



Google Cloud Software Development Kit (SDK) for .NET

Author: Luis Coco Enríquez

Agenda

Topic

What is Google Cloud SDK for .NET?.

Most common use cases.

How to set up your environment.

API Reference.

Agenda

Topic

Code examples with guidance for the Google Cloud SDK for .NET.

Additional code examples for the Google Cloud SDK for .NET.

GitHub repositories.

C# code samples with Google Cloud SDK for .NET.

Google Cloud Platform .NET Cloud Client Libraries

How to create a Google Cloud Free Tier account

1. Create a gmail account. <https://support.google.com/mail/answer/56256?hl=en>
2. Set your payment info (credit card data, paypal,... or other).
3. Open an incognito Chrome window and login in Google Cloud.
4. Login in Google Cloud. <https://console.cloud.google.com/>
5. Activate the free tier subscription.
6. First sample: create a new Bucket (for storing data).

Google Cloud Platform .NET Cloud Client Libraries

Setting up a .NET development environment

1. Install your IDE: Visual Studio or VS Code. <https://visualstudio.microsoft.com/https://code.visualstudio.com/download>
2. Create a Google Cloud project. Create a Google Cloud project to run your apps. Google Cloud projects form the basis for creating, enabling, and using all Google Cloud services.
3. Authentication. Your .NET app must authenticate itself to use Google Cloud APIs. You use Application Default Credentials (ADC), which let you provide credentials for either local development or in a production environment.

For information about setting up ADC, see [Provide credentials to Application Default Credentials](#). For general information about authentication, see [Authentication at Google](#).

<https://cloud.google.com/dotnet/docs/setup?hl=en>

Google Cloud Platform .NET Cloud Client Libraries

Set up Application Default Credentials (ADC)

<https://cloud.google.com/docs/authentication/provide-credentials-adc#local-dev>

1. Install “gcloud CLI” <https://cloud.google.com/sdk/docs/install?hl=en>
2. Create your credential file (run the command):

gcloud auth application-default login

This method stores your credentials in a file on your file system. Any user with access to your file system can use those credentials. When you no longer need these credentials, you should revoke them:

gcloud auth application-default revoke

User credentials might not work for some methods and APIs, such as the *Cloud Translation API* or the *Cloud Vision API*, without extra parameters or configuration. See [Troubleshooting your ADC setup](#).

<https://cloud.google.com/dotnet/docs/setup?hl=en>

Google Cloud Code for VSCode

The screenshot shows the Visual Studio Code interface with the Google Cloud Code extension page open. The left sidebar displays the 'EXTENSIONS: MARKETPLACE' view with a search for 'google'. The main panel shows the 'Google Cloud Code' extension details, including its version (v1.21.8), download count (775,937), and a 4.5-star rating. The extension is described as 'Google Cloud's official IDE extension to develop with your favorite Goog...'. The 'DETAILS' tab is active, showing a description of the extension's capabilities and a list of categories (Snippets, Linters, Debuggers, Other). The 'Extension Resources' section includes links to the Marketplace, Repository, License, and Google Cloud. The 'More Info' section shows the publication date (4/8/2019) and the last update (6/27/2023). The bottom status bar indicates the current file is 'Cloud Code: Installing Cloud SDK...'.

File Edit Selection View Go Run ... Search

EXTENSIONS: MARKETPLACE

google

Google Cloud Code v1.21.8
Google Cloud [google.com](#) | 775,937 | ★★★★★ (35)
Cloud Code is Google Cloud's official IDE extension to develop with your favorite Goog...
[Disable](#) [Uninstall](#) [Settings](#)
This extension is enabled globally.

DETAILS FEATURE CONTRIBUTIONS CHANGELOG RUNTIME STATUS

project as you build, edit, run, and deploy your applications locally or in the cloud. Cloud Code also deeply integrates with [Cloud SDK](#) to provide a unified authentication experience when you develop with Google Cloud Services.

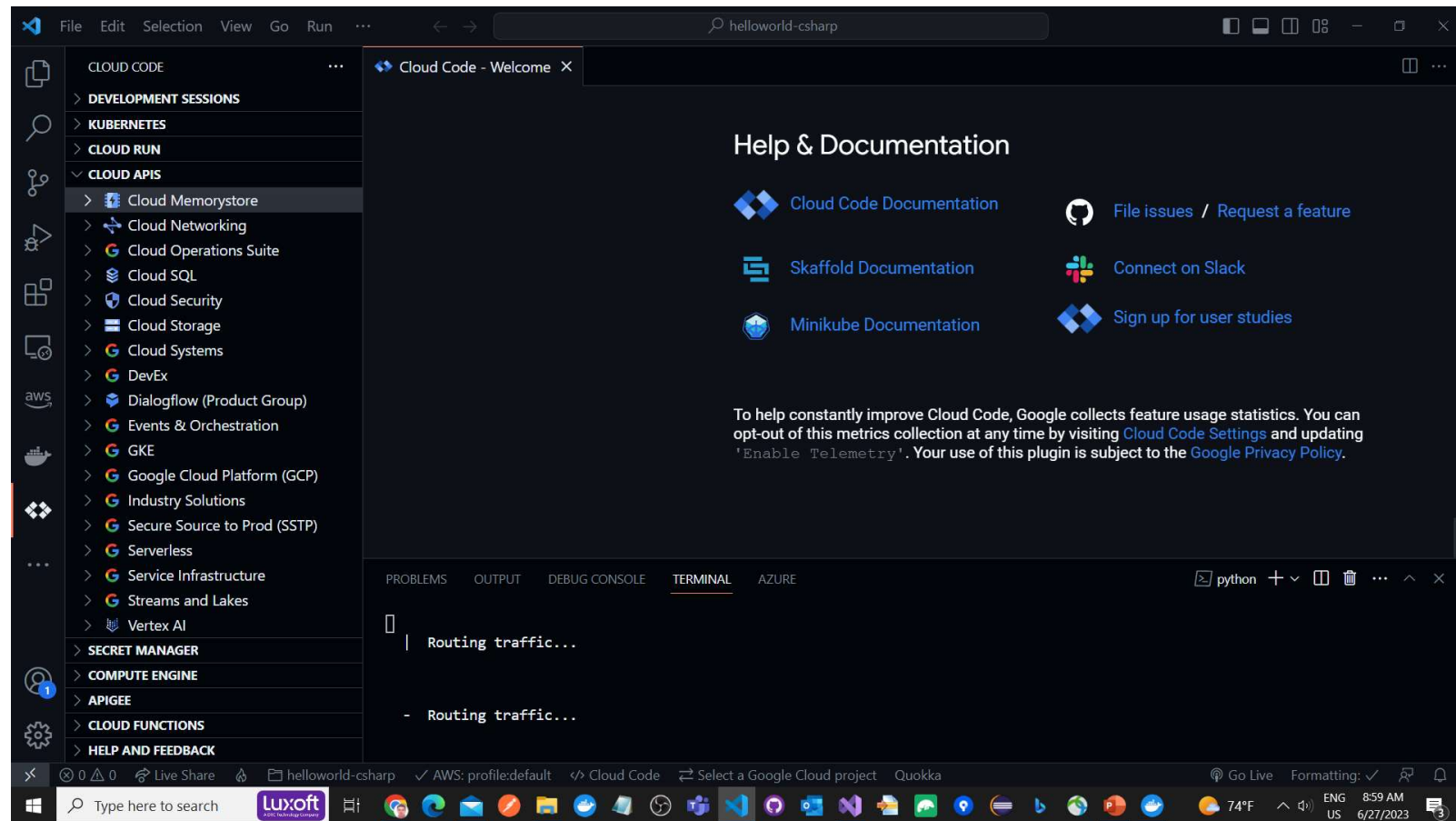
Categories
Snippets Linters
Debuggers Other

