Google Cloud

Partner Certification Academy


# Professional Cloud Developer

The information in this presentation is classified:

## Google confidential & proprietary

⚠ This presentation is shared with you under NDA.

- Do **not** record or take screenshots of this presentation.
- Do **not** share or otherwise distribute the information in this presentation with anyone **inside** or **outside** of your organization.

Thank you!

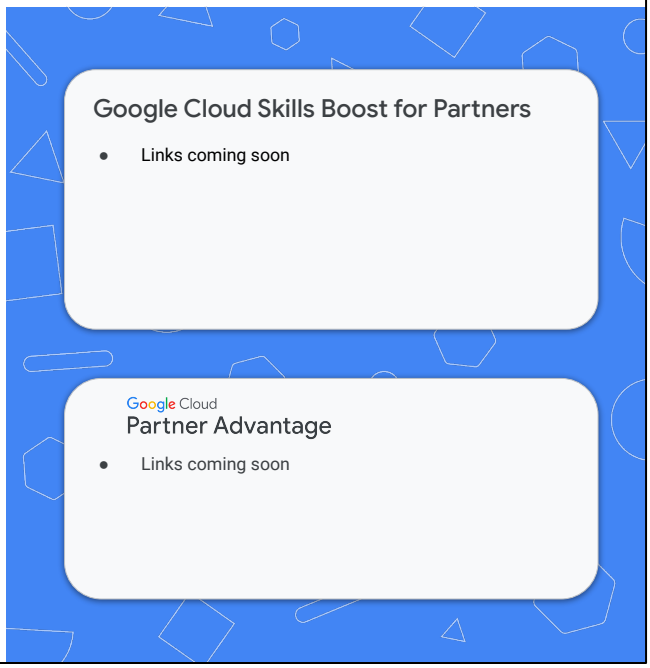
Google Cloud

# Source Materials

Some of this program's content has been sourced from the following resources:

- Google Cloud certification site
- Google Cloud documentation
- Google Cloud console
- Google Cloud courses and workshops
- Google Cloud white papers
- Google Cloud Blog
- Google Cloud YouTube channel
- Google Cloud samples
- Google codelabs
- Google Cloud partner-exclusive resources





# Session logistics

- When you have a question, please:
  - Click the Raise hand button in Google Meet.
  - Or add your question to the Q&A section of Google Meet.
  - Please note that answers may be deferred until the end of the session.
- These slides are available in the Student Lecture section of your Qwiklabs classroom.
- The session is **not recorded.**
- Google Meet does not have persistent chat.
  - If you get disconnected, you will lose the chat history.
  - Please copy any important URLs to a local text file as they appear in the chat.

