

Desarrollo de soluciones Cloud
Proyecto Entrega 4

INTEGRANTES:

Jheisson Orlando Cabezas Vera	j.cabezasv@uniandes.edu.co
Diego Alberto Rodríguez Cruz	da.rodriguezc123@uniandes.edu.co

DOCENTE:
Jesse Padilla Agudelo

UNIVERSIDAD DE LOS ANDES
BOGOTÁ D.C.
2025 – II

CONTENIDO

INTRODUCCIÓN	3
OBJETIVOS.....	4
Objetivo General	4
Objetivos Específicos	4
Modelo de Despliegue - Escalabilidad en la Capa Batch o Worker	5
Arquitectura actual (AS-IS).....	6
Arquitectura propuesta (TO-BE).....	8
Despliegue de infraestructura	9
Funcionamiento de la aplicación	17
CONCLUSIONES.....	22
BIBLIOGRAFÍA.....	23

INTRODUCCIÓN

En la actualidad, la escalabilidad es un aspecto fundamental en el diseño e implementación de aplicaciones web modernas, especialmente cuando se despliegan sobre infraestructuras en la nube. En esta entrega se aborda la implementación de soluciones escalables en la capa batch/worker del backend, con el propósito de garantizar que la aplicación pueda responder eficientemente a la demanda de usuarios y tareas concurrentes. A través del uso de servicios de Amazon Web Services (AWS) como EC2, RDS, S3, SQS/Kinesis, CloudWatch y Auto Scaling, se busca construir una arquitectura flexible, capaz de crecer o reducir su capacidad de manera automática, optimizando recursos y costos. Este trabajo consolida los conocimientos adquiridos en fases previas del curso de Maestría en Ingeniería de Software y enfatiza las buenas prácticas de despliegue, monitoreo y administración en entornos distribuidos.

OBJETIVOS

Objetivo General

Diseñar e implementar una arquitectura de backend escalable en la nube que permita la operación eficiente de una aplicación web mediante el uso de servicios AWS, asegurando el balanceo de carga, el procesamiento asíncrono y la adaptación automática de recursos según la demanda.

Objetivos Específicos

- Configurar una infraestructura en AWS que soporte la ejecución distribuida de la capa web y la capa *worker* mediante instancias EC2.
- Implementar políticas de *autoscaling* para los servidores web y los procesos *batch* que permitan el ajuste dinámico de la capacidad del sistema.
- Integrar un balanceador de carga que distribuya equitativamente las solicitudes entre los distintos servidores web.
- Incorporar un sistema de mensajería asíncrona (Amazon SQS o Kinesis) que garantice la comunicación eficiente entre los componentes web y *worker*.
- Ejecutar pruebas de estrés y capacidad para validar el rendimiento, la estabilidad y la escalabilidad del sistema bajo distintas condiciones de carga.
- Documentar la arquitectura, los resultados de las pruebas y las recomendaciones para futuras mejoras en la escalabilidad y resiliencia de la aplicación.

Modelo de Despliegue - Escalabilidad en la Capa Batch o Worker

Con el fin de lograr que la aplicación web esté diseñada para escalar, se deberá modificar el aplicativo desarrollado en la entrega 3 tal que sea capaz de responder a la demanda de usuarios requerida. La organización ya ha identificado que la aplicación deberá implementar los siguientes servicios:

- Amazon EC2: Ejecución de la capa web y la capa worker. Por decisión de negocio, se han seleccionado instancias de cómputo con 2 vCPU, 2 GiB de RAM y 30 GiB de almacenamiento.
- Amazon RDS: Almacenamiento de la información de la aplicación en una base de datos relacional. En las fases iniciales de la implementación, puede utilizarse una instancia de EC2 y sustituirla por RDS únicamente durante las pruebas de carga.
- Amazon S3: Almacenamiento de los videos originales y los procesados.
- Amazon CloudWatch: Monitoreo de las instancias y los demás servicios empleados.
- Autoscaling: Servicio para escalar la capa web en función de los usuarios concurrentes.
- Load Balancer: Distribución de la carga entre las instancias que exponen el API REST.
- Amazon SQS o Amazon Kinesis: Intercambio de mensajes de forma confiable, rápida y asíncrona.

Esta entrega es una extensión del modelo despliegue de la entrega anterior, es decir, abarca las características y requerimientos previos, además de los ajustes solicitados a continuación. En concreto, el despliegue incluye el uso de un balanceador de carga, hasta 3 servidores web, hasta 3 servidores workers, políticas de autoscaling y sistema de mensajes. A fin de implementar el modelo propuesto, es necesario llevar a cabo las siguientes actividades:

1. Configurar el servicio de balanceo de carga para poder desplegar varios servidores web. Actividad completada en la entrega anterior.
2. Definir e implementar una estrategia para que los servidores web escalen de manera automática. Se deberán establecer las condiciones de escalamiento mediante los servicios de monitoreo, autoscaling y de balanceo de cargas. Actividad completada en la entrega anterior.
3. (20%) Diseñar e implementar una estrategia para que los servidores de procesamiento de archivos (workers) puedan escalar de manera automática. Se

deberán establecer las condiciones de escalamiento utilizando los servicios de monitoreo, autoscaling y cola de mensajes.

4. (20%) Configurar el servicio de mensajería asíncrona (SQS o Kinesis), el cual actuará como sistema de comunicación entre los servidores web y los procesos workers. Se deberán efectuar las modificaciones necesarias para que ahora los servidores web coloquen las solicitudes para procesar nuevos videos en el servicio de mensajería asíncrona (SQS o Kinesis), y los workers extraigan los archivos de ella.
5. (20%) Configurar la capa web (API REST) de la aplicación en alta disponibilidad. Para esto configure el balanceador de carga y los nodos tal que se desplieguen sobre dos zonas de disponibilidad de AWS.
6. (10%) La aplicación web debe satisfacer la totalidad de los requerimientos funcionales que ya fueron estipulados y detallados en el enunciado del proyecto.

Arquitectura actual (AS-IS)

La arquitectura implementada está diseñada para proporcionar **alta disponibilidad, escalabilidad automática y resiliencia** ante fallos, aprovechando los servicios administrados de **Amazon Web Services (AWS)**.

Descripción general:

1. Usuario y acceso público:

El usuario accede a la aplicación a través de Internet, ingresando por la subred pública de la VPC. El tráfico es dirigido al balanceador de carga, que distribuye las solicitudes hacia las instancias disponibles del grupo de auto escalado.

2. Capa web y Redis (Auto Scaling Group):

En la subred privada se encuentra un grupo de Auto Scaling (ASG) que contiene las instancias Amazon EC2 encargadas de ejecutar el servidor web y el servicio de Redis.

- El Auto Scaling Group permite aumentar o reducir dinámicamente el número de instancias EC2 según la carga de trabajo (por ejemplo, basado en el uso de CPU o la cantidad de conexiones activas).
- Redis actúa como sistema de cola o caché, gestionando las tareas enviadas al *worker* y mejorando la velocidad de respuesta.

3. Worker (procesamiento asíncrono):

El componente Worker, desplegado en una instancia EC2 separada, recibe las tareas desde Redis y ejecuta los procesos de backend o de procesamiento intensivo. Este

diseño desacopla el procesamiento de las peticiones web, mejorando la escalabilidad y el rendimiento general del sistema.