Extension Resources
[Marketplace](#)
[Repository](#)
[License](#)
[Google Cloud](#)

More Info
Published 4/8/2019, 17:05:09
Last 6/27/2023, 02:26:04

Cloud Code: Installing Cloud SDK...

Google Cloud Code for VSCode



Google Cloud Code for Visual Studio 2022

The screenshot shows a web browser window displaying the Google Cloud Code for Visual Studio 2022 Codelab. The browser's address bar shows the URL codelabs.developers.google.com/codelabs/cloud-visual-studio#0. The page title is "Install and use Cloud Tools for Visual Studio". On the left, there is a sidebar with a list of steps: 1. Overview, 2. Setup and Requirements, 3. Installation, 4. Authentication, 5. Google Cloud Explorer, 6. Create and Manage Google Compute Engine instances, 7. Create and Deploy ASP.NET apps, 8. Browse Google Cloud Storage, and 9. Congratulations!. The main content area is titled "Install and use Cloud Tools for Visual Studio". It includes a section "Acerca de este codelab" (About this codelab) with the text "Última actualización: ene 7, 2021" (Last update: Jan 7, 2021) and "Escrito por Mete Atamel" (Written by Mete Atamel). Below this is the "1. Overview" section, which states: "Visual Studio is the go-to IDE for Windows and .NET developers. Cloud Tools for Visual Studio is a plugin to build and test Windows and .NET applications locally and deploy them to Google Cloud Platform directly from Visual Studio. Google Cloud Explorer lets you easily browse Google Cloud projects and resources across all of your accounts in Visual Studio. In this codelab, you will learn how to install and use Cloud Tools for Visual Studio and how to interact with Google Cloud resources from Visual Studio directly." At the bottom right of the main content area, there is a blue button labeled "Siguiente" (Next). The Windows taskbar is visible at the bottom of the screen, showing various application icons and the system clock indicating 9:05 AM on 6/27/2023.

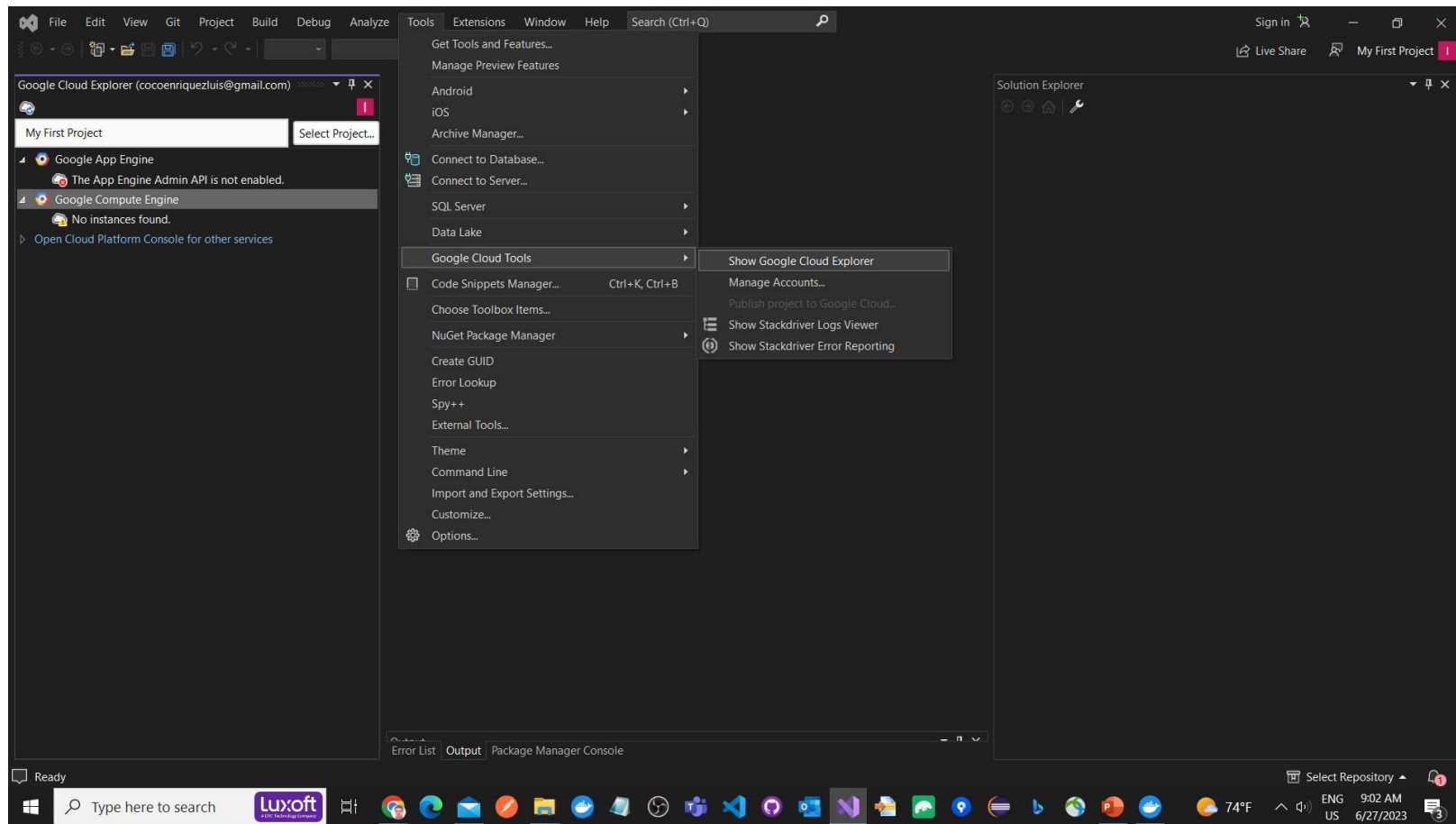
<https://codelabs.developers.google.com/codelabs/cloud-visual-studio#0>

