT_REGION
ENV API_GATEWAY_KEY $API_GATEWAY_KEY

ADD ./requirements.txt ./requirements.txt

RUN pip install -r ./requirements.txt

# copy files
COPY . /admin-project
WORKDIR /admin-project

# start gunicorn web server
CMD ["gunicorn", "-b", "0.0.0.0:5000", "application:application", "--workers=5"]
```

This dockerfile ensures our environment contains all the necessary components before running the application and ensure that each time we deploy, the deployment will run exactly the same. It sets all the environment variables of the environment, copies the admin project folder and launches the application using gunicorn on port 5000.

To test the application image locally, a docker-compose.yml file must be created exactly as the one below, which will build and run the image locally so that testing of the image can be done without deploying the application. You will need to set the environment variables in your terminal before composing this locally:

Docker-compose.yml

```
version: "3.1"

services:
  aws-clouda3-ecs:
    build:
      context: ./
      dockerfile: ./Dockerfile
    args:
      - AWS_ACCESS_KEY_ID=${AWS_ACCESS_KEY_ID}
      - AWS_SECRET_ACCESS_KEY=${AWS_SECRET_ACCESS_KEY}
      - AWS_DEFAULT_REGION=${AWS_DEFAULT_REGION}
      - API_GATEWAY_KEY=${API_GATEWAY_KEY}
    container_name: aws_clouda3_ecs
    ports:
      - 5000:5000
    environment:
      - AWS_ACCESS_KEY_ID=${AWS_ACCESS_KEY_ID}
      - AWS_SECRET_ACCESS_KEY=${AWS_SECRET_ACCESS_KEY}
      - AWS_DEFAULT_REGION=${AWS_DEFAULT_REGION}
      - API_GATEWAY_KEY=${API_GATEWAY_KEY}
    network_mode: bridge
```

9. Deploying with GitHub Actions

To deploy the application automatically, you must first create an 'aws-task-definition.json' file which contains the task definition information copied from the task json on AWS. Click the task definition created in the previous steps and then click the latest revision.

Task Definition Name : aws-ecs-clouda3

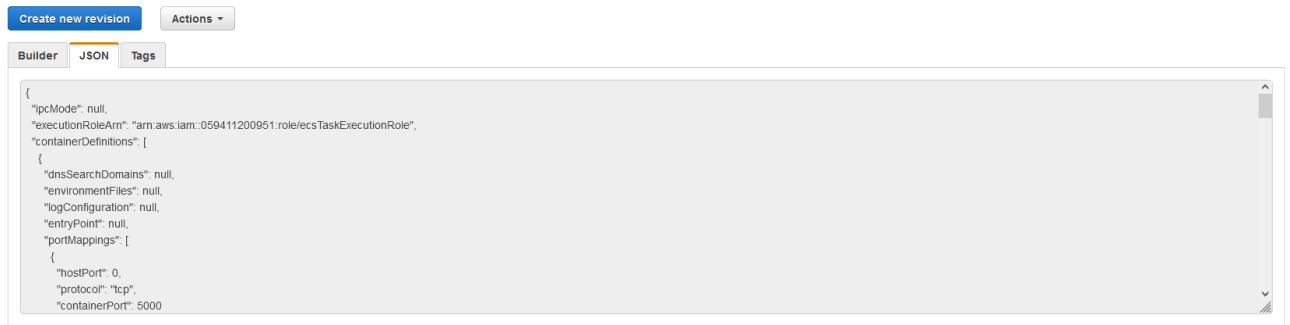
Select a revision for more details

Create new revisionActions ▾

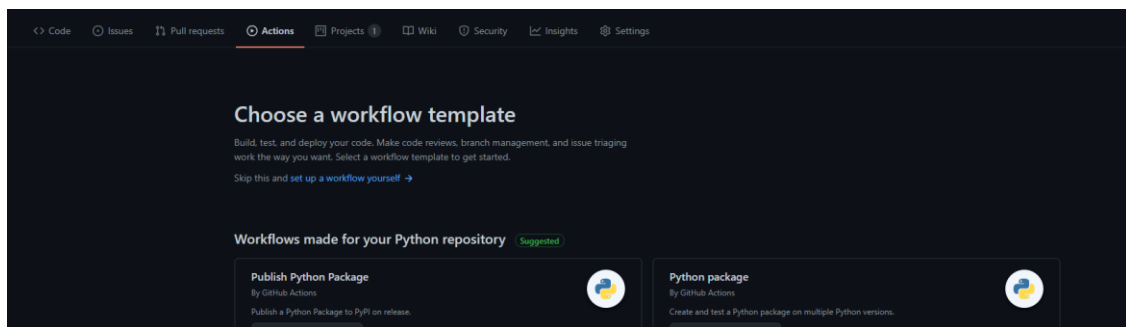
Status: Active Inactive

<input type="checkbox"/>	Task Definition Name : Revision	Status
<input type="checkbox"/>	aws-ecs-clouda3:7	Active

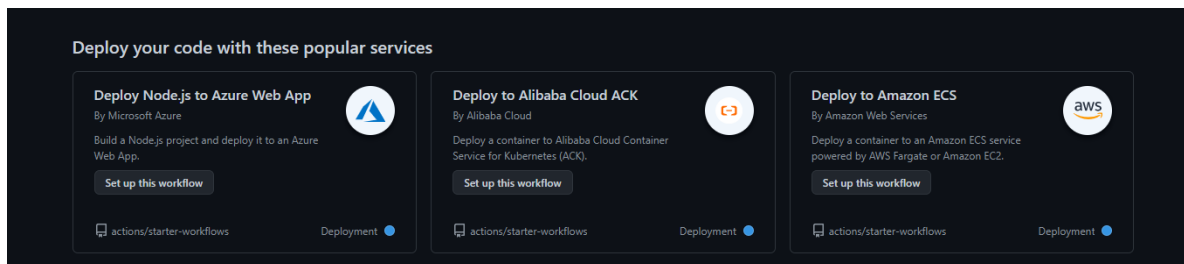
Click the JSON tab and copy the information here into the 'aws-task-definition.json' file:



To create the GitHub Actions workflow, click 'Actions' on GitHub



And scroll down to the Deploy to Amazon ECS service.



Click 'set up this workflow' so that GitHub will create the boilerplate code for the workflow in an aws.yml file which we will configure.

Below release and types, add 'workflow_dispatch' so that the deployment can be run without publishing a version. Give the workflow a name 'Deploy to Amazon ECS':

```

on:
  release:
    types: [created]
    workflow_dispatch:

name: Deploy to Amazon ECS

jobs:
  deploy:
    name: Deploy
    runs-on: ubuntu-latest
    environment: production

    steps:
      - name: Checkout
        uses: actions/checkout@v2

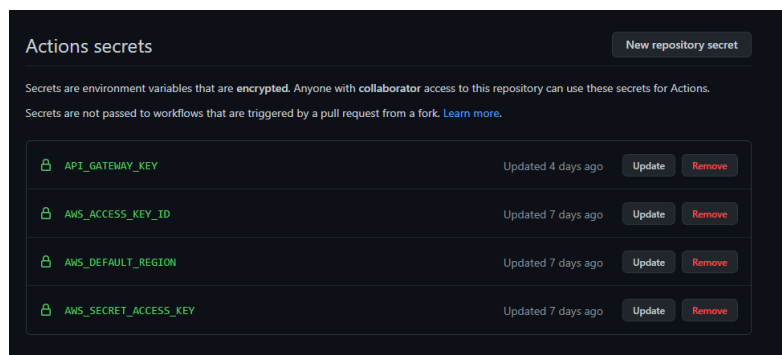
      - name: Configure AWS credentials
        uses: aws-actions/configure-aws-credentials@v1
        with:
          aws-access-key-id: ${ secrets.AWS_ACCESS_KEY_ID }
          aws-secret-access-key: ${ secrets.AWS_SECRET_ACCESS_KEY }
          aws-region: ${ secrets.AWS_DEFAULT_REGION }

      - name: Login to Amazon ECR
        id: login-ecr
        uses: aws-actions/amazon-ecr-login@v1

      - name: Build, tag, and push image to Amazon ECR

```

To set the environment variables for the GitHub deployment, we must set some 'Secrets' in the GitHub settings tab: Set the following secrets to the proper values.



In the aws.yml file set the ECR_REPOSITORY name to the name of the ECR repository created previously, then enter the docker build commands to build the docker file with the environment variables like below.

