

---

# Computação na nuvem

## ISEL – LEIRT / LEIC / LEIM

- Serviço *Google Cloud Storage*
- Controlo de acessos e chamadas via API Java

José Simão [jsimao@cc.isel.ipl.pt](mailto:jsimao@cc.isel.ipl.pt) ; [jose.simao@isel.pt](mailto:jose.simao@isel.pt)

Luís Assunção [lass@isel.ipl.pt](mailto:lass@isel.ipl.pt) ; [luis.assuncao@isel.pt](mailto:luis.assuncao@isel.pt)

# Controlo de acessos aos serviços

- O controlo de acessos à API é sempre feito através de *OAuth 2.0*, obtendo-se um *access-token* para acesso ao recurso pretendido
- No entanto, sem o processo de autorização do utilizador, as aplicações acedem diretamente aos recursos da seguinte forma:

Fluxo ***client credentials***: No caso das API da Google, as aplicações cliente obtêm um *access-token*, após se autenticarem com uma chave privada correspondente a uma **conta de serviço**

<https://developers.google.com/identity/protocols/OAuth2ServiceAccount>



# Contas de serviço

---

- O acesso programático aos serviços é feito através das credenciais de uma ou várias contas de serviço
- As contas de serviço têm:
  - um identificador único (com formato de email)
  - um conjunto de chaves geridas pela *Google Cloud Platform*
  - 1 ou mais **pares de chaves** para uso das aplicações
- As aplicações usam a chave privada da conta de serviço para se autenticarem nos diferentes serviços (*storage, compute engine, etc.*)
- O acesso pode ser feito a partir de computadores fora das infraestruturas da Google (*on-premises* ou outras clouds públicas), ou a partir dos próprios serviços Google

# Criação de conta de serviço

The screenshot displays the Google Cloud Platform IAM & Admin console for project PJBASE-CN2021. The left sidebar shows the navigation menu with 'Service Accounts' selected. The main content area is titled 'Service accounts' and includes a '+ CREATE SERVICE ACCOUNT' button. The 'Create service account' wizard is shown with four numbered steps:

- 1 Service account details**: This step includes a 'Service account name' field with the value 'cn-v2021-storage', a 'Service account ID' field with the value 'cn-v2021-storage-569', and a 'Service account description' field. A green arrow points to the 'CREATE' button.
- 2 Grant this service account access to the project (optional)**: This step is currently empty.
- 3 Grant users access to this service account (optional)**: This step is currently empty.
- 4 Service account details**: This step shows a 'Select a role' dropdown menu with the 'Storage Admin' role selected. A green arrow points to the 'DONE' button.

The 'Storage Admin' role is highlighted in the dropdown menu, showing its permissions: 'Full control of GCS'. The 'DONE' button is located at the bottom of the 'Grant users access to this service account (optional)' step.

# Chave privada da conta de serviço

Service accounts

[+ CREATE SERVICE ACCOUNT](#)

[DELETE](#)

[+ MANAGE ACCESS](#)

## Service accounts for project 'PJBASE-CN2021'

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps or systems running outside Google Cloud.

Organisation policies can be used to secure service accounts and block risky service account features, such as automatic IAM Grants, key creation/upload or the

[Filter](#) Enter property name or value

<input type="checkbox"/>	Email	Status	Name <span>↑</span>
<input type="checkbox"/>	pjbases-cn2021@appspot.gserviceaccount.com		App Engine default service account
<input type="checkbox"/>	cn-v2021-storage@pjbases-cn2021.iam.gserviceaccount.com		cn-v2021-storage
<input type="checkbox"/>	405983611331-compute@developer.gserviceaccount.com		
<input type="checkbox"/>	compute-engine-service@pjbases-cn2021.iam.gserviceaccount.com		
<input type="checkbox"/>	serv-accounts@pjbases-cn2021.iam.gserviceaccount.com		

←

cn-v2021-storage

DETAILS

PERMISSIONS

KEYS

METRICS

LOGS

### Keys

Add a new key pair or upload a public key certificate from an existing key pair. Please note that public certificates need to be in RSA\_X509\_PEM format. [Learn more about upload key formats](#)

