Managing APIs with Google Cloud Endpoints

Developing Applications with Google Cloud Platform

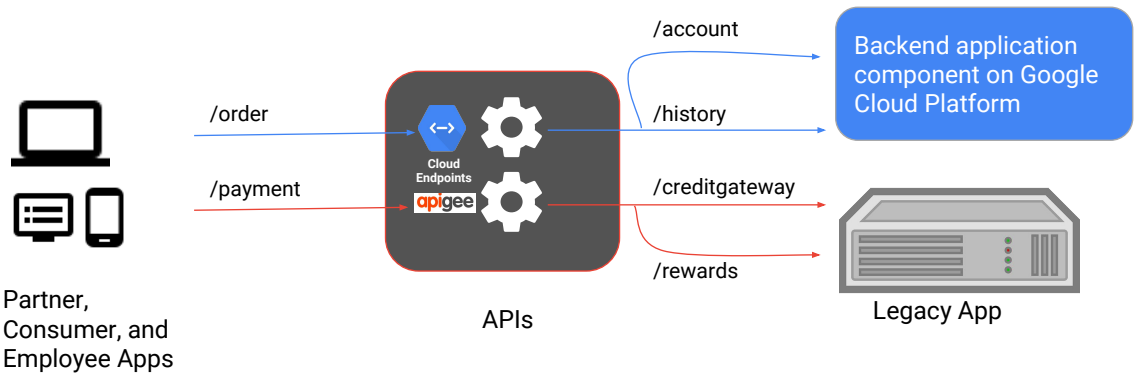
CLOUD ENDPOINTS

DEPLOYING AN API FOR THE QUIZ APPLICATION

Google Cloud

Version 2.0
Last modified: 2017-09-25

Implement an API gateway to make backend functionality available to consumer applications



An API gateway enables clients to retrieve data from multiple services with a single request. An API gateway creates a layer of abstraction and insulates the clients from the partitioning of the application into microservices.

You can use Google Cloud Endpoints to implement API gateways. Additionally, the API for your application can run on backends such as Google App Engine, Google Container Engine, or Compute Engine.

If you have legacy applications that cannot be refactored and moved to the cloud, consider implementing APIs as a facade or adapter layer. Each consumer can then invoke these modern APIs to retrieve information from the backend instead of implementing functionality to communicate using outdated protocols and disparate interfaces.

Using the Apigee API platform, you can design, secure, analyze, and scale your APIs for legacy backends. For more information about the Apigee API platform, see:
Getting Started: http://docs.apigee.com/api-services/content/what-apigee-edge
Training: http://academy.apigee.com/index.php

Images:
https://pixabay.com/en/computer-network-router-server-159829 /
https://material.io/icons/

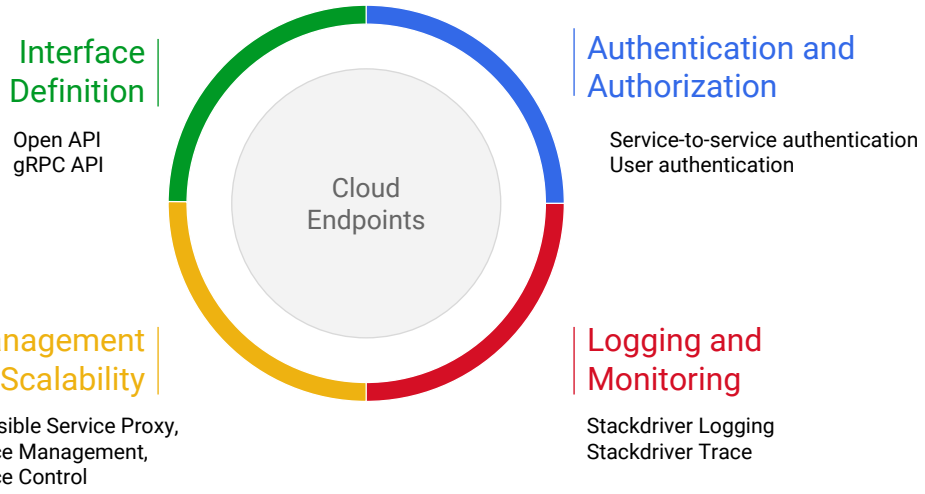
Deploying and managing API on your own can be difficult

Google Cloud

There are a few issues to consider when deploying and managing APIs on your own.