Google Cloud Code for Visual Studio 2022

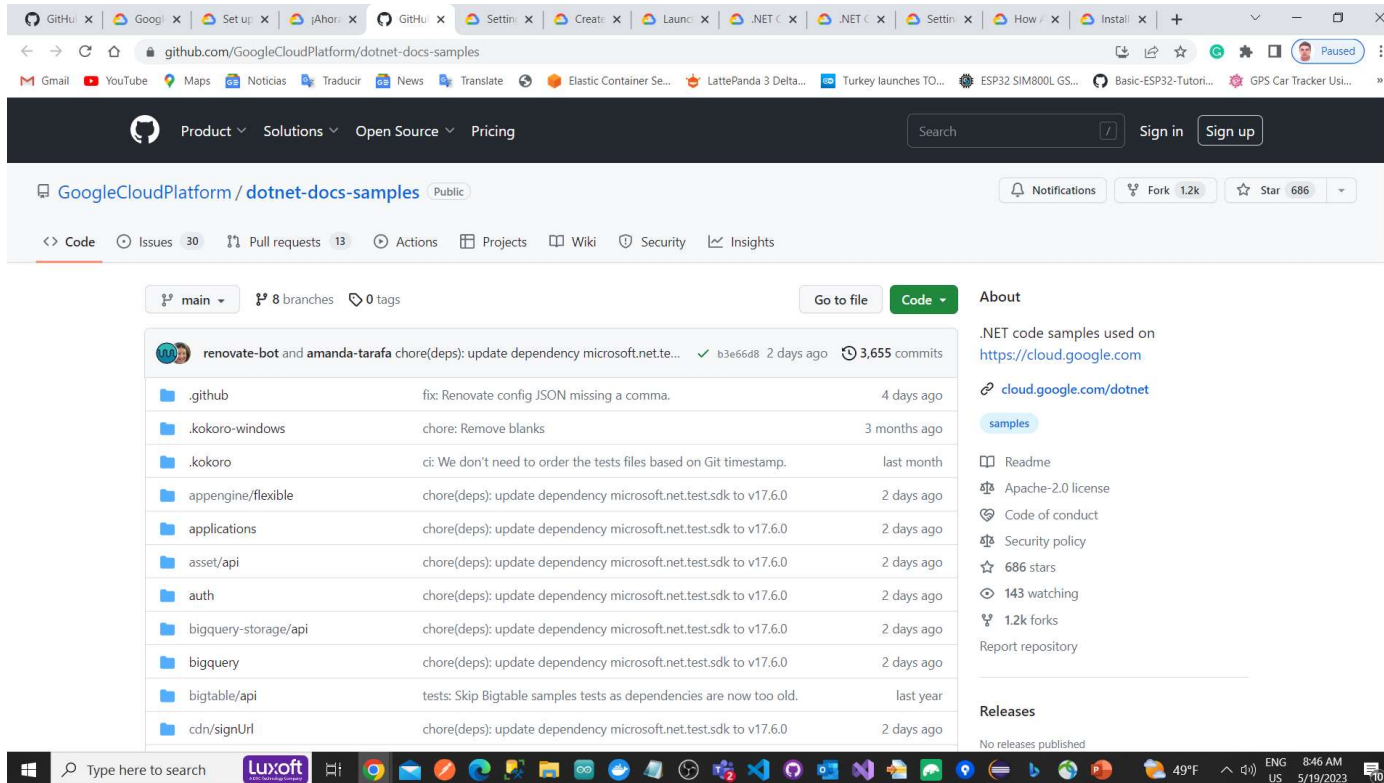
The screenshot shows the Visual Studio Marketplace page for the 'Google Cloud Tools for Visual Studio' extension. The browser's address bar shows the URL: marketplace.visualstudio.com/items?itemName=GoogleCloudTools.GoogleCloudPlatformExtensionforVisualStudio. The page header includes the Visual Studio logo and 'Marketplace' text. Below the header, the extension's name 'Google Cloud Tools for Visual Studio' is displayed, along with the Google Cloud logo, a link to 'google.com', and statistics: '63,437 installs', a star rating of 4.5 (17 reviews), and 'Free'. A green 'Download' button is prominent. The page is divided into sections: 'Overview', 'Q & A', and 'Rating & Review'. The 'Overview' section contains several paragraphs describing the extension's capabilities, such as deploying applications to Google Cloud Platform, adding remote debugging support, and providing project templates. On the right side, there are sections for 'Categories' (Tools, Build, Web, Services), 'Tags' (ASP.NET, Cloud, compute, Deploy, gce, gcp, gcs, Google, Storage, tool), 'Works with' (Visual Studio 2017, 2019), and 'Resources' (License, Copy ID). The Windows taskbar at the bottom shows various application icons and the system clock indicating 9:07 AM on 6/27/2023.

