

Informe:

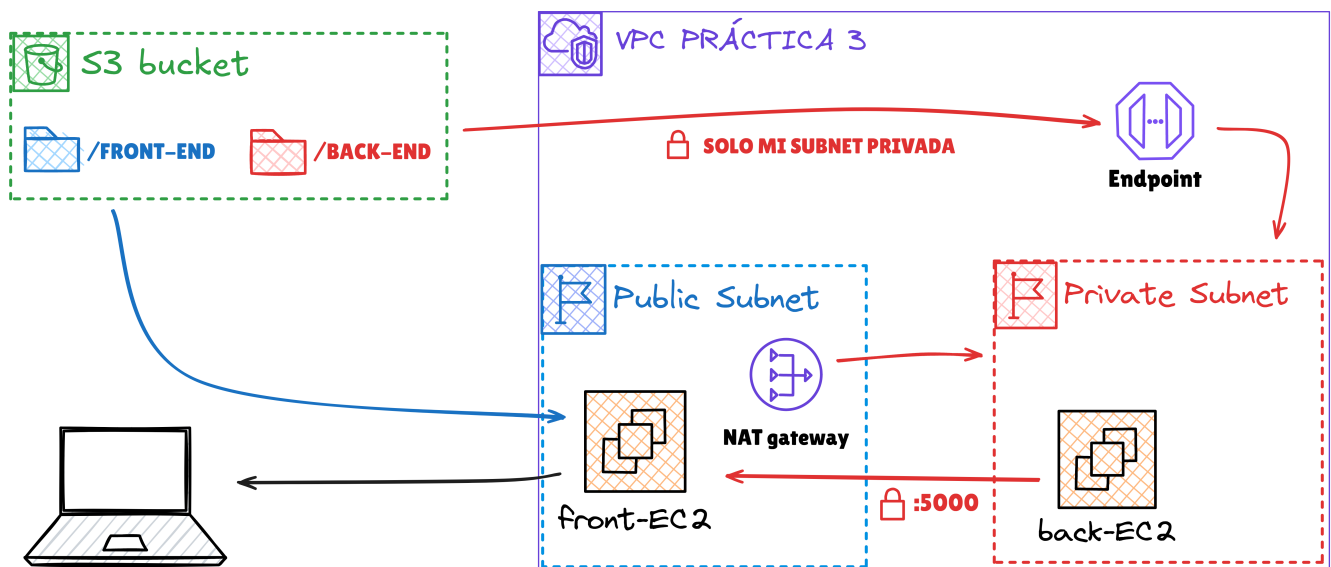
Práctica 3: Redes Virtuales y Seguridad de Acceso

Jordi Blasco Lozano
Infraestructuras y Servicios Cloud
Universidad de Alicante

2 de octubre de 2025

Resumen

Esta práctica tiene como objetivo comprender y configurar la estructura de una VPC con redes públicas y privadas, gestionando la conexión a recursos externos y minimizando las posibles brechas de seguridad. Para el desarrollo de la práctica se llevará a cabo la estructura que se muestra en el siguiente esquema.



Índice

1. Configuración de la VPC	3
1.1. Creación de las subredes	3
1.2. Endpoint	3
1.3. NAT gateway	4
1.4. Tablas de enrutamiento	5
2. Configuración del bucket S3	5
2.1. /Front-end	5
2.2. /Back-end	6
2.3. Permisos y políticas	6
3. Instancias	7
3.1. Grupos de Seguridad	7
3.2. Instancia backend	8
3.2.1. Front User Data	9
3.2.2. Back Logs	9
3.3. Instancia frontend	10
3.3.1. Front User Data	11
3.3.2. Front Logs	11
4. Mejora de la aplicación	12
4.1. Frontend	12
4.2. Backend	13
5. Pruebas	13
5.1. Accesos al bucket	13
5.2. Prueba de despliegue 1	13
5.3. Prueba de despliegue 2	13

1 Configuración de la VPC

Una VPC es una red virtual aislada, cada cuenta de AWS contiene por defecto una VPC. Para poder controlar de una mejor forma las subredes que la conforman, los endpoints, las tablas de enrutamiento y los distintos servicios asociados que contienen, crearemos una nueva VPC y así tendremos acceso a la configuración de la VPC desde el principio.

1.1 Creación de las subredes

Para la práctica propuesta necesitaremos crear dos subredes, una pública donde tendremos nuestra instancia EC2 de frontend y el NAT Gateway. La instancia se utilizará para desplegar nuestra web y el Nat Gateway para conectar a internet las instancias de la subred privada. Finalmente la subred privada contará únicamente con la instancia del backend. Esta deberá de poder descargar librerías de internet y conectarse a la carpeta de backend del bucket S3 para desplegar la aplicación.

