

Manual de Implementación de Infraestructura Web en AWS

Manual de Implementación de Infraestructura Web en AWS . 1

1. Introducción.....	2
2. Arquitectura General.....	2
3. Creación del Entorno de Base de Datos (RDS)	2
4. Configuración del Acceso a la Base de Datos.....	4
5. Despliegue del Backend en AWS Elastic Beanstalk.....	6
6. Despliegue del Frontend en AWS Elastic Beanstalk	9
7. Prueba Final de la Infraestructura	11
8. Conclusión	11
9. Troubleshooting	11

1. Introducción

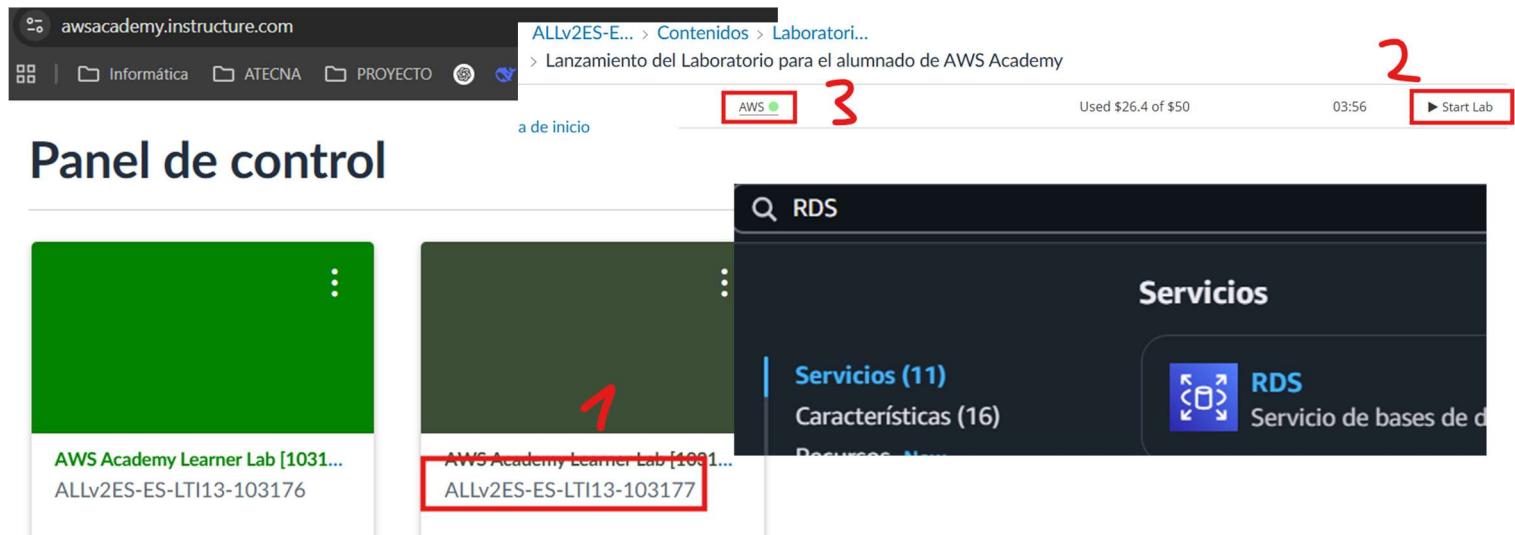
Este documento proporciona una guía detallada para la implantación de una infraestructura basada en AWS utilizando Amazon RDS para la base de datos y Elastic Beanstalk (EBS) para el Backend y Frontend.

2. Arquitectura General

- **Base de Datos:** Amazon RDS para MySQL (HeidiSQL como cliente)
- **Backend:** Web en PHP con Symfony desplegada en AWS Elastic Beanstalk con Docker
- **Frontend:** Aplicación web estática con AngularJS desplegada en AWS Elastic Beanstalk con Docker