```

- name: Login to Amazon ECR
  id: login-ecr
  uses: aws-actions/amazon-ecr-login@v1

- name: Build, tag, and push image to Amazon ECR
  id: build-image
  working-directory: admin-project
  env:
    ECR_REGISTRY: ${ steps.login-ecr.outputs.registry }
    ECR_REPOSITORY: aws-ecs-clouda3
    IMAGE_TAG: ${ github.sha }
  run: |
    # Build a docker container and
    # push it to ECR so that it can
    # be deployed to ECS.
    docker build --build-arg AWS_ACCESS_KEY_ID=${ secrets.AWS_ACCESS_KEY_ID } \
      --build-arg AWS_SECRET_ACCESS_KEY=${ secrets.AWS_SECRET_ACCESS_KEY } \
      --build-arg AWS_DEFAULT_REGION=${ secrets.AWS_DEFAULT_REGION } \
      --build-arg API_GATEWAY_KEY=${ secrets.API_GATEWAY_KEY } \
      -t $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG .
    docker push $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG
    echo "::set-output name=image::$ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG"

```

Finally, set the container-name, location of the aws-task-definition.json file, the aws service and cluster names to the ones made on aws in the aws.yml file. As this particular project was a subfolder of the main project the aws-task-definition.json file is in the admin-project sub folder.

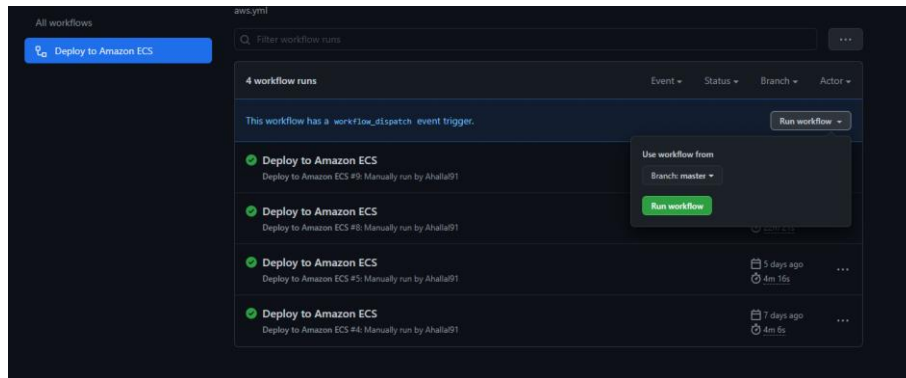
Push these files to the git repository, then click 'Actions' -> 'Deploy to Amazon ECS' -> 'Run Workflow' -> 'Run workflow'

```

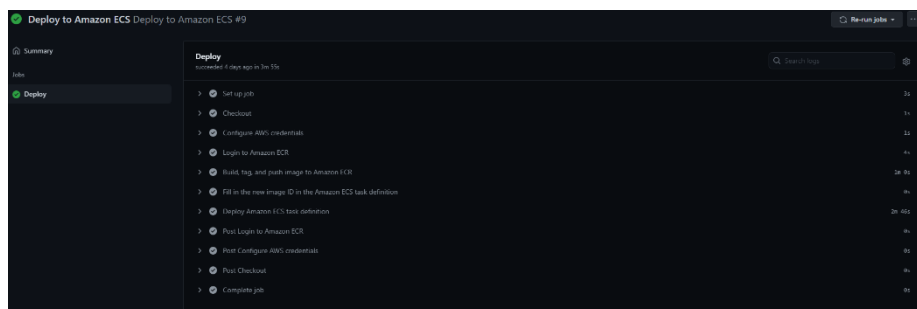
- name: Fill in the new image ID in the Amazon ECS task definition
  id: task-def
  uses: aws-actions/amazon-ecs-render-task-definition@v1
  with:
    task-definition: admin-project/aws-task-definition.json
    container-name: aws-ecs-clouda3
    image: ${ steps.build-image.outputs.image }

- name: Deploy Amazon ECS task definition
  uses: aws-actions/amazon-ecs-deploy-task-definition@v1
  with:
    task-definition: ${ steps.task-def.outputs.task-definition }
    service: aws-ecs-clouda3-service
    cluster: ecs-clouda3-cluster
    wait-for-service-stability: true

```



You can then view the deployment information by clicking the Deployment task once it sets to pending:

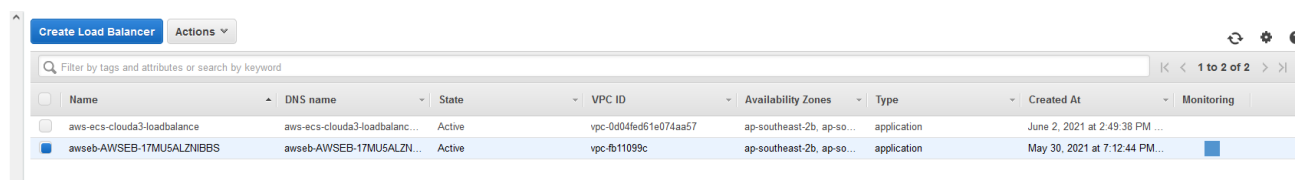


After this deployment step, the admin flask project should be uploaded to admin.cloudfitness.click and should be viewable and working.

10. Add SSL / HTTPS to the admin site.

Before moving onto the next steps, please follow the guide in Section 6: Certificate Manager, to create an AWS certificate to use with HTTPS/SSL, if it has not already been creating.

After this step, you can now create HTTPS redirect in the **load balancer** of the ec2 instance which was previously accessed before when creating the load balancer initially.



Select the load balancer which was used for the ec2 instance, then select 'Listeners' and create a HTTPS redirect to the target we previously created. Set the SSL certificate to the ARN of the one previously created.

Load balancer: **aws-ecs-clouda3-loadbalance**

Description **Listeners** Monitoring Integrated services Tags

Listeners listen for connection requests using their protocol and port. You can add, remove, or update listeners and listener rules.

To view and edit listener attributes, select the listener and choose Edit.

Add listener Edit Delete

<input type="checkbox"/>	Listener ID	Security policy	SSL Certificate	Rules
<input type="checkbox"/>	HTTP : 80 arn...93bc3ffa903b8897 ▾	N/A	N/A	Default: redirecting to HTTPS://#{host}:443/#{path}?#{query} View/edit rules
<input type="checkbox"/>	HTTPS : 443 arn...b42f78cf5142828d ▾	ELBSecurityPolicy-2016-08	Default: 2d47805f-3697-4336-b434-89dc91a8f7b (ACM) View/edit certificates	Default: forwarding to aws-ec2-clouda3-target View/edit rules

aws-ecs-clouda3-loadbalance | **HTTPS:443** (1 rules)

► Rule limits for condition values, wildcards, and total rules.

last **HTTPS 443: default action**
This rule cannot be moved or deleted

IF
✓ Requests otherwise not routed

THEN
Forward to
[aws-ec2-clouda3-target](#) : 1 (100%)
Group-level stickiness: Off

Finally, to ensure that users are always redirected to use HTTPS, redirect the HTTP port 80 to HTTPS.

aws-ecs-clouda3-loadbalance | **HTTP:80** (1 rules)

► Rule limits for condition values, wildcards, and total rules.

last **HTTP 80: default action**
This rule cannot be moved or deleted

IF
✓ Requests otherwise not routed

THEN
Redirect to [https://#{host}:443/#{path}?#{query}](#)
Status code: HTTP_301

Now the admin.cloudfitness.click website will always redirect the user to use HTTPS, and should now be fully functional.

Section 2: Main Site (*Elastic Beanstalk / CodePipeline*) [3] [4] [5]

Before beginning this step of development, a functioning cloud fitness flask application is needed to upload to the code-pipeline.

1. Create the Elastic Beanstalk .ebextensions configuration files.

Before pushing any code to codepipeline, we first must ensure that the environment we set up on Elastic beanstalk will be the correct configuration and launch the flask website correctly. Furthermore, this project creates the tables in the dynamo DB database, so we also need to add this code to run before the launch. Create a folder in the app directory of the flask site called .ebextensions.

2. Creating the python.config

Add a file called python.config to this folder. We will first add some container commands which run just like normal command line interface commands in the container but automatically execute before the flask app deploys. This container command creates the database tables if they do not already exist.

```
container_commands:
  command1:
    command: cd /home/ec2-user && python3 -m venv venv && source
venv/bin/activate && pip3 install boto3 && export AWS_DEFAULT_REGION=ap-
southeast-2 && python3 user_table.py && python3 profile_table.py && python3
comment_table.py && python3 exercise_table.py && deactivate
```

To ensure that the container commands run correctly they should be formatted as follows. It should have the heading **container_commands:** to signify the start of all container commands. The name for this command is **command1:** and the **command:** signifies the command to execute. Each new line should be tabbed over 1 but the command line should have no tabs/new lines in it.

The next step to configure the python.config file is to set the option_settings of the elastic beanstalk environment to know the path of the launch python file for the flask website.

```
option_settings:
  "aws:elasticbeanstalk:container:python":
    WSGIPath: application:application
```

The final format of the file should look similar to this.

