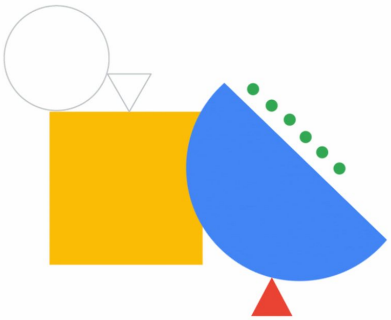
Google Cloud

Securing Cloud Functions



In this module, we discuss how to secure Cloud Functions with identity and network-based access controls.

You learn how to authenticate and authorize access to cloud functions and how to protect Cloud Functions and related data with encryption keys.

Agenda

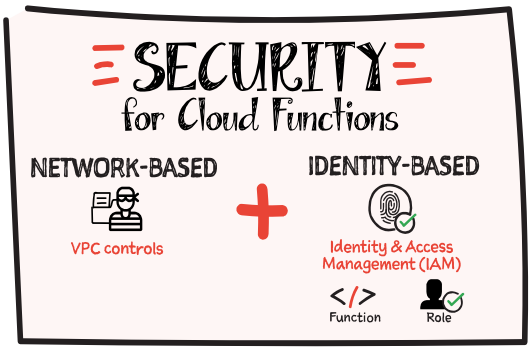
01	Securing access to Cloud Functions
02	Authenticating to Cloud Functions
03	Protecting Cloud Functions
04	Quiz

In this module, you learn about:

- Securing access to Cloud Functions
- Authenticating to Cloud Functions
- Protecting Cloud Functions

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

Securing access to Cloud Functions



Secure access to Cloud Functions with:

- Identity-based access controls
- Network-based access controls

Secure access to Cloud Functions with:

- Identity-based access controls
- Network-based access controls

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

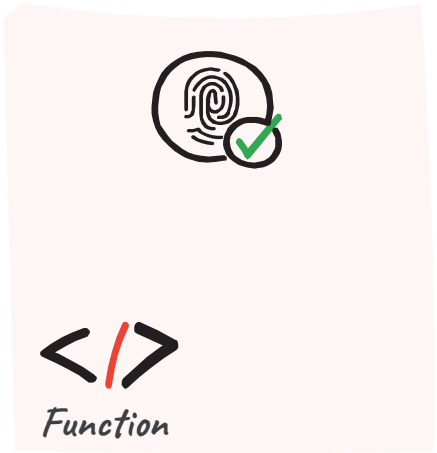
Identity-based access control

To secure access to Cloud Functions with identity:

- Validate the identity (authentication).
- Evaluate the permissions granted to the identity (authorization).

Cloud Functions are deployed as:

- Private (by default)
  - Requires authentication
- Public
  - Does not require authentication



The first step in the process is authentication, to validate the identity credential and make sure that the requestor is who it says it is. Once the requestor's identity has been authenticated, its level of access or permissions can be evaluated. This step is called authorization.

By default, functions are deployed as *private*, and require authentication. You can also deploy a function as *public* and not require authentication.

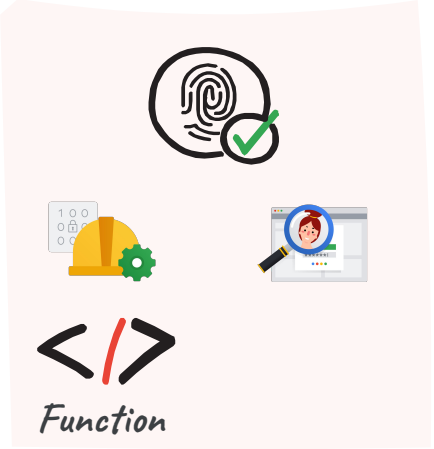
Types of identities

Cloud Functions supports two types of identities:

- Service accounts
  - Identity of an application, VM, or function
- User accounts
  - Represent people

For authentication, Cloud Functions uses tokens:

- OAuth 2.0 access tokens
- ID tokens (OIDC)



Cloud Functions supports two different kinds of identities:

- Service accounts: Serve as the identity of a non-person, like a function, application or a VM.
- User accounts: Represent people - either as individual Google Account holders or as part of a Google Group.

To limit the potential damage that might occur if the service or user account credential leaked, clients create a token based on the credential. The token has a limited lifetime, is passed with the request, and serves to safely authenticate the account.

Cloud Functions uses two types of tokens:

- OAuth 2.0 access tokens: Used to authenticate API calls
- ID tokens. Used to authenticate calls to developer-created code, for example, a function calling another function.

The tokens are created using the OAuth 2.0 framework and Open Identity Connect (OIDC).

