

Desplegar y gestionar la infraestructura en AWS.

Infraestructura III

Andres Bueno Cardona - A00399454

Jeanpaul A. Vidales - A00395404

1. Creación Key-pair manual:

Para iniciar con el despliegue de la infraestructura es necesario crear un recurso de Key-pair de forma manual a través de la consola, esta tarea no se puede automatizar ya que es necesario tener la llave descargada en nuestra máquina local para así hacer ssh hacia a la EC2-Bastion y posteriormente a cualquiera de las instancias.

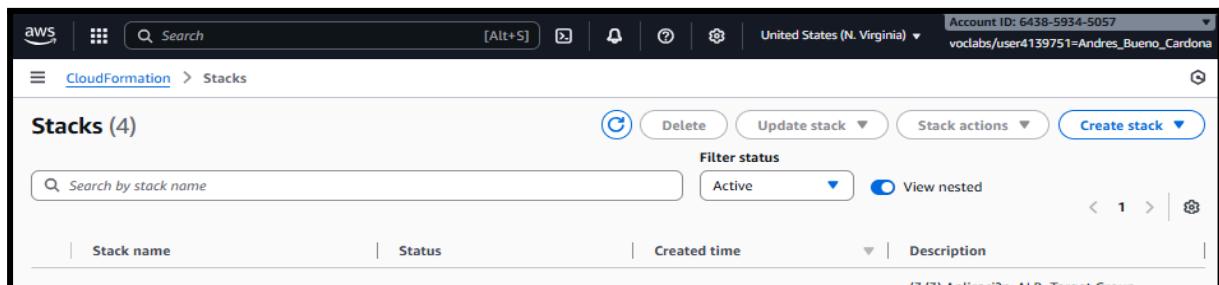
Se hace la creación de llave bajo el nombre de “control” y tipo de llave: RSA.

The screenshot shows the 'Create key pair' wizard in the AWS Lambda console. The 'Key pair' section has a 'Name' field set to 'control'. The 'Key pair type' section has 'RSA' selected. The 'Private key file format' section has '.pem' selected. The 'Tags - optional' section is empty. At the bottom right are 'Cancel' and 'Create key pair' buttons.

Para hacer gestión de la instancia es necesario ejecutar el siguiente comando en la terminal: **ssh-add .\control.pem** para agregar la llave privada al agente SSH que está corriendo, lo que permite que SSH use esa llave automáticamente para autenticarse sin tener que especificarla en cada conexión.

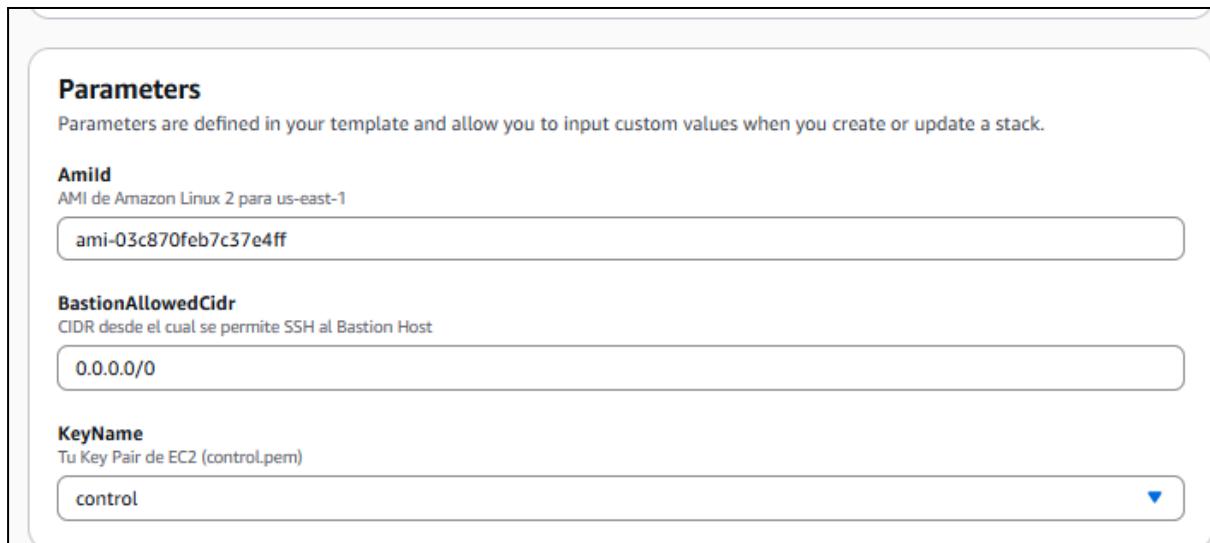
2. Subir templates a Cloudformation:

