

DEployment of microservices

Rajeswari S (2020CFSE018)



August 28, 2021

BITS PILANI

Contents

[1. System Architecture: 3](#_Toc81066343)

[**2.** **AWS Configuration** 3](#_Toc81066344)

[**2.1. Windows 10 - Install and configure AWS CLI** 3](#_Toc81066345)

[**2.2 Configure AWS Command Line using Security Credentials** 3](#_Toc81066346)

[**2.3 . Windows 10 - Install and configure kubectl** 4](#_Toc81066347)

[**2.4. eksctl on windows or linux** 5](#_Toc81066348)

[**3.** **Notification Service** 6](#_Toc81066349)

[**3.1 Introduction:** 6](#_Toc81066350)

[**3.2 deploy serverless application** 6](#_Toc81066351)

[3.3 sendMail -POST API Call 7](#_Toc81066352)

[3.3.1 Sample json message 8](#_Toc81066353)

[**4.** **Create Docker Image & Push Image** 8](#_Toc81066354)

[**4.1 build & push contactsinfo service** 8](#_Toc81066355)

[**4.2 build & push paymentsinfo service** 8](#_Toc81066356)

[**4.3 Build & push ReactJS app** 8](#_Toc81066357)

[5. Kubernetes Cluster Creation: 9](#_Toc81066358)

[**5.1 Create Cluster** 9](#_Toc81066359)

[**5.2 Link IAM role** 9](#_Toc81066360)

[**5.3 Create Node Group** 10](#_Toc81066361)

[**5.4. Get Cluster, node groups, nodes** 10](#_Toc81066362)

[**6.** **Create ALB Ingress Controller** 12](#_Toc81066363)

[**6.1** **Create a Kubernetes service account named alb-rc-ingress-controller in the kube-system namespace** 12](#_Toc81066364)

[**6.2 Create IAM Policy for ALB Ingress Controller** 14](#_Toc81066365)

[**6.3 Create an IAM role for the ALB Ingress Controller and attach the role to the service account** 15](#_Toc81066366)

[**6.4 Deploy ALB Ingress Controller** 17](#_Toc81066367)

[**6.5 Edit ALB Ingress Controller Manifest** 18](#_Toc81066368)

[**6.6 Verify our ALB Ingress Controller is running.** 18](#_Toc81066369)

[**7. Deployment of application** 18](#_Toc81066370)

[**7.11. Deploy microservice into EC2** 18](#_Toc81066371)

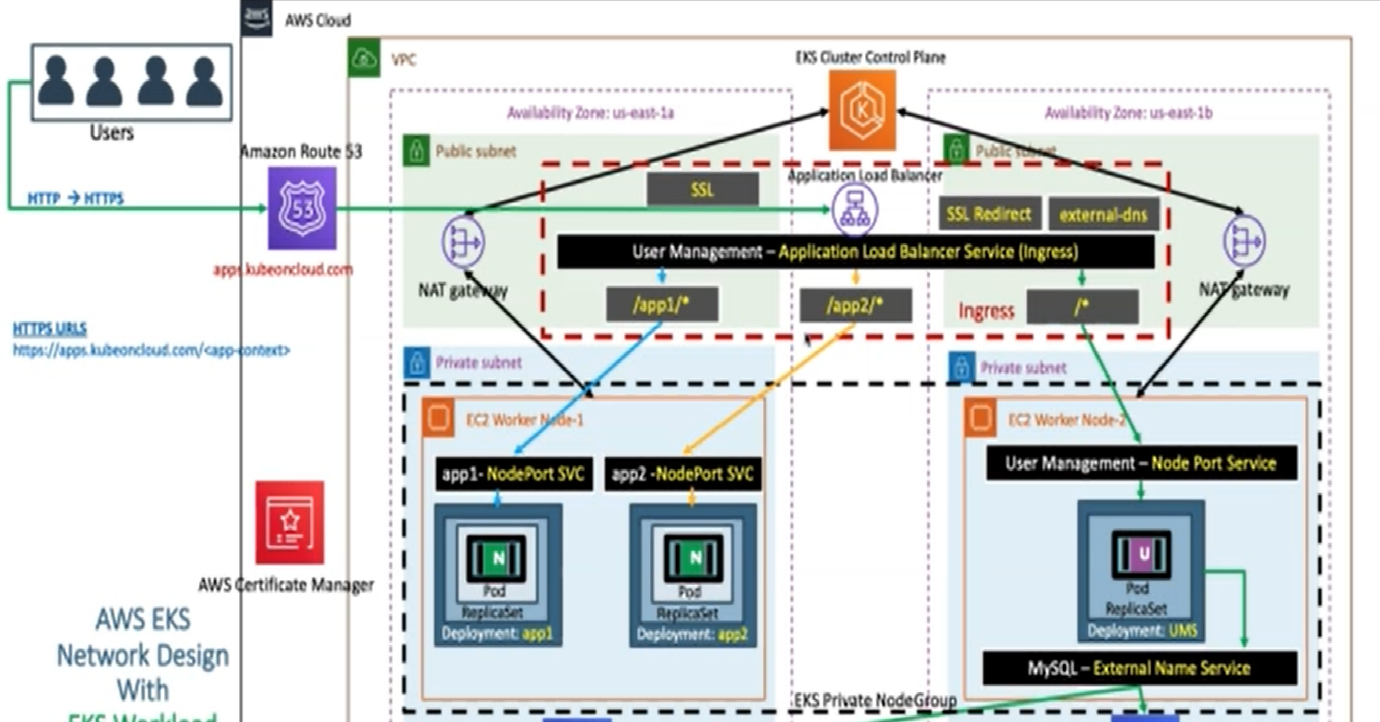
[**7.12. Deploy FrontEnd Application** 21](#_Toc81066372)

[**8. Create Application Load Balancer** 22](#_Toc81066373)

[**8.1 Create ALB kubernetes basic Ingress Manifest** 22](#_Toc81066374)

[**9. Register a Domain in Route53 (if not exists)** 29](#_Toc81066375)

# System Architecture:



# **AWS Configuration**

## **2.1. Windows 10 - Install and configure AWS CLI**

* The AWS CLI version 2 is supported on Windows XP or later.
* The AWS CLI version 2 supports only 64-bit versions of Windows.
* Download Binary: <https://awscli.amazonaws.com/AWSCLIV2.msi>
* Install the downloaded binary (standard windows install)

aws --version

## **2.2 Configure AWS Command Line using Security Credentials**

* Go to AWS Management Console --> Services --> IAM
* Select the IAM User: ec2\_tutorials
* **Important Note:** Use only IAM user to generate **Security Credentials**. Never ever use Root User. (Highly not recommended)
* Click on **Security credentials** tab
* Click on **Create access key**
* Copy Access ID and Secret access key
* Go to command line and provide the required details

aws configure

AWS Access Key ID [None]: ABCDEFGHIAZBERTUCNGG (Replace your creds when prompted)

AWS Secret Access Key [None]: uMe7fumK1IdDB094q2sGFhM5Bqt3HQRw3IHZzBDTm (Replace your creds when prompted)

Default region name [None]: ap-south-1

Default output format [None]: json

* Test if AWS CLI is working after configuring the above

aws ec2 describe-vpcs

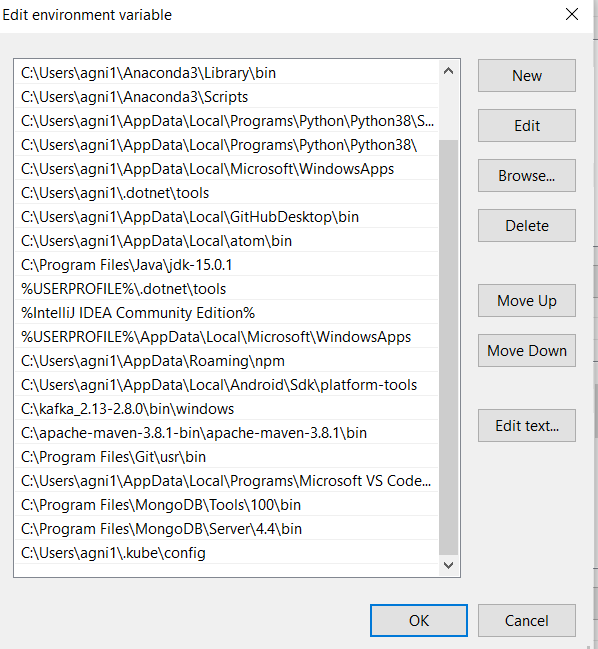
## **2.3 . Windows 10 - Install and configure kubectl**

* Install kubectl on Windows 10

mkdir kubectlbinary

cd kubectlbinary

curl -o kubectl.exe https://amazon-eks.s3.us-west-2.amazonaws.com/1.16.8/2020-04-16/bin/windows/amd64/kubectl.exe

* Update the system **Path** environment variable
*  Verify the kubectl client version

kubectl version --short --client

kubectl version --client

## **2.4. eksctl on windows or linux**

* For windows and linux OS, you can refer below documentation link.
* **Reference:** <https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html#installing-eksctl>

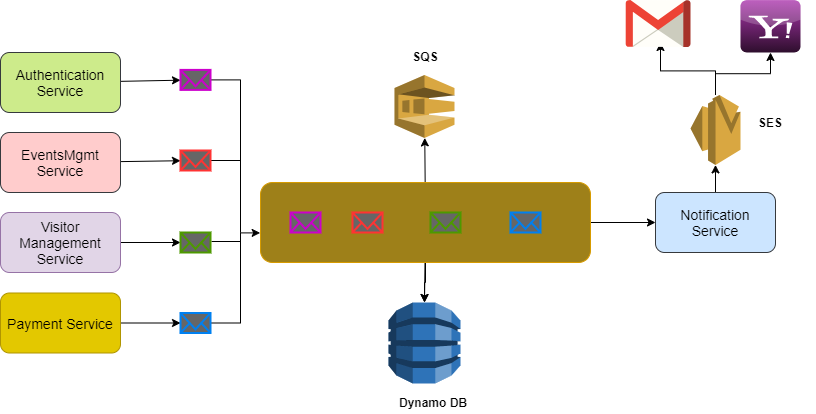
# **Notification Service**

## **3.1 Introduction:**

Email notification service is created using AWS-serverless method. SQS (Simple Queue Service), SES(Simple Email Service), Dynamo DB is used.

SQS: Generated email message is sent to SQS queue and details are recorded in Dynamo DB. sendMail – POST API is called for this purpose

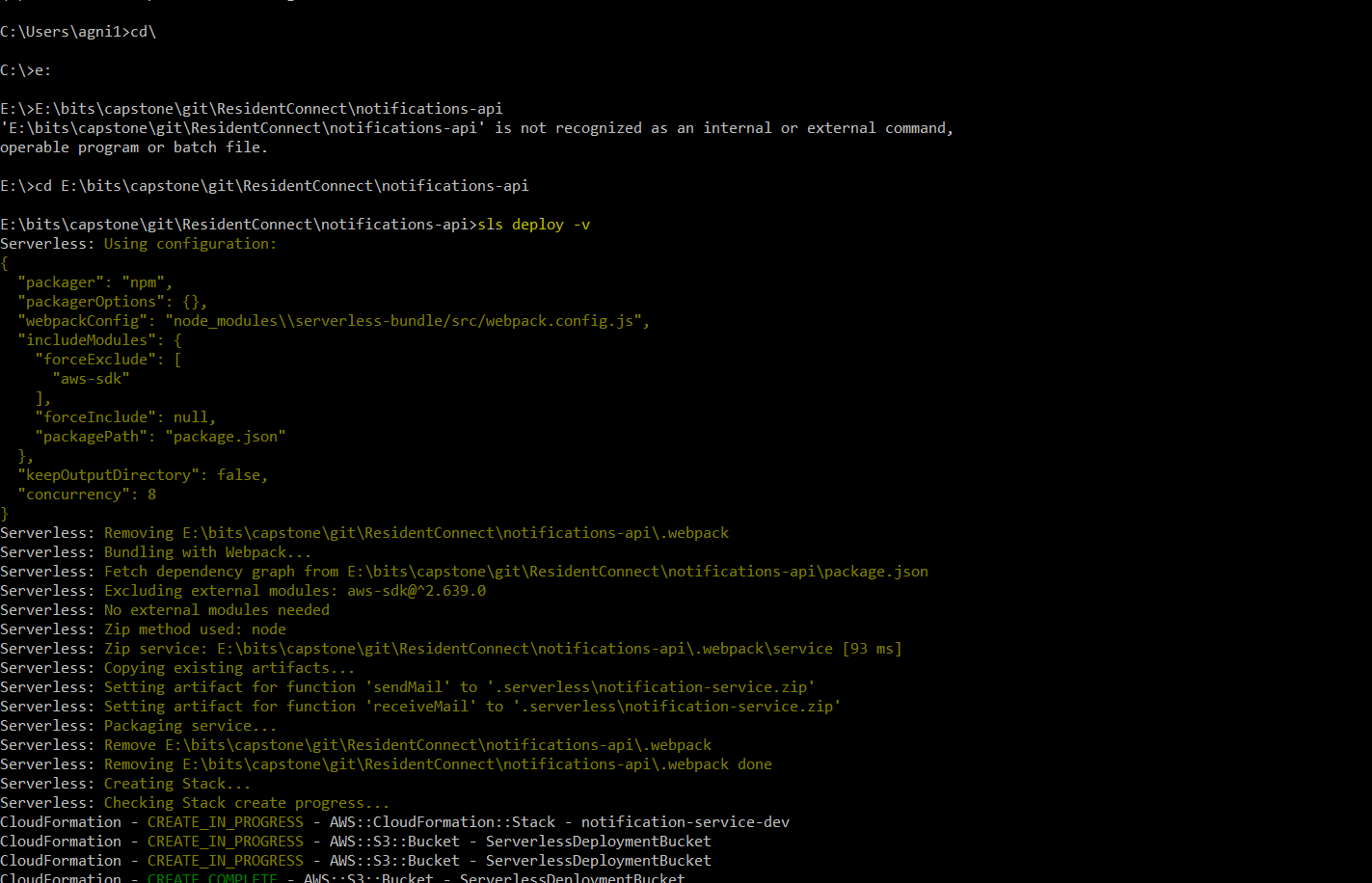
SES : notification service listens SQS using ‘receiveMail’ handler and generate email message. This generated email message sent to corresponding receipts.

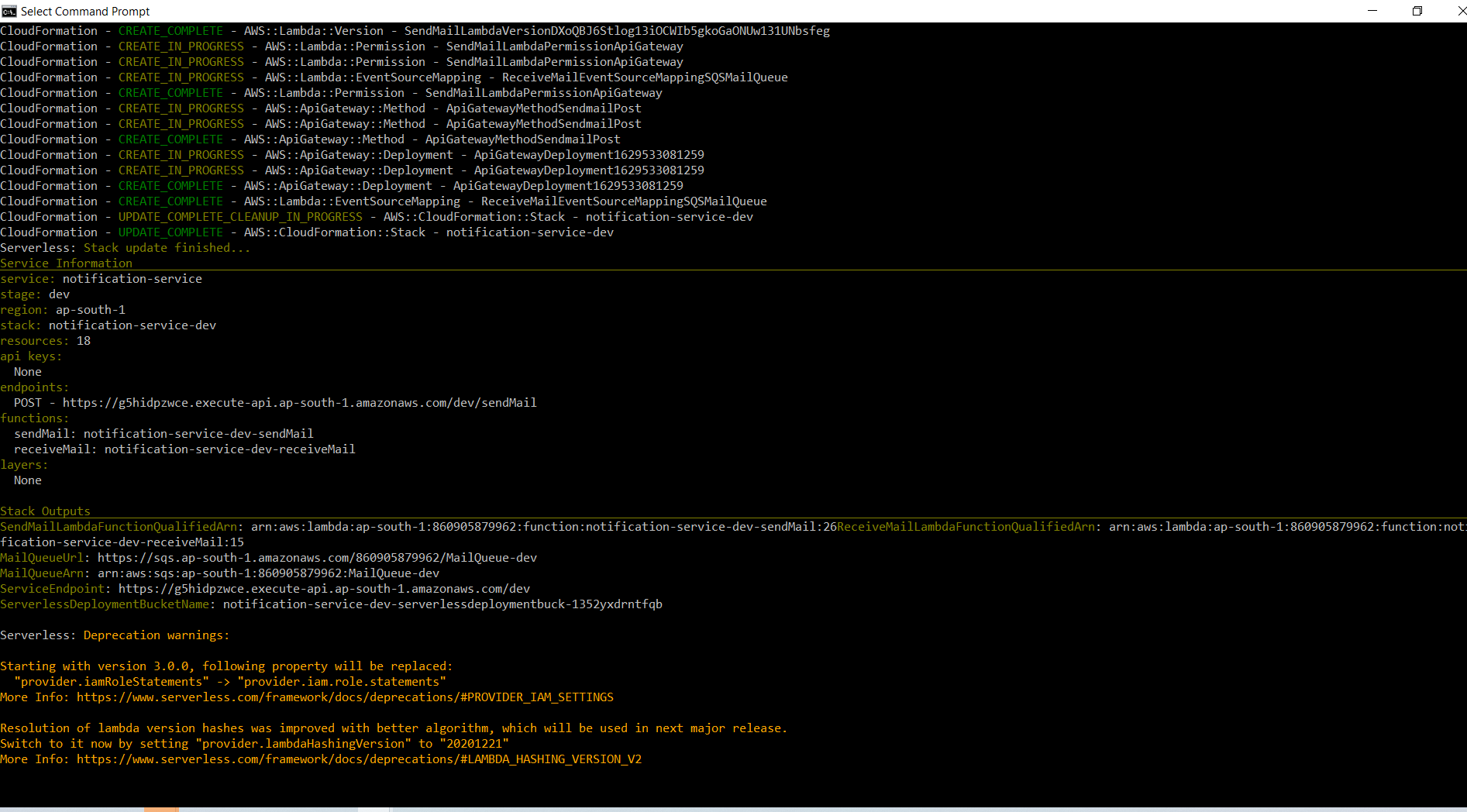


## **3.2 deploy serverless application**

Below comment is used for deploy serverless service.

*sls deploy -v*





Endpoint is created to sendMail (refer above screen for endpoint)

<https://g5hidpzwce.execute-api.ap-south-1.amazonaws.com/dev/sendMail>

## 3.3 sendMail -POST API Call

sendMail call is to use post json message in SQS queue. This message is used to generate email object for SES and SES uses to send the mail.

