Ejercicio 7. PostgreSQL en AWS, transacciones y procedimientos almacenados

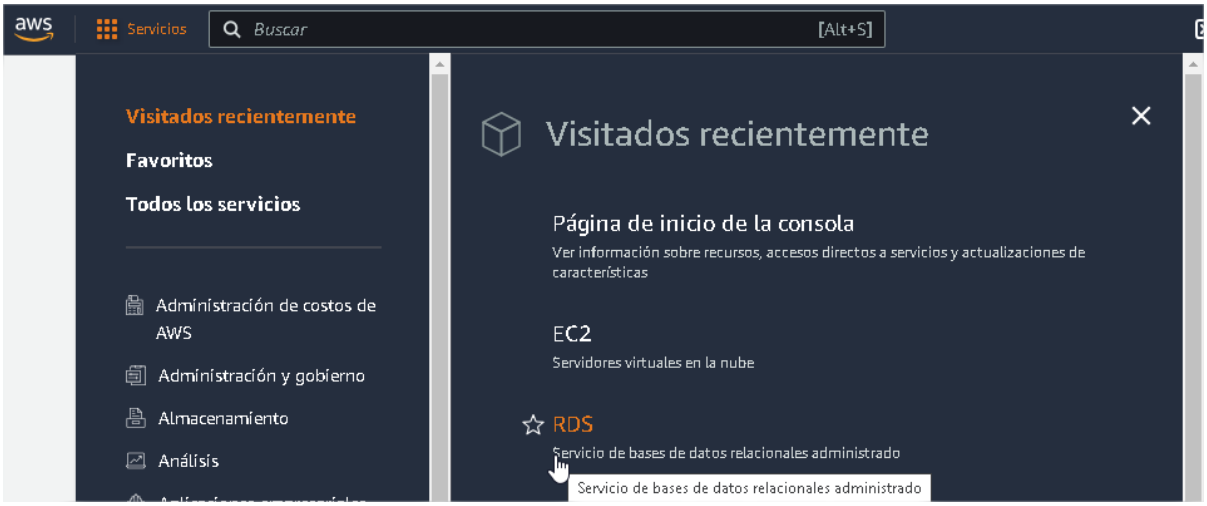
Se ha generado el código SQL para poder crear una base de datos PostgreSQL con los datos de la temporada 2006 de Formula 1. Esta base de datos será gestionada por el servicio Amazon Relational Database Service (Amazon RDS).

El objetivo de este ejercicio es comprobar el funcionamiento de una operación transaccional donde se realiza la inserción de un nuevo piloto y la actualización o creación del equipo para el que pilota.

Además, consumiremos procedimientos almacenados ya realizados.

Paso 1: Configuración de la instancia de base de datos en Amazon RDS

Inicia el laboratorio AWS Academy Learner Lab y busca en la consola de AWS el servicio RDS y haz clic en él.