# Module Agenda

01 Firebase authentication

02 OAuth 2.0 and OpenID Connect

03 Cloud IAP

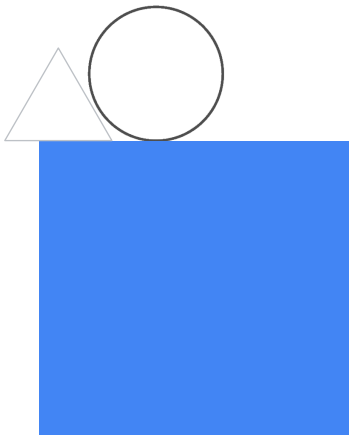
04 Identity Platform

05 Cloud Storage authentication

06 API development

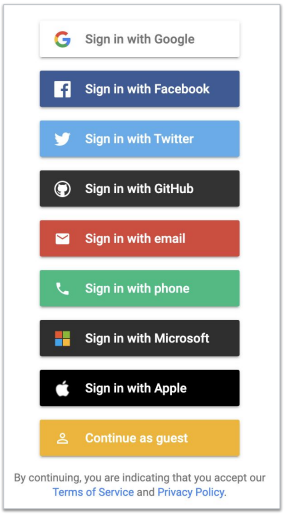
07 Next workshop's assigned content

# Firebase Authentication



# Firebase authentication

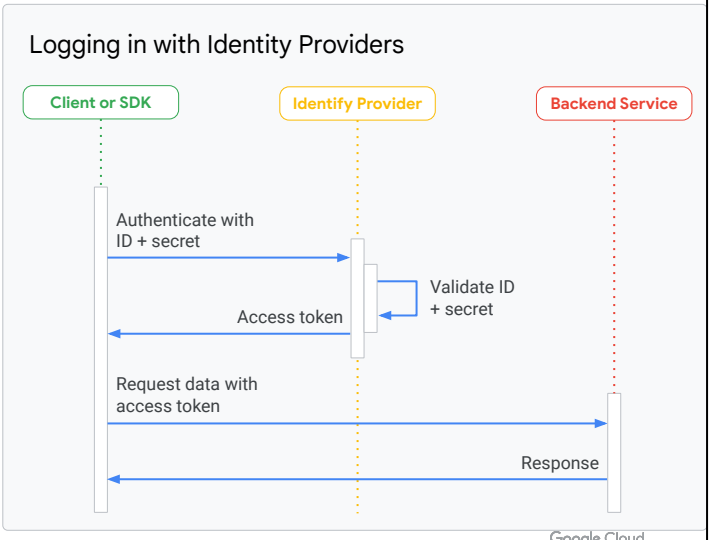
- Multiple options
  - Username/password
  - Sign-in links (No password needed)
    - User enters email address and a link to sign-in is sent to it
    - Email contains a one-time, short-lived authentication code
    - Clicking the link takes the user to a login page where the auth code is exchanged for a  normal, long-lived auth token
  - Phone number authentication
    - The authentication code is to the user's phone via SMS
  - “Social” logins with Federated Identity Providers
    - Sign in with Google, Microsoft, Facebook, etc.



# Social authentication uses OpenID Connect (OIDC)

- The OpenID Connect (OIDC) standard is based on OAuth 2.0, an industry standard for authentication
- User is sent to the login page of the identity provider
  - User logs in there
  - Then redirected back to the application, along with the provider token that says, “yes, this is a valid user, and they have successfully signed in.”

Image from: [What is Firebase Authentication?]



# Firebase authentication made easy

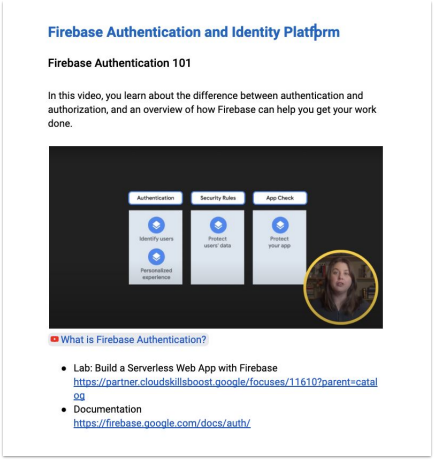
- FirebaseUI is a library that provides a drop-in auth solution
  - Handles the UI flows for signing in users
    - Build on top of Firebase Auth
- Available for iOS, Android and web applications
- Demo available on Github



For iOS: https://firebase.google.com/docs/auth/ios/firebaseui
For Android: https://firebase.google.com/docs/auth/android/firebaseui
For web: https://firebase.google.com/docs/auth/web/firebaseui

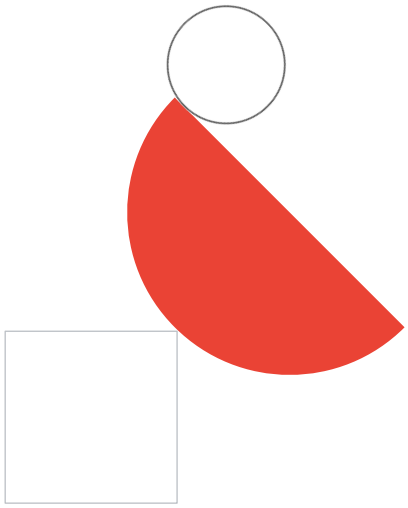
# Guide: Firebase authentication and Identity Platform