- What language or format will you use to describe the interface? For example, will you use a format such as WSDL or Open API to express the schema and constraints of the API?
- How will you authenticate services and users who invoke your API?
- How will you ensure that your API scales to meet demand?
- Does your infrastructure log details about API invocations and provide monitoring metrics?

Cloud Endpoints enables you to deploy and manage APIs



Google Cloud

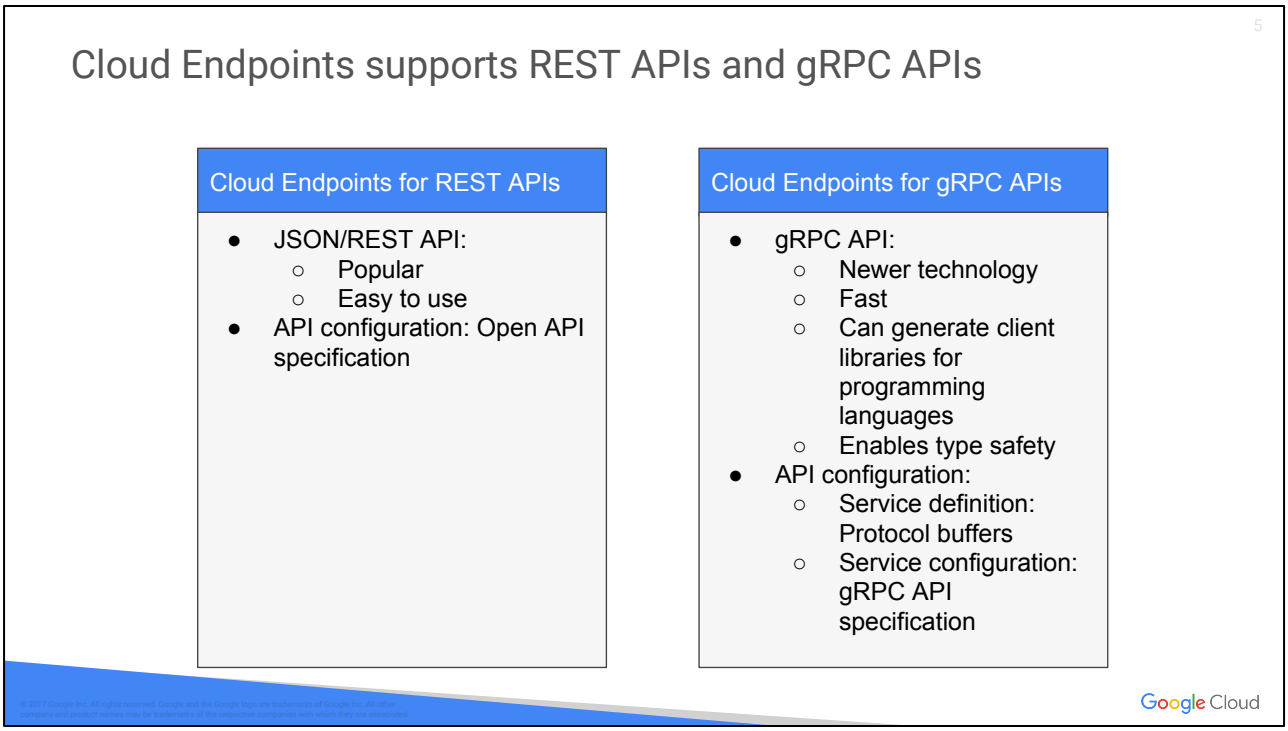
Cloud Endpoints provides the infrastructure support needed to deploy and manage robust, secure, and scalable APIs.

Cloud Endpoints supports the Open API specification and gRPC API specification.

Cloud Endpoints supports service-to-service authentication and user authentication with Firebase, Auth0, and Google authentication.

The Extensible Service Proxy, Google Service Management, and Google Service Control together validate requests, log data, and handle high volumes of traffic.

