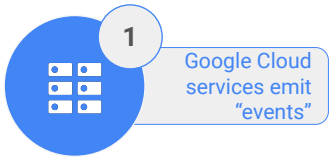
Every developer dreams of a world where the developer doesn't have to set up and manage infrastructure. Especially, if all that's needed is a small bit of logic to do some quick processing on a piece of data.

Cloud Functions make it possible to run a completely serverless environment, where logic can be executed on demand and in response to events. Cloud Functions can act as a glue between disparate applications. With Cloud Functions, you can build a serverless microservices architecture that's highly scalable. You can simply focus on the code and not worry about setting up servers to run that code.

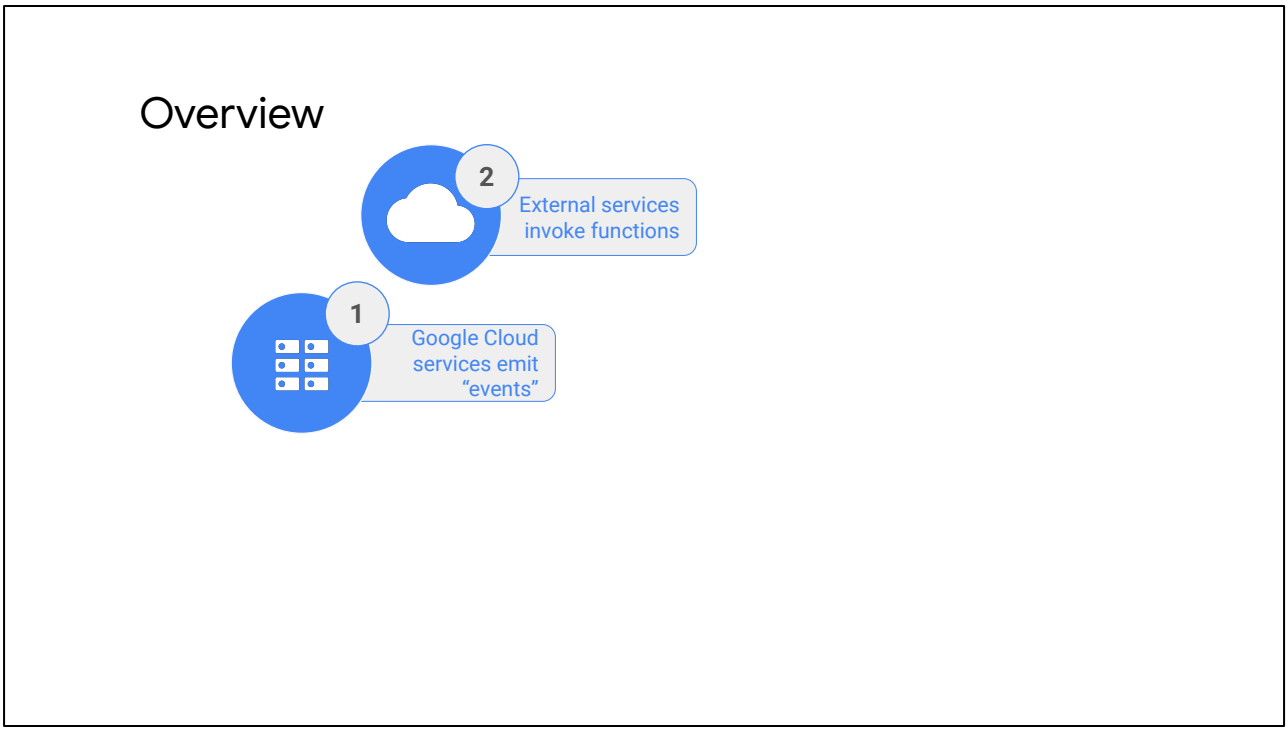
In this module, Using Cloud Functions for Event-Driven Processing:

- You'll learn about Cloud Functions triggers.
- You'll learn how to execute Cloud Functions in response to asynchronous events and synchronous HTTP invocations.
- You'll develop and deploy Cloud functions to perform event-driven processing, and
- You'll learn how to handle logging, error reporting, and monitoring for Cloud Functions.

