## **Deploy Backend Spring Boot en Fly.io**

Al desarrollar una aplicación, la progresión natural es ponerla en línea y ponerla a disposición de los usuarios finales. Básicamente es habilitar la aplicación para su uso, ya sea un ambiente de desarrollo, para realizar pruebas o ponerla en producción. Para que esta tarea sea posible y más fácil, existen numerosas plataformas en la nube disponibles para alojar su aplicación. Fly.io es una de ellas, la cual nos permite implementar cualquier tipo de aplicación backend o frontend.

## **¿Qué es Fly.io?**

Es un servicio de plataforma como servicio o también llamado PaaS.

Básicamente es un conjunto de servicios basados en la nube que permite a los desarrolladores y usuarios empresariales crear aplicaciones a una velocidad que las soluciones en las instalaciones propias no pueden alcanzar.

Por supuesto que, al tratarse de un servicio basado en la nube, no hay necesidad de preocuparse por la configuración y el mantenimiento de servidores, parches, actualizaciones y autenticaciones, entre muchas otras tareas.

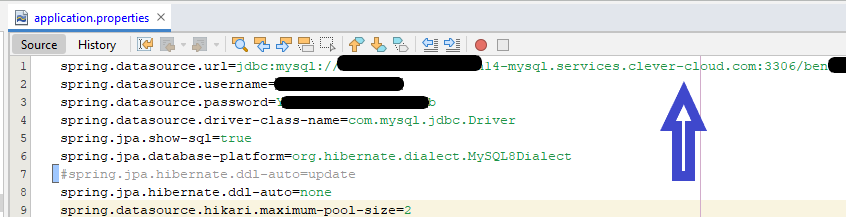
# Desplegar aplicación Springboot en Fly.io usando Dockerfile

Fly.io admite implementar aplicaciones de múltiples formas, pero en este caso lo haremos a través del uso de dockerfile. Eso significa que simplemente implementaremos una imagen de Docker de forma totalmente gratuita.

Para profundizar sobre la generación de imágenes docker de aplicaciones locales de springboot puede seguir la siguiente guía: [**https://spring.io/guides/gs/spring-boot-docker/**](https://spring.io/guides/gs/spring-boot-docker/)

Pre requisitos:

* Conocimientos básicos de creación de imágenes de docker.
* Tener compilación funcionando de proyecto springboot en su computadora local**.**
* Tener configurada la conexión a la base de datos en la nube de clevercloud:



* Tener habilitado CORS en tus endpoint: En algunos casos puede aparecer el error “No hay un encabezado 'Access-Control-Allow-Origin' presente en el recurso solicitado” al intentar invocar la API.

Los errores de uso compartido de recursos entre orígenes (CORS) ocurren cuando un servidor no devuelve los encabezados HTTP que exige el estándar CORS. Para resolver un error CORS de una API REST de API Gateway o API HTTP, debe configurar de nuevo la API para que cumpla con el estándar CORS.

Existen varias alternativas para solucionarlo, lo que se propone es agregar una clase en tu backend para habilitar todos los orígenes que peticionan:

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;

/\*\*

\* Clase que habilita CORS

\* @author YOProgramo

\*/

@EnableWebMvc

@Configuration

public class WebConfig implements WebMvcConfigurer {

@Override

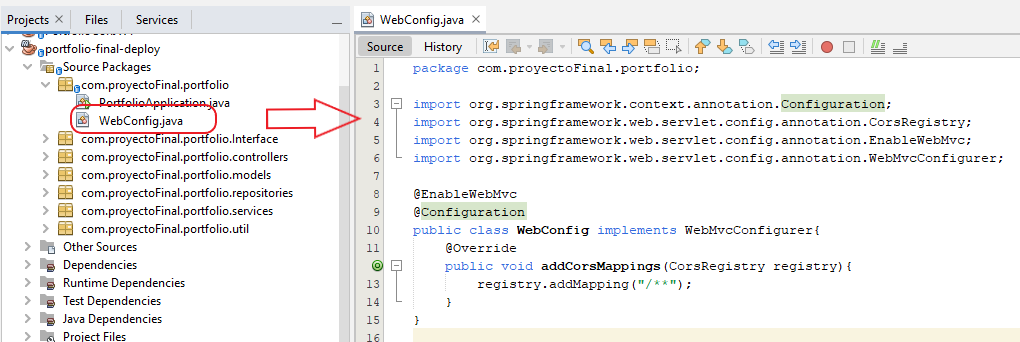
public void addCorsMappings(CorsRegistry registry) {

registry.addMapping(""/\*\*"");