- Provides the social sign-in capabilities offered by Firebase
- Cloud Identity supports many more features including:
  - Multi-factor authentication
  - Enhanced logging of user activity
  - Blocking functions that run custom code as a result of a user registration/sign-in
  - Support for SAML and OpenID Connect providers not supported natively in Firebase
- To use these additional features in Firebase,
  - Upgrade to Firebase Authentication with Identity Platform




Firebase Authentication and Identity Platform

# OAuth 2.0 and OpenID Connect



# Overview - OAuth 2.0

**OAuth 2.0** is the industry-standard protocol, focuses on client developer simplicity while providing specific authorization flows for **web applications**, **desktop applications**, **mobile phones**, and **devices.**

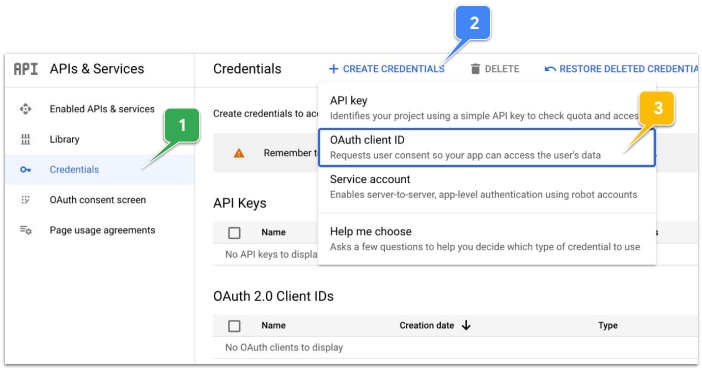
Google APIs use the OAuth 2.0 protocol for **authentication** and **authorization**. Google supports common **OAuth 2.0** scenarios.

- Client-side applications
- Web server applications
- Installed applications
- Applications on limited-input devices
- Service accounts

# Obtain OAuth 2.0 credentials from the Google API Console.

You must create an OAuth client appropriate for the platform on which your app will run, for example:

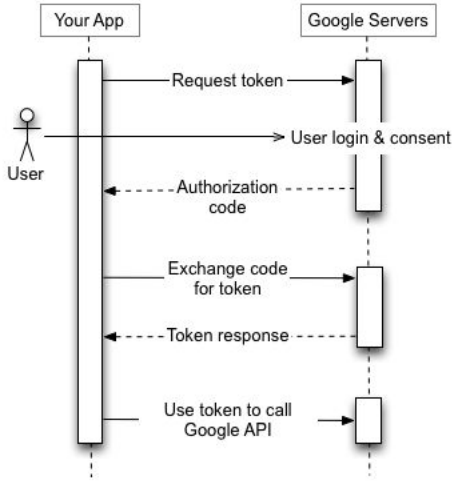
- For server-side or JavaScript web apps use the "web" client type. Do not use this client type for any other application, such as native or mobile apps.
- For Android apps, use the "Android" client type.
- For iOS and macOS apps, use the "iOS" client type.
- For Universal Windows Platform apps, use the "Universal Windows Platform" client type.
- For limited input devices, such as TV or embedded devices, use the "TVs and Limited Input devices" client type.
- For server-to-server interactions, use service accounts.




API's & Services

1. Credentials
2. Create Credentials
3. OAuth client ID

# Scenarios: Client-side (JavaScript) applications

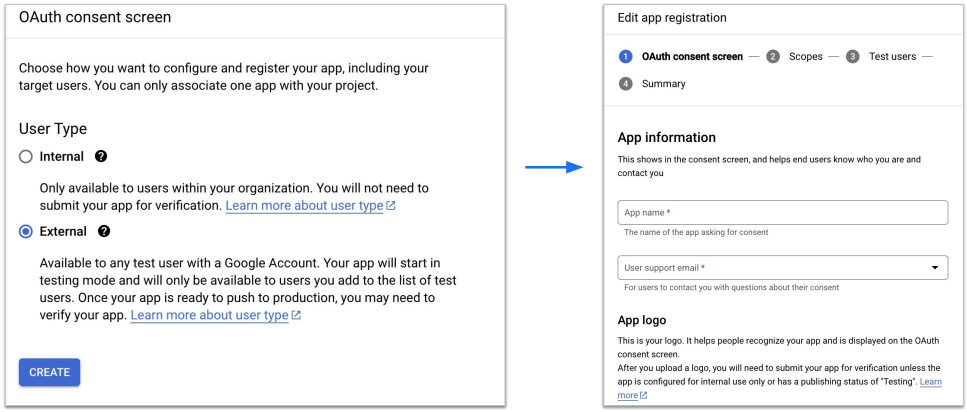



The Google OAuth 2.0 endpoint supports JavaScript applications that run in a browser.