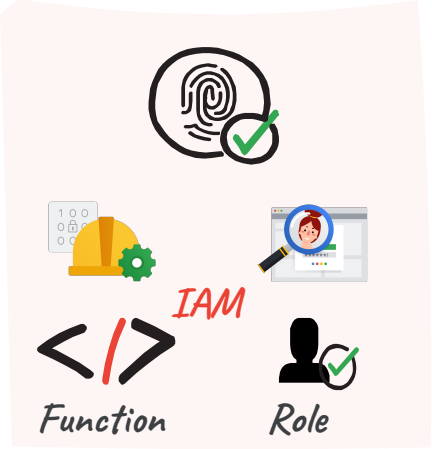
Authorization

To authorize the requesting identity:

- Cloud Functions uses Google Cloud's Identity and Access Management (IAM) service.
- IAM evaluates the permissions of the identity account using roles.

A Role is:

- A set of individual permissions.
- Assigned to an account directly or through a policy.



Once the identity of the requesting entity has been confirmed, the permissions of the identity are evaluated. The permissions are granted when the authenticated account was set up.

Cloud Functions uses Identity and Access Management (IAM) to evaluate permissions using roles.

A role is a set of individual permissions that are grouped together and assigned to the account, either directly or through a policy configuration.

Each individual permission in the role set usually corresponds to a single REST API call exposed by the requested service. For more information on this process, see Authorizing Access with IAM.

Authorizing access for administration

To authorize identities/principals to perform administrative actions on Cloud Functions:

- Add the service or user account to the function.
- Assign IAM roles to the service or user account.

Cloud Functions Roles:

- Cloud Functions Admin
- Cloud Functions Developer
- Cloud Functions Invoker
- Cloud Functions Viewer

gcloud CLI

```
gcloud functions \
add-iam-policy-binding FUNCTION_NAME \
  --member=account_email \
  --role=ROLE
```


To authorize identities to perform administrative actions like creating, updating, and deleting functions, you add principals/identities (a user or service account email) to the function with the appropriate IAM roles.

Roles include permissions that define a set of allowed actions that can be performed on the function.

You can authorize access to a function in the Google Cloud console or with the gcloud CLI.

For a complete list of user roles and service accounts used with Cloud Functions, see the access control with IAM documentation.

For a list of permissions in each role, see the Cloud Functions IAM Roles documentation.

Authorizing access for invocation

To test a function:

- Have a user account role with `cloudfunctions.functions.invoke` permission on the function.
- Generate an ID token for the user account using the gcloud CLI.
- Provide the token in the Authorization header of the request to the function.

gcloud CLI

```
TOKEN=$(gcloud auth print-identity-token)

curl -H "Authorization: bearer $TOKEN" \
https://REGION-PROJECT_ID.cloudfunctions.net/
FUNCTION_NAME
```

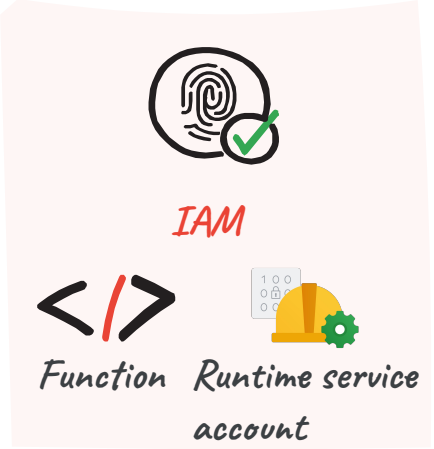

Event-driven functions can only be invoked by the event source that they are subscribed to.

HTTP functions can be invoked by different types of identities, for example, a developer who is testing the function, or a service that uses the function. These identities must provide an ID token with the appropriate permissions for authentication and authorization.

When testing a function as a developer, you must have a user account to access Cloud Functions with a role that contains the appropriate permissions on the function being invoked. Then, generate an ID token using this account and pass the token in the Authorization header of the request to the function.

Function identity

- Every function is associated with a service account that serves as its identity known as the runtime service account.
- Cloud Functions uses a default runtime service account:
  - App Engine default service account (1st gen)
  - Compute Engine default service account (2nd gen)
  - Use for development and testing purposes.
- For production:
  - Use a different individual runtime service account for each function.
  - Grant the service account minimum permissions.



Functions often need access to other resources in Google Cloud to do their work. Every function is associated with a service account that serves as its identity when the function accesses other resources.

The service account that a function uses as its identity is known as its *runtime service account*.

Cloud Functions uses a default runtime service account for function identity:

- Cloud Functions (1st gen) use the App Engine default service account.
- Cloud Functions (2nd gen) use the Compute Engine default service account.