### 3.3.1 Sample json message

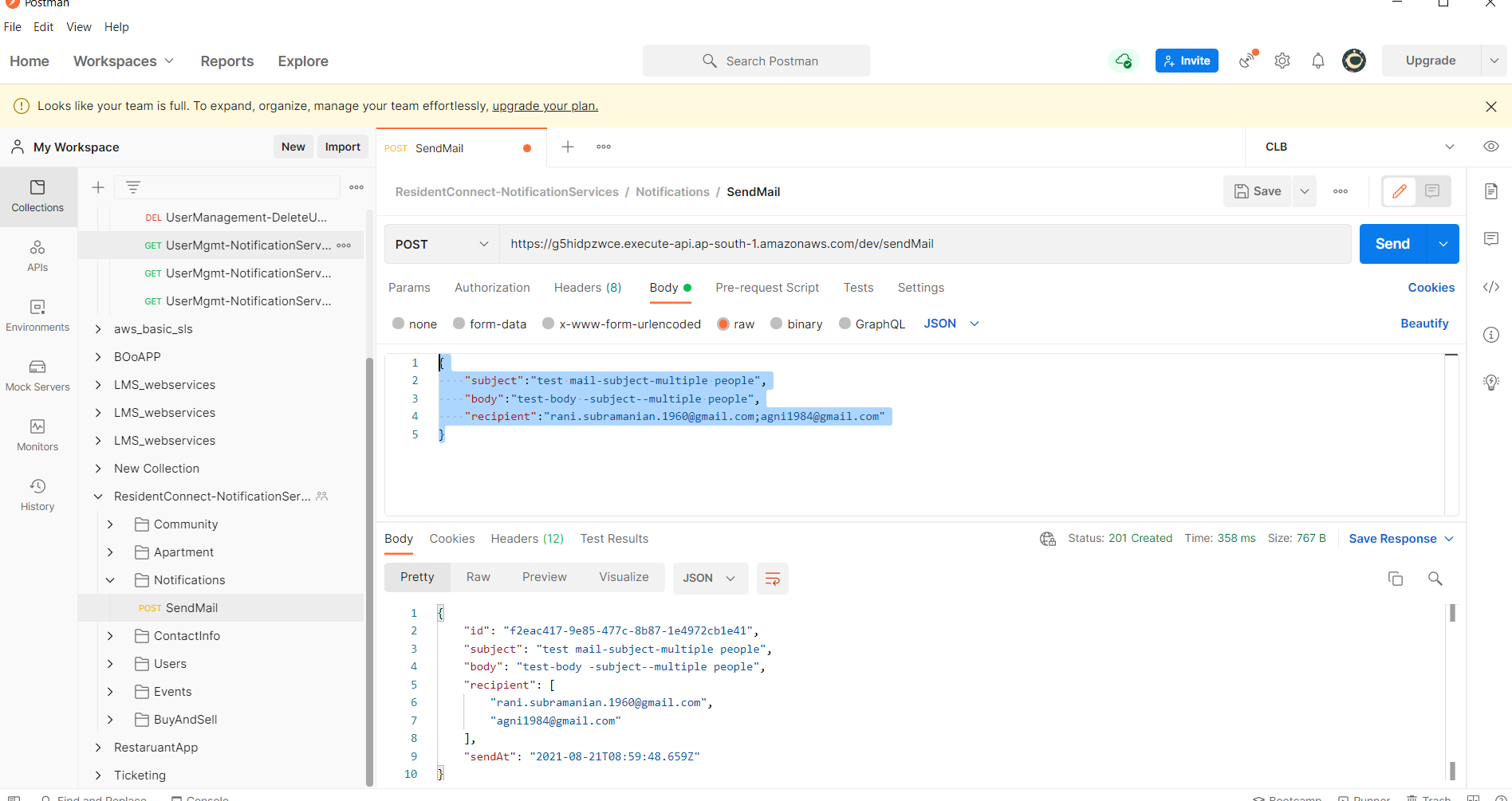
{

    "subject":"test mail-subject-multiple people",

    "body":"test-body -subject--multiple people",

    "recipient":"rani.subramanian.1960@gmail.com;agni1984@gmail.com"

}

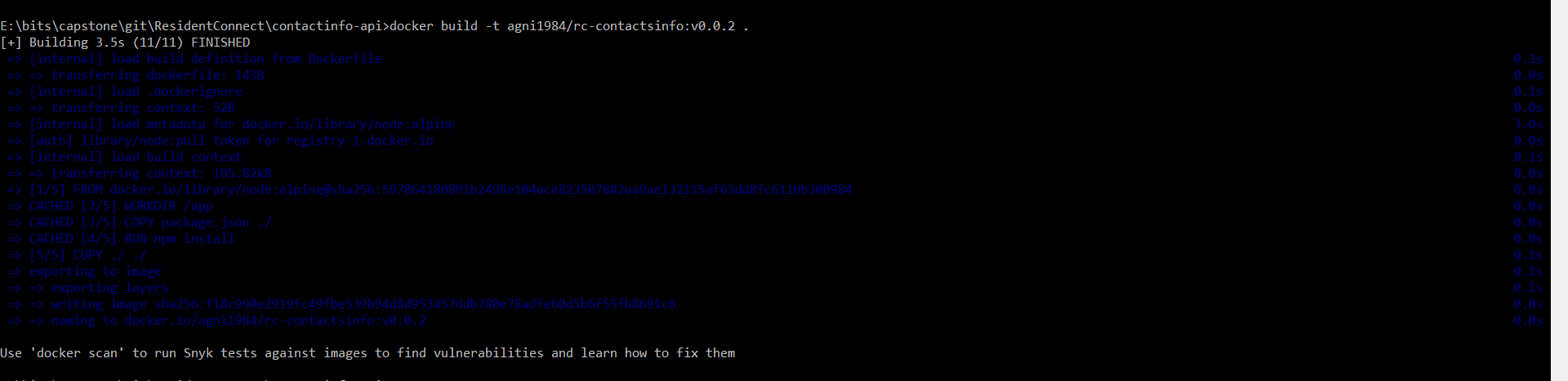


# **Create Docker Image & Push Image**

We are building 2 microservice docker image and push them.

## **4.1 build & push contactsinfo service**

docker build -t agni1984/rc-contactsinfo:v0.0.2 .



docker push agni1984/rc-contactsinfo:v0.0.2

## **4.2 build & push paymentsinfo service**

docker build -t agni1984/rc-paymentsinfo:v0.0.1 .

docker push agni1984/rc-paymentsinfo:v0.0.1

## **4.3 Build & push ReactJS app**

docker build -t agni1984/rc-frontend:v0.0.3 .

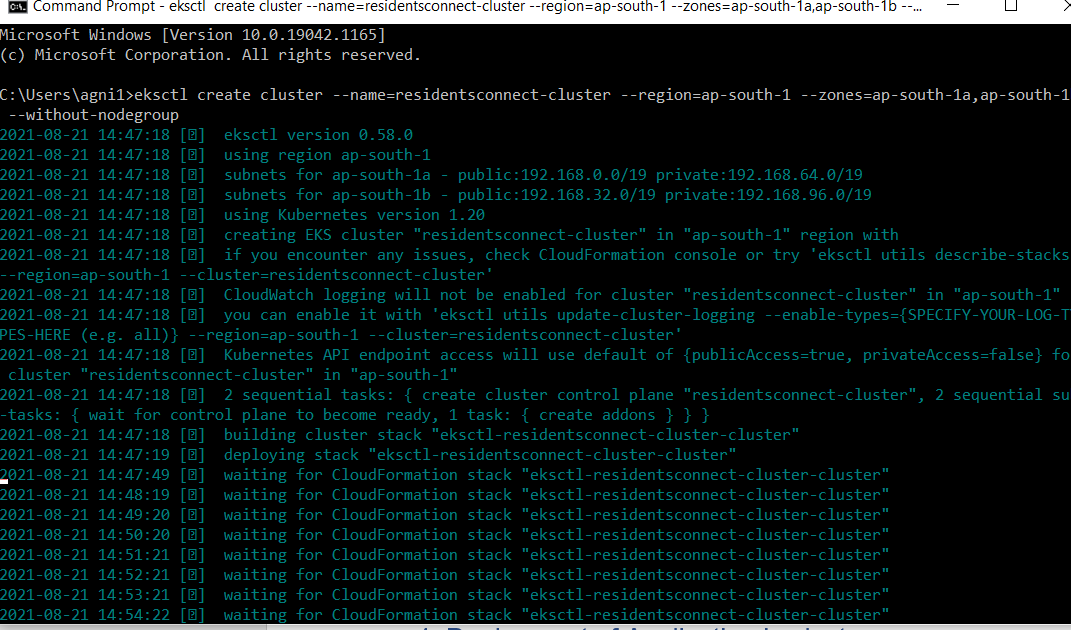
docker push agni1984/rc-frontend:v0.0.3 .

# Kubernetes Cluster Creation:

### **5.1 Create Cluster**

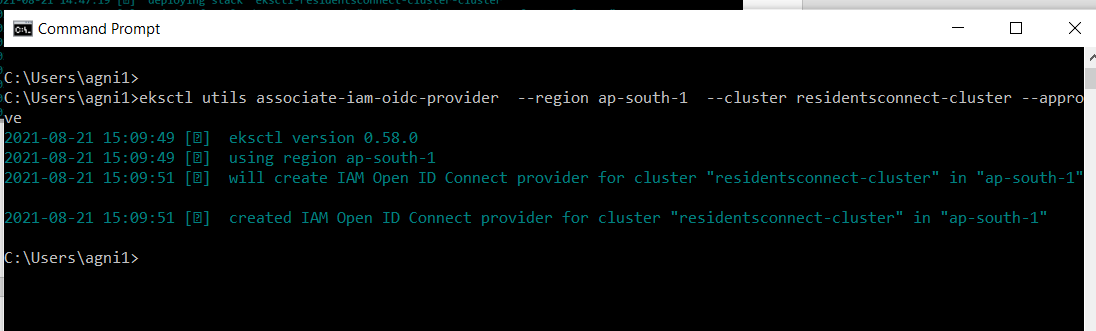
Create cluster called ‘cluster residentsconnect-cluster’

eksctl create cluster --name=residentsconnect-cluster --region=ap-south-1 --zones=ap-south-1a,ap-south-1b --without-nodegroup



### **5.2 Link IAM role**

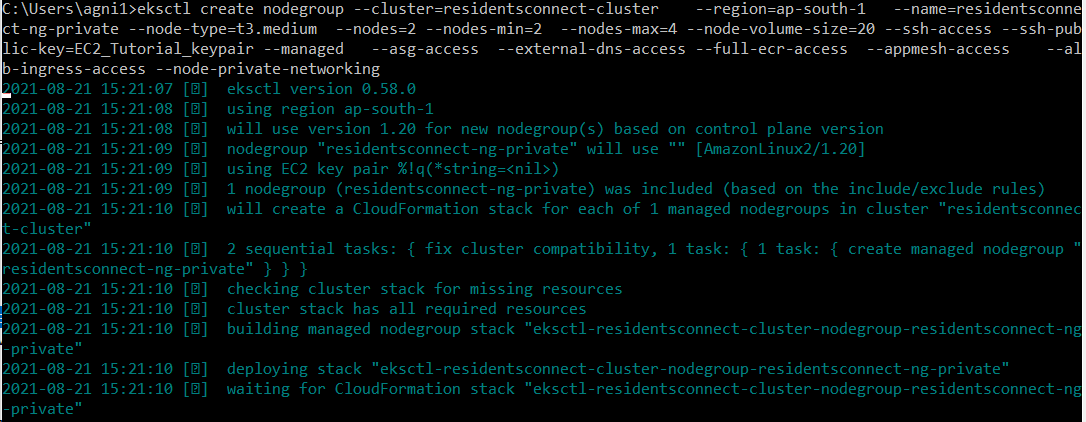
eksctl utils associate-iam-oidc-provider --region ap-south-1 --cluster residentsconnect-cluster -–approve



### **5.3 Create Node Group**

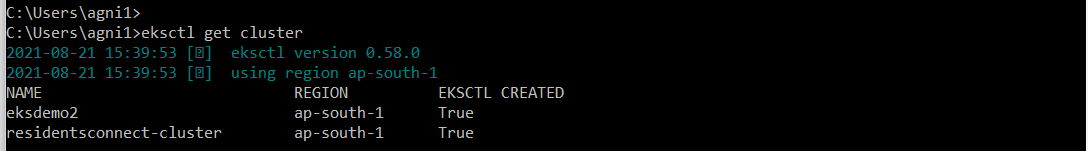
Create private node group called ‘residentsconnect-ng-private’

eksctl create nodegroup --cluster=residentsconnect-cluster --region=ap-south-1 --name=residentsconnect-ng-private --node-type=t3.medium --nodes=2 --nodes-min=2 --nodes-max=4 --node-volume-size=20 --ssh-access --ssh-public-key=EC2\_Tutorial\_keypair --managed --asg-access --external-dns-access --full-ecr-access --appmesh-access --alb-ingress-access --node-private-networking

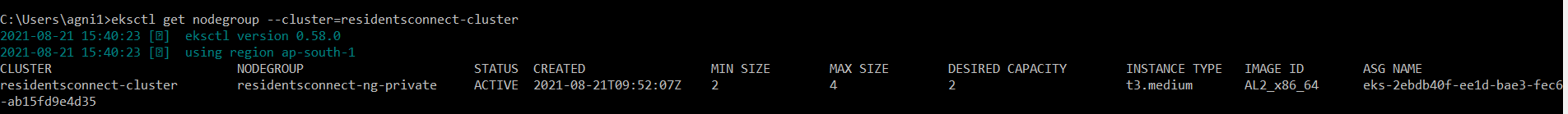


### **5.4. Get Cluster, node groups, nodes**

*eksctl get cluster*

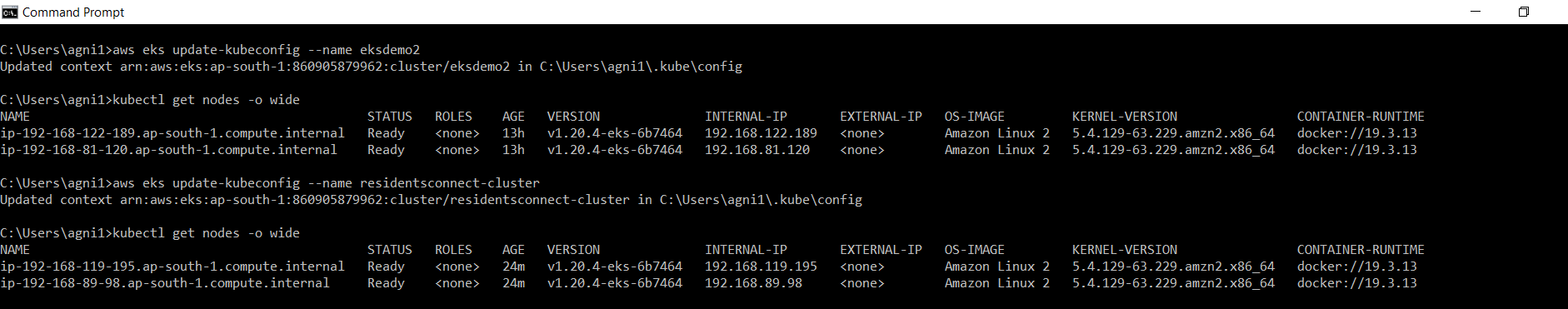


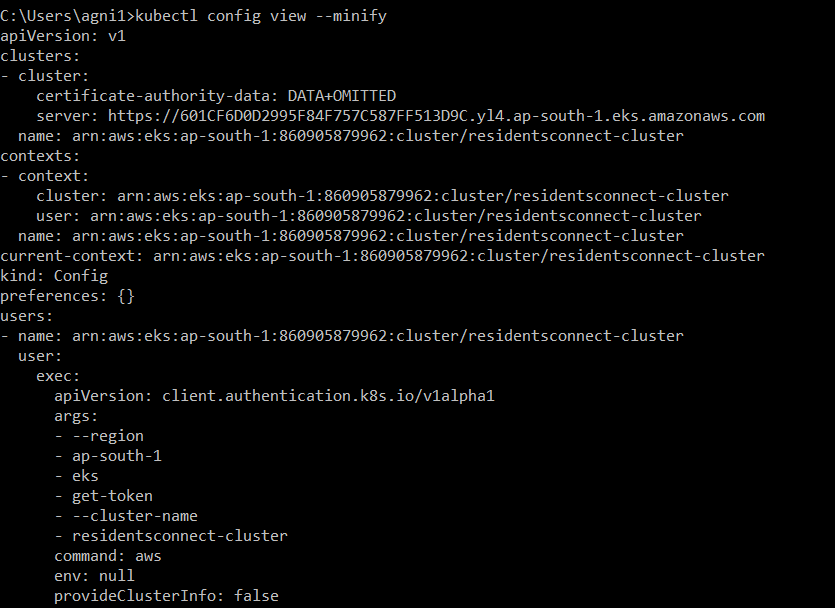
*eksctl get nodegroup --cluster=residentsconnect-cluster*



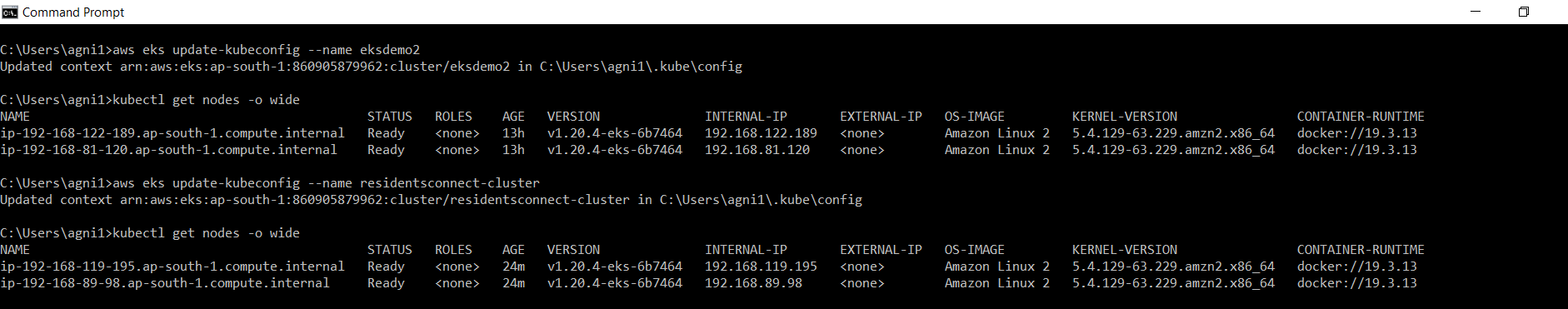
Switch to cluster

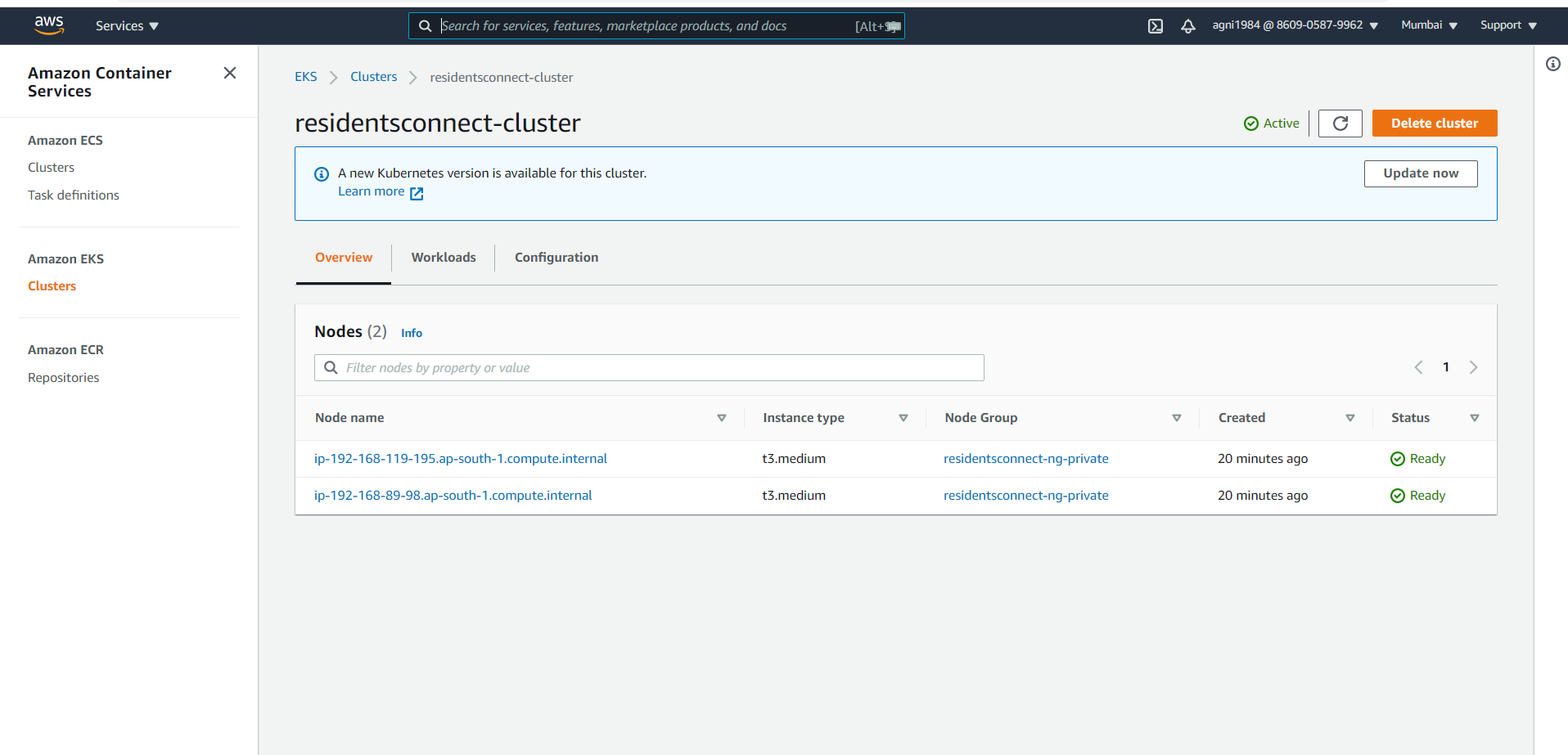
*aws eks update-kubeconfig --name residentsconnect-cluster*





