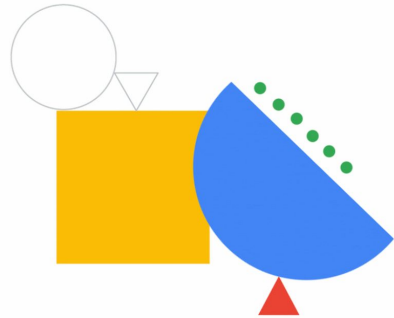Google Cloud

Introduction to Cloud Functions

In this module, we introduce Cloud Functions and discuss their features, and benefits.

We also review some of the use cases for Cloud Functions, and do a lab to create your first function in the Google Cloud console.

Agenda

01	Cloud Functions
02	Features and benefits of Cloud Functions
03	Cloud Functions use cases
04	Supported language runtimes and regions
05	Lab: Developing and Deploying Cloud Functions
06	Quiz

Google Cloud

In this module, you learn about:

- The features and benefits of Cloud Functions, and some of the use cases for deploying Cloud Functions.

- Supported language runtimes and regions.

- Building and deploying Cloud Functions

You'll complete a lab to develop and deploy Cloud Functions from the Google Cloud console and the gcloud CLI.

We'll also do a short quiz on the topics in this module.

What is Cloud Functions?

- Fully managed serverless execution environment
- Scalable functions-as-a-service (FaaS) platform
- Can be used to connect cloud services
- No need to provision infrastructure or manage servers
- Integrated with Cloud Operations for logging and monitoring
- Two product versions:
  - 1st gen
  - 2nd gen built on Cloud Run and Eventarc

Google Cloud

Cloud Functions is a fully managed serverless execution environment on Google Cloud for you to build, deploy, and run functions.
There is no need to provision any infrastructure or manage individual servers.
Depending on your requirements, you can connect to other cloud services from Cloud Functions.

It's also fully integrated with Cloud Operations for observability and diagnosis.

Cloud Functions offers two product versions: Cloud Functions (1st gen), the original version, and Cloud Functions (2nd gen), a new version built on Cloud Run and Eventarc to provide an enhanced feature set.

See Cloud Functions version comparison for more information.

What is Cloud Run?

- A managed compute platform
- Runs containers on Google's infrastructure
- Supports source-based deployment that builds containers for you
- Build full-featured applications with other Google Cloud services

Google Cloud

Cloud Run is a managed compute platform that lets you run containers directly on top of Google's scalable infrastructure.

You can deploy code written in any programming language on Cloud Run if you can build a container image from it.

You can use the source-based deployment option that builds the container for you, when developing your application in Go, Node.js, Python, Java, .NET Core, or Ruby.

Cloud Run works well with other services on Google Cloud, so you can build full-featured applications without spending too much time operating, configuring, and scaling your Cloud Run service.

A Cloud function:

- Is a simple piece of code that implements a single-purpose function.
- Can be invoked directly from an HTTP request.
- Can be triggered in response to an event generated from cloud infrastructure and services.
- Executes in a fully managed environment.
- Is written in one of many supported programming languages:
 - Node.js, Python, Go, Java, .NET, Ruby, and PHP

Google Cloud
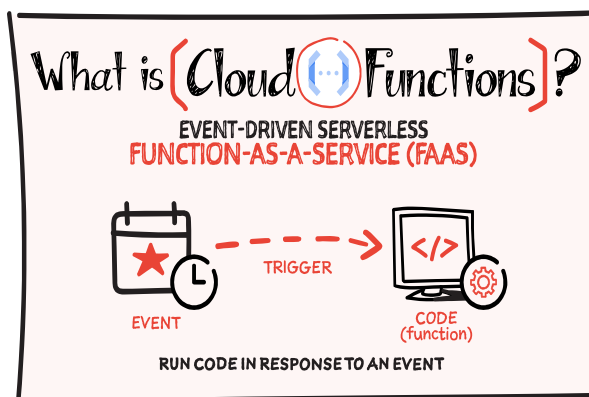
A cloud function is simple code that you write that serves a single function.
It's triggered by an event that is generated by cloud infrastructure and other services.
Cloud Functions execute in a fully managed serverless environment.
You can develop Cloud Functions in any supported language.

Slide graphic from Sketch notes:
https://cloud.google.com/blog/topics/developers-practitioners/learn-cloud-functions-snap

Using Cloud Functions

- A connective layer to connect and extend cloud services.
- Examples:
    - Respond to a file upload to Cloud Storage
   - Respond to a message published to a Pub/Sub topic

Google Cloud
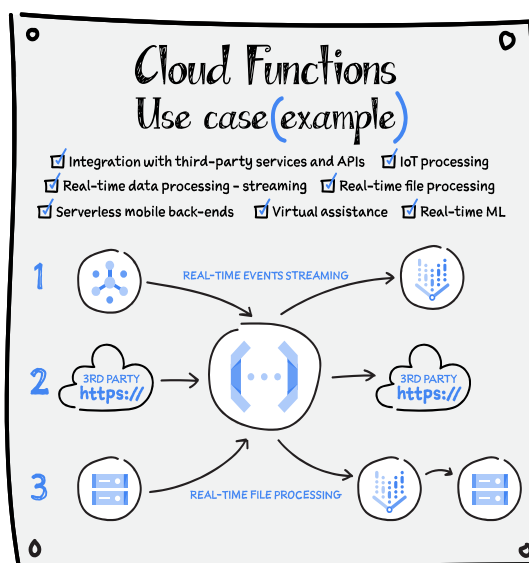
Cloud Functions provides a connective layer of logic that lets you write code to connect and extend cloud services.
For example, a cloud function can respond to a file upload to Cloud Storage, or an incoming message on a Pub/Sub topic.

Slide graphic from Sketch notes:
https://cloud.google.com/blog/topics/developers-practitioners/learn-cloud-functions-snap

To use Cloud Functions:
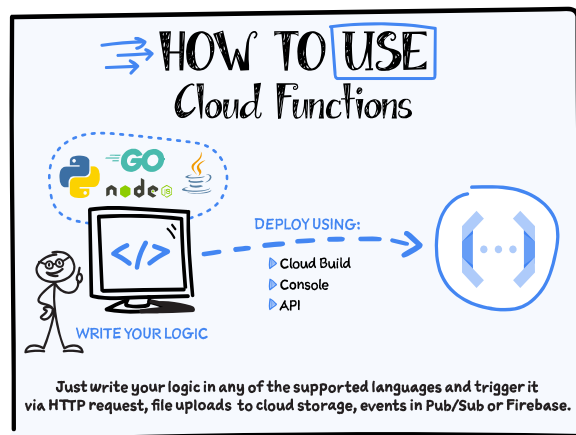
- Write your function logic in any of the supported programming languages.
- Deploy the function.
- Trigger the function from:
  - An HTTP request
  - An Event source

To set up and use Cloud Functions, you:
- Develop the function code or logic in one of the supported programming languages.
- Deploy the function.
- Set up a trigger for the function to execute in response to HTTP requests or supported cloud events.

Slide graphic from Sketch notes:
https://cloud.google.com/blog/topics/developers-practitioners/learn-cloud-functions-snap

Features, benefits, and use cases
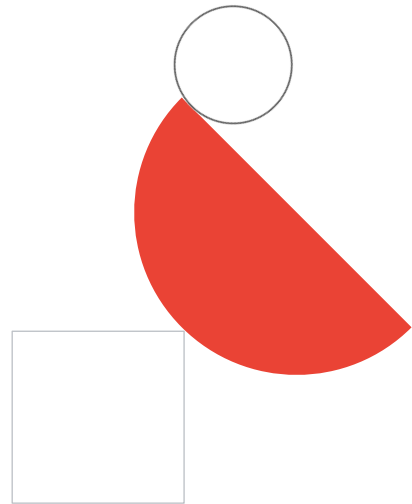
Here are some of the features and benefits of using Cloud Functions:

Cloud Functions

Features:

- ✓ Can be locally developed and tested
- ✓ Seamless authentication
- ✓ HTTP and event-driven
- ✓ Integrates with cloud databases
- ✓ Portable using open source frameworks

Benefits:

- ✓ Augments existing cloud services
- ✓ Serverless
- ✓ Auto scalable
- ✓ Observable
- ✓ Pay as you go

Google Cloud

Features:

- Portability: A function can run in any standard runtime environment for one of the supported languages.
- Local development and testing: You can develop and test functions in a local development environment before deploying to Cloud Functions.
- Seamlessly authenticate your cloud functions with other Google Cloud services using service accounts.
- HTTP and event-driven: Cloud Functions execute in response to cloud events and HTTP requests.
- Access to Cloud Functions is controlled by identity based (with IAM), and network-based (VPC settings) authentication and authorization.
- You can configure Cloud Functions to connect with Cloud SQL, Cloud Bigtable, Cloud Spanner, and Firestore databases.

Benefits:

- Cloud Functions can be used to augment existing cloud services with programming logic.
- Serverless. The software and infrastructure are fully managed by Google. You don't need to manage servers, update frameworks, or patch operating systems.
- Autoscaling. Resources are automatically provisioned in response to events. A function can scale from a few invocations a day to many millions of invocations

- without requiring any additional work.
- Observable. Cloud Functions are integrated with Cloud Operation's monitoring and logging tools for observability and error diagnosis.
- Pricing. A pay as you go pricing model is used. The cost is based on the number of function invocations, how long the function runs (compute time), and any data transfer fees for outbound network traffic.

Types of Cloud Functions
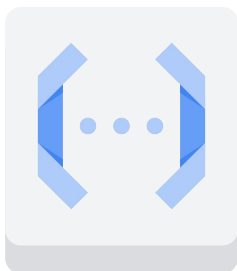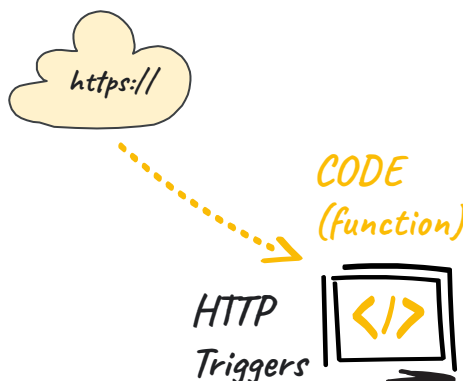
HTTP functions

Event-driven functions
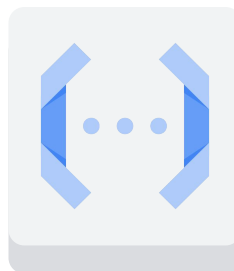
There are two types of Cloud Functions:

- **HTTP functions**, which handle HTTP requests.
- **Event-driven functions**, which handle events from your cloud environment.

HTTP function

- Used to handle HTTP requests
- Provides a URL endpoint
- Uses HTTP triggers
- Ideal for webhooks or single use endpoints
- Supports authenticated or unauthenticated requests

https://

CODE
(function)

HTTP
Triggers

</>

Use HTTP functions with HTTP triggers when you want to invoke a function with an HTTP(S) request.

An HTTP trigger enables a function to run in response to HTTP(S) requests. When you specify an HTTP trigger for a function, the function is assigned a URL at which it can receive requests.

A good use case for using HTTP functions is to implement a webhook that handles http requests from other services.

By default, requests to HTTP functions require authentication. You can choose to allow unauthenticated requests during function deployment.

We discuss securing Cloud Functions in more detail in a later module of this course.

Implementing HTTP functions

HTTP functions

Write a function in your
preferred language which:

- ○ Accepts request/response
  arguments
- ○ Processes the request
- ○ Sends an HTTP response

Google Cloud

To implement an HTTP function, you write an HTTP handler function and register it
with the functions framework for your programming language.

The handler accepts the request and response arguments, processes the request
based on the request method, and sends back an HTTP response.

If the function creates any background tasks like threads, processes, Promise objects,
etc, they must be terminated before the response is sent.

Event-driven function

- Responds to cloud events
- Is implemented using:
    - CloudEvent functions
    - Background functions
- Uses event triggers for:
    - Pub/Sub
    - Cloud Storage
    - Firestore
    - Firebase
    - Eventarc sources
- Integrates with other Google Cloud services that support Pub/Sub

Pub/Sub

Cloud Storage

Other Google Cloud Services

CODE (function)

Firestore

Firebase

Eventarc sources

Event Triggers
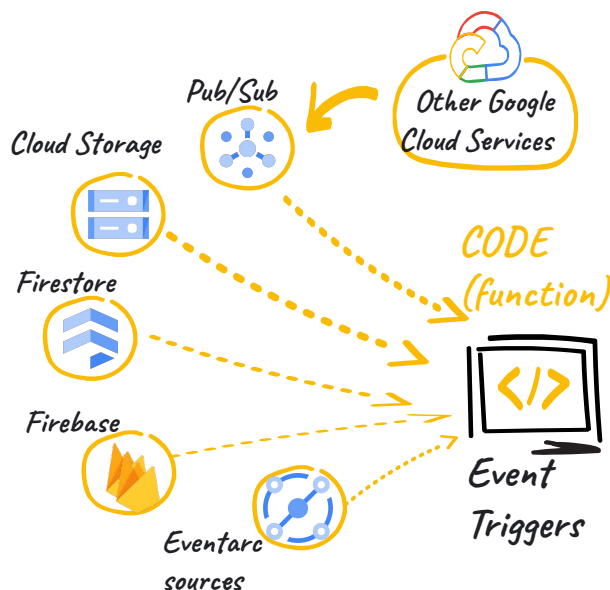
Google Cloud

Use event-driven functions when you want a function to be invoked automatically in response to an event that occurs in your cloud environment.

You implement event-driven functions using CloudEvent functions or Background functions depending on your chosen language runtime, and if you're using Cloud Functions (1st gen) or Cloud Functions (2nd gen).

Event-driven functions use event triggers which react to events within your Google Cloud project.

Event triggers are supported by many Google Cloud services based on if the cloud function is of the 1st generation or 2nd generation.
You can also integrate Cloud Functions with any other Google Cloud service that supports Pub/Sub as an event bus.

We discuss triggers in more detail in a later module of this course.

Implementing event-driven functions

CloudEvent functions

- Based on the CloudEvents specification.
- Registered with the Functions Framework.
- Event data is in CloudEvents format.
- Used by 1st gen (some runtimes) and 2nd gen Cloud Functions.

Background functions

- Older style event-driven function.
- Event data is based on event type.
- Used by 1st gen (some runtimes) Cloud Functions.

Google Cloud

Event-driven functions are executed in response to an event like receipt of a new Pub/Sub message or deletion of a file in Cloud Storage.
You implement an event-driven function as a CloudEvent function or a Background function.

CloudEvent functions are:

- Based on the CloudEvents industry standard specification for describing event data.
- Registered with the Functions Framework, an open source library that wraps user functions within a persistent HTTP application.
- Used by 2nd generation Cloud Functions with all language runtimes.
- Used by 1st generation functions with .NET, Ruby, and PHP.

Background functions are:

- Older style event-driven functions that receive event data based on the type of event.
- Used by 1st generation Cloud Functions with Node.js, Python, Go, and Java runtimes.

Features of 2nd gen Cloud Functions

✓ Larger instances

✓ Extensibility and portability

✓ Increased concurrency

✓ Many more event sources

✓ Longer request processing

✓ Standardized event schema

✓ Fast rollbacks and traffic splitting

✓ Improved developer experience

Google Cloud

2nd generation Cloud Functions have the following features:

- Use instances with up to 16 GB of RAM and 4 vCPUs.
- Process up to 1000 concurrent requests with a single function instance to improve cold starts and latency when scaling.
- Run your HTTP functions for up to 60 minutes and event-driven functions for up to 10 minutes to process longer request workloads.
- With a new revision created every time you deploy your function, you can split traffic between different revisions or rollback your function to a prior revision.
- Leverage Cloud Run's scalable container platform to move your function to Cloud Run or to Kubernetes.
- With Eventarc, event-driven functions can be triggered from more than 125 Google, external SaaS event sources, and custom sources by publishing to Pub/Sub.
- Uses the industry-standard CloudEvents format that enables you to simplify your event handling code.
- In Google Cloud console configure your function's triggers, track deployments, test your functions, and view function metrics.

View the [documentation](#) on the comparison between Cloud Functions 1st and 2nd generations.

Use cases

Data Processing

Respond to Cloud Storage events when a file is created, updated, or deleted.

Validate and transform data, process images, and transcode videos.

API Webhooks/APIs

Process webhook and API calls originating from external systems that can send HTTP requests.

Mobile device backend

Develop a mobile device backend responding to events triggered by Firebase, Google's mobile platform for app developers.
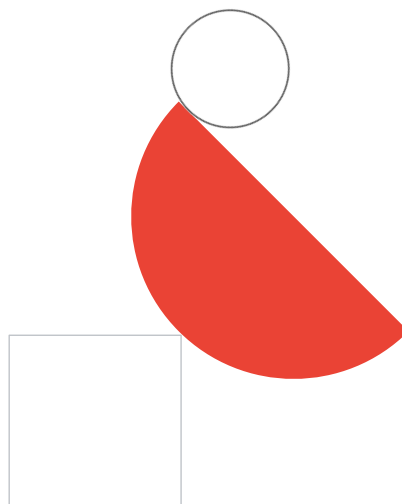
IoT

Stream device data into Pub/Sub, and launch Cloud Functions to process, transform, and store data.

Google Cloud

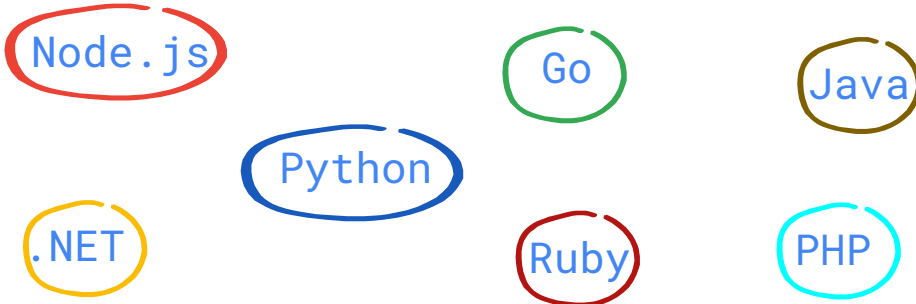Some of the use cases for Cloud Functions include:

- Data processing
- Webhooks and APIs
- Mobile device backend that responds to events triggered by Firebase
- IoT

Languages and regions

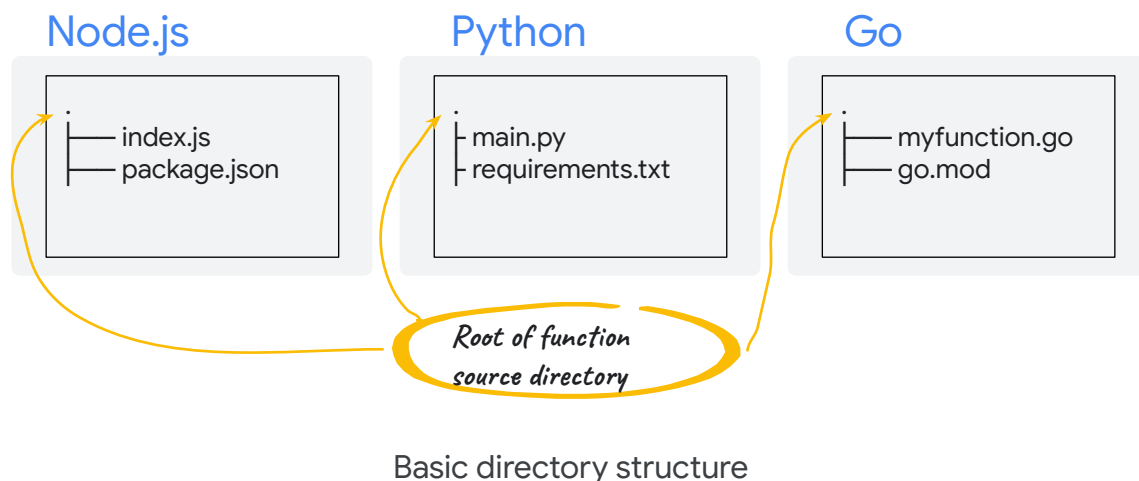Let's review the language runtimes and cloud regions that are currently supported with Cloud Functions.

Language runtimes

Node.js

Go

Java

Python

.NET

Ruby

PHP

Google Cloud

You can develop Cloud Functions in any of these supported languages.

There are specific versions of each language runtime that is supported. Please refer to the Cloud Functions documentation for the version numbers.

Source directory structure

Node.js

```
.
├── index.js
├── package.json
```

Python

```
.
├ main.py
├ requirements.txt
```

Go

```
.
├── myfunction.go
├── go.mod
```

Root of function source directory

Basic directory structure

The language runtime that you choose and the type of function that you want to write determine the structure of your code and function implementation.
For Cloud Functions to locate your function definition, each language runtime has specific requirements for structuring your source code.

Here are the basic directory structures for some of the languages.

For Node.js, by default, Cloud Functions loads source code from a file named index.js at the root of your function directory. To specify a different main source file, use the main field in your package.json file. The package.json file must also include the Functions Framework for Node.js as a dependency.

For Python, Cloud Functions loads source code from a file named main.py at the root of your function directory. The requirements.txt file must include the Functions Framework for Python as a dependency.

For Go, your function must be in a Go package at the root of your project. The go.mod file must include the Functions Framework for Go as a dependency.
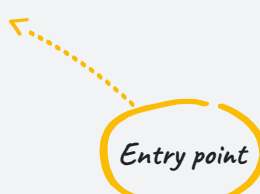
For more details on creating your source code directory structure for your preferred language, refer to the Cloud Functions documentation.

Function entry point

- Code that runs when a function is invoked
- Must be defined in your source code in the *main* file or root package
- Can be a function or class depending on programming language
- Specified when the function is deployed

Node.js

```
const functions =
require('@google-cloud/functions-framework');

//Register an HTTP function with the
//Functions Framework
functions.http('myHttpFunction', (req, res)
=> {
  // Your code here

  // Send an HTTP response
  res.send('OK');
});
```

Entry point

Google Cloud

The function entry point is the code that is executed when the Cloud Function is invoked.

Your source code must define an entry point for your function and must be defined in your main file or root package.

Depending on the programming language, the entry point is a function or a class.

You specify this entry point when you deploy your function.

Cloud Functions locations
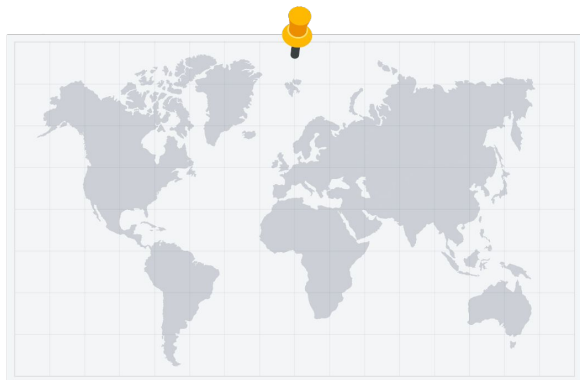
- ✓ Regional
- ✓ Managed by Google
- ✓ Redundantly available across all zones in a region
- ✓ Regional availability:
  - Depends on Cloud Functions generation
  - Affects tier pricing

Google Cloud

https://cloud.google.com/functions/docs/locations

The infrastructure that runs a Cloud Function is located in a specific region and is managed by Google to be redundantly available across all the zones within that region.

When selecting a region to run your Cloud Functions in, your primary considerations should be latency and availability. You can generally select the region closest to your Cloud Function's users, but you should also consider the location of the other Google Cloud products and services that your app uses. Using services across multiple locations can affect your app's latency, and pricing.
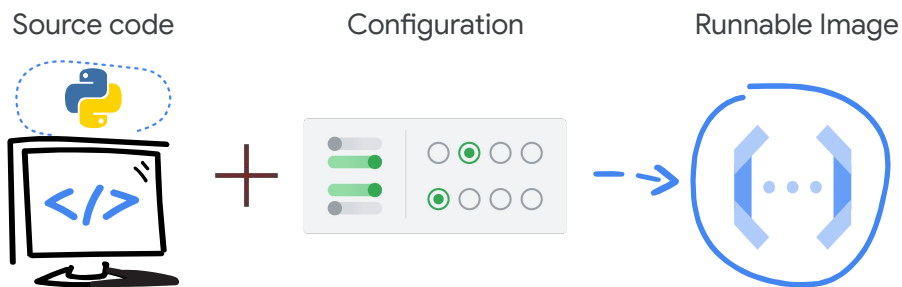
1st and 2nd generation Cloud Functions have different regional availability.

Deploying Cloud Functions
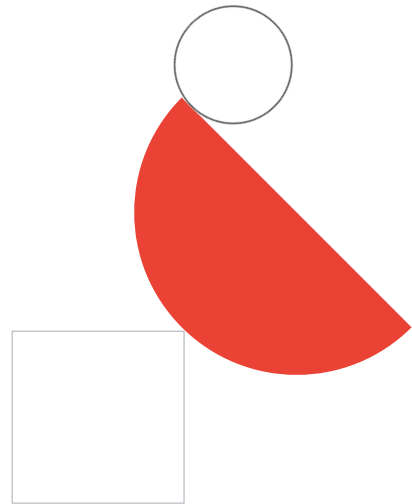
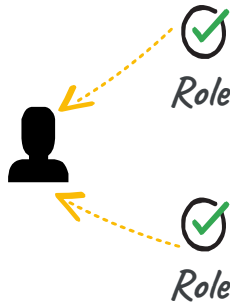Let's review the process to deploy Cloud Functions from source code.

Deployment basics

Source code   Configuration   Runnable Image

The deployment process takes your function source code and configuration settings and builds a runnable image.

Cloud Functions automatically manages the image in order to handle requests to your function.

Required user roles

To be able to deploy Cloud Functions, a user
must have these IAM roles:

- Cloud Functions Developer (or a role with
  the same permissions)
- Service Account User

Google Cloud

A user deploying Cloud Functions must have the Cloud Functions Developer IAM role
or a role that includes the same permissions.

To deploy Cloud Functions, a user must also be assigned the Service Account User
IAM role on the Cloud Functions runtime service account.

Deployment process
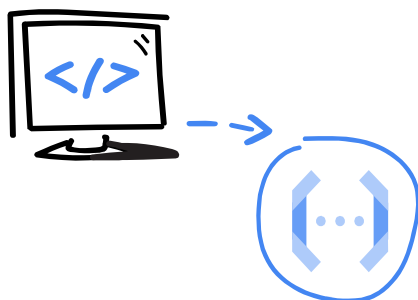
Deploy your function:

- From the Cloud console
- Using Cloud Build
- With the gcloud CLI

gcloud CLI

```
gcloud functions deploy YOUR_FUNCTION_NAME \
[--gen2] \
--region=YOUR_REGION \
--runtime=YOUR_RUNTIME \
--source=YOUR_SOURCE_LOCATION \
--entry-point=YOUR_CODE_ENTRYPOINT \
TRIGGER_FLAGS
```

Google Cloud

You can deploy your function from the Google Cloud console, using Cloud Build, or with the gcloud CLI.

To deploy to 2nd gen Cloud Functions, specify the gen2 flag. If disabled with --no-gen2, Cloud Functions (First generation) will be used.

The region flag specifies the region in which to deploy your function. The runtime flag specifies the language runtime of your function.

The source flag specifies the location of your function source code. The function can be located on your local machine, in Cloud Storage, or in a source repository.

Specify a function or class name in your source code as the entry point of your cloud function with the entry-point flag. The function or class code is executed when your function runs.

To specify the type of trigger and related configuration for your function, provide trigger flags. Refer to the gcloud functions deploy command [documentation](#) for more details.

Deployment sources

You can deploy your function from a:

- Local machine

```
--source=/path/root_directory_of_fun
ction_source_code
```

- Cloud Storage bucket

```
--source=gs://bucket/folder/function
.zip
```

- Source repository (1st gen only)

```
--source=https://source.developers.g
oogle.com/projects/PROJECT_ID/repos/
REPOSITORY_NAME
```
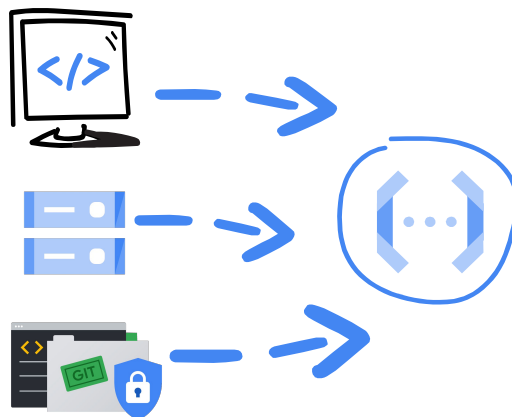
You can specify the location of your function source code for deployment with the source flag.

You can deploy your function from:

- A local machine
  - The value of the source flag is a local file system path to the root directory of the function source code.
  - Optionally use the stage-bucket flag to specify a Cloud Storage bucket to upload your source code to as part of deployment.
  - You can exclude unnecessary files with the .gcloudignore file.
- Cloud Storage
  - The value of the source flag is the cloud storage path to the bucket that contains the function source code packaged as a zip file.
  - Your function source files must be located at the root of the zip file.
  - In Cloud Functions (1st gen), the account performing the deployment needs permission to read from the bucket.
  - In Cloud Functions (2nd gen), the Cloud Functions service agent needs permission to read from the bucket.
- A source repository
  - The value of the source flag is a source repository reference to the location of your function source code.

- You can deploy your function source code from a revision in your repository by specifying revisions/revision name in the source repository path. To specify the location of the source code other than the repository root, append the paths/source_directory_path to the source repository path.
- The Cloud Functions service agent must have the Source Repository Reader (roles/source.reader) IAM role on the repository.
- Deploying from Cloud Source Repositories also enables you to deploy code hosted in a GitHub or Bitbucket repository.

Deployment sources
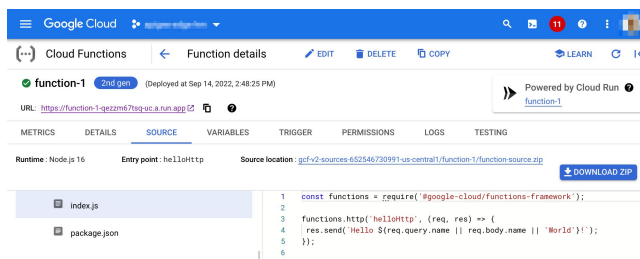
You can also deploy your function from the Google Cloud console inline editor.

Use the editor to:

- View and select the source files in the left pane.
- Create and edit source files in the right pane.

You can also write and deploy a function directly from the Google Cloud console using the provided inline editor.

The editor contains a left pane to view and select source files and the right pane to edit a source file.

Build process
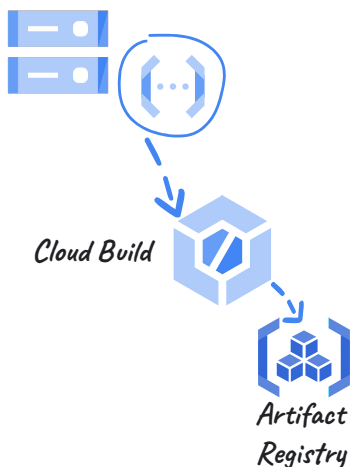
Cloud Functions:

- Stores your function code in Cloud Storage when deployed.

Cloud Build:

- Automatically builds your code into a container image.
- Pushes the image to Container Registry or Artifact Registry.

Build Process:

- Executes in your project.
- Requires the Cloud Build API to be enabled.
- Provides access to all build logs through Cloud Logging.

Cloud Build

Artifact Registry

Google Cloud

When you deploy your function's source code to Cloud Functions, that source is stored in a Cloud Storage bucket.

Cloud Build is a service that executes builds on Google Cloud infrastructure. Cloud Build automatically builds your function source code into a container image and pushes that image to an image registry (either Container Registry or Artifact Registry).

Cloud Functions then accesses this image when it needs to run the container to execute your function.

The process of building the image is entirely automatic and requires no direct input from you.

The Cloud Build API must be enabled for your project. All the resources used in the build process execute in your own user project, and you have access to all the build logs through Cloud Logging.
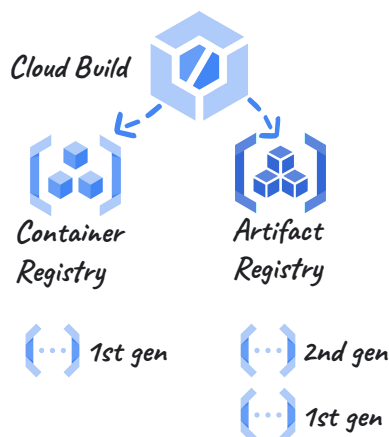
Image registry

Artifact Registry:

- A Google Cloud service used to store and manage software artifacts like container images.
- Integrates with Cloud Build.
- Used by Cloud Functions (2nd gen and 1st gen) to store function images.

Container Registry:

- A private container registry for Docker images.
- Used by Cloud Functions (1st gen) to store function images by default.

Cloud Build

Container Registry

Artifact Registry

1st gen

2nd gen

1st gen

Cloud Functions (2nd gen) exclusively uses Artifact Registry to store the images built from your function source code.

Cloud Functions (1st gen) uses Container Registry by default, but also lets you use Artifact Registry.

Container Registry is a service that provides a private container image registry that supports Docker images and OCI image formats.

Artifact Registry is a service that is used to store and manage software artifacts in private repositories, including container images, and language packages. It expands on the capabilities of Container Registry and is the recommended container registry for Google Cloud. Artifact Registry integrates with Cloud Build to store the packages from your builds.
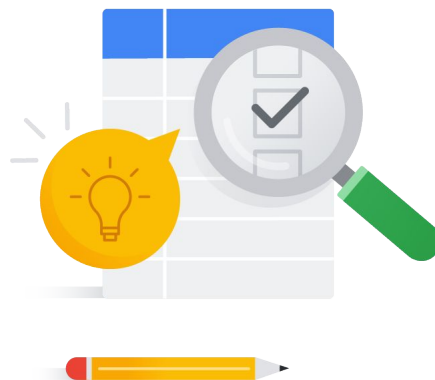
Lab

🕐 45 min   ☺ Individual

Developing and Deploying Cloud Functions

Create simple functions with Google Cloud console and the gcloud CLI.

Google Cloud

In this lab, you create and deploy simple Cloud Functions with Google Cloud Console and the gcloud CLI.

Lab Instructions

🕐 45 min   **☺ Individual**   **🔖 Tasks**

- Write a function that responds to HTTP requests, and deploy the function from Google Cloud console.
- Write a function that responds to Cloud Storage events, and deploy the function using the gcloud CLI.
- Write unit tests for your function and test your function before deployment.
- Deploy multiple revisions of a function.

1   Write an HTTP function.

2   Write an event-driven function for Cloud Storage events.

3   Write unit tests to test your function locally.

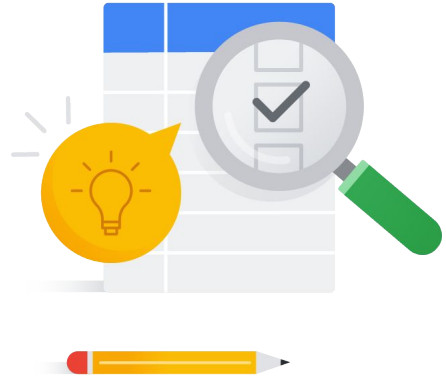**4**   Deploy multiple revisions of a function.

Google Cloud

Quiz

5 min **Group**

Introduction to Cloud Functions

Google Cloud

Let's do a short quiz on this module.

Quiz | Question 1

Question

Which of these statements about Cloud Functions are correct? Select three.

Cloud Functions:

A.   Is a scalable functions-as-a-service platform.

B.   Can be used to extend Cloud services.

C.   Require servers or VMs to be provisioned.

D.   Is integrated with Cloud Logging.

E.   Can only be invoked by sending HTTP requests.

Google Cloud

Quiz | Question 2

Question

What are some of the features and benefits of using Cloud Functions? Select four.

Cloud Functions:

A.   Are triggered by HTTP requests and events from cloud services.

B.   Uses a fixed pricing model.

C.   Can be integrated with Cloud databases.

D.   Can be locally developed and tested.

E.   Supports seamless authentication with IAM.

Quiz | Question 3

Question

An HTTP function: (Select three)

A. Must send back an HTTP response.

B. Can be used to implement a webhook.

C. Can only be invoked with authentication credentials.

D. Is triggered by a request made to its URL endpoint.

Quiz | Question 4

Question

An Event-driven function: (Select three)

A. Can only be implemented as a CloudEvent function.

B. Uses event triggers for various services like Pub/Sub, and Cloud Storage.

C. Responds to events that occur in your cloud infrastructure.

D. Can be triggered from Eventarc sources if it is 2nd generation.

Google Cloud

Review:
Introduction to
Cloud Functions
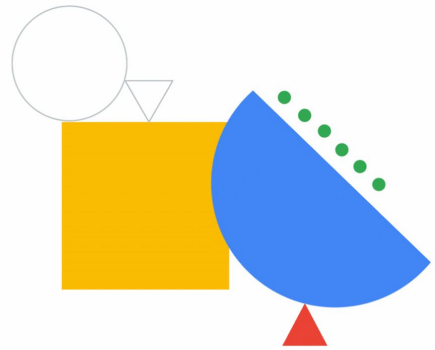
In this module, we introduced Cloud Functions and discussed the types of Cloud Functions, their features and their benefits.

We reviewed some of the use cases for Cloud Functions, and completed a lab to create functions in Google Cloud console, and with the gcloud CLI.