3. Creación del Entorno de Base de Datos (RDS)

1. Iniciar sesión en AWS y acceder a Amazon RDS.

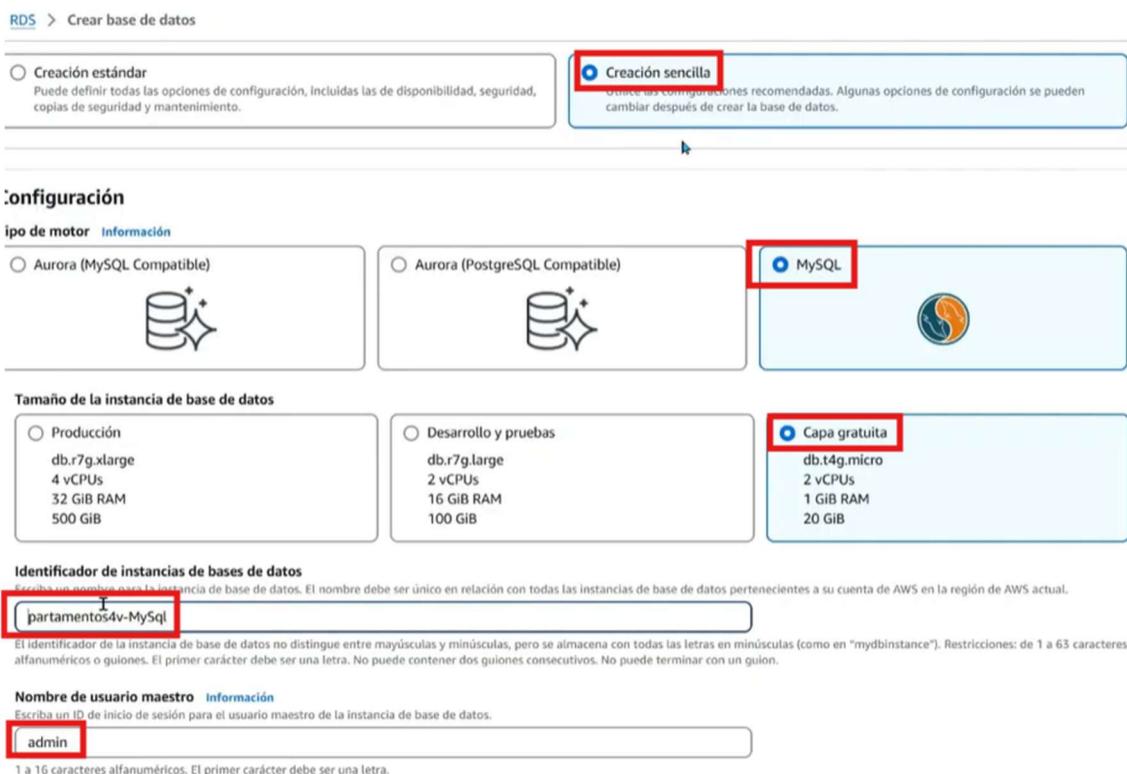


The screenshot shows the AWS Academy interface. A red box labeled '1' highlights the 'Content' section where a lab is listed. A red box labeled '2' highlights the 'Start Lab' button. A red box labeled '3' highlights the 'AWS' icon in the top right corner of the browser window. The browser address bar shows 'awsacademy.instructure.com'. The search bar contains 'RDS'. The sidebar shows 'Servicios (11)', 'Características (16)', and 'Recursos'. The main content area shows the 'RDS' service icon and its description: 'Servicio de bases de datos'. The top right of the screen shows 'Used \$26.4 of \$50' and '03:56'.

En la página de contenidos accedemos al Lanzamiento del Laboratorio y lo lanzamos, buscamos ‘RDS’ en el buscador y entramos.

2. Crear una nueva instancia de RDS con los siguientes parámetros:

Creación sencilla, motor de base de datos: MySQL, capa gratuita, elegimos un nombre y contraseña para la instancia, las demás configuraciones por defecto. Creamos la base de datos (en torno a 5 minutos de espera)



The screenshot shows the 'Create database' wizard in the AWS RDS console. The 'Creation type' section has 'Creation simple' selected (highlighted with a red box). The 'Engine' section has 'MySQL' selected (highlighted with a red box). The 'Instance type' section has 'Free tier' selected (highlighted with a red box), which includes db.t4g.micro, 2 vCPUs, 1 GiB RAM, and 20 GiB storage. The 'Identifier' section shows the identifier 'partamento5v-MySQL' entered in the input field. The 'Master user name' section shows 'admin' entered in the input field.

3. Configurar los **Security Groups**:

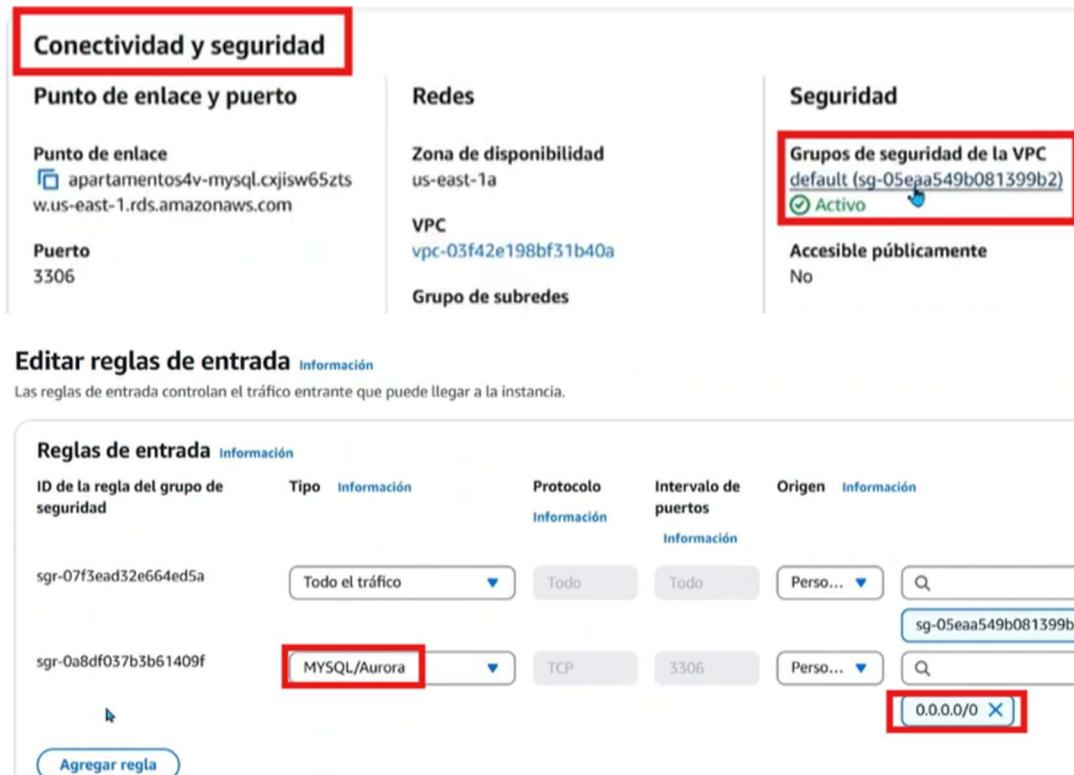
- Agregar una regla de entrada para permitir conexiones en el puerto **3306** desde las direcciones necesarias (Backend y cliente SQL como HeidiSQL).

