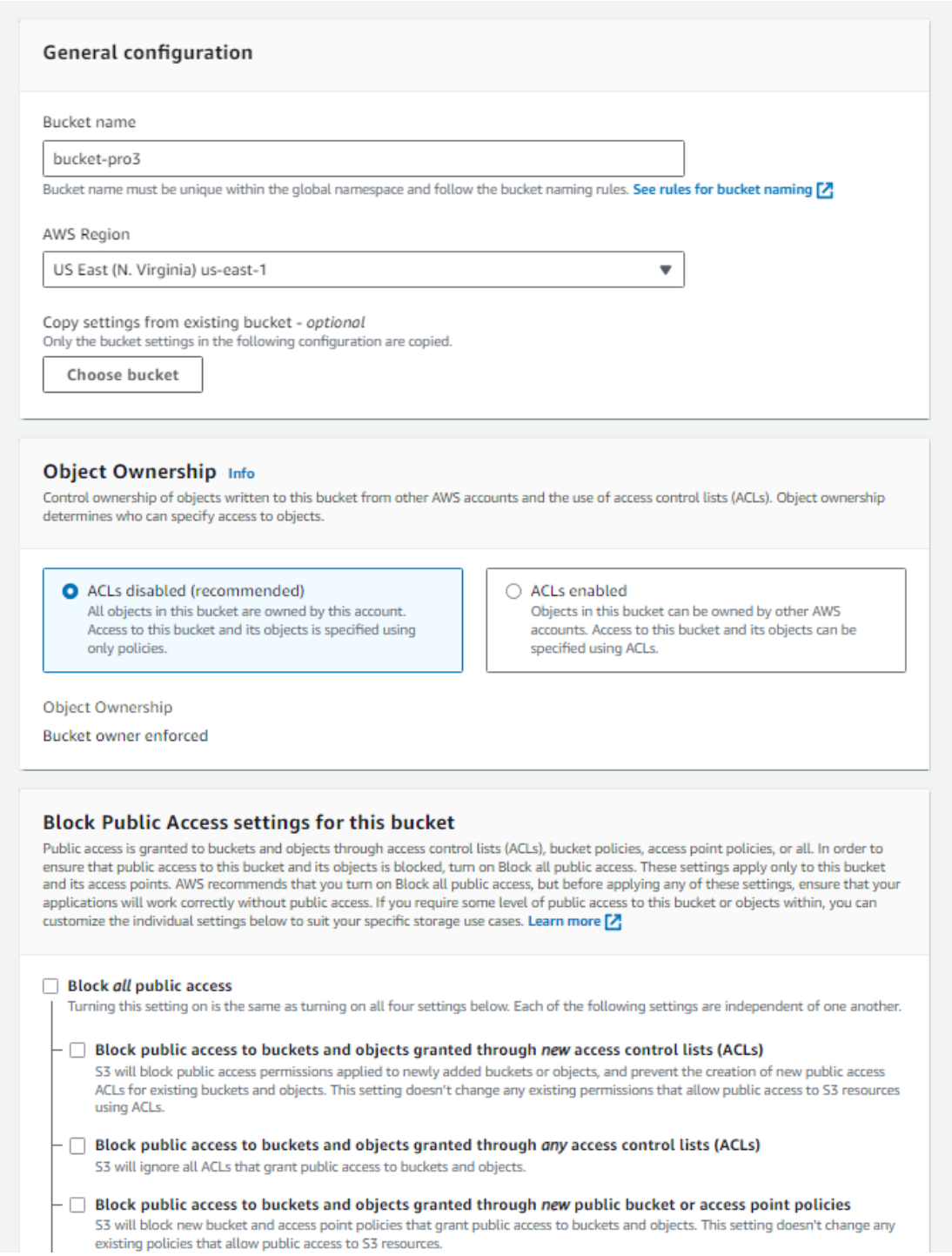
Refactor Monolith to Microservices and Deploy

(Lưu ý: tài liệu viết 2 lần deploy khác nhau có thể có giá trị không mapping với nhau vd: name images docker reverseproxy)

Chúc mọi người sớm pass project – có thắc mắc liên hệ NhanNV13

Create an S3 bucket

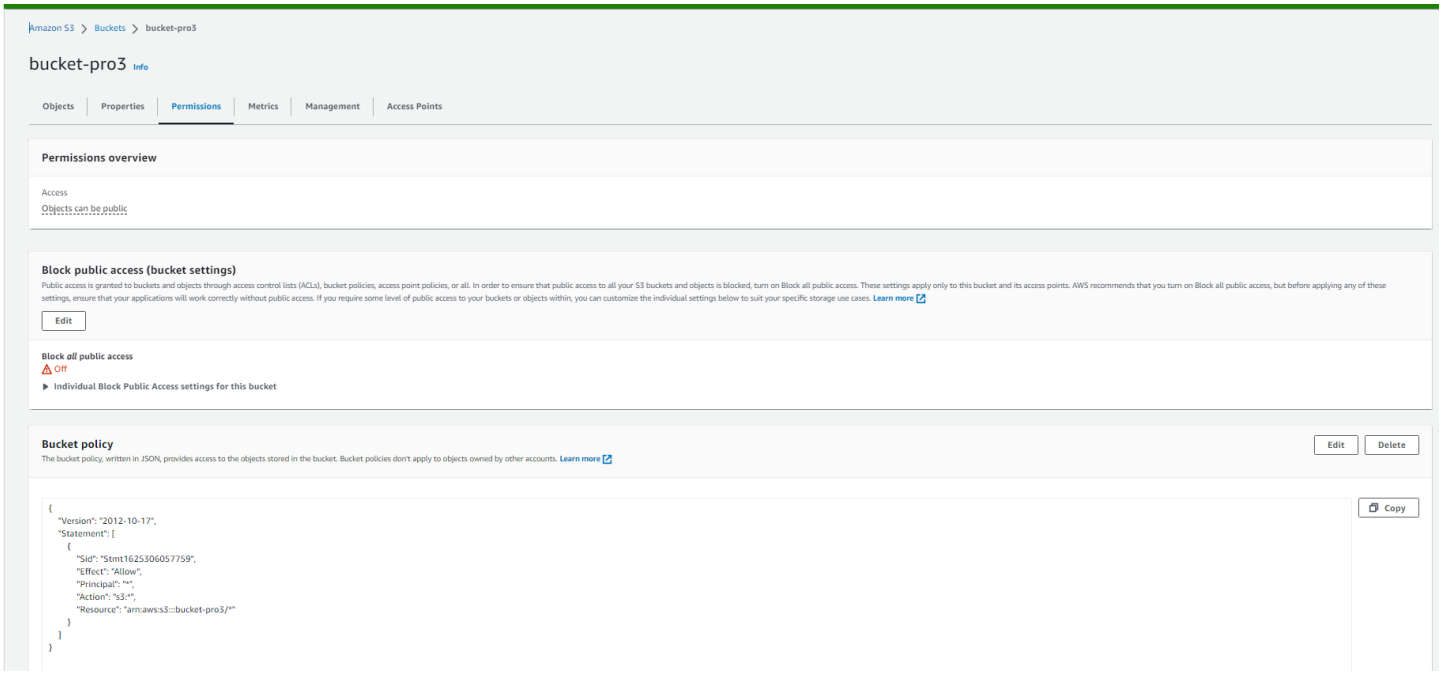


Navigate to S3 from the AWS console.

Create a public S3 bucket with default configuration, such as no versioning and disabled encryption.