https://developers.google.com/identity/protocols/oauth2/javascript-implicit-flow

# Consent Screen

Some requests require an authentication step where the user logs in with their Google account. After logging in, the user is asked whether they are willing to grant one or more permissions that your application is requesting. This process is called user consent.



# Obtain an access token from the Google Authorization Server.

Before your application can access private data using a **Google API**, it must obtain an access token that grants access to that API.

A single access token can grant varying degrees of access to multiple APIs. A variable parameter called **scope** controls the set of resources and operations that an access token permits.

```
function oauthSignIn() {
  var oauth2Endpoint = 'https://accounts.google.com/o/oauth2/v2/auth';
  var form = document.createElement('form');
  form.setAttribute('method', 'GET');
  form.setAttribute('action', oauth2Endpoint);
  var params = {'client_id': 'YOUR_CLIENT_ID',
                'redirect_uri': 'YOUR_REDIRECT_URI',
                'response_type': 'token',
                'scope': 'https://www.googleapis.com/auth/drive.metadata.readonly',
                'include_granted_scopes': 'true',
                'state': 'pass-through value'};
  for (var p in params) {
    var input = document.createElement('input');
    input.setAttribute('type', 'hidden');
    input.setAttribute('name', p);
    input.setAttribute('value', params[p]);
    form.appendChild(input);
  }
  document.body.appendChild(form);
  form.submit();
}
```

OAuth 2 Endpoint

Client ID

Redirect URI from API console

Scopes of access granted

# Get the token and call the API's

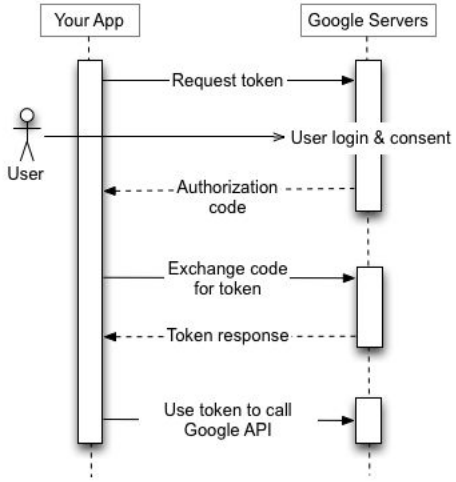
After users approve the consent screen, you can get the access token to call an API.

```
var xhr = new XMLHttpRequest();
xhr.open('GET',
    'https://www.googleapis.com/drive/v3/about?fields=user&' +
    'access_token=' + params['access_token']);
xhr.onreadystatechange = function (e) {
  console.log(xhr.response);
};
xhr.send(null);
```

API Endpoint

Access token from Google Authorization Server

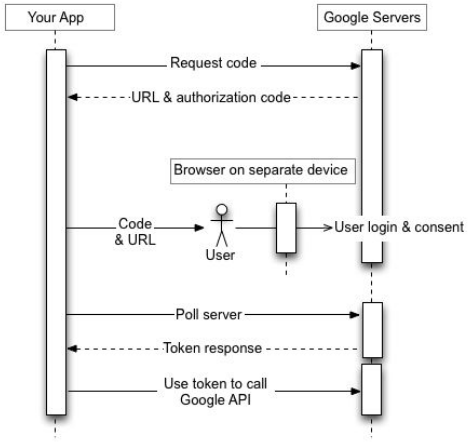
# Scenarios: Web server applications




The Google OAuth 2.0 endpoint supports web server applications that use languages and frameworks such as PHP, Java, Python, Ruby, and ASP.NET.