Use the default service account for testing and development purposes only.

For production, specify a different individual runtime service account when deploying a function, and grant the runtime service account only the minimum set of permissions required to achieve its goal.

Authorizing function to function calls

To authorize function to function calls:

- Grant the `roles/cloudfunctions.invoker` role to the calling function's service account on the receiving function.
- Generate a Google-signed ID token with:
  - Audience field (aud) set to the URL of the receiving function.
- Provide the token in the Authorization header of the request to the function.

gcloud CLI

```
gcloud functions add-iam-policy-binding
RECEIVING_FUNCTION \

--member='serviceAccount:CALLING_FUNCTI
ON_IDENTITY' \
  --role='roles/cloudfunctions.invoker'
```

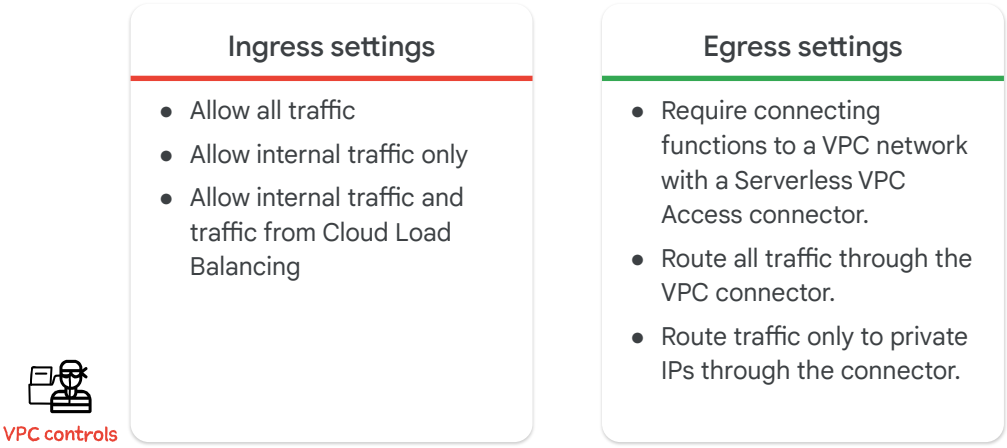

When building services that connect multiple functions, ensure that each function can only send requests to a specific subset of your functions.

To configure a receiving function to accept requests from a specific calling function, grant the Cloud Functions Invoker (roles/cloudfunctions.invoker) role to the calling function's service account on the receiving function.

The calling function must also provide a Google-signed ID token for authentication, with the audience field (aud) set to the URL of the receiving function. The ID token must be sent in an Authorization header in the request to the function.

To learn more about generating the token, view the documentation.

Network-based access control



You can also limit access by specifying network settings for Cloud Functions. Network settings enable you to control network ingress and egress to and from individual functions.

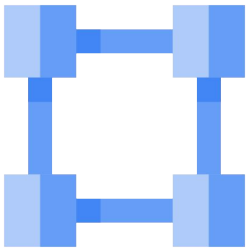
Ingress settings restrict whether a function can be invoked by resources outside of your Google Cloud project or VPC Service Controls service perimeter.

You can choose to allow all traffic, allow internal traffic only, or allow internal traffic and traffic from Cloud Load Balancing.
Internal traffic is traffic from Workflows and VPC networks in the same project or VPC Service Controls perimeter.

Egress settings control the routing of outbound HTTP requests from a function. To specify egress settings, you must connect functions to a VPC network using a Serverless VPC Access connector.

Egress settings control which types of traffic are routed through the connector to your VPC network. You can set egress settings to route all outbound traffic from the function through the connector, or, route only requests to private IPs through the connector.

Using VPC Service Controls



VPC controls

- Set up organization policies that control network settings for functions in the VPC service perimeter to:
  - Only allow internal ingress traffic to Cloud Functions.
  - Require a VPC connector.
  - Restrict allowed VPC connector egress settings.

You can use VPC Service Controls with Cloud Functions to add additional security to your functions.
To do this, you:

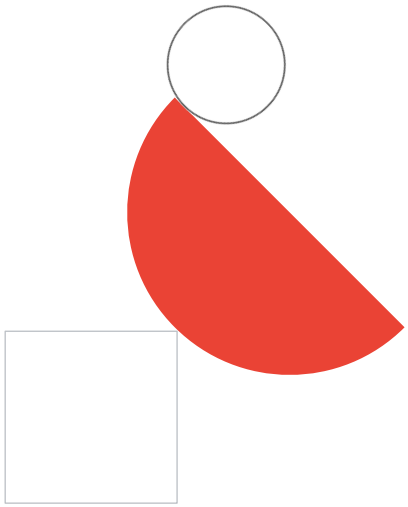
- Create a service perimeter.
- Add one or more projects to the perimeter. For Shared VPCs, add the host and service projects to the perimeter.
- Restrict the Cloud Functions API by setting up organization policies that control the network settings for functions in the service perimeter.