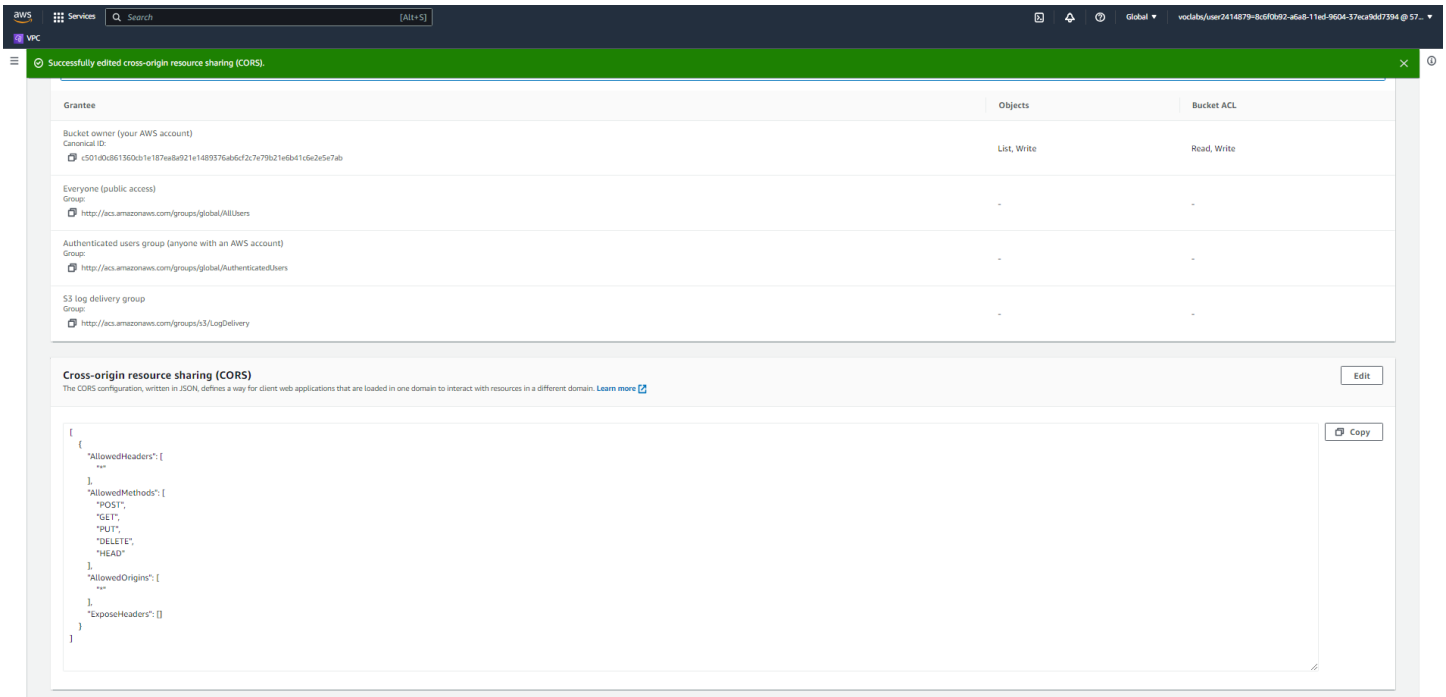
Once your bucket is created, go to the Permissions tab. Add bucket policy allowing other AWS services (Kubernetes) to access the bucket contents

```
{
 "Version":"2012-10-17",
 "Statement":[
   {
     "Sid":"Stmt1625306057759",
     "Principal":"*",
     "Action":"s3:*",
     "Effect":"Allow",
     "Resource":"arn:aws:s3:::[bucket-name]"
   }
 ]
}
```

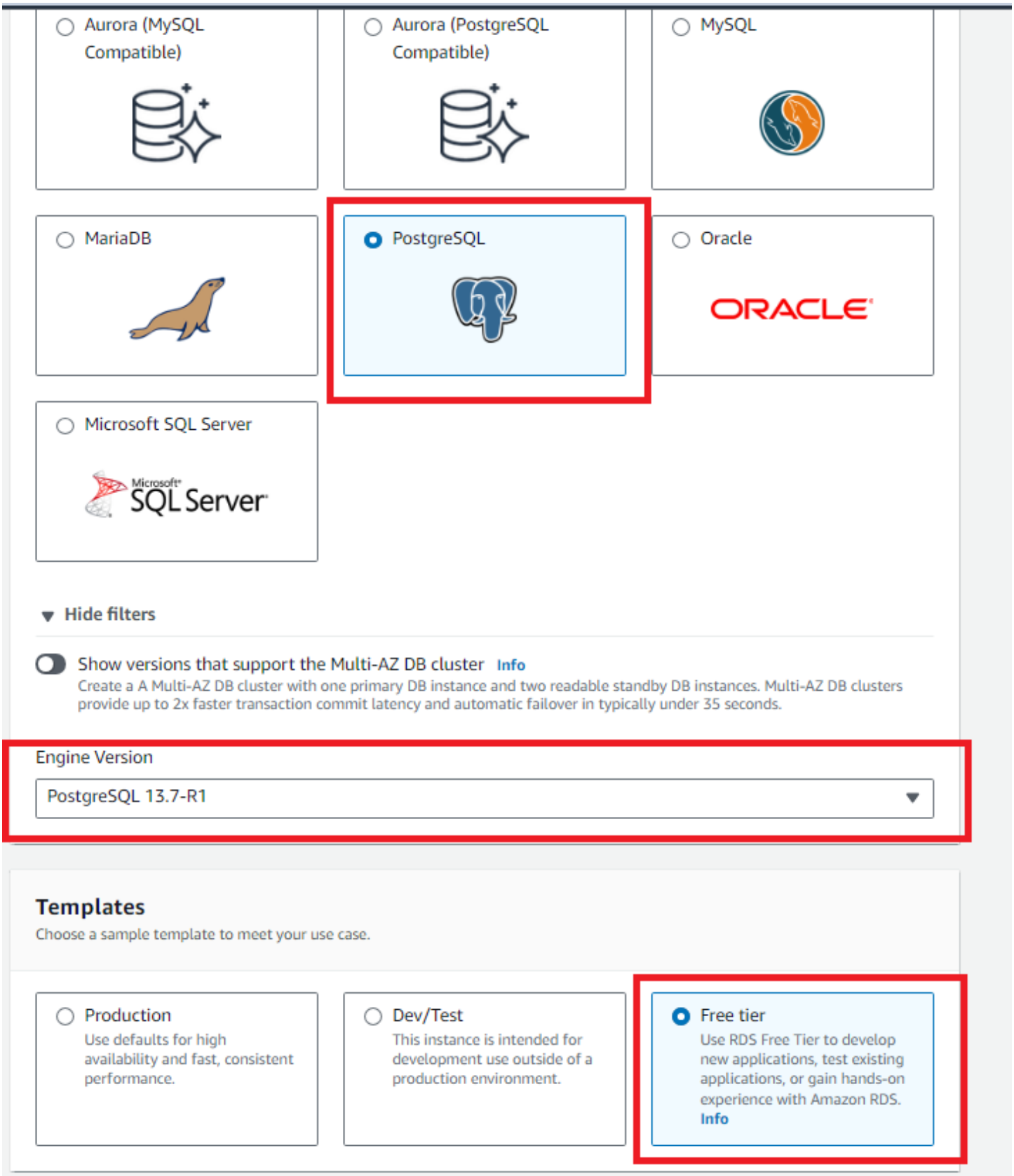


NhanNV13 – Nguyen Van Nhan

Add the CORS configuration to allow the application running outside of AWS to interact with your bucket. You can use the following configuration:



Create PostgreSQL database



Settings

DB instance identifier Info

Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

database-1

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

Credentials Settings

Master username Info

Type a login ID for the master user of your DB instance.

postgres

1 to 16 alphanumeric characters. The first character must be a letter.

☐ Manage master credentials in AWS Secrets Manager

Manage master user credentials in Secrets Manager. RDS can generate a password for you and manage it throughout its lifecycle.

If you manage the master user credentials in Secrets Manager, some RDS features aren't supported.
Learn more

☐ Auto generate a password

Amazon RDS can generate a password for you, or you can specify your own password.

Master password Info

Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), '(single quote), "(double quote) and @ (at sign).

Confirm master password Info

Default VPC (vpc-076f8e9b95b27e930)
6 Subnets, 6 Availability Zones

Only VPCs with a corresponding DB subnet group are listed.

After a database is created, you can't change its VPC.

DB subnet group Info

Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.

default

Public access Info

- (●) Yes
  RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.
- ( ) No
  RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

VPC security group (firewall) Info

Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.

- ( ) Choose existing
  Choose existing VPC security groups
- (●) Create new
  Create new VPC security group

New VPC security group name

postgres

Availability Zone Info

No preference

RDS Proxy

RDS Proxy is a fully managed, highly available database proxy that improves application scalability, resiliency, and security.

[ ] Create an RDS Proxy Info

RDS automatically creates an IAM role and a Secrets Manager secret for the proxy. RDS Proxy has additional costs. For more information, see Amazon RDS Proxy pricing.

Certificate authority - *optional* Info

Using a server certificate provides an extra layer of security by validating that the connection is being made to an Amazon database. It does so by checking the server certificate that is automatically installed on all databases that you provision.

rds-ca-2019 (default)

If you don't select a certificate authority, RDS chooses one for you.

Monitoring

☐ Turn on Performance Insights

▶ **Additional configuration**

Enhanced Monitoring

▶ Additional configuration

Database options, encryption turned on, backup turned on, backtrack turned off, maintenance, CloudWatch Logs, delete protection turned off.

