**EBook Gratis**

# APRENDIZAJE

# google-cloud-platform

Free unaffiliated eBook created from **Stack Overflow contributors.**

#google-cloud-platform

# Tabla de contenido

# Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: google-cloud-platform

It is an unofficial and free google-cloud-platform ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official google-cloud-platform.



Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

# Capítulo 1: Empezando con google-cloud-platform

## Observaciones

Esta sección proporciona una descripción general de qué es la plataforma de nube de Google y por qué un desarrollador puede querer usarla.

También debe mencionar cualquier tema importante dentro de la plataforma de nube de Google y vincular a los temas relacionados. Dado que la Documentación para la plataforma en la nube de Google es nueva, es posible que deba crear versiones iniciales de esos temas relacionados.

## Examples

### Instalación o configuración

Instrucciones detalladas para configurar o instalar google-cloud-platform.

Lea Empezando con google-cloud-platform en línea: https://riptutorial.com/es/google-cloud-platform/topic/6864/empezando-con-google-cloud-platform

# Capítulo 2: ¿Conectar Google Cloud SQL con aplicaciones y herramientas?

## Examples

### ¿Cómo conectar Google Cloud SQL con aplicaciones (como Google App Engine) y herramientas como (mySQL workbench)?

En este documento veremos cómo crear una instancia de Google Cloud SQL y conectarlas en su aplicación Google App Engine y en la herramienta de administración MySQL Workbench.

**Google Cloud SQL:**

Google Cloud SQL es un servicio de base de datos totalmente administrado que facilita la configuración, el mantenimiento, la administración y la administración de sus bases de datos MySQL relacionales en la nube.

Google Cloud SQL proporciona una base de datos relacional que puede utilizar con su aplicación App Engine. Cloud SQL es una base de datos MySQL que vive en la nube de Google.

referir:

> https://cloud.google.com/sql/
>
> https://cloud.google.com/sql/docs/

**Creación de instancias de SQL:**

Una instancia de Google Cloud SQL es una base de datos MySQL alojada en la nube de Google.

1. Vaya a la página de Instancias de SQL Cloud en la Consola de Google Cloud Platform ( https://console.cloud.google.com/sql/instances) y haga clic en Crear instancia.
2. Haga clic en Elegir primera generación, ingrese un nombre y elija un nivel para la instancia y haga clic en Crear.
3. Una vez que la instancia termine de inicializarse, seleccione la instancia para abrirla.
4. En Control de acceso> Usuarios, haga clic en Crear cuenta de usuario y cree un usuario con el nombre de raíz y especifique una contraseña (root_password). Esto crea el usuario de MySQL 'root' @ '%'.
5. En Bases de datos, haga clic en Nueva base de datos y cree una base de datos con un nombre de base de datos (nombre de base de datos)

**MySQL Workbench:**

MySQL Workbench es una herramienta visual unificada para arquitectos de bases de datos, desarrolladores y administradores de bases de datos. MySQL Workbench proporciona modelado de datos, desarrollo de SQL y herramientas de administración integrales para la configuración del servidor, la administración de usuarios, las copias de respaldo y mucho más.

consulte http://www.mysql.com/products/workbench/

Ahora veremos cómo conectarse a su base de datos de instancia de Google Cloud SQL con MySQL Workbench.

***Configurando el acceso***

1. Vaya a la página de Instancias de SQL Cloud en la Consola de Google Cloud Platform y seleccione la instancia.

2. En Control de acceso> Dirección IP, haga clic en Solicitar dirección IPv4 y cópielo (Dirección_IPv4_instancia). Es necesario para conectar su base de datos de instancia de Google Cloud SQL con herramientas de administración como MySQL Workbench.