4. Almacenamiento de objetos (Amazon S3):

El servicio Amazon S3 reemplaza el antiguo servidor NFS, proporcionando un sistema de almacenamiento de objetos altamente disponible y escalable. Aquí se almacenan tanto los archivos originales como los resultados procesados.

5. Base de datos (Amazon RDS – PostgreSQL):

Los datos estructurados de la aplicación se gestionan en Amazon RDS con PostgreSQL, un servicio administrado que garantiza disponibilidad, recuperación ante fallos y escalabilidad vertical.

6. Escalabilidad y resiliencia:

- El Auto Scaling Group se encarga de ajustar automáticamente la capacidad de la capa web y Redis según la demanda.
- El balanceador de carga distribuye el tráfico de manera equitativa entre las instancias activas.
- En caso de que una instancia falle, el ASG lanza una nueva de forma automática, manteniendo la continuidad del servicio.

Beneficios principales:

- Escalabilidad automática según la demanda.
- Alta disponibilidad ante fallos de instancias.
- Separación de responsabilidades entre capa web, procesamiento y almacenamiento.

Uso de servicios administrados que reducen la carga operativa.

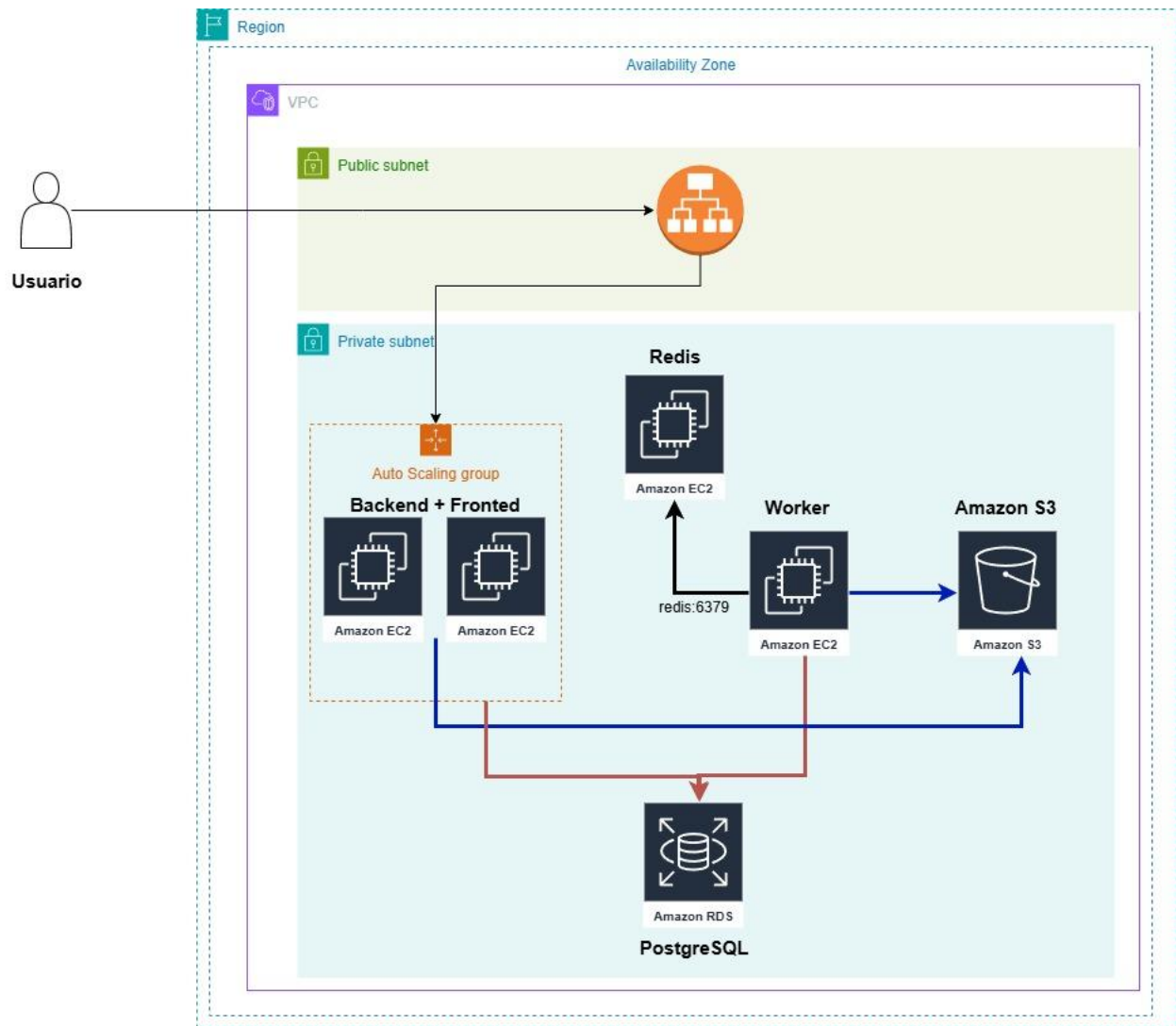


Figura 1. Arquitectura distribuida con escalamiento automático para el backend y frontend. (ANTES con Redis)

Arquitectura propuesta (TO-BE)

La arquitectura implementada está diseñada para proporcionar **alta disponibilidad, escalabilidad automática y resiliencia** ante fallos, aprovechando los servicios administrados de **Amazon Web Services (AWS)**.