https://developers.google.com/identity/protocols/oauth2/web-server

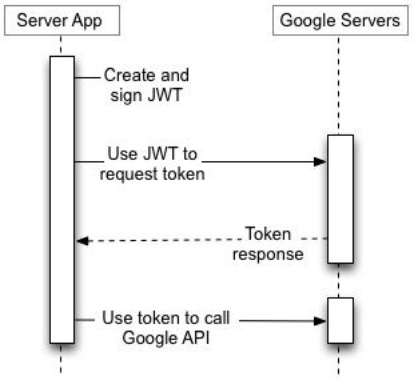
# Scenarios: Applications on limited-input devices




The Google OAuth 2.0 endpoint supports applications that run on limited-input devices such as game consoles, video cameras, and printers.

https://developers.google.com/identity/protocols/oauth2/limited-input-device

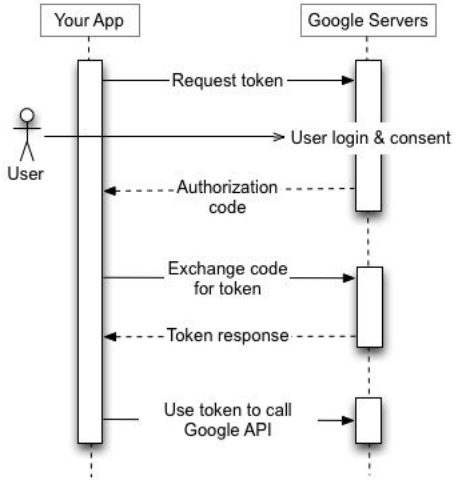
# Scenarios: Service accounts




Google APIs such as the Prediction API and Google Cloud Storage can act on behalf of your application without accessing user information.

https://developers.google.com/identity/protocols/oauth2/service-account

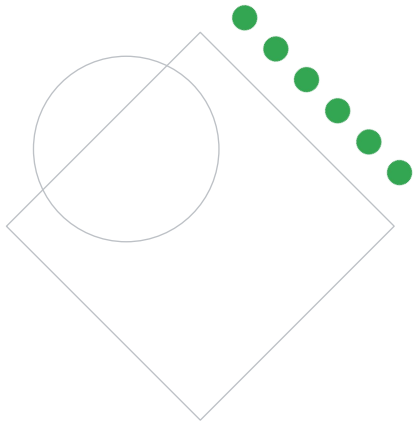
# Scenarios: Installed applications




The Google OAuth 2.0 endpoint supports applications that are installed on devices such as computers, mobile devices, and tablets.

https://developers.google.com/identity/protocols/oauth2/native-app

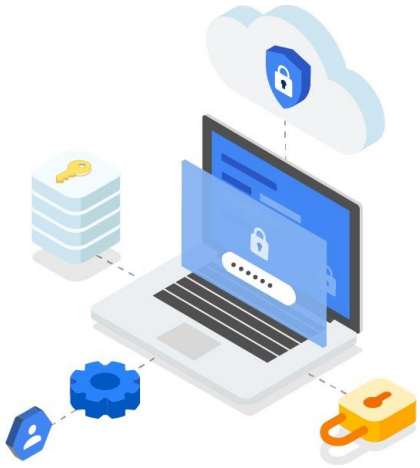
# Cloud IAP



# Move the access controls with IAM, BeyondCorp (ZeroTrust) and IAP

Security perimeter for resource isolation & adaptive access controls.

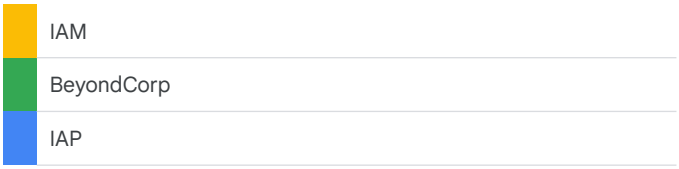
Shifting **access controls** from the **network perimeter** to **individual users**, **BeyondCorp** enables secure work from **virtually any location** without the need for a traditional **VPN**.