*kubectl get nodes -o wide*



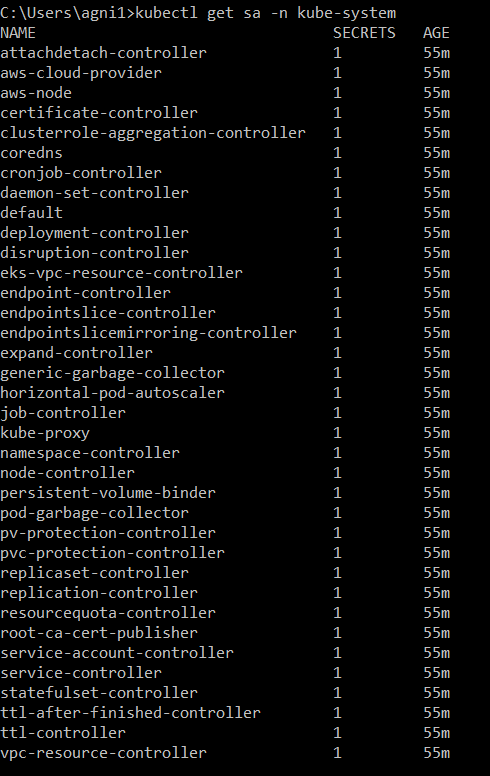


# **Create ALB Ingress Controller**

### **Create a Kubernetes service account named alb-rc-ingress-controller in the kube-system namespace**

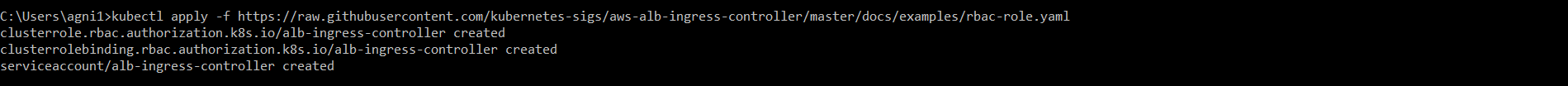
# List Service Accounts

kubectl get sa -n kube-system



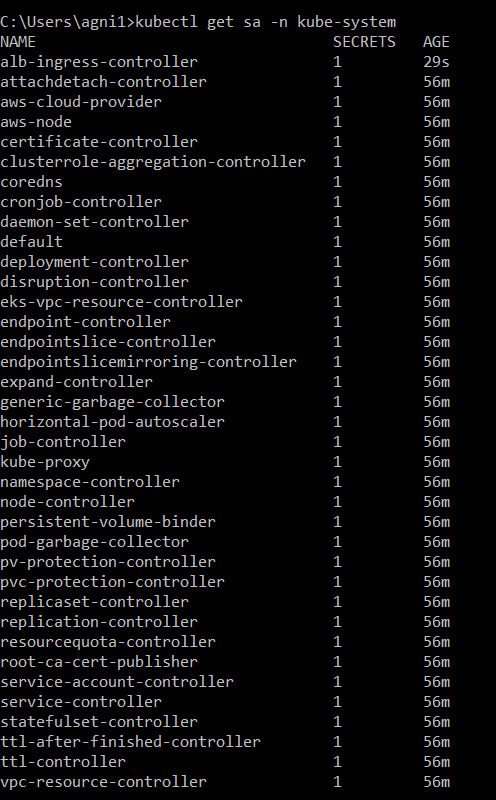
# Create ClusterRole, ClusterRoleBinding & ServiceAccount

kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/master/docs/examples/rbac-role.yaml



# List Service Accounts

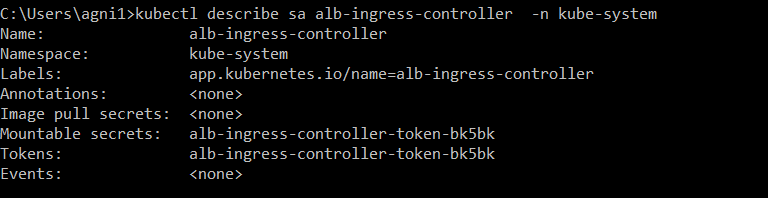
kubectl get sa -n kube-system



Now alb-ingress-controller added

# Describe Service Account alb-ingress-controller

kubectl describe sa alb-ingress-controller -n kube-system



### **6.2 Create IAM Policy for ALB Ingress Controller**

This IAM policy will allow our ALB Ingress Controller pod to make calls to AWS APIs

#### **6.2.1. IAM Policy Creation:**

Create manually using AWS management console and give full access to ELB

* Go to Services -> IAM -> Policies -> Create Policy
* Add ELB full access
  + Click on **Add Additional Permissions**
    - **Service:** ELB
    - **Actions:** All ELB actions (elasticloadbalancing:\*)
    - **Resources:** All Resources
* Click on **Review Policy**
  + **Name:** ALBIngressControllerIAMPolicy
  + **Description:** This IAM policy will allow our ALB Ingress Controller pod to make calls to AWS APIs
* Click on **Create Policy**



#### **6.2.2. Make a note of Policy ARN**

* Make a note of Policy ARN as we are going to use that in next step when creating IAM Role.

arn:aws:iam::860905879962:policy/ALBIngressControllerIAMPolicy

### **6.3 Create an IAM role for the ALB Ingress Controller and attach the role to the service account**

* Applicable only with eksctl managed clusters
* This command will create an AWS IAM role and bounds that to Kubernetes service account

# Replaced region, name, cluster and policy arn (Policy arn we took note in step-4.5.2)

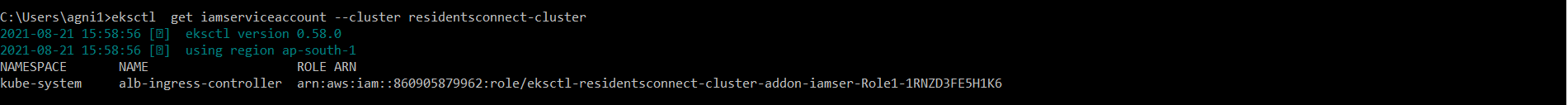
eksctl create iamserviceaccount --region ap-south-1 --name alb-ingress-controller --namespace kube-system --cluster residentsconnect-cluster --attach-policy-arn arn:aws:iam::860905879962:policy/ALBIngressControllerIAMPolicy --override-existing-serviceaccounts --approve



#### **6.3.1. Verify using eksctl cli**

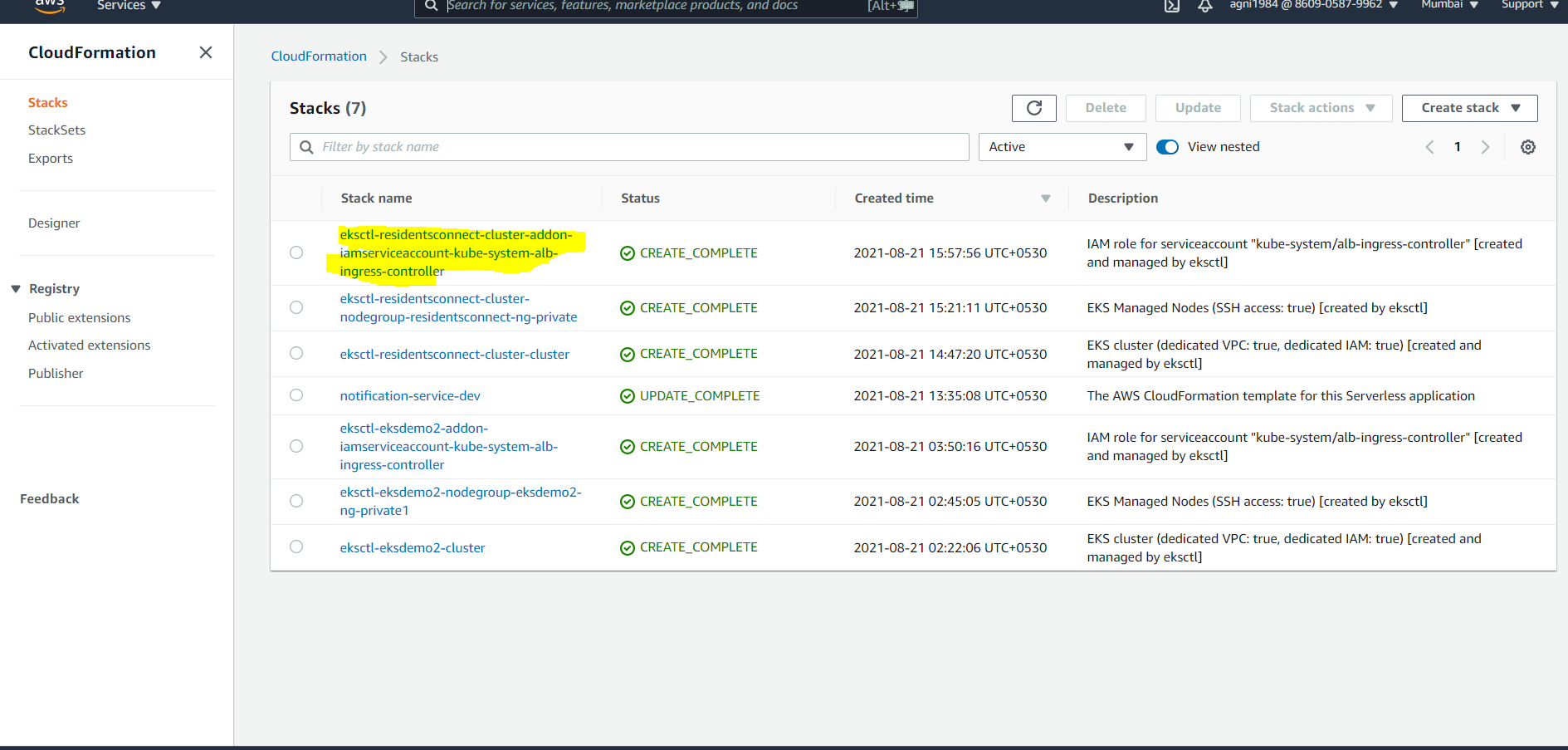
# Get IAM Service Account

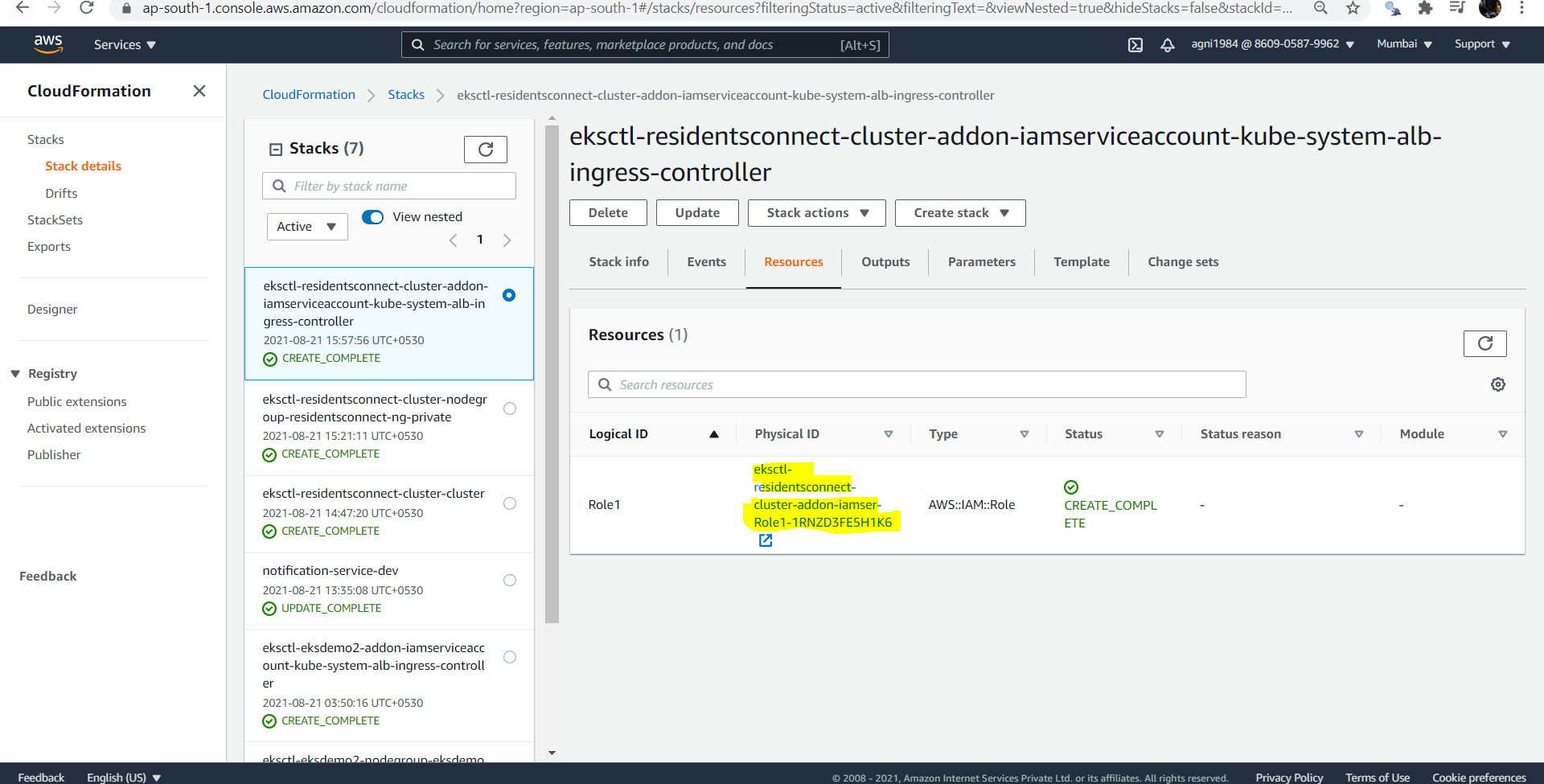
eksctl get iamserviceaccount --cluster residentsconnect-cluster



#### **6.3.2 Verify CloudFormation Template eksctl created & IAM Role**

* Goto Services -> CloudFormation
* **CFN Template Name:** eksctl-residentsconnect-cluster -addon-iamserviceaccount-kube-system-alb-ingress-controller
* Click on **Resources** tab
* Click on link in **Physical Id** to open the IAM Role
* Verify it has **ALBIngressControllerIAMPolicy** associated





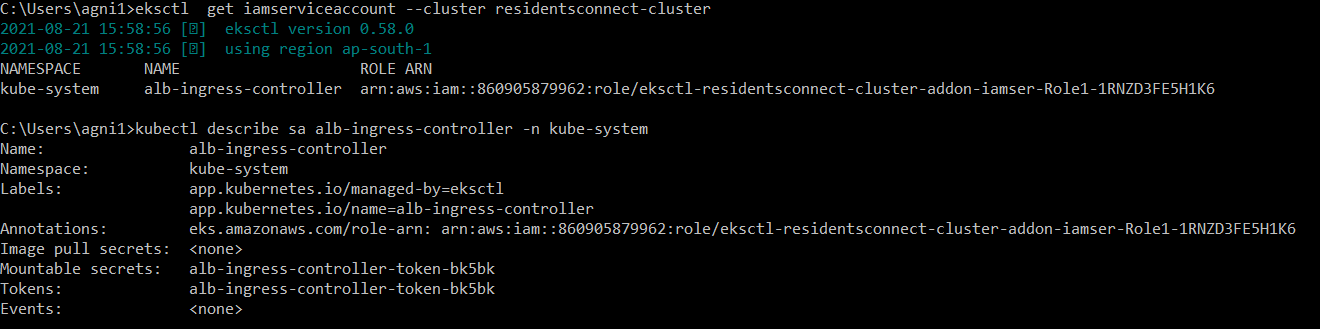


#### **6.3.3. Verify k8s Service Account**

# Describe Service Account alb-ingress-controller

kubectl describe sa alb-ingress-controller -n kube-system

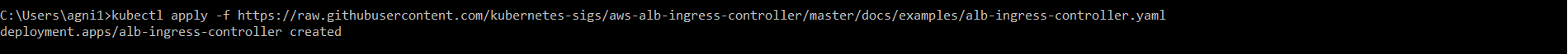
* **Observation:** You can see that newly created Role ARN is added in Annotations confirming that **AWS IAM role bound to a Kubernetes service account**



### **6.4 Deploy ALB Ingress Controller**

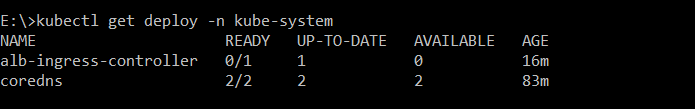
# Deploy ALB Ingress Controller

kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/master/docs/examples/alb-ingress-controller.yaml



# Verify Deployment

kubectl get deploy -n kube-system



### **6.5 Edit ALB Ingress Controller Manifest**

Edit ALB Ingress Controller manifest and add clustername field - --cluster-name= residentsconnect-cluster

# Edit Deployment

kubectl edit deployment.apps/alb-ingress-controller -n kube-system

# Replaced cluster-name with our cluster-name residentsconnect-cluster

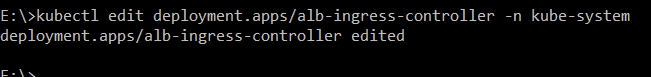
spec:

containers:

- args:

- --ingress-class=alb

- --cluster-name=residentsconnect-cluster



### **6.6 Verify our ALB Ingress Controller is running.**

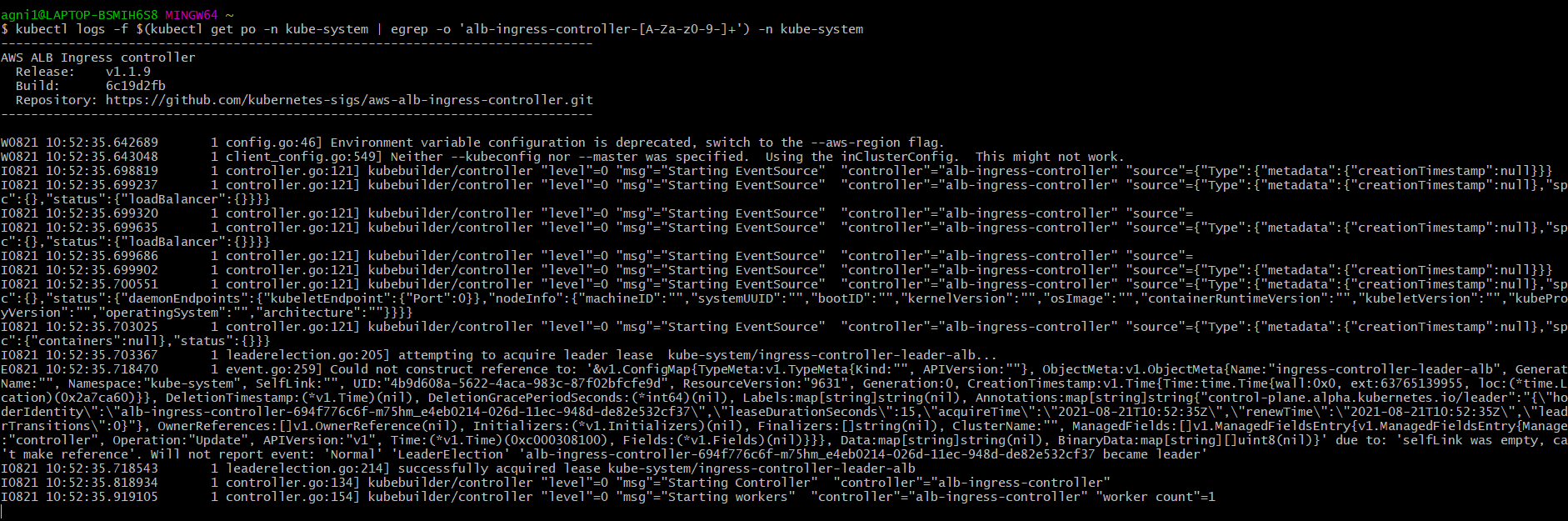
* Verify for the pod starting with alb-ingress-controller
* We will know if all our above steps are working or not in our next section **08-02-ALB-Ingress-Basic**, if ALB not created then we something is wrong.