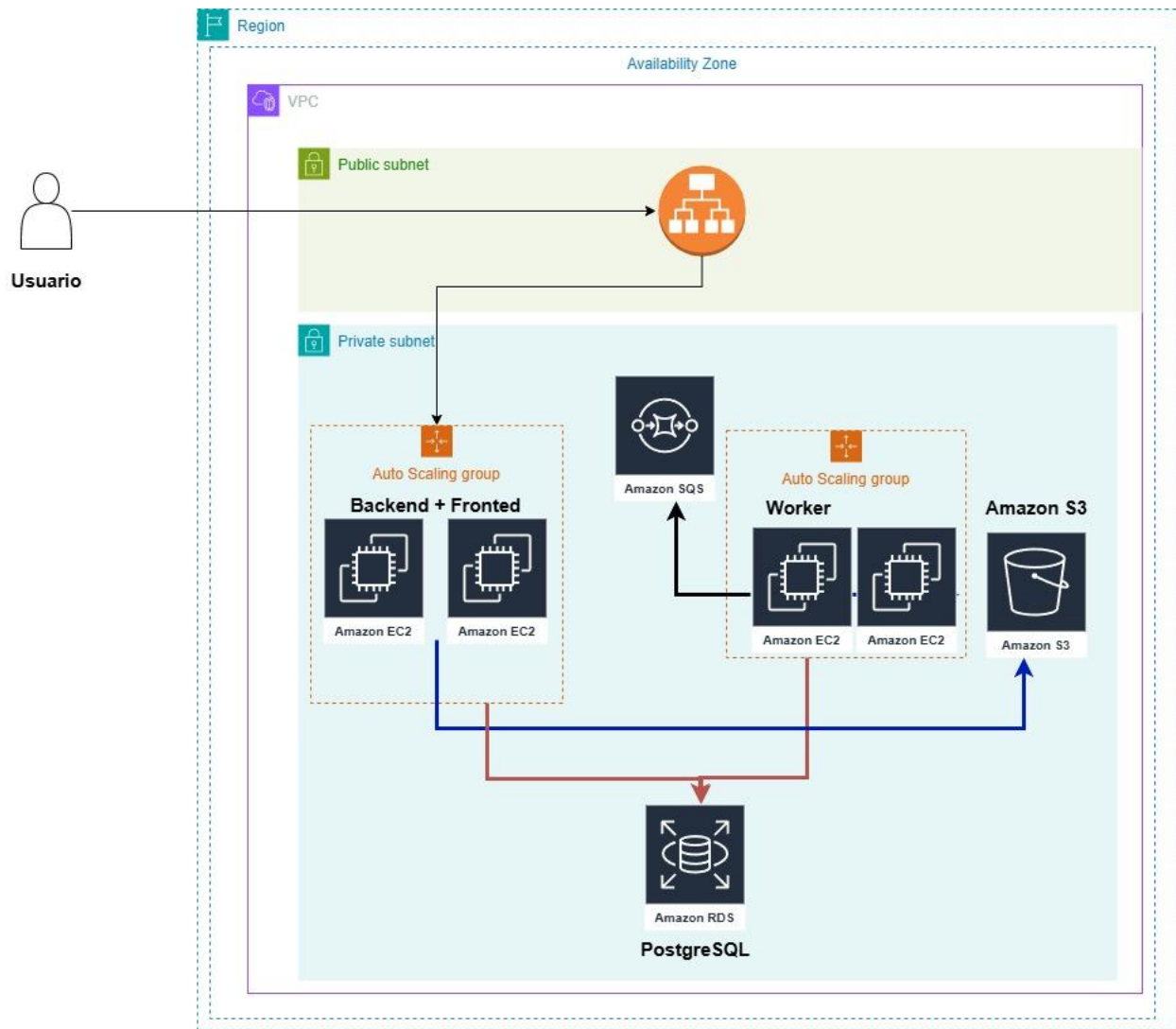


Figura 2. Arquitectura distribuida con escalamiento automático y SQS.

Despliegue de infraestructura

Para el despliegue de la infraestructura se optó por utilizar la **AWS CLI** para la ejecución de **CloudFormation**, con el cual se crean los servicios necesarios para el despliegue. A continuación, se describen los pasos que se siguieron para la implementación de la infraestructura:

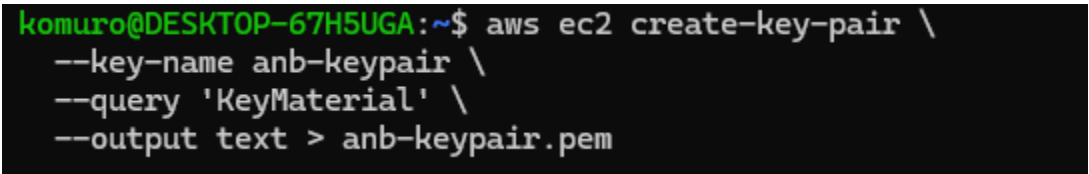
Paso 1 – Configurar AWS Cli

- **Opción 1 – Exportación de variables**

- `export AWS_ACCESS_KEY_ID=$(aws configure get aws_access_key_id)`

Paso 3 – Creación de la llave de conexión (anb-keypair.pem)

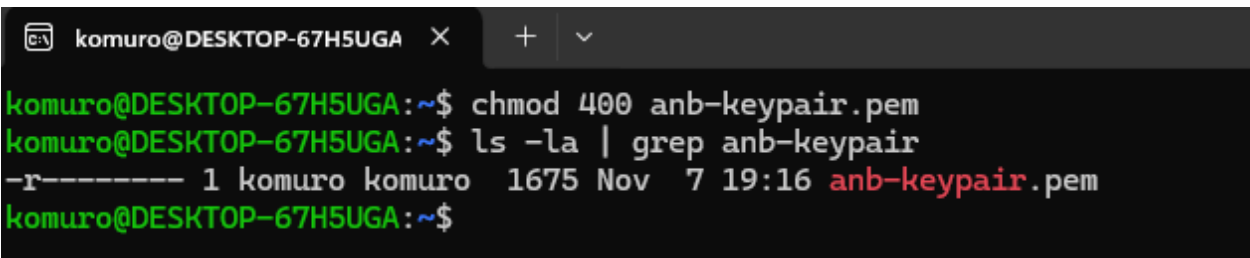
```
aws ec2 create-key-pair \  
--key-name anb-keypair \  
--query 'KeyMaterial' \  
--output text > anb-keypair.pem
```



```
komuro@DESKTOP-67H5UGA:~$ aws ec2 create-key-pair \  
--key-name anb-keypair \  
--query 'KeyMaterial' \  
--output text > anb-keypair.pem
```

Figura 4. Creación de llave

```
chmod 400 anb-keypair.pem
```



```
komuro@DESKTOP-67H5UGA:~$ chmod 400 anb-keypair.pem  
komuro@DESKTOP-67H5UGA:~$ ls -la | grep anb-keypair  
-r----- 1 komuro komuro 1675 Nov  7 19:16 anb-keypair.pem  
komuro@DESKTOP-67H5UGA:~$
```

Figura 4. Asignación de permisos a la llave

Paso 4 – Generación del secreto JWT y la contraseña para RDS

En este paso se generan las credenciales necesarias para la autenticación y el acceso seguro a los servicios de la aplicación. Es importante recordar que estos valores deben manejarse mediante variables de entorno y nunca almacenarse directamente en el código fuente, con el fin de proteger la información sensible.

El secreto JWT (*JSON Web Token Secret*) se utiliza para firmar y verificar los tokens de autenticación. Se recomienda generar un valor aleatorio seguro utilizando openssl:

- # Generar JWT Secret de 32 bytes en formato hexadecimal

```
JWT_SECRET=$(openssl rand -hex 32)
```

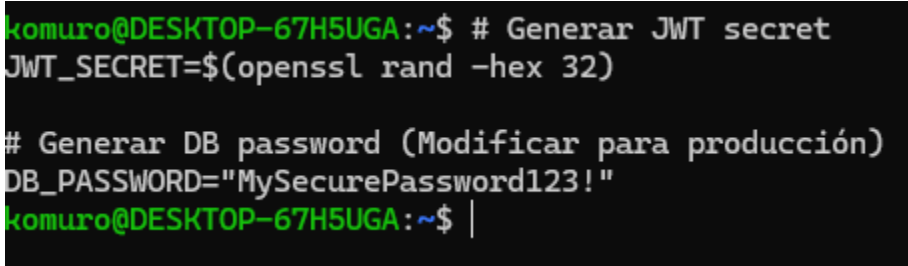
Generar la contraseña para la base de datos RDS

La contraseña del usuario de la base de datos RDS debe ser segura y cumplir con las políticas de contraseñas de AWS (mínimo 8 caracteres, incluyendo mayúsculas, minúsculas, números y símbolos).

Durante el entorno de desarrollo o pruebas se puede asignar una contraseña fija; sin embargo, en producción debe generarse dinámicamente y almacenarse de forma segura.