<https://marketplace.visualstudio.com/items?itemName=GoogleCloudTools.GoogleCloudPlatformExtensionforVisualStudio>

Google Cloud Code for Visual Studio 2022

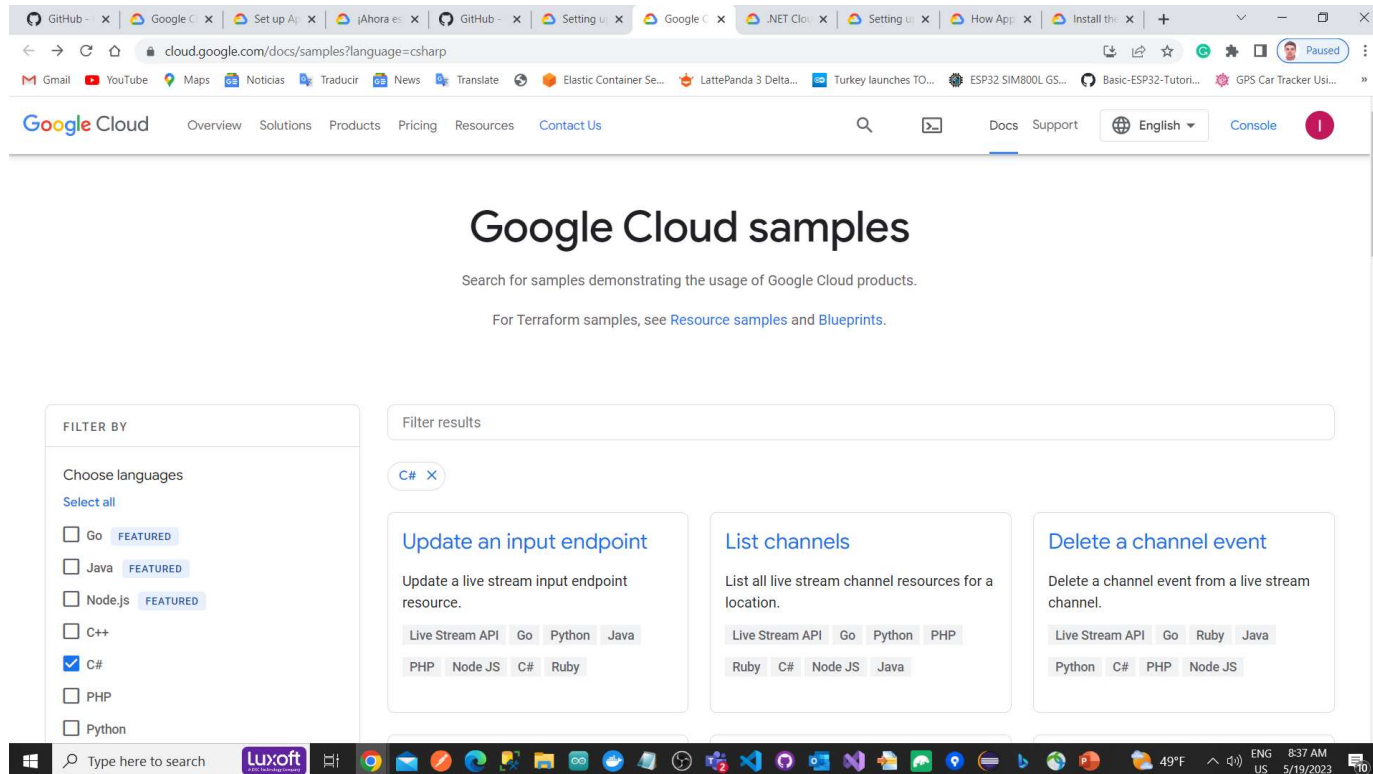


Google Cloud Platform .NET Cloud Client Libraries



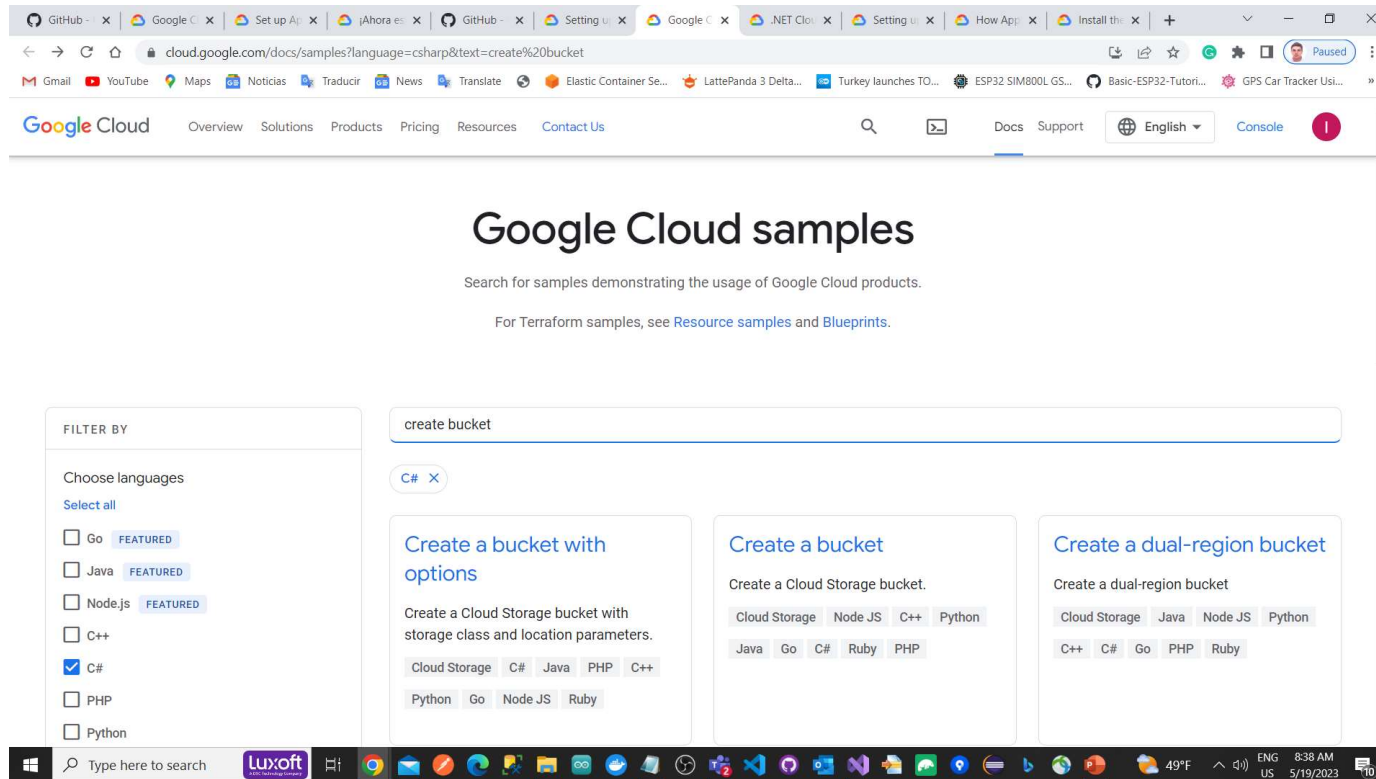
<https://github.com/GoogleCloudPlatform/dotnet-docs-samples>