Ahora se va a hacer uso de la herramienta brindada por AWS llamada “CloudFormation”, se puede acceder a ella desde la barra de búsqueda de servicios. Al ver el apartado, se va a seleccionar la opción de “Create Stack”

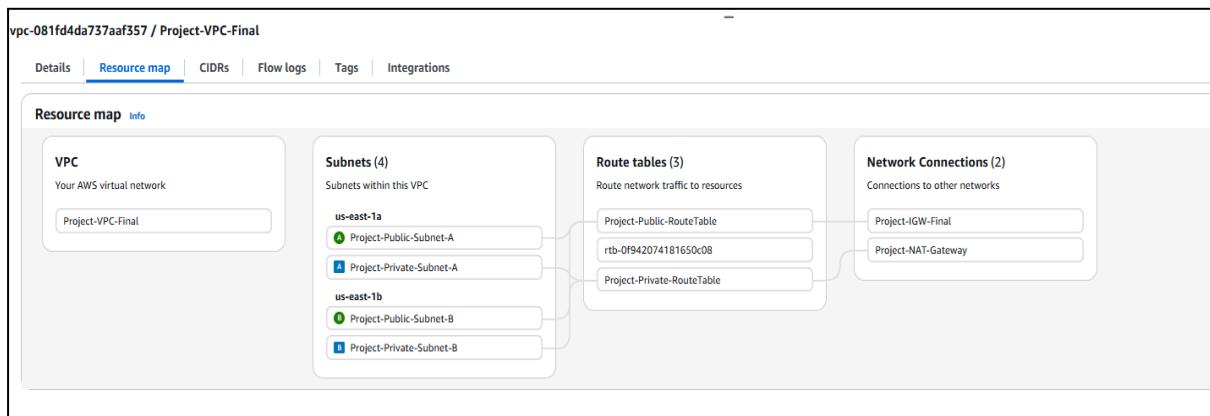


Una vez ahí, seleccionar la opción “With new resources”. Al momento de la creación, usar la opción “Upload a template file”, ya que así lo definimos en el alcance del proyecto y usar un link S3 no era una opción pues había conflictos con el tema de permisos y lectura de archivos. (**Esta es la misma secuencia de pasos que se seguirán para subir todas las templates.**)

Inicialmente se cargará la plantilla correspondiente a todos nuestros componentes de red(network-template.yml). Esta es la estructura base sobre la cual se crearán los demás recursos, al subirla se encontrarán los siguiente parámetros, que ya están completados de forma automática.



La template crea toda la infraestructura base necesaria para alojar una aplicación de forma segura. Primero construye una VPC como red principal y agrega un Internet Gateway para permitir salida a internet desde las subredes públicas. Luego crea dos subredes públicas (para el Bastion y servicios expuestos) y dos subredes privadas (para instancias internas y bases de datos), junto con sus respectivas tablas de ruteo, donde las públicas enrutan al IGW y las privadas usan un NAT Gateway, lo que permite a las instancias privadas acceder a internet sin ser accesibles desde afuera. Se crean tres Security Groups: uno para permitir SSH al Bastión, otro para controlar tráfico hacia la aplicación, y uno exclusivo para la base de datos. Finalmente, se despliega un Bastión Host en una subred pública para servir como punto seguro de acceso SSH a toda la infraestructura privada.



Al finalizar la creación es necesario revisar la sección de “outputs” pues es donde tendremos los identificadores de los recursos de red los cuales se necesitaran como parámetro de las siguientes templates.