* Identity Platform may be used as well

# Cloud IAP



# Cloud IAP

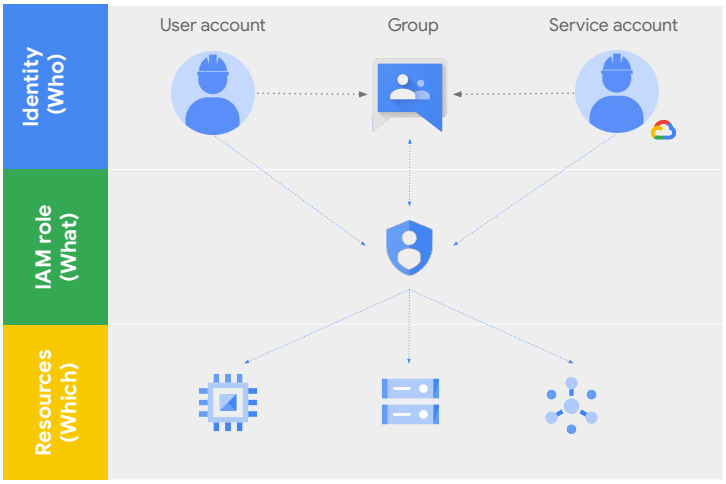
IAM

BeyondCorp

IAP

# Cloud IAM

IAM lets you adopt the security principle of least privilege, in order to **grant only the necessary access** to resources.

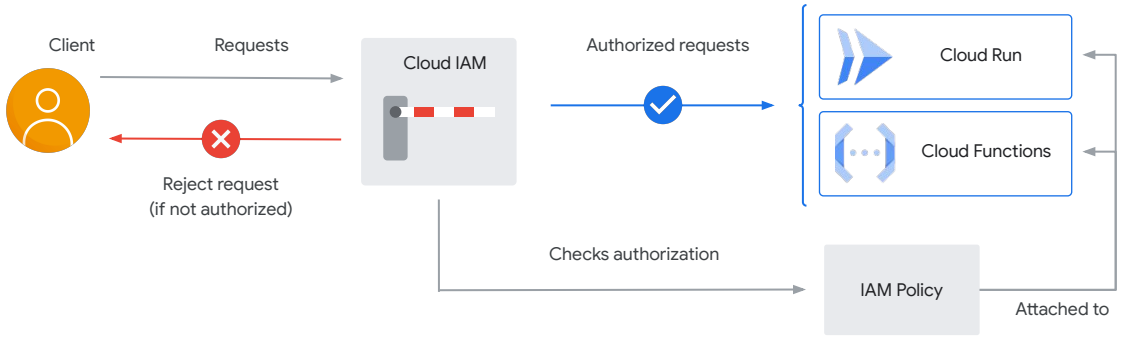



An **IAM role** is a collection of permissions, which determine what operations are allowed on a resource.

The binding of an identity to an IAM role is called an IAM policy.

IAM policies can be applied at the Organization, Folder, and Project levels. For certain Google Cloud solutions, they can be applied at the Resource level as well.

## Cloud IAM - Service invocation






**Cloud IAM** checks every incoming request to a Cloud Run service. All requests to a Cloud Run service pass IAM.

You configure IAM with an IAM policy that attaches to the Cloud Run service.