   *nota:* se le cobrará por la dirección IPv4 a $ 0.01 cada hora que la instancia esté inactiva y $ 0.1 cada hora que la instancia esté activa

1. La 'dirección IP' de Google para encontrar su dirección IP pública

2. En Control de acceso> Autorización> Redes autorizadas, haga clic en Agregar red e ingrese su dirección IP.

3. En Control de acceso> Usuarios, cree un usuario con nombre de usuario (nombre de usuario), contraseña (contraseña) y la opción 'Permitir cualquier host seleccionado'. Se recomienda usar una cuenta de usuario separada para acceder desde WorkBench

***Conectando***

1. En la vista de inicio de MySQL Workbench, haga clic en Nueva conexión.

1. En la ventana Configurar nueva conexión, proporcione un nombre de conexión, nombre de host y nombre de usuario

1. Haga clic en Conexión de prueba. Se te solicitará una contraseña.

1. Una vez que la conexión MySQL se haya realizado correctamente, haga clic en Aceptar y haga clic en la conexión guardada para abrir el Editor SQL

**Motor de aplicaciones de Google:**

Google App Engine es una plataforma para crear aplicaciones web escalables y backends móviles. App Engine escalará tu aplicación automáticamente.

consulte https://cloud.google.com/appengine

Ahora veremos cómo configurar una conexión entre una aplicación de App Engine y una instancia

de Cloud SQL.

***Configurando el acceso***

1. Vaya a la página de Instancias de SQL Cloud en la Consola de Google Cloud Platform y seleccione la instancia.

1. En Control de acceso> Autorización> Aplicaciones de App Engine autorizadas, haga clic en Agregar ID de aplicación e ingrese la ID de la aplicación. Haga clic en Listo y Guardar.

1. En Descripción general> Propiedades Copie el 'Nombre de conexión de instancia' (nombre_de_Conexión_instancia)

1. En su proyecto de aplicación web de Google, war / WEB-INF / appengine-web.xml add, true </use-google-connector-j>

***Ejemplo de código:***

Un ejemplo para Google App Engine - Java Standard Environment

```
public static Connection connect() throws ClassNotFoundException, SQLException {
    String url = null;

    {
        if (SystemProperty.environment.value() == SystemProperty.Environment.Value.Production)
{

            // Connecting from App Engine.
            Class.forName(Messages.getString("com.mysql.jdbc.GoogleDriver"));
            url =
Messages.getString("jdbc:google:mysql://{{Instance_Connection_Name}}/{{DataBase_Name}}?user=root&passwo

        } else {
            // Connecting from an external network or localhost
            Class.forName(Messages.getString("com.mysql.jdbc.Driver"));
            url =
Messages.getString("jdbc:mysql://{{Instance_IPv4_address}}:3306/{{DataBase_Name}}?user={{userName}}&pas

        }

        Connection conn = DriverManager.getConnection(url);

        return conn;
    }
}
```

Lea ¿Conectar Google Cloud SQL con aplicaciones y herramientas? en línea: https://riptutorial.com/es/google-cloud-platform/topic/10772/-conectar-google-cloud-sql-con-aplicaciones-y-herramientas-

# Capítulo 3: Motor de aplicaciones de Google

## Introducción

Google App Engine (GAE) es una oferta de Plataforma como Servicio (PaaS) en Google Cloud Platform, que retira la infraestructura para que usted se centre en el código de su aplicación web. App Engine maneja tanto la ampliación automática como la reducción de instancias a pedido para su aplicación web en función del número de solicitudes.

App Engine está disponible en 2 tipos de entornos: Estándar y Flexible y admite los siguientes lenguajes de programación: Go, Java, Python, PHP, Node.JS, Ruby y .NET.

## Examples

### app.yaml para la aplicación Php en un entorno flexible

```
runtime: php
vm: true
api_version: 1

runtime_config:
  document_root: web
```

### Instalando gcloud cli

Así es como instala la herramienta de línea de comandos `gcloud` , que forma parte del Google Cloud SDK.

```
$ curl https://sdk.cloud.google.com | bash
```

### Iniciar sesión e inicializar

Autorízate, serás navegado a la página de inicio de sesión de Google.

```
$ gcloud auth login
```

Inicialice o reinicialice `gcloud` , puede configurar sus configuraciones aquí.

```
$ gcloud init
```

Muestra información sobre el entorno actual de `gcloud` .

```
$ gcloud info
```

Muestra la lista de configuraciones de SDK activas.

```
$ gcloud config list
```