Una vez creada la BBDD, entramos en ella y seleccionamos modificar. Bajamos hasta donde pone Configuración adicional y seleccionamos Accesible públicamente en el puerto 3306 que viene por defecto.



The screenshot shows the 'Additional Configuration' section of the RDS instance configuration. Under 'Access public' (Acceso público), 'Accessible publicly' (Accesible públicamente) is selected (highlighted with a red box). This option allows RDS to assign a public IP address to the instance. Under 'Port of the database' (Puerto de la base de datos), the port number '3306' is specified.

En el apartado principal de la instancia de BBDD vamos a Conectividad y seguridad y seleccionamos en Grupos de seguridad de la VPC. Aquí, crearemos una regla de entrada para permitir el tráfico de todas las IPs a MySQL



Conectividad y seguridad

Punto de enlace y puerto

- Punto de enlace: apartamentos4v-mysql.cxjsw65zts.us-east-1.rds.amazonaws.com
- Puerto: 3306

Redes

- Zona de disponibilidad: us-east-1a
- VPC: vpc-03f42e198bf31b40a
- Grupo de subredes:

Seguridad

- Grupos de seguridad de la VPC: default (sg-05eaa549b081399b2) (Activo)
- Accesible públicamente: No

Editar reglas de entrada Información

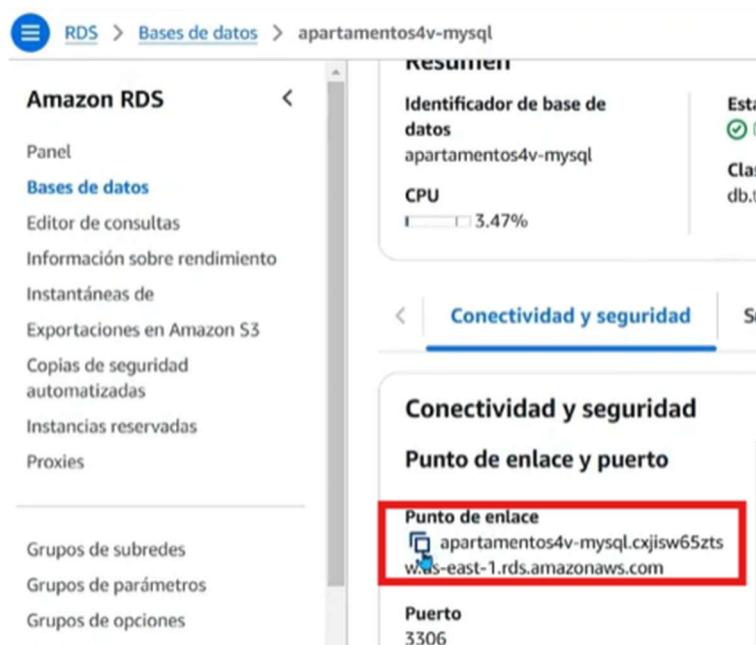
Las reglas de entrada controlan el tráfico entrante que puede llegar a la instancia.

ID de la regla del grupo de seguridad	Tipo	Información	Protocolo	Intervalo de puertos	Origen	Información
sgr-07f3ead32e664ed5a	Todo el tráfico	Todo	Todo	Perso... ▾	<input type="text"/> sg-05eaa549b081399b2 X	
sgr-0a8df037b3b61409f	MySQL/Aurora	TCP	3306	Perso... ▾	<input type="text"/> 0.0.0.0/0 X	

Agregar regla

4. Configuración del Acceso a la Base de Datos

1. Obtener el **endpoint** de la base de datos en la consola de AWS RDS.



RDS > Bases de datos > apartamentos4v-mysql

Resumen

Amazon RDS

- Panel
- Bases de datos**
- Editor de consultas
- Información sobre rendimiento
- Instantáneas de
- Exportaciones en Amazon S3
- Copias de seguridad automatizadas
- Instancias reservadas
- Proxies