Estimated monthly costs

The Amazon RDS Free Tier is available to you for 12 months. Each calendar month, the free tier will allow you to use the Amazon RDS resources listed below for free:

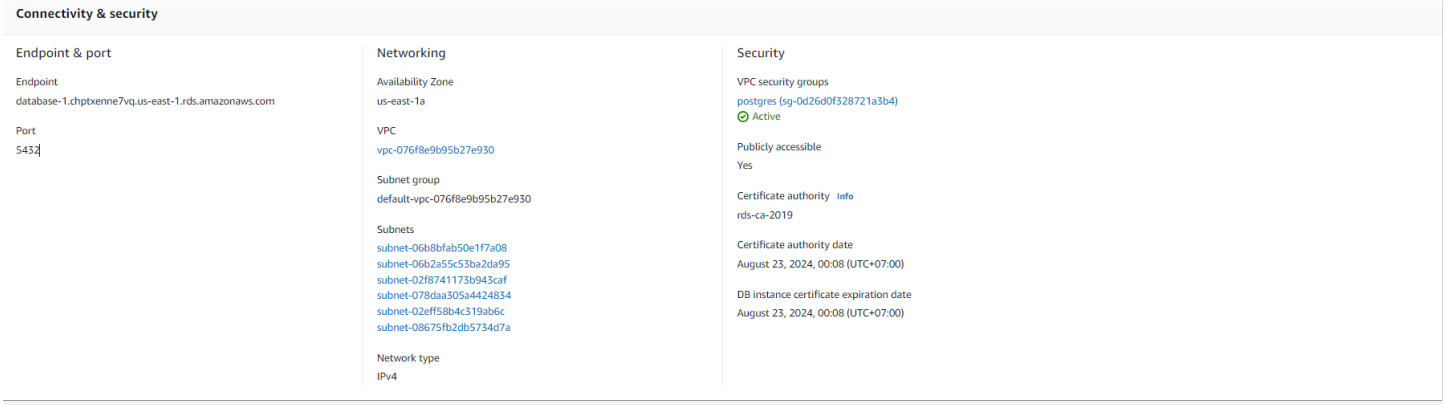
- 750 hrs of Amazon RDS in a Single-AZ db.t2.micro, db.t3.micro or db.t4g.micro Instance.
- 20 GB of General Purpose Storage (SSD).
- 20 GB for automated backup storage and any user-initiated DB Snapshots.

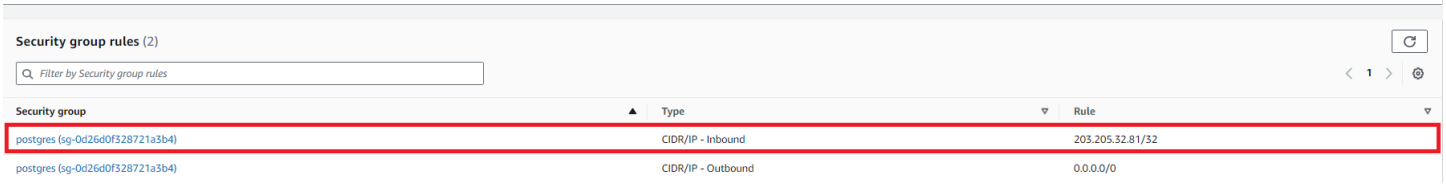
Learn more about AWS Free Tier.

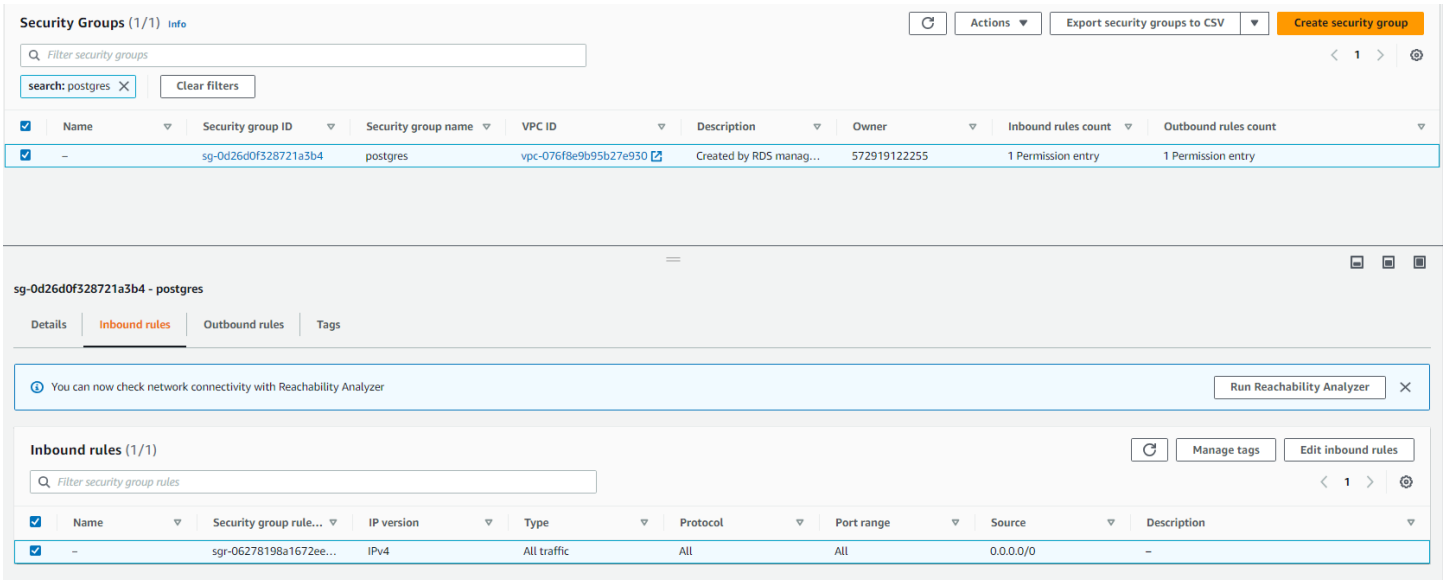
When your free usage expires or if your application use exceeds the free usage tiers, you simply pay standard, pay-as-you-go service rates as described in the Amazon RDS Pricing page.

You are responsible for ensuring that you have all of the necessary rights for any third-party products or services that you use with AWS services.

Cancel | **Create database**








```
ubuntu@ip-172-31-40-56:~$ psql -h database-1.chptxenne7vq.us-east-1.rds.amazonaws.com -U postgres
Password for user postgres:
psql (15.4 (Ubuntu 15.4-1.pgdg20.04+1), server 13.7)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, compression: off)
Type "help" for help.

postgres=>
```

Install docker

https://docs.docker.com/engine/install/ubuntu/

curl -fsSL https://get.docker.com -o get-docker.sh

sudo sh get-docker.sh

```
ubuntu@ip-172-31-40-56:~$ curl -fsSL https://get.docker.com -o get-docker.sh
ubuntu@ip-172-31-40-56:~$  sudo sh get-docker.sh
# Executing docker install script, commit: c2de0811708b6d9015ed1a2c80f02c9b70c8ce7b
+ sh -c apt-get update -qq >/dev/null
+ sh -c DEBIAN_FRONTEND=noninteractive apt-get install -y -qq apt-transport-https ca-certificates curl >/dev/null
+ sh -c install -m 0755 -d /etc/apt/keyrings
+ sh -c curl -fsSL "https://download.docker.com/linux/ubuntu/gpg" | gpg --dearmor --yes -o /etc/apt/keyrings/docker.gpg
+ sh -c chmod a+r /etc/apt/keyrings/docker.gpg
+ sh -c echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu focal stable" > /etc/apt/sources.list.d/docker.list
+ sh -c apt-get update -qq >/dev/null
+ sh -c DEBIAN_FRONTEND=noninteractive apt-get install -y -qq docker-ce docker-ce-cli containerd.io docker-compose-plugin docker-ce-rootless-extras docker-buildx-plugin >/dev/null
```

sudo chmod 666 /var/run/docker.sock

```
ubuntu@ip-172-31-40-56:~$ docker -v
Docker version 24.0.5, build ced0996
ubuntu@ip-172-31-40-56:~$
```

https://hub.docker.com/

export DOCKER_USERNAME=<your_username>

export DOCKER_PASSWORD=<your_password>

docker login -u="${DOCKER_USERNAME}" -p="${DOCKER_PASSWORD}"

```
ubuntu@ip-172-31-40-56:~$ docker login -u="${DOCKER_USERNAME}" -p="${DOCKER_PASSWORD}"
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
ubuntu@ip-172-31-40-56:~$
```

Install kubenctl

https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/

sudo apt-get update

sudo apt-get install -y apt-transport-https ca-certificates curl

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg --dearmor –o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update

sudo apt-get install -y kubectl

```
ubuntu@ip-172-31-40-56:~$ kubectl version --client
Client Version: v1.28.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
ubuntu@ip-172-31-40-56:~$
```

NhanNV13 – Nguyen Van Nhan

Install awscli

sudo apt install awscli

```
untu@ip-172-31-40-56:~/project3$ aws -v
te: AWS CLI version 2, the latest major version of the AWS C
install-cliv2.html

age: aws [options] <command> <subcommand> [<subcommand> ...]
 see help text, you can run:

aws help
aws <command> help
aws <command> <subcommand> help
s: error: the following arguments are required: command
untu@ip-172-31-40-56:~/project3$
```