Generar contraseña segura para RDS (solo para entornos de desarrollo)

DB_PASSWORD="MySecurePassword123!"



```
komuro@DESKTOP-67H5UGA:~$ # Generar JWT secret
JWT_SECRET=$(openssl rand -hex 32)

# Generar DB password (Modificar para producción)
DB_PASSWORD="MySecurePassword123!"
komuro@DESKTOP-67H5UGA:~$ |
```

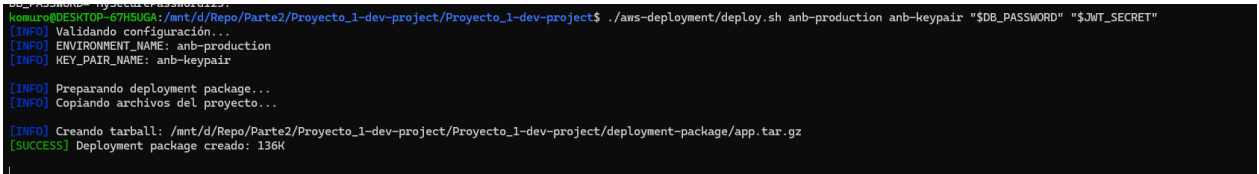
Figura 5. Generación de claves

Paso 5 – Ejecución del script de despliegue (Cloudformation)

Una vez configuradas las credenciales y variables de entorno necesarias, se procede a realizar el despliegue automatizado de la aplicación sobre la infraestructura de AWS. Para ello, se utiliza el script `deploy.sh`, ubicado dentro del directorio `aws-deployment` del proyecto.

Este script tiene como objetivo provisionar los recursos en AWS (instancias EC2, balanceador de carga, grupos de *autoscaling*, entre otros) y configurar la aplicación web junto con su capa *worker* para el entorno de producción.

`./aws-deployment/deploy.sh` `anb-production` `anb-keypair` `"$DB_PASSWORD"`
`"$JWT_SECRET"`



```
komuro@DESKTOP-67H5UGA:/mnt/d/Repo/Parte2/Proyecto_1-dev-project/Proyecto_1-dev-project$ ./aws-deployment/deploy.sh anb-production anb-keypair "$DB_PASSWORD" "$JWT_SECRET"
[INFO] Validando configuración...
[INFO] ENVIRONMENT_NAME: anb-production
[INFO] KEY_PAIR_NAME: anb-keypair

[INFO] Preparando deployment package...
[INFO] Copiando archivos del proyecto...

[INFO] Creando tarball: /mnt/d/Repo/Parte2/Proyecto_1-dev-project/Proyecto_1-dev-project/deployment-package/app.tar.gz
[SUCCESS] Deployment package creado: 136K
```

Figura 6. Deploy de infraestructura

Descripción de los parámetros:

- `anb-production` → nombre del entorno de despliegue. En este caso, corresponde al entorno de producción configurado en AWS.
- `anb-keypair` → nombre del key pair registrado en AWS EC2, utilizado para la autenticación SSH de las instancias creadas.
- `"$DB_PASSWORD"` → contraseña de la base de datos RDS, generada en el paso anterior. Se pasa como argumento para su uso seguro dentro del script.
- `"$JWT_SECRET"` → clave secreta utilizada para la generación y validación de tokens JWT en la aplicación.
-

Consideraciones importantes

- Asegúrese de ejecutar el comando desde la raíz del proyecto para que las rutas relativas dentro del script funcionen correctamente.
- Verifique que el *key pair* (`anb-keypair`) exista en la cuenta de AWS y esté correctamente configurado en la región seleccionada.
- Los valores de las variables `$DB_PASSWORD` y `$JWT_SECRET` deben haberse generado previamente y no deben registrarse en texto plano en ningún archivo del repositorio.
- Durante la ejecución, el script puede tardar varios minutos, ya que realizará el aprovisionamiento y la configuración completa de los recursos en AWS.

Paso 6 – Verificación del despliegue en AWS CloudFormation

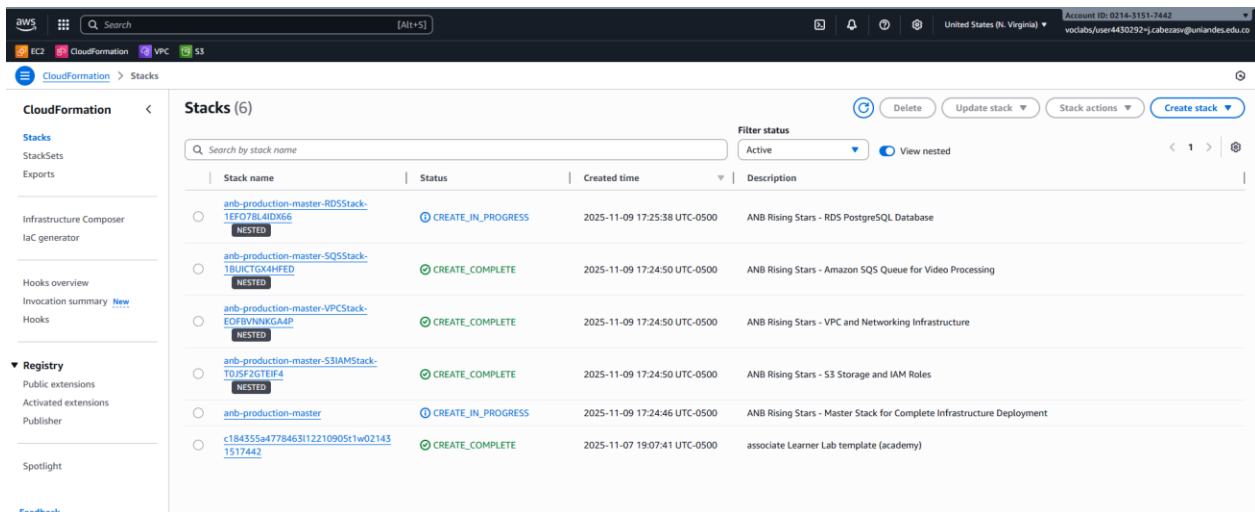
Una vez ejecutado el script `deploy.sh`, se inicia el proceso de provisionamiento automatizado de los recursos definidos en las plantillas de infraestructura. Este proceso puede observarse desde el servicio AWS CloudFormation, donde se crean las stacks correspondientes a cada componente del sistema.

En la consola de AWS, dentro de la sección CloudFormation → Stacks, se visualizan los diferentes módulos desplegados bajo el entorno `anb-production`. Entre ellos se encuentran:

- `anb-production-master` → Stack principal que coordina la creación de los recursos dependientes.
- `anb-production-master-VPCStack` → Capa de red, encargada de la configuración de la VPC, subredes y reglas de seguridad.

- `anb-production-master-S3IAMStack` → Configura los buckets de Amazon S3 y los roles de IAM necesarios para la aplicación.
- `anb-production-master-SQSStack` → Implementa la cola de mensajería Amazon SQS utilizada para el procesamiento de tareas batch/worker.
- `anb-production-master-RDSStack` → Despliega la base de datos PostgreSQL en Amazon RDS.
- Plantilla asociada al laboratorio (`c184355a...`) → Corresponde al entorno base proporcionado por AWS Academy.

