# Task: DevOps Assessment II

# Overview

Dockerize and deploy the following NodeJS application on the AWS EC2 instances.

Further, you are required to add autoscaling and server health monitoring for the above application

You are also required to add a CRON job to move application generated logs to the AWS S3 buckets

# Application

Link: <https://github.com/strongloop/express-example-app/blob/master/index.js>

# Requirements

Following are the requirements that your solution must have

# 1. Dockerize the Application

Write a Dockerfile to containerize the above application

💡 Use the Linux Docker image

# 2. Deploy the Application on the AWS EC2 instance

Here you are required to do the following

1. Spawn the EC2 instance and deploy the application on the EC2 instance
2. Create an Application Load Balancer in order to communicate with the Application

# 3. Implement the Auto-Scaling for the Application

Implement the autoscaling for the application using the CPU\_USAGE metric i.e.

if CPU usage becomes greater than 65% then add more EC2 instances, and if CPU usage is less than 65% then we want to run the minimum amount of EC2 instances

# 4. Add AWS Cloudwatch Alarms

Add the following 2 Cloudwatch alarms to notify the users

1. Notify if the application is unhealthy and is not accepting the clients’ request
2. Notify if the application returns more than 100 5xx HTTP error codes to the clients

# 5. Create CRON Job for Logs collection

Create a CRON job that runs on the daily basis and copy the log files from the EC2 instances to the S3

💡 You can assume the application’s logs are in the storage directory of the application e.g. ~/application/storage/logs/\*.log

# Playground Rules

1. You must only use the [Amazon AWS](https://aws.amazon.com/) cloud service provider.
2. You are required to include all of the services in your Cost Analysis.

# Deliverables

You are required to submit the following

1. Dockerfile of the application
2. Guide to setup CRON expression and Bash script that copies the log files from EC2 to S3
3. Cloudformation template containing all the AWS resources required to power the application
4. Link to **Cost Analysis.**

# Deadline

The deadline for this task is **Four (4)** days from the time you receive this email.

Architecture

Diagram

Description automatically generated

**AWS Services Used:**

AWS CloudFormation

Docker (Dockerfile)

ECR Repository

ECS Containers on EC2

Application Load Balancer

Auto Scaling Group

CloudWatch Alarm

S3 Buckets

AWS Cost Analysis

**Application:**

Nodejs

**Automation:**

Cronjob

Bash script

Application:

git clone https://github.com/strongloop/express-example-app

1) Build Docker Image and upload to ECR Repository

2) Prepare CloudFormation manifesto file in YAML.

1. 01-nodjsvpc.yml

2. 02-alb-external.yml

3. 03-service-ec2-public-lb.yml

## Installation

git clone https://github.com/strongloop/express-example-app

cd express-example-app

npm install

Dockerfile

#vi Dockerfile

==============================================================

FROM node:latest

# Create app directory

WORKDIR /usr/src/app

# Install app dependencies

# A wildcard is used to ensure both package.json AND package-lock.json are copied

# where available (npm@5+)

COPY package\*.json ./

RUN npm install

# If you are building your code for production

# RUN npm ci --only=production

# Bundle app source

COPY . .

EXPOSE 8080

CMD [ "node", "server.js" ]

======================================================

#docker build -t nodejs-app .

#docker run -it -p 8080:3000 nodejs-app

#docker images

#docker containers ps -a

#docker logs containers id

root@ip-10-0-0-152:~/express-example-app# docker logs 791e595b7415

Listening on port: 3000

**Test Docker Image**

root@ip-10-0-0-152:~/express-example-app# docker exec -it 791e595b7415 /bin/bash

root@791e595b7415:/usr/src/app#

root@791e595b7415:/usr/src/app# curl -i localhost:3000

HTTP/1.1 200 OK

X-Powered-By: Express

Content-Type: text/html; charset=utf-8

Content-Length: 56

ETag: W/"38-vSrShT01T++m++D95f4/cShFkUA"

Date: Thu, 09 Jun 2022 13:59:16 GMT

Connection: keep-alive

Keep-Alive: timeout=5

Hello, World! Request served from ::ffff:127.0.0.1:3000

root@791e595b7415:/usr/src/app#

Install Docker in EC2 Bastion in public subnet

# curl -fsSL https://get.docker.com -o get-docker.sh ( this will download shell script in the machine)

# sh get-docker.sh ( This will execute the shell script, which will install docker )

**Nodejs Application**

#git clone <https://github.com/strongloop/express-example-app.git>

**Install awscli**

#install aws cli

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

apt install unzip

unzip awscliv2.zip

sudo ./aws/install

root@ip-10-0-0-152:~/express-example-app# sudo ./aws/install

You can now run: /usr/local/bin/aws --version

root@ip-10-0-0-152:~/express-example-app#

./aws/install -i /usr/local/aws-cli -b /usr/local/bin

aws –version

root@ip-10-0-0-152:~/express-example-app# aws --version

aws-cli/2.7.7 Python/3.9.11 Linux/5.13.0-1022-aws exe/x86\_64.ubuntu.20 prompt/off

root@ip-10-0-0-152:~/express-example-app#

========

**ECR Registration:**

#aws ecr create-repository –repository-name nodejs-rep

Push images into ECR Repository

**Execute cloudformation stack**

aws cloudformation –create-stack –stack-name vpc –tempalte-body <file://$PWD/nodjsvpc.yaml> --capabilities CAPABILITY\_IAM

aws cloudformation create-stack --stack-name alb --template-body <file://$PWD/02-alb-external.yml>

aws cloudformation –create-stack –stack-name nodejsapp –tempalte-body <file://$PWD/03-service-ec2-public-lb.yml>

**Delete Stack**

aws cloudformation –delete-stack –stack-name nodejsapp

aws cloudformation –delete-stack –stack-name alb

aws cloudformation –delete-stack –stack-name nodjsvpc

**Cost Analysis:**

1)workload for application is minimal. So run task on minimal containers.

2)run containers only during business hour if not used in the off-business hour

3) Copy log files into S3 Glacier

Source file repository: