Secure Spark APP

Johan Nicolas Cortes Torres
Escuela Colombiana de Ingenieria Julio Garivito
AREP

August 2020

Abstract

En el presente documento se encontrará información acerca de la elaboración de un laboratorio que se basaba en la introduccion al uso de dos servicios REST seguros y hacer que la conexion entre estos dos sea segura. El laboratorio consiste en crear dos maquinas virtual en AWS que contengan dos servicios distintos en cada maquina que se conecten de manera segura.

1 Introduction

El laboratorio se divide en tres partes fundamentales. La primera parte trata de la implementacion de dos servicios REST seguros, es decir con protocolo HTTPS. La segunda parte trata de conectar de manera segura ambos servicios. La tercera parte trata de montar ambos servicio a maquinas virtuales alojadas en AWS y erificar que la conexion entre ambas se realice correctamete.

Un servidor web es un programa que utiliza HTTP (Hypertext Transfer Protocol) para mostrar los archivos como paginas web, en respuesta a las diferentes solicitudes que hace un usuario.

Spark es un simple y expresivo marco web de Java/Kotlin DSL construido para un desarrollo rápido. La intención de Sparks es proporcionar una alternativa para los desarrolladores de Java/Kotlin que quieran desarrollar sus aplicaciones web de la forma más expresiva posible y con un mínimo de complejidad.

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

El Protocolo seguro de transferencia de hipertexto HTTPS es un protocolo de aplicación basado en el protocolo HTTP, destinado a la transferencia segura de

datos de hipertexto, es decir, es la versión segura de HTTP

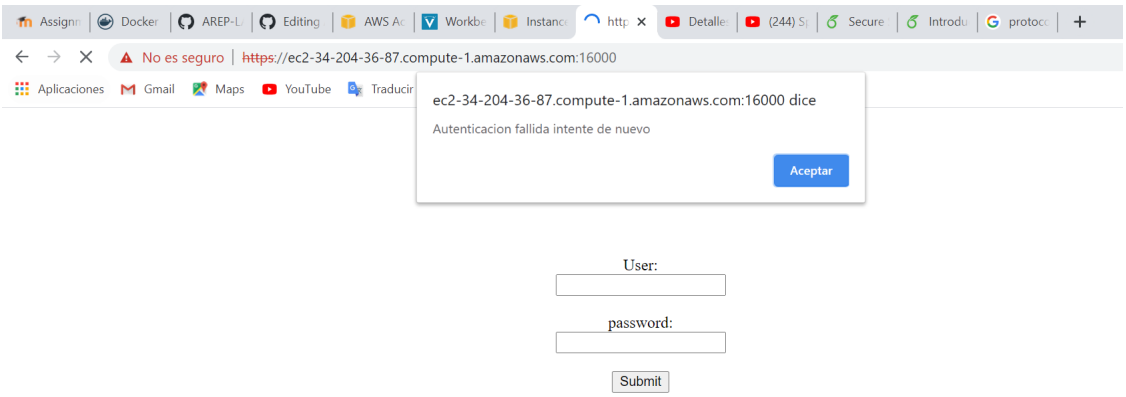
Amazon Web Services (AWS) es una colección de servicios de computación en la nube pública que en conjunto forman una plataforma de computación en la nube, ofrecidas a través de Internet por Amazon.com.

2 Arquitectura

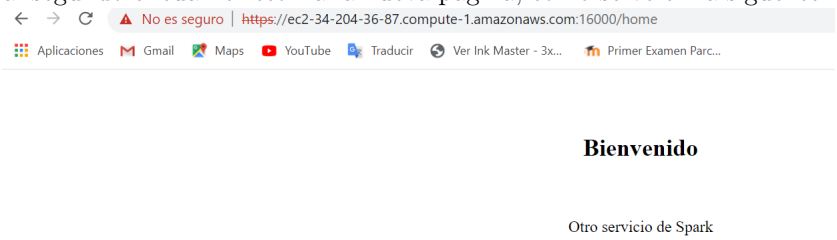
La arquiectura se basa en la existencia de dos clases principales y una clases secundarias. Las clases pricipales son: SparkWebServer y Balancer, ambos son servicios REST. Las clases secundarias son: JavaCliente y MongoDb, ambas hechas para realizar petciones a otros servicios. Ademas de estos, el proyecto se basa en archivos docker-compose y Dockerfile, los cuales permite crear imagenes y contenedores en Docker

2.1 Creacion de servicios REST

Para la implementacion de los servicio REST se usaron las clases SecureSparkServicesApp y OtherSparkService, ambas se basaba en Spark para construir el servicio correctamente. Ambas utilizarn el metodo "secure()" de spark para manejar el protocolo HTTPS. La primera clase permite hacer un login, verificando que el usuario se autentique correctamente, de resto no le permite acceder a al otro servicio REST. El segundo servicio rest solo funciona si el usuario esta autenticado, este es una pagina sencilla donde el se le indica al usuario que lo que muetra la pagina es desde otro servico web
Cuando el usuario no se autentica correctamente se le avisa y no le permite avanzar de la pagina de login, como se ve en la siguiente imagen:



Cuando el usuario se autentica correctamente, el primer servicio hace un request al segundo el cual le retorna la nueva pagina, como se ve en la siguente imagen:



2.2 Conexion entre servicios

Para la conectar los servicios y que esta conexion fuera segura, se implemento la clase UrlReader, la cual se basa en dos metodos para la correcta conexion segura y el manejo de la respuesta.

- urlprueba: Metodo que genera la conexion y verifica la seguridad de esta.
- readURL: Metodo que recive realiza la request hacia el segundo servicio.

2.3 Servicios hacia AWS

Para montar cada servico a una maquina de AWS se uso imagenes alojadas en docker hub, para mayor facilidad, esto de hizo mediante una configuracion en

Docker basada en dos archivos esenciales:

- docker-compose.yml: Este archivo permite crear varias imagenes alojadas en un contenedor de Docker, este permitia asignar a cada imagen su servicio, donde se encontraba su contexto y los puertos por los que se iba a ejecutar en Docker.
- Dockerfile: Este archivo permite crear una imagen de una clase especifica, es decir, mediante un clase alojada en el proyecto un puerto y el comando java -cp, nos crea la imagen de Docker para cada clase.

3 Conclusion

Concluyendo se realizó correctamente la implementación completa del laboratorio propuesto por la clase de AREP, se logro implementar dos servicios REST seguros mediante el protocolo HTTPS, ademas de esto se logro establecer una conexion segura entre estos dos servicios, permitiendo que, solo si se esta logeado se pueda acceder a los servicios. Finalmente se logro montar ambos servicios en dos maquina virtuales alojadas en AWS (cada servicio en una maquina diferente, todo esto mediante Docker.

References

[1] Laboratorio
http://campusvirtual.escuelaing.edu.co/moodle/pluginfile.php/230587/mod$_resource/content/0/TallerAll$
2020.

[2] Repositorio
https://github.com/jnicolasct/AREP-LAB7
2020.

[3] Video
https://www.youtube.com/watch?v=Ue0ADYVYltE
2020.