Muestre la lista de cuentas en las que se almacenan las credenciales en el sistema local.

```
$ gcloud auth list
```

Muestra las opciones de ayuda.

```
$ gcloud help
```

## Conectar a la instancia de Cloud SQL usando Php

Crear una instancia de Cloud SQL

```
$dsn = "/cloudsql/PROJECT:REGION:INSTANCE;dbname=DATABASE";
$user = "USER";
$password = "PASSWORD";
$db = new PDO($dsn, $user, $password); //Whatever is your favorite MySQL connection method
```

La parte importante aquí es `/cloudsql/PROJECT:REGION:INSTANCE;` . Aquí nos estamos conectando a la instancia de Cloud SQL a través de un socket Unix. Puede encontrar esto en propiedades de instancia como `Instance connection name` .

## Escribir en el almacenamiento en la nube

App Engine no permite escribir en archivos, en lugar de eso, escribe y lee desde Cloud Storage.

Escribir en el almacenamiento en la nube

```
$file = 'gs://<your-bucket>/hello.txt';
file_put_contents($file, 'hello world');
```

Leer desde el almacenamiento en la nube

```
$contents = file_get_contents($file);
var_dump($contents);
```

## Registro y monitoreo de registro

Simplemente use la función de syslog estándar de PHP para escribir registros

```
syslog(LOG_INFO, "Authorized access");
syslog(LOG_WARNING, "Unauthorized access");
```

Puede ver los registros del "Registro del controlador de pila" ( https://console.cloud.google.com/logs )

Lea Motor de aplicaciones de Google en línea: https://riptutorial.com/es/google-cloud-

platform/topic/9944/motor-de-aplicaciones-de-google

# Capítulo 4: Nuevo proyecto con el cliente API de Cloud Resource Manager para .NET

## Introducción

Utilizaremos las bibliotecas de cliente API de Google para .NET para esta muestra.

Hay otras bibliotecas. Por favor, consulte la Explicación de las bibliotecas cliente de Google para más detalles.

Utilizaremos la API de Cloud Resource Manager para crear y administrar proyectos .

Empecemos.

## Observaciones

Poniendolo todo junto...

Deberías tener dos archivos. El primer archivo se llama `packages.config` o `project.json` .

Vamos a nombrar el segundo archivo `Program.cs` . Todo el código que se incluyó en las secciones anteriores se puede pegar en un solo método principal:

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

using Google.Apis.Auth.OAuth2;
using Google.Apis.CloudResourceManager.v1;
using Google.Apis.Services;

using Data = Google.Apis.CloudResourceManager.v1.Data;

namespace OurFirstProject
{
    public class Program
    {
        private const string projectId = [YOUR-PROJECT-ID];
        private const string applicationName = "Test";

        public static void Main(string[] args)
        {

            var scopes = new String[] {
                CloudResourceManagerService.Scope.CloudPlatform
            };

            GoogleCredential credential = Task.Run(
                () => GoogleCredential.GetApplicationDefaultAsync()
            ).Result;
```

```
            if (credential.IsCreateScopedRequired)
            {
                credential = credential.CreateScoped(scopes);
            }

            CloudResourceManagerService service = new CloudResourceManagerService(
                new BaseClientService.Initializer()
                {
                    HttpClientInitializer = credential,
                    ApplicationName = applicationName
                }
            );

            Console.WriteLine("1. Create Project");
            Data.Operation operation1 = service.Projects.Create(
                new Data.Project()
                {
                    ProjectId = projectId,
                }
            ).Execute();

            Console.Write("2. Awaiting Operation Completion");
            Data.Operation operation2;
            do
            {
                operation2 = service.Operations.Get(operation1.Name).Execute();
                Console.WriteLine(operation2.Done.ToString());
                System.Threading.Thread.Sleep(1000);
            } while (operation2.Done != true);

            Console.WriteLine();
            Console.WriteLine("Enter to continue");
            Console.ReadLine();

            Console.WriteLine("3. Deleting Project");
            var operation3 = service.Projects.Delete(projectId).Execute();
        }
    }
}
```

Si está utilizando Windows y Visual Studio, "Inicio"

Si está utilizando Linux, primero debe restaurar los paquetes y luego ejecutar la aplicación