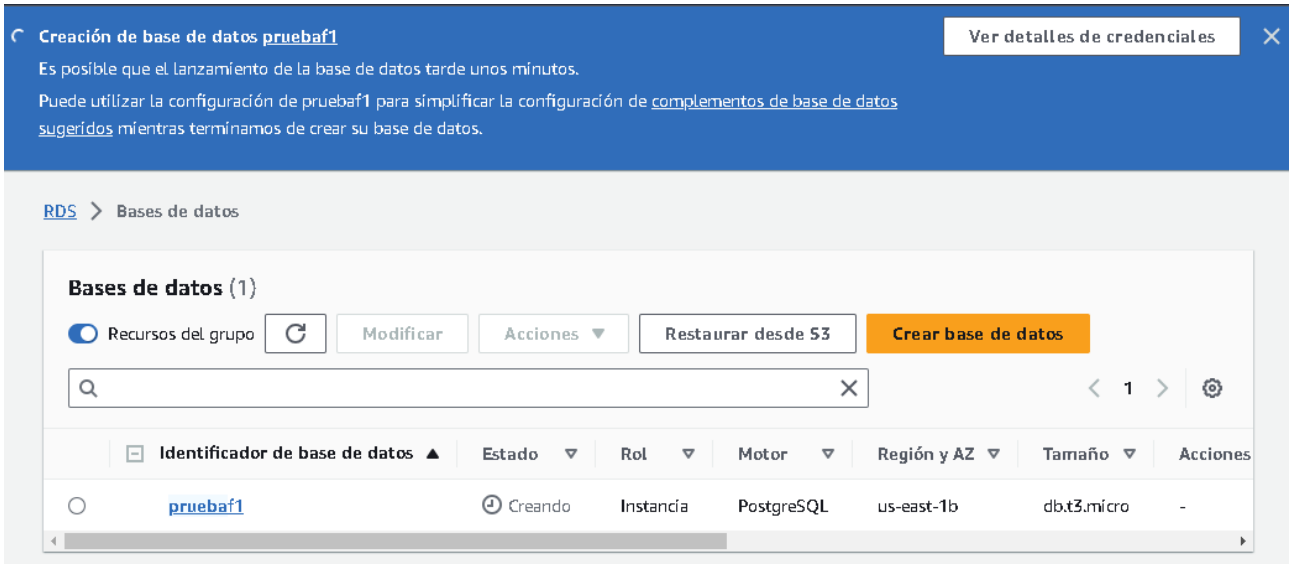
En el menú vertical izquierdo, haz clic en Bases de Datos, donde inicialmente te aparecerá un listado vacío de instancias de bases de datos. Haz clic en el botón naranja **Crear base de datos**, el cual abre una nueva página configuraremos las características de nuestra instancia.

Las opciones a marcar son:

- En **Elegir un método de creación de base de datos** selecciona **Creación estándar**
- En **Opciones de motor**, dentro de **Tipo de motor** selecciona **PostgreSQL**.
- En **Plantillas**, selecciona **Capa gratuita**.
- En **Configuración**:
    - En **Identificador de instancias de bases de datos**, escribe el **nombre que quieres que identifique** a tu instancia dentro del listado que tendrás en Amazon.
    - En el apartado **Configuración de credenciales**, pon como **Nombre de usuario maestro** el usuario maestro de la instancia, por defecto **postgres**. Añade una **contraseña maestra** de mínimo 8 caracteres para identificarte como el usuario maestro. Como estamos haciendo pruebas, utiliza una contraseña sencilla y **no relacionada con tus contraseñas personales**.

- En **Conectividad**:
    - En **Acceso Público**, marca **Sí**, para que la instancia y, por tanto, las bases de datos que contenga sean accesibles desde Internet y, por tanto, desde tu aplicación Java.
    - **En Grupo de seguridad de VPC** (firewall) déjalo como está, con el **grupo de seguridad default**. A posteriori comprobaremos que el tráfico entrante a PostgreSQL está permitido.
- En **Autenticación de bases de datos** debes marcar la opción de **Autenticación con contraseña**.
- En **Configuración adicional**, dentro de las **Opciones de base de datos**, añade un **Nombre de base de datos inicial** como, por ejemplo, **f12006**

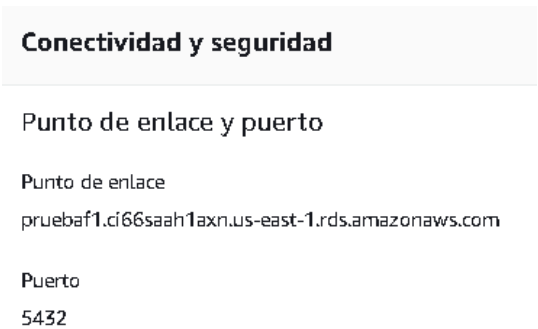
Ya puedes bajar hasta debajo de la página y darle a **Crear base de datos**. La instancia de base de datos estará en proceso de creación.



Una vez recibáis el mensaje de que la instancia de base de datos se ha creado correctamente, hacemos clic en su identificador para conocer algunos detalles de su configuración.

En la nueva ventana que se abre, tenemos información en diversos apartados de los cuales nos interesan sobre todo 2: **Conectividad y seguridad**, y **Configuración**.

En **Conectividad y seguridad** tenemos, por un lado, la información del **punto de enlace** de la instancia, la cual utilizaremos junto al **puerto** de escucha para formar la URL con la que conectarnos a ella.