En la captura, puede observarse que varios de los stacks ya presentan el estado `CREATE_COMPLETE`, lo que indica una creación exitosa. Otros, como el `RDSStack` y el `master stack`, pueden encontrarse temporalmente en `CREATE_IN_PROGRESS`, dado que la configuración de la base de datos y los servicios asociados puede tardar algunos minutos.



Stack name	Status	Created time	Description
anb-production-master-RDSStack-1EF078L4IDX66 <small>NESTED</small>	CREATE_IN_PROGRESS	2025-11-09 17:25:38 UTC-0500	ANB Rising Stars - RDS PostgreSQL Database
anb-production-master-SQSStack-1BUJCTGX4H4ED <small>NESTED</small>	CREATE_COMPLETE	2025-11-09 17:24:50 UTC-0500	ANB Rising Stars - Amazon SQS Queue for Video Processing
anb-production-master-VPCStack-E0FBVNNKGA4P <small>NESTED</small>	CREATE_COMPLETE	2025-11-09 17:24:50 UTC-0500	ANB Rising Stars - VPC and Networking Infrastructure
anb-production-master-S3IAMStack-T0J5F2GTEIF4 <small>NESTED</small>	CREATE_COMPLETE	2025-11-09 17:24:50 UTC-0500	ANB Rising Stars - S3 Storage and IAM Roles
anb-production-master	CREATE_IN_PROGRESS	2025-11-09 17:24:46 UTC-0500	ANB Rising Stars - Master Stack for Complete Infrastructure Deployment
c184355a778463122109051w021431517442	CREATE_COMPLETE	2025-11-07 19:07:41 UTC-0500	associate Learner Lab template (academy)

Figura 7. Verificación CloudFormation

Acciones recomendadas después del despliegue

- Esperar la finalización completa del proceso hasta que todos los stacks muestren el estado `CREATE_COMPLETE`.
- Verificar la creación de los recursos asociados a cada componente:
 - En EC2, confirmar que las instancias web y worker se hayan iniciado correctamente.
 - En RDS, revisar el estado de la base de datos y su accesibilidad.
 - En SQS, validar que la cola de mensajes esté activa y visible.

- En S3, comprobar que los buckets creados correspondan a los definidos en el proyecto.
- Registrar los nombres y ARNs de los recursos generados, ya que serán necesarios para configurar variables de entorno y monitoreo.
- Monitorear los logs y métricas en CloudWatch para confirmar que la infraestructura se está ejecutando sin errores.

Paso 7 – Ejecución de las migraciones de base de datos

Una vez completado el despliegue de la infraestructura y verificado que las instancias del grupo de Auto Scaling están saludables, se procede a ejecutar las migraciones SQL sobre la base de datos PostgreSQL desplegada en Amazon RDS.

Stacks (8)

Search by stack name

Filter status: Active View nested

Stack name	Status	Created time	Description
anb-production-master-WorkersStack-1PBZAF8I7GLT NESTED	CREATE_COMPLETE	2025-11-09 17:35:51 UTC-0500	ANB Rising Stars - Auto Scaling Group for Worker Instances (Video Processing)
anb-production-master-ALBAutoScalingStack-CY8AKX89V2OW NESTED	CREATE_COMPLETE	2025-11-09 17:32:19 UTC-0500	ANB Rising Stars - Application Load Balancer and Auto Scaling Group for API Layer
anb-production-master-RDSStack-1EFO78L4IDXG6 NESTED	CREATE_COMPLETE	2025-11-09 17:25:38 UTC-0500	ANB Rising Stars - RDS PostgreSQL Database
anb-production-master-SQSStack-1BUJCTGX4HFED NESTED	CREATE_COMPLETE	2025-11-09 17:24:50 UTC-0500	ANB Rising Stars - Amazon SQS Queue for Video Processing
anb-production-master-VPCStack-E0FBVNNKG44P NESTED	CREATE_COMPLETE	2025-11-09 17:24:50 UTC-0500	ANB Rising Stars - VPC and Networking Infrastructure
anb-production-master-S3IAMStack-T0JSF2GTEIF4 NESTED	CREATE_COMPLETE	2025-11-09 17:24:50 UTC-0500	ANB Rising Stars - S3 Storage and IAM Roles
anb-production-master	CREATE_COMPLETE	2025-11-09 17:24:46 UTC-0500	ANB Rising Stars - Master Stack for Complete Infrastructure Deployment
c184355a477846311221090511w021431517442	CREATE_COMPLETE	2025-11-07 19:07:41 UTC-0500	associate Learner Lab template (academy)

Figura 8. Verificación CloudFormation

Estas migraciones crean las tablas, índices y estructuras requeridas por la aplicación para su correcto funcionamiento.

I. Obtener la información de conexión

Del resultado del despliegue, el sistema muestra los siguientes datos de conexión:

- *Endpoint* RDS: anb-production-postgres.cojtsmcpd5q.us-east-1.rds.amazonaws.com
- *Base de datos*: proyecto_1
- *Usuario*: postgres

Asegúrate de tener la variable de entorno con la contraseña de la base de datos (\$DB_PASSWORD) configurada antes de continuar.

II. Ejecutar las migraciones

En el servidor donde está desplegada la aplicación (o desde un cliente con acceso a la base de datos RDS), ejecuta el siguiente comando:

```
for file in /opt/anb-app/db/*.up.sql; do
  psql -h anb-production-postgres.cojtsmcopd5q.us-east-1.rds.amazonaws.com -U
postgres -d proyecto_1 -f "$file"
done
```

Este script recorre todos los archivos SQL con extensión .up.sql ubicados en el directorio /opt/anb-app/db/ y los ejecuta de manera secuencial sobre la base de datos proyecto_1.

III. Confirmar la correcta aplicación de las migraciones

Para validar que las migraciones se aplicaron correctamente, puedes conectarte al RDS usando psql:

```
psql -h anb-production-postgres.cojtsmcopd5q.us-east-1.rds.amazonaws.com -U postgres
-d proyecto_1
```

Luego, dentro de la consola de PostgreSQL, verifica la existencia de las tablas:

```
\dt
```

Deberías ver listadas las tablas definidas en tus archivos .up.sql.

```
[ec2-user@ip-10-0-1-237 ~]$ psql -h anb-production-postgres.cojtsmcopd5q.us-east-1.rds.amazonaws.com -U postgres -d proyecto_1
Password for user postgres:
psql (15.14, server 15.12)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, compression: off)
Type "help" for help.

proyecto_1=> \dt
          List of relations
Schema |      Name      | Type  | Owner
-----+-----+-----+-----
public | task_results   | table | postgres
public | user_sessions  | table | postgres
public | users          | table | postgres
public | videos         | table | postgres
public | votes          | table | postgres
(5 rows)

proyecto_1=> \
```