Conectividad y seguridad

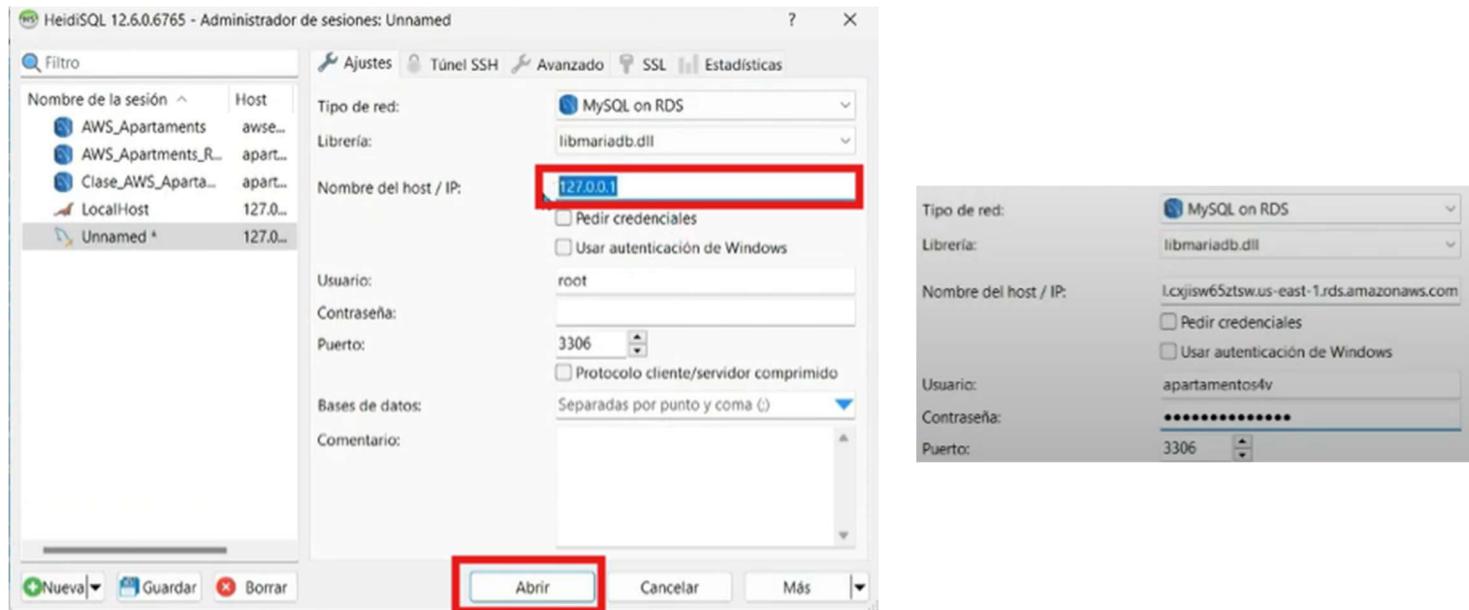
Punto de enlace y puerto

Punto de enlace: apartamentos4v-mysql.cxjsw65zts.us-east-1.rds.amazonaws.com

Puerto: 3306

2. Conectar a la base de datos desde HeidiSQL u otro cliente SQL.

En el cliente HeidiSQL, creamos una nueva conexión y pegamos el endpoint de nuestra BBDD creada en AWS en el recuadro, usuario y contraseña (recomiendo apuntarte las credenciales)



3. Ejecutar el script SQL de creación y carga de la base de datos.

Al abrir la conexión, deberemos cargar nuestro archivo (.sql) o crear uno.

Os dejo la ruta a mi GitHub donde se encuentran todos los archivos necesarios, se encuentran en la carpeta ‘trabajoAWSEBS’.

<https://github.com/pablooibarren/ContenedoresDocker.git>

```

1 -- 
2 -- Host: apartamentos4v-mysql.cxjsw65ztsw.us-east-1.rds.amazonaws.com
3 -- Version del servidor: 8.0.39 - Source distribution
4 -- SO del servidor: Linux
5 -- HeidiSQL Version: 12.6.0.6765
6 --
7 
8 /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
9 /*!40101 SET NAMES utf8 */;
10 /*!40103 SET NAMES utf8mb4 */;
11 /*!40103 SET @OLD_TIME_ZONE=@TIME_ZONE */;
12 /*!40103 SET TIME_ZONE='+00:00' */;
13 /*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHE
14 /*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
15 /*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
16 
17 
18 -- Volcando estructura de base de datos para apartamento4V
19 CREATE DATABASE IF NOT EXISTS `apartamento4V` /*!40100 DEFAULT CHARACTER SET
20 USE `apartamento4V`;