Configuraremos la VPC con las dos subredes publicas y dos privadas de forma que tengamos dos zonas de disponibilidad para mejorar la estabilidad del sistema. Por ahora no configuraremos el Nat Gateway, lo crearemos posteriormente.

The screenshot shows the AWS Management Console interface for creating a new VPC. The left sidebar contains navigation links for 'VPC', 'Sub VPC', and 'Crear VPC'. The main content area is titled 'Configuración de la VPC' and includes several sections for configuring the VPC. The 'Recursos que se van a crear' section has two radio buttons: 'Solo la VPC' and 'VPC y más', with 'VPC y más' selected. The 'Generación automática de etiquetas de nombre' section has a checkbox for 'Generar automáticamente' which is checked, and a text input field containing 'proyecto'. The 'Bloque de CIDR IPv4' section has a text input field for the CIDR block, with '10.0.0.0/16' entered. The 'Bloque de CIDR IPv6' section has two radio buttons: 'Sin bloque de CIDR IPv6' (selected) and 'Bloque de CIDR IPv6 proporcionado por Amazon'. The 'Tenencia' section has a dropdown menu set to 'Predeterminado'. The 'Número de zonas de disponibilidad (AZ)' section has three radio buttons: '1', '2' (selected), and '3'. The right sidebar shows a 'Vista previa' (Preview) section with a diagram of the VPC structure. The diagram shows a VPC named 'proyecto-vpc' with four subnets: 'us-east-1a' (public), 'us-east-1b' (public), 'proyecto-subnet-private1-us-east-1a', and 'proyecto-subnet-private2-us-east-1b'. There are also three route tables: 'proyecto-rtb-public', 'proyecto-rtb-private1-us-east-1a', and 'proyecto-rtb-private2-us-east-1b'.

Figura 1: Configuración de la VPC

1.2 Endpoint

Para que la instancia de la sub red privada descargue los archivos que tiene que ejecutar debe de poder conectarse a la parte privada del bucket (la carpeta /back-end), para ello lo primero que debemos de configurar será nuestra puerta de enlace (endpoint). El endpoint nos servirá como credencial para que las políticas del bucket nos proporcionen permisos al entrar desde este endpoint. El endpoint conectará las dos subredes privadas al completo con el servicio de AWS S3, de forma que permita la carga y descarga de recursos a cualquier bucket S3 dentro de la región de la VPC.

En el apartado de crear punto de conexión elegiremos servicios AWS, seleccionaremos el servicio de S3 tipo Gateway, después escogemos nuestra VPC y por ultimo lo conectaremos a nuestras dos subredes privadas.

Servicios (1/2)

Nombre del servicio	Propietario	Tipo	Región de servicio
<input checked="" type="radio"/> com.amazonaws.us-east-1.s3	amazon	Gateway	us-east-1
<input type="radio"/> com.amazonaws.us-east-1.s3	amazon	Interface	us-east-1

Configuración de red
 Seleccione la VPC en la que se va a crear el punto de conexión.
VPC
 Cree el punto de enlace de VPC en la VPC de la misma región de AWS desde la que accederá a un recurso.

Tablas de enrutamiento (2/4) [Información](#)

Nombre	ID de tabla de enrutamiento	Principal	ID asociado
<input type="checkbox"/> -	rtb-0693fd6c2de727c1c	Sí	-
<input checked="" type="checkbox"/> proyecto-rtb-private2-us-east-1b	rtb-07e49a0808f779f06 (proyecto-rtb-private2-us-east-1b)	No	subnet-08b5cff29b4511ec0 (proyecto-subnet-...)
<input type="checkbox"/> proyecto-rtb-public	rtb-0b4bbfddc95faa590 (proyecto-rtb-public)	No	2 subredes
<input checked="" type="checkbox"/> proyecto-rtb-private1-us-east-1a	rtb-0cba116233deba6cd (proyecto-rtb-private1-us-east-1a)	No	subnet-043ed112acad99c18 (proyecto-subne-...)

ⓘ Cuando se usa un punto de enlace, las direcciones IP de origen de las instancias de las subredes afectadas para el acceso al servicio de AWS en la misma región serán direcciones IP privadas, no públicas. Es posible que se descarten las conexiones existentes desde las subredes afectadas al servicio de AWS que usen direcciones IP públicas. Asegúrese de no tener tareas críticas en ejecución cuando cree o modifique un punto de enlace.