```
container_commands:
  command1:
    command: cd /home/ec2-user && python3 -m venv venv && source venv/bin/activate && pip3 install boto3 && export AWS_DEFAULT_REGION=ap-southeast-2 && python3 user_tabl

option_settings:
  "aws:elasticbeanstalk:container:python":
    WSGIPath: application:application
```

Where the commands for command1 are written in a single line.

Creating the securelistener-alb.config

Please refer to this AWS developer guide for more specific configurations

<https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/configuring-https-elb.html>

We will now configure the https configurations for the elastic beanstalk flask site so that users can access the site with facebook login API.

Create a file in the .ebextensions folder called **securelistener-alb.config** this file name tells the environment this file will configure the application load balancer.

Enter the following details, port 443 directly relates to HTTPS

```
option_settings:
  aws:elbv2:listener:443:
    ListenerEnabled: 'true'
    Protocol: HTTPS
    SSLCertificateArns: arn:aws:acm:ap-southeast-2:059411200951:certificate/2d47805f-3697-4336-b434-89dc91a8ff7b
```

the SSL certificate should already be created from the admin site, please use the same certificate identifier, or if one has not been created, follow the steps in Section 6: Certificate Manager. Then set the SSLCertificatesArns to the ARN on the certificate.

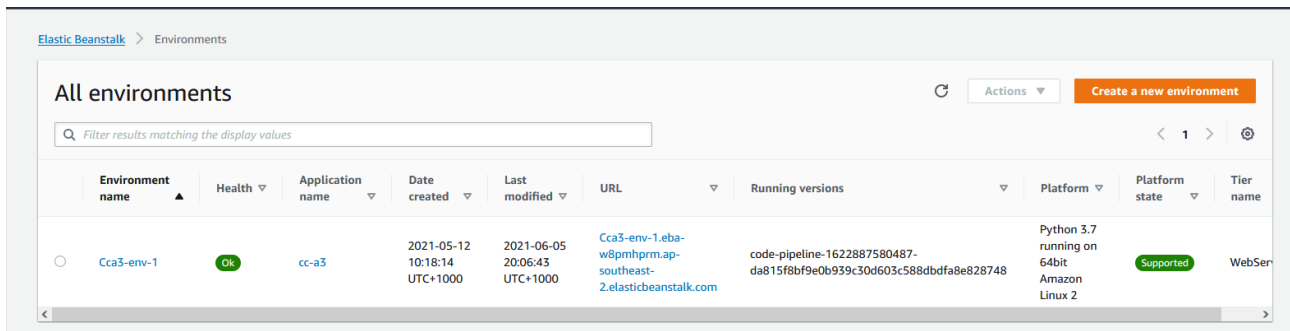
Finally to ensure that all our traffic is redirected to HTTPS add this code above the option_settings HTTPS redirect we just configured