Outputs (8)		
Key	Value	Description
AppSecurityGroupId	sg-029cbcc1400c5c77f	Security Group de la App
BastionPublicIP	3.89.79.2	IP pública del Bastion Host para SSH
DBSecurityGroupId	sg-083b224e87d2ab5f2	Security Group de la DB
PrivateSubnetAId	subnet-01487dc8445bc8f48	Subred privada A
PrivateSubnetBId	subnet-0fecdcda6c42b35ddc	Subred privada B
PublicSubnetAId	subnet-0cf9390e3310252e5	Subred pública A
PublicSubnetBId	subnet-0c99fb1fcc94a3681	Subred pública B
VpcId	vpc-081fd4da737aaf357	ID de la VPC

Ahora se debe subir la segunda template(db-template.yaml), correspondiente a la base de datos. Va a requerir ingresar de forma manual una contraseña para la base de datos, en este campo podemos poner cualquier valor, luego se debe asignar el security group y las identificaciones de las subredes privadas A y B que se crearon anteriormente.

Specify stack details

Provide a stack name

Stack name

Enter a stack name
Stack name must contain only letters (a-z, A-Z), numbers (0-9), and hyphens (-) and start with a letter. Max 128 characters. Character count: 0/128.

Parameters
Parameters are defined in your template and allow you to input custom values when you create or update a stack.

DBName
Nombré de la base de datos

ecommerce_db

DBPassword
Contraseña maestra de la base de datos

.....

DBSecurityGroupId
Security Group de la DB (de la stack de red)

sg-0ca8b64dbfaa04904

DBUser
Usuario maestro de la base de datos

postgres

PrivateSubnetAId
Subred privada A (de la stack de red)

subnet-01487dc8445bc8f48

PrivateSubnetBId
Subred privada B (de la stack de red)

subnet-0fecdcda6c42b35ddc

Cancel Previous Next

Esta template despliega la capa de base de datos dentro de la red existente, utilizando las subredes privadas y el security group previamente creados. Define un DBSubnetGroup que asegura que el RDS se distribuya en subredes privadas para mayor seguridad, y luego crea una instancia de RDS PostgreSQL, inaccesible desde internet y protegida por el SG de la base de datos, permitiendo conexiones únicamente desde las instancias de aplicación. También configura parámetros esenciales como nombre, usuario, contraseña, almacenamiento y ausencia de acceso público. Finalmente, expone el endpoint y el puerto para que la aplicación pueda conectarse de forma interna.

El puerto y el endpoint que va a usar la aplicación lo encontramos en la sección de outputs, de tal modo:

Outputs (2)			
Key	Value	Description	Export name
DBEndpointAddress	database-proyect-rdsinstance-jonix8pfpcuk.c8vymxutapr.us-east-1.rds.amazonaws.com	Endpoint (host) de la base de datos	-
DBEndpointPort	5432	Puerto de la base de datos	-

Finalmente vamos a desplegar la aplicación en una instancia, para ello se va a subir la última template (app-template.yaml), siendo esta la que mas parametros va a solicitar, especialmente de la base de datos, ya que con estos parámetros es que se van a reemplazar las variables de entorno definidas en el .env de nuestro proyecto, de modo que ya no esté conectado a una base de datos local y se conecte a la AWS RDS que definimos en el paso anterior.

Parameters	
Parameters are defined in your template and allow you to input custom values when you create or update a stack.	
AmiId	AMI de Amazon Linux 2 para us-east-1 ami-03c870fb7c37e4ff
AppSecurityGroupId	Security Group de la App (de la stack de red) sg-029ebbc1400c5c77f
DBHost	Endpoint de la DB (DBEndpointAddress de la stack de DB) database-proyect-rdsinstance-jonix8pfpcuk.c8vymxutapr.us-east-1.rds.amazonaws.com
DBName	Nombre de la base de datos (igual a la stack de DB) ecommerce_db
DBPassword	Contraseña de la base de datos (igual a la stack de DB) *****
DBPort	Puerto de la DB (DBEndpointPort de la stack de DB) 5432
DBUser	Usuario maestro de la base de datos (igual a la stack de DB) postgres
GitRepoURL	La URL HTTPS de tu repositorio Git https://github.com/AndresBueno420/JustBuy-Commerce.git
InstanceRoleName	El Perfil de Instancia de IAM (encontrado en tu sandbox) myS3Role
KeyName	Tu Key Pair de EC2 (control.pem) control

Es necesario diferenciar el security group, en este caso es el de la app como tal, el cual cuenta con reglas diferentes al de la base de datos, también se debe de usar el output de endpoint mencionado anteriormente en “DBHost”, necesitará la url de nuestro repositorio para así clonarlo y hacer el despliegue de la aplicación misma, se definió que tuviese el s3 role ya que era el que menos problemas presento, ya que no se contaba con ningún rol de laboratorio.

