About API Gateway

cloud.google.com/api-gateway/docs/about-api-gateway

Web-based services today provide a huge variety of functionality, meaning everything from map, weather, and image services, to games, auctions, and many other service types. Service providers have many options for how to implement, deploy, and manage their services. For example, one service might be developed in Java or .NET, while another uses Node.js.

Backend implementations can also vary for a single service provider. A service provider might have legacy services implemented using one architecture, and new services implemented using a completely different architecture.

Regardless of implementation, web-based services all require a way to make the services available to app developers. Often these services are exposed as a set of HTTP endpoints. Depending on the service, the endpoint might also return data, formatted as XML or JSON, to the client app.

About Google Cloud Platform services

When developing your services on the Google Cloud Platform (GCP), you have many options for how you implement the services, such as Cloud Functions, Cloud Run, and App Engine standard environment. The flexibility of GCP means you choose the correct backend architecture for your service requirements.

App developers are the customers of backend services. App developers consume your services to implement apps for mobile devices or tablets, through apps running in a browser, or through any other type of app that can make a service request.

Exposing services publicly over the web can be challenging. To be successful, a service provider must:

- Authenticate access to the service
- Secure data transport between clients and the service
- Protect the service from malicious attacks
- Scale the service as usage increases or decreases
- Provide the backend operations team with a way to monitor and track service usage
- Track usage to provide accurate billing information

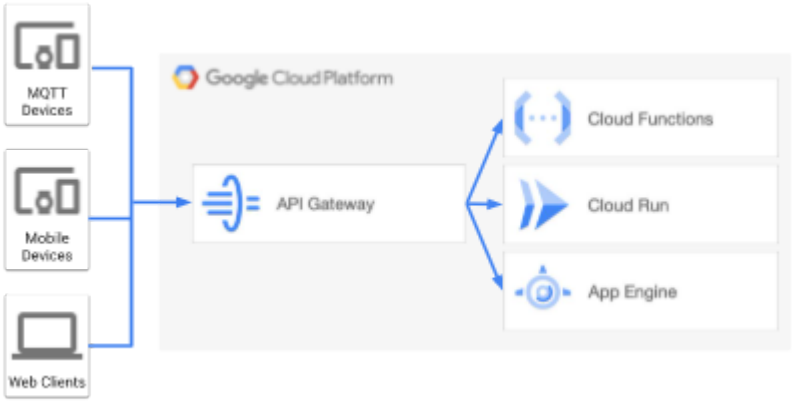
Also, if your services use different interfaces and protocols, then accessing these services can be a challenge to app developers. Developers must not only learn and understand each service interface, but they must also monitor the different services for changes, and then update and redeploy apps as necessary.

API Gateway

API Gateway enables you to provide secure access to your services through a well-defined REST API that is consistent across all of your services, regardless of service implementation. A consistent API:

- Makes it easy for app developers to consume your services
- Enables you to change the backend service implementation without affecting the public API
- Enables you to take advantage of the scaling, monitoring, and security features built into the Google Cloud Platform (GCP)

The following image shows app developers making requests to your backend services through API Gateway:



Using API Gateway, app developers consume your REST APIs to implement apps. Because all APIs are hosted on API Gateway, app developers see a consistent interface across all backend services.

By deploying your APIs on API Gateway, you can update the backend service, or even move the service from one architecture to another, without having to change the API. As long as the API to your service stays consistent, app developers will not have to modify deployed apps because of underlying changes on your backend.

API Gateway is a distributed API management system that also provides hosting, logging, monitoring, and other features to help you create, share, maintain, and secure your APIs. API Gateway is natively integrated with GCP and handles all the tasks involved in processing concurrent API calls, including traffic management, authorization, and monitoring.

What is an API?

An API is an interface that makes it easy for one application to consume capabilities or data from another application. By defining stable, simple, and well-documented entry points, APIs enable developers to easily access and reuse application logic built by other developers.

For example, the following table describes an example of a REST API that could return information about a book:

Property	Value	Description
URL	https://www.mybooksapi.com/books/info	Return the title, author, and publishing date of a book based on its International Standard Book Number (ISBN).
HTTP Verb	GET	Make a GET request to the API.
Query param	`isbn`	Pass the ISBN number of the book, meaning the book's ID.
Response data	`{ "title" : "book_title", "author" : "author_name", "published" : "publish_date" }`	JSON object containing book details.
Response code	200	Request successful.

Using this information, you could make the following cURL request to this API to get information about a book:

```
curl -X GET https://www.mybooksapi.com/books/info?isbn=0385504217
```

Because this service has a well-defined API, including a description of data formats and HTTP response codes, the app developer does not need to know anything about the underlying implementation of the backend service.

Since applications that consume APIs are sensitive to changes, APIs also imply a contract between API providers and API consumers. The contract assures that over time the API will change in a predictable manner. For example, the book API might be updated to add additional query parameters, such as `title` or `author`, or change the response JSON to add additional information about the book.

Defining an API

You define an API deployed on API Gateway as an OpenAPI 2.0 spec. The key components of an API definition include:

- The URL, or entry point, of the backend service
- The data format of any data passed on a request to the API
- The data format of any data returned by the service in the response from the API
- The authentication mechanism used to control access to the service

After you define your API, use the gcloud command line interface to upload it to an API config on GCP:

Deploying an API config on API Gateway

To create your API, you deploy the API config on API Gateway. Use the `gcloud` command to deploy the API config:

After the API config is deployed, your clients can make REST calls to the API.

Managing an API

Once deployed and running, you can monitor API activity, such as usage metrics and logs. When a client makes a request to your API, API Gateway logs information about the request and response. API Gateway also tracks latency, traffic, and errors.

Over time, you might want to update a deployed API to add new functionality, improve performance, or to fix issues with the API. To update a deployed API, you simply update the OpenAPI spec for the API definition, then upload and redeploy the API.

Controlling API access

API Gateway lets you configure your API to require authentication before the client can access the API. Currently, API Gateway supports the same authentication mechanism and syntax as used by Cloud Endpoints, including using:

You can also use the Google Cloud Platform Console to share your API with other developers so they can enable your API and generate API keys to call it.

Along with defining an authentication mechanism to verify a user's identity, your API also needs to decide what the authenticated user can do with your API. For more information, see the GCP Auth guide.

What's next

- API Gateway Architecture