Using Cloud Endpoints, Stackdriver Logging and Stackdriver Trace, you can view detailed logs, trace lists, and metrics related to traffic volume, latency, size of requests and responses, and errors.



JSON/HTTP 1.1–based REST APIs are popular and easy to use. To enable Cloud Endpoints for REST APIs, create the API configuration in a YAML file based on the Open API specification.

gRPC is a newer, faster technology. You can generate client libraries for various programming languages. Your application can then make type-safe, remote server calls as if they were local calls. To enable Cloud Endpoints for gRPC APIs, create your service definition by using protocol buffers, and then create a service configuration by using the gRPC API specification.

Cloud Endpoints supports transcoding of HTTP/JSON calls into gRPC. You clients can access your gRPC API by using HTTP/JSON.

For more information about Cloud Endpoints for REST APIs, see https://cloud.google.com/endpoints/docs/openapi/open-api-spec.

For more information about gRPC, see:

- gRPC: https://grpc.io/
- Announcing gRPC Alpha for Google Cloud Pub/Sub: https://cloud.google.com/blog/big-data/2016/03/announcing-grpc-alpha-for-google-cloud-pubsub
- Cloud Endpoints for gRPC APIs: https://cloud.google.com/endpoints/docs/grpc/about-grpc

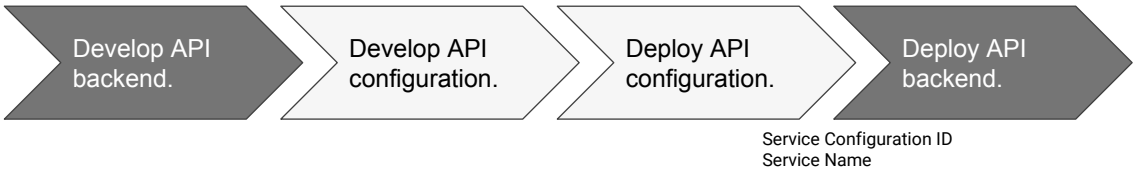
- Transcoding HTTP/JSON to gRPC:
  https://cloud.google.com/endpoints/docs/grpc/transcoding

Cloud Endpoints for REST APIs

Google Cloud

In the remainder of this presentation, we will focus on API management for REST APIs.

Develop and deploy your API configuration and API backend



Google Cloud

In the remainder of this presentation, we will focus on API management for REST APIs.

Let's dive into Google Cloud Endpoints for REST APIs.

This diagram illustrates the high-level steps to make your API backend available as Cloud Endpoints API.

After you develop your REST API backend, create an API configuration file that describes your Cloud Endpoints API.

Deploy the API configuration by using the `gcloud service-management deploy` command. The `gcloud` command returns the service configuration ID and service name. Specify this service configuration ID and service name in your API backend's configuration file, such as the app.yaml for App Engine flexible environment deployments.

Finally, deploy the API backend.

Use the Open API specification as the Interface Definition Language

```
swagger: "2.0"
info:
  description: "My new API."
  title: "Cloud Endpoints Example"
  version: "1.0.0"
host: "quiz-api.endpoints.YOUR-PROJECT-ID.cloud.goog"
basePath: ...
paths: ...
securityDefinitions: ...
```