```

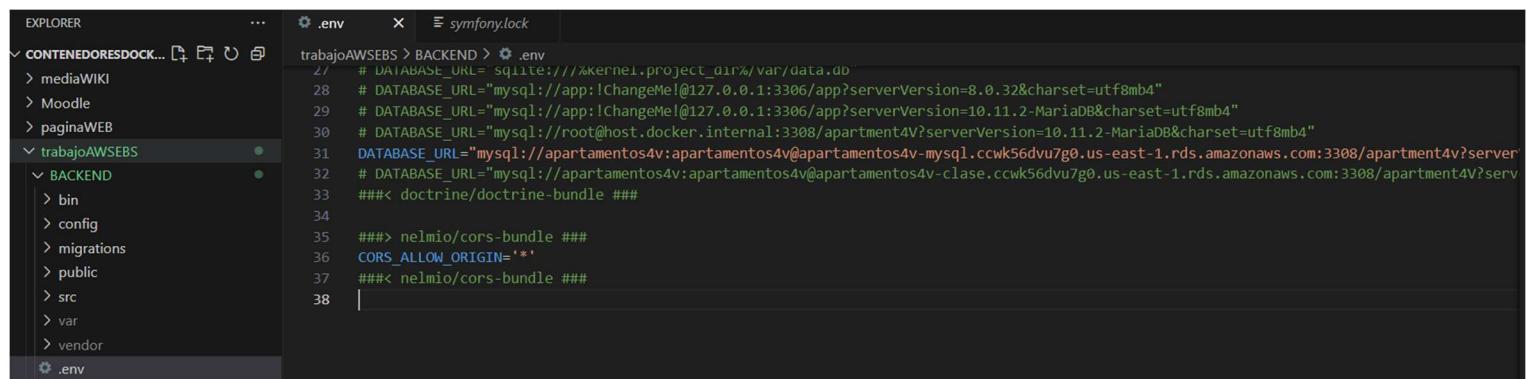
Ejecutamos la primera línea que es la que crea la BBDD y después seleccionamos y ejecutamos todo lo demás.

5. Despliegue del Backend en AWS Elastic Beanstalk

1. Configurar las variables de entorno en AWS Elastic Beanstalk:

Para ello, en VScode abrimos la carpeta del código del Backend y abrimos el archivo (.env) para cambiar la ruta DATABASE_URL.

Vamos a nuestro RDS creado anteriormente y copiamos el código de enlace de nuestra base de datos para copiarlo posteriormente en la siguiente línea: **apartamentos4v** es usuario, después de los dos puntos **apartamentos4v** es la contraseña y después de '@' pegamos el código de enlace de nuestra BBDD con el puerto correspondiente al final como se muestra en la imagen.



```

EXPLORER ... .env symfony.lock
CONTENEDORESDOCK... 🔍 ⚡ 🛡
> mediaWIKI
> Moodle
> paginaWEB
trabajoAWSEBS 🔍
  BACKEND 🔍
    > bin
    > config
    > migrations
    > public
    > src
    > var
    > vendor
    .env

trabajoAWSEBS > BACKEND > .env
27 # DATABASE_URL=sqlite:///__kernel.project_dir/var/data.db
28 # DATABASE_URL="mysql://app:!ChangeMe!@127.0.0.1:3306/app?serverVersion=8.0.32&charset=utf8mb4"
29 # DATABASE_URL="mysql://app:!ChangeMe!@127.0.0.1:3306/app?serverVersion=10.11.2-MariaDB&charset=utf8mb4"
30 # DATABASE_URL="mysql://root@host.docker.internal:3308/apartamento4v?serverVersion=10.11.2-MariaDB&charset=utf8mb4"
31 DATABASE_URL="mysql://apartamentos4v:apartamentos4v@mysql.ccwk56dvu7g0.us-east-1.rds.amazonaws.com:3308/apartamento4v?serverVersion=8.0.32& charset=utf8mb4"
32 # DATABASE_URL="mysql://apartamentos4v:apartamentos4v@apartamentos4v-clase.ccwk56dvu7g0.us-east-1.rds.amazonaws.com:3308/apartamento4v?serverVersion=10.11.2-MariaDB& charset=utf8mb4"
33 ##### doctrine/doctrine-bundle #####
34
35 #####> nelmio/cors-bundle #####
36 CORS_ALLOW_ORIGIN='*'
37 #####< nelmio/cors-bundle #####
38

```

Después, tendremos que crear un archivo llamado ‘Dockerfile’. Copiarlo de GitHub. Debería tener una forma parecida a la siguiente:

Aquí tienes una explicación de lo que hace el Dockerfile línea por línea:

```

trabajoAWSEBS > BACKEND > Dockerfile > ...
1 # Imagen base de PHP con Apache
2 FROM php:8.2-apache
3
4 # Instala las extensiones necesarias para Symfony
5 RUN apt-get update && apt-get install -y \
6     git \
7     unzip \
8     libicu-dev \
9     libzip-dev \
10    libonig-dev \
11    mariadb-client \
12    && docker-php-ext-install intl pdo pdo_mysql zip opcache
13
14 # Instalar Composer
15 COPY --from=composer:latest /usr/bin/composer /usr/bin/composer
16
17 # Configura el directorio de trabajo
18 WORKDIR /var/www/html
19
20 # Copia los archivos del proyecto al contenedor
21 COPY .
22
23 # Configura permisos para Symfony
24 RUN chown -R www-data:www-data var/cache var/log
25
26 # Configuración del VirtualHost de Apache
27 RUN echo "<VirtualHost *:80>\n\
28     DocumentRoot /var/www/html/public\n\
29     <Directory /var/www/html/public>\n\
30         AllowOverride All\n\
31         Require all granted\n\
32         # Reescritura para Symfony (equivalente a .htaccess)\n\
33         <IfModule mod_rewrite.c>\n\
34             RewriteEngine On\n\
35             RewriteCond %{REQUEST_FILENAME} !-f\n\
36             RewriteCond %{REQUEST_FILENAME} !-d\n\
37             RewriteRule ^(.*)$ index.php [QSA,L]\n\
38         </IfModule>\n\
39     </Directory>\n\
40 </VirtualHost>" > /etc/apache2/sites-available/000-default.conf
41
42 # Habilita el módulo rewrite de Apache para Symfony
43 RUN a2enmod rewrite
44
45 # Expone el puerto 80
46 EXPOSE 80
47
48 # Comando por defecto para iniciar Apache
49 CMD ["apache2-foreground"]