```
dotnet restore
dotnet run
```

La salida debe ser similar a:

```
Compiling Projects for .NETCoreApp,Version=v1.1

Compilation succeeded.
    0 Warning(s)
    0 Error(s)

Time elapsed 00:00:01.4161926
```

```
1. Create Project

2. Awaiting Operation Completion



True

Enter to continue

3. Deleting Project
```

El paso "En espera" contendrá líneas en blanco (con suerte) que terminan en "Verdadero".

# Examples

## Empezando

El código funcionará en Windows y Linux.

Se ejecuta en .NET Core o .NET Standard. La única diferencia es que, si desea ejecutar en .NET Core, debe usar `project.json` . Si desea ejecutar en .NET Standard, debe usar `packages.config` .

La API de API Client Library para Cloud Resource Manager está disponible en nuget:

https://www.nuget.org/packages/Google.Apis.CloudResourceManager.v1/

La versión de la escritura es: 1.22.0.809

Lo haremos de 2 maneras:

- Windows con .NET Standard; y
- Linux con .NET Core.

**Windows**

Si está utilizando Visual Studio, cree una nueva "Aplicación de consola" de Visual C # ". De lo contrario, cree un directorio para el proyecto y cree un archivo en él llamado `packages.config` . packages.config es para .NET Standard pero estamos usando .NET Standard con Windows. Solo puedes ejecutar .NET Standard en Windows. Reemplace el contenido de `packages.config` con:

```
<?xml version="1.0" encoding="utf-8"?>
<packages>
  <package id="Google.Apis" version="1.22.0"
    targetFramework="net452" />
  <package id="Google.Apis.Auth" version="1.22.0"
    targetFramework="net452" />
  <package id="Google.Apis.CloudResourceManager.v1" version="1.22.0.809"
    targetFramework="net452" />
  <package id="Google.Apis.Core" version="1.22.0"
    targetFramework="net452" />
</packages>
```

### Linux

Cree un directorio para el proyecto y cree un archivo en él llamado `project.json` . `project.json` es para .NET Core pero en este ejemplo estamos usando .NET Core con Linux. Puede ejecutar .NET Core en Linux o en Windows. Reemplace los contenidos de `project.json` con:

```
{
  "version": "1.0.0-*",
  "buildOptions": {
    "debugType": "portable",
    "emitEntryPoint": true
  },
  "dependencies": {},
  "frameworks": {
    "netcoreapp1.1": {
      "dependencies": {
        "Microsoft.NETCore.App": {
          "type": "platform",
          "version": "1.1.0"
        },
        "Google.Apis.CloudResourceManager.v1":"1.22.0.809"
      },
      "imports": "dnxcore50"
    }
  }
}
```

## Credenciales predeterminadas de la aplicación

No lo repetiré todo aquí: " Las credenciales predeterminadas de la aplicación proporcionan una forma sencilla de obtener credenciales de autorización para usarlas mediante las API de Google".

Si puede usar las credenciales predeterminadas de la aplicación, haga.

Hay un paso adicional que deberá realizar antes de usar las Credenciales predeterminadas de la aplicación como su identidad al llamar a las API desde su máquina:

```
gcloud auth application-default login [yourname@gmail.com]
```

Aquí es por qué preferirá usar las credenciales predeterminadas de la aplicación:

```
var scopes = new String[] {
    CloudResourceManagerService.Scope.CloudPlatform
};

GoogleCredential credential = Task.Run(
    () => GoogleCredential.GetApplicationDefaultAsync()
).Result;

if (credential.IsCreateScopedRequired)
{
    credential = credential.CreateScoped(scopes);
}
```

... ¡Es todo el código que necesita para autorizar llamadas a (cualquier) API de Google Cloud!

Usaremos el objeto de `credential` en el siguiente paso para hacer llamadas contra el servicio de Google ...

## Llamando a cualquier método (!) En cualquier servicio de Google (!)

Una vez que se haya familiarizado con el código para llamar a un método en un servicio de Google, podrá deducir cómo llamar a *cualquier* método en *cualquier* servicio de Google.

Primero, realizamos una conexión al servicio utilizando el objeto de `credential` instanciado en el ejemplo anterior:

```
CloudResourceManagerService service = new CloudResourceManagerService(
    new BaseClientService.Initializer()
    {
        HttpClientInitializer = credential,
        ApplicationName = "Our First Google API Client"
    }
);
```

Entonces podemos llamar métodos proporcionados por el servicio. ¿Qué métodos están disponibles?

https://cloud.google.com/resource-manager/docs/apis

¿Cuál es la solicitud REST subyacente para `Projects.Create` ?

https://cloud.google.com/resource-manager/reference/rest/v1/projects/create

OK ... Vamos a escribir el código.

El código espera un valor de cadena para `projectId` . Los ID de proyecto son identificadores únicos. Te recomiendo que uses un sistema para nombrar tus proyectos para ayudarte a identificarlos.

`Projects.Create` espera un objeto `Data.Project` . Esta propiedad obligatoria del objeto uno, el ID del proyecto que es todo lo que proporcionaremos, pero también podríamos proporcionar un nombre del proyecto, etiquetas, detalles del padre del proyecto, etc.

```
Data.Operation operation1 = service.Projects.Create(
    new Data.Project()
    {
        ProjectId = projectId,
    }
).Execute();
```

La creación del proyecto se maneja de forma asíncrona. Se nos da un objeto de `Operation` que debemos sondear para determinar cuándo se crea el proyecto. Las operaciones tienen una propiedad de nombre que identifica de forma exclusiva la operación. La siguiente sección de código sondea la plataforma "¿Ya terminamos?". El proyecto se creará cuando nuestra nueva

operación incluya una propiedad `Done` que sea `True` .

```
Data.Operation operation2;
do
{
    operation2 = service.Operations.Get(operation1.Name).Execute();
    System.Threading.Thread.Sleep(1000);
} while (operation2.Done != true);
```

Para completar, y es de esperar que dentro de muchos años después de un uso feliz de su proyecto, es posible que deba eliminar su proyecto. Simplemente llamamos a Projects.Delete y proporcione nuestra ID de proyecto. Esto también devuelve una operación y realmente deberíamos encuestar esta operación hasta que se complete definitivamente. Nuestro proyecto será eliminado.

```
var operation3 = service.Projects.Delete(projectId).Execute();
```

¡Eso es!

Lea Nuevo proyecto con el cliente API de Cloud Resource Manager para .NET en línea: https://riptutorial.com/es/google-cloud-platform/topic/9523/nuevo-proyecto-con-el-cliente-api-de-cloud-resource-manager-para--net

# Capítulo 5: Nuevo proyecto con el cliente API de Cloud Resource Manager para Python

## Introducción

Usaremos las bibliotecas de cliente API de Google para Python para esta muestra.

Hay otras bibliotecas. Por favor, consulte la Explicación de las bibliotecas cliente de Google para más detalles.

Utilizaremos la API de Cloud Resource Manager para crear y administrar proyectos .

Empecemos.

## Observaciones

Poniendolo todo junto...

Si siguió los ejemplos anteriores, debería estar en un directorio llamado `my_project_folder` y, con suerte, contiene un subdirectorio llamado `venv` .

Asegúrese de que su virtualenv está activado.

Todo el código se puede colocar en un archivo, llamémoslo `create_project.py` .

Cree el archivo `create_project.py` en `my_project_folder` .