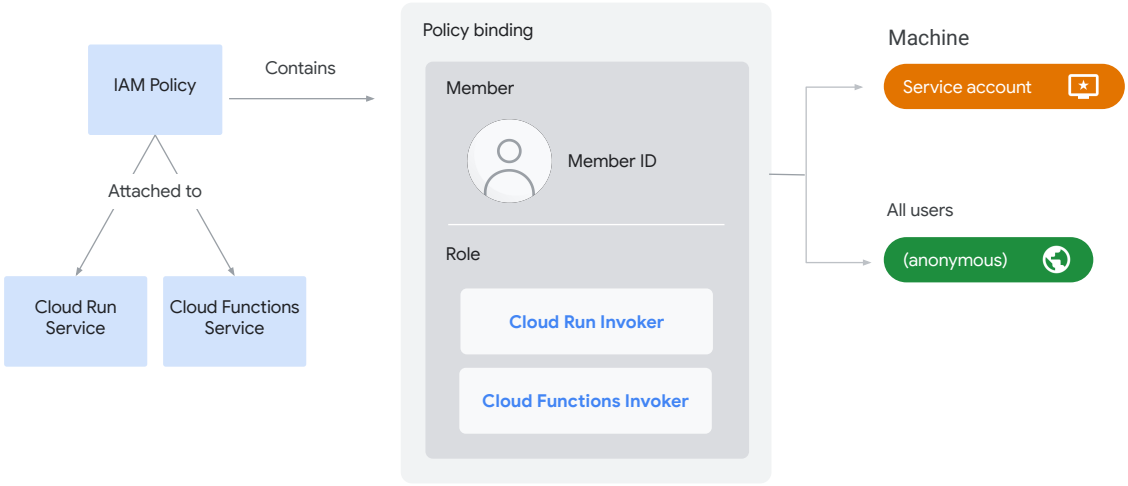
IAM protects every Cloud Run service, similar to how it protects every administrative API on Google Cloud.

For every incoming request, IAM:

- Identities the caller if there is an “Authorization” header part of the request
- Checks whether there is a corresponding entry in the IAM policy

Only authorized requests are forwarded.

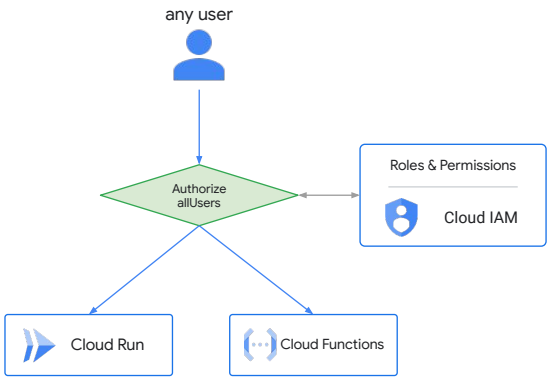
# Cloud IAM - Service invocation



# Publicly accessible services

If your service is a **public API or website**, anonymous/public access can be granted by assigning the Cloud Run / Cloud Functions Invoker role to the `allUsers` member type.

**Note:** Organization policy constraint `iam.allowedPolicyMemberDomains` (aka Domain restricted sharing) may limit the possibility to include `allUsers` member type in the IAM policy.