Google Cloud Platform .NET Cloud Client Libraries



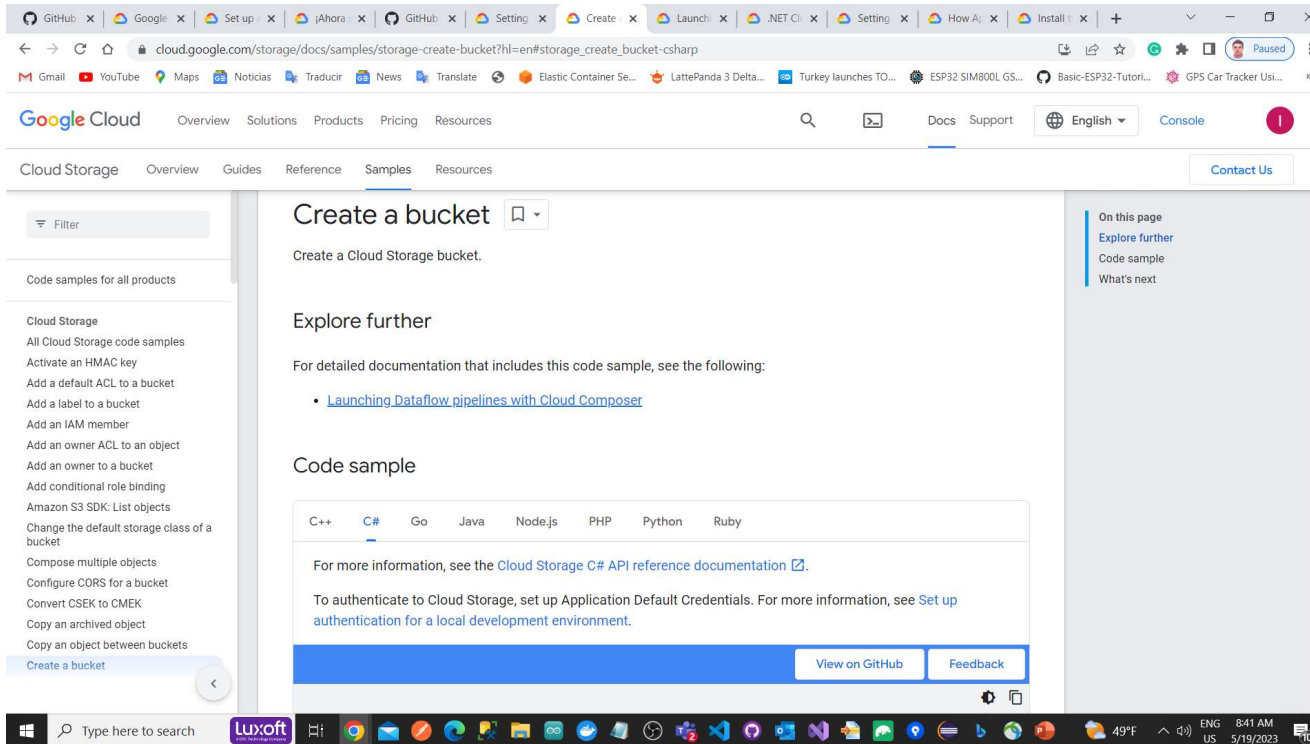
<https://cloud.google.com/docs/samples?language=csharp>

Google Cloud Platform .NET Cloud Client Libraries



<https://cloud.google.com/docs/samples?language=csharp&text=create%20bucket>

Google Cloud Platform .NET Cloud Client Libraries



https://cloud.google.com/storage/docs/samples/storage-create-bucket?hl=en#storage_create_bucket-csharp

Google Cloud Platform .NET Cloud Client Libraries

The screenshot shows a web browser displaying the Google Cloud Platform .NET Cloud Client Libraries documentation for the Cloud Storage C# API. The page is titled "Cloud Storage C# API reference documentation" and includes a sidebar with a list of code samples. The main content area shows the C# code for creating a bucket, with a "View on GitHub" button and a "Feedback" button. The code is as follows:

```
using Google.Apis.Storage.v1.Data;
using Google.Cloud.Storage.V1;
using System;

public class CreateBucketSample
{
    public Bucket CreateBucket(
        string projectId = "your-project-id",
        string bucketName = "your-unique-bucket-name")
    {
        var storage = StorageClient.Create();
        var bucket = storage.CreateBucket(projectId, bucketName);
        Console.WriteLine($"Created {bucketName}.");
        return bucket;
    }
}
```

The sidebar on the left lists various code samples for Cloud Storage, including "Create a bucket", "Add a label to a bucket", "Add an IAM member", "Add an owner ACL to an object", "Add an owner to a bucket", "Add conditional role binding", "Amazon S3 SDK: List objects", "Change the default storage class of a bucket", "Compose multiple objects", "Configure CORS for a bucket", "Convert CSEK to CMEK", "Copy an archived object", "Copy an object between buckets", and "Create a bucket". The "Create a bucket" sample is currently selected.

Google Cloud Platform .NET Cloud Client Libraries

The screenshot shows the Google Cloud Platform .NET Cloud Client Libraries documentation page. The browser window has multiple tabs open, including GitHub, Google, and various development-related sites. The address bar shows the URL cloud.google.com/dotnet/docs/reference. The Google Cloud logo is at the top left, with navigation links for Overview, Solutions, Products, Pricing, and Resources. The main content area is titled ".NET Cloud Client Libraries" and includes a "Send feedback" button. The text explains that these libraries are the recommended way to access Google Cloud APIs programmatically, reducing boilerplate code. It also provides a link to "Cloud Client Libraries explained" for more information. A search bar is present, and a table lists the available libraries and their corresponding package names.