```
Resources:
  AWSEBV2LoadBalancerListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      LoadBalancerArn:
        Ref: AWSEBV2LoadBalancer
      Port: 80
      Protocol: HTTP
      DefaultActions:
        - Type: redirect
          RedirectConfig:
            Host: "#{host}"
            Path: "/#{path}"
            Port: "443"
            Protocol: "HTTPS"
            Query: "#{query}"
            StatusCode: "HTTP_301"
```

This will redirect all HTTP port 80 traffic to port 443 automatically in our application load balancer.

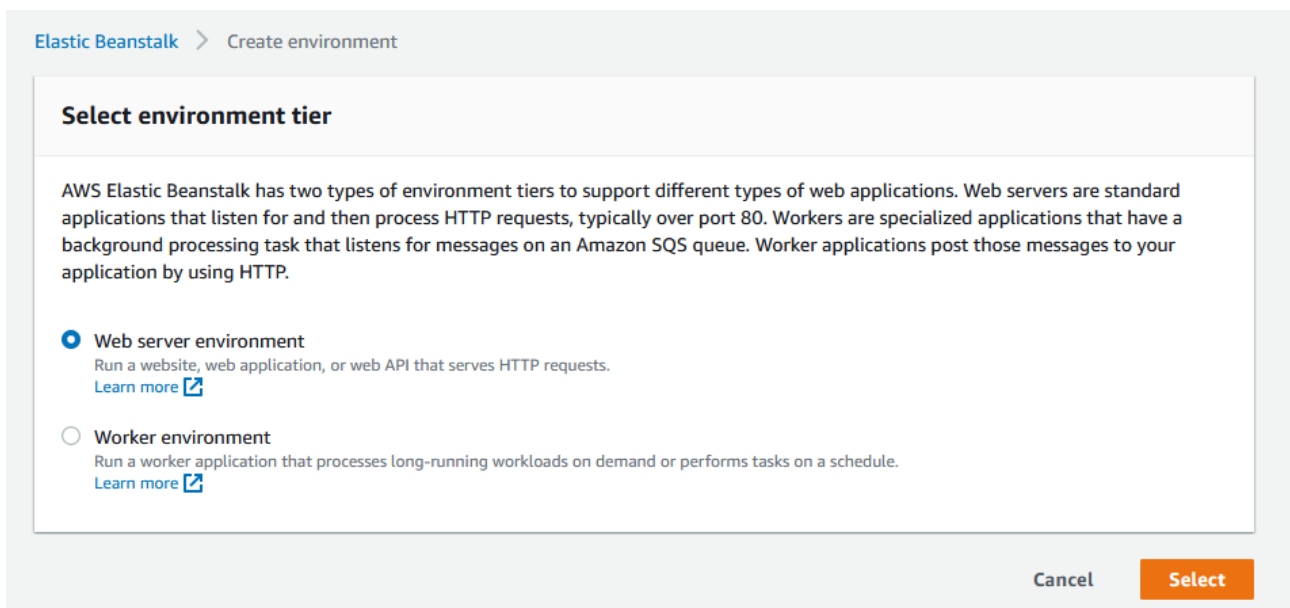
3. Create the Elastic Beanstalk application environment

Go to Elastic Beanstalk in AWS and click 'Create new environment'



Environment name	Health	Application name	Date created	Last modified	URL	Running versions	Platform	Platform state	Tier name
Cca3-env-1	Ok	cc-a3	2021-05-12 10:18:14 UTC+1000	2021-06-05 20:06:43 UTC+1000	Cca3-env-1.eba-w8pmhprm.ap-southeast-2.elasticbeanstalk.com	code-pipeline-1622887580487-da815f8bf9e0b939c30d603c588dbdfa8e828748	Python 3.7 running on 64bit Amazon Linux 2	Supported	WebSer

In the 'select environment tier' click 'Web server environment'



Select environment tier

AWS Elastic Beanstalk has two types of environment tiers to support different types of web applications. Web servers are standard applications that listen for and then process HTTP requests, typically over port 80. Workers are specialized applications that have a background processing task that listens for messages on an Amazon SQS queue. Worker applications post those messages to your application by using HTTP.

- ☒ **Web server environment**
Run a website, web application, or web API that serves HTTP requests.
[Learn more](#)
- ☐ **Worker environment**
Run a worker application that processes long-running workloads on demand or performs tasks on a schedule.
[Learn more](#)

[Cancel](#) [Select](#)

The next screen lets you set the application name and environment name. Set these to meaningful values.

Create a web server environment

Launch an environment with a sample application or your own code. By creating an environment, you allow AWS Elastic Beanstalk to manage AWS resources and permissions on your behalf. [Learn more](#)

Application information

Application name

application-name

Up to 100 Unicode characters, not including forward slash (/).

► Application tags (optional)

Environment information

Choose the name, subdomain, and description for your environment. These cannot be changed later.

Environment name

Applicationname-env

Domain

Leave blank for autogenerated value .ap-southeast-2.elasticbeanstalk.com

Check availability

Description

You will now need to upload the initial base code of the application as a zip file. Zip the whole flask application and select 'Upload your code', select 'Local file' and then upload the zip file project.

Application code

☐ Sample application
Get started right away with sample code.

☐ Existing version
Application versions that you have uploaded for application-name.

-- Choose a version --

☒ Upload your code
Upload a source bundle from your computer or copy one from Amazon S3.

Version label
Unique name for this version of your application code.

application-name-source

Source code origin
Maximum size 512 MB

☒ Local file

☐ Public S3 URL

Choose file

☐ No file uploaded

► Application code tags

Cancel Configure more options Create environment

Finally click 'Create environment'

4. Creating the CodePipeline for the website.

Codepipeline will allow us to automatically deploy code to our website in the event of changes, rather than having to re-zip the application each time and upload these zip files. Go to CodePipeline in AWS and click 'Create Pipeline', give the pipeline a name and click 'new service role' then click 'next'

Choose pipeline settings [Info](#)

Pipeline settings

Pipeline name
Enter the pipeline name. You cannot edit the pipeline name after it is created.
test-name
No more than 100 characters

Service role

☒ **New service role**
Create a service role in your account

☐ **Existing service role**
Choose an existing service role from your account

Role name
AWSCodePipelineServiceRole-ap-southeast-2-test-name
Type your service role name

☒ **Allow AWS CodePipeline to create a service role so it can be used with this new pipeline**

► **Advanced settings**

Cancel **Next**

Click 'GitHub (version 2)' as the service provider then connect the repository that holds the code for the flask application in the 'connection' and 'repository name' area then click 'master/main' branch and click 'next's

Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.
GitHub (Version 2)

New GitHub version 2 (app-based) action
To add a GitHub version 2 action in CodePipeline, you create a connection, which uses GitHub Apps to access your repository. Use the options below to choose an existing connection or create a new one. [Learn more](#)

Connection
Choose an existing connection that you have already configured, or create a new one and then return to this task.
arn:aws:codestar-connections:ap-southeast-2:059411200951:connection/34l or **Connect to GitHub**

Ready to connect
Your Github connection is ready for use.

Repository name
Choose a repository in your GitHub account.
Ahallal91/cloudcomputinga3
<account>/<repository-name>

Branch name
Choose a branch of the repository.
master

Change detection options
☒ **Start the pipeline on source code change**
Automatically starts your pipeline when a change occurs in the source code. If turned off, your pipeline only runs if you start it manually or on a schedule.

Output artifact format
Choose the output artifact format.

☒ **CodePipeline default**
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include git metadata about the repository.

☐ **Full clone**
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full git clone. Only supported for AWS CodeBuild actions.

In 'Add build stage' click 'skip build stage'

Add build stage [Info](#)

Build - optional

Build provider
This is the tool of your build project. Provide build artifact details like operating system, build spec file, and output file names.

[Cancel](#) [Previous](#) [Skip build stage](#) [Next](#)

In 'add deploy stage' select 'AWS Elastic Beanstalk' as the deploy provider and set the application name and environment name to the one previously created in Elastic Beanstalk

Add deploy stage [Info](#)

You cannot skip this stage
Pipelines must have at least two stages. Your second stage must be either a build or deployment stage. Choose a provider for either the build stage or deployment stage.

Deploy

Deploy provider
Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

Region

Application name
Choose an application that you have already created in the AWS Elastic Beanstalk console. Or create an application in the AWS Elastic Beanstalk console and then return to this task.

Environment name
Choose an environment that you have already created in the AWS Elastic Beanstalk console. Or create an environment in the AWS Elastic Beanstalk console and then return to this task.

[Cancel](#) [Previous](#) [Next](#)

Click 'next' then click 'Create pipeline' after reviewing the details. This pipeline will now publish to Elastic Beanstalk when GitHub master branch is pushed to or you manually deploy the application.

- 5. Connecting Route 53 Domain to the application.**
See Section n-3b: Route 53 (*Routing traffic – Main site*).

Section 3: Amazon Lambda

1. Create a lambda function

Navigate to AWS Lambda and click *Create function*. Select “*Author from scratch*” as the function type. Give the function a name and select *Python 3.8* as the runtime.

Lambda > Functions > Create function

Create function [Info](#)

Choose one of the following options to create your function.

Author from scratch ☒
Start with a simple Hello World example.

Use a blueprint ☐
Build a Lambda application from sample code and configuration presets for common use cases.

Container image ☐
Select a container image to deploy for your function.

Browse serverless app repository ☐
Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

Function name
Enter a name that describes the purpose of your function.

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

To use AWS Lambda with DynamoDB, the IAM policy *AmazonDynamoDBFullAccess* needs to be attached to the function.

For now, under permissions, select *Create a new role with basic Lambda permissions*. This will create a new IAM role with only the *AWSLambdaBasicExecution* policy attached. Leave all other options as they are and click *Create function*.

Permissions [Info](#)

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

- ☒ Create a new role with basic Lambda permissions
- ☐ Use an existing role
- ☐ Create a new role from AWS policy templates

i Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Lambda will create an execution role named `example-role-halpmavq`, with permission to upload logs to Amazon CloudWatch Logs.

► Advanced settings


Cancel

Create function


2. Add full DynamoDB access to the lambda's permissions

Select the newly created lambda function from the *Functions* section of AWS Lambda:

Lambda > Functions

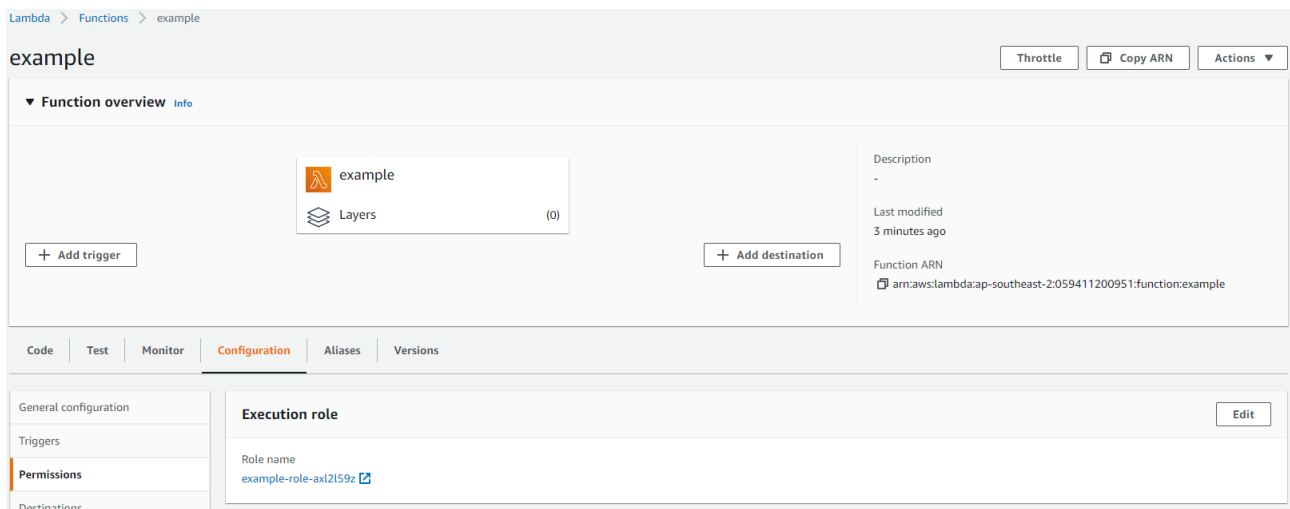
Functions (2) Last fetched in 0 seconds  **Actions** ▼ **Create function**

< 1 >



	Function name ▼	Description	Package type ▼	Runtime ▼	Code size ▼	Last modified ▼
<input type="radio"/>	workouts		Zip	Python 3.8	1.5 kB	6 days ago
<input type="radio"/>	example		Zip	Python 3.8	299.0 byte	7 minutes ago

Navigate to *Configuration* → *Permissions* to access the role attached to the function.



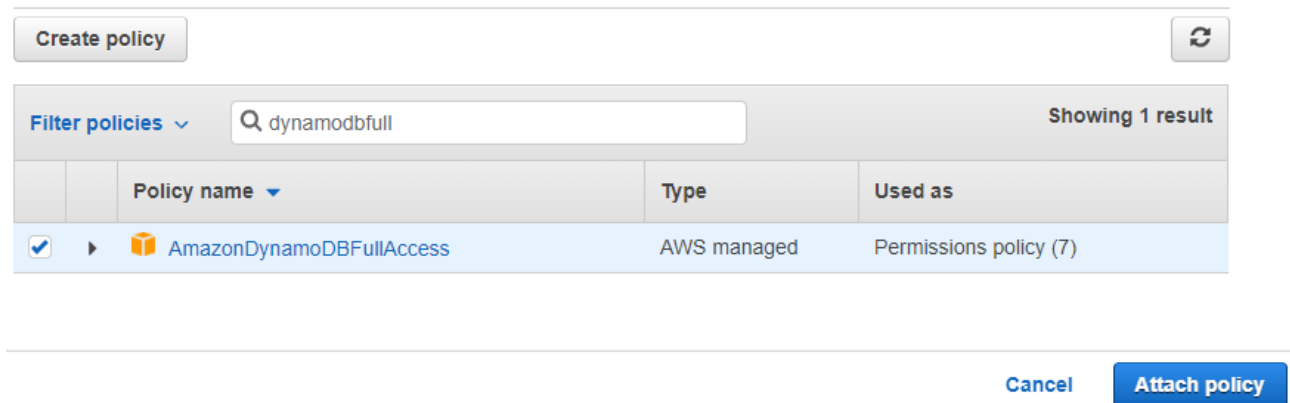
Select the role to be redirected to IAM. Click *Attach policies*:



And search for *AmazonDynamoDBFullAccess*. Click the checkbox on the left and then click *Attach policy*. The Lambda function now has full access to DynamoDB.

Add permissions to example-role-axl2l59z

Attach Permissions



3. Edit the lambda function

Select the newly created function again and edit the code as needed. A sample of the lambda function used to control our REST API is shown below:

```
125 if len(response['items']) != 0:
126     response = response['items'][0]
127 else:
128     response = "No exercise with type '" + path_parameters["type"] + "' and name '" + path_parameters["name"] + "' was found."
129 return json.dumps(response, indent=2, cls=DecimalEncoder)
130
131
132 def lambda_handler(event, context):
133     responseObject = {}
134     responseObject['statusCode'] = 200
135     responseObject['headers'] = {}
136     responseObject['headers']['Content-Type'] = 'application/json'
137     # responseObject['body'] = json.dumps(event, indent=2, cls=DecimalEncoder)
138     # return responseObject
139
140     resources = {
141         "/": root,
142         "/exercises": get_all_exercises,
143         "/exercises/types": get_all_exercise_types,
144         "/exercises/{type}": get_exercises_by_type,
145         "/exercises/{type}/{name}": get_exercise_by_type_and_name,
146         "/exercises/approved": get_approved_exercises,
147         "/exercises/pending": get_pending_exercises
148     }
149
150     path_parameters = event["pathParameters"]
151
152     if event['httpMethod'] == 'GET':
153         responseObject['body'] = resources[event["resource"]](path_parameters)
154         return responseObject
155
156     if event['httpMethod'] == 'PUT':
157         return put_exercise_approval(event)
158
159
```

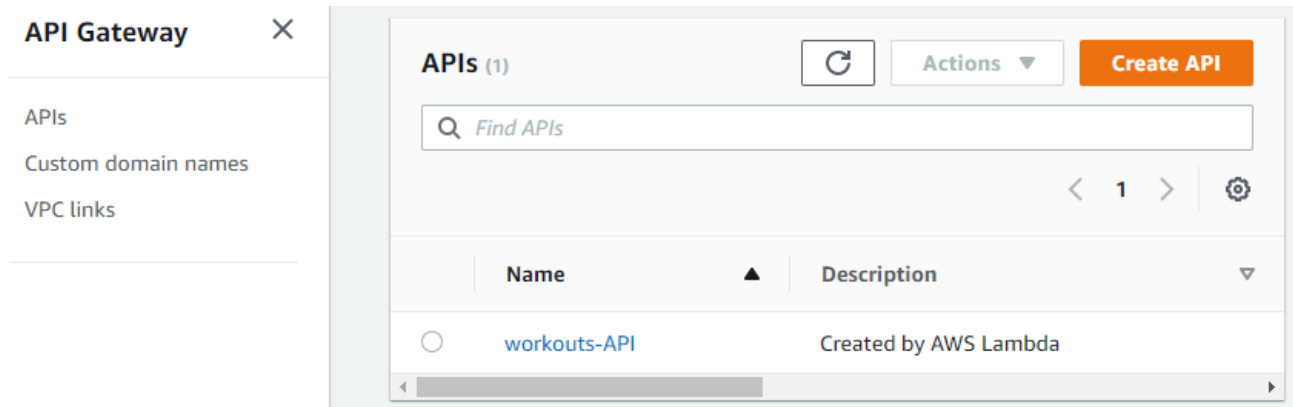
Whenever a change is made to the lambda function, click *Deploy* above the code editing section.

Note that the request information accessed by the lambda function is provided by API Gateway, which is handled in the next section.

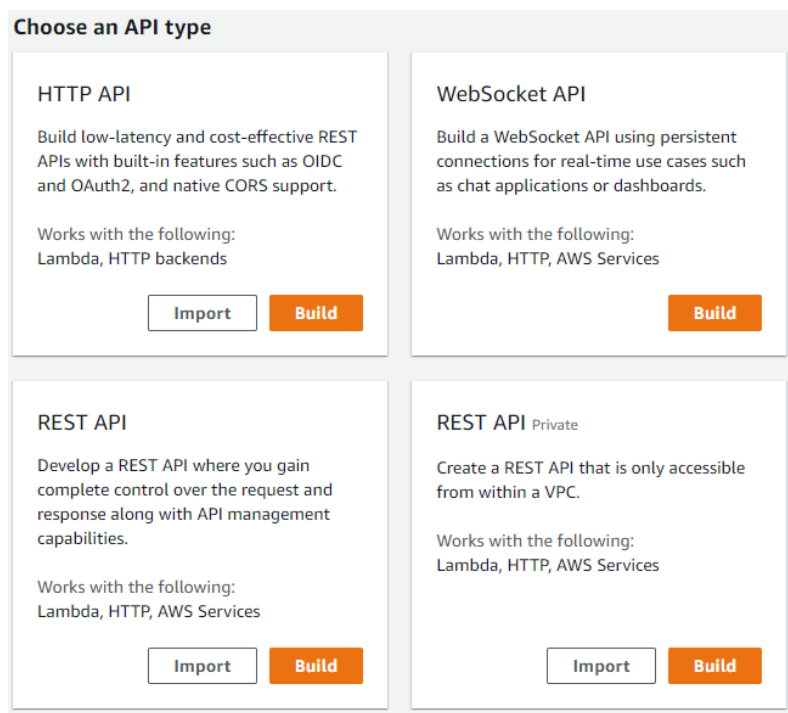
Section 4: API Gateway[6]

1. Create a new API

Navigate to AWS API Gateway. To create a new API, click the orange *Create API* button.



Select REST API (*not private*) and click *Build*.



Enter an API name and description. For our purposes, a regional endpoint is used, but an edge optimized API is more suitable for websites accessed globally. Click *Create API*.

Amazon API Gateway

APIs > Create

Show all hints ?

APIs

Custom Domain Names

VPC Links

Choose the protocol

Select whether you would like to create a REST API or a WebSocket API.

☒ REST
 ☐ WebSocket

Create new API

In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

☒ New API
 ☐ Clone from existing API
 ☐ Import from Swagger or Open API 3
 ☐ Example API

Settings

Choose a friendly name and description for your API.

API name*

example-API

Description

an example API

Endpoint Type

Regional

?

* Required

Create API

You will be redirected to the newly created API. If not, you can access the API from the AWS API Gateway landing page.

2. Registering a custom domain name

From herein, we will assume that a custom domain has been registered for the API Gateway following the instructions in Section n-1 (Route 53, *Domain name registration*).

To assign a custom domain name to the API, click *Custom Domain Names* in the left menu:

Amazon API Gateway

APIs > example-API (1jo6guwmi1) > Resources > / (snb6wll49f)

APIs

Custom Domain Names

VPC Links

Resources

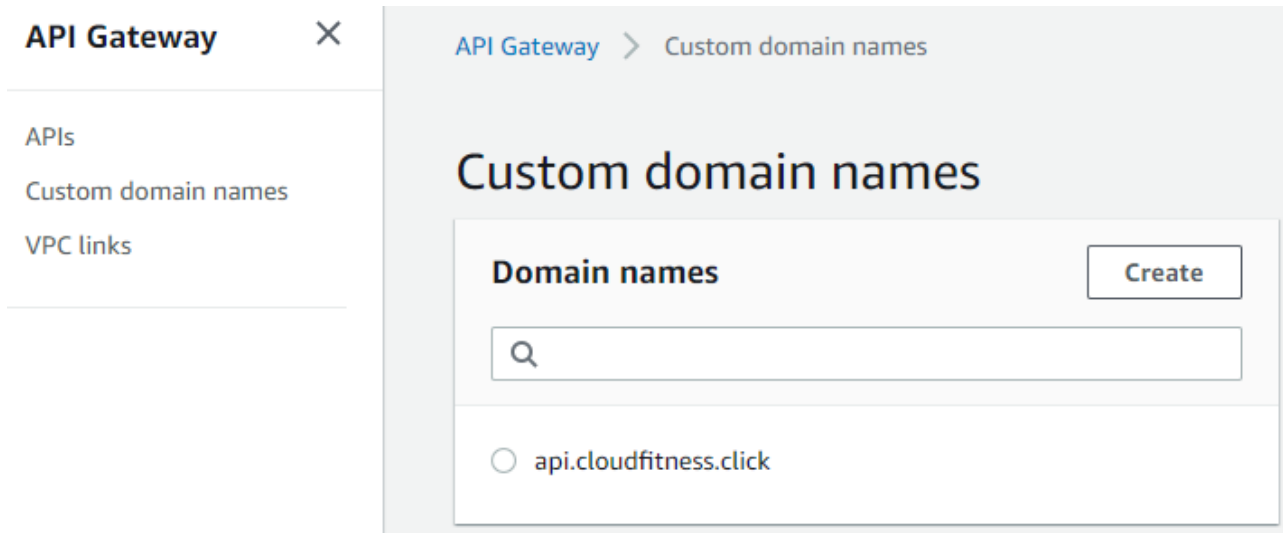
Actions

/ Methods

/

API: example-API

Resources




Register the domain name using the search box provided. The domain name entered here should correspond to the subdomain you wish to be routed to as configured in Route 53 (see Section n-3c, Route 53: Routing to the REST API). For our website, the custom domain name given is *api.cloudfitness.click*. This will forward HTTPS requests from the given domain to API Gateway, and that information will be forwarded to the appropriate lambda function as described in the next section.


3. Adding resources



Resources correspond to the URLs visited by users. The root resource */* corresponds to the base subdomain registered as a custom domain name, such as *api.cloudfitness.com/*. By default, no methods are available at this resource for newly created APIs. Resources are added in a tree structure.

Resources can either be named with an absolute name or a variable name. Variable names are enclosed in { curly braces } and allow users to visit endpoints with variable parameters. For example, our application has a variable resource under */exercises/{type}*, which will list all exercises of a certain type if the user replaces {type} with a valid workout type, e.g., */exercises/compound/*.

To add a resource, click the *Actions* dropdown menu and select *Create Resource*. Do not configure the resource as a proxy resource. Give the resource a name, which typically matches the value entered as a resource path. If the resource path should be a variable resource path, enclose the value in curly braces, i.e., "{example}" rather than "example". Leave *Enable API Gateway CORS* unchecked and click *Create Resource*.

Resources **Actions**  **New Child Resource**


Use this page to create a new child resource for your resource. 

Configure as  **proxy resource** ☐ 

Resource Name*

Resource Path*

You can add path parameters using brackets. For example, the resource path **{username}** represents a path parameter called 'username'. Configuring **/{proxy+}** as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to /foo. To handle requests to /, add a new ANY method on the / resource.

Enable API Gateway CORS ☐ 

* Required

[Cancel](#) [Create Resource](#)

4. Adding methods to resources

To add a method to a resource, click the *Actions* dropdown and select *Create Method*. A dropdown will appear underneath the resource name, where you can select HTTP methods such as GET, POST, PUT, and so on. Select GET.

APIs

Custom Domain Names


VPC Links

API: example-API

Resources

Stages

Authorizers

Resources **Actions**  **/ Methods**



RESOURCE ACTIONS

- Create Method
- Create Resource
- Enable CORS
- Edit Resource Documentation

API ACTIONS

- Deploy API
- Import API
- Edit API Documentation
- Delete API