A la derecha, en **Seguridad**, tenemos un enlace a la configuración del grupo de seguridad default, que es el que hemos seleccionado para la instancia de base de datos. Recordemos que el trafico entrante a las instancias de AWS queda denegado hasta que no se marque explícitamente lo contrario. Si indagamos en las reglas de entrada de ese grupo encontramos lo siguiente:

Es decir, se permite el tráfico entrante desde cualquier origen por el puerto 5432, el puerto estándar de PostgreSQL, y también por el puerto 3306, estándar de otros sistemas gestores de bases de datos como MySQL o MariaDB. En cierto modo, se podría considerar que mantener este puerto 3306 abierto es una vulnerabilidad ya que no se le va a dar uso, pero vamos a dejarlo como está por sencillez.

En la pestaña **Configuración** tenemos información sobre la **versión del motor** (útil para buscar clientes compatibles), del **nombre de base de datos** creado y del **usuario maestro**.

Paso 2: Importar fichero SQL en base de datos

Aunque tenemos creada la instancia de la base de datos con una base de datos inicial, dicha base de datos está vacía, por lo que queremos importar un fichero SQL para poblarla.

La forma más sencilla desde Lliurex es utilizar el terminal y el cliente **psql** de PostgreSQL que tenéis instalado y que también se puede instalar en cualquier sistema operativo.

Ejecuta el siguiente comando reemplazando la información por la que se adecue a tu ejemplo (te recomiendo usar psql --help si tienes dudas):

psql -h punto_de_enlace -U usuario_maestro -d nombre_base_datos -f fichero.sql

En mi caso, el comando ejecutado y el resultado de su ejecución se puede ver a continuación:

```
ubuntu@ip-172-31-43-121:~$ psql -h pruebaf1.ci66saah1axn.us-east-1.rds.amazonaws.com -U postgres -d f12006 -f f12006-pg.sql
Password for user postgres:
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
INSERT 0 18
INSERT 0 11
INSERT 0 27
INSERT 0 18
INSERT 0 396
CREATE FUNCTION
CREATE FUNCTION
```

Paso 3: Transacciones

Para comprobar cómo funcionan las transacciones vamos a realizar la inserción de dos nuevos pilotos y un nuevo equipo.

Para el primer piloto a crear utiliza los datos de Carlos Sainz de la práctica anterior. El equipo al que pertenece es el equipo de nueva creación Seat F1.

Aunque conceptualmente parezca sencillo tener un piloto con un equipo asociado y que al insertar el piloto también se inserte el equipo, en la práctica no es tan sencillo porque requerimos de insertar primero el equipo y conocer el ID con el que se ha insertado para utilizarlo en la inserción del piloto como clave ajena a la tabla de equipos.

Para la inserción (o actualización) de un equipo, nos puede ayudar el concepto **UPSERT**, que hace referencia a una operación que hace un UPDATE o un INSERT, dependiendo si hay un registro con el ID especificado en la base de datos. En Java podemos implementar un UPSERT de la siguiente forma:

```
String sql = "INSERT INTO tabla (columna 1, columna2) VALUES (valor1, valor2)  "  +
             "ON CONFLICT (id) DO NOTHING RETURNING id;";
```

Antes de finalizar la transacción, crea el segundo piloto del equipo, al que podemos llamar Manuel Alomá, con código de piloto ALO. Ejecuta el código y debería dar un error ya que el código ALO pertenece a Fernando Alonso, teniendo la estructura de la tabla drivers una restricción donde el código de piloto debe ser único.

Mira el contenido de la tabla de pilotos y equipos para asegurarte que ni el equipo Seat ni el piloto Caros Sainz (y muchos menos Manuel Alomá) se han insertado.

Modifica ahora el código a Manuel Aloma a ALM y prueba la ejecución de la transacción mostrando la tabla de pilotos y equipos.

Paso 4: Uso de procedimientos almacenados

PostgreSQL admite procedimientos almacenados y funciones para hacer que las consultas de SQL sean reutilizables.

Nuestra base de datos contiene dos funciones a las que puedes llamar y utilizar. Dichas funciones son:

- ***get_results_by_driver(cod)***, que recibe un código de piloto y devuelve sus resultados para cada carrera de la temporada.
- ***get_drivers_standings()***, que no recibe ningún parámetro y devuelve la clasificación final del mundial.