Figura 8. Verificación importación base de datos en el RDS

Comprobación del despliegue

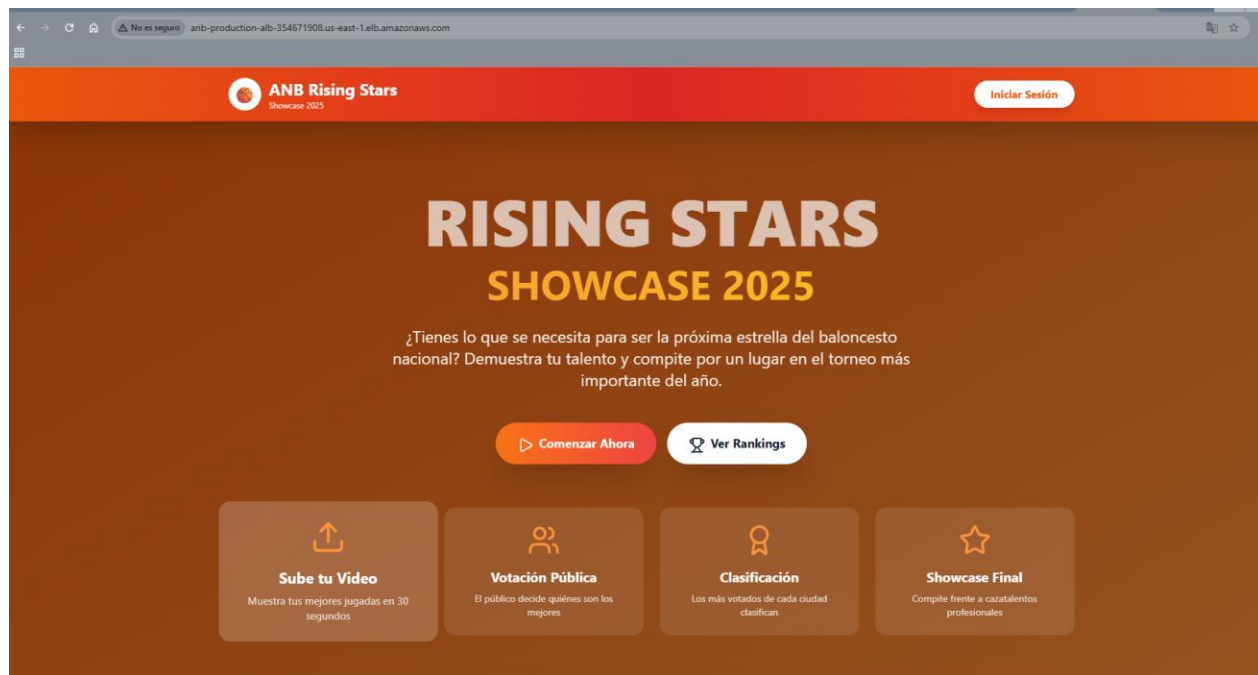


Figura 9. Portal de votación

Con esto se valida que el despliegue de la aplicación se realizó de manera exitosa. El siguiente paso consiste en verificar el funcionamiento operativo del sitio, asegurando que la comunicación con el servicio **Amazon SQS** se encuentre activa y que las tareas en la capa *worker* estén procesando correctamente los mensajes enviados desde la API web.

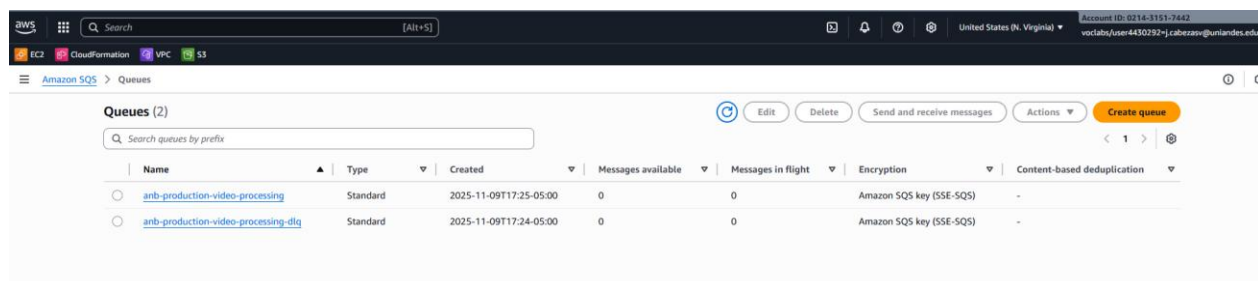


Figura 10. Amazon SQS

Funcionamiento de la aplicación

El funcionamiento de la aplicación inicia con el acceso desde el navegador web a la dirección pública asignada al **WebServer**. Allí se despliega el portal de autenticación (figura

11), el cual permite a los usuarios registrarse (figura 12) e iniciar sesión para acceder a las funcionalidades principales de la solución.

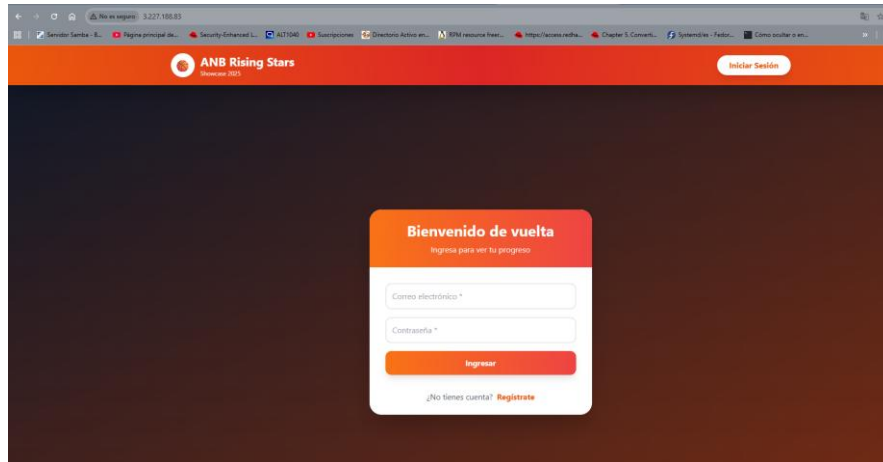


Figura 11. Portal de autenticación

Como se evidencia en la figura 11 el panel de autenticación cuenta con un apartado de registro, esto con el fin de que los usuarios puedan registrarse para iniciar su proceso de votaciones.

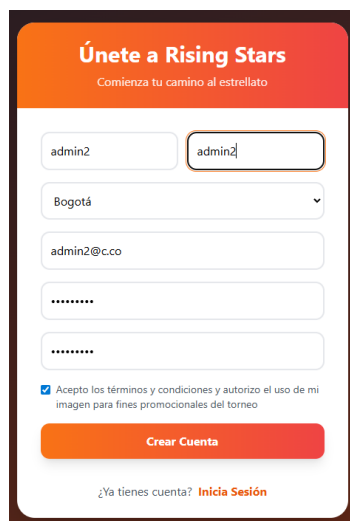


Figura 12. Registro de usuario

Una vez autenticados, los usuarios pueden interactuar con el sistema mediante la API desplegada en la instancia correspondiente. Esta API fue levantada mediante contenedores Docker definidos en el CloudFormation, garantizando la estandarización y portabilidad del servicio.