# Verify if alb-ingress-controller pod is running

kubectl get pods -n kube-system

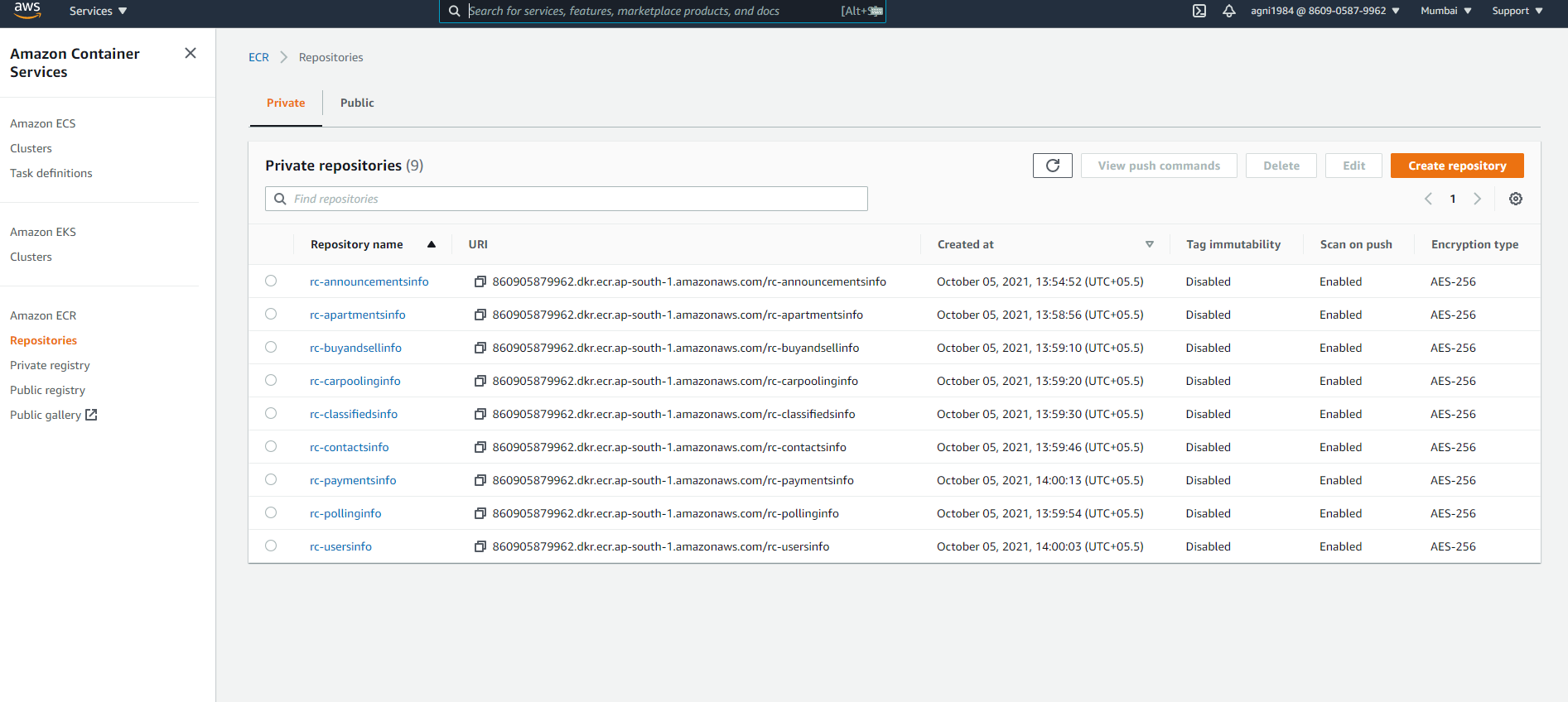
# Verify logs

kubectl logs -f $(kubectl get po -n kube-system | egrep -o 'alb-ingress-controller-[A-Za-z0-9-]+') -n kube-system



# **7. Deployment of application in ECR**

## **7.1 Create ECR repository for each microservices.**



## **7.2. Build and Push Image into ECR Respository:**

Run below command to push image into ECR

### **rc-announcementsinfo**

aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-announcementsinfo

docker build -t rc-announcementsinfo .

docker tag rc-announcements:latest 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-announcementsinfo:latest

docker push 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-announcementsinfo:latest

### **rc-apartmentsinfo**

aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-apartmentsinfo

docker build -t rc-apartmentsinfo .

docker tag rc-apartmentsinfo:latest 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-apartmentsinfo:latest

docker push 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-apartmentsinfo:latest

### **rc-buyandsellinfo**

aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-buyandsellinfo

docker build -t rc-buyandsellinfo .

docker tag rc-buyandsellinfo:latest 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-buyandsellinfo:latest

docker push 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-buyandsellinfo:latest

### **rc-carpoolinginfo**

aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-carpoolinginfo

docker build -t rc-carpoolinginfo .

docker tag rc-carpoolinginfo:latest 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-carpoolinginfo:latest

docker push 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-carpoolinginfo:latest

### **rc-classifiedsinfo**

aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-classifiedsinfo

docker build -t rc-classifiedsinfo .

docker tag rc-classifiedsinfo:latest 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-classifiedsinfo:latest

docker push 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-classifiedsinfo:latest

### **rc-contactsinfo**

aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-contactsinfo

docker build -t rc-contactsinfo .

docker tag rc-contactsinfo:latest 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-contactsinfo:latest

docker push 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-contactsinfo:latest

**rc-pollinginfo**

aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-pollinginfo

docker build -t rc-pollinginfo .

docker tag rc-pollinginfo:latest 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-pollinginfo:latest

docker push 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-pollinginfo:latest

**rc-usersinfo**

aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-usersinfo

docker build -t rc-usersinfo .

docker tag rc-usersinfo:latest 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-usersinfo:latest

docker push 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-usersinfo:latest

**rc-paymentsinfo**

aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-paymentsinfo

docker build -t rc-paymentsinfo .

docker tag rc-paymentsinfo:latest 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-paymentsinfo:latest

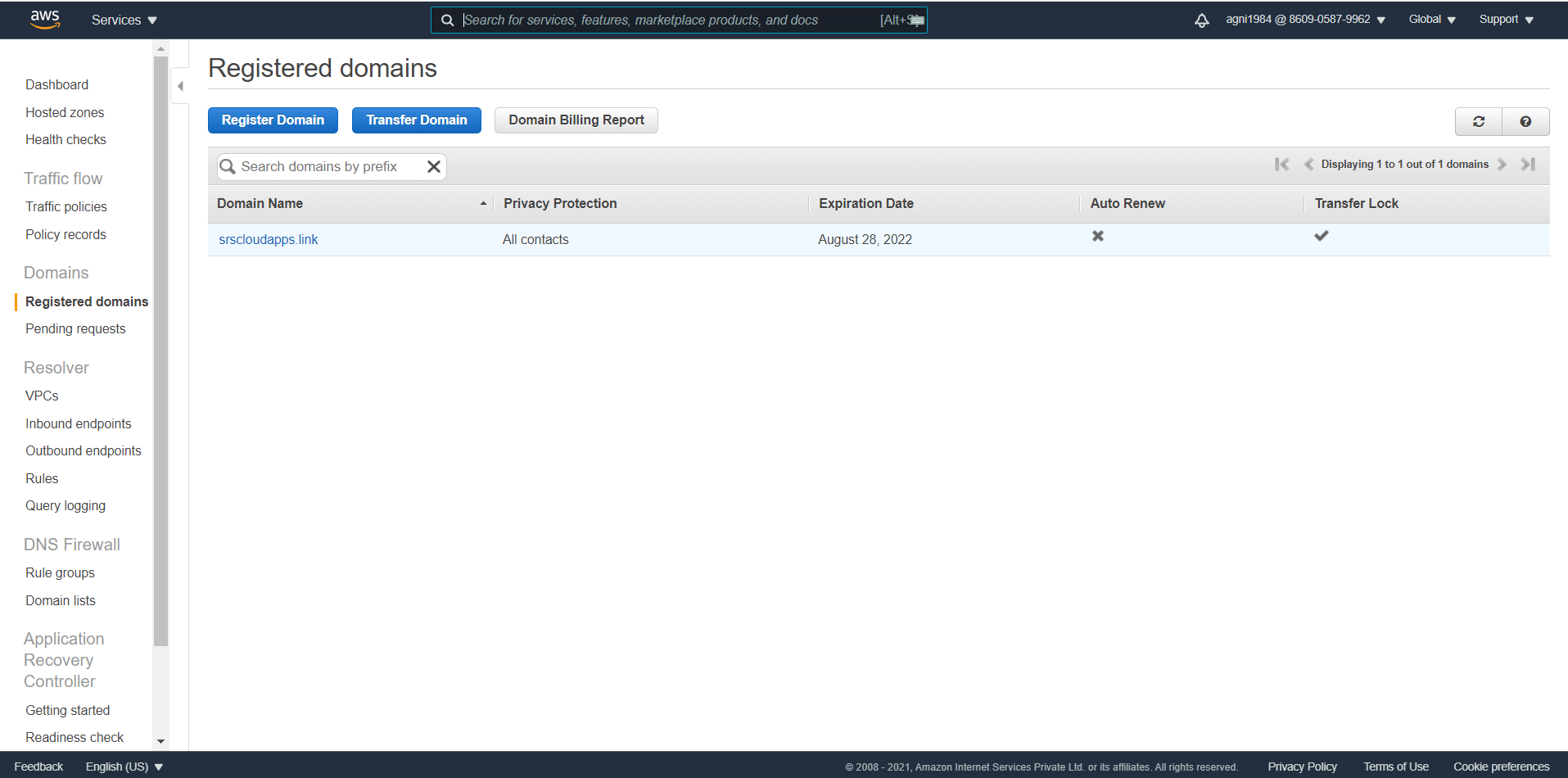
docker push 860905879962.dkr.ecr.ap-south-1.amazonaws.com/rc-paymentsinfo:latest

# **8.** **Create Application Load Balancer**

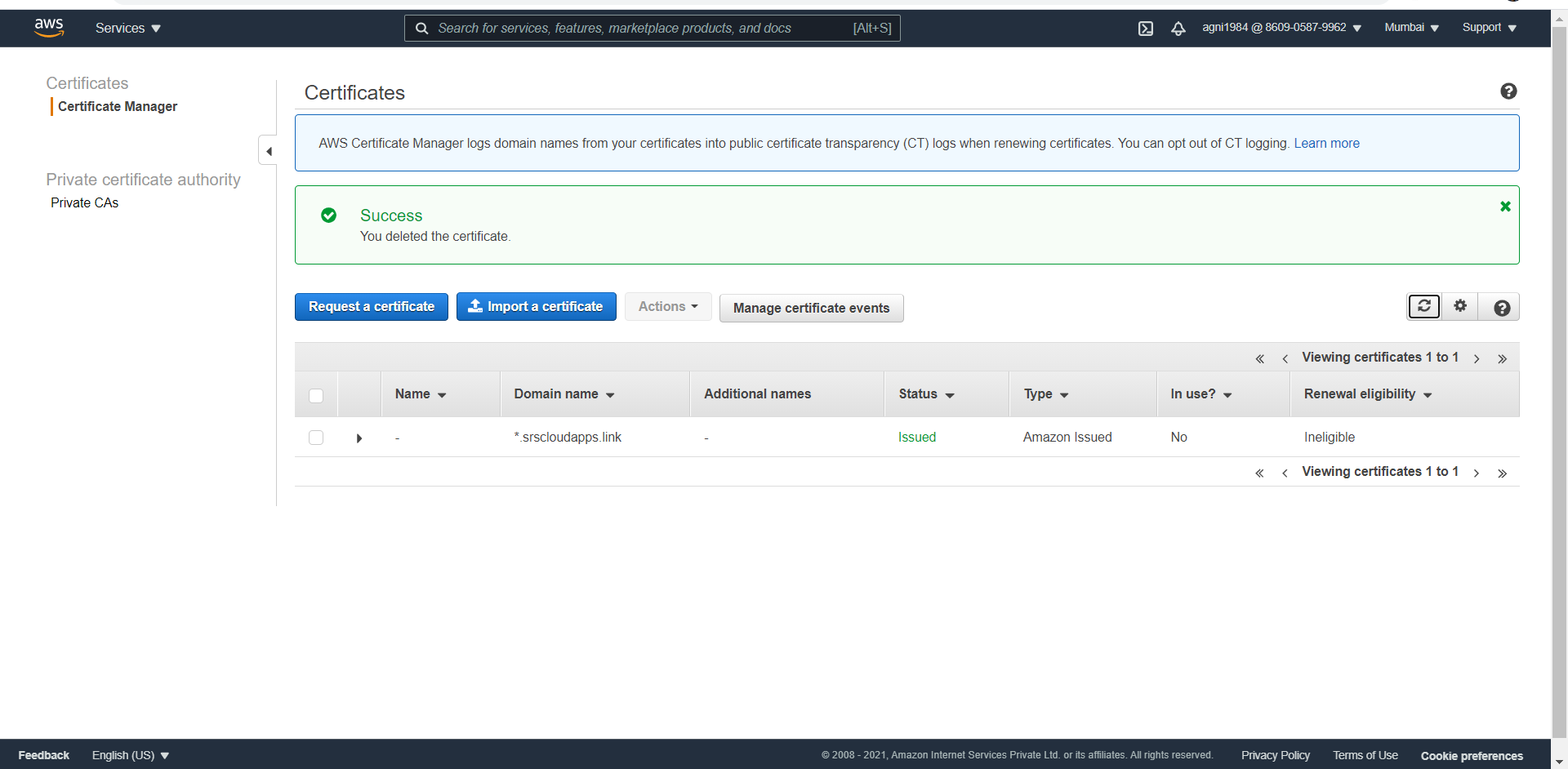
## 8.1. Create SSL Certificate

### **8.1.1 Register a Domain in Route53 (if not exists)**

* Goto Services -> Route53 -> Registered Domains
* Click on **Register Domain**
* Provide **desired domain: somedomain.com** and click on **check** (In my case its going to be kubeoncloud.com)
* Click on **Add to cart** and click on **Continue**
* Provide your **Contact Details** and click on **Continue**
* Enable Automatic Renewal
* Accept **Terms and Conditions**
* Click on **Complete Order**



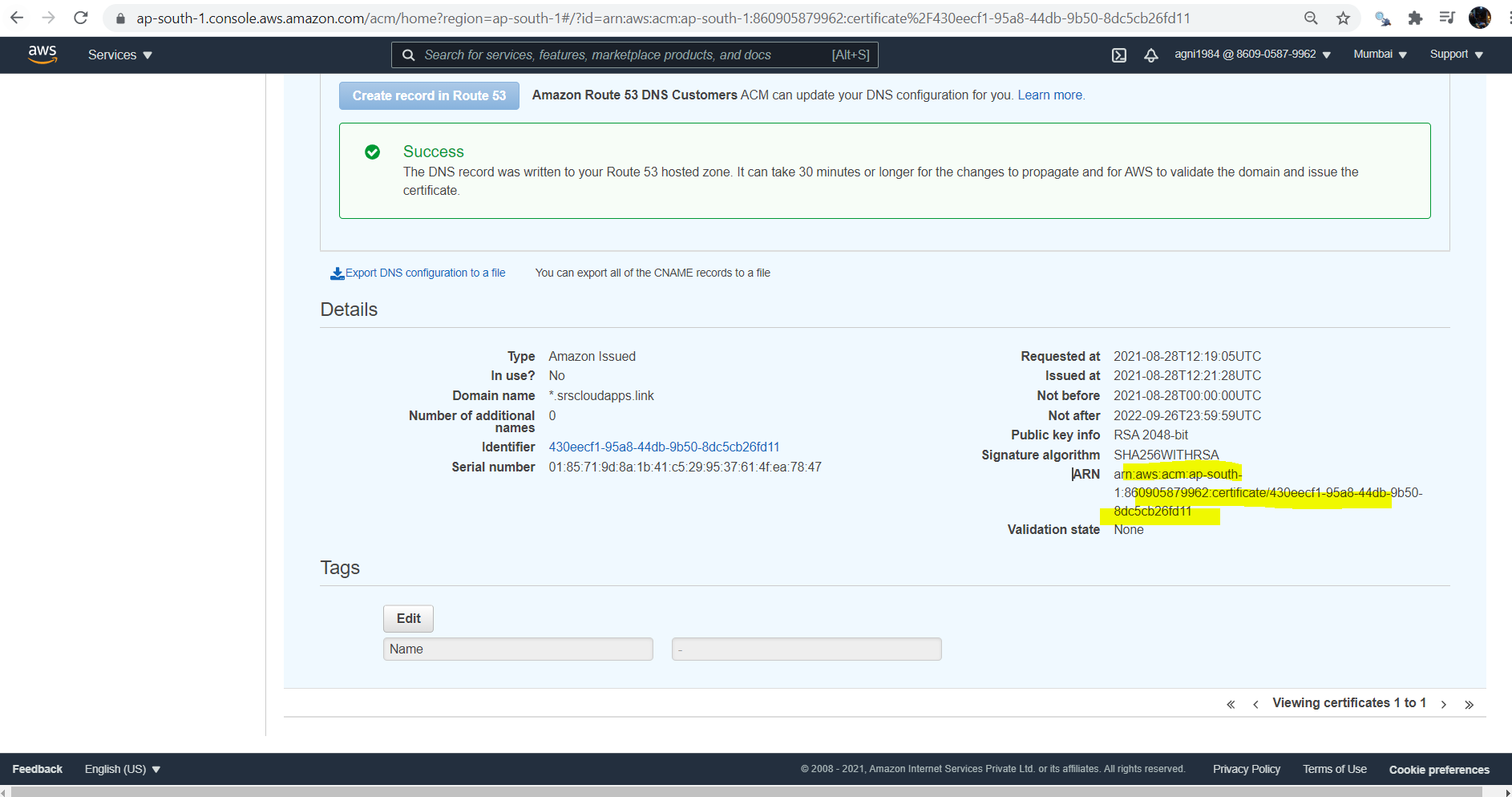
### **8.1.2 Create a SSL Certificate in Certificate Manager**