Libraries	
Access Approval	Google.Cloud.AccessApproval.V1
Access Context Manager	Google.Identity.AccessContextManager.Type Google.Identity.AccessContextManager.V1
Advisory Notifications	Google.Cloud.AdvisoryNotifications.V1

<https://cloud.google.com/dotnet/docs/reference>

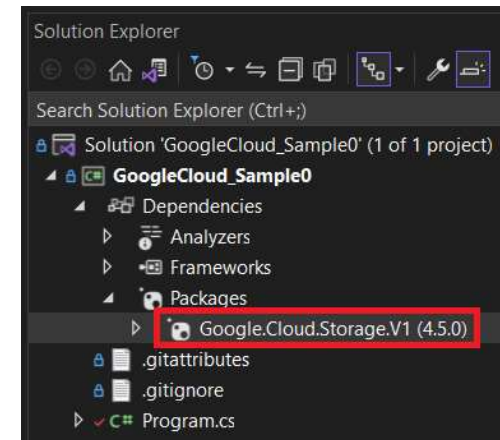
Google Cloud Platform .NET Cloud Client Libraries

Buckets

```
using Google.Apis.Storage.v1.Data;
using Google.Cloud.Storage.V1;

string projectId = "focus-cache-387205";
string bucketName = "luiscocoenriquezsegundo";
string localPath = "C://NetChapterArticles.txt";
string objectName = "NetChapterArticles.txt";

//-----
//Create Bucket
//-----
var storage = StorageClient.Create();
var bucket = storage.CreateBucket(projectId, bucketName);
Console.WriteLine($"Created {bucketName}.");
//-----
//List Buckets
//-----
//var storage = StorageClient.Create();
var bucketsluis = storage.ListBuckets(projectId);
Console.WriteLine("Buckets:");
foreach (var bucketluis in bucketsluis)
{
    Console.WriteLine(bucketluis.Name);
}
//-----
//File Upload
//-----
using var fileStream = File.OpenRead(localPath);
storage.UploadObject(bucketName, objectName, null, fileStream);
Console.WriteLine($"Uploaded {objectName}.");
```



```
//-----
//File Download
//-----
string downloadlocalpath = "C:\\New folder\\DownloadedFile.txt";
using var outputFile = File.OpenWrite(downloadlocalpath);
storage.DownloadObject(bucketName, objectName, outputFile);
Console.WriteLine($"Downloaded {objectName} to {localPath}.");
//-----
//List Files
//-----
var storageObjects = storage.ListObjects(bucketName);
Console.WriteLine($"Files in bucket {bucketName}:");
foreach (var storageObject in storageObjects)
{
    Console.WriteLine(storageObject.Name);
}
```

Google Cloud Platform .NET Cloud Client Libraries

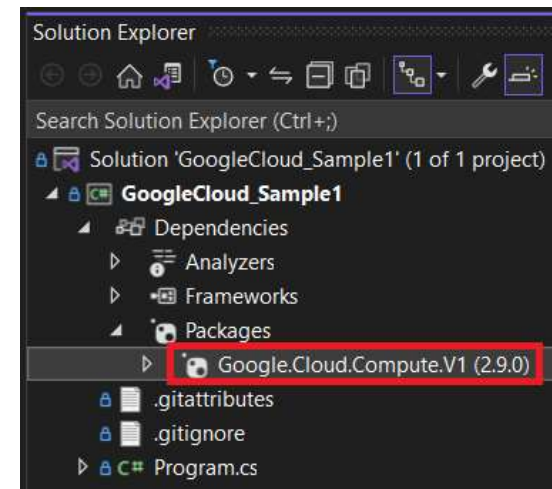
Creating a Virtual Machine

```
using Google.Cloud.Compute.V1;

string projectId = "focus-cache-387205";
string zone = "europe-southwest1-a";
string machineName = "luis-test-machine";
string machineType = "e2-micro";
string diskImage = "projects/debian-cloud/global/images/family/debian-10";
long diskSizeGb = 10;
string networkName = "default";

Instance instance = new Instance
{
    Name = machineName,
    MachineType = $"zones/{zone}/machineTypes/{machineType}",
    Disks =
    {
        new AttachedDisk
        {
            AutoDelete = true,
            Boot = true,
            Type = ComputeEnumConstants.AttachedDisk.Type.Persistent,
            InitializeParams = new AttachedDiskInitializeParams
            {
                SourceImage = diskImage,
                DiskSizeGb = diskSizeGb
            }
        }
    },
    NetworkInterfaces = { new NetworkInterface { Name = networkName } }
};
```

```
InstancesClient client = await InstancesClient.CreateAsync();
var instanceCreation = await client.InsertAsync(projectId, zone, instance);
await instanceCreation.PollUntilCompletedAsync();
```



Google Cloud Platform .NET Cloud Client Libraries

Deleting a Virtual Machine

```
using Google.Api;
using Google.Cloud.Compute.V1;

string projectId = "focus-cache-387205";
string zone = "europe-southwest1-a";
string machineName = "luis-test-machine";

InstancesClient client = await InstancesClient.CreateAsync();

// Stop the VM instance before deleting it.
var stopRequest = new StopInstanceRequest
{
    Project = projectId,
    Zone = zone,
    Instance = machineName
};

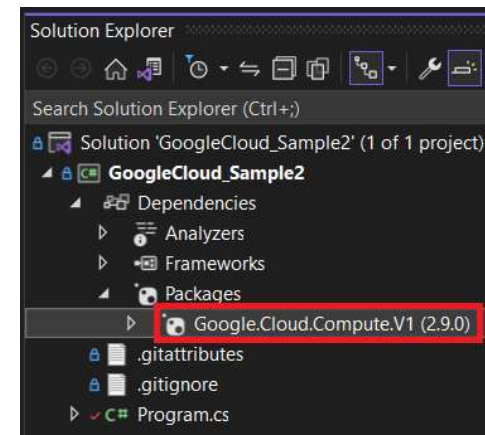
await client.StopAsync(stopRequest);
```

```
// Start the VM instance before deleting it.
//var startRequest = new StartInstanceRequest
//{
//    Project = projectId,
//    Zone = zone,
//    Instance = machineName
//};

//await client.StartAsync(startRequest);

// Make the request to delete a VM instance.
var instanceDeletion = await client.DeleteAsync(projectId, zone, machineName);

//Wait for the operation to complete using client-side polling.
await instanceDeletion.PollUntilCompletedAsync();
```