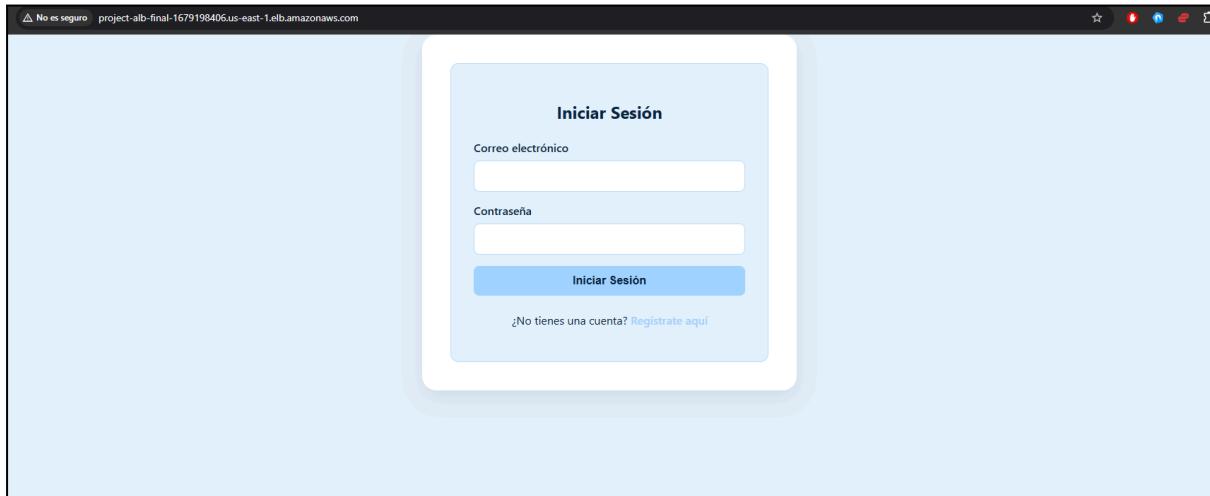
MPAccessToken	Tu Access Token de Mercado Pago
NotificationEmail	Correo donde SNS enviar? alertas atkinsonvi2@gmail.com
PrivateSubnetAId	Subred privada A (de la stack de red) subnet-01487dc8445bc8f48
PrivateSubnetBId	Subred privada B (de la stack de red) subnet-0fecdc4c6c42b35ddc
PublicSubnetAId	Subred pÙblica A (de la stack de red) subnet-0cf9390e3310252e5
PublicSubnetBId	Subred pÙblica B (de la stack de red) subnet-0c99fb1fcc94a3681
SessionSecret	Secreto para express-session
VpcId	ID de la VPC (de la stack de red) vpc-081fd4da737aaaf357

Finalmente se deben de especificar todas las subredes, no hace falta ver el valor de los outputs, ya que saldrán en el dropdown. Los valores que se tienen de forma local, son el de MPAccessToken el cual hace referencia a nuestro token para usar la API de pruebas brindada por mercado pago y también el session secret utilizado para autenticación. (No hace falta conectarse a la instancia pues ya se va a hacer la inicialización del server y población de la base de datos con el script de user data.)

Esta template despliega toda la capa de aplicación, un Application Load Balancer (ALB) que recibe tráfico público, un Target Group para enrutar solicitudes, y un Launch Template que define cómo debe crearse cada instancia de la aplicación, incluyendo instalación, descarga del repositorio, variables de entorno y arranque del proceso con PM2. Luego crea un Auto Scaling Group (ASG) en subredes privadas para ejecutar la aplicación con escalado automático basado en la carga, manteniendo alta disponibilidad. También agrega un Security Group dedicado para el ALB, así como un sistema de notificaciones mediante SNS, con suscripción por correo y una alarma de CloudWatch que envía alertas cuando la CPU supera cierto umbral. Finalmente, expone la URL del ALB para acceder a la aplicación.