Figura 2: Panel de crear endpoint

1.3 NAT gateway

Para que una instancia en la subred privada pueda acceder a Internet (en nuestro caso, para descargar librerías), se debe usar un NAT Gateway. El NAT Gateway se despliega en una subred pública y utiliza el Internet Gateway (IGW) de la VPC. El IGW dirige el tráfico de Internet hacia el NAT Gateway, permitiendo la comunicación de carga y descarga saliente a Internet. Posteriormente veremos como conectarnos mediante las tablas de enrutamiento desde la subred privada.

Podríamos no tener que configurar ningún NAT Gateway si las librerías que vayamos a usar las tuviéramos directamente en el bucket S3, de forma que solo nos haría falta el Endpoint S3 y nos saldría más barato el sistema en una práctica real.

Para crear el NAT Gateway debemos de clicar en el botón de crear gateway NAT y seleccionar la red publica de nuestra VPC. Para un entorno real debemos de crear un NAT Gateway por zona de disponibilidad, es decir, en nuestro caso dos.

Crear gateway NAT [Información](#)

Servicio administrado de traducción de direcciones de red (NAT) de alta disponibilidad que las instancias de subredes privadas pueden utilizar para conectarse a servicios de otras VPC, redes locales o Internet.

Configuración de gateway NAT

Nombre - opcional
 Cree una etiqueta con una clave de "Nombre" y el valor que especifique.

 El nombre puede tener un máximo de 256 caracteres.

Subred
 Seleccione una subred en la que va a crear la gateway NAT.

Tipo de conectividad
 Seleccione un tipo de conectividad para la gateway NAT.
☒ Pública
☐ Privada

ID de asignación de IP elástica [Información](#)
 Asigne una dirección IP elástica a la gateway NAT.

Configuraciones adicionales [Información](#)

Etiquetas
 Una etiqueta es una marca que se asigna a un recurso de AWS. Cada etiqueta consta de una clave y un valor opcional. Puede utilizar las etiquetas para buscar y filtrar sus recursos o hacer un seguimiento de los costos de AWS.

Clave

Valor - opcional

Puede agregar 49 más etiquetas.

Figura 3: Panel de crear gateway NAT

1.4 Tablas de enrutamiento

Los dos servicios que hemos visto anteriormente (El Endpoint y el NAT Gateway) debemos de vincularlos para que la subred privada tenga acceso.

La forma de hacer esto es entrando dentro de las tablas de enrutamiento de las dos subredes privadas. El Endpoint S3 se vincula automáticamente al elegir las subredes en el momento de crearlo, y el NAT Gateway debemos de añadirlo editando las rutas de las tablas de enrutamiento y ingresando el nat-id que hemos creado.

Destino	Destino	Estado	Propagada	Route Origin
pl-63a5400a 10.0.0.0/16	vpce-0729f3d641ddc68a8	Activo	No	CreateRoute
	local	Activo	No	CreateRouteTable
0.0.0.0/0	Puerta de enlace NAT	Activo	No	CreateRoute

Figura 4: Panel de editar rutas

2 Configuración del bucket S3

El bucket S3 debe de disponer de todo el código y archivos que necesiten las instancias para poder ser ejecutadas y funcionar correctamente. Al mismo tiempo debemos asegurarnos de restringir el acceso para evitar accesos no autorizados y minimizar el riesgo de ataque informático. Crearemos un bucket con un nombre único (el mio se llamará `backeet-p3`) y usaremos el almacenamiento de uso general. Posteriormente configuraremos la seguridad del almacenamiento.

Crear bucket

Los buckets son contenedores de datos almacenados en S3.

Configuración general

Región de AWS
EE.UU. Este (Norte de Virginia) us-east-1

Tipo de bucket

☒ **Uso general**
Recomendado para la mayoría de los casos de uso y patrones de acceso. Los buckets de uso general son del tipo de bucket de S3 original. Permiten una combinación de clases de almacenamiento que almacenan objetos de forma redundante en múltiples zonas de disponibilidad.

☐ **Directorio**
Recomendado para casos de uso de baja latencia. Estos buckets utilizan únicamente la clase de almacenamiento S3 Express One Zone, que proporciona un procesamiento más rápido de los datos dentro de una única zona de disponibilidad.

Nombre del bucket

bucket-p3

Copiar la configuración del bucket existente: *opcional*
Solo se copia la configuración del bucket en los siguientes ajustes.

Elegir el bucket

Formato: s3://bucket/prefijo

Figura 5: Panel de crear bucket

2.1 /Front-end

Los archivos de `index.html` y `app_front.js` los guardaremos en una carpeta pública, la carpeta `front-end`, esta carpeta será accesible desde cualquier lugar. De esta forma no hace falta utilizar ningún endpoint que sirva como credencial, ni ningún rol para descargar estos archivos en nuestra instancia.

