05

Outras operações SDK

Transcrição

Dando prosseguimento à utilização do SDK do AWS, vamos executar novas operações com o Java, linguagem que estamos utilizando.

Criando um *bucket* via SDK

Já conseguimos listar os *buckets*! O que vamos fazer agora é criar outro para ver se o mesmo aparece na listagem. Todas as operações a serem trabalhadas estão relacionadas à classe `AmazonS3`, da qual já temos um objeto.

Então, vamos criar uma string com o nome do *bucket* a ser criado:

```
public class S3Exemplo {

    public static void main(String[] args) {

        // restante do código omitido

        String bucketName = "alura-s3-sdk";
        System.out.println("Criando bucket...");

        System.out.println("Listando buckets...");
        List<Bucket> listBuckets = s3.listBuckets();
        for (Bucket bucket : listBuckets) {
            System.out.println(bucket);
        }
    }

}
```

Agora, chamaremos o método `createBucket()`, passando para ele o nome do *bucket* que queremos criar:

```
public class S3Exemplo {

    public static void main(String[] args) {

        // restante do código omitido

        String bucketName = "alura-s3-sdk";
        System.out.println("Criando bucket...");
        s3.createBucket(bucketName);

        System.out.println("Listando buckets...");
        List<Bucket> listBuckets = s3.listBuckets();
        for (Bucket bucket : listBuckets) {
            System.out.println(bucket);
        }
    }
```

```
}
```

Ao executarmos a classe, veremos o *bucket* na listagem, logo, ele foi criado com sucesso. Mas normalmente não faremos esta operação de criação de *bucket* via SDK, e sim, por exemplo, pelo envio de um arquivo. Pensando na Alura, através de uma interface web o instrutor conseguiria fazer o upload dos arquivos necessários para o conteúdo textual do seu curso.

Upload de arquivo via SDK

Para subir um arquivo, poderemos pegar uma imagem e colocá-la na raiz do projeto. Feito isso, para representá-la programaticamente, teremos que trabalhar com as classes do `java.io`. Visto isso, vamos criar um novo `File`:

```
public class S3Exemplo {

    public static void main(String[] args) {

        // restante do código omitido

        System.out.println("Enviando arquivo...");
        File file = new File("amazon.png");
    }

}
```

Com o arquivo no código, pediremos para o S3 enviar um objeto por meio do método `putObject`, que possui algumas variações, sendo que utilizaremos aquela que recebe duas strings e o `File` como parâmetro. O primeiro parâmetro é o nome do *bucket*, e o segundo é o nome da *key* (pois quando salvamos um objeto, definimos o seu nome), e o terceiro parâmetro é o arquivo a ser enviado:

```
public class S3Exemplo {

    public static void main(String[] args) {

        // restante do código omitido

        System.out.println("Enviando arquivo...");
        File file = new File("amazon.png");
        s3.putObject(bucketName, "amazon-s3.png", file);
    }

}
```

Antes de executarmos a classe, é importante **comentarmos a linha que for criar o *bucket***, ou seja, a chamada do método `createBucket()`, pois já criamos o *bucket* e não conseguiremos criá-lo novamente:

```
public class S3Exemplo {

    public static void main(String[] args) {

        String accessKey = "Digite aqui sua Access Key ID";
```

```
        String secretKey = "Digite aqui dua Secret Access Key";

        BasicAWSCredentials awsCredentials = new BasicAWSCredentials(accessKey,
                secretKey);

        AmazonS3 s3 = AmazonS3ClientBuilder
                .standard()
                .withCredentials(
                        new AWSStaticCredentialsProvider(awsCredentials))
                .withRegion(Regions.SA_EAST_1).build();

        String bucketName = "alura-s3-sdk";
        // System.out.println("Criando bucket...");
        // s3.createBucket(bucketName);

        System.out.println("Listando buckets...");
        List<Bucket> listBuckets = s3.listBuckets();
        for (Bucket bucket : listBuckets) {
            System.out.println(bucket);
        }

        System.out.println("Enviando arquivo...");
        File file = new File("amazon.png");
        s3.putObject(bucketName, "amazon-s3.png", file);
    }

}
```

Executaremos a classe e a mensagem de envio é exibida no console. Mas como visualizaremos se o arquivo realmente existe no *bucket*? Poderemos acessá-lo via interface web, mas como estamos trabalhando de maneira programática, é possível listarmos os objetos de um *bucket*.