With the organization policies in place:

- HTTP functions will only accept traffic that originates from a VPC network within the service perimeter.
- All functions must use a Serverless VPC Access connector.
- Functions must route all egress traffic through your VPC network.

View the VPC Service Controls with Cloud Functions documentation for more details.

Protecting Cloud Functions



Customer-managed encryption keys

- Use Cloud KMS to create and manage encryption keys.
- Keys are owned by you, and known as customer-managed encryption keys (CMEK).
- To protect data associated with a function, deploy the function with a CMEK.
- If the key is disabled or destroyed, the function data is inaccessible.

You can use Cloud Key Management Service (KMS) to create and manage encryption keys that protect your Cloud Functions and related data at rest.
The keys are known as customer-managed encryption keys (CMEK) and are owned by you and not controlled by Google.
They can be stored as software keys, in an HSM cluster, or externally.

Deploying a function with a CMEK protects the data associated with it by using an encryption key that only you can access.
If the key is disabled or destroyed, no one (including you) can access the data that is protected by the key.

CMEKs

CMEKs encrypt:


Function source code uploaded for deployment and stored by Google in Cloud Storage.


The container image built from your function source code.


Each instance of the function that is deployed.


Data at rest that is used for internal event transport channels.

The following types of Cloud Functions data are encrypted when using a CMEK:

- Function source code uploaded for deployment and stored by Google in Cloud Storage, and used in the build process.
- The results of the function build process, including:
  - The container image built from your function source code.
  - Each instance of the function that is deployed.
- Data at rest for internal event transport channels.

Setting up CMEK for Cloud Functions

To set up CMEK for Cloud Functions:

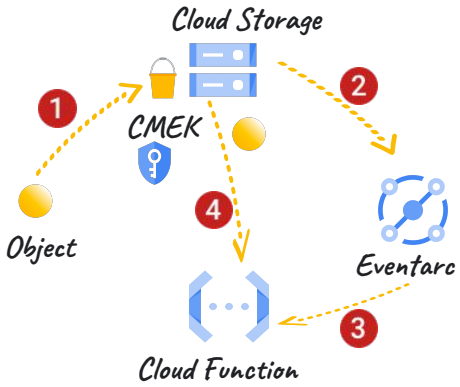
- Create a single-region encryption key.
- Create a CMEK-protected Artifact Registry repository to store your function images.
- Grant these service accounts access to the key:
  - Cloud Functions service account
  - Artifact Registry service account
  - Cloud Storage service account
- Enable CMEK on your function.

To set up CMEK for Cloud Functions:

- Create a single-region encryption key.
- To store your function images, create an Artifact Registry repository with CMEK enabled. Use the same key for the repository as the one used to enable CMEK for the function.
- Grant the Cloud Functions, Artifact Registry, and Cloud Storage service accounts access to the key.
- Enable CMEK on your function.

CMEK use case

1. An object is uploaded to a Cloud Storage bucket with a CMEK enabled on the bucket.
2. An event is triggered in Eventarc after the object is created or finalized.
3. A Cloud Function that is configured with the trigger is invoked.
4. The function retrieves the decrypted object from Cloud Storage.



A sample use case of using CMEKs involves storing and accessing objects in Cloud Storage buckets.

You can use customer-managed encryption keys on individual objects, or configure your bucket to use a key by default on all new objects added to the bucket.

Objects that are uploaded are stored encrypted with the CMEK.

A 2nd gen Cloud Function can be triggered from Eventarc whenever there are changes to objects in the Cloud Storage bucket.

The function can then retrieve the decrypted object from the bucket in Cloud Storage. You can also implement a function to encrypt individual objects and upload them to Cloud Storage.

To use CMEKs, you must grant your project's Cloud Storage service account or service agent access to the key.

Grant access to the key

To grant service accounts access to the key:

- Add each service account as a principal of the key.
- Grant the service account the Cloud KMS CryptoKey Encrypter/Decrypter role.

gcloud CLI

```
gcloud kms keys add-iam-policy-binding KEY \
--keyring KEY_RING \
--location LOCATION \
--member serviceAccount:SERVICE_AGENT_EMAIL \
--role \
roles/cloudkms.cryptoKeyEncrypterDecrypter
```


To grant service accounts access to the key, add each service account as a principal of the key and then grant the service account the Cloud KMS CryptoKey Encrypter/Decrypter role.

