# Aula prática sobre o uso do Simple Storage Service (S3) da Amazon AWS

O Amazon S3 tem uma interface simples de serviços da web que você pode usar para armazenar e recuperar qualquer quantidade de dados, a qualquer momento, em qualquer lugar da web.

Conceitos básico

## Buckets

Um bucket é um contêiner para objetos armazenados no Amazon S3. Cada objeto está contido em um bucket. Por exemplo, se o objeto chamado photos/puppy.jpg estiver armazenado no bucket johnsmith, ele poderá ser endereçado usando o URL <https://johnsmith.s3.amazonaws.com/photos/puppy.jpg>

Buckets atendem a várias finalidades:

Eles organizam o namespace do Amazon S3 no nível mais elevado.

Identificam a conta responsável por alterações no armazenamento e na transferência de dados.

Exercem uma função no controle do acesso.

Servem como a unidade de agregação para relatório de uso.

É possível configurar os buckets para que sejam criados em uma região específica da AWS. Para obter mais informações, consulte Acessar um bucket. Você também pode configurar um bucket para que sempre que um objeto for adicionado, o Amazon S3 gere um ID exclusivo de versão e o atribua ao objeto. Para obter mais informações, consulte Usar versionamento.

Para obter mais informações sobre buckets, consulte Trabalho com buckets do Amazon S3.

## Objetos

Os objetos são as entidades fundamentais armazenadas no Amazon S3. Os objetos consistem em metadados e dados de objeto. A porção de dados não é visível para o Amazon S3. Os metadados são um conjunto de pares de nome e valor que descrevem o objeto. Eles incluem alguns metadados padrão, como a data da última modificação, e metadados HTTP padrão, como Content-Type. Você também pode especificar metadados personalizados no momento em que o objeto é armazenado.

Um objeto é identificado exclusivamente em um bucket por uma chave (nome) e um ID de versão. Para obter mais informações, consulte Chaves e Usar versionamento.

## Chaves

Uma chave é um identificador exclusivo de um objeto em um bucket. Cada objeto em um bucket tem exatamente uma chave. A combinação de um bucket, uma chave e um ID de versão identificam exclusivamente cada objeto. Portanto, é possível pensar no Amazon S3 como um mapa de dados básico entre "bucket + chave + versão" e o objeto em si. Cada objeto no Amazon S3 pode ser endereçado exclusivamente por meio da combinação do endpoint de serviço da web, do nome de bucket, da chave e, opcionalmente, de uma versão. Por exemplo, no URL http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl, "doc" é o nome do bucket e "2006-03-01/AmazonS3.wsdl" é a chave.

Para obter mais informações sobre chaves de objeto, consulte Chaves de objeto.

## Regiões

Você pode escolher a região geográfica da AWS onde o Amazon S3 armazenará os buckets criados. É possível escolher uma região para otimizar a latência, minimizar os custos ou atender a requisitos regulatórios. Os objetos armazenados em uma região nunca saem dela, a não ser que você os transfira explicitamente para outra região. Por exemplo, os objetos armazenados na região UE (Irlanda) nunca saem dela.

## Operações

Veja a seguir as operações mais comuns que você executará por meio da API.

### Operações comuns

**Criar um bucket** – crie e dê um nome ao seu próprio bucket para armazenar seus objetos.

**Gravar um objeto** – armazene dados criando ou substituindo um objeto. Ao gravar um objeto, você especifica uma chave exclusiva no namespace de seu bucket. Essa também é uma boa hora para especificar qualquer controle de acesso desejado no objeto.

**Ler um objeto** – leia os dados de volta. Você pode baixar os dados via HTTP ou BitTorrent.

**Excluir um objeto** – exclua alguns de seus dados.

**Listar chaves** – liste as chaves contidas em um de seus buckets. Você pode filtrar a lista de chaves com base em um prefixo.

## Interfaces de programação de aplicativos (APIs) do Amazon S3

A arquitetura do Amazon S3 foi desenvolvida para ser neutra em termos de linguagem de programação, usando as interfaces compatíveis da AWS para armazenar e recuperar objetos.