Interface Definition

Google Cloud

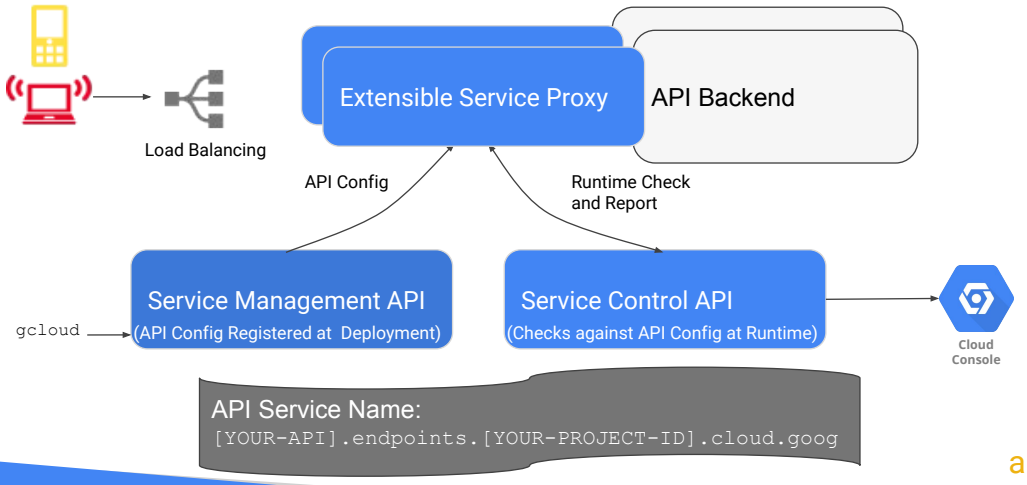
Create an API configuration file that describes your Cloud Endpoints API. The API configuration is a YAML file that describes the API using the Open API specification. The Open API specification and its extensions enable you to describe the surface of the API and security definitions. You can specify security definitions to for user authentication and service-to-service authentication.

For more information about the schema of the objects in the Open API Specification, see: https://github.com/OAI/OpenAPI-Specification.

Various tools are available to help you create and manage your Open API specification. For more information, see:

- Swagger Editor: http://editor.swagger.io/
- Swagger Tools and Integrations: https://swagger.io/open-source-integrations/
- OpenAPI / Swagger Resource List for API Developers: https://blog.runscope.com/posts/openapi-swagger-resource-list-for-api-developers

Service Management API, Service Control API, and Extensible Service Proxy form the core of Cloud Endpoints



Management and Scalability

Google Cloud

When you deploy the API configuration, the configuration is registered with the Google Service Management API and shared with the Extensible Service Proxy.

Service Management uses the host value in your deployment configuration file to create a new Cloud Endpoints service with the name [YOUR-API].endpoints.[YOUR-PROJECT-ID].cloud.goog (if it does not exist), and then configures the service according to your OpenAPI configuration file. Cloud Endpoints uses DNS-compatible names to uniquely identify services. Because projects in Google Cloud Platform are guaranteed to have a globally unique name, you can use your project name to create a unique API service name such as `quiz-api.endpoints.my-project-id.cloud.goog` . You can also map your own DNS name to your API.

The Extensible Service Proxy is an NGINX-based proxy that runs in front of the API backend and injects Cloud Endpoints functionality such as authentication, monitoring, and logging. The proxy uses techniques such as heavy caching and asynchronous calls to remain lightweight and highly performant.

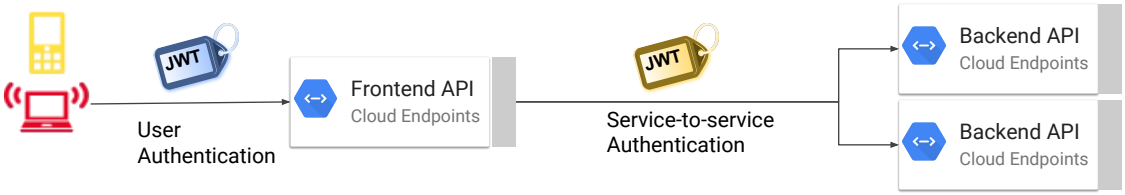
At runtime, Cloud Endpoints can receive calls from any source, such as mobile applications, web applications, and other services. The calls are load balanced and routed to the Extensible Service Proxy. The Extensible Service Proxy works with the Service Control API to check the request against the API configuration and verify that the request can be passed through to the backend. If the request authenticates successfully, the Extensible Service Proxy passes it on to the API backend. The

Service Control API logs information about incoming requests. These log messages and metrics can be viewed by using the Cloud Console.