Overview

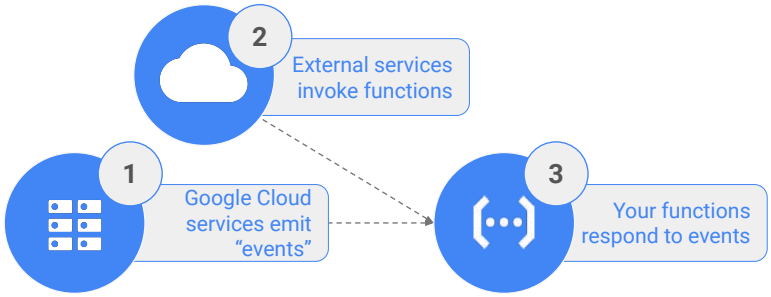


Google Cloud services emit events, such as when files are uploaded to a Google Cloud Storage bucket or messages are published to a Pub/Sub topic.

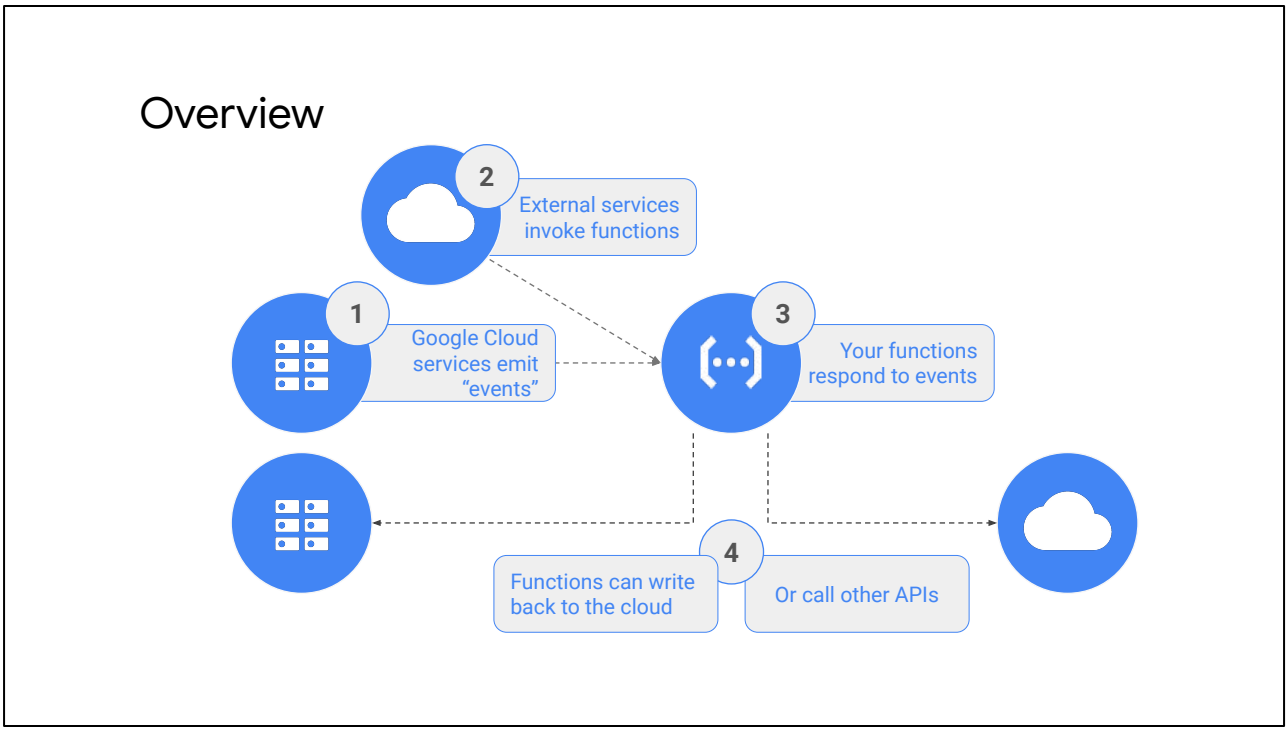


External services can invoke functions in response to events in those systems, such as when a commit is made to a Github repository.