```

Uso como imagen base php:8.2-apache, que ya incluye PHP 8.2 y Apache preconfigurado.

Actualiza el índice de paquetes (apt-get update).

Instala herramientas necesarias (git, unzip).

Instala bibliotecas para extensiones (libicu-dev, libzip-dev, libonig-dev).

Instala mariadb-client para conectarse a bases de datos MariaDB/MySQL.

Usa docker-php-ext-install para habilitar extensiones de PHP requeridas por Symfony (intl, pdo, pdo_mysql, zip, opcache).

Copia el ejecutable de composer desde la imagen oficial a /usr/bin/composer, permitiendo su uso en el contenedor.

Define /var/www/html como el directorio de trabajo dentro del contenedor.

Copia todo el contenido del proyecto local al contenedor.

Cambia el propietario de var/cache y var/log a www-data (usuario de Apache en Linux), asegurando que Symfony pueda escribir logs y caché.

Sobrescribe la configuración del VirtualHost predeterminado (000-default.conf).

Define que Apache escuche en el puerto 80.

Especifica que el directorio raíz del servidor web es /var/www/html/public.

Permite la sobreescritura (AllowOverride All) y acceso a todos (Require all granted).

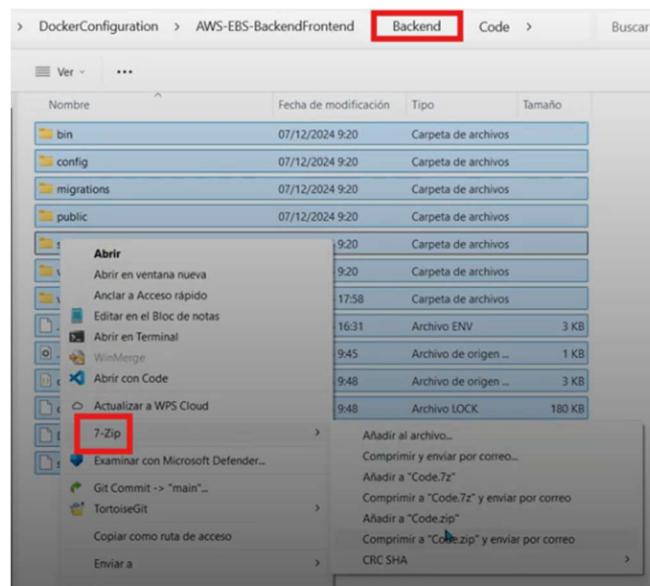
Configura la reescritura de URLs para Symfony, redirigiendo todas las solicitudes a index.php si no son archivos o directorios existentes.

Habilita el módulo mod_rewrite, necesario para la reescritura de URLs en Symfony.

Expone el puerto 80 para permitir conexiones HTTP.

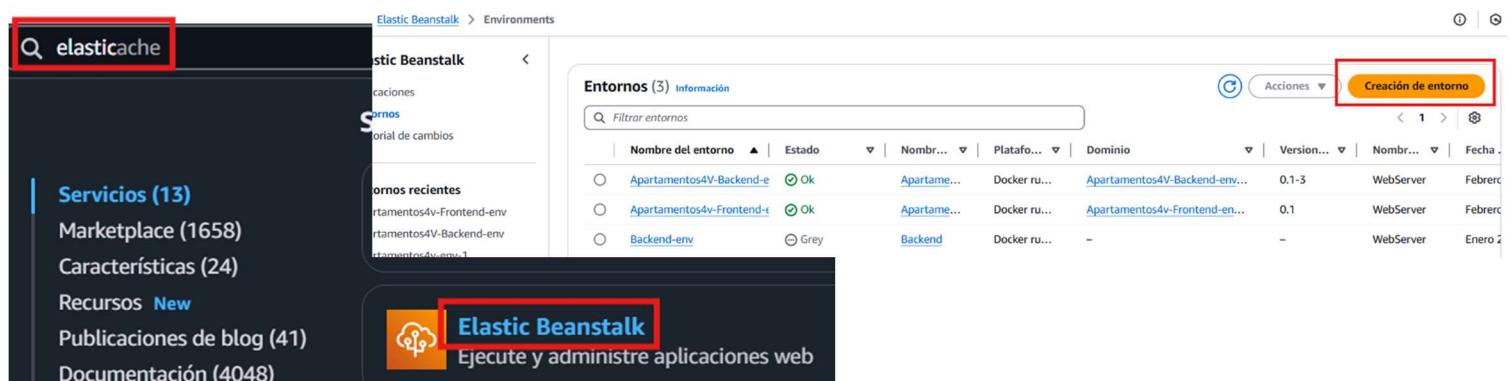
Ejecuta Apache en primer plano (apache2-foreground), manteniendo el contenedor en ejecución.