Refactor the Backend API

```
ubuntu@ip-172-31-40-56:~$ git clone https://github.com/udacity/cd0354-monolith-to-microservices-project.git project3
Cloning into 'project3'...
remote: Enumerating objects: 197, done.
remote: Total 197 (delta 0), reused 0 (delta 0), pack-reused 197
Receiving objects: 100% (197/197), 5.63 MiB | 30.53 MiB/s, done.
Resolving deltas: 100% (29/29), done.
ubuntu@ip-172-31-40-56:~$ tree -L 2 project3/
project3/
├── CODEOWNERS
├── Classroom_Project_Instructions
│   ├── Part_0_Prerequisites_and_Getting_Started.md
│   ├── Part_III_TravisCI.md
│   ├── Part_II_Microservices_Application.md
│   ├── Part_IV_Container_Orchestration.md
│   ├── Part_I_Monolithic_Application.md
│   └── Part_V_Logging.md
├── LICENSE.txt
├── README.md
├── screenshots
│   └── README.md
├── set_env.sh
├── udagram-api
│   ├── mock
│   ├── package-lock.json
│   ├── package.json
│   ├── src
│   ├── tsconfig.json
│   └── tslint.json
├── udagram-api.postman_collection.json
└── udagram-frontend
    ├── angular.json
    ├── e2e
    ├── ionic.config.json
    ├── package-lock.json
    ├── package.json
    ├── src
    ├── tsconfig.json
    ├── tslint.json
    └── udagram_tests

9 directories, 22 files
ubuntu@ip-172-31-40-56:~$
```

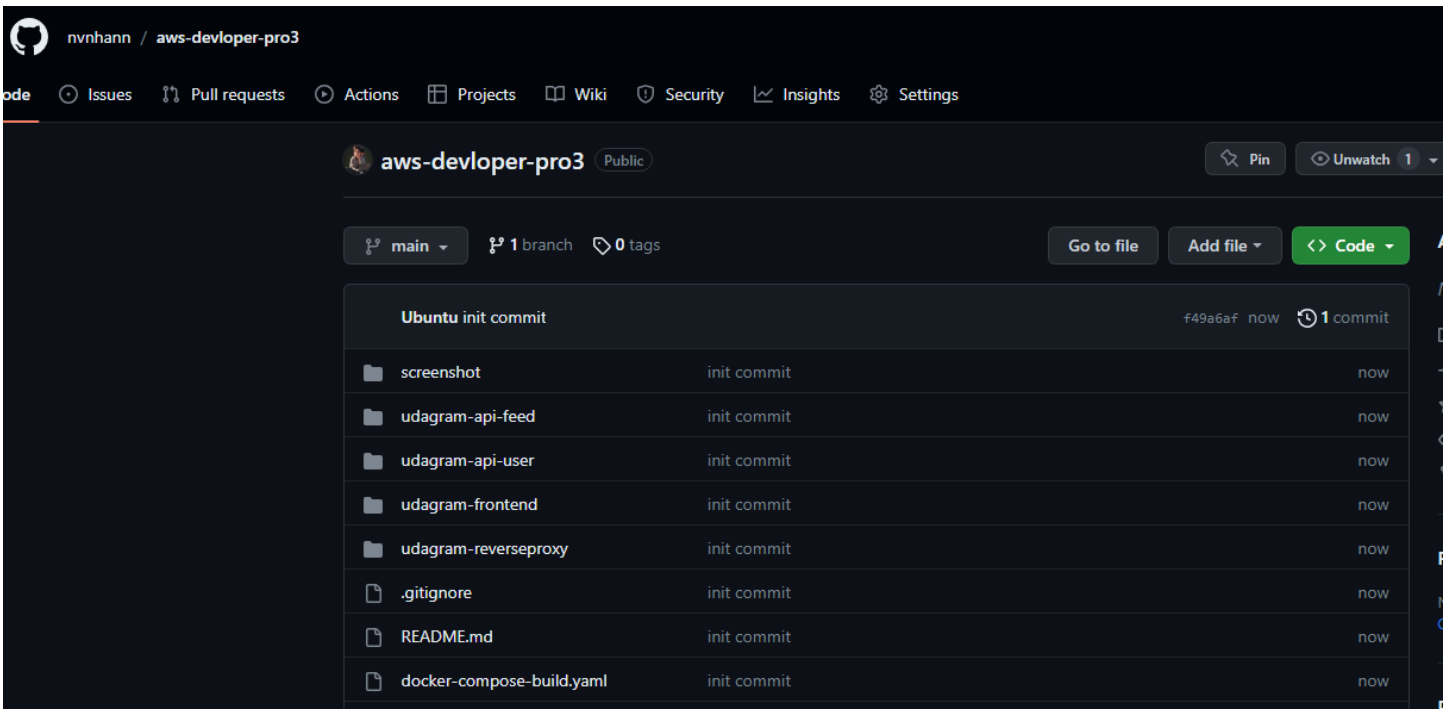
NhanNV13 – Nguyen Van Nhan

Refer: https://github.com/nvnhann/aws-devloper-pro3.git

```
ubuntu@ip-172-31-40-56:~$ git clone https://github.com/nvnhann/Cloud-developer.git
Cloning into 'Cloud-developer'...
remote: Enumerating objects: 318, done.
remote: Counting objects: 100% (318/318), done.
remote: Compressing objects: 100% (240/240), done.
remote: Total 318 (delta 56), reused 318 (delta 56), pack-reused 0
Receiving objects: 100% (318/318), 11.96 MiB | 32.47 MiB/s, done.
Resolving deltas: 100% (56/56), done.
ubuntu@ip-172-31-40-56:~$ cp Cloud-developer/Refactor-Monolith-to-Microservices/ .
cp: -r not specified; omitting directory 'Cloud-developer/Refactor-Monolith-to-Microservices/'
ubuntu@ip-172-31-40-56:~$ cp -r Cloud-developer/Refactor-Monolith-to-Microservices/ .
ubuntu@ip-172-31-40-56:~$ mv Refactor-Monolith-to-Microservices/ project3
ubuntu@ip-172-31-40-56:~$ tree -L 2 project3/
project3/
├── README.md
├── docker-compose-build.yaml
├── docker-compose.yaml
├── screenshot
│   ├── build_CI.png
│   ├── describe_hpa.png
│   ├── describe_service_1.png
│   ├── describe_service_2.png
│   ├── describe_service_3.png
│   ├── dockerhub.png
│   ├── kubectl_get_info.png
│   ├── newpost.png
│   ├── pod_pogs.png
│   ├── post_success.png
│   └── register.png
├── udagram-api-feed
│   ├── Dockerfile
│   ├── mock
│   ├── package-lock.json
│   ├── package.json
│   ├── src
│   ├── tsconfig.json
│   └── tslint.json
├── udagram-api-user
│   ├── Dockerfile
│   ├── mock
│   ├── package-lock.json
│   ├── package.json
│   ├── src
│   ├── tsconfig.json
│   └── tslint.json
├── udagram-frontend
│   ├── Dockerfile
│   ├── angular.json
│   ├── e2e
```

NhanNV13 – Nguyen Van Nhan

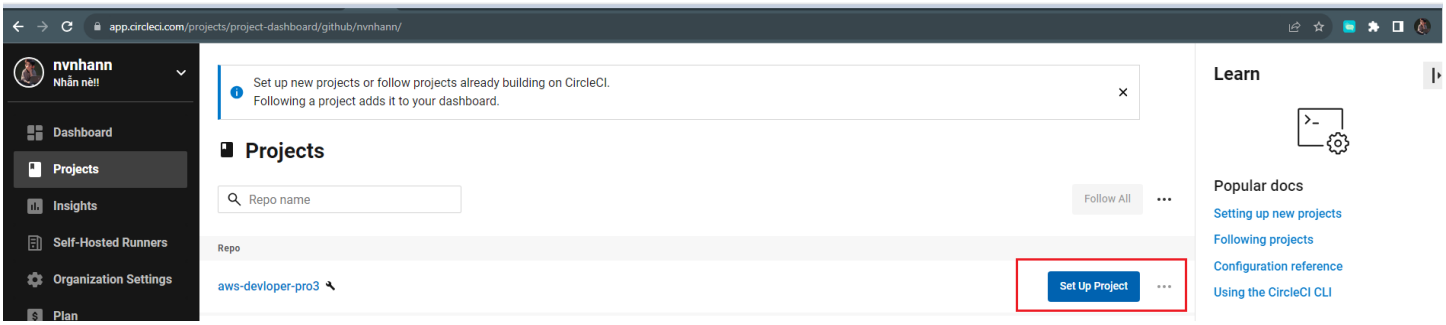
```
ubuntu@ip-172-31-40-56:~$ tree -L 2 project3/
project3/
├── README.md
├── docker-compose-build.yaml
├── docker-compose.yaml
├── screenshot
│   ├── build_CI.png
│   ├── describe_hpa.png
│   ├── describe_service_1.png
│   ├── describe_service_2.png
│   ├── describe_service_3.png
│   ├── dockerhub.png
│   ├── kubectl_get_info.png
│   ├── newpost.png
│   ├── pod_pogs.png
│   ├── post_success.png
│   └── register.png
├── udagram-api-feed
│   ├── Dockerfile
│   ├── mock
│   ├── package-lock.json
│   ├── package.json
│   ├── src
│   ├── tsconfig.json
│   └── tslint.json
├── udagram-api-user
│   ├── Dockerfile
│   ├── mock
│   ├── package-lock.json
│   ├── package.json
│   ├── src
│   ├── tsconfig.json
│   └── tslint.json
├── udagram-frontend
│   ├── Dockerfile
│   ├── angular.json
│   ├── e2e
│   ├── ionic.config.json
│   ├── package-lock.json
│   ├── package.json
│   ├── src
│   ├── tsconfig.json
│   ├── tslint.json
│   └── udagram_tests
└── udagram-reverseproxy
    ├── Dockerfile
    └── nginx.conf

12 directories, 33 files
ubuntu@ip-172-31-40-56:~$
```

