Storage Products (https://cloud.google.com/products/storage/)
Documentation (https://cloud.google.com/storage/docs/) Reference

Cloud Storage Client Libraries

This page shows how to get started with the Cloud Client Libraries for the Google Cloud
Storage API. Read more about the client libraries for Cloud APIs, including the older Google APIs
Client Libraries, in Client Libraries Explained
 (https://cloud.google.com/apis/docs/client-libraries-explained).

Installing the client library

C++	C#	GO	JAVA	MORE ▾

See the GitHub README (https://github.com/googleapis/google-cloud-cpp/blob/master/README.md) for
details about this client library's requirements and install dependencies.

Setting up authentication

To run the client library, you must first set up authentication by creating a service account and
setting an environment variable. Complete the following steps to set up authentication. For
other ways to authenticate, see the GCP authentication documentation
 (https://cloud.google.com/docs/authentication/production).

CLOUD CONSOLE   COMMAND LINE

1. In the Cloud Console, go to the **Create service account key** page.

   GO TO THE CREATE SERVICE ACCOUNT KEY PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/APIS

2. From the **Service account** list, select **New service account**.

3. In the **Service account name** field, enter a name.

4. From the **Role** list, select **Project** > **Owner**.
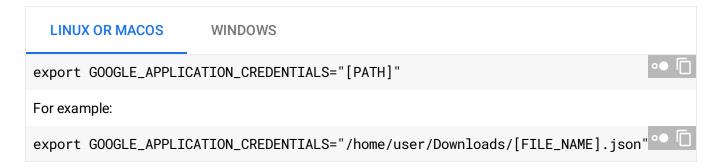
   ★ **Note**: The **Role** field authorizes your service account to access resources. You can view and change
   this field later by using the Cloud Console (https://console.cloud.google.com/). If you are developing

> a production app, specify more granular permissions than **Project > Owner**. For more information,
> see granting roles to service accounts
> (https://cloud.google.com/iam/docs/granting-roles-to-service-accounts).

5. Click **Create**. A JSON file that contains your key downloads to your computer.

Provide authentication credentials to your application code by setting the environment variable *GOOGLE_APPLICATION_CREDENTIALS*. Replace *[PATH]* with the file path of the JSON file that contains your service account key, and *[FILE_NAME]* with the filename. This variable only applies to your current shell session, so if you open a new session, set the variable again.

LINUX OR MACOS   **WINDOWS**

```
export GOOGLE_APPLICATION_CREDENTIALS="[PATH]"
```

For example:

```
export GOOGLE_APPLICATION_CREDENTIALS="/home/user/Downloads/[FILE_NAME].json"
```

Using the client library

The following example shows how to use the client library.

C++   **C#**   **GO**   **JAVA**   **MORE** ▾

OGLE-CLOUD-CPP/BLOB/MASTER/GOOGLE/CLOUD/STORAGE/EXAMPLES/STORAGE_QUICKSTART.CC)

FEEDBACK (#)

```cpp
#include "google/cloud/storage/client.h"
#include <iostream>

int main(int argc, char* argv[]) {
  if (argc != 3) {
    std::cerr << "Missing project id and/or bucket name.\n";
    std::cerr << "Usage: storage_quickstart <bucket-name> <project-id>\n";
    return 1;
  }
```

```cpp
  std::string bucket_name = argv[1];
  std::string project_id = argv[2];

  // Create aliases to make the code easier to read.
  namespace gcs = google::cloud::storage;

  // Create a client to communicate with Google Cloud Storage. This client
  // uses the default configuration for authentication and project id.
  google::cloud::StatusOr<gcs::Client> client =
      gcs::Client::CreateDefaultClient();
  if (!client) {
    std::cerr << "Failed to create Storage Client, status=" << client.status()
              << "\n";
    return 1;
  }

  google::cloud::StatusOr<gcs::BucketMetadata> bucket_metadata =
      client->CreateBucketForProject(bucket_name, project_id,
                                     gcs::BucketMetadata());

  if (!bucket_metadata) {
    std::cerr << "Error creating bucket " << bucket_name
              << ", status=" << bucket_metadata.status() << "\n";
    return 1;
  }

  std::cout << "Created bucket " << bucket_metadata->name() << "\n";

  return 0;
}
```

Note: These samples create a bucket in the default <u>us multi-region</u>
(https://cloud.google.com/storage/docs/locations) with a default storage class of `Standard Storage`
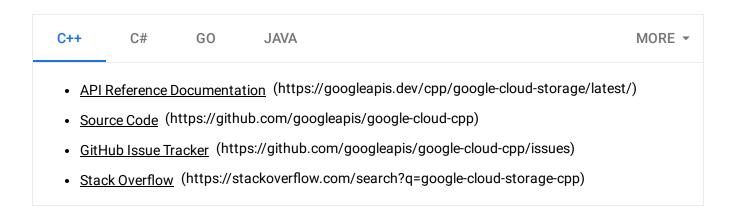(https://cloud.google.com/storage/docs/storage-classes#standard).

More Examples

For more examples of using client libraries with Cloud Storage, see the following how-to guides:

- Creating Storage Buckets
  (https://cloud.google.com/storage/docs/creating-buckets#storage-create-bucket-code_samples)

- Uploading Objects
  (https://cloud.google.com/storage/docs/uploading-objects#storage-upload-object-code-sample)

- Downloading Objects
  (https://cloud.google.com/storage/docs/downloading-objects#storage-download-object-code-sample)

- List of additional guides (https://cloud.google.com/storage/docs/how-to)

Additional resources

C++	C#	GO	JAVA	MORE ▾

- API Reference Documentation (https://googleapis.dev/cpp/google-cloud-storage/latest/)
- Source Code (https://github.com/googleapis/google-cloud-cpp)
- GitHub Issue Tracker (https://github.com/googleapis/google-cloud-cpp/issues)
- Stack Overflow (https://stackoverflow.com/search?q=google-cloud-storage-cpp)

Last updated January 8, 2020.