```
import json
import time

from apiclient.discovery import build
from oauth2client.client import GoogleCredentials

SERVICE_NAME = "cloudresourcemanager"
SERVICE_VERSION = "v1"

# Don't forget to replace YOUR-PROJECT-ID with your choice of Project ID
# Even though the code deletes the Project, change this value each time
PROJECT_ID = "YOUR-PROJECT-ID"

credentials = GoogleCredentials.get_application_default()

service = build(
    SERVICE_NAME,
    SERVICE_VERSION,
    credentials=credentials)

operation1 = service.projects().create(
    body={
        "project_id": PROJECT_ID
    }
```

```
).execute()

print(json.dumps(
    operation1,
    sort_keys=True,
    indent=3))

name = operation1["name"]
while True:
    operation2 = service.operations().get(
        name=name
    ).execute()
    print(json.dumps(
        operation2,
        sort_keys=True,
        indent=3))
    if "done" in operation2:
        if (operation2["done"]):
            break
    time.sleep(1)

raw_input("Press Enter to delete the Project...")

operation3 = service.projects().delete(
    projectId=PROJECT_ID
).execute()
```

Guarde el archivo y, desde el símbolo del sistema, escriba:

```
python create_project.py
```

La salida debe ser similar a:

```
{
   "metadata": {
      "@type": "type.googleapis.com/google.cloudresourcemanager.v1.ProjectCreationStatus",
      "createTime": "2017-12-31T00:00:00.000Z"
   },
   "name": "operations/pc.1234567890123456789"
}
...
{
   "done": true,
   "metadata": {
      "@type": "type.googleapis.com/google.cloudresourcemanager.v1.ProjectCreationStatus",
      "createTime": "2017-12-31T00:00:00.000Z"
      "gettable": true,
      "ready": true
   },
   "name": "operations/pc.1234567890123456789",
   "response": {
      "@type": "type.googleapis.com/google.cloudresourcemanager.v1.Project",
      "createTime": "2017-12-31T00:00:00.000Z",
      "lifecycleState": "ACTIVE",
      "projectId": "your-project-id",
      "projectNumber": "123456789012"
   }
}
```

```
...
Press Enter to delete the Project...
```

# Examples

## Credenciales predeterminadas de la aplicación

No lo repetiré todo aquí: " Las credenciales predeterminadas de la aplicación proporcionan una forma sencilla de obtener credenciales de autorización para usarlas mediante las API de Google".

Si puede usar las credenciales predeterminadas de la aplicación, haga.

Hay un paso adicional que deberá realizar antes de usar las Credenciales predeterminadas de la aplicación como su identidad al llamar a las API desde su máquina:

```
gcloud auth application-default login [yourname@gmail.com]
```

Aquí es por qué preferirá usar las credenciales predeterminadas de la aplicación:

```
scopes = [
    "https://www.googleapis.com/auth/cloud-platform"
]
credentials = GoogleCredentials.get_application_default()
if credentials.create_scoped_required():
    credentials = credentials.create_scoped(scopes)
```

En verdad, generalmente puede salirse con la suya solo con:

```
credentials = GoogleCredentials.get_application_default()
```

... ¡Es todo el código que necesita para autorizar llamadas a (cualquier) API de Google Cloud!

Usaremos el objeto de `credential` en el siguiente paso para hacer llamadas contra el servicio de Google ...

## Llamando a cualquier método (!) En cualquier servicio de Google (!)

Una vez que se haya familiarizado con el código para llamar a un método en un servicio de Google, podrá deducir cómo llamar a **cualquier** método en **cualquier** servicio de Google.

Primero, realizamos una conexión al servicio utilizando el objeto de `credential` instanciado en el ejemplo anterior:

```
service = build(
    SERVICE_NAME,
    SERVICE_VERSION,
    credentials=credentials)
```

Entonces, podemos llamar métodos proporcionados por el servicio. ¿Qué métodos están

disponibles?

https://cloud.google.com/resource-manager/docs/apis

¿Cuál es la solicitud REST subyacente para Projects.Create?

https://cloud.google.com/resource-manager/reference/rest/v1/projects/create

OK ... Vamos a escribir el código.

El método de `create` espera un cuerpo que contenga mínimamente la ID del proyecto. Los ID de proyecto son identificadores únicos. Le recomiendo que use un sistema para nombrar sus proyectos para ayudarlo a identificarlos. El método también acepta un nombre de proyecto, etiquetas, detalles de los padres del proyecto, etc.