Overview



Cloud Functions are triggered in response to these events.

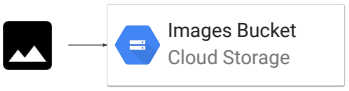


Cloud Functions can invoke other APIs or write data back to the cloud.

Cloud Functions enable event-driven, serverless, highly scalable microservices

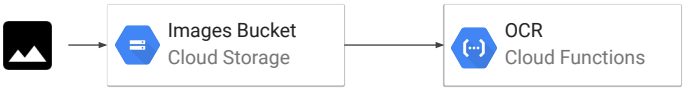
With Cloud Functions, you can develop an application that is event-driven, serverless, and highly scalable. Each function is a lightweight microservice that enables you to integrate application components and data sources. Cloud Functions are ideal for microservices that require a small piece of code to quickly process data related to an event. Cloud Functions are priced according to how long your function runs, the number of times it is invoked, and the resources that you provision for the function.

Cloud Functions enable event-driven, serverless, highly scalable microservices



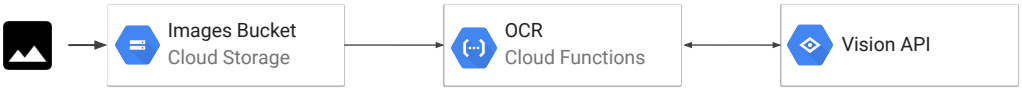
Consider an application that allows users to upload images that contain text to a bucket in Google Cloud Storage.

Cloud Functions enable event-driven, serverless, highly scalable microservices



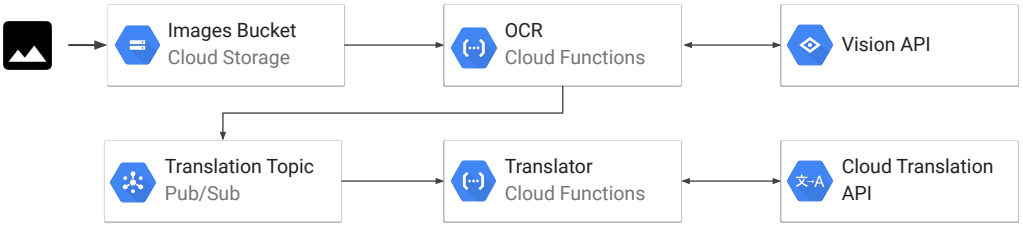
The OCR Cloud Function is triggered when a new image is uploaded to the Images bucket.

Cloud Functions enable event-driven, serverless, highly scalable microservices



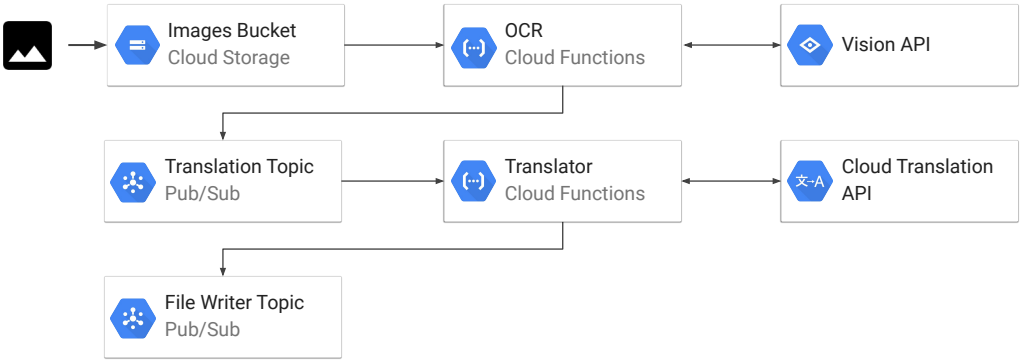
This function uses the Vision API to extract text from images and then queues up the text in Pub/Sub for translation.