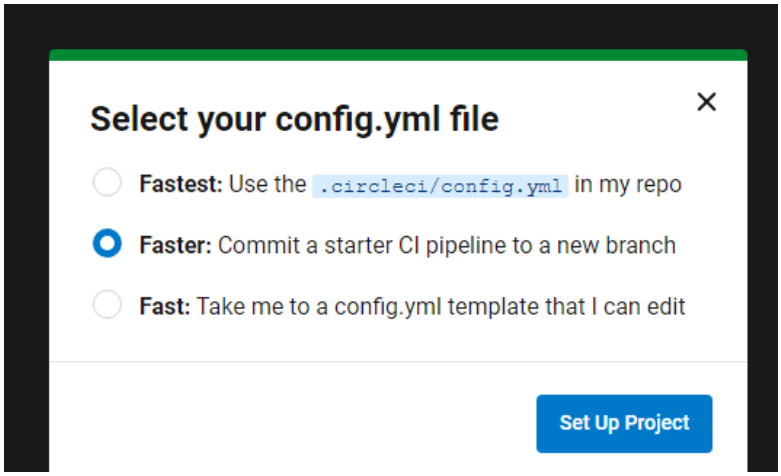


Continuous Integration

Setup circleci

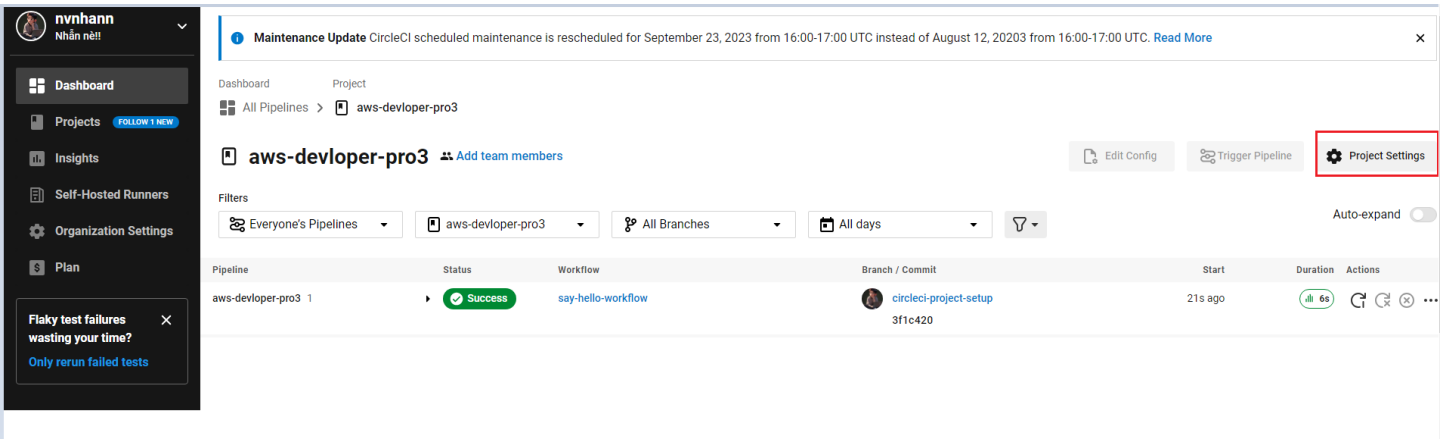
https://circleci.com/signup/

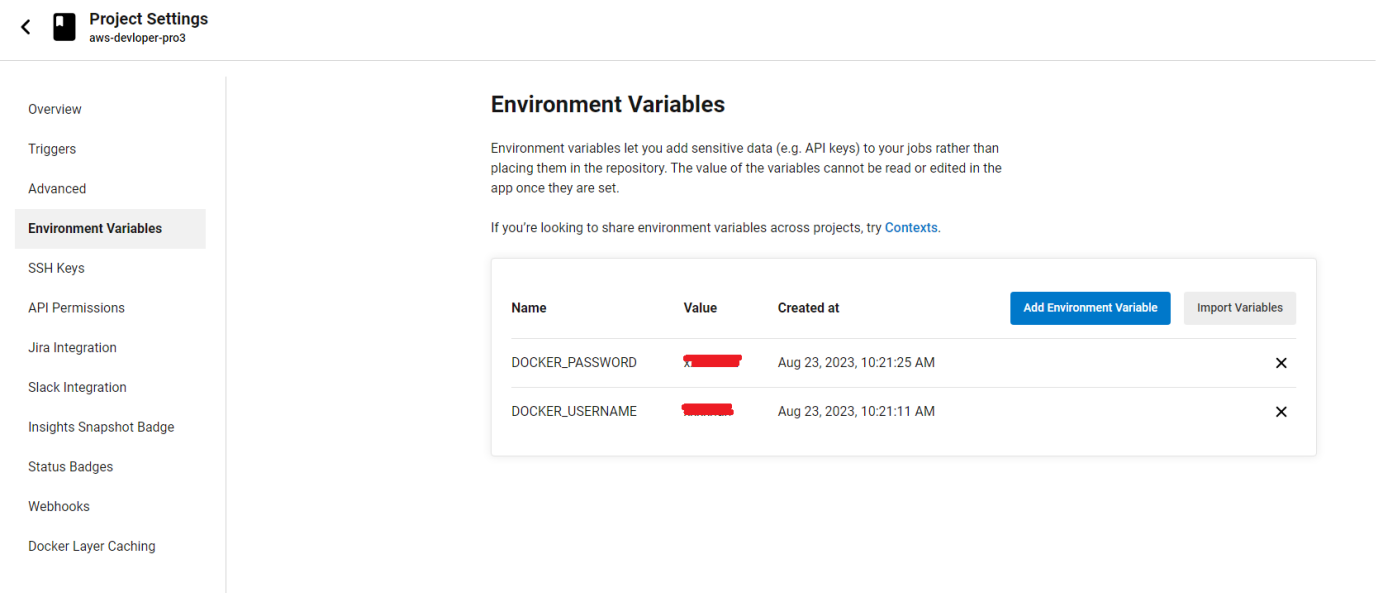




NhanNV13 – Nguyen Van Nhan

Add environment





NhanNV13 – Nguyen Van Nhan

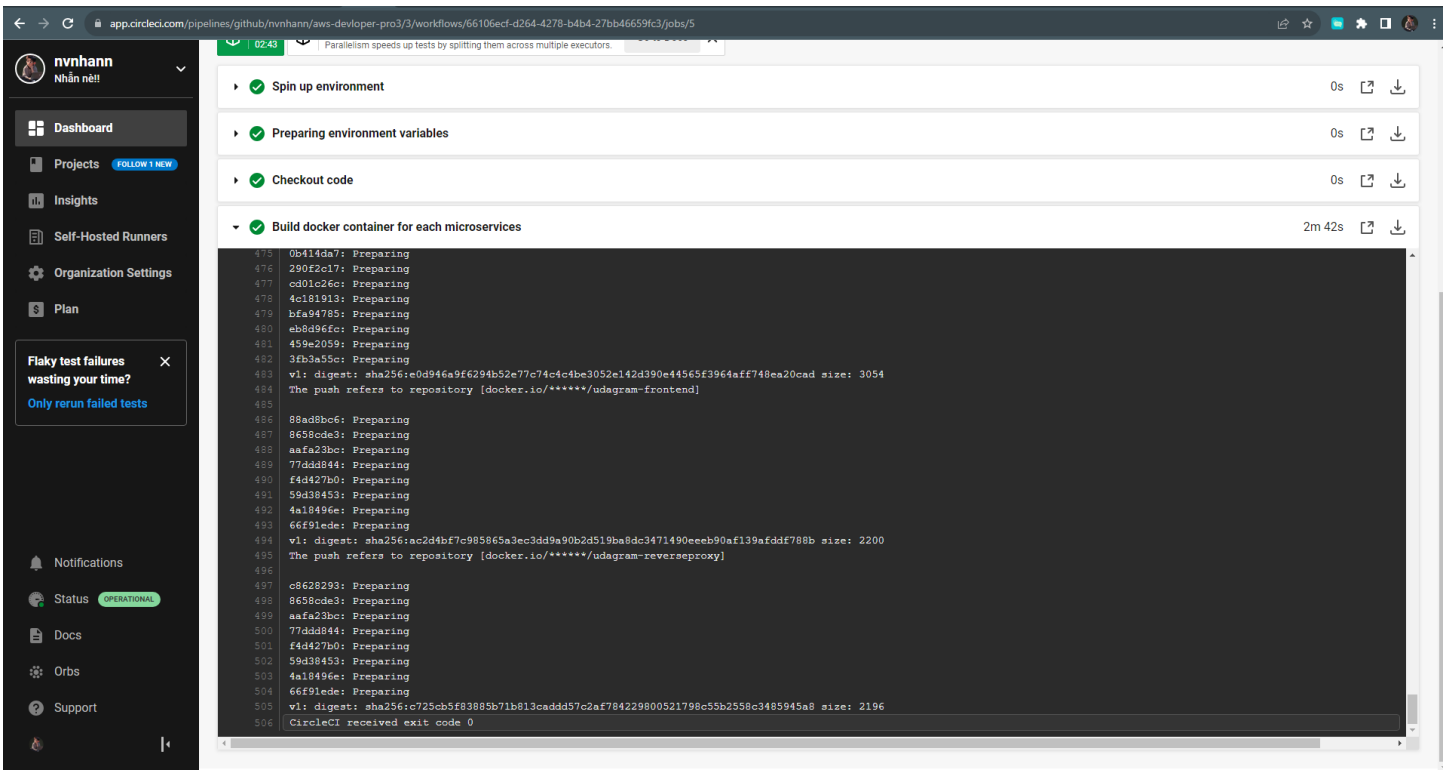
Config circleci

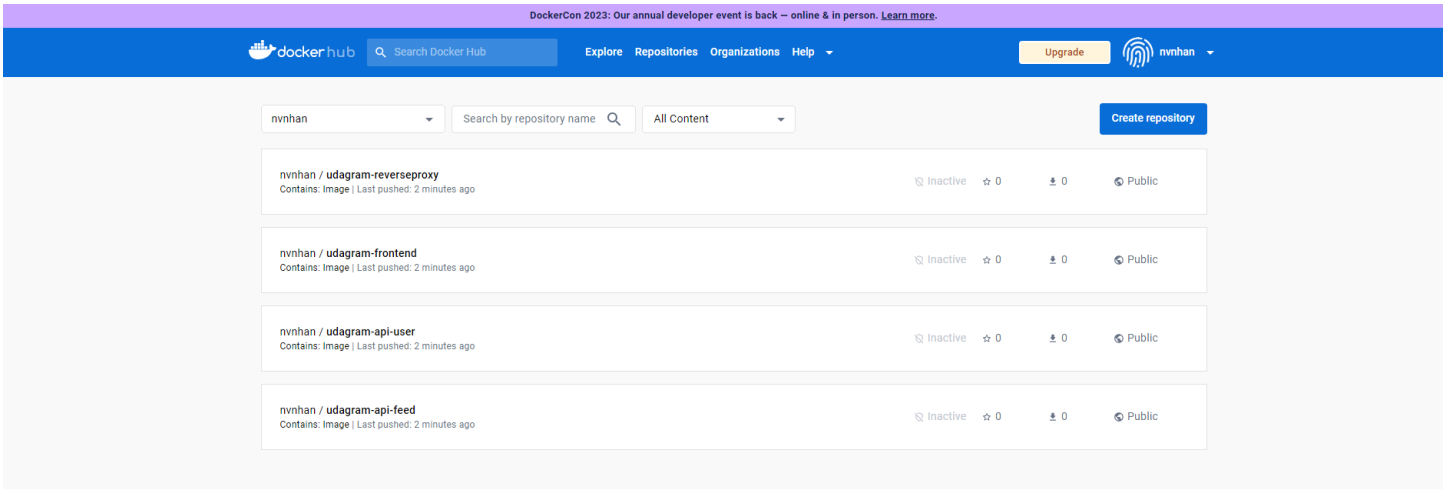
Coding - Untitled-1

```yaml
version: 2.1

jobs:
  lint-app:
    docker:
      - image: circleci/node:16
    steps:
      - checkout
  build-docker:
    machine: true
    steps:
      - checkout
      - run:
          name: Build docker container for each microservices
          command: |
            echo "--------- Installing dependencies-----------"
            curl -fsSL https://deb.nodesource.com/setup_16.x | sudo -E bash -
            sudo apt-get install -y nodejs
            node -v
            npm -v

            echo "--------- The images is Building ------------"

            docker build -t udagram-api-feed ./udagram-api-feed
            docker tag udagram-api-feed nvnhan/udagram-api-feed:v1

            docker build -t udagram-api-user ./udagram-api-user
            docker tag udagram-api-user nvnhan/udagram-api-user:v1

            docker build -t udagram-frontend ./udagram-frontend
            docker tag udagram-frontend nvnhan/udagram-frontend:v1

            docker build -t udagram-reverseproxy ./udagram-reverseproxy
            docker tag udagram-reverseproxy nvnhan/udagram-reverseproxy:v1

            echo "-------- All images succesfully built----------"

            echo " ------- login in to hub--------"

            docker login -u $DOCKER_USERNAME -p $DOCKER_PASSWORD