Outputs (2)			
<input type="text"/> Search outputs			
Key	Value	Description	Export name
LoadBalancerURL	Project-ALB-Final-1679198406.us-east-1.elb.amazonaws.com	The public URL for your E-Commerce application	-
SnsTopicArn	arn:aws:sns:us-east-1:643859345057:project-app-notifications	ARN del SNS Topic de notificaciones	-

Se da click a dicho link y ya tendremos acceso a la aplicación

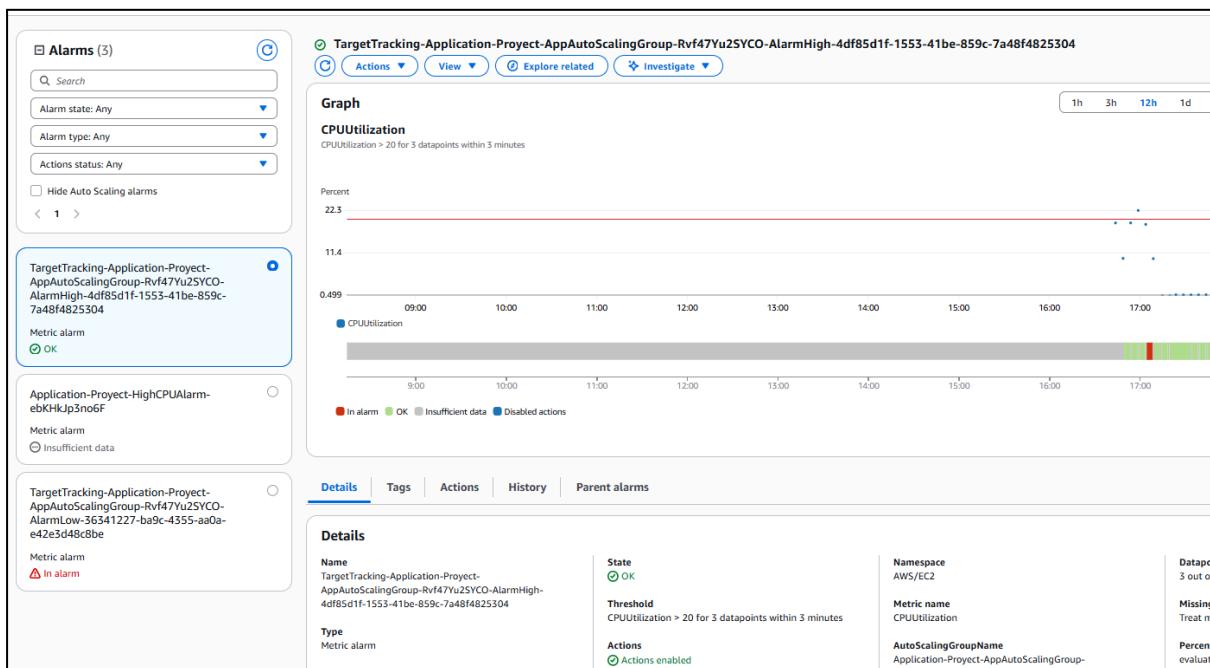


3. Monitoreo y control.

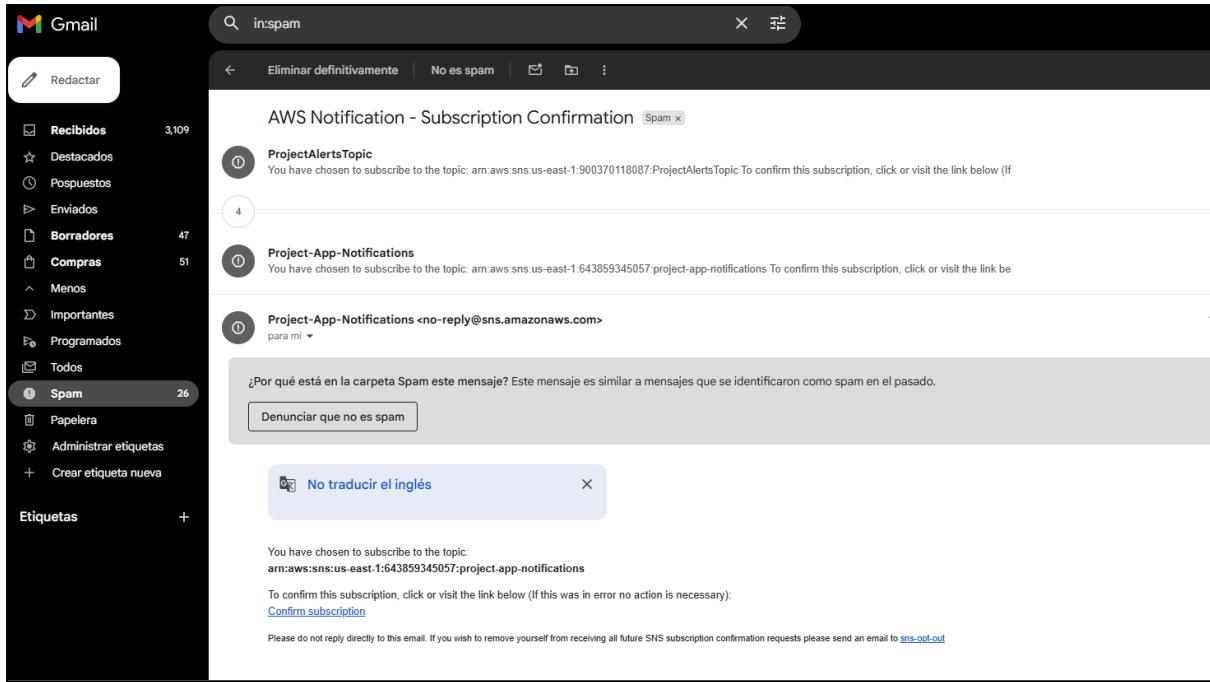
Finalmente podemos revisar las instancias desde la consola en el apartado de EC2.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP
Project-EC2-App-Final	i-0bf2f3a4d852e463e	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a	-	-
Project-Bastion-Host	i-01c1fc299ccce41d5	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a	ec2-3-89-79-2.compute...	3.89.79.2

