

Configuraciones básicas Firebase/Springboot/CleverCloud/Heroku

En base a errores comunes surgidos a la hora de realizar el release o deploy de nuestro proyecto Full Stack se formulan una serie de modificaciones, las cuales detallo a continuación:

1- Error: Incompatibilidad de versiones de Java y/o Maven

[ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.8.1:compile (default-compile) on project NOMBRE_DE_TU_PROYECTO: Fatal error compiling: invalid target release: 8 -> [Help 1] (también puede darse en las versiones 11 y 17)

Esto se debe a la incompatibilidad de las versiones de Java y/o Maven que tienes en tu PC Local con las que intenta instalar heroku remotamente.

Para solucionarlo debes indicarle a heroku las versiones que usas en tu proyecto, creando un archivo de propiedades de sistema (**system.properties**) especificando las versiones de Java y/o Maven que usas localmente, y luego subirlos para heroku los tenga en cuenta. Básicamente debes seguir los siguientes pasos:

*******(Aclaración: las versiones son de ejemplo, por favor coloca las versiones que tienes en tu pc)*******

1- Crear el archivo **system.properties** en la raíz de tu proyecto con el siguiente contenido:

```
java.runtime.version=11
maven.version=3.6.2
```

2- Agregar a git el archivo, ejecutando en la raíz de tu proyecto:

```
git add system.properties
```

O

```
git add .
```

3- Hacer commit de los cambios:

```
git commit -m "se agrega archivo de configuraciones de sistema"
```

4- Finalmente realizar:

```
git push heroku master
```

referencia: <https://devcenter.heroku.com/articles/java-support#specifying-a-java-version>

2- Error: Error timeout heroku/clevercloud:

- Abrir la url: https://tools.heroku.support/limits/boot_timeout
- En la ventana que se abre, seleccionar nuestra app, desplazar el control para aumentar el tiempo de espera y presionar el botón **Change Boot Timeout**:

The screenshot shows the 'Change Boot Timeout' page on Heroku. At the top, the title 'Change Boot Timeout' is displayed in purple. Below the title, there is explanatory text about the default boot timeout of 60 seconds and the option to increase it. A dropdown menu labeled 'Which application?' shows 'backend-portfolio-springboot'. Below this, a slider for 'Boot Timeout' is set to 120 seconds. A 'Reset to Default' button is next to the slider. At the bottom, a large purple button says 'Change Boot Timeout'.

- Por último, seguir las instrucciones:

```
Success!

The boot timeout for backend-portfolio-springboot has been updated to 120 seconds. The new boot timeout will take effect after your next deployment. You can redeploy your code without any changes by running:

$ git commit --allow-empty -m"Empty commit"
$ git push heroku master
```

3- Error: CleverCloud Supera el límite de conexiones

A partir de Spring Boot 2 se agrega por defecto Hikari como pool de conexiones. Por defecto se crean 10 conexiones a la base de datos. El problema es que Clever Cloud en su versión gratuita solo permite 5 conexiones.

Plans

Note for Shared databases

As Shared databases (DEV) are shared between multiple applications and delays could appear in case of an high demand.

If this delays create problems in your application or are problematic, we recommend you to use a dedicated database (XS plans and above).

Plan	vCPUs	RAM	Disk size	Connection limit	Logs	Metrics	Price (day)	Price (30 days)
DEV	Shared	Shared	10 MiB	5	No	No	€0.00	€0.00
XXS Small Space	1	512 MiB	512 MiB	15	Yes	Yes	€0.17	€5.00
XXS Medium Space	1	512 MiB	1 GiB	15	Yes	Yes	€0.20	€5.90
XXS Big Space	1	512 MiB	2 GiB	15	Yes	Yes	€0.23	€6.80
XS Tiny Space	1	1 GiB	2 GiB	75	Yes	Yes	€0.43	€13.00
XS Small Space	1	1 GiB	5 GiB	75	Yes	Yes	€0.73	€22.00

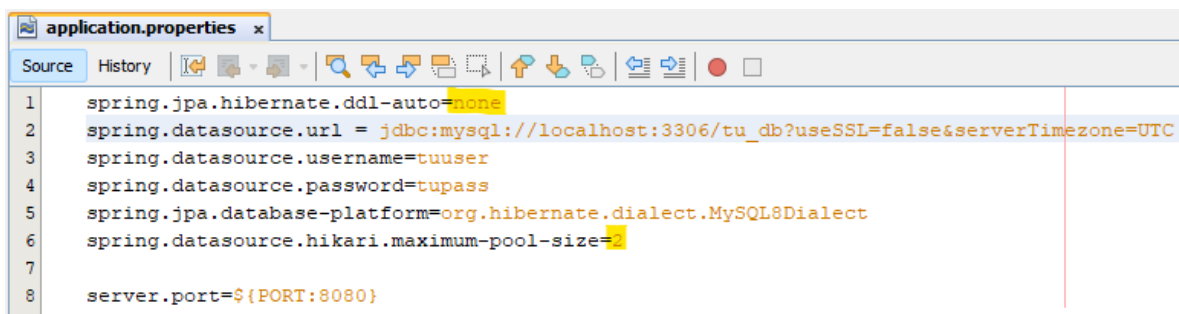
Es por esta razón que una vez que se inicia el backend ya no es posible conectarse a la base desde otra herramienta como MySQL Workbench por ejemplo.

Para solucionar el problema, se puede configurar el pool de conexiones para que el backend no consuma todas las conexiones disponibles.

Para esto, debes agregar al archivo *application.properties*:

spring.datasource.hikari.maximum-pool-size=2

lo cual configura el pool de conexiones para que cree 2 conexiones como máximo a la base de datos, que para el caso del portfolio son suficientes. De esta forma se puede conectar a la base de datos desde una herramienta externa ya que hay conexiones disponibles, teniendo el servicio SpringBoot en ejecución.



```
1 spring.jpa.hibernate.ddl-auto=none
2 spring.datasource.url=jdbc:mysql://localhost:3306/tu_db?useSSL=false&serverTimezone=UTC
3 spring.datasource.username=tuuser
4 spring.datasource.password=tupass
5 spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
6 spring.datasource.hikari.maximum-pool-size=2
7
8 server.port=${PORT:8080}
```

Solución alternativa (no tan eficiente):

Antes de ejecutar tu app, desde clevercloud prueba cerrar todas las conexiones (como adjunto en la imagen).

MySQL CLI

```
mysql -h byltxhqxmr70i2lqsc7-mysql.services.clever-cloud.com -P 3306 -u  
uvb2oi6lmxvf2s8g -p byltxhqxmr70i2lqsc7
```

Reset Database

Reset database

Click this button to reset your database. Every table will be deleted. Your backups will not be deleted.

Kill all connections

Kill all connections to the database

Could not get the number of connections to your database.

Click this button to terminate all current connections on your database. If you have an application running, this might make your application crash.

4- Solución Error: CORS deshabilitados

1- Crear el archivo **WebConfig.java** en la raíz de tu proyecto con el siguiente contenido:

```
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
```

```
/**
 *
 * @author Ema
 */
@EnableWebMvc
@Configuration
public class WebConfig implements WebMvcConfigurer {
    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**");
    }
}
```

2- Agregar a git el archivo, ejecutando en la raíz de tu proyecto:

```
git add WebConfig.java
```

O

```
git add .
```

3- Hacer commit de los cambios:

```
git commit -m "se agrega archivo que habilita CORS"
```

4- Finalmente realizar:

```
git push heroku master
```

PD: Ejemplo de cómo quedaría