Google Cloud Platform .NET Cloud Client Libraries

Listing Virtual Machines

```
using Google.Cloud.Compute.V1;

string projectId = "focus-cache-387205";

InstancesClient client = await InstancesClient.CreateAsync();
IList<Instance> allInstances = new List<Instance>();

// Make the request to list all VM instances in a project.
await foreach (var instancesByZone in client.AggregatedListAsync(projectId))
{
    Console.WriteLine($"Instances for zone: {instancesByZone.Key}");
    foreach (var instance in instancesByZone.Value.Instances)
    {
        Console.WriteLine($"-- Name: {instance.Name}");
        allInstances.Add(instance);
    }
}
```

Listing Virtual Machines Types

```
using Google.Cloud.Compute.V1;

MachineTypesClient machineTypesClient = MachineTypesClient.Create();
ImagesClient imagesClient = ImagesClient.Create();

// List machine types
var machineTypesList = machineTypesClient.List(new ListMachineTypesRequest
{
    Project = "focus-cache-387205",
    Zone = "europe-southwest1-a"
});

Console.WriteLine("Machine Types:");
foreach (MachineType machineType in machineTypesList)
{
    Console.WriteLine($"- {machineType.Name}");
}

Console.WriteLine();
```

Google Cloud Platform .NET Cloud Client Libraries

Datastore

```
using Google.Cloud.Datastore.V1;

string projectId = "focus-cache-387205";

DatastoreDb db = DatastoreDb.Create(projectId);

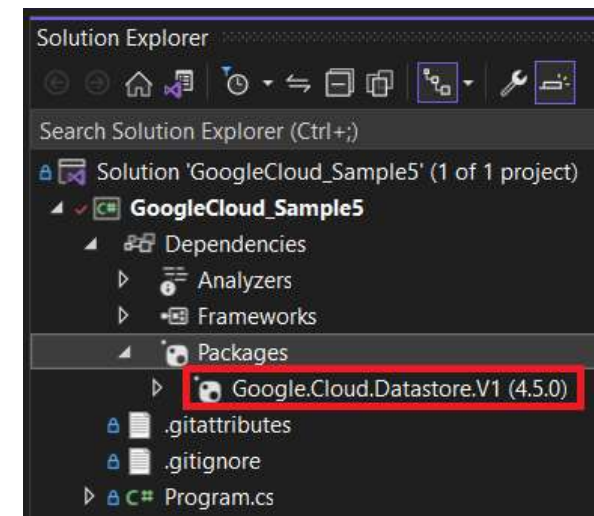
string kind = "Task";
string name = "sampletask1";

KeyFactory keyFactory = db.CreateKeyFactory(kind);
Key key = keyFactory.CreateKey(name);

var task = new Entity
{
    Key = key,
    ["description"] = "Buy milk"
};

using (DatastoreTransaction transaction = db.BeginTransaction())
{
    // Saves the task
    transaction.Upsert(task);
    transaction.Commit();

    Console.WriteLine($"Saved {task.Key.Path[0].Name}: {(string)task["description"]}");
}
```



Google Cloud Platform .NET Cloud Client Libraries

Firestore

```
using Google.Cloud.Firestore;

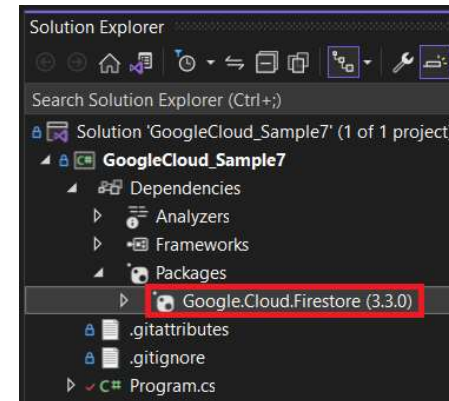
string project = "focus-cache-387205";

AddData1(project).Wait();
RetrieveAllDocuments(project).Wait();

void InitializeProjectId(string project)
{
    FirestoreDb db = FirestoreDb.Create(project);
    Console.WriteLine("Created Cloud Firestore client with project ID: {0}",
        project);
}

async Task AddData1(string project)
{
    FirestoreDb db = FirestoreDb.Create(project);
    DocumentReference docRef = db.Collection("users").Document("alovelace");
    Dictionary<string, object> user = new Dictionary<string, object>
    {
        { "First", "Ada" },
        { "Last", "Lovelace" },
        { "Born", 1815 }
    };
    await docRef.SetAsync(user);
    Console.WriteLine("Added data to the avelace document in the users
collection.");
}
```

```
async Task RetrieveAllDocuments(string project)
{
    FirestoreDb db = FirestoreDb.Create(project);
    CollectionReference usersRef = db.Collection("users");
    QuerySnapshot snapshot = await usersRef.GetSnapshotAsync();
    foreach (DocumentSnapshot document in snapshot.Documents)
    {
        Console.WriteLine("User: {0}", document.Id);
        Dictionary<string, object> documentDictionary = document.ToDictionary();
        Console.WriteLine("First: {0}", documentDictionary["First"]);
        if (documentDictionary.ContainsKey("Middle"))
        {
            Console.WriteLine("Middle: {0}", documentDictionary["Middle"]);
        }
        Console.WriteLine("Last: {0}", documentDictionary["Last"]);
        Console.WriteLine("Born: {0}", documentDictionary["Born"]);
        Console.WriteLine();
    }
}
```