Primero le daremos a crear carpeta y posteriormente le asignaremos un nombre y cargaremos los archivos mencionados anteriormente.

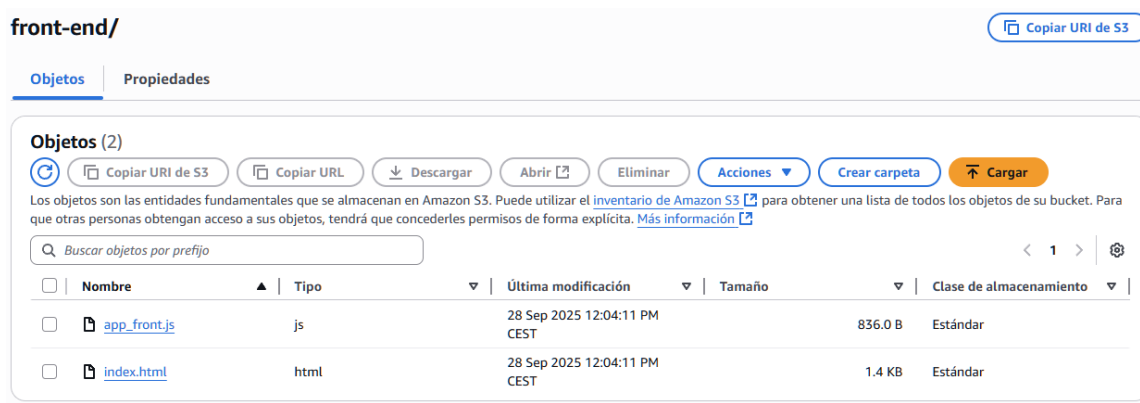


Figura 6: Archivos S3 /front-end

2.2 /Back-end

Para la parte de la lógica de negocio y datos sensibles (como nuestro modelo de inferencia), crearemos otra carpeta, esta privada, la carpeta `back-end`. En nuestro caso los archivos “`app_backend.py`” y el “`modelo.pkl`” son los más críticos, por lo que debemos de controlar estrictamente el acceso a esta carpeta.

Haremos lo mismo que con la carpeta anterior, la crearemos y subiremos nuestros archivos. La seguridad de esta carpeta la manejaremos en el paso siguiente.

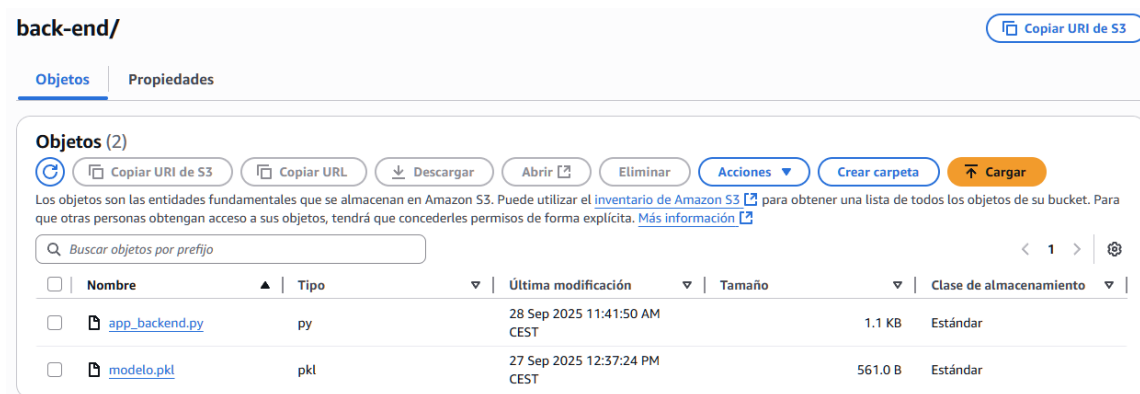


Figura 7: Archivos S3 /back-end

2.3 Permisos y políticas

Esta es la parte más importante en la creación del bucket. Anteriormente al crear el bucket, por defecto hemos bloqueado el acceso público al S3. Lo que debemos de hacer ahora es abrir los accesos, y mediante la política del bucket permitir únicamente el acceso a descargar los archivos de la carpeta /back-end que provengan del Endpoint que hemos creado anteriormente. Esto lo que hará será permitir los accesos a la carpeta de todas las instancias que se encuentren dentro de nuestra red privada.

Al crear una política restringida, la carpeta de /front-end no será accesible, ya que se aplica el principio de denegación por defecto, esto significa que si una acción no está explícitamente permitida, estará denegada. Por lo que permitiremos el acceso a descargar los archivos de la carpeta /front-end a todos los usuarios también en la política del bucket.