Cloud Functions enable event-driven, serverless, highly scalable microservices



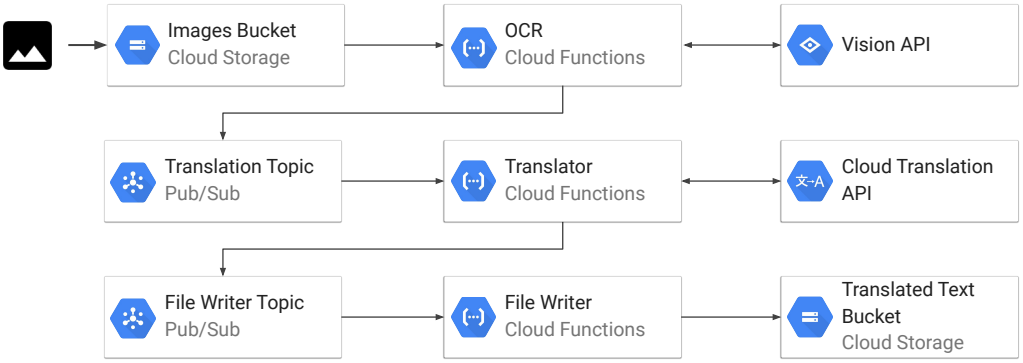
The Translator Cloud Function, triggered by the Pub/Sub topic, invokes the Cloud Translation API to translate the text.

Cloud Functions enable event-driven, serverless, highly scalable microservices



It queues up the translated text in the File Writer topic in Pub/Sub.

Cloud Functions enable event-driven, serverless, highly scalable microservices



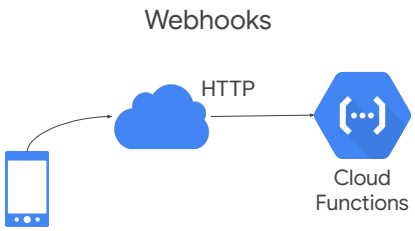
The File Writer Cloud Function writes the translated text for each image as separate files in Cloud Storage.

In this example, all components of the application use fully managed services and APIs such as Cloud Storage, Pub/Sub, Cloud Functions, the Vision API, and the Cloud Translation API. These services scale automatically depending on the volume of incoming data and required compute resources. This scalability and reliability enables you to focus on your application code.

Cloud Functions use cases

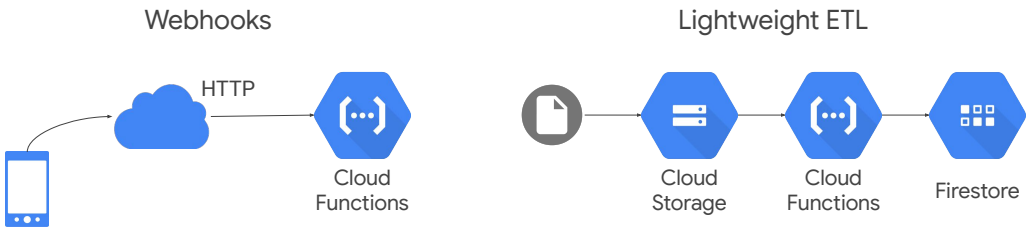
Cloud Functions can be used in a variety of use cases that require lightweight microservices or event-driven processing.

Cloud Functions use cases



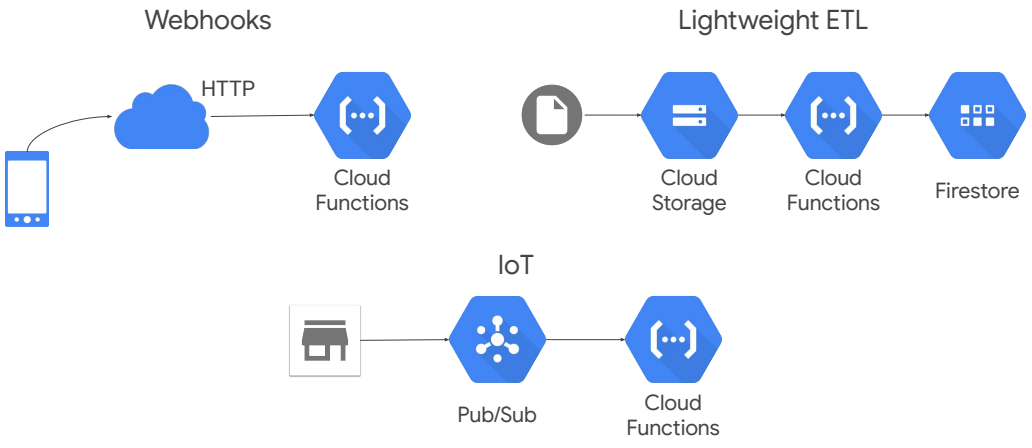
Cloud Functions can serve as webhooks. For example, you can set up a webhook that is invoked automatically after each commit to a Git repository. External and internal clients can make direct HTTP calls to invoke microservices that are deployed as Cloud Functions.

