Guia de Acesso ao S3 para Aplicação

# Formas de Acesso ao S3:

* Static Web Site (Público) (Apenas HTTP, não aceita HTTPS, apenas download, não faz upload)
* REST API com autenticação (HTTP/HTTPS Based)
* AWS SDK com autenticação (Java, Ruby, PHP, .Net)
* HTTP com URL Pré-Assinada por chamada SDK

# Formas de Acesso do Portal:

* Disco Local
* NFS via EFS ou Storage Gateway
* REST API (HTTP/HTTPS Based) (controle total do bucket)
* AWS SDK com autenticação (controle total do Bucket)

# Formas de Acesso do Smart:

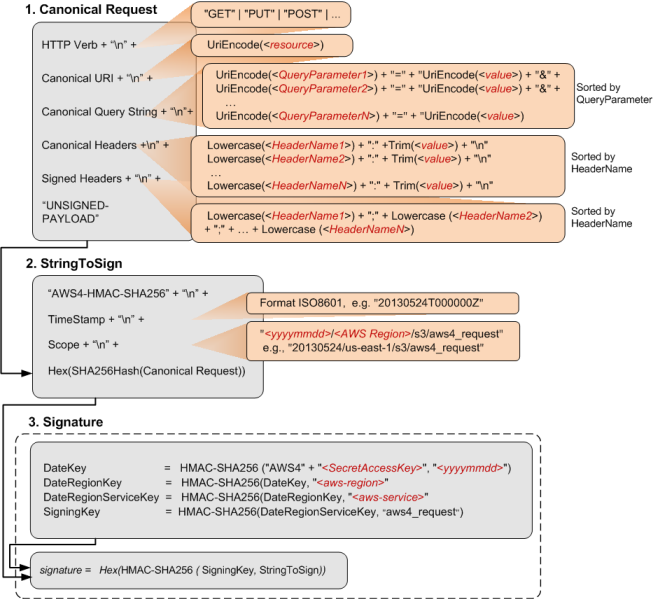
* HTTP via Static Web Site: Necessário URL criptografada – Problemas de Segurança
* REST API com autenticação (HTTP/HTTPS Based): Necessário codificar o protocolo
* AWS SDK com autenticação: Necessario acessar as APIs ou desenvolver
* Via Portal: Portal acessa no S3 e devolve para o Smart.
* HTTP com URL Pré-assinada pelo portal

# Acesso via REST API (HTTP based)

<https://docs.aws.amazon.com/AmazonS3/latest/dev/RESTAPI.html>

## Gerando String de Autenticação (AWS Signature Version 4):

<http://docs.aws.amazon.com/AmazonS3/latest/API/sig-v4-authenticating-requests.html>



## Autenticando Requisições: Usando “HTTP Authorization Header” (preferível)

<http://docs.aws.amazon.com/AmazonS3/latest/API/sigv4-auth-using-authorization-header.html>

### Transferindo uma carga em um Único Pedaço:

<http://docs.aws.amazon.com/AmazonS3/latest/API/sig-v4-header-based-auth.html>

### Transferindo uma carga em Múltiplos Pedaços:

<http://docs.aws.amazon.com/AmazonS3/latest/API/sigv4-streaming.html>

## Autenticando Requisições: Usando “Query String Parameters”

<http://docs.aws.amazon.com/AmazonS3/latest/API/sigv4-query-string-auth.html>

# Acesso via URL Pré-Assinada (HTTP/HTTPS Based)

<http://docs.aws.amazon.com/AmazonS3/latest/dev/ShareObjectPreSignedURLJavaSDK.html>

## Exemplo:

AmazonS3 s3client = new AmazonS3Client(new ProfileCredentialsProvider());

java.util.Date expiration = new java.util.Date();

long msec = expiration.getTime();

msec += 1000 \* 60 \* 60; // 1 hour.

expiration.setTime(msec);

GeneratePresignedUrlRequest generatePresignedUrlRequest =

new GeneratePresignedUrlRequest(bucketName, objectKey);

generatePresignedUrlRequest.setMethod(HttpMethod.GET); // Default.

generatePresignedUrlRequest.setExpiration(expiration);

URL s = s3client.generatePresignedUrl(generatePresignedUrlRequest);

# Acesso via AWS SDK (Java)

<https://docs.aws.amazon.com/AmazonS3/latest/dev/MakingAuthenticatedRequests.html>

## AWS SDK for Java API Reference

<http://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/>

## Acessos ao Vector Cielo Vendas:

* Usuário Portal: **cvd-portal** - [arn:aws:iam::853982059333:user/cvd-portal]
* Usuario Smart: **cvd-smart** - [arn:aws:iam::853982059333:user/cvd-smart]
* Access Key: Será enviado por e-mail
* Secret Access Key: Será enviado por e-mail
* Bucket: **vek-cvd-arc** - [arn:aws:s3:::vek-cvd-arc]