For more information, see:

- Cloud Endpoints Architectural Overview: https://cloud.google.com/endpoints/docs/openapi/architecture-overview
- Naming Your APi: https://cloud.google.com/endpoints/docs/openapi/naming-your-api-service

Enable user authentication and service-to-service authentication




```
securityDefinitions:
  firebase:
    authorizationUrl: ""
    flow: "implicit"
    type: "oauth2"
    # Replace YOUR-PROJECT-ID with your project ID
    x-google-issuer:
"https://securetoken.google.com/YOUR-PROJECT-ID"
    x-google-jwks_uri:
"https://www.googleapis.com/service_accounts/v1/metadata
/x509/securetoken@system.gserviceaccount.com"
    x-google-audiences: "YOUR-CLIENT-ID"
security:
  - firebase: []
```

- Create a service account for service-to-service authentication.
- Use a Google ID Token to sign the token.

Authentication and Authorization

Google Cloud

With Cloud Endpoints API, you can authenticate users who are attempting to invoke your frontend APIs. Cloud Endpoints supports user authentication with Firebase, Auth0, Google authentication, and other custom authentication methods. After the user signs in, the authentication providers send a signed JSON Web Token (JWT pronounced jot) to Cloud Endpoints. Cloud Endpoints checks that the JWT is signed by the provider that you specified in your API configuration.

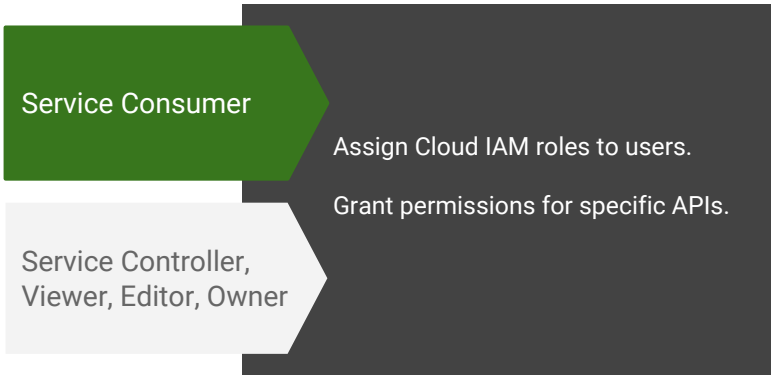
For service-to-service authentication, create a service account. Use a Google ID Token to sign the token.

For more information, see:

- Authenticating service-to-service calls with Google Cloud Endpoints (Google Cloud Next '17): https://www.youtube.com/watch?v=4PgX3yBJEyw
- Service-to-service authentication: https://cloud.google.com/endpoints/docs/openapi/service-to-service-auth

Images:
https://pixabay.com/en/key-ring-key-tag-label-plain-157133/

You can restrict API access



Authentication and Authorization

Google Cloud

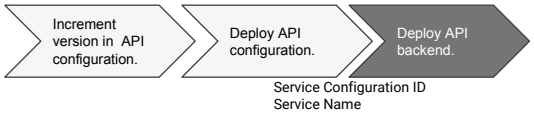
To restrict access to your Cloud Endpoints API, you can assign Cloud Identity and Access Management (Cloud IAM) roles to specific users. You can grant access to a specific API or to the entire Cloud project.

To give users the ability to enable your service in their own Cloud project and invoke its APIs, assign the Service Consumer role to them. This is the most common use case. You can assign the Service Controller, Viewer, Editor, or Owner roles to give users greater permissions to view and manage service configurations and projects.

For more information about granting API access, see https://cloud.google.com/endpoints/docs/openapi/api-access-overview.

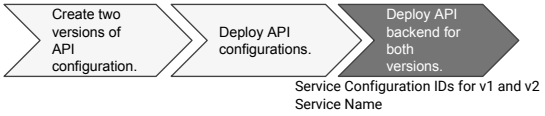
You can deploy multiple versions of your API in production

Backwards-Compatible Changes




```
apiconfig.yaml
info:
  description: "Cloud Endpoints API example."
  title: "Endpoints Example"
  version: "1.2"
host: "quiz-api.endpoints.example-project.cloud.goog"
```

Backwards-Incompatible Changes




```
apiconfig_v1.yaml
info:
  description: "Cloud Endpoints API example."
  title: "Endpoints Example"
  version: "1.1"
host: "quiz-api.endpoints.example-project.cloud.goog"
basePath: "/v1"

apiconfig_v2.yaml
info:
  description: "Cloud Endpoints API example."
  title: "Endpoints Example"
  version: "2"
host: "quiz-api.endpoints.example-project.cloud.goog"
basePath: "/v2"
```

Management and Scalability

Google Cloud

You might make small changes, bug fixes, and performance enhancements to your API backend. These changes are usually backward compatible. In such cases, it is good practice to increment the version attribute in the Cloud Endpoints API configuration and then redeploy the API configuration and the API backend.