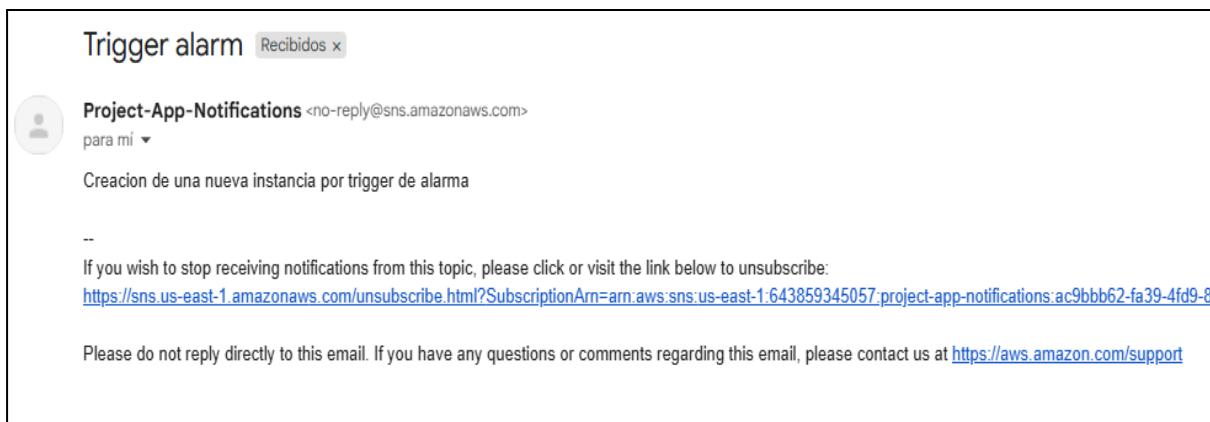
Podremos revisar el estado de la gpu y su asociación con la alarma en el apartado de cloudwatch:



Es necesario confirmar la suscripción al topic creado por el SNS para que lleguen correos relacionados con las alarmas.



Llegan correos informándonos sobre el estado de la instancia.



Para verificar el autoscaling, se hizo una prueba de stress realizando uso del siguiente comando:

- ab -n 50000 -c 1000
<http://Project-ALB-Final-1679198406.us-east-1.elb.amazonaws.com/login.html>
- ab -n 50000 -c 1000
<http://Project-ALB-Final-1679198406.us-east-1.elb.amazonaws.com/catalog.html>

Lo anterior se ve validado en la imagen previa de cloudwatch, ya que la sección en rojo indica que en ese momento se sobrepasó el threshold que permite, que en este caso se definió como 20% ya que la aplicación no es tan pesada.