O Amazon S3 fornece uma interface REST e uma interface SOAP. Elas são semelhantes, mas há algumas diferenças. Por exemplo, na interface REST, os metadados são retornados em cabeçalhos HTTP. Como só oferecemos suporte a solicitações HTTP de até 4 KB (sem incluir o corpo), a quantidade de metadados que você pode fornecer é restrita.

# Atividade prática

Nessa atividade vamos acessar o S3 a partir da API Java disponível.

O acesso programático requer um

Abaixo os passos que são necessários para acessar o S3 a partir da Linguagem Java

1. Acessar o console da Amazon

<https://197368741842.signin.aws.amazon.com/console>

usuário: alunosS3c e senha: Uniara2019#

1. Criar credenciais de acesso programático no IAM (não disponível para contas acadêmicas e vamos usar uma conta pronta – não criar)
2. Criar um novo projeto Maven no Eclipse
3. Configurar o projeto para usar Java 8 no pom.xml do Maven

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.target>1.8</maven.compiler.target>

<maven.compiler.source>1.8</maven.compiler.source>

</properties>

1. Acrescentar as dependências abaixo no Maven

<dependencies>

<dependency>

<groupId>com.amazonaws</groupId>

<artifactId>aws-java-sdk-s3</artifactId>

<version>1.11.637</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

<scope>test</scope>

</dependency>

<dependencies>

1. Acessar o serviço S3 no console da AWS via Web
2. Criar um test case com o código abaixo para criação de buckets

**public** **class** S3CreateBucket {

String bucketName = "SEU RA"; // SUBSTITUA PELO SEU RA

**static** AmazonS3 *s3Client*;

**static** Regions *clientRegion* = Regions.***US\_EAST\_1***;

@BeforeClass

**public** **static** **void** init() {

*s3Client* = AmazonS3ClientBuilder.*standard*()

.withCredentials(**new** SystemPropertiesCredentialsProvider())

.withRegion(*clientRegion*)

.build();

}

@Test

**public** **void** testCreateBucket() {

**if** (!*s3Client*.doesBucketExistV2(bucketName)) {

*s3Client*.createBucket(**new** CreateBucketRequest(bucketName));

}

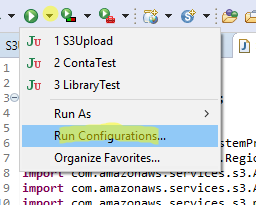
String bucketLocation = *s3Client*.getBucketLocation(**new** GetBucketLocationRequest(bucketName));

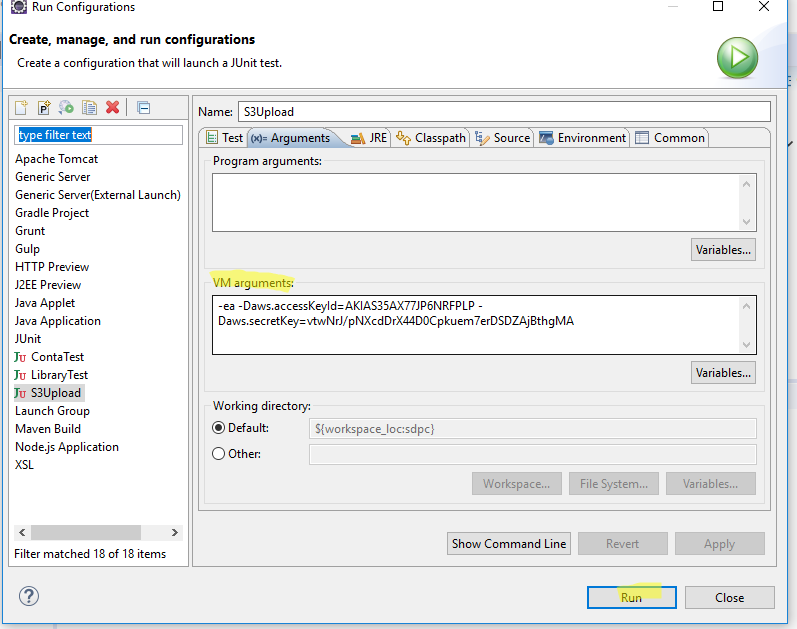
System.***out***.println("Localizacao do bucket " + bucketName + ": " + bucketLocation);

}

}

1. Executar o programa como um JUnit Test – vai falhar
2. Configurar a execução do JUnit para passar as credenciais abaixo como argumentos da JVM