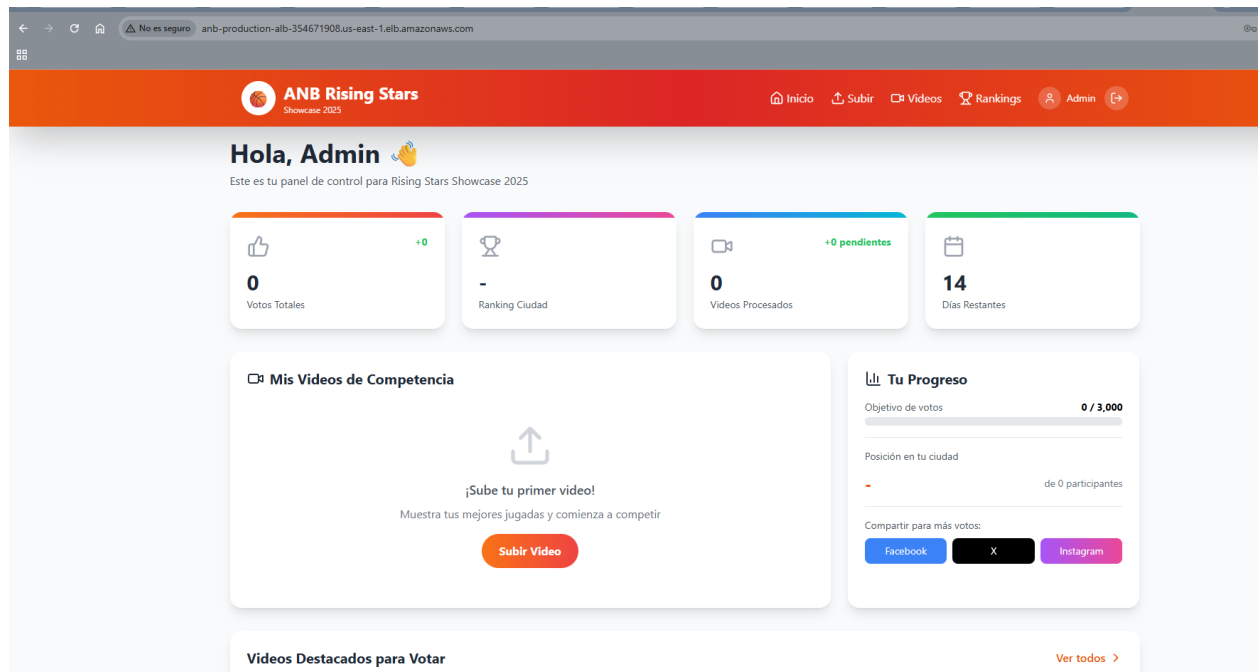


Figura 13. Panel de votación y cargue de video

Dentro de las funciones de carga de video, se ofrece la opción de mantener el video en modo público (visible para votación) o configurarlo como privado para uso interno.