The service account email used is that of the:

- Cloud Functions Service Agent
- Artifact Registry Service Agent
- Cloud Storage Service Agent

Enable CMEK for a function

To enable CMEK for a function:

- Deploy the function specifying the key and Artifact registry repository.
- Use the Google Cloud console or gcloud CLI.

Cloud Functions always uses the **primary** version of a key.

A particular key version cannot be specified when enabling CMEK.

gcloud CLI

```
gcloud functions deploy FUNCTION \
--kms-key KEY \
--docker-repository=REPOSITORY
```


After setting up an Artifact Registry repository with CMEK enabled and granting the required service agents access to your key, you can enable CMEK for your function by deploying the function and specifying the key and repository.

Cloud Functions uses the primary version of a key for CMEK protection. You cannot specify a particular key version to use when enabling CMEK for your functions.

Destroying or disabling keys

If a key is destroyed or disabled:

- Active instances of functions protected by the key are not shut down.
- Executions of these functions that are already in progress will continue to run.
- Any new executions of active instances may fail.
- Executions that require new instances will fail.

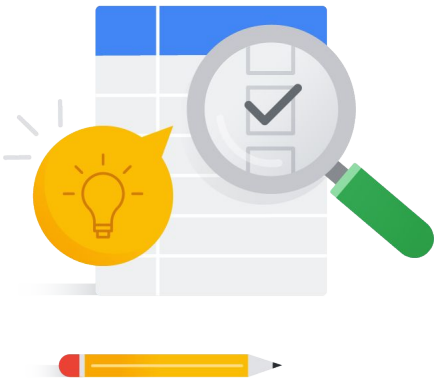
If a key is destroyed or disabled, or the requisite permissions on it are revoked:

- Active instances of functions protected by the key are not shut down.
- Executions of these functions that are already in progress will continue to run.
- New executions will fail as long as Cloud Functions does not have access to the key.
- Executions that require new function instances will fail.

Quiz

5 min | Group

Securing Cloud Functions



Let’s do a short quiz on this module.

Quiz | Question 1

Question

To limit access to Cloud Functions, what methods can you use? Select two.

A. Use HTTPS.
B. Use identity-based access controls.
C. Use a username and password.
D. Use network-based access controls.
E. Use an encryption key.

Quiz | Question 2

Question

Cloud Functions uses IAM to authorize the requesting identity. Which predefined IAM roles are used by Cloud Functions? Select four.

A. Cloud Functions Admin

B. Cloud Functions Reader

C. Cloud Functions Developer

D. Cloud Functions Invoker

E. Cloud Functions Viewer

Quiz | Question 3

Question

Which statements about function identity are correct? Select three.

A. Every function is associated with a runtime service account that serves as its identity.

B. Cloud Functions uses the App Engine default service account as the default runtime service account for 1st gen functions.

C. In production environments, you should use a runtime service account for a function's identity with the maximum set of permissions.

D. Cloud Functions uses the Compute Engine default service account as the default runtime service account for 2nd gen functions.

Quiz | Question 4

Question

Which network setting allows Cloud Functions to route all outbound traffic from a function through a VPC network?

A. Ingress setting to allow all traffic.

B. Egress setting to route traffic only to private IPs through a Serverless VPC connector.

C. Ingress setting to allow internal traffic and traffic from Cloud Load Balancing.

D. Egress setting to route all traffic through a Serverless VPC connector.

Quiz | Question 5

Question

A customer-managed encryption key (CMEK) is used to encrypt a function and its related data. What is the impact when such a key is disabled or destroyed?

A. Executions that require new function instances will fail.

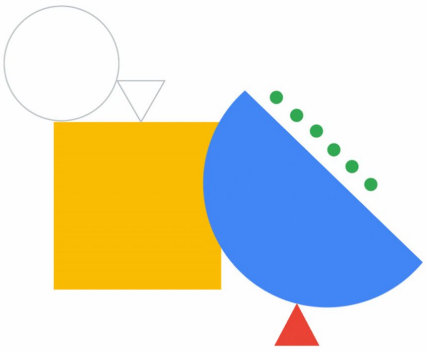
B. Active instances of the function are shut down.

C. There is no impact to the function.

D. Executions of the function that are already in progress are terminated.

Google Cloud

Review: Securing Cloud Functions



In this module, we discussed how to securely access and authenticate to Cloud Functions with identity and network-based access controls.

You also learned about function identity, and how to protect Cloud Functions with customer-managed encryption keys.