Select *Lambda* function as the integration type, and tick *Use Lambda Proxy integration*. By selecting *Use Lambda Proxy Integration*, the entire HTTP request is passed to the lambda function, allowing all headers to be accessed in the lambda (see Section 3-3 (Amazon Lambda: Edit the lambda function) for an example usage). Select the appropriate region and the lambda function that should receive the request information and leave *Use Default Timeout* checked. Click *Save* and click *OK* when prompted to *Add Permission to Lambda Function*.

Resources **Actions**  / - GET - Setup 

▼ /

GET

Choose the integration point for your new method.

Integration type ☒ Lambda Function ⓘ

☐ HTTP ⓘ

☐ Mock ⓘ

☐ AWS Service ⓘ

☐ VPC Link ⓘ

Use Lambda Proxy integration ☒ ⓘ

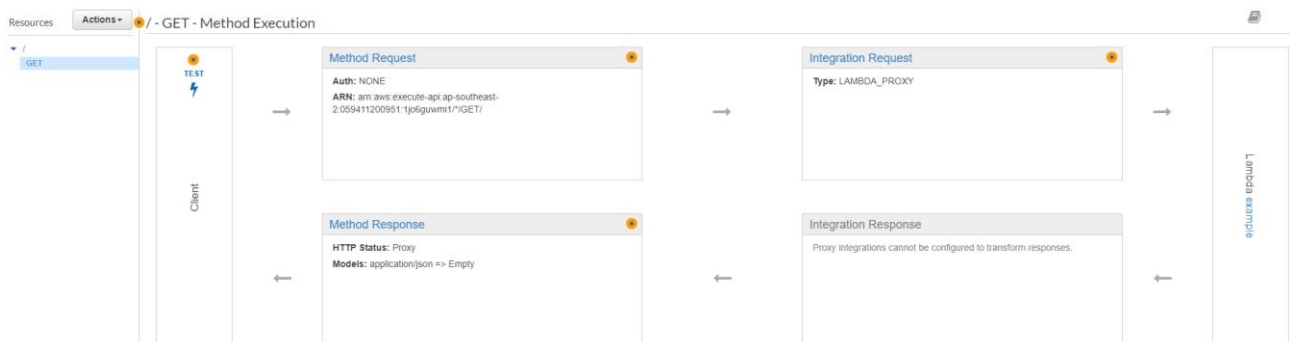
Lambda Region

Lambda Function

Use Default Timeout ☒ ⓘ

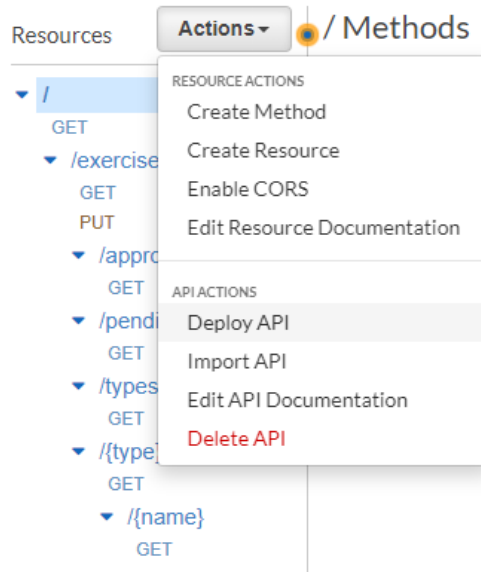
Save

The GET method under / will now contain a method execution pathway. Note the lambda we selected on the right side of the diagram.



5. Create and deploy the final API

To deploy the API, click the *Actions* dropdown and select *Deploy API*.



On the first deployment, a new deployment stage must be selected. On future deployments, this stage can be reused. Whenever a change is made to the API, ensure that the API is deployed.

Deploy API

Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

Deployment stage	<div>[New Stage]</div>
Stage name*	<div>example-deploy</div>
Stage description	<div>example</div>
Deployment description	<div>example</div>

Cancel

Deploy

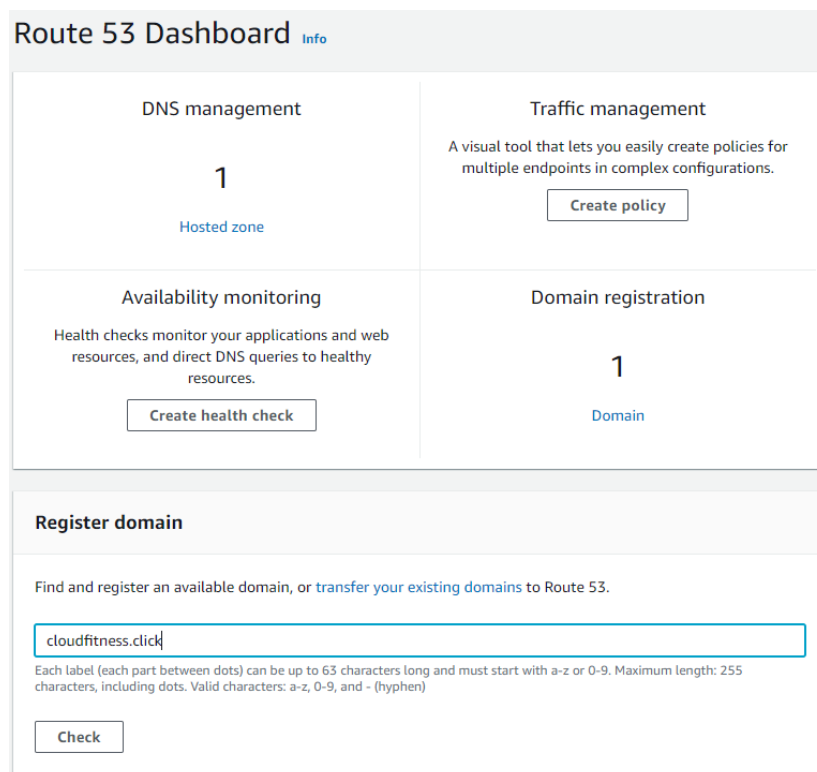
The final API Gateway setup for our application is shown below. Each resource and method are configured as described above. Note that the /exercises resource also contains a PUT method used via the admin site.

-
- ▼ /
 - GET
 - ▼ /exercises
 - GET
 - PUT
 - ▼ /approved
 - GET
 - ▼ /pending
 - GET
 - ▼ /types
 - GET
 - ▼ /{type}
 - GET
 - ▼ /{name}
 - GET

Section 5: Route 53

1. Domain name registration

Navigate to the Route 53 Dashboard and search for a domain name using the search bar contained under the section “Register domain”:



The screenshot shows the AWS Route 53 Dashboard. At the top, there's a header 'Route 53 Dashboard' with an 'Info' link. Below this, the dashboard is divided into four main sections: 'DNS management' (showing 1 Hosted zone), 'Traffic management' (with a 'Create policy' button), 'Availability monitoring' (with a 'Create health check' button), and 'Domain registration' (showing 1 Domain). Below these sections is a 'Register domain' section. It contains the text: 'Find and register an available domain, or [transfer your existing domains](#) to Route 53.' Below this is a search input field containing 'cloudfitness.click'. Under the input field, there is a small text block: 'Each label (each part between dots) can be up to 63 characters long and must start with a-z or 0-9. Maximum length: 255 characters, including dots. Valid characters: a-z, 0-9, and - (hyphen)'. At the bottom of the 'Register domain' section is a 'Check' button.

If the domain is available, purchase the domain and wait for DNS propagation to indicate that the purchased domain is useable. We recommend purchasing *.click* domains since they are the cheapest at \$5 AUD.

2. Setup HTTPS certification

For use with API gateway, the domain name must be usable under HTTPS. See **Step 10** (*Add SSL / HTTPS to the admin site*) of *Section 1: Admin Site* to see how this is done.

3. Routing traffic

Using the side panel, navigate to “Hosted zones”, which should list the domain registered above:

Route 53 > Hosted zones

Hosted zones (1) View details Edit Delete Create hosted zone

Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.

Filter hosted zones by property or value

	Domain name	Type	Created by	Record count	Description	Hosted zone ID
<input type="radio"/>	cloudfitness.click	Public	Route 53	7	HostedZone created by Route53 Regis...	Z01373421UFEMNYMI2KHS

Select the domain name, and then click *Create record*. A separate record is created for the admin site, the main site, and the REST API handled by API Gateway.

a. Routing the admin site

Enter “admin” under *Record name* to create a record for *admin.cloudfitness.click*. Select “A” as the record type. Check the *Alias* slider next to “Route traffic to” and select “Alias to Application and Classic Load Balancer”. Select the appropriate region and the application load balancer created in section 1. Leave all other options as they are.

Route 53 > Hosted zones > cloudfitness.click > Create record

Quick create record Info Switch to wizard Add another record

▼ Record 1 Delete

Record name Info .cloudfitness.click
 Valid characters: a-z, 0-9, ! " # \$ % & ' () * + , - . / : ; < = > ? @ [\] ^ _ ` { | } . ~

Record type Info

Route traffic to Info ☒ Alias

Routing policy Info

Evaluate target health ☒ Yes

Cancel Create records

b. Routing the main site

Leave the record name blank to create a record for *www.cloudfitness.click*. Select “A” as the record type. Check the *Alias* slider next to “Route traffic to” and select “Alias to Elastic Beanstalk Environment”. Select the appropriate region and the application load balancer created in section 1. Leave all other options as they are.

Route 53 > Hosted zones > cloudfitness.click > Create record

Quick create record [Info](#) [Switch to wizard](#) [Add another record](#)

▼ Record 1 [Delete](#)

Record name [Info](#)

 cloudfitness.click

Valid characters: a-z, 0-9, ! " # \$ % & ' () * + , - / : ; < = > ? @ [\] ^ _ ` { | } . ~

Record type [Info](#)

A – Routes traffic to an IPv4 address and so... ▼

Route traffic to [Info](#) ☒ Alias

Alias to Elastic Beanstalk environment ▼

Asia Pacific (Sydney) [ap-southeast-2] ▼

Q Cca3-env-1.eba-w8pmhprm.ap-southeast X

Routing policy [Info](#)

Simple routing ▼

Evaluate target health ☒ Yes

Cancel [Create records](#)

c. Routing to the REST API

Enter “api” under *Record name* to create a record for *api.cloudfitness.click*. **Ensure this is the same name as the custom domain name specified for the corresponding API in API gateway.** Otherwise, the API will not appear as a selectable endpoint.

Select “A” as the record type. Check the *Alias* slider next to “Route traffic to” and select “Alias to API Gateway API”.