}

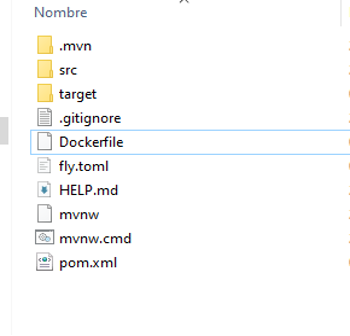
}

Su clase debería quedar tal cual muestra el ejemplo:



# A continuación, se detallan los pasos a seguir:

1. Crear archivo *Dockerfile*, en la carpeta de tu proyecto springboot:



2. Al archivo creado agregar la ***dependencia*** necesaria para levantar una versión de JDK igual a la que usas en tu proyecto local, en el ejemplo suponemos que tenemos la versión 11, por lo tanto elegimos una versión JDK11: ***amazoncorretto:11-alpine-jdk*** (Existen múltiples paquetes de JDK, puedes elegir el que prefieras en: <https://hub.docker.com/_/openjdk>) y la referencia del archivo **.jar** que contiene la compilación de nuestro proyecto springboot:

Código base:

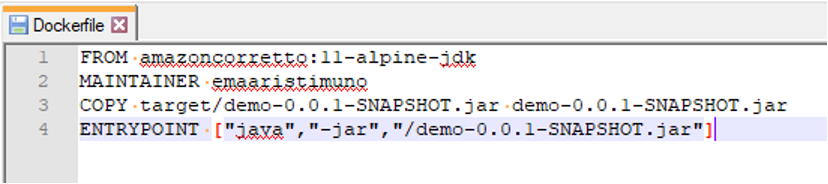
***FROM amazoncorretto:11-alpine-jdk***

***MAINTAINER emaaristimuno***

***COPY target/NAME-YOUR-FILE-BUILD-SPRINGBOOT.jar NAME-YOUR-FILE-BUILD-SPRINGBOOT.jar***

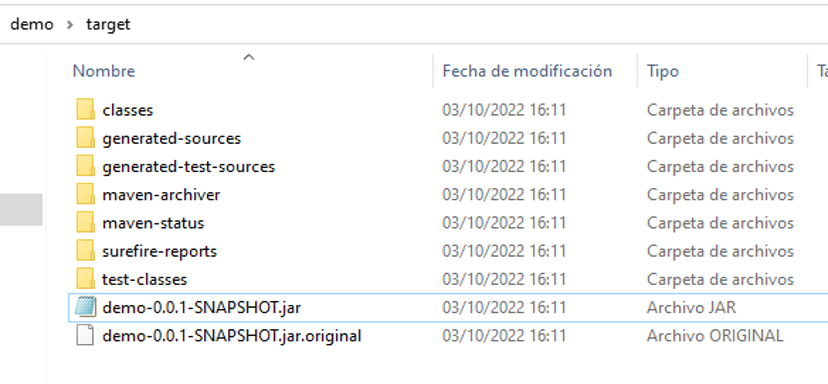
***ENTRYPOINT ["java","-jar","/NAME-YOUR-FILE-BUILD-SPRINGBOOT.jar"]***

*Ejemplo:*

****

Para el ejemplo, la compilación que se tendrá en cuenta para crear la imagen se encuentra dentro de la carpeta target:

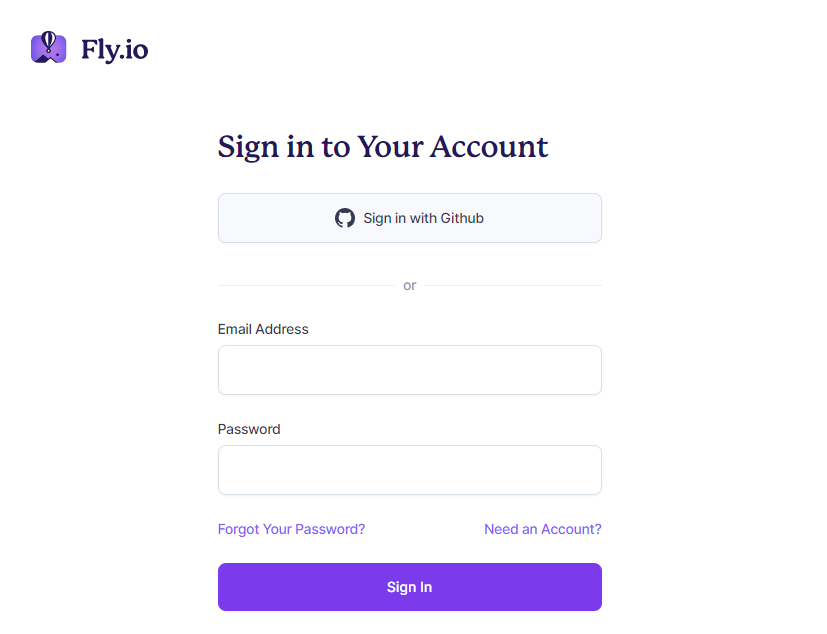
***target/demo-0.0.1-SNAPSHOT.jar***

******

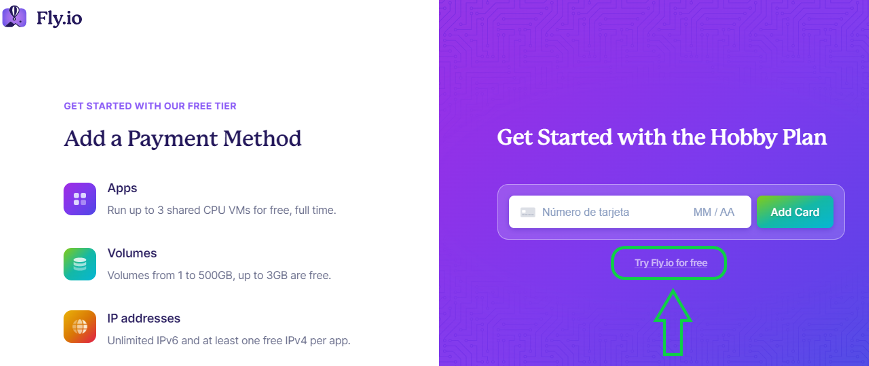
3. Instalar utilidad de comandos para trabajar con Fly: [flyctl](https://fly.io/docs/hands-on/install-flyctl/). Ejecutando en Powershell de Windows: ***iwr https://fly.io/install.ps1 -useb | iex***

******

4. Si esta es tu primera vez con Fly.io, tu próximo paso será [Registrarte](https://fly.io/docs/hands-on/sign-up/) (usa tu cuenta de github):



5. Te pedirá que ingreses un método de pago, pero no te preocupes puedes elegir ***Try Fly.io for Free*** como plan para comenzar:



6. Iniciar sesión en Fly: ***flyctl auth signup***



Esto lo llevará a la página de registro donde puede:

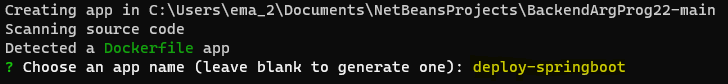
***Regístrese con correo electrónico***: Ingrese su nombre, correo electrónico y contraseña.

***Regístrese con GitHub***: si tiene una cuenta de GitHub, puede usarla para registrarse. Esté atento al correo electrónico de confirmación que se enviara, que le dará un enlace para establecer una contraseña; necesitará establecer una contraseña para que podamos verificar activamente que es usted para algunas operaciones de Fly.io.