Cloud Functions use cases



You can use Cloud Functions for lightweight extract-transform-load (ETL) operations. For example, when a file is uploaded to Cloud Storage, a Cloud Function can be triggered to transform and upload the contents to a database.

Cloud Functions use cases



You can also use Cloud Functions to process IoT streaming data or other application messages that are published to a Pub/Sub topic.

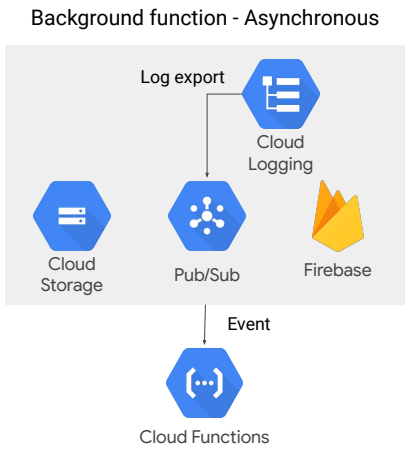
Cloud Functions features

- Pay as you go
- No servers to provision, manage or upgrade
- Highly scalable, automatically scaling based on load
- Pay for your function's execution time, pay nothing when it's idle
- Write your functions in Node.js, Python, Go or Java.

Cloud Functions is a scalable pay-as-you-go functions as a service (FaaS), which enables you to run your code with zero server management. Cloud Functions is highly scalable, automatically scaling up and down in response to events. You are only billed for your function's execution time, metered to the nearest 100 milliseconds. You pay nothing when your function is idle.

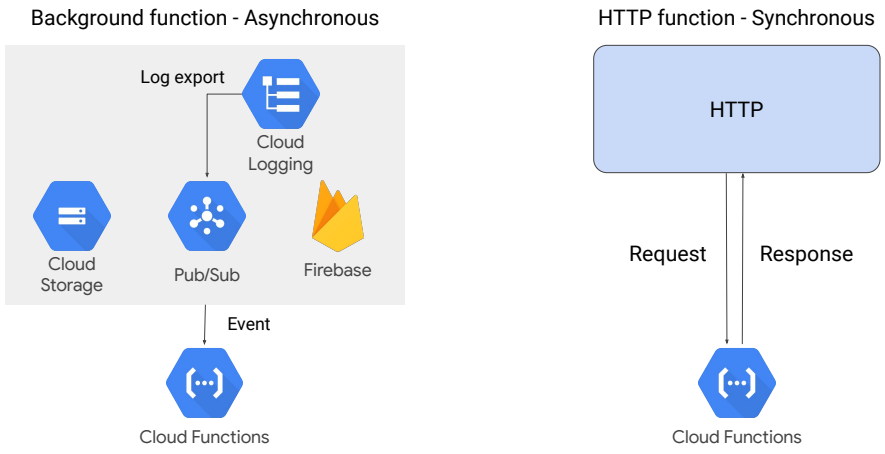
Cloud Functions can be written in Node.js, Python, Go, and Java, and are executed in language-specific runtimes.