```
operation1 = service.projects().create(
  body={
    "project_id": PROJECT_ID
  }
).execute()
```

La creación del proyecto se maneja de forma asíncrona. Se nos da un objeto de operación que debemos sondear para determinar cuándo se crea el proyecto. Las operaciones tienen una propiedad de nombre que identifica de forma exclusiva la operación. La siguiente sección de código sondea la plataforma "¿Ya terminamos?". El proyecto se creará cuando nuestra nueva operación incluya una propiedad `Done` que sea `True` .

```
name = operation1["name"]
while True:
    operation2 = service.operations().get(
        name=name
    ).execute()
    if "done" in operation2:
        if (operation2["done"]):
            break
    time.sleep(1)
```

Para completar, y es de esperar que dentro de muchos años después de un uso feliz de su proyecto, es posible que deba eliminar su proyecto. Simplemente llamamos al método de eliminación y proporcionamos nuestra ID de proyecto. Esto también devuelve una operación, pero te lo dejo para que encuestes la operación hasta que se complete.

```
operation3 = service.projects().delete(
    projectId=PROJECT_ID
).execute()
```

¡Eso es!

## Empezando

Necesitarás poder ejecutar Python.

Python está disponible para Linux, Mac OS X y Windows.

Recomiendo pip y virtualenv .

Pip es la herramienta recomendada para instalar paquetes de Python.

Las bibliotecas API de Google Cloud están disponibles como paquetes pip.

Un entorno virtual (también conocido como "virtualenv") es una "herramienta para mantener las dependencias requeridas por diferentes proyectos en lugares separados".

Crea un directorio para tu proyecto y luego:

```
cd my_project_folder
virtualenv venv
```

Puede asignar un nombre a su carpeta virtualenv que no sea "venv", pero este nombre es un recordatorio útil del propósito del directorio. virtualenv debería mostrar algo similar a:

```
New python executable in venv/bin/python
Installing setuptools, pip, wheel...done.
```

Se crea el virtualenv pero, para usarlo, debemos `activate` . Cuando hayamos terminado de usarlo, es una buena práctica (aunque no es obligatorio) `deactivate` también.

```
source venv/bin/activate
```

Para indicar que estamos en el virtualenv llamado venv, cambia el indicador de comando en mi máquina Linux. Usted permanecerá en el directorio de su proyecto. Su kilometraje puede variar, pero:

```
(venv) user@host
```

Ahora vamos a actualizar pip e instalar las bibliotecas de cliente API de Google

```
pip install --upgrade pip
pip install --upgrade google-api-python-client
```

Todo está bien si escribe `pip freeze` , debería ver algo similar a esta lista. Tus versiones pueden ser más altas:

```
pip freeze
google-api-python-client==1.6.2
httplib2==0.10.3
oauth2client==4.0.0
pyasn1==0.2.3
pyasn1-modules==0.0.8
rsa==3.4.2
six==1.10.0
uritemplate==3.0.0
```

Su entorno de Python ya está listo, las bibliotecas de cliente API de Google están instaladas. Podemos escribir nuestro código Python. Por favor continúe con el siguiente ejemplo.

Cuando haya terminado con el proyecto, es una buena práctica `deactivate` el virtualenv. Por favor no hagas esto ahora ya que vamos a escribir un código. Pero, cuando hayas terminado, vuelve aquí y:

```
deactivate
```

Si desea volver a la virtualenv, simplemente vuelva a ejecutar el comando de `activate`

Lea Nuevo proyecto con el cliente API de Cloud Resource Manager para Python en línea: https://riptutorial.com/es/google-cloud-platform/topic/9536/nuevo-proyecto-con-el-cliente-api-de-cloud-resource-manager-para-python

# Creditos

| S. No | Capítulos | Contributors |
|---|---|---|
| 1 | Empezando con google-cloud-platform | Community |
| 2 | ¿Conectar Google Cloud SQL con aplicaciones y herramientas? | Newton Joshua |
| 3 | Motor de aplicaciones de Google | Milindu Sanoj Kumarage, Tuxdude |
| 4 | Nuevo proyecto con el cliente API de Cloud Resource Manager para .NET | DazWilkin |
| 5 | Nuevo proyecto con el cliente API de Cloud Resource Manager para Python | DazWilkin |