Google Cloud Platform .NET Cloud Client Libraries

Pub/Sub

```
using Google.Cloud.PubSub.V1;
using Grpc.Core;

string projectId = "focus-cache-387205";
string topicId = "luis-topic-1";
string subscriptionId = "subscription-first";

//Create a topic
PublisherServiceApiClient publisher = PublisherServiceApiClient.Create();
var topicName = TopicName.FromProjectTopic(projectId, topicId);
Topic topic = null;

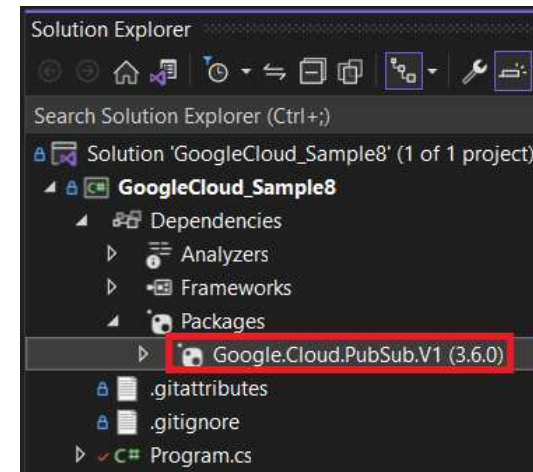
try
{
    topic = publisher.CreateTopic(topicName);
    Console.WriteLine($"Topic {topic.Name} created.");
}
catch (RpcException e) when (e.Status.StatusCode == StatusCode.AlreadyExists)
{
    Console.WriteLine($"Topic {topicName} already exists.");
}
```

//Create a Subscription for the topic

```
SubscriberServiceApiClient subscriber = SubscriberServiceApiClient.Create();

SubscriptionName subscriptionName =
    SubscriptionName.FromProjectSubscription(projectId, subscriptionId);
Subscription subscription = null;

try
{
    subscription = subscriber.CreateSubscription(subscriptionName, topicName,
        pushConfig: null, ackDeadlineSeconds: 60);
}
catch (RpcException e) when (e.Status.StatusCode == StatusCode.AlreadyExists)
{
    // Already exists. That's fine.
}
```



Google Cloud Platform .NET Cloud Client Libraries

Pub/Sub

```
//Publish messages to the above created topic
```

```
PublisherClient publisher1 = await PublisherClient.CreateAsync(topicName);

int publishedMessageCount = 0;

List<string> messageTexts = new List<string>();
messageTexts.Add("First message");
messageTexts.Add("Second message");
messageTexts.Add("Third message");
messageTexts.Add("Fourth message");
messageTexts.Add("Fifth message");

var publishTasks = messageTexts.Select(async text =>
{
    try
    {
        string message = await publisher1.PublishAsync(text);
        Console.WriteLine($"Published message {message}");
        Interlocked.Increment(ref publishedMessageCount);
    }
    catch (Exception exception)
    {
        Console.WriteLine($"An error occurred when publishing message {text}: {exception.Message}");
    }
});
await Task.WhenAll(publishTasks);
```

Google Cloud Platform .NET Cloud Client Libraries

Pub/Sub

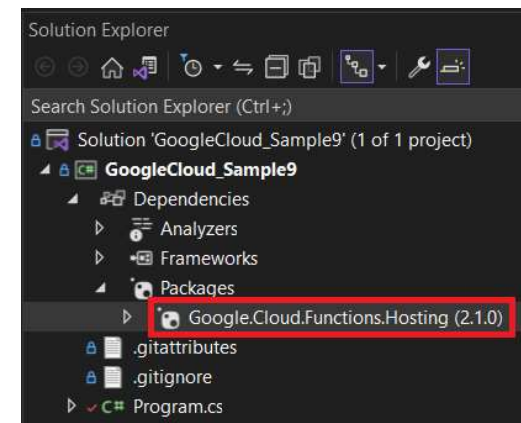
```
int messages_Count_received = PullMessagesSync(projectId, subscriptionId, true);

int PullMessagesSync(string projectId, string subscriptionId, bool acknowledge)
{
    SubscriptionName subscriptionName = SubscriptionName.FromProjectSubscription(projectId, subscriptionId);
    SubscriberServiceApiClient subscriberClient = SubscriberServiceApiClient.Create();
    int messageCount = 0;
    try
    {
        // Pull messages from server,
        PullResponse response = subscriberClient.Pull(subscriptionName, maxMessages: 20);
        foreach (ReceivedMessage msg in response.ReceivedMessages)
        {
            string text = System.Text.Encoding.UTF8.GetString(msg.Message.Data.ToArray());
            Console.WriteLine($"Message {msg.Message.MessageId}: {text}");
            Interlocked.Increment(ref messageCount);
        }
        if (acknowledge && messageCount > 0)
        {
            subscriberClient.Acknowledge(subscriptionName, response.ReceivedMessages.Select(msg => msg.AckId));
        }
    }
    catch (RpcException ex) when (ex.Status.StatusCode == StatusCode.Unavailable)
    {
        // UNAVAILABLE due to too many concurrent pull requests pending for the given subscription.
    }
    return messageCount;
}
```

Google Cloud Platform .NET Cloud Client Libraries

Functions

```
using Google.Cloud.Functions.Framework;  
using Microsoft.AspNetCore.Http;  
  
namespace HelloWorld;  
  
public class Function : IHttpFunction  
{  
    public async Task HandleAsync(HttpContext context)  
    {  
        await context.Response.WriteAsync("Hello World!");  
    }  
}
```



<https://cloud.google.com/functions/docs/concepts/overview>

Google Cloud Platform .NET Cloud Client Libraries

Functions

```
using CloudNative.CloudEvents;
using Google.Cloud.Functions.Framework;
using Google.Events.Protobuf.Cloud.Storage.V1;
using Microsoft.Extensions.Logging;
using System.Threading;
using System.Threading.Tasks;

namespace HelloGcs;

/// <summary>
/// Example Cloud Storage-triggered function.
/// This function can process any event from Cloud Storage.
/// </summary>
public class Function : ICloudEventFunction<StorageObjectData>
{
    private readonly ILogger _logger;

    public Function(ILogger<Function> logger) =>
        _logger = logger;

    public Task HandleAsync(CloudEvent cloudEvent, StorageObjectData data, CancellationToken cancellationToken)
    {
        _logger.LogInformation("Event: {event}", cloudEvent.Id);
        _logger.LogInformation("Event Type: {type}", cloudEvent.Type);
        _logger.LogInformation("Bucket: {bucket}", data.Bucket);
        _logger.LogInformation("File: {file}", data.Name);
        _logger.LogInformation("Metageneration: {metageneration}", data.Metageneration);
        _logger.LogInformation("Created: {created:s}", data.TimeCreated?.ToDateTimeOffset());
        _logger.LogInformation("Updated: {updated:s}", data.Updated?.ToDateTimeOffset());
        return Task.CompletedTask;
    }
}
```