-Daws.accessKeyId=AKIAS35AX77JP6NRFPLP -Daws.secretKey=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

1. Testar novamente a execução via JUnit – deve funcionar
2. Criar outros test cases para a Criação de Arquivo, baixar arquivo e listar arquivos do S3 conforme código abaixo, adaptando o nome do bucket, nomes e locais dos arquivos que vão ser feito upload e download para o S3.
3. Apresentar para contabilização de nota bimestral

## Referências

INTRODUÇÃO ao Amazon S3. Https://docs.aws.amazon.com/pt\_br/AmazonS3/latest/dev/Introduction.html: Amazon AWS, 25 set. 2019. Website.

## Anexos

### Criar arquivo

**public** **class** S3UploadFile {

String bucketName = "malara1";

String s3FileName = "pasta/cluster.txt";

String localFileName = "C:\\temp\\cluster.txt";

**static** AmazonS3 *s3Client*;

**static** Regions *clientRegion* = Regions.***US\_EAST\_1***;

@BeforeClass

**public** **static** **void** init() {

*s3Client* = AmazonS3ClientBuilder.*standard*()

.withCredentials(**new** SystemPropertiesCredentialsProvider())

.withRegion(*clientRegion*)

.build();

}

@Test

**public** **void** testCreateFile() {

**if** (!*s3Client*.doesBucketExistV2(bucketName)) {

*s3Client*.createBucket(**new** CreateBucketRequest(bucketName));

}

PutObjectRequest request = **new** PutObjectRequest(bucketName, s3FileName, **new** File(localFileName));

ObjectMetadata metadata = **new** ObjectMetadata();

metadata.setContentType("plain/text");

metadata.addUserMetadata("x-amz-meta-title", "someTitle");

request.setMetadata(metadata);

*s3Client*.putObject(request);

System.***out***.println("Arquivo enviado com sucesso para " + bucketName + ":/" + s3FileName);

}

}

### Listar Arquivos do Bucket

**public** **class** S3ListFiles {

String bucketName = "malara1";

**static** AmazonS3 *s3Client*;

**static** Regions *clientRegion* = Regions.***US\_EAST\_1***;

@BeforeClass

**public** **static** **void** init() {

*s3Client* = AmazonS3ClientBuilder.*standard*()

.withCredentials(**new** SystemPropertiesCredentialsProvider())

.withRegion(*clientRegion*)

.build();

}

@Test

**public** **void** testListFiles() {

**if** (!*s3Client*.doesBucketExistV2(bucketName)) {

*s3Client*.createBucket(**new** CreateBucketRequest(bucketName));

}

System.***out***.format("Objetos no S3 bucket %s:\n", bucketName);

**final** AmazonS3 s3 = AmazonS3ClientBuilder

.*standard*()

.withCredentials(**new** SystemPropertiesCredentialsProvider())

.withRegion(*clientRegion*).build();

ListObjectsV2Result result = s3.listObjectsV2(bucketName);

List<S3ObjectSummary> objects = result.getObjectSummaries();

**for** (S3ObjectSummary os : objects) {

System.***out***.println("\* " + os.getKey());

}

}

}

### Fazer Download de arquivo do bucket

**public** **class** S3Download {

String bucketName = "malara1";

String s3FileName = "pasta/cluster.txt";

String dwFileName = "C:\\temp\\cluster-dw.txt";

**static** AmazonS3 *s3Client*;

**static** Regions *clientRegion* = Regions.***US\_EAST\_1***;

@BeforeClass

**public** **static** **void** init() {

*s3Client* = AmazonS3ClientBuilder.*standard*()

.withCredentials(**new** SystemPropertiesCredentialsProvider())

.withRegion(*clientRegion*)

.build();

}

@Test

**public** **void** testDownloadFile() **throws** Exception {

S3Object o = *s3Client*.getObject(bucketName, s3FileName);

S3ObjectInputStream s3is = o.getObjectContent();

FileOutputStream fos = **new** FileOutputStream(**new** File(dwFileName));

**byte**[] read\_buf = **new** **byte**[1024];

**int** read\_len = 0;

**while** ((read\_len = s3is.read(read\_buf)) > 0) {

fos.write(read\_buf, 0, read\_len);

}

s3is.close();

fos.close();

}

}