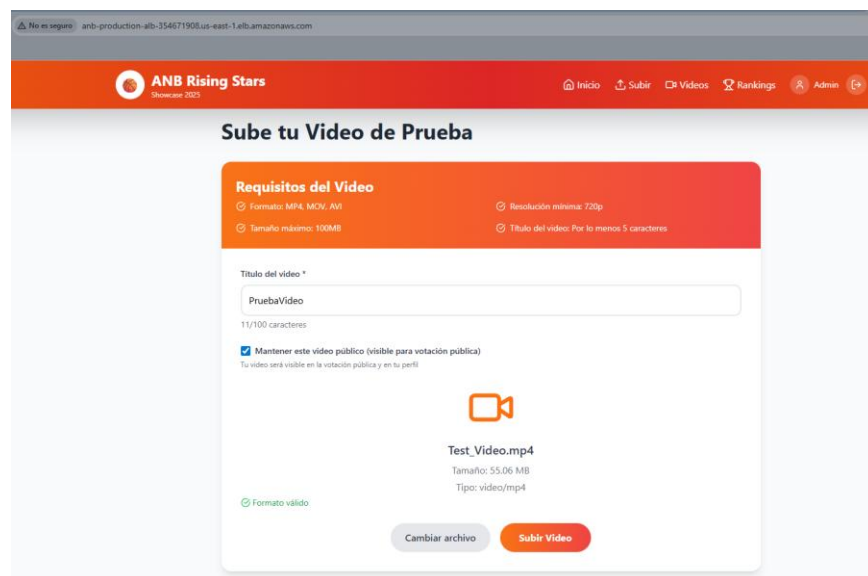


Figura 11. Panel de cargue de video

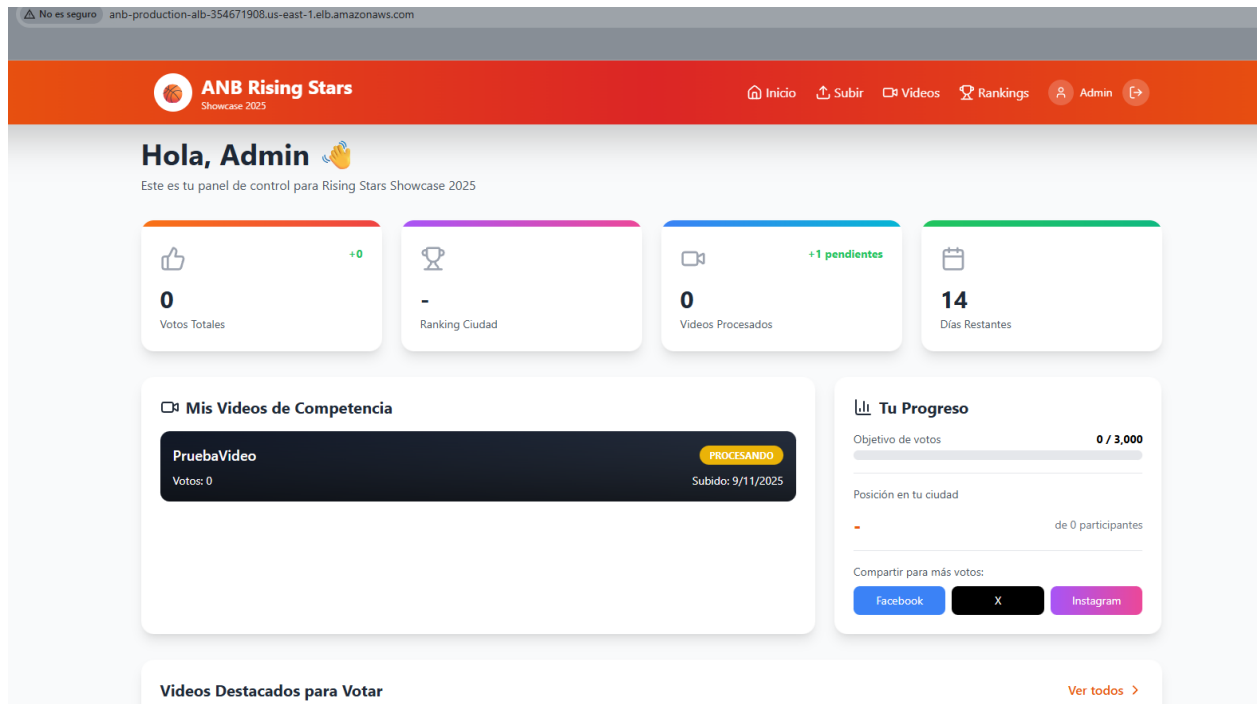


Figura 12. Procesamiento del video

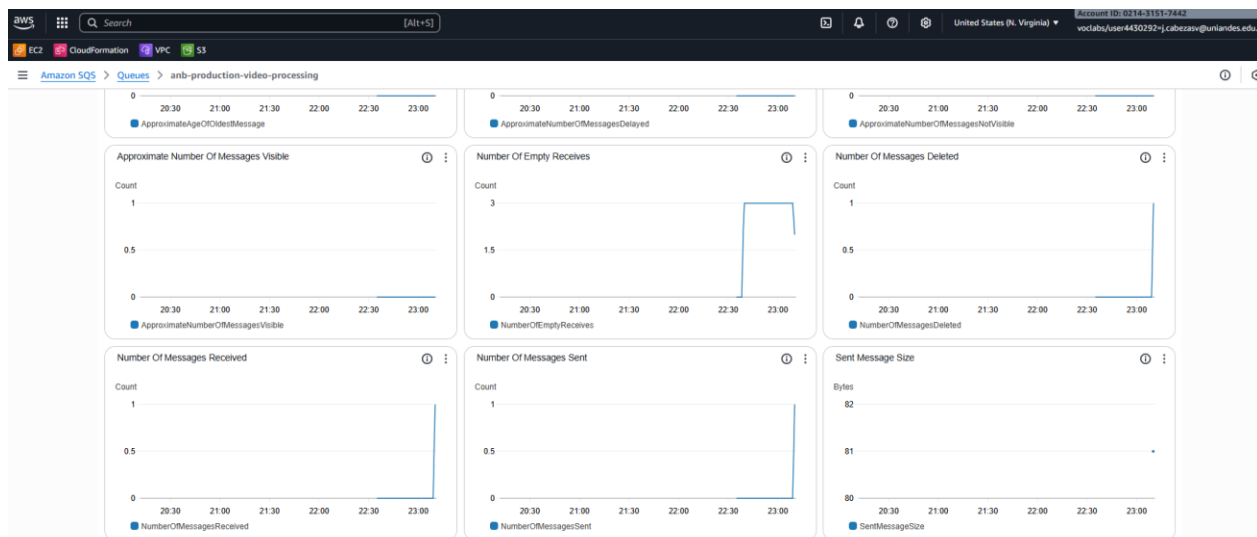


Figura 13. Verificación de procesamiento del video en Amazon SQS

La **Figura 13** presenta la ejecución del servicio **Amazon SQS**, evidenciando que la carga y el procesamiento de mensajes se realizaron exitosamente. Este resultado confirma el correcto funcionamiento del mecanismo de mensajería asíncrona, garantizando una comunicación efectiva entre la capa web y los procesos *worker* dentro de la arquitectura distribuida.

Una vez que los videos son cargados y procesados por los *workers*, los demás usuarios pueden visualizarlos en la plataforma y emitir sus votos sin interrupciones.

Gracias a esta separación de responsabilidades entre los componentes, la aplicación permite que las acciones de los usuarios —como subir contenido, votar por los videos o consultar resultados en tiempo real— se ejecuten de forma fluida, mientras que el procesamiento pesado se realiza de manera independiente y confiable en los *workers* administrados a través de **Amazon SQS**.

Con ello, se confirma que la **aplicación de votación de videos** opera correctamente sobre la arquitectura distribuida desplegada en **AWS**, demostrando una integración exitosa entre los servicios de cómputo, mensajería y procesamiento en la nube.

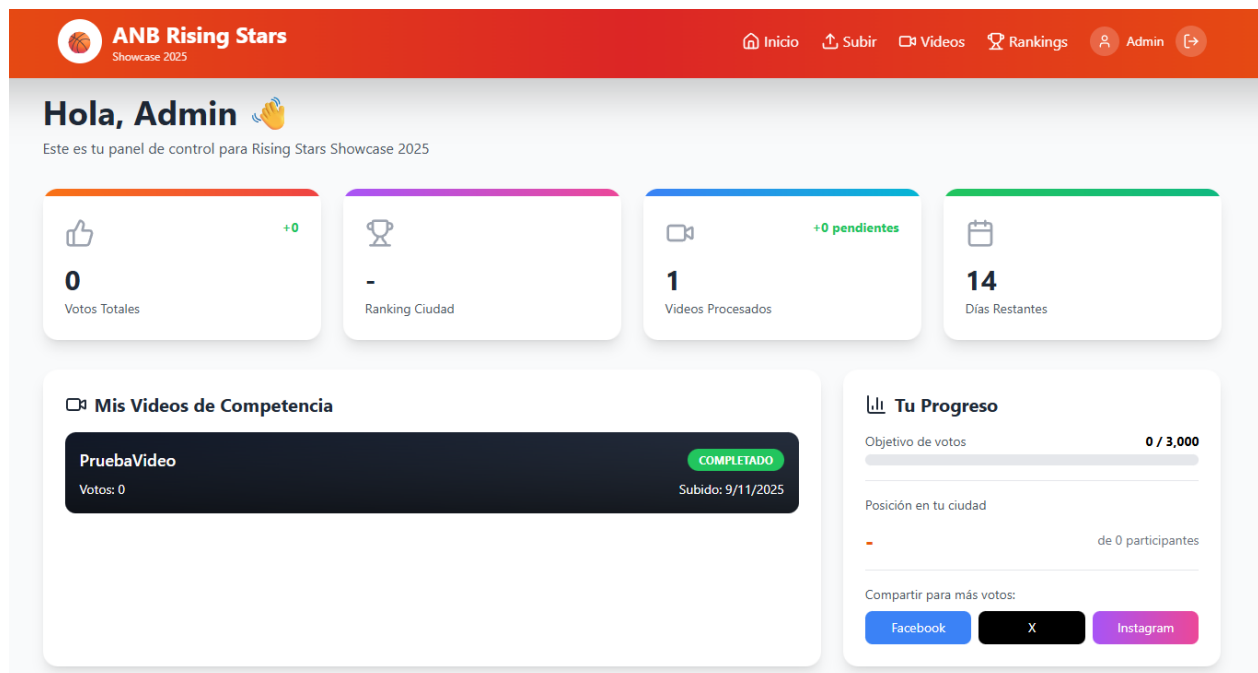


Figura 14. Confirmación de carga del video

Repositorio: [GitHub - development-cloud-solutions/Proyecto_1](https://github.com/development-cloud-solutions/Proyecto_1)

CONCLUSIONES

- La implementación de una arquitectura escalable en la nube permite que las aplicaciones web soporten cargas variables de trabajo sin comprometer el rendimiento ni la disponibilidad del servicio.
- El uso de herramientas como Auto Scaling, SQS y ELB en AWS demuestra la efectividad de los enfoques de escalabilidad horizontal y la importancia de la automatización en el manejo de recursos.
- Las pruebas de estrés evidenciaron la necesidad de un monitoreo continuo y ajustes periódicos en las políticas de escalado para mantener un equilibrio entre costos y desempeño.
- Esta práctica reafirma que la adopción de servicios gestionados en la nube facilita el desarrollo de soluciones robustas, flexibles y sostenibles en entornos de producción reales.

BIBLIOGRAFÍA

- Amazon Web Services Academy. (s. f.). *[Laboratorio de aprendizaje de AWS Academy]*. AWS Academy. Recuperado el 29 de septiembre de 2025, de <https://awsacademy.instructure.com/courses/130819/modules/items/12511346>