Una vez tengamos los archivos bien configurados y preparados deberemos crear una carpeta zip con todo el contenido del Backend



2. Crear un nuevo entorno en Elastic Beanstalk:

En el buscador de nuestro laboratorio buscamos Elastic Beanstalk. Una vez dentro, seleccionamos ‘Creación de entorno’.



Elastic Beanstalk > Environments

Entornos (3) Información

Acciones ▾ Creación de entorno

Nombre del entorno	Estado	Plataforma	Dominio	Version...	Nombr...	Fecha...
Apartamentos4V-Backend-e	Ok	Apartame...	Docker ru...	Apartamentos4V-Backend-env...	0.1-3	WebServer
Apartamentos4V-Frontend-e	Ok	Apartame...	Docker ru...	Apartamentos4V-Frontend-en...	0.1	WebServer
Backend-env	Grey	Backend	Docker ru...	-	-	WebServer

Lo más importante de la configuración del Elastic Beanstalk es lo siguiente:

Plataforma Información

Tipo de plataforma

Plataforma administrada
Plataformas publicadas y mantenidas por Amazon Elastic Beanstalk. [Más información](#)

Plataforma personalizada
Plataformas creadas y de su propiedad. Esta opción no está disponible si no tiene plataformas.

Plataforma
Docker

Ramificación de la plataforma
Docker running on 64bit Amazon Linux 2023

Versión de la plataforma
4.4.3 (Recommended)

Código de aplicación Información

Aplicación de ejemplo

Versión existente
Versiones de la aplicación que ha cargado.

Cargar el código
Cargue un paquete de código fuente desde su equipo o copie uno desde Amazon S3.

Etiqueta de versión
Nombre único para esta versión del código de la aplicación.
Etiqueta de versión

Origen del código fuente. Tamaño máximo de 500 MB

Archivo local

URL pública de S3

Acceso al servicio

Los roles de IAM, asumidos por Elastic Beanstalk como rol de servicio, y los perfiles de instancia de EC2 permiten administrar su entorno. Tanto el rol de IAM como el perfil de instancia deben estar asociados a políticas admisibles permisos necesarios. [Más información](#)

Rol de servicio

- Crear y utilizar un nuevo rol de servicio
- Usar un rol de servicio existente

Roles de servicio existentes

Elija un rol de IAM existente para que Elastic Beanstalk asuma como rol de servicio. El rol de IAM existente debe tener las políticas admisibles necesarias.

LabRole



Par de claves de EC2

Seleccione un par de claves de EC2 para iniciar sesión de forma segura en sus instancias de EC2. [Más información](#)

vockey



Perfil de instancia de EC2

Elija un perfil de instancia de IAM con políticas administradas que permitan a las instancias de EC2 realizar las operaciones necesarias.

LabInstanceProfile



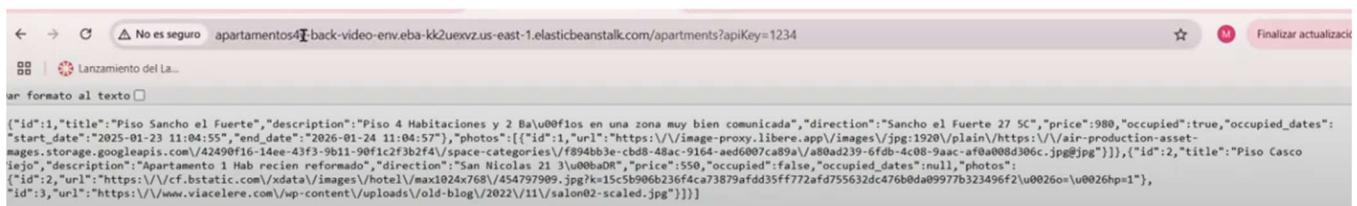
[Ver los detalles de los permisos](#)

Cargamos el código desde un archivo local seleccionando el zip creado anteriormente.

Una vez creado el EBS deberemos probar que el Backend funciona correctamente.

3. Probar el backend accediendo a:

<http://ipebs/apartments?apiKey=1234>



Debe devolver datos de la base de datos si la conexión está correctamente configurada.

6. Despliegue del Frontend en AWS Elastic Beanstalk

Los pasos son similares al EBS del Frontend

1. Crear un nuevo entorno en Elastic Beanstalk:

- Tipo de aplicación: **Docker**

- Subir el archivo ZIP con el Dockerfile y el código del frontend (AngularJS).