Nos dirigiremos dentro de nuestro bucket a permisos, desbloquearemos el permiso de accesos públicos y editaremos la política para que la acción GetObject se permita utilizar dentro de baqueet-p3/back-end/ solamente desde nuestra vpce anterior añadiendo la condición de StringEquals. Posteriormente hacemos

lo mismo con la carpeta del front-end pero sin usar ninguna condición. De este modo queda la carpeta back-end restringida y accesible únicamente mediante el subnet privado y la carpeta front-end abierta a todo el mundo.

Otra forma de hacerlo sin usar roles sería usando la ip privada de la instancia y hacer que se permita el acceso solamente si coincide la ip de la política con la ip de la instancia. Esto no es muy util en nuestro caso ya que estaríamos cambiando la política todo el rato y no podríamos usar de manera correcta los UserData de las instancias.

Bloque 1: política-s3.json

```
1 {
2   "Version": "2012-10-17",
3   "Statement":
4   [
5     {
6       "Sid": "AllowBackendFromVPCEndpoint",
7       "Effect": "Allow",
8       "Principal": "*",
9       "Action": "s3:GetObject",
10      "Resource": "arn:aws:s3:::baqueet-p3/back-end/*",
11      "Condition":
12      {
13        "StringEquals":
14        {"aws:SourceVpce": "vpce-05b2e68f0d4435792"}
15      }
16    },
17    {
18      "Sid": "PublicReadForFrontEndAssets",
19      "Effect": "Allow",
20      "Principal": "*",
21      "Action": "s3:GetObject",
22      "Resource": "arn:aws:s3:::baqueet-p3/front-end/*"
23    }
24  ]
25 }
```

3 Instancias

Las instancias serán desplegadas dentro de nuestra VPC, la instancia que ejecute el front será desplegada en la subnet pública mientras que la instancia que ejecute el back será ejecutada en la subnet privada. Para que sea menos tedioso el despliegue utilizaremos los User Data, es un archivo .sh que se ejecutan al iniciar cada instancia. Estos User Data tendrán sentencias “echo” para que cuando nos metamos a monitorear y comprobar los logs de cada instancia sepamos en que ha fallado de manera rápida y práctica.

3.1 Grupos de Seguridad

Lo primero que tenemos que hacer antes de lanzar las instancias será crear dos grupos de seguridad, uno para el frontend el que permita todas las entradas http y otro para el backend que unicamente permita entradas del grupo de seguridad del frontend mediante un TCP personalizado en el puerto 5000 que es el que usaremos para lanzar el back.

Reglas de entrada Información

ID de la regla del grupo de seguridad	Tipo <small>Información</small>	Protocolo <small>Información</small>	Intervalo de puertos <small>Información</small>	Origen <small>Información</small>	Descripción: opcional <small>Información</small>	
sgr-0b2157912edc578b6	TCP personalizado ▼	TCP	5000	Per... ▼	Q	Eliminar

Agregar regla

sg-0d8cbbd4a708a3417

X

Cancelar

Previsualizar los cambios

Guardar reglas

Reglas de entrada Información

ID de la regla del grupo de seguridad	Tipo <small>Información</small>	Protocolo <small>Información</small>	Intervalo de puertos <small>Información</small>	Origen <small>Información</small>	Descripción: opcional <small>Información</small>	
sgr-000fdd0dafdfb0e31	HTTP ▼	TCP	80	Per... ▼	Q	Eliminar

Agregar regla

0.0.0.0 X

Cancelar

Previsualizar los cambios

Guardar reglas

⚠ Las reglas cuyo origen es 0.0.0.0 o ::/0 permiten a todas las direcciones IP acceder a la instancia. Recomendamos configurar reglas de grupo de seguridad para permitir el acceso únicamente desde direcciones IP conocidas.

Cancelar

Previsualizar los cambios

Guardar reglas

Figura 8: Reglas de los grupos privado y público

3.2 Instancia backend

La instancia backend contará con un par de laves, elegiremos nuestra VPC y nuestra subred privada, posteriormente le asignaremos el grupo de seguridad de la sección anterior y deshabilitaremos la asignación de ip publica ya que nuestra instancia backend solo va a ser accesible mediante su ip privada desde el puerto 5000 y desde el frontend. Es por esto también por lo que creamos primero el backend, asi sabremos nuestra dirección ip privada cuando se despliegue el back y podremos cambiar el archivo del front javascript a la ip correspondiente.