## Acessando via IAM User em Java

<https://docs.aws.amazon.com/AmazonS3/latest/dev/AuthUsingAcctOrUserCredJava.html>

<https://javatutorial.net/java-s3-example>

## Manipulando objetos na Amazon S3:

### Usando o Provedor de Credencias Padrão: [DefaultAWSCredentialsProviderChain](https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/index.html?com/amazonaws/auth/DefaultAWSCredentialsProviderChain.html)

Procura automaticamente 1 credencial nas seguintes opções:

* Variável de ambiente do SO
  + aws\_access\_key\_id=
  + aws\_secret\_access\_key=
* Propriedades de Sistema do Java / Variáveis do Java
  + aws.accessKeyId=
  + aws.secretKey=
* Credencial Padrão no arquivo de Profiles
  + ~.aws/credentials
* Amazon ECS Container
* Instance Profile Credentials (VM EC2)

Chamada:

* awsCreds = new DefaultAWSCredentialsProviderChain();

### Utilizando as credencias já definidas nas seguintes situações:

1. Pelo arquivo ~.aws/credentials: [ProfileCredentialsProvider](https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/index.html?com/amazonaws/auth/profile/ProfileCredentialsProvider.html)
   1. AWSCredentials awsCreds = new ProfileCredentialsProvider().getCredentials(); // Default Profile
   2. AWSCredentials awsCreds = new ProfileCredentialsProvider(“cvd-portal”).getCredentials(); // Profile cvd-portal
2. Variáveis de ambiente do SO: [EnvironmentVariableCredentialsProvider](https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/index.html?com/amazonaws/auth/EnvironmentVariableCredentialsProvider.html)
   1. AmazonS3 s3Client = AmazonS3ClientBuilder.standard().withCredentials(new EnvironmentVariableCredentialsProvider()).build();
3. Variáveis de ambiente do Java / Propriedades de Sistema: [SystemPropertiesCredentialsProvider](https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/index.html?com/amazonaws/auth/SystemPropertiesCredentialsProvider.html)
4. Usando as credenciais da VM EC2: [InstanceProfileCredentialsProvider](https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/index.html?com/amazonaws/auth/InstanceProfileCredentialsProvider.html)
5. Passando suas próprias credencias: [AWSStaticCredentialsProvider](https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/index.html?com/amazonaws/auth/AWSStaticCredentialsProvider.html), [AWSCredentials](https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/index.html?com/amazonaws/auth/AWSCredentials.html), [BasicAWSCredentials](https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/index.html?com/amazonaws/auth/BasicAWSCredentials.html)
   1. BasicAWSCredentials awsCreds = new BasicAWSCredentials("access\_key\_id", "secret\_key\_id");
   2. AWSCredentials awsCreds = new BasicAWSCredentials("YourAccessKeyID", "YourSecretAccessKey");
   3. AmazonS3 s3Client = AmazonS3ClientBuilder.standard().withCredentials(new AWSStaticCredentialsProvider(awsCreds)).build();

### Outros exemplos de acesso:

credentialsProvider = new ProfileCredentialsProvider();

AmazonS3 s3client = new AmazonS3Client(new ProfileCredentialsProvider());

AmazonS3 s3client = new AmazonS3Client(awsCreds);

### Criando Bucket

String bucketName = "vek-cvd-arc";

s3client.createBucket(bucketName);

### Apagando um Bucket

s3client.deleteBucket(bucketName);

### Listando um Bucket

for (Bucket bucket : s3client.listBuckets()) {

System.out.println(" - " + bucket.getName());

}

### Criando uma pasta

public static void createFolder(String bucketName, String folderName, AmazonS3 client) {

// create meta-data for your folder and set content-length to 0

ObjectMetadata metadata = new ObjectMetadata();

metadata.setContentLength(0);

// create empty content

InputStream emptyContent = new ByteArrayInputStream(new byte[0]);

// create a PutObjectRequest passing the folder name suffixed by /

PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName,

folderName + SUFFIX, emptyContent, metadata);

// send request to S3 to create folder

client.putObject(putObjectRequest);

}

### Apagando uma Pasta (a pasta deve estar vazia)

s3client.deleteObject(bucketName, folderName);

### Subindo um Arquivo

String fileName = folderName + SUFFIX + "testvideo.mp4";

s3client.putObject(new PutObjectRequest(bucketName, fileName,

new File("C:\\Users\\user\\Desktop\\testvideo.mp4")));

### Baixando um Arquivo

AmazonS3 s3Client = new AmazonS3Client(new ProfileCredentialsProvider());

S3Object object = s3Client.getObject(

new GetObjectRequest(bucketName, key));

InputStream objectData = object.getObjectContent();

// Process the objectData stream.

objectData.close();

### Apagando um Arquivo

s3client.deleteObject(bucketName, fileName);