2. Configurar el JavaScript del frontend para apuntar a la URL del backend en AWS.

Buscamos las líneas donde estén las URLs y las cambiamos por la ruta correcta. Deberemos sustituirlo por la URL de nuestro Backend, con la que acabamos de probar la conexión.

```
JS main-ZKT6JCA3.js ×

Frontend > Code > JS main-ZKT6JCA3.js > [e] vd > [f] <function> > [e] e > [f] constructor
  _ngcontent-%COMP%_j1T0Hl-S1ze...
  [_ngcontent-%COMP%]{display:inline-block;color:#fff;text-decoration:none;transition:background-color .3s,{background-color:#005b63}.card-price[_ngcontent-%COMP%]{position:absolute;right:10px}.card-price[_ngcontent-%COMP%] span[_ngcontent-%COMP%]{color:red;font-weight:700}.card-content[_ngcontent-%COMP%] hr[_ngcontent-%COMP%]{margin:0 0 10px;border:none;height:2px;background-color:#ccc;font-weight:700}.card-occupied[_ngcontent-%COMP%]{font-weight:700;font-size:1rem}.pics[_ngcontent-%COMP%]{width:100%;height:300px;margin-bottom:10px;object-fit:cover});let t=e;return t}();var vd=()=>{
  let e=class e{constructor(n){this.apiUrl="http://localhost:8080/apartments?apiKey=1234"}getApartments(){
    http.get(this.apiUrl)};e.prototype.getApartments=async(p...
  
```

Después, deberemos crear un archivo ‘Dockerfile’ al igual que en el Backend. (Copiar de GitHub)

```
trabajoAWSEBS > FRONTEND > Dockerfile > FROM
1  FROM ubuntu:20.04
2
3  # Update repositories index
4  RUN apt update --fix-missing && apt upgrade -y
5
6  # Install util tools
7  RUN ln -snf /usr/share/zoneinfo/UTC /etc/localtime && echo "UTC" > /etc/timezone
8  RUN apt install -y apt-utils git rsync nano vim unzip curl wget software-properties-common mysql-client
9
10 # Install Apache
11 RUN apt-get install -y apache2
12
13 # Config Apache
14 ENV APACHE_RUN_USER=www-data
15 ENV APACHE_RUN_GROUP=www-data
16 ENV APACHE_LOG_DIR=/var/log/apache2
17 RUN chown www-data:www-data -R /var/www
18 RUN chmod -R 755 /var/www
19
20 RUN a2enmod rewrite headers
21 RUN service apache2 start
22 WORKDIR /var/www/html
23 COPY Code/. /var/www/html
24
25 EXPOSE 80
26
27 # Ejecutar Apache2 en primer plano
28 CMD ["apache2ctl", "-D", "FOREGROUND"]
```

3. Implementar y esperar a que el entorno esté disponible.
4. Probar que el frontend funciona correctamente y consume el backend desplegado.

7. Prueba Final de la Infraestructura

1. Acceder a la URL del frontend en AWS Elastic Beanstalk.
2. Navegar por la aplicación y verificar que los datos cargan correctamente desde el backend.
3. Validar que las conexiones entre los componentes están funcionando.
4. Realizar pruebas adicionales para asegurar estabilidad y seguridad.

8. Conclusión

Siguiendo estos pasos, se habrá implementado con éxito una infraestructura web en AWS con RDS para la base de datos, un backend en Symfony y un frontend en AngularJS desplegados en Elastic Beanstalk con Docker.

9. Troubleshooting

A continuación, se describen algunos problemas comunes que pueden surgir durante la implantación de la infraestructura y cómo solucionarlos.

No puedo conectarme a la base de datos RDS desde HeidiSQL

Posibles causas y soluciones:

- **Error de conexión (Host 'XXX.XXX.XXX.XXX' is not allowed)**
Verifica que en Amazon RDS el **Security Group** permite conexiones en el puerto 3306 desde tu IP local.
Asegúrate de que el usuario de la base de datos tiene permisos suficientes (GRANT ALL PRIVILEGES).
- **Fallo en el hostname**
Asegúrate de que estás usando el **endpoint correcto** de RDS y no "localhost".
Puedes encontrar el endpoint en la consola de AWS RDS.

- **Puerto incorrecto**

Si en XAMPP tienes MySQL en 3308, asegúrate de que HeidiSQL está intentando conectarse al **puerto 3306 de RDS**, no al de XAMPP.

El backend en Elastic Beanstalk no puede conectarse a la base de datos

Posibles causas y soluciones:

- **⚠ Error "Access denied for user"**

◆ Revisa que las credenciales en el archivo .env del backend están bien configuradas

El frontend no se comunica con el backend

Posibles causas y soluciones:

- **El frontend tiene la URL incorrecta del backend**

En el código del frontend (AngularJS), revisa que la URL de las peticiones apunta al dominio o IP del backend en EBS

Prueba llamar directamente al backend desde el navegador (<http://backend-ipebs.apartments.com/api/apartments?apiKey=1234>) para confirmar que responde correctamente.

Elastic Beanstalk no despliega correctamente el backend/frontend

Posibles causas y soluciones:

- **Error de permisos en el despliegue**

Asegúrate de que el Dockerfile del backend y frontend tiene los permisos correctos (chmod +x si es necesario).

Si hay errores, revisa los logs de EBS en la consola de AWS.

- **Error "Application failed to start"**

Verifica que en Dockerrun.aws.json están bien definidas las imágenes de Docker.

Prueba ejecutar el contenedor en local (docker-compose up) para asegurarte de que funciona antes de subirlo.