▼ Configuraciones de red Información

VPC : obligatorio Información

vpc-0c6c93ea9601a9f33 (proyecto-vpc)

10.0.0.0/16

↻

Subred Información

subnet-043ed112acad99c18 proyecto-subnet-private1-us-east-1a

VPC: vpc-0c6c93ea9601a9f33 Propietario: 381492194835

Zona de disponibilidad: us-east-1a (use1-az6)

Tipo de zona: Zona de disponibilidad Direcciones IP disponibles: 4091

CIDR: 10.0.128.0/20)

↻

Crear nueva subred [↗](#)

Asignar automáticamente la IP pública Información

Desactivar

▼

Firewall (grupos de seguridad) Información

Un grupo de seguridad es un conjunto de reglas de firewall que controlan el tráfico de la instancia. Agregue reglas para permitir que un tráfico específico llegue a la instancia.

○ Crear grupo de seguridad

● Seleccionar un grupo de seguridad existente

Grupos de seguridad comunes Información

Seleccionar grupos de seguridad

▼

backend-sg sg-02db52395438b3d9c X

VPC: vpc-0c6c93ea9601a9f33

↻

Compare reglas de grupo de seguridad

Los grupos de seguridad que agrega o elimine aquí se agregarán a todas las interfaces de red o se eliminarán de ellas.

► Configuración de red avanzada

Figura 9: Configuración ec2 back

8

3.2.1 Front User Data

El siguiente .sh el user data que voy a utilizar, se configura en la parte final de detalles avanzados al crear la instancia. En él instalaremos todas las dependencias que necesite nuestra aplicación back, crearemos los directorios para alojar los archivos, descargaremos los archivos del S3 y desplegaremos el servidor ejecutando el archivo python. Todo esto se hará automáticamente cuando se inicie la instancia.

Bloque 2: back user data sh

```
#!/bin/bash

echo "--- Iniciando script de User Data para el Back-End ---"

# 1. Actualizar sistema e instalar Python y pip
echo "Actualizando sistema e instalando Python y pip..."
yum update -y
yum install python3-pip -y
echo "Python y pip instalados."

# 2. Instalar dependencias de Python necesarias para la aplicación
echo "Instalando dependencias de Python (Flask, joblib, scikit-learn)..."
pip3 install flask joblib scikit-learn
echo "Dependencias de Python instaladas."

# 3. Crear directorios para la aplicación
echo "Creando directorios de la aplicación en /app..."
mkdir -p /app
cd /app
echo "Directorio creado y nos hemos movido a /app."

# 4. Descargar los archivos del back-end desde S3 usando wget
echo "Descargando archivos del back-end desde S3..."
wget -O app_backend.py https://baqueet-p3.s3.us-east-1.amazonaws.com/back-end/app_backend.py
wget -O modelo.pkl https://baqueet-p3.s3.us-east-1.amazonaws.com/back-end/modelo.pkl

# 5. Verificar que los archivos existen antes de continuar
if [ ! -f "app_backend.py" ] || [ ! -f "modelo.pkl" ]; then
    echo "ERROR: Uno o ambos archivos del back-end no se encontraron después de la descarga."
    exit 1
fi

# 6. Ejecutar el servidor de Flask en segundo plano
echo "Iniciando el servidor de Flask..."
python3 app_backend.py &

echo "--- Script de User Data del Back-End finalizado con éxito. ---"
```

3.2.2 Back Logs

Cuando nos salgan los siguientes logs dentro del monitoreo de la instancia sabremos que tenemos el servidor desplegado sin errores.

Bloque 3: back logs

```
1 [ 44.966280] cloud-init[1633]: * Serving Flask app 'app_backend'
2 [ 44.966623] cloud-init[1633]: * Debug mode: off
3 [ 44.978437] cloud-init[1633]: WARNING: This is a development server. Do not use
  it in a production deployment. Use a production WSGI server instead.
4 [ 44.978585] cloud-init[1633]: * Running on all addresses (0.0.0.0)
5 [ 44.978809] cloud-init[1633]: * Running on http://127.0.0.1:5000
6 [ 44.979093] cloud-init[1633]: * Running on http://10.0.142.191:5000
7 [ 44.979476] cloud-init[1633]: Press CTRL+C to quit
```

3.3 Instancia frontend

Por el otro lado la instancia del frontend contará con el mismo par de llaves y la misma VPC, elegiremos una de las subredes públicas y habilitaremos la opción de asignar IP pública, elegiremos el grupo de seguridad del frontend y cargaremos nuestro User Data.

▼ Configuraciones de red [Información](#)

VPC : obligatorio [Información](#)

vpc-0c6c93ea9601a9f33 (proyecto-vpc)
10.0.0.0/16

↻

Subred [Información](#)

subnet-043ed112acad99c18 proyecto-subnet-private1-us-east-1a
VPC: vpc-0c6c93ea9601a9f33 Propietario: 381492194835
Zona de disponibilidad: us-east-1a (use1-az6)
Tipo de zona: Zona de disponibilidad Direcciones IP disponibles: 4091
CIDR: 10.0.128.0/20

↻ [Crear nueva subred](#)

Asignar automáticamente la IP pública [Información](#)

Desactivar

▼

Firewall (grupos de seguridad) [Información](#)

Un grupo de seguridad es un conjunto de reglas de firewall que controlan el tráfico de la instancia. Agregue reglas para permitir que un tráfico específico llegue a la instancia.