            echo "-------------- Pushing images to dockerhub-------"

            docker push nvnhan/udagram-api-feed:v1
            docker push nvnhan/udagram-api-user:v1
            docker push nvnhan/udagram-frontend:v1
            docker push nvnhan/udagram-reverseproxy:v1

workflows:
  default:
    jobs:
      - lint-app
      - build-docker:
          requires: [lint-app]
```

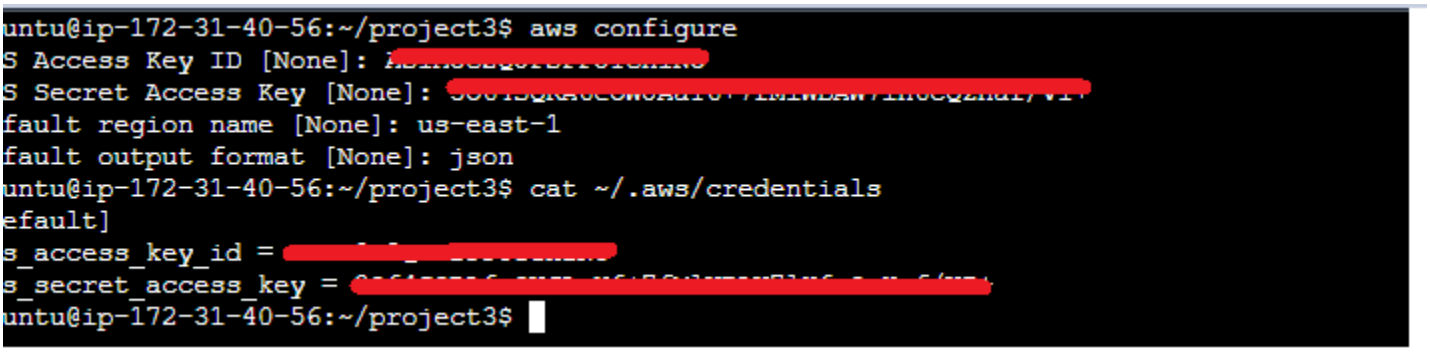
Building circleci

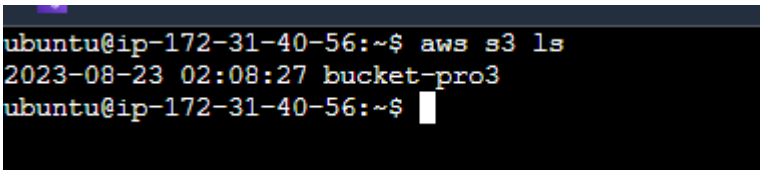
NhanNV13 – Nguyen Van Nhan





Config awscli





Container Orchestration with Kubernetes

Create an EKS Cluster using the EKSCTL Tool

sudo curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | sudo tar xz -C /usr/local/bin

Update config
aws eks update-kubeconfig --region us-east-1 --name udacity

Create eks

eksctl create cluster --name udacity --region=us-east-1 --nodes-min=2 --nodes-max=3

Deployment

Recall that while splitting the monolithic app into microservices, you used the values saved in the environment variables, as well as AWS CLI was configured locally. Similar values are required while instantiating containers from the Dockerhub images.

ConfigMap: Create env-configmap.yaml, and save all your configuration values (non-confidential environments variables) in that file.

Secret: Do not store the PostgreSQL username and passwords in the env-configmap.yaml file. Instead, create env-secret.yaml file to store the confidential values, such as login credentials. Unlike the AWS credentials, these values do not need to be Base64 encoded.

Secret: Create aws-secret.yaml file to store your AWS login credentials. Replace __INSERT_AWS_CREDENTIALS_FILE__BASE64___ with the Base64 encoded credentials (not the regular username/password).