If you make changes that are not backwards compatible and will break consumers of your API, deploy two versions of your Cloud Endpoints API by creating a separate API configuration for each version. The `gcloud service-management deploy` command will return a different service configuration ID for each version. Update the API backend configuration of each version with the corresponding service configuration id.

You can also delete versions of your API when they are no longer in use. Make sure to announce your deprecation and phase-out plans to the consumers of the API well in advance. This will give them time to evaluate and migrate to newer versions of your API.

For more information, see:

- Versioning an API: https://cloud.google.com/endpoints/docs/versioning-an-api
- Deleting an API and API instances: https://cloud.google.com/endpoints/docs/openapi/deleting-an-api-and-instances

You can deploy your API in multiple environments

Development	quiz-api.endpoints.**jane-dev-project**.cloud.goog/v1/extract
Staging	quiz-api.endpoints.**staging-project**.cloud.goog/v1/extract
Production Alpha	quiz-api.endpoints.**prod-alpha-project**.cloud.goog/v1/extract
Production	quiz-api.endpoints.**prod-project**.cloud.goog/v1/extract

Management and Scalability

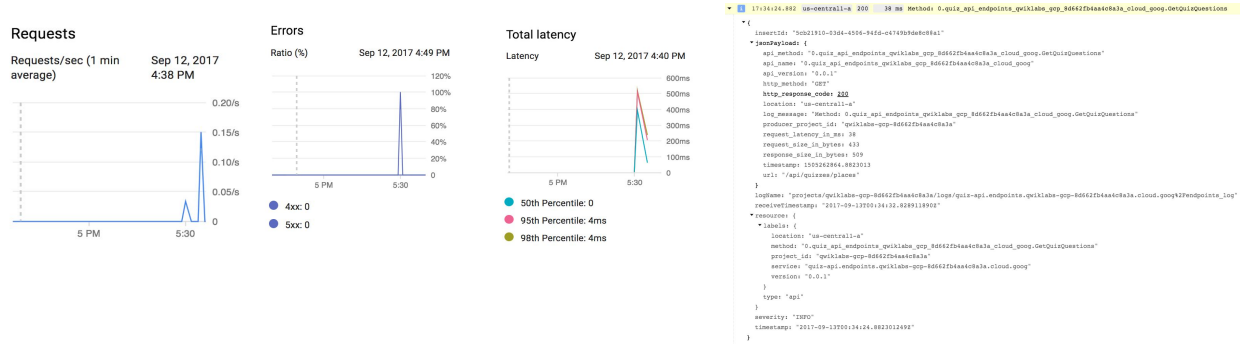
Google Cloud

When developing and deploying APIs, you will have separate environments for developing, staging, limited releases such as a private alpha, and production.

You can create separate projects for each environment and deploy your Cloud Endpoints API and backend with separate endpoints for your consumers. For example, developers can run their unit tests against the development environment. You can run integration tests against the staging endpoint. A separate project for private alphas provides a high degree of isolation from other environments.

For more information about naming conventions for projects, see https://cloud.google.com/endpoints/docs/openapi/planning-cloud-projects.

You can view error logs and metrics from the Cloud Endpoints dashboard



Method	Requests	Avg req size	Avg resp size	4xx	5xx	Latency 98%	Latency 95%	Latency median
GET /api/quizzes/*	10	430 B	509 B	0	0	472 ms	393 ms	66 ms

Logging and Monitoring

Google Cloud

In the Cloud Endpoints dashboard, you can see metrics related to requests, 4xx and 5xx errors, and latency. You can also view logs in Stackdriver Logging to see detailed information about requests to each API.

More resources

API Design Guide

https://cloud.google.com/apis/design/

Authenticating service-to-service calls with Google Cloud Endpoints

https://youtu.be/4PgX3yBJEyw