Block service account key creation using [organization policies](#). [Learn more about setting organization policies for service accounts](#)

ADD KEY ▾

Create new key

Upload existing key

Key creation date

Key expiry date

# Chave privada da conta de serviço

**Atenção:**  
A perda da chave privada implica gerar uma nova

← cn-v2021-storage

DETAILS PERMISSIONS **KEYS** METRICS LOGS

### Keys

Add a new key pair or upload a public key certificate from an existing key pair. Please note that public certificates need to be in RSA\_X509\_PEM format. [Learn more about upload key formats](#)

Block service account key creation using [organization policies](#).  
[Learn more about setting organization policies for service accounts](#)

ADD KEY ▾

- Create new key
- Upload existing key

Key creation date	Key expiry date
-------------------	-----------------

### Create private key for 'storage-service-account'

Downloads a file that contains the private key. Store the file securely because this key cannot be recovered if lost.

Key type

- ☒ JSON  
Recommended
- ☐ P12  
For backward compatibility with code using the P12 format

CANCEL CREATE

```
{  
  "type": "service_account",  
  "project_id": "PJBASE-CN2021",  
  "private_key_id": ...  
}
```

Opção “**Create**” permite guardar em ficheiro local com nome à escolha, por exemplo:  
*PJBASE-CN2021-ab472gd.json*

# API Java

---

- A API Java permite, entre outras operações
  - A criação e remoção de *bucket*
  - *Upload, download* e remoção de *blobs*
  - Modificação de *metadados*
  - Alteração de *permissões*
  - Operações em *batch*
- A documentação da API está no seguinte *link*:

<https://googleapis.dev/java/google-cloud-storage/latest/index.html>
- Os slides seguintes apresentam troços de código que ilustram as operações essenciais

# Autenticação no acesso ao serviço

- Caso geral: Variável de ambiente com chave

`GOOGLE_APPLICATION_CREDENTIALS=<pathname do ficheiro json com chave>`

```
static void authWithEnvironmentalVariable() {  
  
    StorageOptions options = StorageOptions.getDefaultInstance();  
  
    String projID = options.getProjectId();  
  
    Storage storage = options.getService();  
  
    // operations with storage object  
    // - create bucket  
    // - create blob  
    // ...  
}
```



# Classes e localização de *storage*

```
// classe StorageClass existente  
// na API Java
```

```
StorageClass[] CLASSES =  
    new StorageClass[] {  
        StorageClass.STANDARD,  
        StorageClass.NEARLINE,  
        StorageClass.COLDLINE,  
        StorageClass.ARCHIVE  
    };
```

## Regions

Continent  
North America

All regions are at least 100 miles apart.

### Region Name

Region Description

**NORTHAMERICA-NORTHEAST1** Montréal  
**US-CENTRAL1** Iowa  
**US-EAST1** South Carolina  
**US-EAST4** Northern Virginia  
**US-WEST1** Oregon  
**US-WEST2** Los Angeles  
**US-WEST3** Salt Lake City

South America

**SOUTHAMERICA-EAST1** São Paulo

Europe

**EUROPE-NORTH1** Finland  
**EUROPE-WEST1** Belgium  
**EUROPE-WEST2** London  
**EUROPE-WEST3** Frankfurt  
**EUROPE-WEST4** Netherlands  
**EUROPE-WEST6** Zürich

Asia

**ASIA-EAST1** Taiwan  
**ASIA-EAST2** Hong Kong  
**ASIA-NORTHEAST1** Tokyo  
**ASIA-NORTHEAST2** Osaka  
**ASIA-NORTHEAST3** Seoul  
**ASIA-SOUTH1** Mumbai  
**ASIA-SOUTHEAST1** Singapore

Australia

**AUSTRALIA-SOUTHEAST1** Sydney

## Multi-regions

### Multi-Region Name

Multi-Region Description

**ASIA**

Data centers in Asia

**EU**

Data centers within member states of the European Union1

**US**

Data centers in the United

1 Object data added to a bucket in the EU multi-region is not stored in the EUROPE-WEST2

## Dual-regions

### Dual-Region Name

Dual-Region Description

**EUR4**

EUROPE-NORTH1 and EUROPE-WEST4.