Mac/Linux users: If you've configured your AWS CLI locally, check the contents of ~/.aws/credentials file using cat ~/.aws/credentials . It will display the aws_access_key_id and aws_secret_access_key for your AWS profile(s). Now, you need to select the applicable pair of aws_access_key from the output of the cat command above and convert that string into base64

```
ubuntu@ip-172-31-40-56:~/project3/deployment$ ll
total 36
drwxrwxr-x 2 ubuntu ubuntu 4096 Aug 23 16:20 ./
drwxrwxr-x 9 ubuntu ubuntu 4096 Aug 23 04:07 ../
-rw-rw-r-- 1 ubuntu ubuntu  803 Aug 23 15:27 aws-secret.yaml
-rw-rw-r-- 1 ubuntu ubuntu 2337 Aug 23 06:02 backend-feed-deployment.yaml
-rw-rw-r-- 1 ubuntu ubuntu 2409 Aug 23 06:07 backend-user-deployment.yaml
-rw-rw-r-- 1 ubuntu ubuntu  630 Aug 23 14:48 frontend.yaml
-rw-rw-r-- 1 ubuntu ubuntu  684 Aug 23 08:41 reverseproxy.yaml
-rw-rw-r-- 1 ubuntu ubuntu  343 Aug 23 04:36 set-env-configmap.yaml
-rw-rw-r-- 1 ubuntu ubuntu  146 Aug 23 05:49 set-env-secret.yaml
-rw-rw-r-- 1 ubuntu ubuntu    0 Aug 23 15:25 test.txt
ubuntu@ip-172-31-40-56:~/project3/deployment$
```

File: aws-secret.yaml

Coding - Untitled-3

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: nvnhan-secret
type: Opaque
data:
  credentials: <credential_base64>
```

File: set-env-configmap.yaml

Coding - Untitled-3

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: env-config
data:
  POSTGRES_USERNAME: <postgres_username>
  POSTGRES_PASSWORD: <postgres_password>
  POSTGRES_HOST: <db_host>
  POSTGRES_DB: <db_name>
  AWS_BUCKET: <bucket_name>
  AWS_REGION: <region>
  AWS_PROFILE: default
  JWT_SECRET: testing
  URL: http://localhost:8100
```

NhanNV13 – Nguyen Van Nhan

File: set-env-secret.yaml

Coding - Untitled-3

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: env-secret
type: Opaque
data:
  POSTGRESS_USERNAME: <postgres_username_base64>
  POSTGRESS_PASSWORD: <postgre_password_base64>
```

File: backend-user-deployment.yaml

Coding - Untitled-3

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: api-user
spec:
  selector:
    matchLabels:
      run: api-user
  replicas: 2
  template:
    metadata:
      labels:
        run: api-user
    spec:
      containers:
        - name: api-user
          image: nvnhan/udagram-api-user:v1
          imagePullPolicy: Always  # Set the imagePullPolicy to Always
          env:
            - name: POSTGRES_USERNAME
              valueFrom:
                configMapKeyRef:
                  name: env-config
                  key: POSTGRES_USERNAME
            - name: POSTGRES_PASSWORD
              valueFrom:
                configMapKeyRef:
                  name: env-config
                  key: POSTGRES_PASSWORD
            - name: POSTGRES_HOST
              valueFrom:
                configMapKeyRef:
                  name: env-config
                  key: POSTGRES_HOST
            - name: POSTGRES_DB
              valueFrom:
                configMapKeyRef:
                  name: env-config
                  key: POSTGRES_DB
            - name: AWS_BUCKET
              valueFrom:
                configMapKeyRef:
                  name: env-config
                  key: AWS_BUCKET
            - name: AWS_REGION
              valueFrom:
                configMapKeyRef:
                  name: env-config
                  key: AWS_REGION
            - name: AWS_PROFILE
              valueFrom:
                configMapKeyRef:
                  name: env-config
                  key: AWS_PROFILE
            - name: JWT_SECRET
              valueFrom:
                configMapKeyRef:
                  name: env-config
                  key: JWT_SECRET
            - name: URL
              valueFrom:
                configMapKeyRef:
                  name: env-config
                  key: URL
          volumeMounts:
            - name: nvnhan-secret
              mountPath: "/root/.aws/"
              readOnly: true
          ports:
            - containerPort: 80
          resources:
            limits:
              cpu: 500m
            requests:
              cpu: 200m
      volumes:
        - name: nvnhan-secret
          secret:
            secretName: nvnhan-secret

---
apiVersion: v1
kind: Service
metadata:
  labels:
    service: api-user
  name: api-user
spec:
  ports:
    - name: "8080"
      port: 8080
      targetPort: 8080
  selector:
    run: api-user
```

NhanNV13 – Nguyen Van Nhan

File: backend-feed-deployment.yaml

Coding - Untitled-3

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: api-feed
spec:
  selector:
    matchLabels:
      run: api-feed
  replicas: 2
  template:
    metadata:
      labels:
        run: api-feed
    spec:
      containers:
        - name: api-feed
          image: nvnhan/udagram-api-feed:v1
          env:
            - name: POSTGRES_USERNAME
              valueFrom:
                configMapKeyRef:
                  name: env-config
                  key: POSTGRES_USERNAME
            - name: POSTGRES_PASSWORD
              valueFrom:
                configMapKeyRef:
                  name: env-config
                  key: POSTGRES_PASSWORD
            - name: POSTGRES_HOST
              valueFrom:
                configMapKeyRef:
                  name: env-config
                  key: POSTGRES_HOST
            - name: POSTGRES_DB
              valueFrom:
                configMapKeyRef:
                  name: env-config
                  key: POSTGRES_DB
            - name: AWS_BUCKET
              valueFrom:
                configMapKeyRef:
                  name: env-config
                  key: AWS_BUCKET
            - name: AWS_REGION
              valueFrom:
                configMapKeyRef:
                  name: env-config
                  key: AWS_REGION
            - name: AWS_PROFILE
              valueFrom:
                configMapKeyRef:
                  name: env-config
                  key: AWS_PROFILE
            - name: JWT_SECRET
              valueFrom:
                configMapKeyRef:
                  name: env-config
                  key: JWT_SECRET
            - name: URL
              valueFrom:
                configMapKeyRef:
                  name: env-config
                  key: URL
          volumeMounts:
            - name: nvnhan-secret
              mountPath: "/root/.aws/"
              readOnly: true
          ports:
            - containerPort: 80
          resources:
            limits:
              cpu: 500m
            requests:
              cpu: 200m
      volumes:
        - name: nvnhan-secret
          secret:
            secretName: nvnhan-secret

---
apiVersion: v1
kind: Service
metadata:
  labels:
    service: api-feed
  name: api-feed
spec:
  ports:
    - name: "8080"
      port: 8080
      targetPort: 8080
  selector:
    run: api-feed
```

NhanNV13 – Nguyen Van Nhan

File: reverseproxy.yaml

Coding - Untitled-3

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: reverseproxy
spec:
  selector:
    matchLabels:
      run: reverseproxy
  replicas: 2
  template:
    metadata:
      labels:
        run: reverseproxy
    spec:
      containers:
      - name: reverseproxy
        image: nvnhan/reverseproxy:v1
        imagePullPolicy: Always
        ports:
        - containerPort: 80
        resources:
          limits:
            cpu: 1000m
          requests:
            cpu: 500m

---
apiVersion: v1
kind: Service
metadata:
  labels:
    service: reverseproxy
  name: reverseproxy
spec:
  ports:
  - name: "8080"
    port: 8080
    targetPort: 8080
  selector:
    service: reverseproxy
```

NhanNV13 – Nguyen Van Nhan

File: frontend.yaml

Coding - Untitled-3

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
spec:
  selector:
    matchLabels:
      run: frontend
  replicas: 2
  template:
    metadata:
      labels:
        run: frontend
    spec:
      containers:
      - name: frontend
        image: nvnhan/udagram-frontend:v1
        imagePullPolicy: Always
        ports:
        - containerPort: 80
        resources:
          limits:
            cpu: 500m
          requests:
            cpu: 200m

---
apiVersion: v1
kind: Service
metadata:
  name: frontend
  labels:
    run: frontend
spec:
  ports:
  - port: 80
    protocol: TCP
  selector:
    run: frontend
```

NhanNV13 – Nguyen Van Nhan

kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
kubectl apply -f set-env-secret.yaml
kubectl apply -f set-env-configmap.yaml
kubectl apply -f aws-secret.yaml
kubectl apply -f backend-user-deployment.yaml
kubectl apply -f backend-feed-deployment.yaml
kubectl apply -f reverseproxy.yaml

kubectl expose deployment reverseproxy --type=LoadBalancer --name=reverseproxy-ep --port=8080

```
ubuntu@ip-172-31-40-56:~/project3/deployment$ kubectl get services
NAME              TYPE           CLUSTER-IP       EXTERNAL-IP                                                              PORT(S)          AGE
api-feed          ClusterIP      10.100.43.69     <none>                                                                   8080/TCP         5h49m
api-user          ClusterIP      10.100.145.204   <none>                                                                   8080/TCP         5h49m
kubernetes        ClusterIP      10.100.0.1       <none>                                                                   443/TCP          6h
reverseproxy      ClusterIP      10.100.36.171    <none>                                                                   8080/TCP         5h2m
reverseproxy-ep   LoadBalancer   10.100.117.200   a20e2206d47fe40258b463e966fb4004-407380140.us-east-1.elb.amazonaws.com   8080:32370/TCP   13s
ubuntu@ip-172-31-40-56:~/project3/deployment$
```

Update udagram-frontend/src/environments/environment.ts file - Replace the keyword localhost in the http://localhost:8080/api/v0 string with the External-IP of the reverseproxy deployment

