Google Cloud Storage 与 Google Cloud CDN集成注意事项

Status: (Draft)
Authors: chandlerding@google.com,
Last Updated: 2019-12-18

并非所有 HTTP 响应都可以被Cloud CDN缓存。Cloud CDN 只缓存那些满足本部分所列全部要求的响应。其中一些要求在 RFC 7234 中进行了规定，而其他要求则专门针对 Cloud CDN。

只有满足以下所有条件时，您才能将响应存储在 Cloud CDN 缓存中：

- 该响应由启用了 Cloud CDN 的后端服务或后端存储分区提供。
- 该响应是对 `GET` 请求作出的响应。
- 该响应的状态代码是 `200`、`203`、`206`、`300`、`301`、`302`、`307` 或 `410`。
- **该响应具有 `Cache-Control: public` 标头。**
- 该响应具有 `Cache-Control: s-maxage`、`Cache-Control: max-age` 或 `Expires` 标头。
- 该响应具有 `Content-Length`、`Content-Range` 或 `Transfer-Encoding` 标头。
- 该响应的大小小于或等于大小上限。

    - 如果源服务器支持字节范围Range请求，则为 5 TB
    - 如果源服务器不支持字节范围Range请求，则为 10 MB

对于GCS存储桶，您可以通过以下几种方法来满足这些要求：

- 将单独的对象设为可公开读取。
- 将存储桶设为可公开读取。

设置存储桶公开访问的方法

创建一个新的存储桶

```
$ gsutil mb -l us-central1 gs://bucket
```

将存储桶中新对象的默认访问策略设置为允许匿名访问

```
$ gsutil defacl ch -u AllUsers:R gs://bucket
```

此后上传到存储桶的的新对象默认Cache-Control元数据为 Cache-Control : public, max-age=3600

注意：

对于设置了“Bucket Policy Only”的存储桶，即使设置了公开访问，桶中对象的默认Cache-Control元数据为 Cache-Control : private, max-age=0，无法被Cloud CDN缓存。此时应参考后续章节内容，为每个Object单独设置Cache-Control元数据以便Cloud CDN缓存该对象。或者在90天内关闭“Bucket Policy Only”功能。

设置自定义错误响应

```
$ touch index.html
$ gsutil cp index.html gs://bucket/
$ gsutil web set -m index.html -e index.html gs://bucket
```

针对每个Object单独设置Cache-Control元数据

使用gsutil 命令行工具

以下命令将本地images目录的所有文件拷贝到GCS并设置Cache-Control元数据

```
gsutil -m -h "Cache-Control:public,max-age=2678400" cp -r localdir \
gs://bucket/
```

以下命令将递归设置存储桶内所有对象的Cache-Control元数据

```
gsutil -m setmeta -h "Cache-Control:public,s-maxage=2678400" -r
gs://bucket/
```

详见
https://cloud.google.com/storage/docs/gsutil/addlhelp/WorkingWithObjectMetadata
https://cloud.google.com/storage/docs/gsutil/commands/setmeta

使用Cloud Storage Java SDK

参考
https://raw.githubusercontent.com/googleapis/google-cloud-java/master/google-cloud-examples/src/main/java/com/google/cloud/examples/storage/StorageExample.java

使用Java SDK上传文件到GCS存储桶时设置Cache-Control元数据

```
Storage storage = StorageOptions.getDefaultInstance().getService();

BlobId blobId = BlobId.of("bucket", "blob_name");

BlobInfo blobInfo =
BlobInfo.newBuilder(blobId).setContentType("text/plain").setCacheCobtrol("
public,s-maxage=3600").build();

Blob blob = storage.create(blobInfo, "Hello, Cloud
Storage!".getBytes(UTF_8));
```

使用Java SDK更新已有对象的Cache-Control元数据

```java
Storage storage = StorageOptions.getDefaultInstance().getService();

BlobId blobId = BlobId.of("bucket", "blob_name");

Blob blob = storage.get(BlobId);

blob.toBuilder().setCacheControl("public,s-maxage=3600").build().update();
```

使用GCS JSON API

You can send upload requests to the JSON API in any of the following ways:

- Simple upload: `uploadType=media`. Use this if both of the following apply:
  - The file is small enough to upload again in its entirety if the connection fails.
  - There is no object metadata to send as part of the request. To perform a simple upload, refer to the JSON API tab in the Uploading objects guide.
- Multipart upload: `uploadType=multipart`. Use this if both of the following apply:
  - The file is small enough to upload again in its entirety if the connection fails.
  - You want to include object metadata as part of the request. To perform a multipart upload, refer to Performing a multipart upload.
- Resumable upload: `uploadType=resumable`. Use this for more reliable transfer, which is especially important with large files. Resumable uploads are a good choice for most applications, since they also work for small files at the cost of one additional HTTP request per upload. To perform a resumable upload, refer to Performing a resumable upload.

方法一: 使用GCS JSON API执行resumable upload

初始化上传会话时

```
POST
https://www.googleapis.com/upload/storage/v1/b/myBucket/o?uploadType=resumable HTTP/1.1
Authorization: Bearer [YOUR_AUTH_TOKEN]
Content-Length: 66
Content-Type: application/json; charset=UTF-8
X-Upload-Content-Type: image/jpeg
X-Upload-Content-Length: 2000000

{
  "name": "myObject",
  "cacheControl": "public,s-maxage=3600"
}
```

方法二： 使用GCS JSON API执行multipart upload

The following example shows a multipart upload request to a bucket named `myBucket`:

```
POST
https://www.googleapis.com/upload/storage/v1/b/myBucket/o?uploadType=multipart HTTP/1.1
Authorization: Bearer [YOUR_AUTH_TOKEN]
Content-Type: multipart/related; boundary=foo_bar_baz
Content-Length: [NUMBER_OF_BYTES_IN_ENTIRE_REQUEST_BODY]

--foo_bar_baz
Content-Type: application/json; charset=UTF-8

{
  "name": "myObject",
  "cacheControl": "public,s-maxage=3600"
}

--foo_bar_baz
Content-Type: image/jpeg

[JPEG_DATA]

--foo_bar_baz--
```

方法三： 使用GCS JSON API执simple upload

Simple upload不支持在上传时指定对象cache-control元数据，需要在上传完成后再更新cache-control元数据

参考
https://cloud.google.com/storage/docs/viewing-editing-metadata#set-object-metadata-json

```
PATCH
https://www.googleapis.com/storage/v1/b/[BUCKET_NAME]/o/[OBJECT_NAME]
HTTP/1.1
Authorization: Bearer [YOUR_AUTH_TOKEN]
Content-Type: application/json
Content-Length: [NUMBER_OF_BYTES_IN_REQUEST_BODY]

{
  "cacheControl": "public,s-maxage=3600"
}
```