Cloud Functions can have asynchronous and synchronous triggers

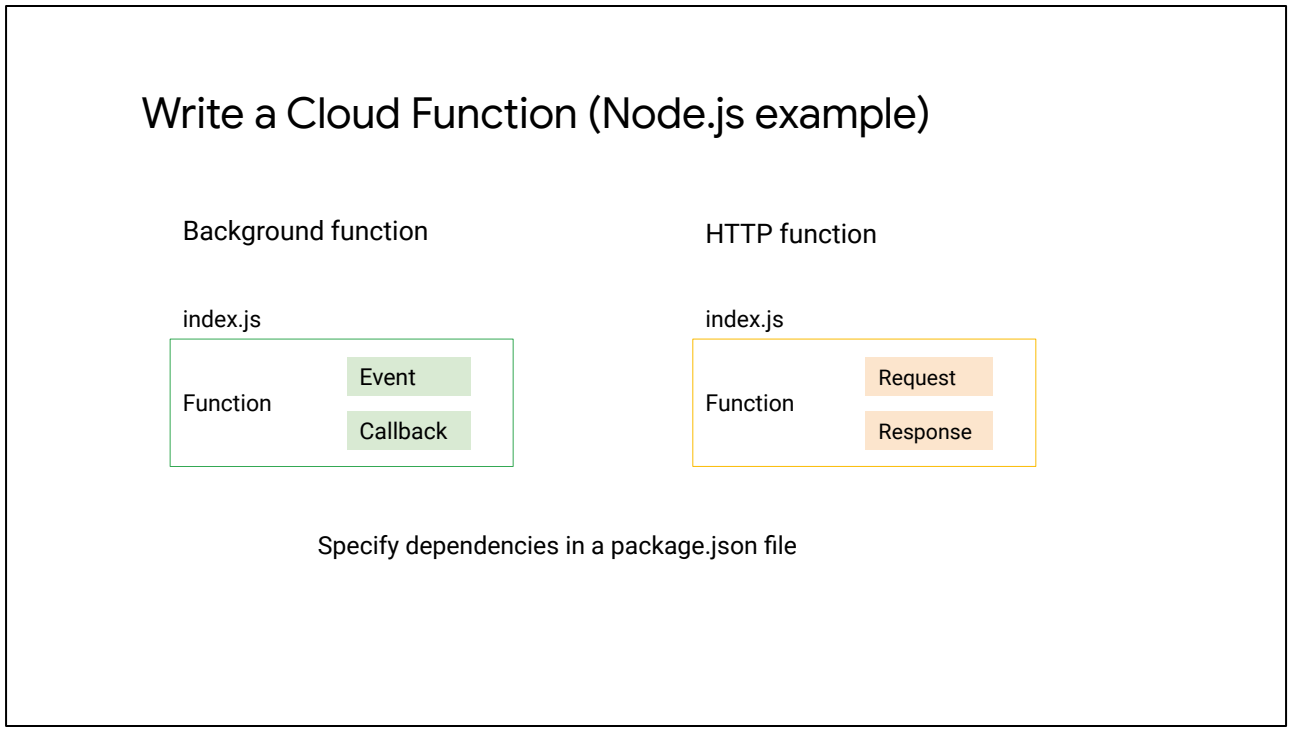


Cloud functions can be triggered asynchronously in response to events such as those in Cloud Storage, Pub/Sub or Firebase. Asynchronously triggered functions are called background functions.

Cloud Functions can have asynchronous and synchronous triggers



Cloud Functions can also be invoked synchronously by direct HTTP request responses. Functions invoked in this manner are called HTTP functions. Remember Cloud functions have a default timeout value of 60 seconds. Ensure that the function will complete execution before timing out. For HTTP functions, it's important to keep the execution time to a minimum, to avoid a negative user experience.



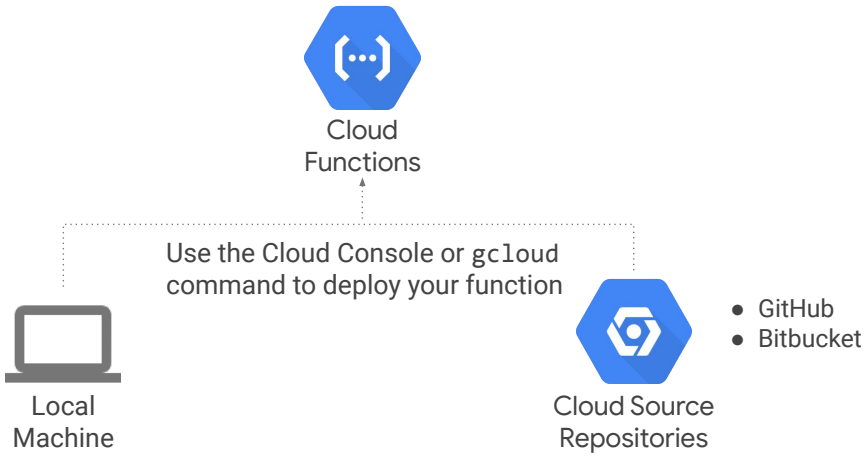
Let's look at how you can write, deploy, and monitor Cloud Functions.

You can write the code for Cloud Functions as Node.js functions in an index.js file. A background function takes an event and callback function as input parameters. An HTTP function takes request and response objects as input parameters.

You do not need to upload zip files with packaged dependencies. You can specify dependencies in a package.json file. The Cloud Functions service automatically installs all dependencies before running your code.

Cloud Functions provides access to a local disk mount point, /tmp, that you can use for temporary file processing. Writing to the /tmp location uses the memory that you have allocated to your function. Make sure to clean up any temporary files.

Deploy your Cloud Function



You can deploy your function from your local machine by using the Cloud Console or the gcloud command. The gcloud command automatically zips your code and uploads it to the Cloud Storage staging bucket that you specify.

You can also deploy your code directly from an independent Google Cloud Source Repository, or from a Cloud Source Repository that is automatically synchronized with a Github or Bitbucket repository.

Cloud Functions supports logging, error reporting, and monitoring

INFO log level: console.log(...)
ERROR log level: console.error(...)
DEBUG log level: internal system messages

Cloud Logging

You can view the output from your console.log and console.error messages in Cloud Logging.

Cloud Functions supports logging, error reporting, and monitoring

INFO log level: console.log(...)
ERROR log level: console.error(...)
DEBUG log level: internal system messages

Cloud Logging

Errors thrown or reported manually

Error Reporting

If your code throws errors, or in case of other uncaught errors, Cloud Functions automatically captures these errors in Error Reporting. If you only want to report these errors to Error Reporting without throwing the error up to the caller, you can programmatically report these errors to Error Reporting.

Cloud Functions supports logging, error reporting, and monitoring

INFO log level: console.log(...)
ERROR log level: console.error(...)
DEBUG log level: internal system messages

Cloud Logging

Errors thrown or reported manually

Error Reporting

Number of invocations, execution time, memory usage

Cloud Functions

In the Cloud Console, you can view function metrics related to the number of invocations, execution time, and memory usage.

Demo 05: Exploring Cloud Functions

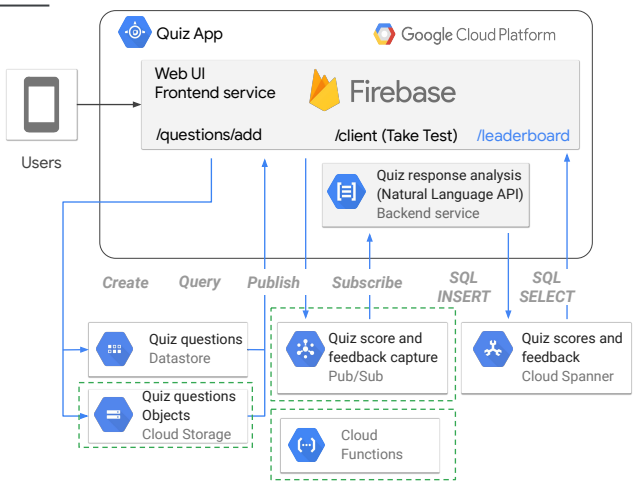
Process Pub/Sub Data Using Cloud Functions

Duration: 45 minutes

Lab objectives

Create a Cloud Function that responds to Pub/Sub messages

Deploy multiple files to a Cloud Function




Google Cloud