```
ubuntu@ip-172-31-40-56:~/project3$ grep -r "8080/api"
udagram-frontend/src/environments/environment.prod.ts:  apiHost: 'http://a20e2206d47fe40258b463e966fb4004-407380140.us-east-1.elb.amazonaws.com:8080/api/v0'
udagram-frontend/src/environments/environment.ts:  apiHost: 'http://a20e2206d47fe40258b463e966fb4004-407380140.us-east-1.elb.amazonaws.com:8080/api/v0'
ubuntu@ip-172-31-40-56:~/project3$
```

Build a new frontend image, and push it to the Dockerhub. While building a new image, it is recommended to use a different tag each time, as shown in the example below

docker build -t udagram-frontend ./udagram-frontend
docker tag udagram-frontend nvnhan/udagram-frontend:v1
docker push nvnhan/udagram-frontend:v1

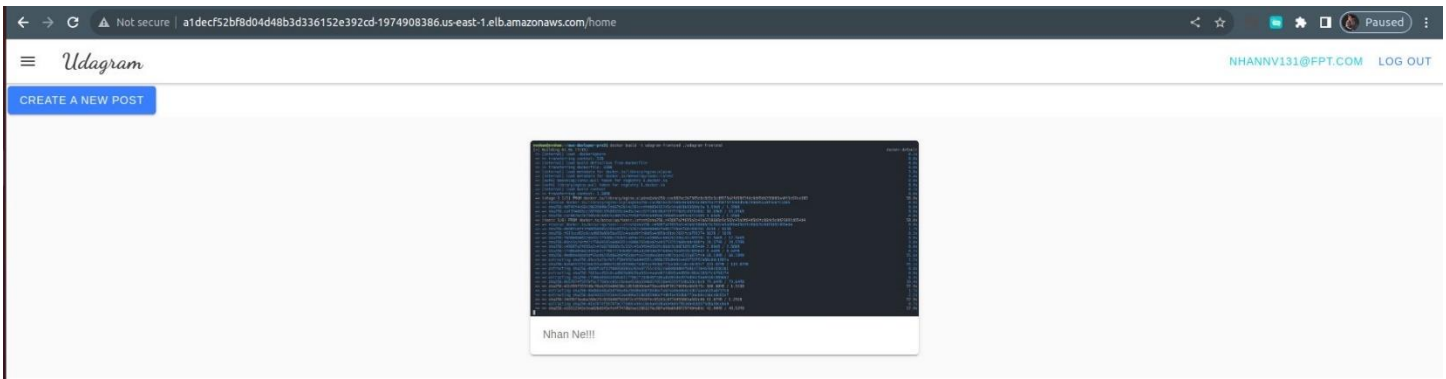
```
nvnhan@nvnhan:~/aws-devloper-pro3$ docker build -t udagram-frontend ./udagram-frontend
[+] Building 61.8s (7/15)                                                                                    docker:default
 => [internal] load .dockerignore                                                                                      0.1s
 => => transferring context: 52B                                                                                       0.0s
 => [internal] load build definition from Dockerfile                                                                   0.0s
 => => transferring dockerfile: 430B                                                                                   0.0s
 => [internal] load metadata for docker.io/library/nginx:alpine                                                        3.8s
 => [internal] load metadata for docker.io/beevelop/ionic:latest                                                       3.4s
 => [auth] beevelop/ionic:pull token for registry-1.docker.io                                                          0.0s
 => [auth] library/nginx:pull token for registry-1.docker.io                                                           0.0s
 => [internal] load build context                                                                                      0.1s
 => => transferring context: 1.38MB                                                                                    0.0s
 => [stage-1 1/2] FROM docker.io/library/nginx:alpine@sha256:cac882be2b7305e0c8d3e3cd0575a2fd58f5fde6dd5d6299605aa0f3e67ca385   58.0s
```

```
nvnhan@nvnhan:~/aws-devloper-pro3$ docker tag udagram-frontend nvnhan/udagram-frontend:v1
nvnhan@nvnhan:~/aws-devloper-pro3$ docker images
REPOSITORY                TAG       IMAGE ID       CREATED          SIZE
nvnhan/udagram-frontend   v1        b7c44fd5a7ad   31 seconds ago   60MB
udagram-frontend          latest    b7c44fd5a7ad   31 seconds ago   60MB
nvnhan@nvnhan:~/aws-devloper-pro3$ docker push nvnhan/udagram-frontend:v1
The push refers to repository [docker.io/nvnhan/udagram-frontend]
fb0c08cd8255: Pushing [==================================================>]  18.16MB
fe438658cde3: Layer already exists
5a06aafa23bc: Layer already exists
5a5177ddd844: Layer already exists
2f0df4d427b0: Layer already exists
c92e59d38453: Layer already exists
53b44a18496e: Layer already exists
66ff66f91ede: Layer already exists
4693057ce236: Layer already exists
```

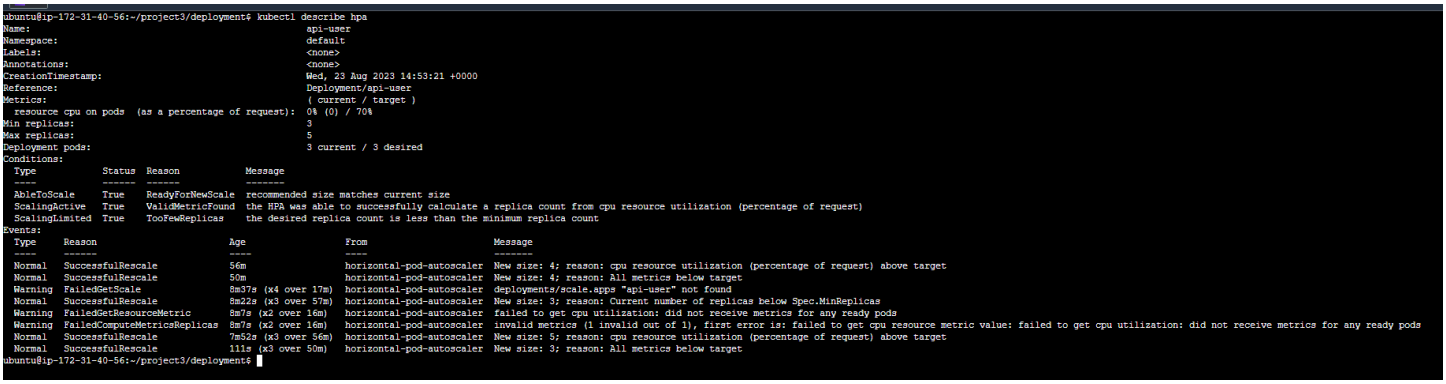
kubectl apply -f frontend.yaml

NhanNV13 – Nguyen Van Nhan

kubectl expose deployment frontend --type=LoadBalancer --name=frontend-ep



kubectl autoscale deployment api-user --cpu-percent=70 --min=3 --max=5
kubectl describe hpa



```
ubuntu@ip-172-31-40-56:~/project3/deployment$ kubectl describe hpa
Name:                                                     api-user
Namespace:                                                default
Labels:                                                   <none>
Annotations:                                              <none>
CreationTimestamp:                                        Wed, 23 Aug 2023 14:53:21 +0000
Reference:                                                Deployment/api-user
Metrics:                                                  ( current / target )
  resource cpu on pods  (as a percentage of request):     0% (0) / 70%
Min replicas:                                             3
Max replicas:                                             5
Deployment pods:                                          3 current / 3 desired
Conditions:
  Type            Status  Reason            Message
  ----            ------  ------            -------
  AbleToScale     True    ReadyForNewScale  recommended size matches current size
  ScalingActive   True    ValidMetricFound  the HPA was able to successfully calculate a replica count from cpu resource utilization (percentage of request)
  ScalingLimited  True    TooFewReplicas    the desired replica count is less than the minimum replica count
Events:
  Type     Reason                        Age                  From                       Message
  ----     ------                        ----                 ----                       -------
  Normal   SuccessfulRescale             56m                  horizontal-pod-autoscaler  New size: 4; reason: cpu resource utilization (percentage of request) above target
  Normal   SuccessfulRescale             50m                  horizontal-pod-autoscaler  New size: 4; reason: All metrics below target
  Warning  FailedGetScale                8m37s (x4 over 17m)  horizontal-pod-autoscaler  deployments/scale.apps "api-user" not found
  Normal   SuccessfulRescale             8m22s (x3 over 57m)  horizontal-pod-autoscaler  New size: 3; reason: Current number of replicas below Spec.MinReplicas
  Warning  FailedGetResourceMetric       8m7s (x2 over 16m)   horizontal-pod-autoscaler  failed to get cpu utilization: did not receive metrics for any ready pods
  Warning  FailedComputeMetricsReplicas  8m7s (x2 over 16m)   horizontal-pod-autoscaler  invalid metrics (1 invalid out of 1), first error is: failed to get cpu resource metric value: failed to get cpu utilization: did not receive metrics for any ready pods
  Normal   SuccessfulRescale             7m52s (x3 over 56m)  horizontal-pod-autoscaler  New size: 5; reason: cpu resource utilization (percentage of request) above target
  Normal   SuccessfulRescale             111s (x3 over 50m)   horizontal-pod-autoscaler  New size: 3; reason: All metrics below target
ubuntu@ip-172-31-40-56:~/project3/deployment$
```

Screenshot submission: here