Práctica 1. Uso de S3 con Spring y despliegue en EC2

Enunciado

Aplicación básica (6 pts)

Un cliente acude a CodeURJC solicitando que desarrollemos una API REST que permita manejar los recursos de S3 de manera sencilla, ya que están descontentos con la que usan actualmente. Sus aplicaciones ya usan una interfaz de API definida, por lo que debemos ceñirnos a ella:

- **GET** /api/buckets/

  Recupera todos los buckets de nuestra cuenta.

- **GET** /api/buckets/<bucket_name>

  Recupera un bucket concreto dado su nombre

- **POST** /api/buckets/<bucket_name>

  Crea un nuevo bucket vacío.

- **POST** /api/buckets/<bucket_name>/uploadObject

  Sube un nuevo objeto a un bucket.

- **DELETE** /api/buckets/<bucket_name>

  Borra un bucket (solo si está vacío)

- **DELETE** /api/buckets/<bucket_name>/<object_name>

  Borra un objeto de un bucket

Nos proporciona además un fichero de Postman que utilizan para probar dicha API.

Además, se desea que esta aplicación sea desplegada en una instancia EC2 de forma que la API sea accesible y el cliente pueda probarla. El alumno entregará pruebas de que la aplicación ha estado disponible en forma de Video/GIF dónde se vea la instancia creada en el panel de EC2, así como que está disponible a través de su DNS público.

Dada nuestra extensa experiencia en la tecnología Java, hemos decidido que usaremos SpringBoot junto al SDK de AWS. Dado que la aplicación estará públicamente expuesta, y permite crear recursos en S3, debe estar securizada:

- Es necesario que la aplicación se despliegue en el puerto 443 (puerto HTTPS por defecto).
- Puede utilizarse un usuario en memoria, no es necesario utilizar base de datos.

Estos requisitos se consideran mínimos para que la práctica se pueda considerar aprobada. Además, se incluyen a continuación algunos requisitos adicionales que se valorarán en la nota.

Funcionalidad adicional (1 pto)

Aparte de los métodos descritos, el cliente quiere extender la API de la siguiente manera:

- Permitir que en la subida de objetos a un bucket se pueda definir si el objeto es público o privado.
- Implementar un nuevo método en el controlador que permita copiar archivos de un bucket a otro

Utilización de security group propio para la instancia (1 pto)

Se valorará el uso de un security group específico para la instancia. Se mostrará en el vídeo y este security group deberá filtrar los accesos entrantes de forma que no se permita acceder a la aplicación salvo a IPs del rango de la empresa (193.147.*.*).

NOTA: Este rango de direcciones IP es el de la URJC, si estáis fuera de la red de la universidad, podeis abrir un browser (e incluso una máquina virtual) de forma gratuita en el portal MyApps de la URJC: https://myapps.urjc.es/myapps. Todas las aplicaciones del MyApps funcionan en la red interna de la universidad bajo el mismo rango de IPv4.

Creación de una AMI con la aplicación (2 pts)