**NAM4**

US-CENTRAL1 and US-EAST1.

# API - entidades

---

```
BucketInfo.newBuilder("a-unique-bucket-name")  
    // See here for possible values:  
    // http://g.co/cloud/storage/docs/storage-classes  
    .setStorageClass(StorageClass.STANDARD)  
    // See here for possible values:  
    // http://g.co/cloud/storage/docs/bucket-locations#location-mr  
    .setLocation("EUROPE-WEST1")  
    .build();
```

```
BlobId blobId = BlobId.of(bucketName, blobName);  
String contentType = Files.probeContentType(pathnameFile);  
BlobInfo blobInfo = BlobInfo.newBuilder(blobId)  
    .setContentType(contentType)  
    .build();
```

# API - *interface Storage*

---

**Bucket** create(**BucketInfo** bucketInfo)

**Blob** create(**BlobInfo**, byte[])

**Bucket** get(String)

**Blob** get(**BlobId**)

**boolean** delete(**BlobId**)

// retorna um Iterable<Bucket>

**list().iterateAll()**

// retorna canal para *download* um *stream* de bytes

**ReadChannel** reader(**BlobId** blob)

// retorna canal para upload num stream de bytes

**WriteChannel** writer(**BlobInfo** blobInfo)

# API - classe Blob

---

**boolean delete()**

**void downloadTo(Path path)**

// retorna canal para *download* um *stream* de bytes

**ReadChannel reader()**

// atualiza o metadado **cache-control** para não ter cache

**blob.toBuilder().setCacheControl("no-cache").build().update()**

// atualiza os metadados do blob com pares <key, value>

**Map<String, String> newMetadata=new HashMap<String, String>()**

**{ { put("key1", "value1"); put("key2", "value2"); } };**

**blob.toBuilder().setMetadata(newMetadata).build().update()**

// atualiza a Access Control List (ACL) do blob

**Acl createAcl(Acl acl)**

# List project Buckets and Blobs

```
public void listBuckets(String projID) {
    System.out.println("Buckets in Project=" + projID + ":");
    for (Bucket bucket : storage.list().iterateAll()) {
        System.out.println("  " + bucket.toString());
        for (Blob blob : bucket.list().iterateAll()) {
            System.out.println("    "+blob.toString());
        }
    }
}
```

```
    Bucket{name=cn-bucket-public}
        Blob{bucket=cn-bucket-public, name=fotobugio, g
        Blob{bucket=cn-bucket-public, name=isellogo, ge
    Bucket{name=cn-myfotos}
        Blob{bucket=cn-myfotos, name=20150202_134117.jp
        Blob{bucket=cn-myfotos, name=D:\Disciplinas\Com
        Blob{bucket=cn-myfotos, name=forajogo.PNG, gene
        Blob{bucket=cn-myfotos, name=pictures/nasa.jpg,
    Bucket{name=cn-v1819-62d61n}
        Blob{bucket=cn-v1819-62d61n, name=PDFfile.pdf,
```

# Criar/Apagar *Bucket*

```
public Bucket CreateBucket(String bucketName, StorageClass storageClass,  
                           String location) {  
  
    Bucket bucket = storage.create(  
        BucketInfo.newBuilder(bucketName)  
        // See here for possible values:  
        //http://g.co/cloud/storage/docs/storage-classes  
        .setStorageClass(storageClass)  
        // Possible values:  
        // http://g.co/cloud/storage/docs/bucket-locations#Location-mr  
        .setLocation(location)  
        .build());  
  
    return bucket;  
}
```

```
public void deleteBucket(String bucketName) {  
    Bucket bucket = storage.get(bucketName);  
    bucket.delete();  
}
```

# Upload de Blob no Bucket (i)

```
public BlobId uploadBlobToBucket(String bucketName, String blobName,  
                                String absFileName) throws Exception {  
    Path uploadFrom=Paths.get(absFileName);  
    String contentType=Files.probeContentType(uploadFrom);  
    BlobId blobId = BlobId.of(bucketName, blobName);  
    BlobInfo blobInfo =  
        BlobInfo.newBuilder(blobId).setContentType(contentType).build();  
    if (Files.size(uploadFrom) < 1_000_000) {  
        byte[] bytes = Files.readAllBytes(uploadFrom);  
        // create the blob in one request.  
        storage.create(blobInfo, bytes);  
    } else {  
        // blobs maiores que 1Mbyte  
        ...  
    }  
    return blobId;  
}
```