Route 53 > Hosted zones > cloudfitness.click > Create record

Quick create record [Info](#) [Switch to wizard](#) [Add another record](#)

▼ Record 1 [Delete](#)

Record name [Info](#)

 cloudfitness.click

Valid characters: a-z, 0-9, ! " # \$ % & ' () * + , - / : ; < = > ? @ [\] ^ _ ` { | } . ~

Record type [Info](#)

A – Routes traffic to an IPv4 address and so... ▼

Route traffic to [Info](#) ☒ Alias

Alias to API Gateway API ▼

Asia Pacific (Sydney) [ap-southeast-2] ▼

Q d-qan33aeyb1.execute-api.ap-southeast-: X

Routing policy [Info](#)

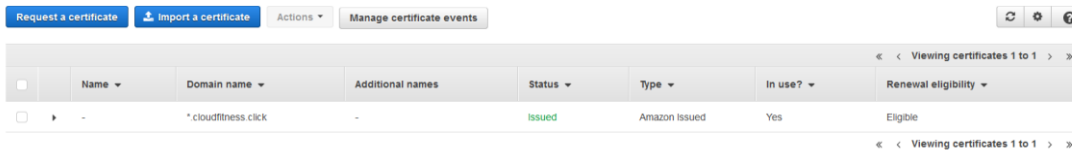
Simple routing ▼

Evaluate target health ☒ Yes

Cancel [Create records](#)

Section 6: Certificate Manager

To create a certificate that will allow the use of HTTPS/SSL on the sites go to 'Certificate Manager' in the aws console. Click 'request a certificate'.



Request a certificate	Import a certificate	Actions	Manage certificate events				
< < Viewing certificates 1 to 1 > >							
	Name	Domain name	Additional names	Status	Type	In use?	Renewal eligibility
<input type="checkbox"/>	-	*.cloudfitness.click	-	Issued	Amazon Issued	Yes	Eligible
< < Viewing certificates 1 to 1 > >							

Then click 'request a public certificate and click 'Request a certificate'

Request a certificate

Choose the type of certificate for ACM to provide.

- ☒ **Request a public certificate** - Request a public certificate from Amazon. By default, public certificates are trusted by browsers and operating systems. [Learn more.](#)
- ☐ **Request a private certificate** - No Private CAs available for issuance. [Learn more.](#)

Cancel Request a certificate

Fill in the domain name for the certificate as *.cloudfitness.click and click 'Next'. (this * before the domain name signifies that the certificate can be for any type of domain name [name].cloudfitness.click)

Add domain names



Type the fully qualified domain name of the site you want to secure with an SSL/TLS certificate (for example, www.example.com). Use an asterisk (*) to request a wildcard certificate to protect several sites in the same domain. For example: *.example.com protects www.example.com, site.example.com and images.example.com.

Domain name*

*At least one domain name is required

*.cloudfitness.click

Add another name to this certificate

You can add additional names to this certificate. For example, if you're requesting a certificate for "www.example.com", you might want to add the name "example.com" so that customers can reach your site by either name. [Learn more.](#)

Cancel Next

Then click 'DNS' validation. Click 'Next'.

Select validation method

Choose how AWS Certificate Manager (ACM) validates your certificate request. Before we issue your certificate, we need to validate that you own or control the domains for which you are requesting the certificate. ACM can validate ownership by using DNS or by sending email to the contact addresses of the domain owner.

- ☒ **DNS validation**
- Choose this option if you have or can obtain permission to modify the DNS configuration for the domains in your certificate request. [Learn more.](#)
- ☐ **Email validation**
- Choose this option if you do not have permission or cannot obtain permission to modify the DNS configuration for the domains in your certificate request. [Learn more.](#)

Cancel Previous Next

We will not add any tags so skip the 'Add tags' section and click 'Review'. Then in the validation section press 'Create a record in Route 53'. This adds the Certificate to route 53

automatically and validates the certificate. It should appear as a record of type CNAME in Route 53, so you can verify this.

<input type="checkbox"/>	_cbf3026d53e06343...	CNAME	Simple	-	_a5da253cb1f7ea090e88b7c70617f42b.xrchbtpdjs.acm-validations.aws.
--------------------------	----------------------	-------	--------	---	---

When adding HTTPS/SSL to applications you will need to copy the ARN from the certificate in the certificate manager to enable the use of the certificate.

Status

Status

Issued

Detailed status

The certificate was issued at 2021-06-01T04:14:12UTC

Domain	Validation status
*.cloudfitness.click	Success

[Export DNS configuration to a file](#) You can export all of the CNAME records to a file

Details

Type	Amazon Issued	Requested at	2021-06-01T04:13:47UTC
In use?	Yes	Issued at	2021-06-01T04:14:12UTC
Domain name	*.cloudfitness.click	Not before	2021-06-01T00:00:00UTC
Number of additional names	0	Not after	2022-06-30T23:59:59UTC
Identifier	2d47805f-3697-4336-b434-89dc91a8ff7b	Public key info	RSA 2048-bit
Serial number	0b:93:32:11:b2:96:8d:74:6f:56:a3:68:14:42:e2:2a	Signature algorithm	SHA256WITHRSA
Associated resources	arn:aws:elasticloadbalancing:ap-southeast-2:059411200951:loadbalancer/app/aws-ecs-clouda3-loadbalance/7ef5d61ea412ec26, arn:aws:elasticloadbalancing:ap-southeast-2:059411200951:loadbalancer/app/awseb-AWSEB-17MUSALZNIIBBS/2c125840430138a6, arn:aws:elasticloadbalancing:ap-southeast-2:798376113853:loadbalancer/app/prod-syd-1-cdtls-1-2-126/ddb0bcc3b07f1318, arn:aws:elasticloadbalancing:ap-southeast-2:798376113853:loadbalancer/app/prod-syd-1-cdtls-1-2-133/2a5060409389fce9, arn:aws:elasticloadbalancing:ap-southeast-2:798376113853:loadbalancer/app/prod-syd-1-cdtls-1-2-83/9ddea4aa4da91e34	ARN	arn:aws:acm:ap-southeast-2:059411200951:certificate/2d47805f-3697-4336-b434-89dc91a8ff7b
		Validation state	None

User Manual

Main site

Viewing exercises: Users can view exercises without being a registered user or by being registered. They can see a brief overview of each exercises via the home page:

'All life is an experiment. The more experiments you make, the better.' : Ralph Waldo Emerson



Bench press

Views: 57 Upload Date: 2021-06-05 20:01:04.913869



Romanian deadlift

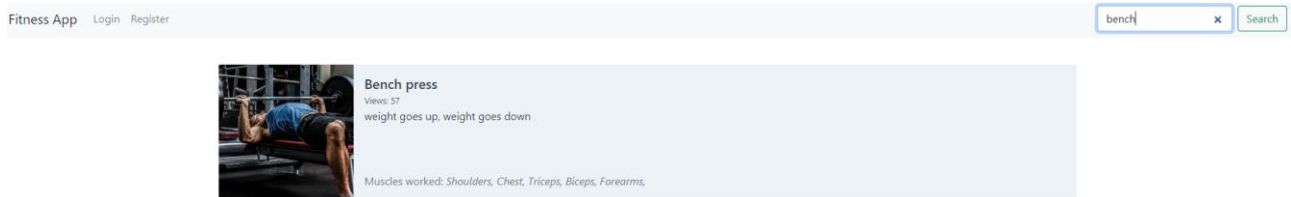
Views: 4 Upload Date: 2021-06-05 20:03:47.694504



Squat

Views: 11 Upload Date: 2021-06-05 20:04:46.806712

Or via the search function:



Viewing exercise detail: Users can click the exercises from the front page or search to be redirected to the main exercise information page with comments, likes and views all viewable by the user. Logged in users can add exercises to their profiles from this page.