# Configurando Região Padrão:

<http://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-credentials.html>

# Set up the AWS SDK for Java

<http://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-install.html>

# Configurando AWS Toolkit for Eclipse IDE

<https://aws.amazon.com/eclipse/>

# AWS Toolkit for Eclipse User Guide

<https://docs.aws.amazon.com/toolkit-for-eclipse/v1/user-guide/>

# Exemplo 1:

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import com.amazonaws.AmazonClientException;

import com.amazonaws.AmazonServiceException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;

import com.amazonaws.services.s3.AmazonS3;

import com.amazonaws.services.s3.AmazonS3Client;

import com.amazonaws.services.s3.model.GetObjectRequest;

import com.amazonaws.services.s3.model.S3Object;

public class GetObject {

private static String bucketName = "\*\*\* provide bucket name \*\*\*";

private static String key = "\*\*\* provide object key \*\*\*";

public static void main(String[] args) throws IOException {

AmazonS3 s3Client = new AmazonS3Client(new ProfileCredentialsProvider());

try {

System.out.println("Downloading an object");

S3Object s3object = s3Client.getObject(new GetObjectRequest(

bucketName, key));

System.out.println("Content-Type: " +

s3object.getObjectMetadata().getContentType());

displayTextInputStream(s3object.getObjectContent());

// Get a range of bytes from an object.

GetObjectRequest rangeObjectRequest = new GetObjectRequest(

bucketName, key);

rangeObjectRequest.setRange(0, 10);

S3Object objectPortion = s3Client.getObject(rangeObjectRequest);

System.out.println("Printing bytes retrieved.");

displayTextInputStream(objectPortion.getObjectContent());

} catch (AmazonServiceException ase) {

System.out.println("Caught an AmazonServiceException, which" +

" means your request made it " +

"to Amazon S3, but was rejected with an error response" +

" for some reason.");

System.out.println("Error Message: " + ase.getMessage());

System.out.println("HTTP Status Code: " + ase.getStatusCode());

System.out.println("AWS Error Code: " + ase.getErrorCode());

System.out.println("Error Type: " + ase.getErrorType());

System.out.println("Request ID: " + ase.getRequestId());

} catch (AmazonClientException ace) {

System.out.println("Caught an AmazonClientException, which means"+

" the client encountered " +

"an internal error while trying to " +

"communicate with S3, " +

"such as not being able to access the network.");

System.out.println("Error Message: " + ace.getMessage());

}

}

private static void displayTextInputStream(InputStream input)

throws IOException {

// Read one text line at a time and display.

BufferedReader reader = new BufferedReader(new

InputStreamReader(input));

while (true) {

String line = reader.readLine();

if (line == null) break;

System.out.println(" " + line);

}

System.out.println();

}

}

# Exemplo 2:

import java.io.IOException;

import com.amazonaws.AmazonClientException;

import com.amazonaws.AmazonServiceException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;

import com.amazonaws.services.s3.AmazonS3;

import com.amazonaws.services.s3.AmazonS3Client;

import com.amazonaws.services.s3.model.ListObjectsRequest;

import com.amazonaws.services.s3.model.ListObjectsV2Request;

import com.amazonaws.services.s3.model.ListObjectsV2Result;

import com.amazonaws.services.s3.model.ObjectListing;

import com.amazonaws.services.s3.model.S3ObjectSummary;

public class ListKeys {

private static String bucketName = "\*\*\*bucket name\*\*\*";

public static void main(String[] args) throws IOException {

AmazonS3 s3client = new AmazonS3Client(new ProfileCredentialsProvider());

try {

System.out.println("Listing objects");

final ListObjectsV2Request req = new ListObjectsV2Request().withBucketName(bucketName).withMaxKeys(2);

ListObjectsV2Result result;

do {

result = s3client.listObjectsV2(req);

for (S3ObjectSummary objectSummary :

result.getObjectSummaries()) {

System.out.println(" - " + objectSummary.getKey() + " " +

"(size = " + objectSummary.getSize() +

")");

}

System.out.println("Next Continuation Token : " + result.getNextContinuationToken());

req.setContinuationToken(result.getNextContinuationToken());

} while(result.isTruncated() == true );

} catch (AmazonServiceException ase) {

System.out.println("Caught an AmazonServiceException, " +

"which means your request made it " +

"to Amazon S3, but was rejected with an error response " +

"for some reason.");

System.out.println("Error Message: " + ase.getMessage());

System.out.println("HTTP Status Code: " + ase.getStatusCode());

System.out.println("AWS Error Code: " + ase.getErrorCode());

System.out.println("Error Type: " + ase.getErrorType());

System.out.println("Request ID: " + ase.getRequestId());

} catch (AmazonClientException ace) {

System.out.println("Caught an AmazonClientException, " +

"which means the client encountered " +

"an internal error while trying to communicate" +

" with S3, " +

"such as not being able to access the network.");

System.out.println("Error Message: " + ace.getMessage());

}

}

}