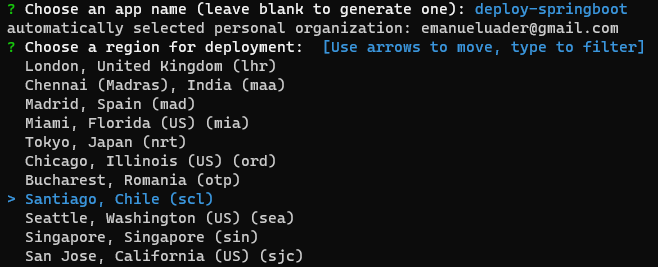
7. Cada aplicación Fly.io necesita unarchivo ***fly.toml*** para decirle al sistema cómo nos gustaría implementarlo. Ese archivo se puede generar automáticamente con el comando ***flyctl launch***, que hará algunas preguntas para configurar todo. Repasémoslo ahora:

7.1 Ejecutar los siguientes comandos dentro del directorio donde se encuentra el dockerfile:

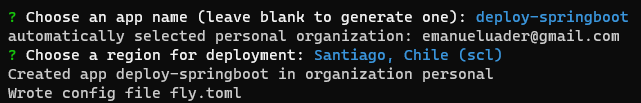
******

7.2 Definir un nombre para nuestra aplicación desplegada en Fly: 

7.3 Seleccionar la región que levantara nuestra app:

**

Nos creará un archivo ***fly.toml*** con las configuraciones necesarias:



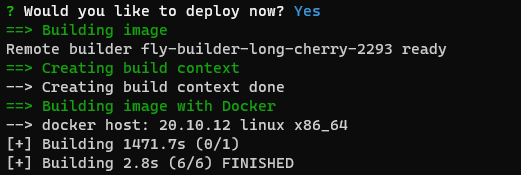
7.4 Nos pregunta si queremos instalar una Base de Datos Postgresql, seleccionar que ***No***:

****

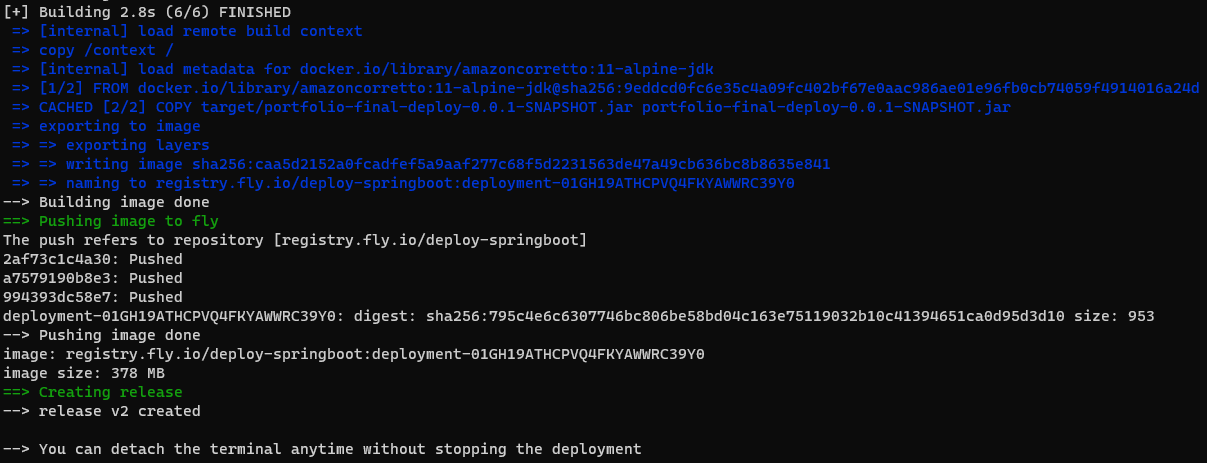
7.5 Generar el Deploy respondiendo ***Yes*** a la pregunta:



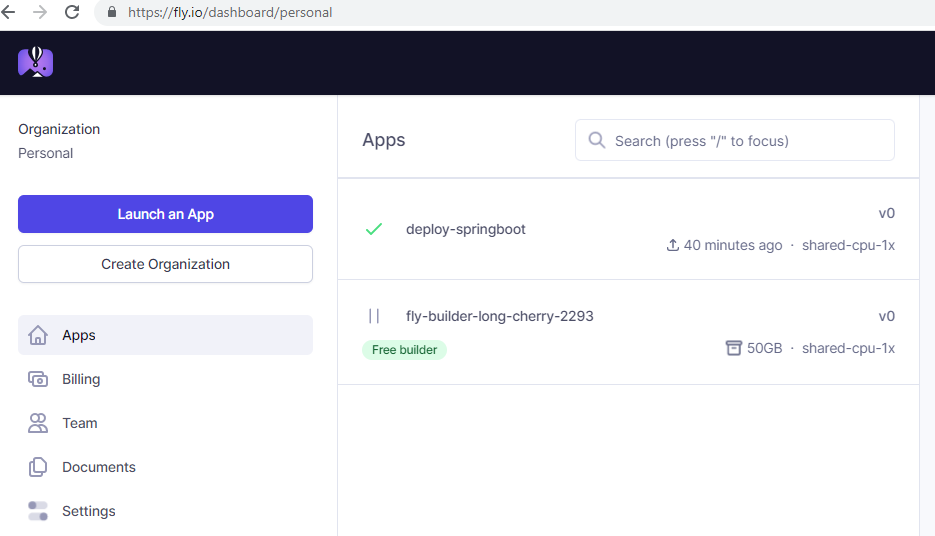
Esperar un momento hasta que se genere todos los procesos necesarios para subir la imagen de docker a fly (Según su conexión de internet puede demorar entre 5 y 30 minutos :/):



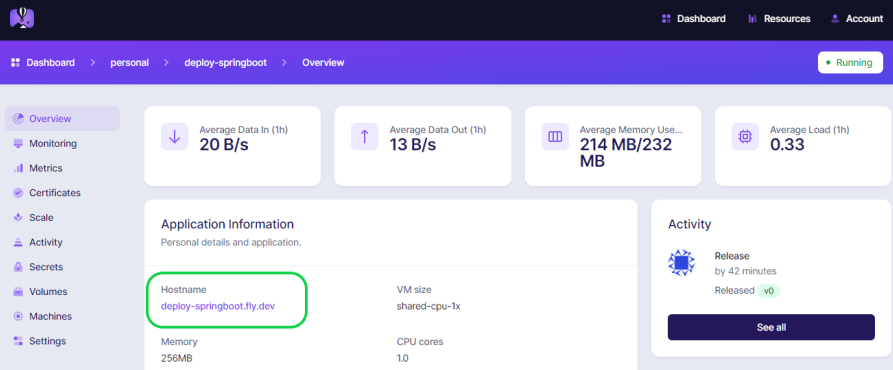
Si el despliegue se realizó de manera exitosa, la consola arrojará los siguientes mensajes:



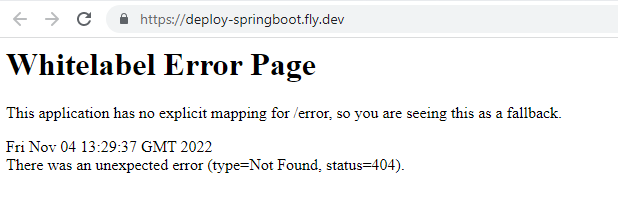
8. Para revisar la aplicación desplegada deberás dirigirte a <https://fly.io/dashboard>

******

9. Seleccionar la app, Ejemplo: ***deploy-springboot***

******

10. Finalmente, verificar el correcto funcionamiento del servidor:

******

*Referencia:* https://fly.io/docs/languages-and-frameworks/dockerfile/