☐ Crear grupo de seguridad

☒ Seleccionar un grupo de seguridad existente

Grupos de seguridad comunes [Información](#)

Seleccionar grupos de seguridad

▼

backend-sg sg-02db52395438b3d9c ✕
VPC: vpc-0c6c93ea9601a9f33

↻ [Compare reglas de grupo de seguridad](#)

Los grupos de seguridad que agrega o elimine aquí se agregarán a todas las interfaces de red o se eliminarán de ellas.

► Configuración de red avanzada

Figura 10: Configuración EC2 back

3.3.1 Front User Data

Para este otro user data debemos de hacer una modificación y pegar la ip privada del backend en la sentencia sed que reemplaza la string de IP_PRIVADA_DEL_BACKEND (que se encuentra dentro del código javascript) por nuestra ip privada. Haciéndolo de este modo nos olvidamos de estar modificando el archivo dentro del bucket S3. El sh hará lo mismo que el anterior, descargar dependencias he instalarlas, descargar el código del S3 y ejecutarlo. Como vemos en los User Data usamos siempre “wget” en vez de “aws s3”, esto lo hacemos porque solamente hemos permitido la acción de GetObject en la política del S3.

Bloque 4: front user data sh

```
#!/bin/bash

echo "--- Iniciando script de User Data para el Front-End ---"

# 1. Actualizar sistema e instalar Node.js desde el repositorio de NodeSource
echo "Actualizando sistema e instalando Node.js..."
yum update -y
# Añadir el repositorio de Node.js 18
curl -fsSL https://rpm.nodesource.com/setup_18.x | bash -
# Instalar Node.js (incluye npm)
yum install -y nodejs
# Instalar dependencias globales
npm install -g express axios
echo "Node.js instalado."

# 2. Crear directorios, descargar archivos e instalar dependencias
echo "Creando directorios de la aplicación en /app..."
mkdir -p /app/public
cd /app
echo "Directorio creado y nos hemos movido a /app."

# 3. Descargar los archivos del front-end desde S3
# ... (tus comandos wget van aquí) ...
wget -O app_front.js https://baqueet-p3.s3.amazonaws.com/front-end/app_front.js
wget -O ./public/index.html https://baqueet-p3.s3.amazonaws.com/front-end/index.html
echo "Archivos descargados."

# Instalar dependencias LOCALMENTE
echo "Instalando dependencias de Node.js..."
npm install express axios
echo "Dependencias instaladas."

# 4. Verificar que los archivos existen antes de continuar
if [ ! -f "app_front.js" ] || [ ! -f "./public/index.html" ]; then
    echo "ERROR: Uno o ambos archivos del front-end no se encontraron después de la descarga."
    exit 1
fi

# 5. Reemplazar la IP del backend en el archivo de la aplicación
echo "Reemplazando la IP del backend"
sed -i 's/IP_PRIVADA_DEL_BACKEND/10.0.142.191/g' app_front.js
echo "IP reemplazada."

# 6. Ejecutar el servidor de Node.js en segundo plano
echo "Iniciando el servidor de Node.js..."

node app_front.js &

echo "--- Script de User Data del Front-End finalizado con éxito. ---"
```

3.3.2 Front Logs

Cuando nos salgan los siguientes logs dentro del monitoreo de la instancia sabremos que tenemos el servidor desplegado sin errores.