Listando objetos via SDK

Para isto, existe o método `listObjects()`, que recebe como parâmetro um `ListObjectsRequest()`, que teremos que configurar com o nome do *bucket*:

```
public class S3Exemplo {

    public static void main(String[] args) {

        // restante do código omitido

        System.out.println("Listando objetos do bucket...");
        s3.listObjects(new ListObjectsRequest().withBucketName(bucketName));
    }

}
```

O método `listObjects()` nos retorna um `ObjectListing` que irá nos auxiliar a percorrer todos os objetos, uma vez que ele possui o método `getObjectSummaries()` com as informações dos objetos. Poderemos iterar pelas informações dos objetos através de um `foreach`:

```
public class S3Exemplo {

    public static void main(String[] args) {

        // restante do código omitido

        System.out.println("Listando objetos do bucket...");
        ObjectListing listObjects = s3.listObjects(new ListObjectsRequest()
                .withBucketName(bucketName));
        for (S3ObjectSummary objectSummary : listObjects.getObjectSummaries()) {

        }
    }

}
```

No `objectSummary` temos informações do objeto, como o seu nome, disponível pelo método `getKey()`, e o seu tamanho, por meio do método `getSize()`:

```
public class S3Exemplo {

    public static void main(String[] args) {

        // restante do código omitido

        System.out.println("Listando objetos do bucket...");
        ObjectListing listObjects = s3.listObjects(new ListObjectsRequest()
                .withBucketName(bucketName));
        for (S3ObjectSummary objectSummary : listObjects.getObjectSummaries()) {
            System.out.println("*" + objectSummary.getKey() + " - "
                    + objectSummary.getSize());
        }
    }

}
```

Agora é possível executarmos a classe, e veremos que os objetos do *bucket* são listados, mais precisamente, o `amazon-s3.png`, justamente o objeto que subimos programaticamente.

Removendo objetos via SDK

Se conseguimos criar um *bucket*, fazer um upload e listar os objetos, deve ser possível também **remover objetos via SDK**. Então, vamos deletar o objeto que acabamos de enviar para o balde, utilizando o método `deleteObject()`, e passando para ele o local em que se encontra o objeto que queremos remover. Neste caso, o objeto está no *bucket*, e passaremos também sua *key*:

```
public class S3Exemplo {

    public static void main(String[] args) {

        // restante do código omitido
```

```
        System.out.println("Deletando objeto...");
        s3.deleteObject(bucketName, "amazon-s3.png");
    }

}
```

Mas antes de executarmos a classe, vamos comentar o código que faz o upload do arquivo, pois não queremos que um novo envio seja feito. Após a remoção, listaremos novamente os objetos. A classe `S3Exemplo` ficará assim:

```
public class S3Exemplo {

    public static void main(String[] args) {

        String accessKey = "Digite aqui sua Access Key ID";
        String secretKey = "Digite aqui dua Secret Access Key";

        BasicAWSCredentials awsCredentials = new BasicAWSCredentials(accessKey,
                secretKey);

        AmazonS3 s3 = AmazonS3ClientBuilder
                .standard()
                .withCredentials(
                        new AWSStaticCredentialsProvider(awsCredentials))
                .withRegion(Regions.SA_EAST_1).build();

        String bucketName = "alura-s3-sdk";
        // System.out.println("Criando bucket...");
        // s3.createBucket(bucketName);

        System.out.println("Listando buckets...");
        List<Bucket> listBuckets = s3.listBuckets();
        for (Bucket bucket : listBuckets) {
            System.out.println(bucket);
        }

        // System.out.println("Enviando arquivo...");
        // File file = new File("amazon.png");
        // s3.putObject(bucketName, "amazon-s3.png", file);

        System.out.println("Listando objetos do bucket...");
        ObjectListing listObjects = s3.listObjects(new ListObjectsRequest()
                .withBucketName(bucketName));
        for (S3ObjectSummary objectSummary : listObjects.getObjectSummaries()) {
            System.out.println("*" + objectSummary.getKey() + " - "
                    + objectSummary.getSize());
        }

        System.out.println("Deletando objeto...");
        s3.deleteObject(bucketName, "amazon-s3.png");

        System.out.println("Listando objetos do bucket...");
        ObjectListing listObjects2 = s3.listObjects(new ListObjectsRequest()
                .withBucketName(bucketName));
        for (S3ObjectSummary objectSummary : listObjects2.getObjectSummaries()) {
            System.out.println("*" + objectSummary.getKey() + " - "
                    + objectSummary.getSize());
```

```
            }
        }

    }
```

Agora sim, executaremos a classe e veremos que após a remoção do objeto, nada é listado! Era exatamente isso que queríamos. Como sabíamos que o *bucket* possuía apenas um objeto, quando o deletamos, o *bucket* deverá ficar vazio, não imprimindo nada em sua listagem.

Removendo *buckets* via SDK

Como criamos este *bucket* somente para exemplificarmos o uso do SDK, poderemos removê-lo usando o método `deleteBucket()`, para o qual passaremos o *bucket* a ser removido:

```
public class S3Exemplo {

    public static void main(String[] args) {

        // restante do código omitido

        s3.deleteBucket(bucketName);
    }

}
```

É importante lembrarmos que só é possível removermos *buckets* vazios. Após a remoção, se comentarmos o código que o cria e executarmos a classe novamente, receberemos um erro, pois estamos tentando listar objetos de um *bucket* que já não existe.

Então, nesta aula vimos que as operações realmente podem ser executadas via SDK de maneira programática, de acordo com nossa necessidade.