# Upload de Blob no Bucket (ii)

```
// blobs maiores que 1Mbyte
try (WriteChannel writer = storage.writer(blobInfo)) {
    byte[] buffer = new byte[1024];
    try (InputStream input=Files.newInputStream(uploadFrom)) {
        int limit;
        while ((limit = input.read(buffer)) >= 0) {
            try {
                writer.write(ByteBuffer.wrap(buffer, 0, limit));
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
    }
}
```

The try-with-resources statement is a try statement that declares one or more resources. A resource is an object that must be closed after the program is finished with it. The try-with-resources statement ensures that each resource is closed at the end of the statement.

<https://docs.oracle.com/javase/tutorial/essential/exceptions/tryResourceClose.html>



# Download de Blob (i)

---

```
public void downloadBlobFromBucket(String bucketName, String blobName,  
                                   String absFileName) throws Exception {  
    Path downloadTo = Paths.get(absFileName);  
    BlobId blobId = BlobId.of(bucketName, blobName);  
    Blob blob = storage.get(blobId);  
    PrintStream writeTo = new PrintStream(Files.newOutputStream(downloadTo));  
  
    if (blob.getSize() < 1_000_000) {  
        // Blob is small read all its content in one request  
        byte[] content = blob.getContent();  
        writeTo.write(content);  
    } else {  
        // When Blob size is big use the blob's channel reader  
        ... // in next slide  
    }  
    writeTo.close();  
}
```

# Download de Blob (ii)

---

*// When Blob size is big use the blob's channel reader*

```
try (ReadChannel reader = blob.reader()) {  
    WritableByteChannel channel = Channels.newChannel(writeTo);  
    ByteBuffer bytes = ByteBuffer.allocate(64 * 1024);  
    while (reader.read(bytes) > 0) {  
        bytes.flip();  
        channel.write(bytes);  
        bytes.clear();  
    }  
}
```

# Alterar permissões do Blob

- Como colocar um Blob público com permissão de leitura

```
BlobId blobId = BlobId.of(bucketName, blobName);
```

```
Blob blob = storage.get(blobId);
```

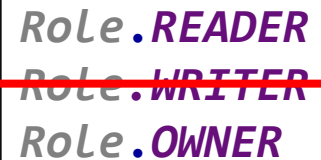
```
Acl.Entity aclEnt = Acl.User.ofAllUsers();
```

```
Acl.Role[] roles = Acl.Role.values();
```

```
Acl.Role role = Acl.Role.READER;
```

```
Acl acl = Acl.newBuilder(aclEnt, role).build();
```

```
blob.createAcl(acl);
```



*Role*.*READER*  
~~*Role*.*WRITER*~~  
*Role*.*OWNER*

# Operações em *batch* com notificação

```
StorageBatch batch = storage.batch();
for (Blob blob : bucket.list().iterateAll()) {
    final String blobName = blob.getName();
    batch.delete(blob.getBlobId()).notify(
        new BatchResult.Callback<Boolean, StorageException>() {
            @Override
            public void success(Boolean aBoolean) {
                System.out.println(blobName+" was deleted");
            }
            @Override
            public void error(StorageException e) {
                System.out.println(blobName+" Error!!!");
            }
        });
}
batch.submit();
```

# Dependência para a Google Storage API (*pom.xml*)

---

...

**<dependencies>**

*<!-- https://mvnrepository.com/artifact/com.google.cloud/google-cloud-storage -->*

**<dependency>**

**<groupId>***com.google.cloud***</groupId>**

**<artifactId>***google-cloud-storage***</artifactId>**

**<version>***2.6.0***</version>**

**</dependency>**

**</dependencies>**

...

**Em anexo é fornecido um projeto base *Maven IntelliJ* com um ficheiro *pom.xml* completo e uma classe principal para testar as operações sobre *GCP Storage***