* Pre-requisite: You should have a registered domain in Route53
* Go to Services -> Certificate Manager -> Create a Certificate
* Click on **Request a Certificate**
  + Choose the type of certificate for ACM to provide: Request a public certificate
  + Add domain names: \*.yourdomain.
  + Select a Validation Method: **DNS Validation**
  + Click on **Confirm & Request**
* **Validation**
  + Click on **Create record in Route 53**
* Wait for 5 to 10 minutes and check the **Validation Status**
* 

### **8.1.3 Add annotations related to SSL**

SSL certification would be added in annotation section of Ingress controller. Get the ARN of newly created SSN certificate

arn:aws:acm:ap-south-1:860905879962:certificate/430eecf1-95a8-44db-9b50-8dc5cb26fd11



Add below 2 lines in ingress

SSL Setting - 1

## SSL Settings

alb.ingress.kubernetes.io/listen-ports: '[{"HTTPS":443}, {"HTTP":80}]'

alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:us-east-1:411686525067:certificate/8adf7812-a1af-4eae-af1b-ea425a238a67

#  kubectl apply -f ALB-Ingress.yml

# Annotations Reference:  https://kubernetes-sigs.github.io/aws-alb-ingress-controller/guide/ingress/annotation/

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

  name: ingress-rc-restapp-service

  labels:

    app: rc-restapp

  annotations:

    # Ingress Core Settings

    kubernetes.io/ingress.class: "alb"

    alb.ingress.kubernetes.io/scheme: internet-facing

    # Health Check Settings

    alb.ingress.kubernetes.io/healthcheck-protocol: HTTP

    alb.ingress.kubernetes.io/healthcheck-port: traffic-port

    #alb.ingress.kubernetes.io/healthcheck-path: /health-status

    alb.ingress.kubernetes.io/healthcheck-interval-seconds: '60'

    alb.ingress.kubernetes.io/healthcheck-timeout-seconds: '10'

    alb.ingress.kubernetes.io/success-codes: '200'

    alb.ingress.kubernetes.io/healthy-threshold-count: '2'

    alb.ingress.kubernetes.io/unhealthy-threshold-count: '10'

    ## SSL Settings

    alb.ingress.kubernetes.io/listen-ports: '[{"HTTPS":443}, {"HTTP":80}]'

    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:ap-south-1:860905879962:certificate/430eecf1-95a8-44db-9b50-8dc5cb26fd11

spec:

  rules:

    - http:

        paths:

          - path: /api/contacts/\*

            backend:

              serviceName: rc-contactsinfo-srv

              servicePort: 4001

          - path: /api/payments/\*

            backend:

              serviceName: rc-payments-srv

              servicePort: 4012

          - path: /\*

            backend:

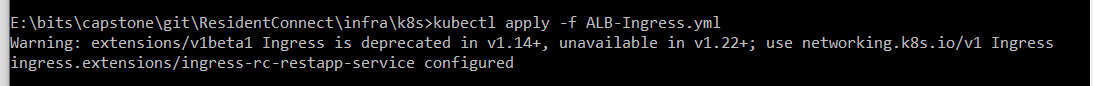
              serviceName: rc-frontend-srv

              servicePort: 3000

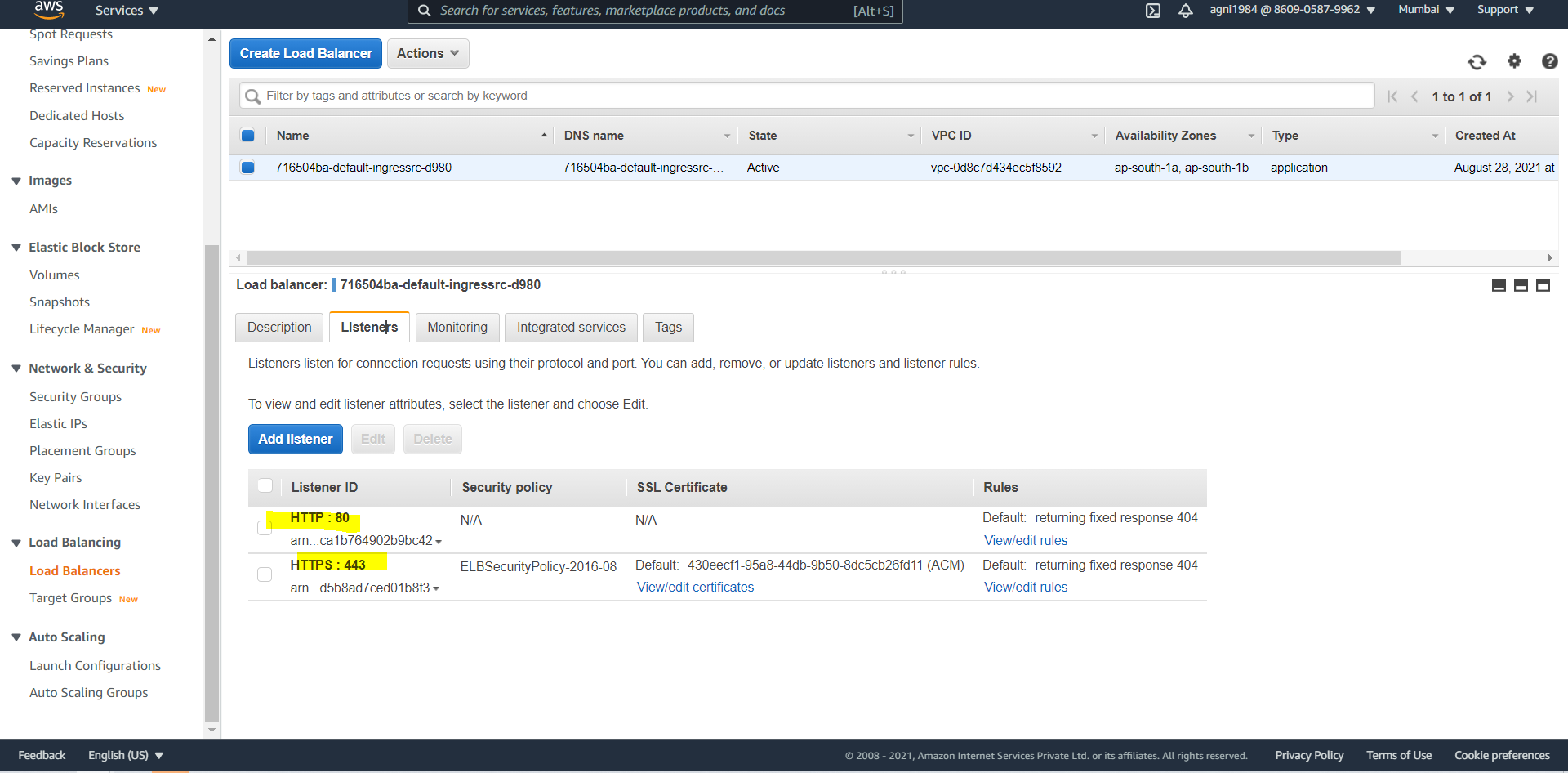
### **8.1.4 Deploy Load Balancer with SSL:**

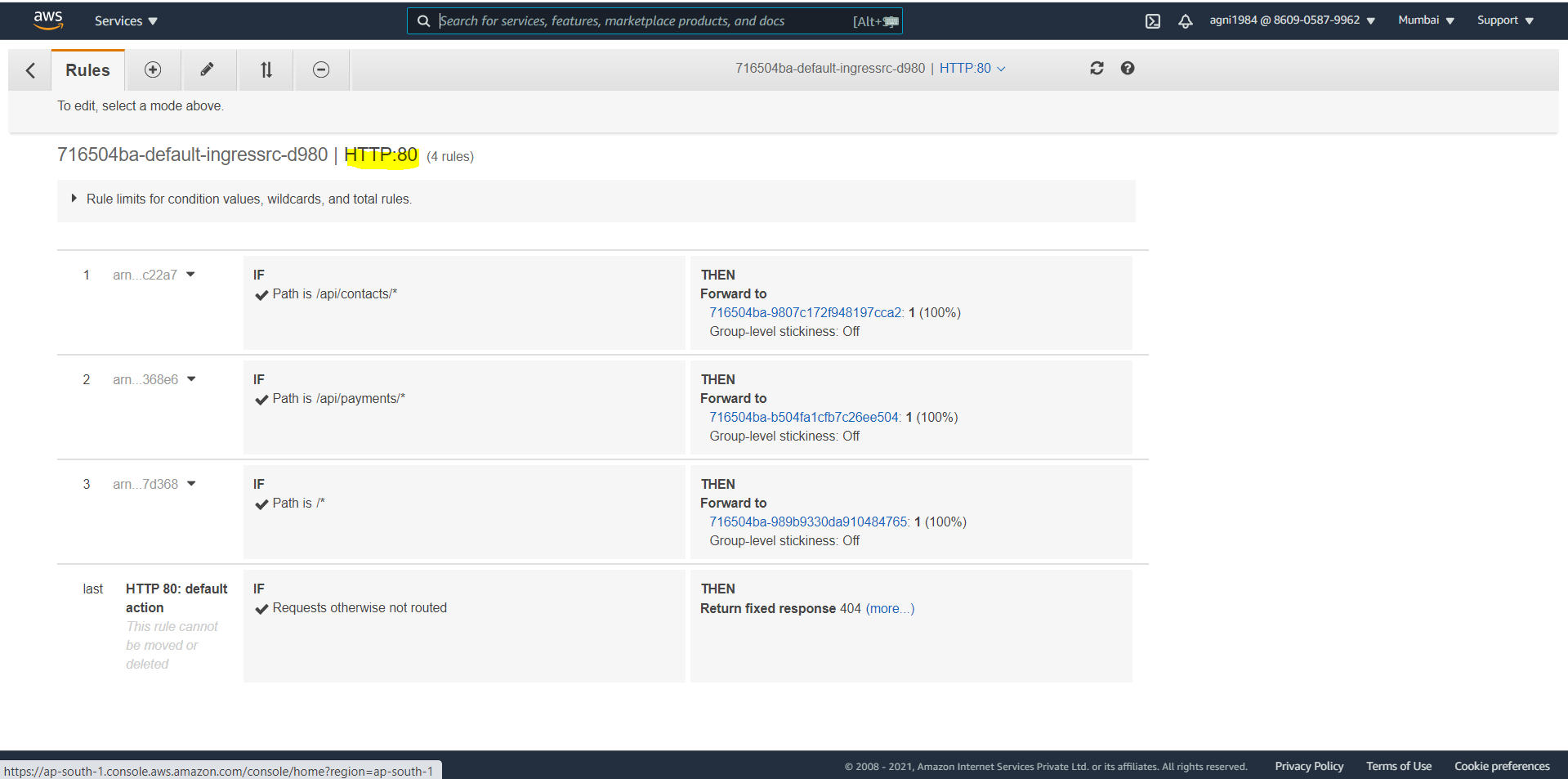
Redeploy load balancer by running below command,

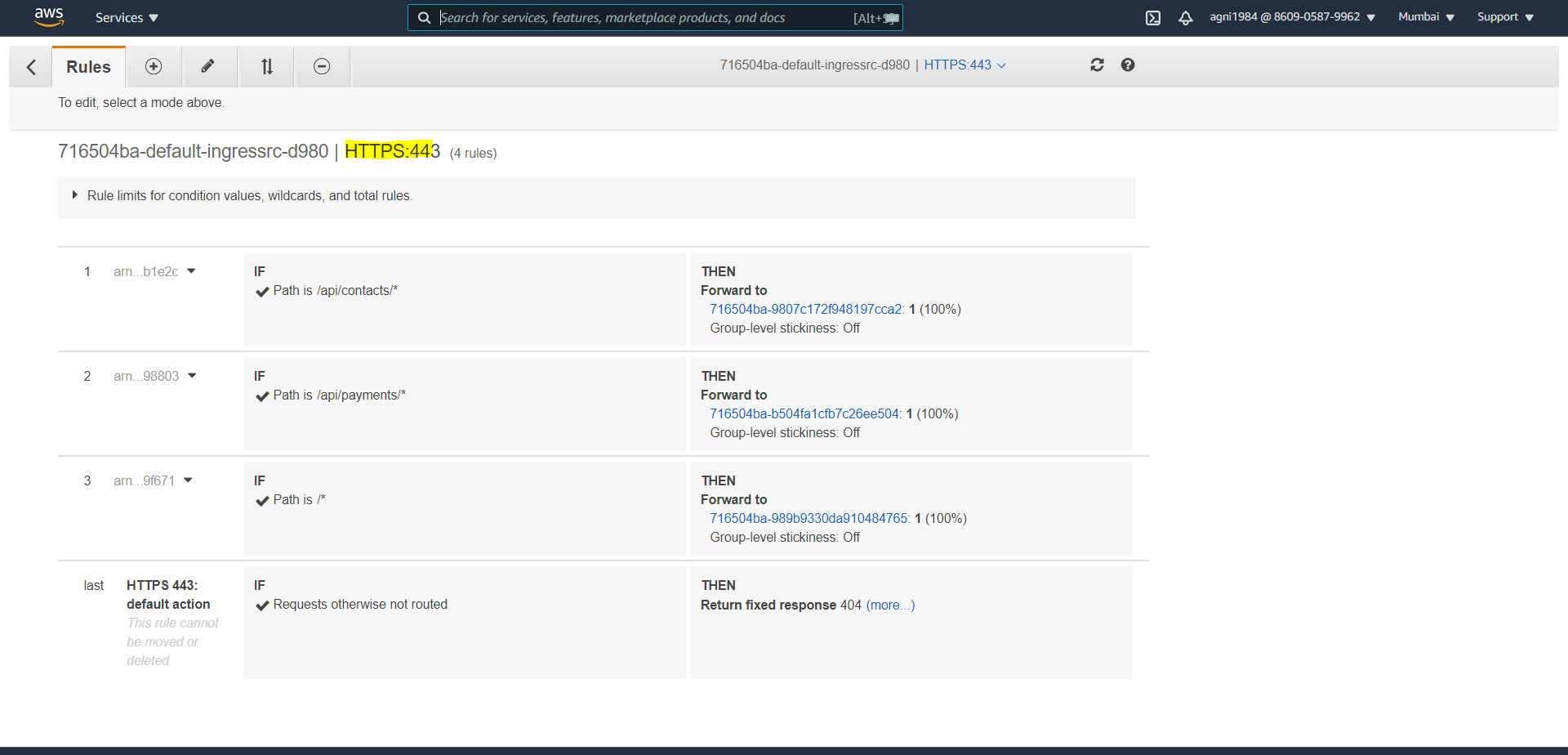
kubectl apply -f ALB-Ingress.yml

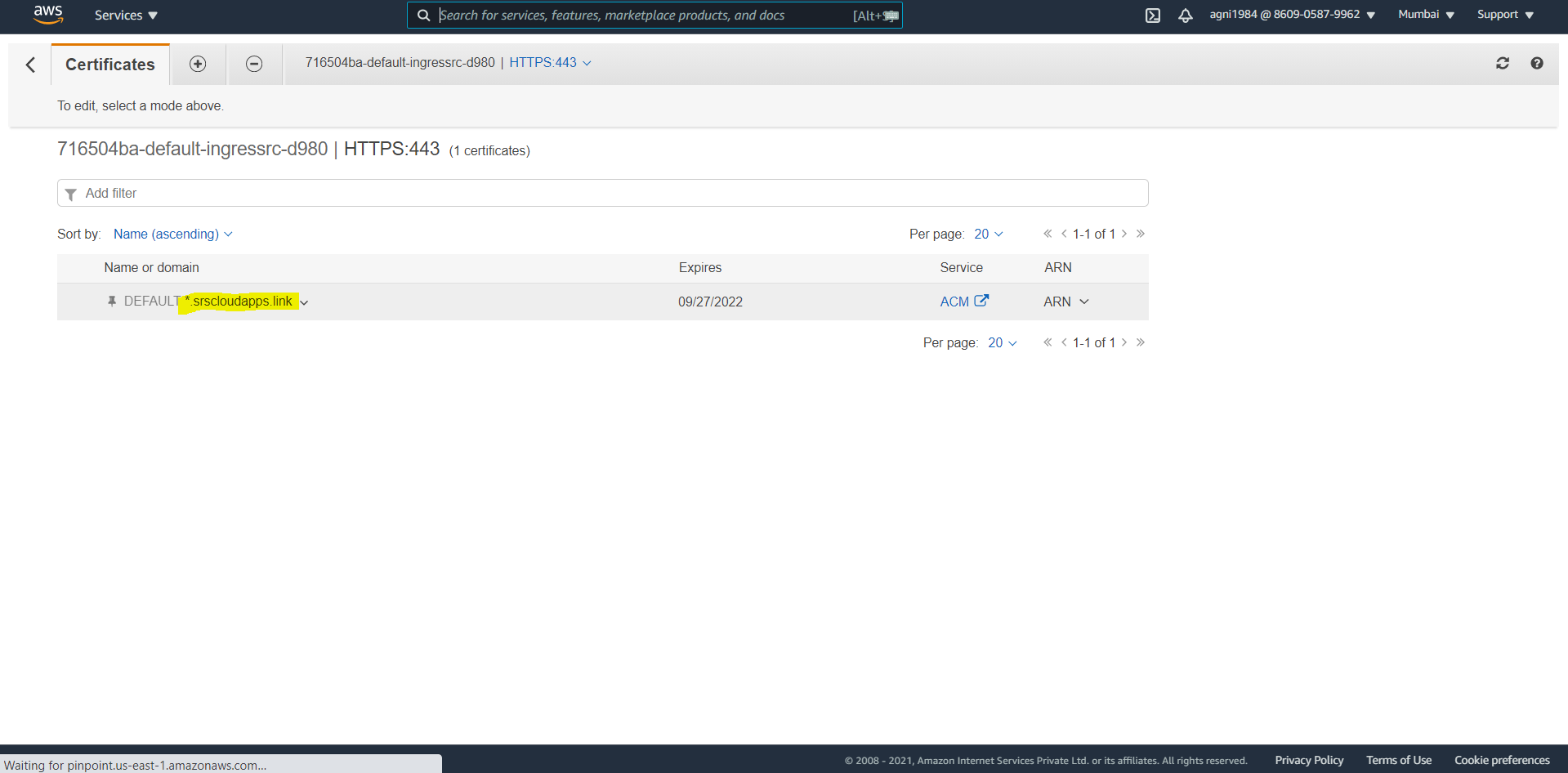


* **Verify**
  + Load Balancer - Listeneres (Verify both 80 & 443)
  + Load Balancer - Rules (Verify both 80 & 443 listeners)
  + Target Groups - Group Details (Verify Health check path)
  + Target Groups - Targets (Verify all 3 targets are healthy)
  + Verify ingress controller from kubectl