Bloque 5: front logs

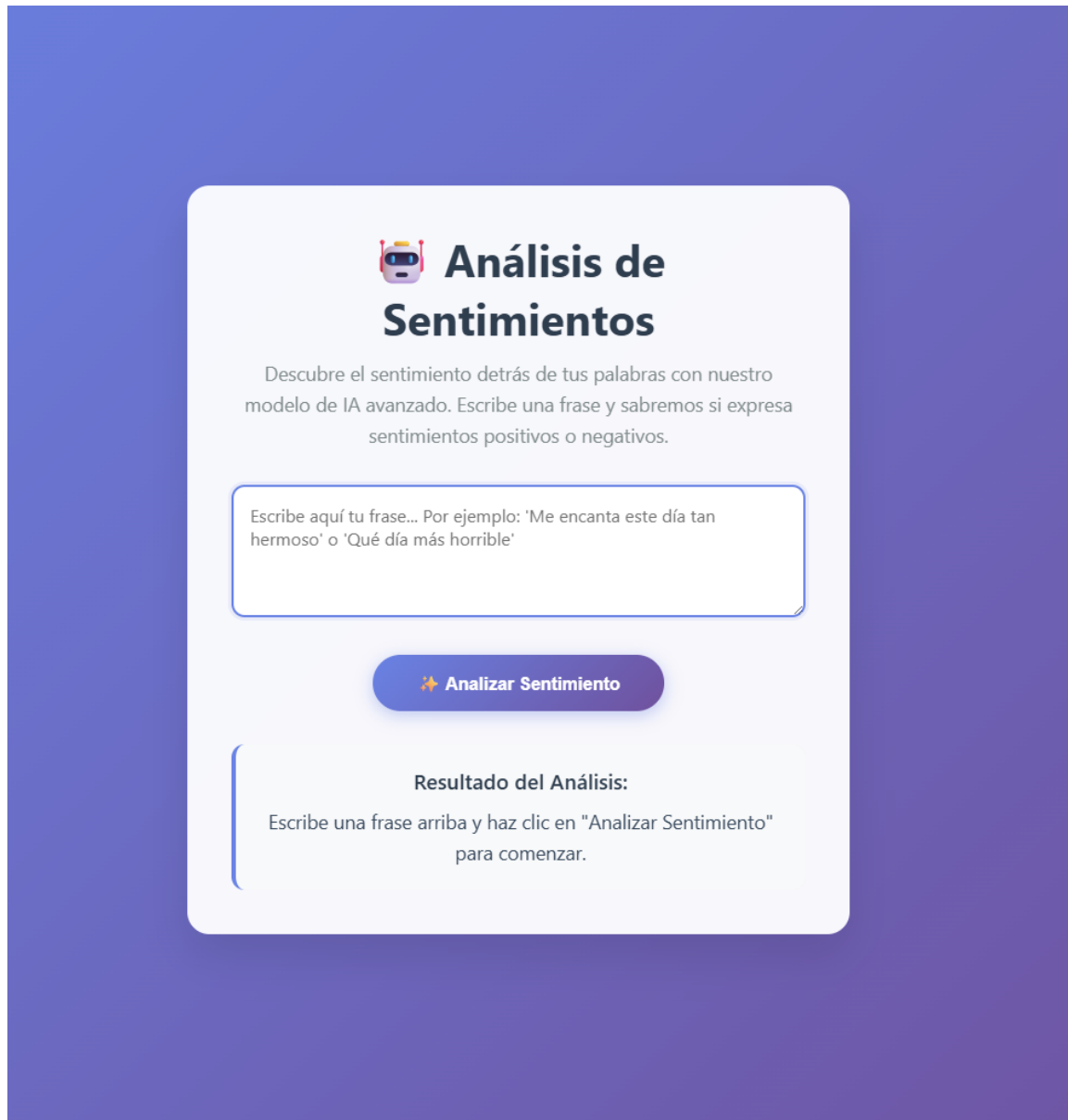
```
1 [ 39.309930] cloud-init[1633]: Cloud-init v. 22.2.2 finished at Wed, 01 Oct 2025
21:42:02 +0000. Datasource DataSourceEc2. Up 39.29 seconds
2 [ 39.687491] cloud-init[1633]: Servidor front-end escuchando en el puerto 80
```


4 Mejora de la aplicación

Para mejorar la aplicación se me ocurrió la idea de lanzar otro tipo de servicio cambiando unicamente la parte de código de front y el modelo. Quería lanzar un modelo barato de ejecutar y a la vez más util que la regresión lineal, así que se me ocurrió usar un modelo de analisis de sentimientos que te digera si una frase de un comentario de cualquier tipo era buena o mala. Para hacer esto simplemente vamos a dejar la estructura del proyecto tal como está y unicamente cambiaremos los archivos del S3.

4.1 Frontend


Para el frontend usaremos un html más bonito con ayuda de la IA y el javascript le cambiamos unicamente el mensaje de error.



 **Análisis de Sentimientos**

Descubre el sentimiento detrás de tus palabras con nuestro modelo de IA avanzado. Escribe una frase y sabremos si expresa sentimientos positivos o negativos.

Escribe aquí tu frase... Por ejemplo: 'Me encanta este día tan hermoso' o 'Qué día más horrible'

 **Analizar Sentimiento**

Resultado del Análisis:

Escribe una frase arriba y haz clic en "Analizar Sentimiento" para comenzar.

Figura 11: index.html

4.2 Backend

5 Pruebas

5.1 Accesos al bucket

5.2 Prueba de despliegue 1

5.3 Prueba de despliegue 2