weight goes up, weight goes down

Enter comment...

Post

test

test@test.com

2021-06-05 21:17:25.062426

User Register: To register, users must enter an email address and password with password confirmation.

Register

Email address

Password

Password Confirmation

Register

Have an account? [Log in!](#)

User login: Users can login via their facebook accounts if they do not wish to register a new account or login with an account they registered with. They will need to complete the reCAPTCHA before being allowed to login.

Login

Email address

Password



I'm not a robot



reCAPTCHA
[Privacy](#) • [Terms](#)

Log in



Log In

Don't have an account? [Sign up!](#)

User exercise upload after login: Users can upload exercises under two main types, ones which will only be viewable by the user on the personal page, and others which can be uploaded to the main site but need to be approved by an admin first.

Upload Exercise

Name

Type of exercise

Compound

Exercise Level

Beginner

Muscle groups (check all that apply)

Neck	Traps	Shoulders	Chest	Triceps	Biceps	Forearms	Abdominals	Middle back	Lats	Lower back
Glutes			Quadriceps			Hamstrings			Calves	

Description

Video URL (optional)

Exercise Image (optional)

Choose file No file chosen

Upload To Personal Profile or Website

Upload To Website

Personal Page

Upload To Website

Upload Exercise

Name

my yoga routine

Type of exercise

Cardio

Exercise Level

Intermediate

Muscle groups (check all that apply)

Neck	Traps	Shoulders	Chest	Triceps	Biceps	Forearms	Abdominals	Middle back	Lats	Lower back
Glutes			Quadriceps			Hamstrings			Calves	

Description

an easy yoga routine

Video URL

<https://www.youtube.com/embed/XClviBT3Txc>

Exercise Image

Choose file bruce-mars-gJtDg6WfMIQ-unsplash.jpg

Upload To Personal Profile or Website

Personal Page

Upload

User personal profile: Users can view personally uploaded exercises or exercises which have been added to the main site and then to their profile. To distinguish these different exercise categories, users are prompted by a bin icon for their own personal exercises if they choose to delete them they will be permanently removed. However, exercises on the main site will still stay on the site itself.

Hi! test@test.com



Personal profile exercise: Personal profile exercises do not have views/likes or comments.



Logout: users can easily logout by clicking the logout button on the top of the page.

API Access

Outside users can access the public api via api.cloudfitness.click and use the data for their own personal use.

GET: <https://api.cloudfitness.click/exercises> to get list of all exercises

GET: <https://api.cloudfitness.click/exercises/approved> to get list of all approved exercises.

GET: <https://api.cloudfitness.click/exercises/pending> to get list of all pending exercises.

GET: <https://api.cloudfitness.click/exercises/types> to get a list of types you can pass through to get specific exercises.

GET: <https://api.cloudfitness.click/exercises?type={type}&name={name}> to get a specific exercise

Normal users cannot access PUT requests which is only available to admin with the use of an API key.

Admin Access

Login: to use the admin functions of the admin site (user: admin, password: password) all admins must login.

Login

Username


Password

Log in

View Pending exercises: To approve pending exercises, admins must visit the pending tab and view the exercise information in the table. They can click approve if they wish the exercise to be displayed on the site.

Fitness App - Admin Console Pending Approved Logout



Pending List

Approved	Likes	Level	Views	Video Url	Description	Image	Name	Muscle Groups	Exercise Type	Upload Date	Pending
False	0.0	Advanced	11.0	https://www.youtube.com/embed/bEv6CCg2BC8	Become a t-rex		squat	['Abdominals', 'Middle back', 'Lower back', 'Glutes', 'Quadriceps', 'Hamstrings']	compound	2021-06-05 20:04:46.806712	<button>Approve</button>

View Approved exercises: If admins wish to change approved exercises back to pending to remove them from the main site, they can visit the approved tab and select the 'Pending' button on the appropriate exercise.

Fitness App - Admin Console Pending Approved Logout

Approved List

Approved	Likes	Level	Views	Video Url	Description	Image	Name	Muscle Groups	Exercise Type	Upload Date	Approved
True	1.0	Intermediate	64.0	https://www.youtube.com/embed/-MAARwVXoek	weight goes up, weight goes down		bench-press	['Shoulders', 'Chest', 'Triceps', 'Biceps', 'Forearms']	compound	2021-06-05 20:01:04.913869	<button>Pending</button>
True	0.0	Intermediate	4.0	https://www.youtube.com/embed/Zj-2vd4-P14I	The superior way to build hamstrings		romanian-deadlift	['Abdominals', 'Lower back', 'Hamstrings']	compound	2021-06-05 20:03:47.694504	<button>Pending</button>

References

- [1] Healey, "'Sweat tech' is on the rise, with 39% of Australians turning to digital platforms, apps and at-home fitness technology in the last year", *Business Insider Australia*, 2021. [Online]. Available: <https://www.businessinsider.com.au/sweat-tech-fitness-apps-australia-2021-3>. [Accessed: 05- Jun- 2021].
- [2] Alex Damiani. *Flask to AWS ECS Series*. (Jan. 3, 2021). Accessed: June. 1, 2021. [Online Video]. Available: https://www.youtube.com/watch?v=kqa_cchAMLY&t=24s
- [3] Cloud Path. *AWS CodePipeline tutorial | Build a CI/CD Pipeline on AWS*. (Apr. 29, 2019). Accessed: May. 10, 2021. [Online Video]. Available: <https://www.youtube.com/watch?v=NwzJCSPSPZs>
- [4] AWS. *Getting started using Elastic Beanstalk*. (2021). Accessed: May. 2, 2021. [Documentation]. Available: <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html>
- [5] AWS. *Configuring an Application Load Balancer*. (2021). Accessed: May. 5, 2021. [Documentation]. Available: <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/environments-cfg-alb.html>
- [6] Implementing a REST API with AWS (API Gateway, Lambda, and DynamoDB) (2020). Accessed: May 31. 2021. [Online]. Available: <https://levelup.gitconnected.com/implementing-a-rest-api-with-aws-api-gateway-lambda-and-dynamodb-c62b8a1f6182>