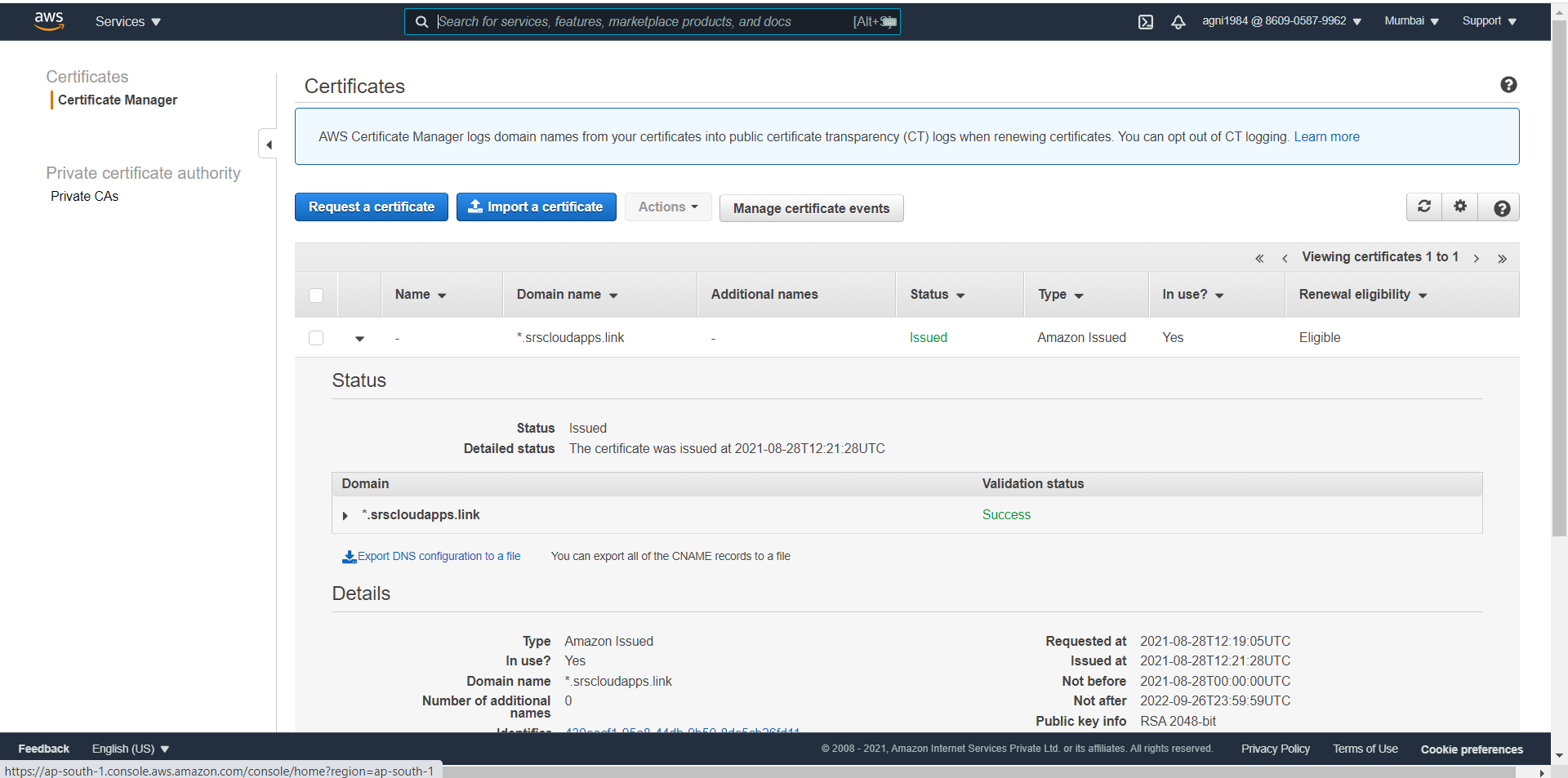


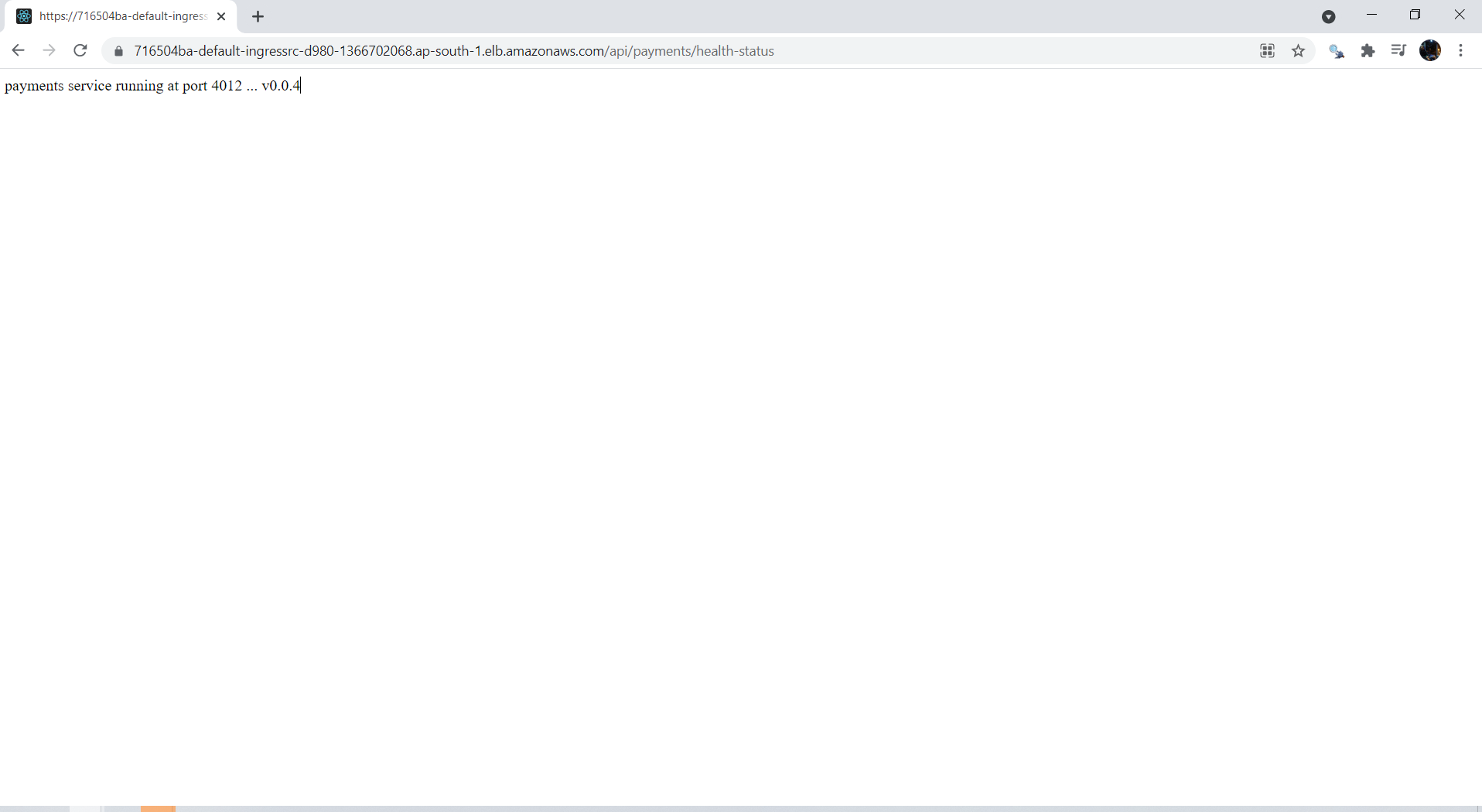


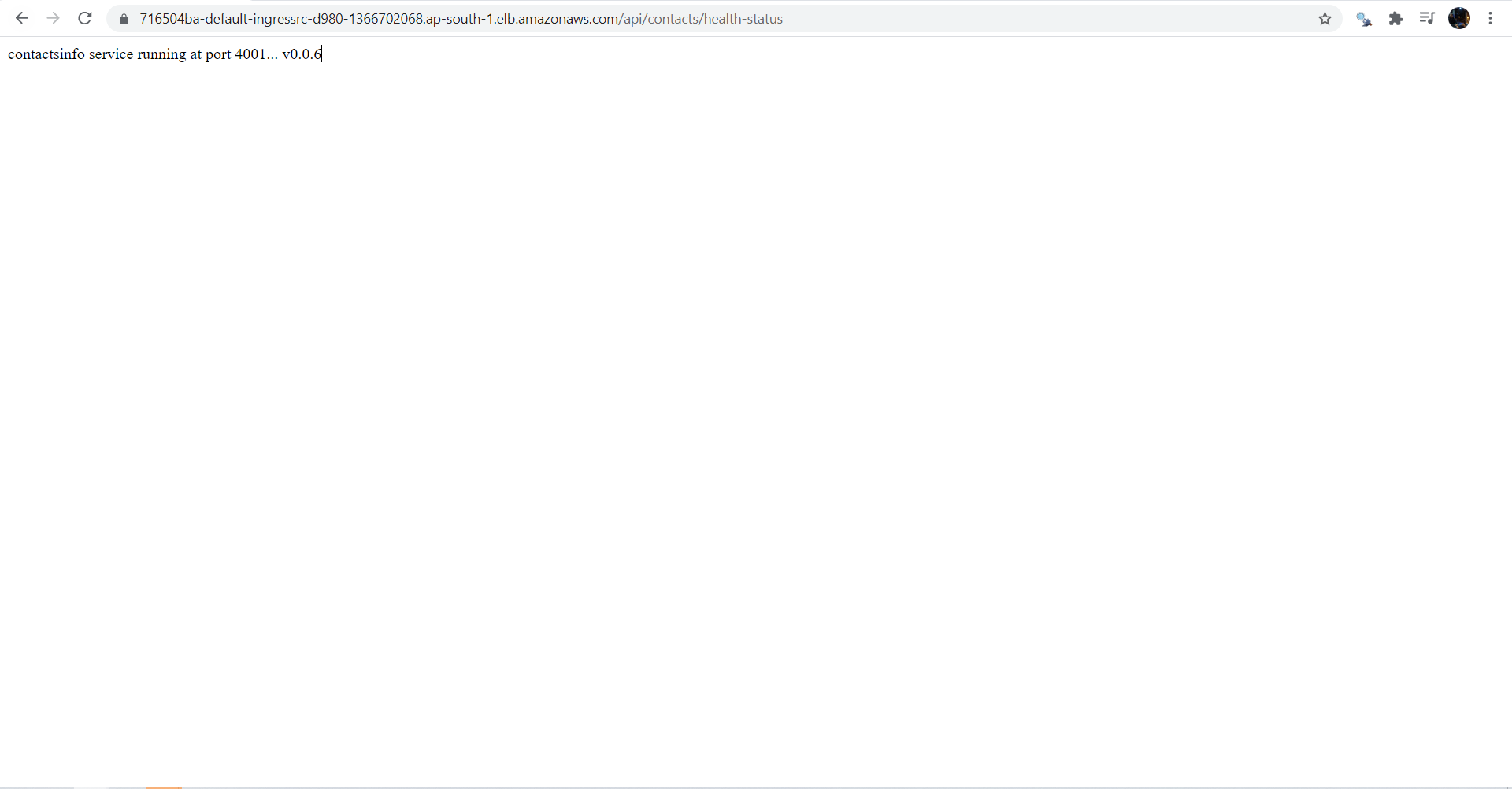


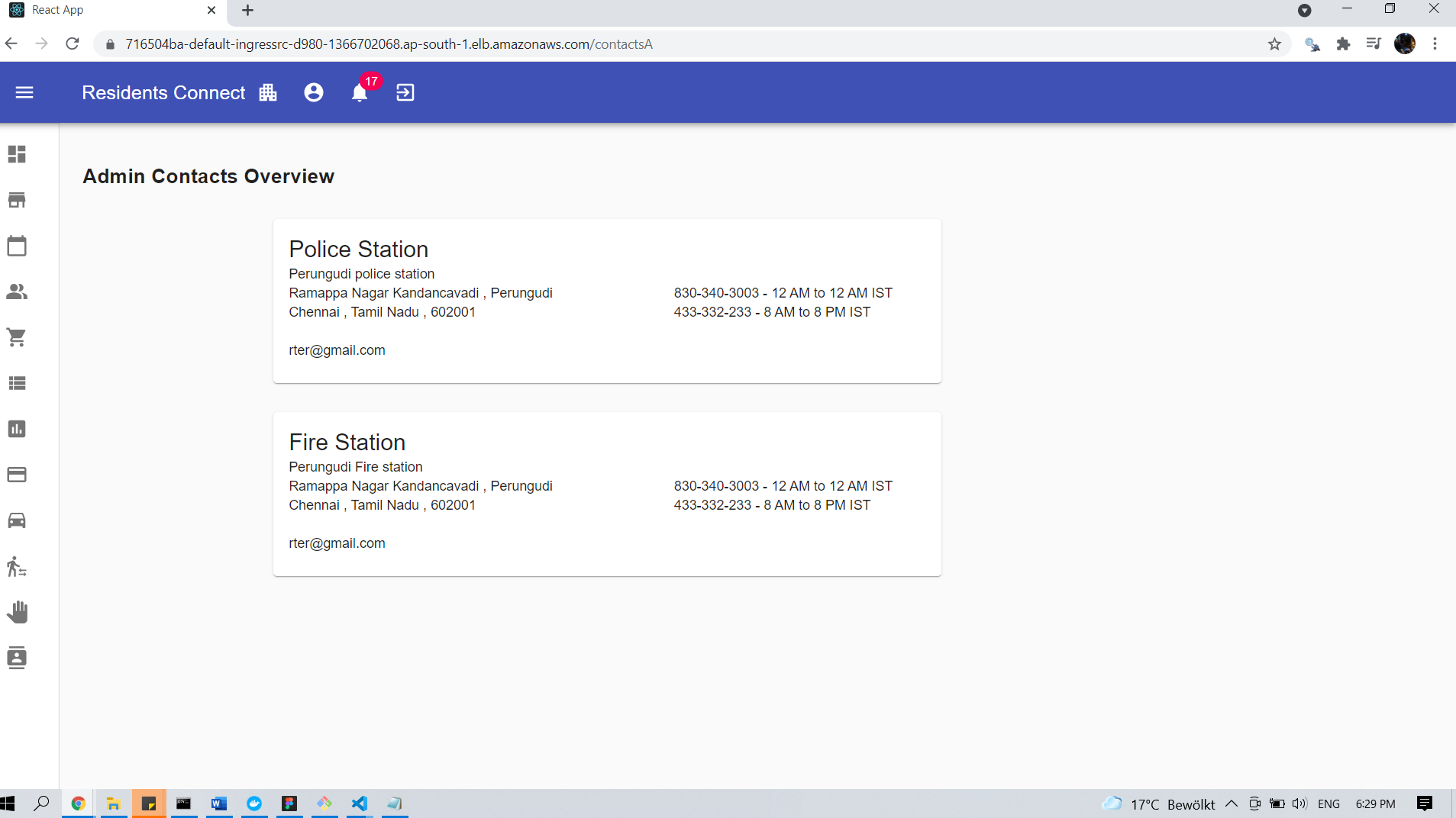


Click on ACM





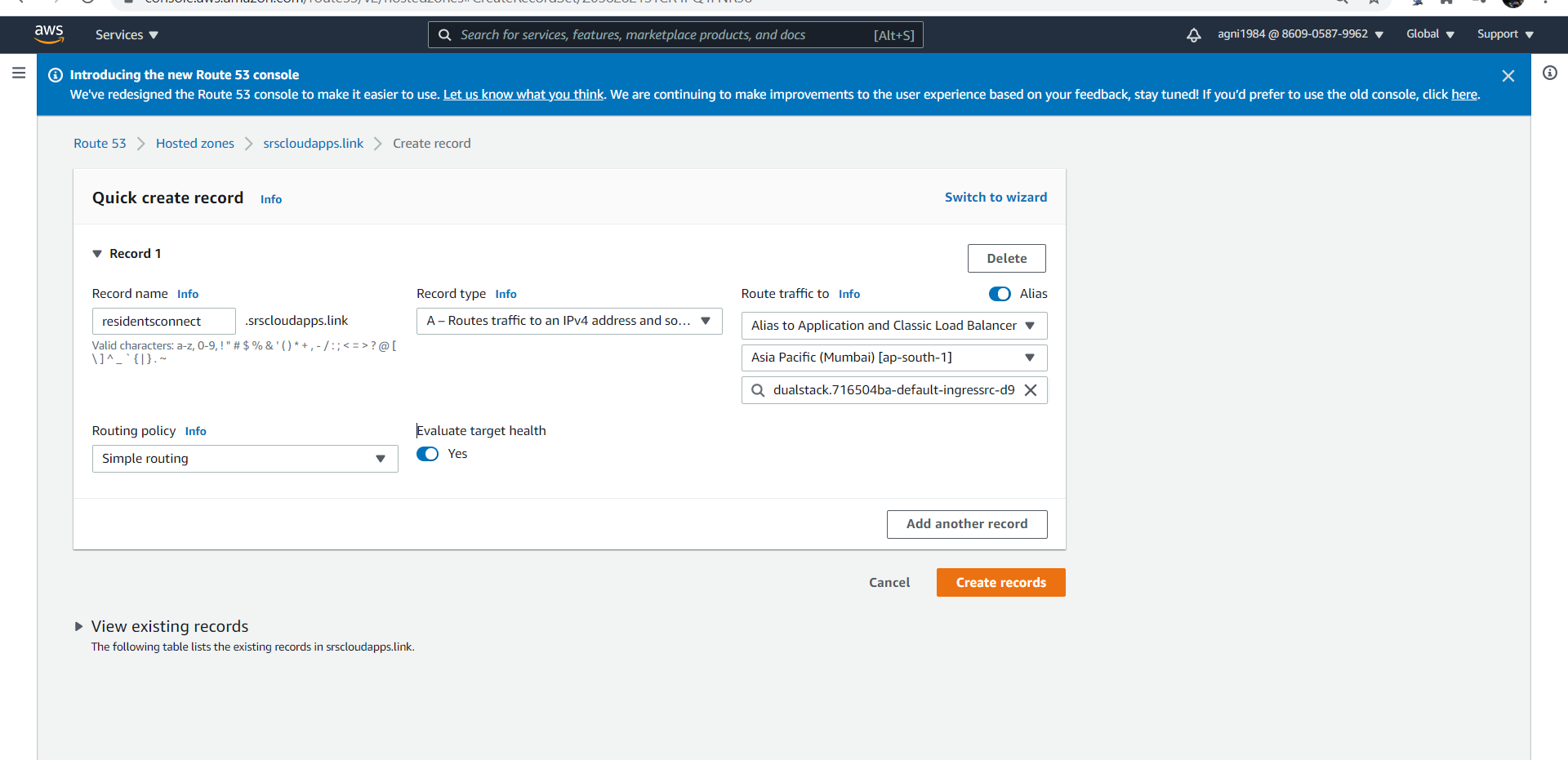


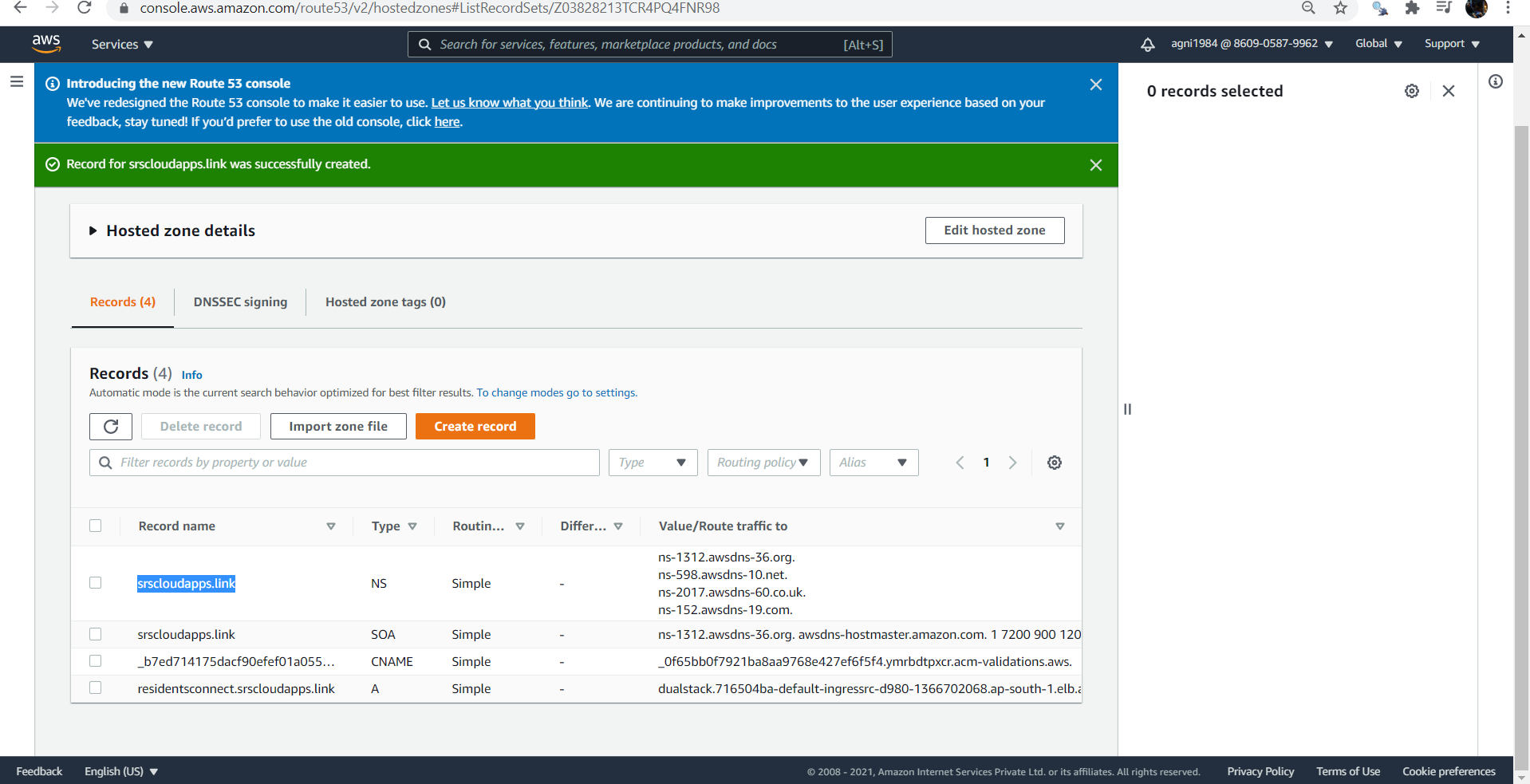


## 8.2. Add DNS in Route53

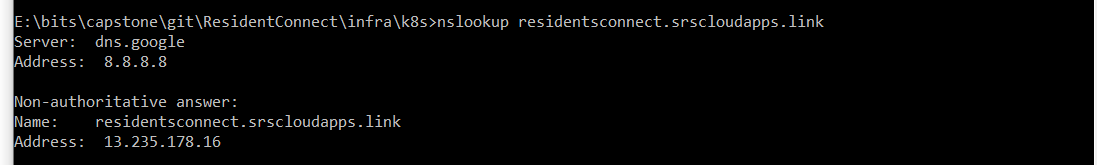
### **8.2.1 Add DNS to route 53**

* Go to **Services -> Route 53**
* Go to **Hosted Zones**
  + Click on **yourdomain.com** (in my case srscloudapps.link)
* Create a **Record Set**
  + **Name:** residentsconnect-dev.srscloudapps.link
  + **Alias:** yes
  + **Alias Target:** Copy our ALB DNS Name here (Sample: 716504ba-default-ingressrc-d980-1366702068.ap-south-1.elb.amazonaws.com)
  + Click on **Create**





Check Namespace for new domain



### **8.2.2 Access Application using newly registered DNS Name**

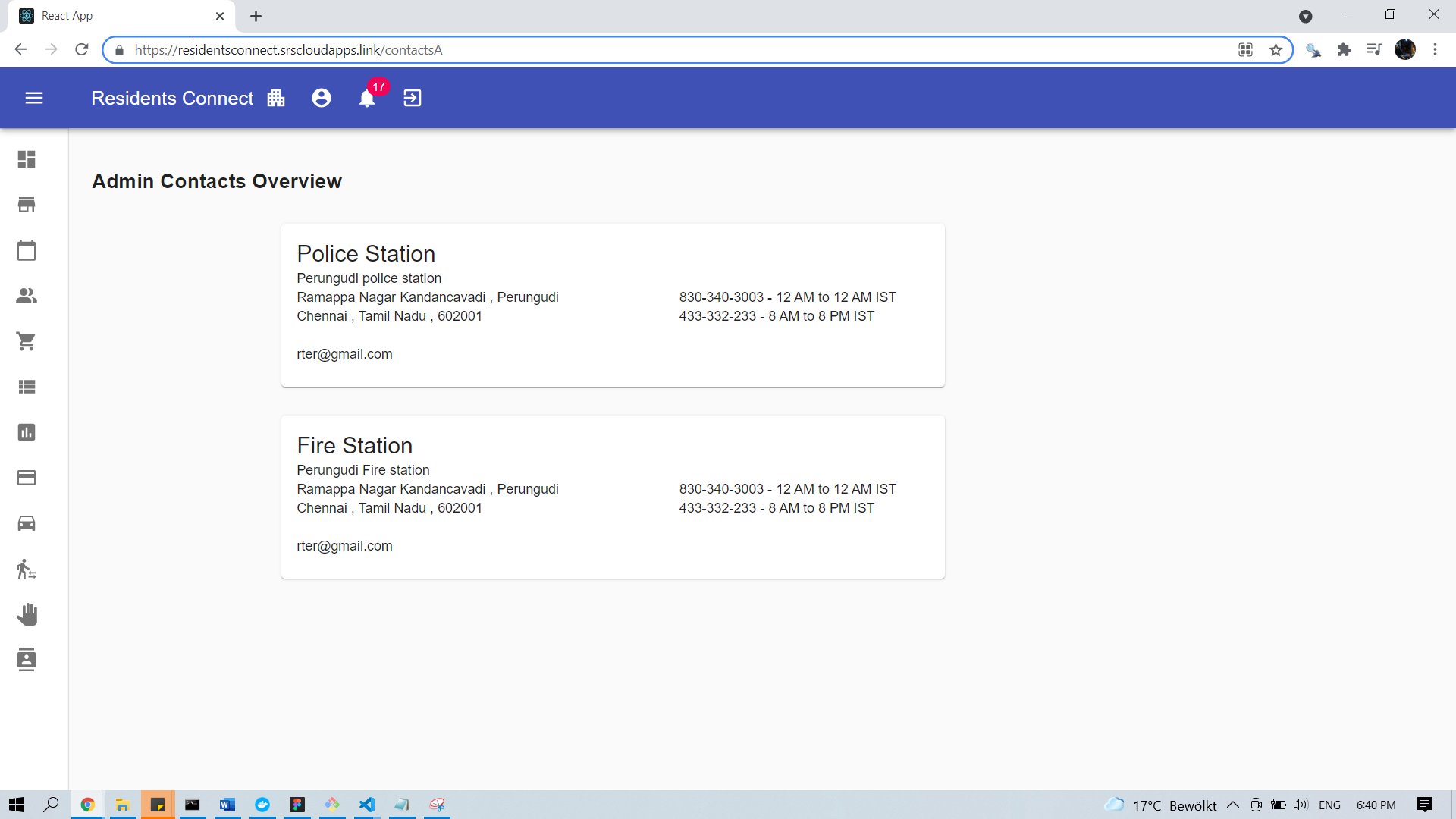
* **Access Application**
* **Important Note:** Instead of srscloudapps.link you need to replace with your registered Route53 domain (Refer pre-requisite Step-02)

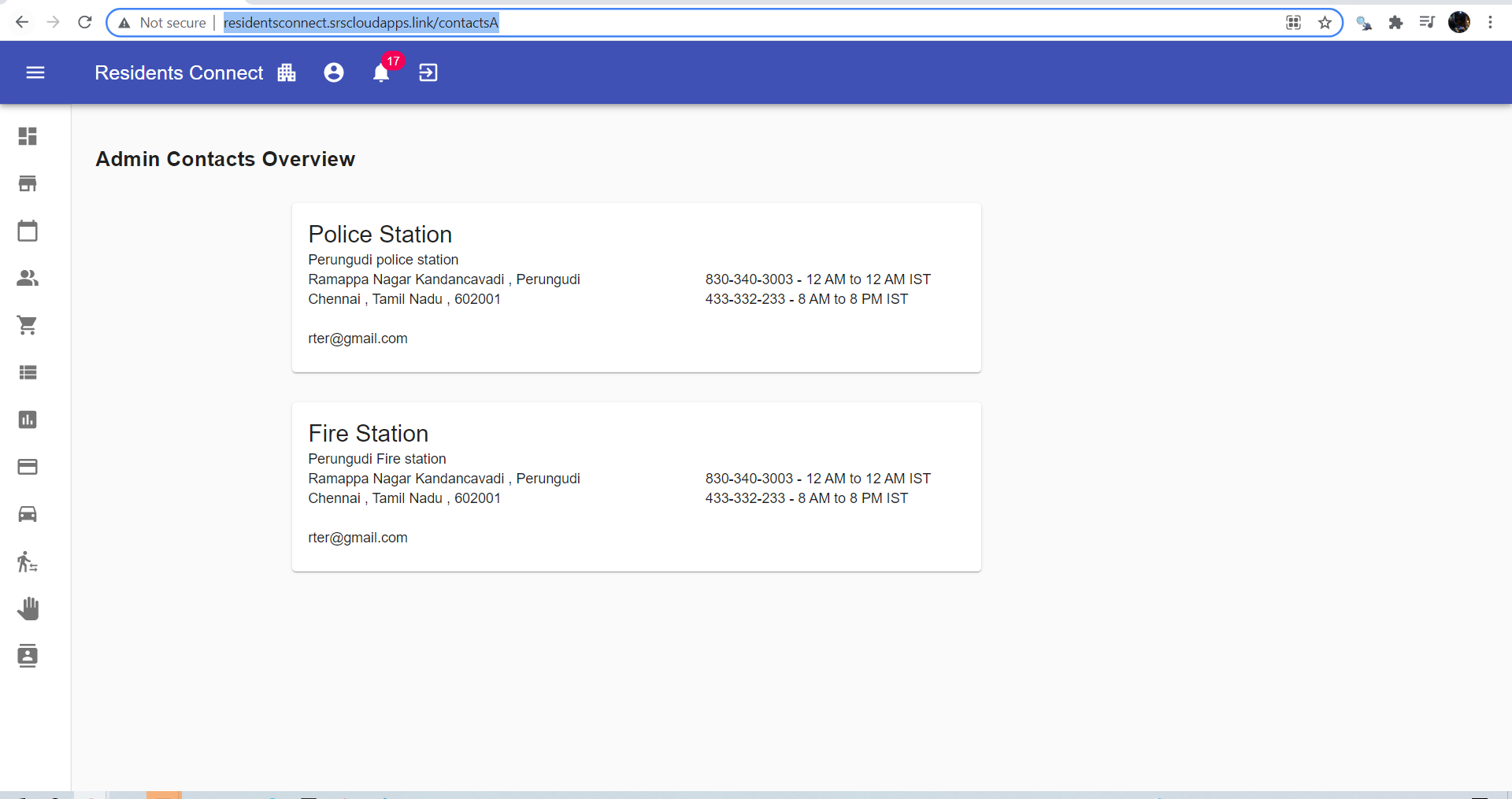
# HTTP URLs

http://residentsconnect.srscloudapps.link/api/contacts/health-status http://residentsconnect.srscloudapps.link/api/payments/health-status http://residentsconnect.srscloudapps.link/contactsA

# HTTPS URLs

https://residentsconnect.srscloudapps.link/api/contacts/health-status https://residentsconnect.srscloudapps.link/api/payments/health-status https://residentsconnect.srscloudapps.link/contactsA





## **8.3. AWS ALB Ingress Controller - Implement HTTP to HTTPS Redirect**

### **8.3.1 Add annotations related to SSL Redirect**

* Redirect from HTTP to HTTPS
* Provides a method for configuring custom actions on a listener, such as for Redirect Actions.
* The action-name in the annotation must match the serviceName in the ingress rules, and servicePort must be use-annotation.
* **Reference for Custom Actions:** <https://kubernetes-sigs.github.io/aws-alb-ingress-controller/guide/ingress/annotation/#actions>

#### **Change-1: Add the Custom Action Annotation**

# SSL Redirect Setting

alb.ingress.kubernetes.io/actions.ssl-redirect: '{"Type": "redirect", "RedirectConfig": { "Protocol": "HTTPS", "Port": "443", "StatusCode": "HTTP\_301"}}'

* **SSL Redirect JSON**

{

"Type":"redirect",