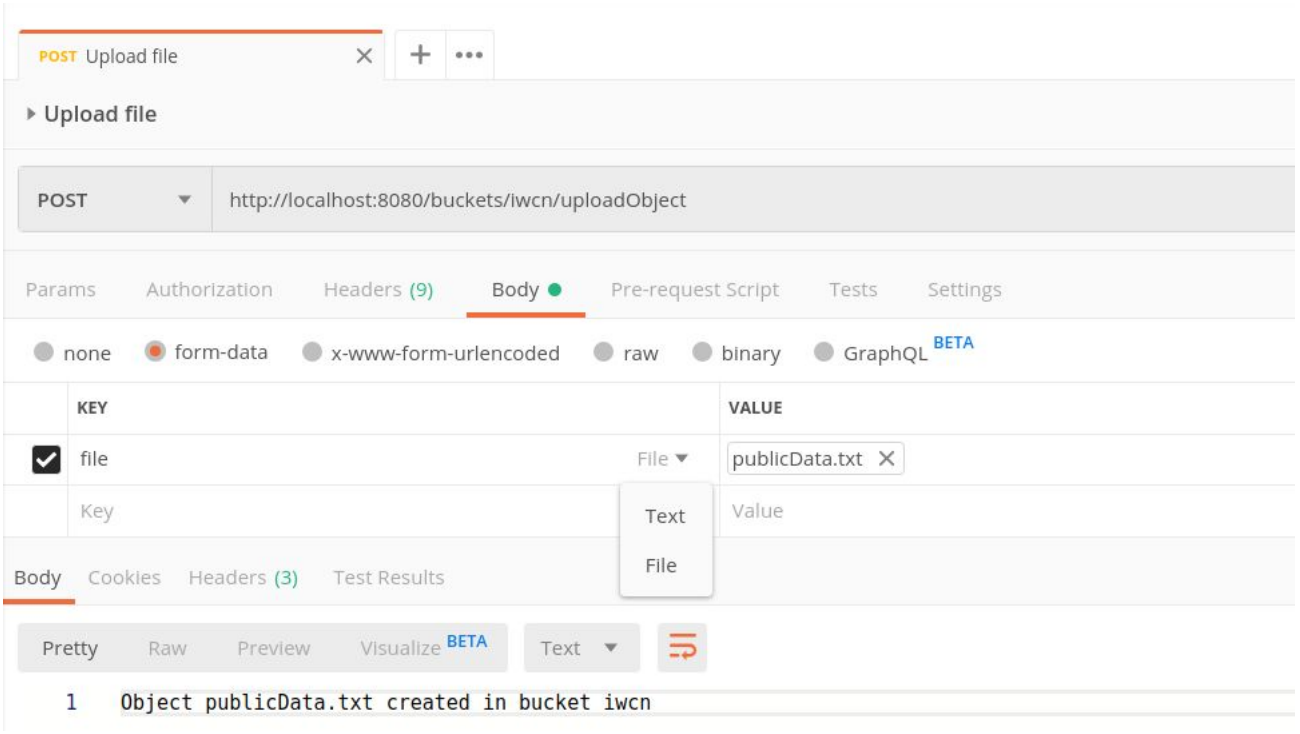
Se valorará la creación de una AMI con la aplicación. Se obtendrá el máximo de puntos si al arrancar una instancia a partir de esta AMI la aplicación arranca automáticamente (los pasos seguidos se definirán en un archivo README.md dentro del proyecto. Se incluirá en el vídeo la AMI creada y cómo se arranca la instancia a partir de ella.

Una vez grabado el video de que la aplicación funciona, **se detendrá y borrará la instancia** que contiene la aplicación para evitar costes adicionales, así como la **AMI y SNAPSHOTS** creadas.

Material de ayuda

Subida de archivos desde Postman

Para poder ejecutar la subida de archivos por REST desde Postman debemos pasarle en el cuerpo de la petición un parámetro de tipo File.



Recibir un archivo desde Java

```
@PostMapping("/{bucketName}/uploadObject")
public ResponseEntity<String> uploadFile(@PathVariable String bucketName,
@RequestParam("file") MultipartFile multiPartFile){
        String fileName = multiPartFile.getOriginalFilename();
        File file = new File(System.getProperty("java.io.tmpdir")+"/"+fileName);
        multiPartFile.transferTo(file);
        # SAVE TO S3 AND RETURN HTTP RESPONSE
}
```

Inicio de la aplicación al arrancar el sistema:

https://medium.com/@benmorel/creating-a-linux-service-with-systemd-611b5c8b91d6

https://medium.com/@ameyadhamnaskar/running-java-application-as-a-service-on-centos-599609d0c641

Documentación oficial del SDK:

https://docs.aws.amazon.com/es_es/AWSJavaSDK/latest/javadoc/com/amazonaws/services/s3/AmazonS3.html

Formato de entrega

La práctica se entregará teniendo en cuenta los siguientes aspectos:

- La práctica se entregará como un fichero .zip del proyecto Maven junto con el video o GIF que muestre que se ha desplegado la aplicación. El nombre del fichero .zip será el correo URJC del alumno (sin @alumnos.urjc.es).
- El proyecto se puede crear con cualquier editor o IDE.

Las prácticas se podrán realizar de forma individual o por parejas. En caso de que la práctica se haga por parejas:

- Sólo será entregada por uno de los alumnos
- El nombre del fichero .zip contendrá el correo de ambos alumnos separado por guión. Por ejemplo `p.perezf2019-z.gonzalez2019.zip`