"RedirectConfig":{

"Protocol":"HTTPS",

"Port":"443",

"StatusCode":"HTTP\_301"

}

#### **Change-2: Add the following Path as first ingress rule in the Rules section**

- path: /\* # SSL Redirect Setting

backend:

serviceName: ssl-redirect

servicePort: use-annotation

### **8.3.2 Deploy LoadBalancer with redirect SSL rule**

Deploy ingresscontroller to change ALB setting. Refer ALB-Ingress.yml file below,

#  kubectl apply -f ALB-Ingress.yml

# Annotations Reference:  https://kubernetes-sigs.github.io/aws-alb-ingress-controller/guide/ingress/annotation/

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

  name: ingress-rc-restapp-service

  labels:

    app: rc-restapp

  annotations:

    # Ingress Core Settings

    kubernetes.io/ingress.class: "alb"

    alb.ingress.kubernetes.io/scheme: internet-facing

    # Health Check Settings

    alb.ingress.kubernetes.io/healthcheck-protocol: HTTP

    alb.ingress.kubernetes.io/healthcheck-port: traffic-port

    #alb.ingress.kubernetes.io/healthcheck-path: /health-status

    alb.ingress.kubernetes.io/healthcheck-interval-seconds: '60'

    alb.ingress.kubernetes.io/healthcheck-timeout-seconds: '10'

    alb.ingress.kubernetes.io/success-codes: '200'

    alb.ingress.kubernetes.io/healthy-threshold-count: '2'

    alb.ingress.kubernetes.io/unhealthy-threshold-count: '10'

    ## SSL Settings

    alb.ingress.kubernetes.io/listen-ports: '[{"HTTPS":443}, {"HTTP":80}]'

    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:ap-south-1:860905879962:certificate/430eecf1-95a8-44db-9b50-8dc5cb26fd11

    # SSL Redirect Setting

    alb.ingress.kubernetes.io/actions.ssl-redirect: '{"Type": "redirect", "RedirectConfig": { "Protocol": "HTTPS", "Port": "443", "StatusCode": "HTTP\_301"}}'

spec:

  rules:

    - http:

        paths:

          - path: /\* # SSL Redirect Setting

            backend:

              serviceName: ssl-redirect

              servicePort: use-annotation

          - path: /api/contacts/\*

            backend:

              serviceName: rc-contactsinfo-srv

              servicePort: 4001

          - path: /api/payments/\*

            backend:

              serviceName: rc-payments-srv

              servicePort: 4012

          - path: /\*

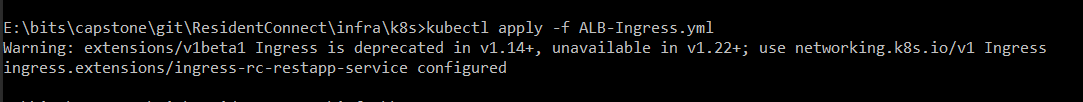
            backend:

              serviceName: rc-frontend-srv

              servicePort: 3000

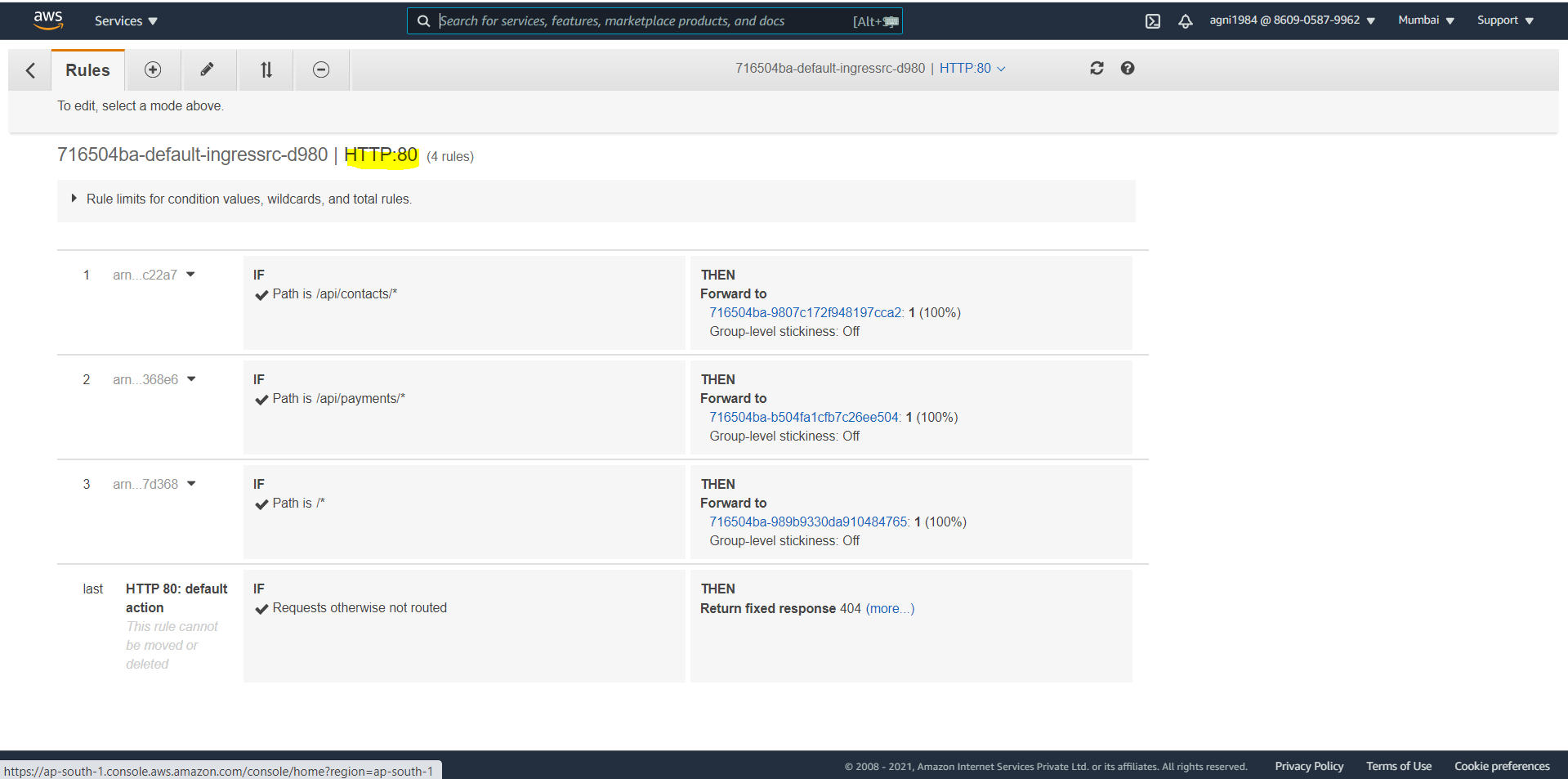
Redeploy ALB

kubectl apply -f ALB-Ingress.yml

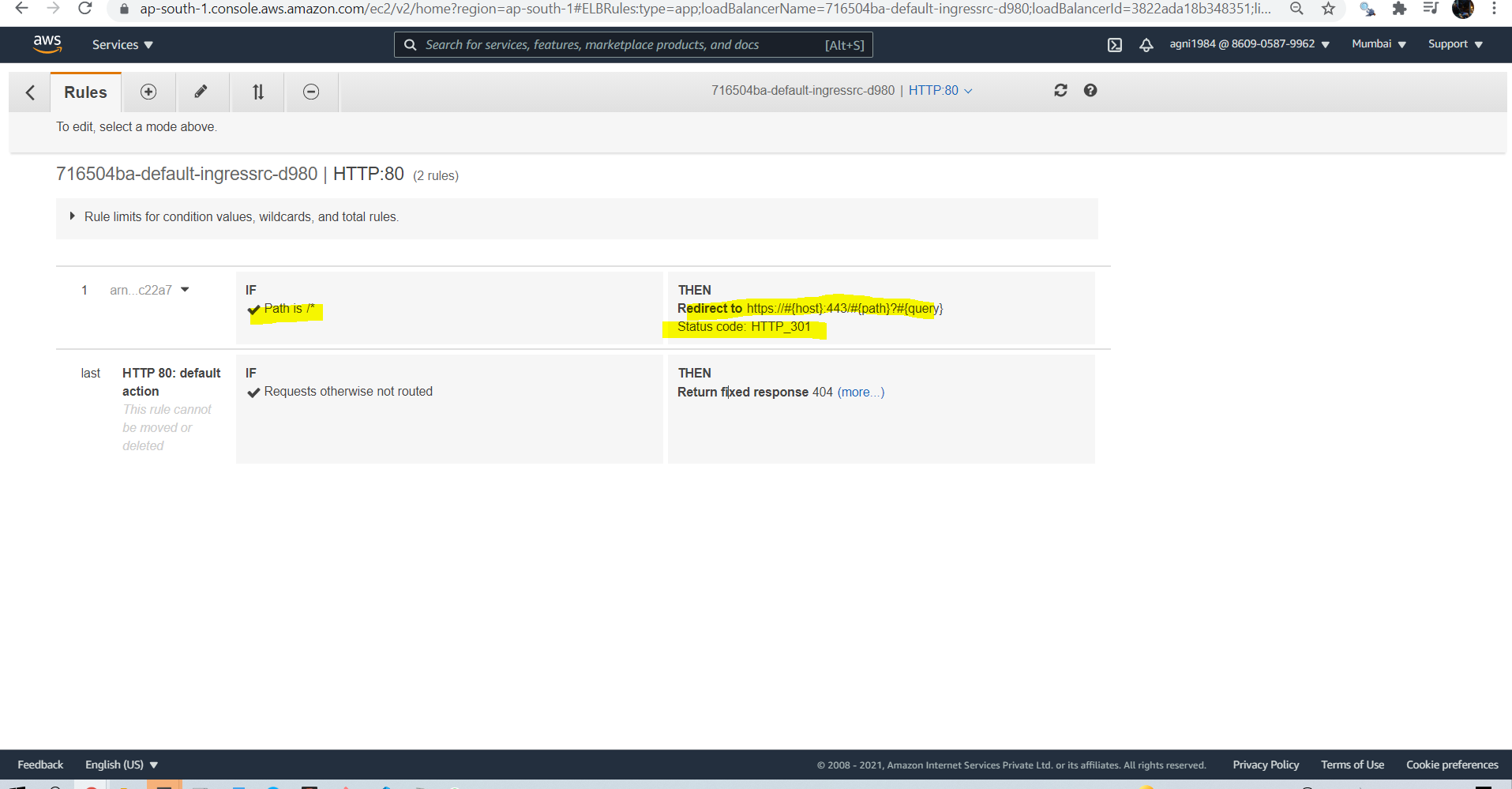


HTTP:80 rule setting should be changed

Before redirect setup



After redirect setup



* **Verify**
  + Load Balancer - Listeneres (Verify both 80 & 443)
  + Load Balancer - Rules (Verify both 80 & 443 listeners)
  + Target Groups - Group Details (Verify Health check path)
  + Target Groups - Targets (Verify all 3 targets are healthy)
  + Verify ingress controller from kubectl
* kubectl get ingress



### **8.3.3 Access Application using newly registered DNS Name**

* **Access Application**

# HTTP URLs (Should Redirect to HTTPS)

http://residentsconnect.srscloudapps.link/api/contacts/health-status http://residentsconnect.srscloudapps.link/api/payments/health-status http://residentsconnect.srscloudapps.link/contactsA

# HTTPS URLs

https://residentsconnect.srscloudapps.link/api/contacts/health-status https://residentsconnect.srscloudapps.link/api/payments/health-status <https://residentsconnect.srscloudapps.link/contactsA>

## 8.4 Create ALB kubernetes basic Ingress Manifest

Create ingress

99-ALB-Ingress.yml

#  kubectl apply -f 99-ALB-Ingress.yml

# Annotations Reference:  https://kubernetes-sigs.github.io/aws-alb-ingress-controller/guide/ingress/annotation/

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

  name: ingress-rc-restapp-service

  labels:

    app: rc-restapp

  annotations:

    # Ingress Core Settings

    kubernetes.io/ingress.class: "alb"

    alb.ingress.kubernetes.io/scheme: internet-facing

    # Health Check Settings

    alb.ingress.kubernetes.io/healthcheck-protocol: HTTP

    alb.ingress.kubernetes.io/healthcheck-port: traffic-port

    #alb.ingress.kubernetes.io/healthcheck-path: /health-status

    alb.ingress.kubernetes.io/healthcheck-interval-seconds: '60'

    alb.ingress.kubernetes.io/healthcheck-timeout-seconds: '10'

    alb.ingress.kubernetes.io/success-codes: '200'

    alb.ingress.kubernetes.io/healthy-threshold-count: '2'

    alb.ingress.kubernetes.io/unhealthy-threshold-count: '10'

    ## SSL Settings

    alb.ingress.kubernetes.io/listen-ports: '[{"HTTPS":443}, {"HTTP":80}]'

    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:ap-south-1:860905879962:certificate/430eecf1-95a8-44db-9b50-8dc5cb26fd11

    # SSL Redirect Setting

    alb.ingress.kubernetes.io/actions.ssl-redirect: '{"Type": "redirect", "RedirectConfig": { "Protocol": "HTTPS", "Port": "443", "StatusCode": "HTTP\_301"}}'

    external-dns.alpha.kubernetes.io/hostname: 'residentsconnect-stg.srscloudapps.link'

spec:

  rules:

    - http:

        paths:

          - path: /\* # SSL Redirect Setting

            backend:

              serviceName: ssl-redirect

              servicePort: use-annotation

          - path: /api/community/\*

            backend:

              serviceName: rc-apartments-srv

              servicePort: 4000

          - path: /api/contacts/\*

            backend:

              serviceName: rc-contactsinfo-srv

              servicePort: 4001

          - path: /api/users/\*

            backend:

              serviceName: rc-users-srv

              servicePort: 4002

          - path: /api/adverts/\*

            backend:

              serviceName: rc-buyandsell-srv

              servicePort: 4004

          - path: /api/classifieds/\*

            backend:

              serviceName: rc-classifieds-srv

              servicePort: 4005

          - path: /api/carpoolings/\*

            backend:

              serviceName: rc-carpooling-srv

              servicePort: 4006

          - path: /api/pollings/\*

            backend:

              serviceName: rc-polling-srv

              servicePort: 4007

          - path: /api/announcements/\*

            backend:

              serviceName: rc-announcements-srv

              servicePort: 4010

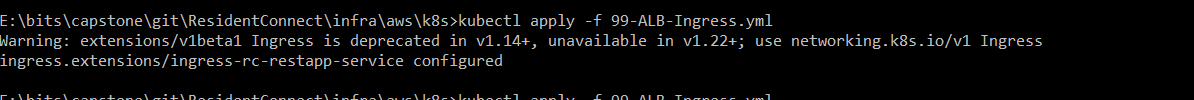
          - path: /api/payments/\*

            backend:

              serviceName: rc-payments-srv

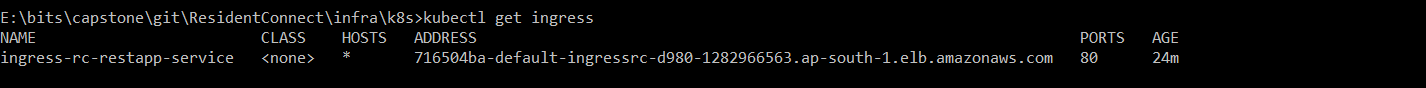
              servicePort: 4012

kubectl apply -f 99-ALB-Ingress.yml



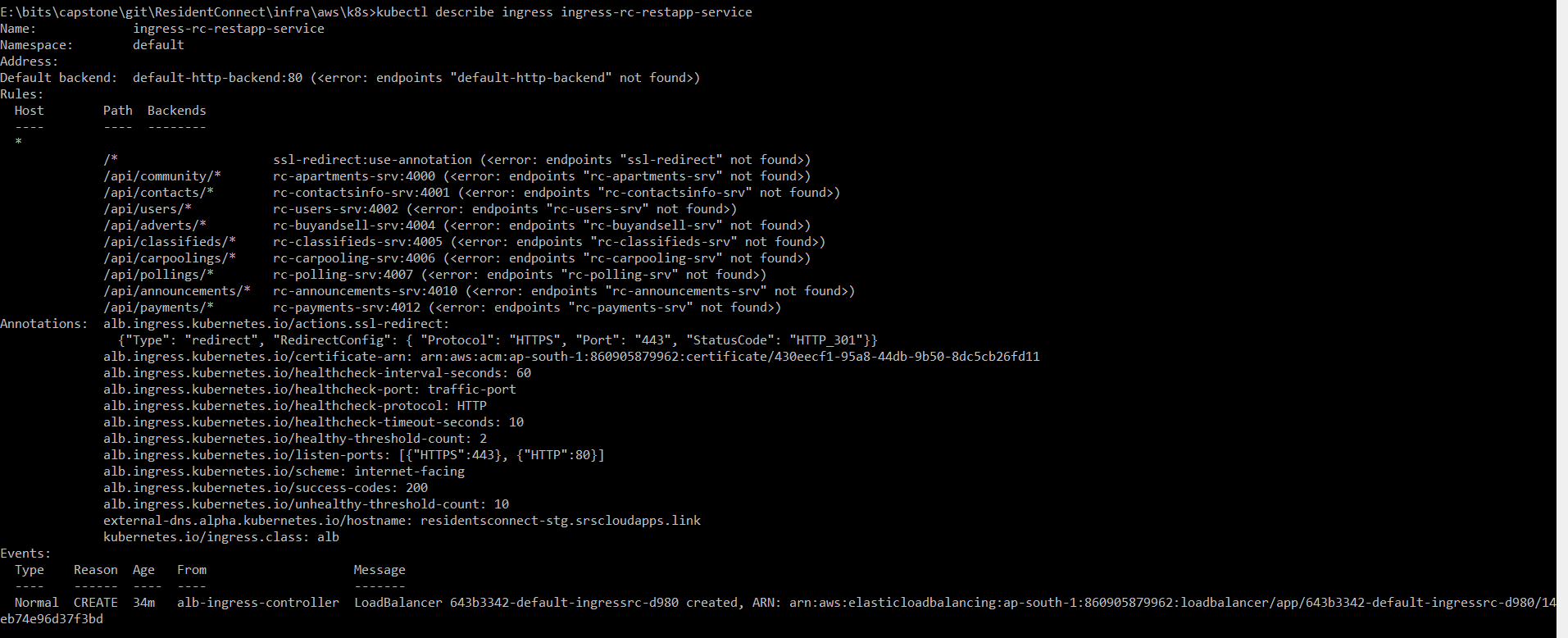
# Get List of Ingress (Make a note of Address field)

kubectl get ingress



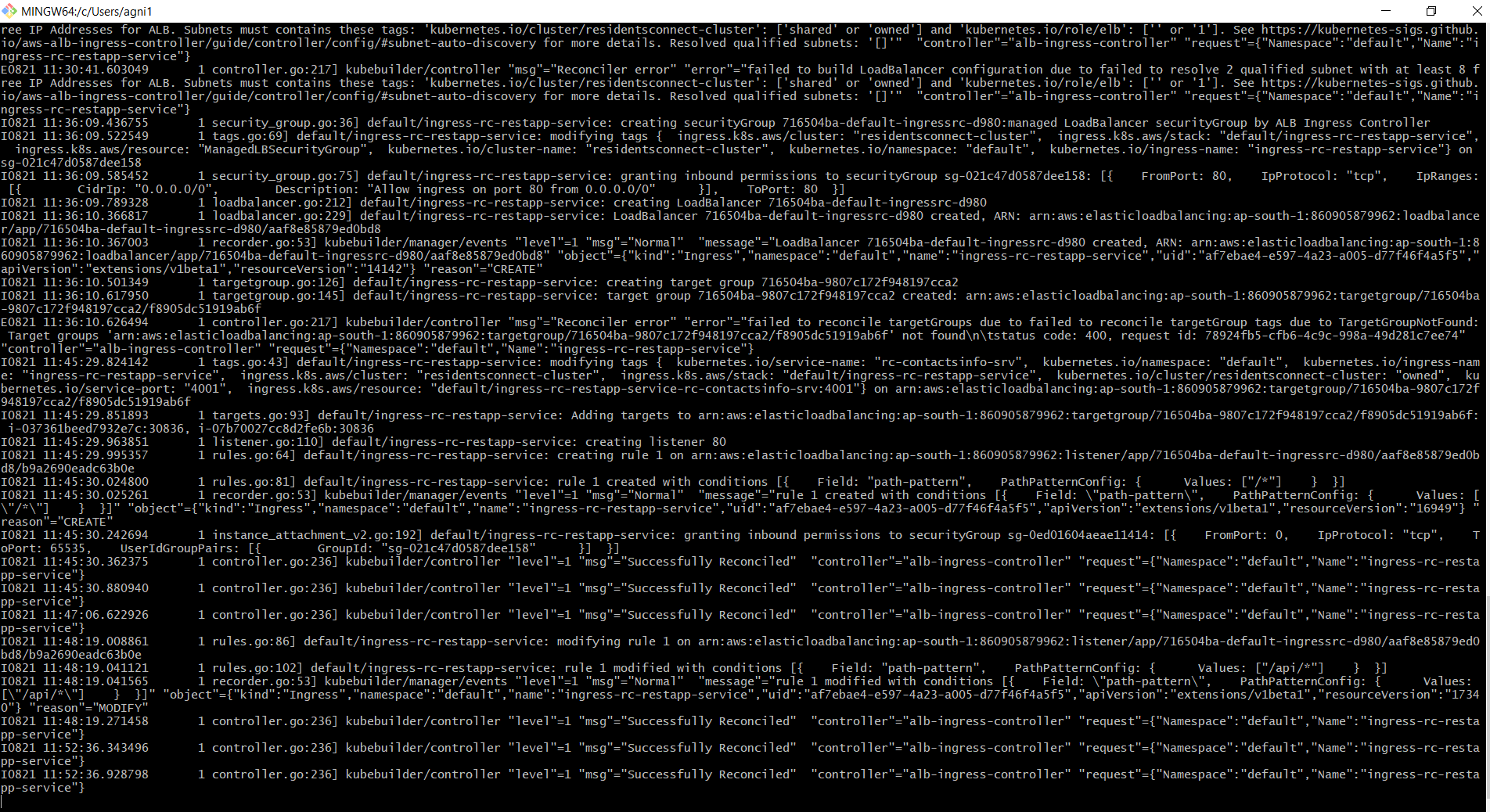
# Describe Ingress Controller

kubectl describe ingress ingress-rc-restapp-service



# Verify ALB Ingress Controller logs

kubectl logs -f $(kubectl get po -n kube-system | egrep -o 'alb-ingress-controller-[A-Za-z0-9-]+') -n kube-system



Note:

* We should not see anything like below log in ALB Ingress Controller log, if we see we did something wrong with ALB Ingress Controleer deployment primarily in creating IAM Policy, Service Account & Role and Associating Role to Service Account.

07:28:39.900001 1 controller.go:217] kubebuilder/controller "msg"="Reconciler error" "error"="failed to build LoadBalancer configuration due to unable to fetch subnets. Error: WebIdentityErr: failed to retrieve credentials\ncaused by: AccessDenied: Not authorized to perform sts:AssumeRoleWithWebIdentity\n\tstatus code: 403, request id: 3d54741a-4b85-4025-ad11-73d4a3661d09" "controller"="alb-ingress-controller" "request"={"Namespace":"default","Name":"ingress-usermgmt-restapp-service"}

* **VERY VERY IMPORTANT NOTE:** Additionally if you see any errors as below, please go to VPC -> EKS VPC -> Subnets -> For both Public Subnets, add the tag as kubernetes.io/cluster/residentsconnect-cluster= shared

E0507 15:40:13.134304 1 controller.go:217] kubebuilder/controller "msg"="Reconciler error" "error"="failed to build LoadBalancer configuration due to failed to resolve 2 qualified subnet with at least 8 free IP Addresses for ALB. Subnets must contains these tags: 'kubernetes.io/cluster/eksdemo1': ['shared' or 'owned'] and 'kubernetes.io/role/elb': ['' or '1']. See https://kubernetes-sigs.github.io/aws-alb-ingress-controller/guide/controller/config/#subnet-auto-discovery for more details. Resolved qualified subnets: '[]'" "controller"="alb-ingress-controller" "request"={"Namespace":"default","Name":"ingress-usermgmt-restapp-service"}