# Cloud IAP

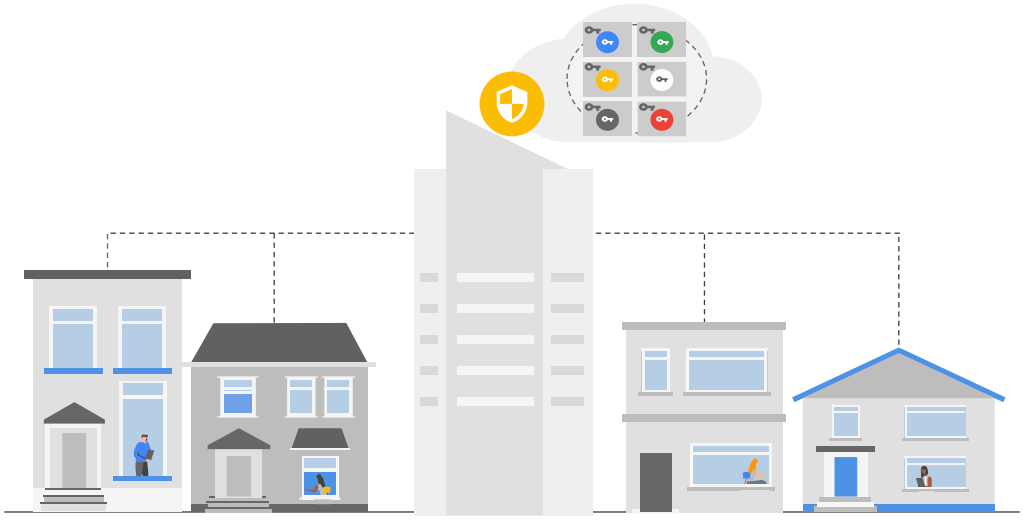
IAM

BeyondCorp

IAP

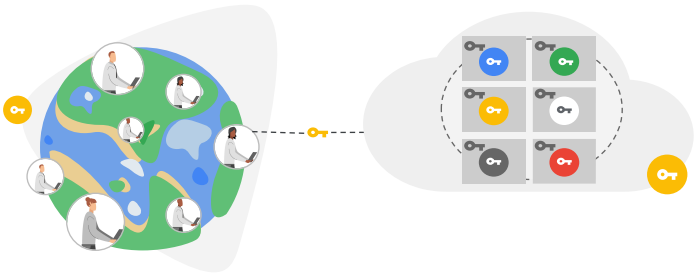
# Traditional enterprise security is focused on perimeter security

# Modern threat landscapes require a new way of thinking



# Zero trust solutions

Bringing together product, platform, and process



Hybrid workforce | Hybrid and multicloud | Service-based architecture | Kubernetes and containers

Accelerate zero trust with Google's prescriptive guidance:

01 Zero trust project discovery

02 Zero trust maturity workshop

03 Value-driven recommendations

# What is zero trust?

Trust must be validated before granting access to a resource.

### No implicit trust

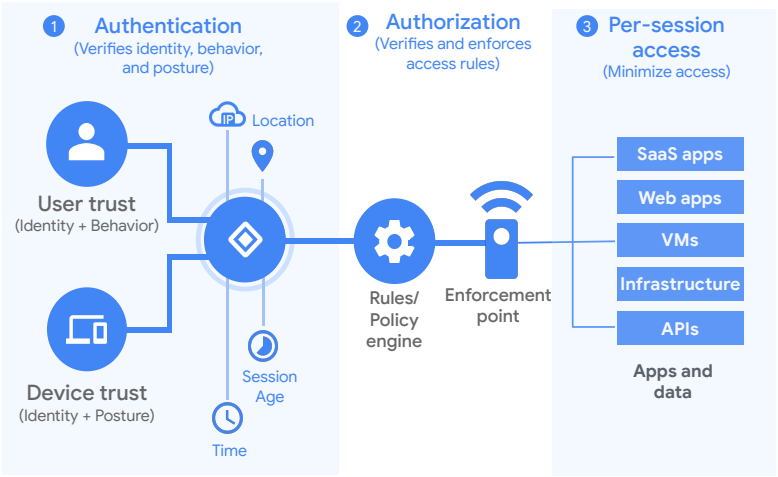
A “trust boundary” no longer applies to a modern infrastructure.

### Authentication and authorization

Authentication and authorization are discrete functions that must be performed on a session basis.

### Protect resources, not network segments

Network location is no longer a core component of enterprise security posture.



# BeyondCorp

**BeyondCorp** is **Google's** implementation of the zero trust model. It builds upon a decade of experience at **Google**, combined with ideas and best practices from the community.

By shifting **access controls** from the **network perimeter** to **individual users**, **BeyondCorp** enables secure work from **virtually any location** without the need for a traditional **VPN**.

BeyondCorp allows for single sign-on, access control policies, access proxy, and user- and device-based authentication and authorization.

# BeyondCorp Enterprise features

**Context-Aware Access** for SAML applications and Admin console

**Data Loss Prevention** to protect sensitive content from being **printed**

**URL categorization** and reporting

**Chrome Enterprise reporting connector** to report Chrome events (including **extension install**) to third-party providers

**Cloud Console** Context-Aware Access **dry run** prior to policy enforcement

**Context-Aware** Access support for **Microsoft Intune** events

**Identity-aware proxy** supports TCP forwarding for non-Google Cloud environments

**Identity-aware proxy re-authentication** via login, secure key, or enrolled second factors

**Identity-aware proxy** optimized authentication for **allowed-domain** users

**SAML Group membership mapping** included in the SAML response

**Certificate-based access** for VPC Service Controls and for apps hosted on Google Cloud

**Troubleshooting** with the Policy Troubleshooter for BeyondCorp Enterprise IAP and Remediator for Google Workspace

# Cloud IAP

IAM

BeyondCorp

IAP

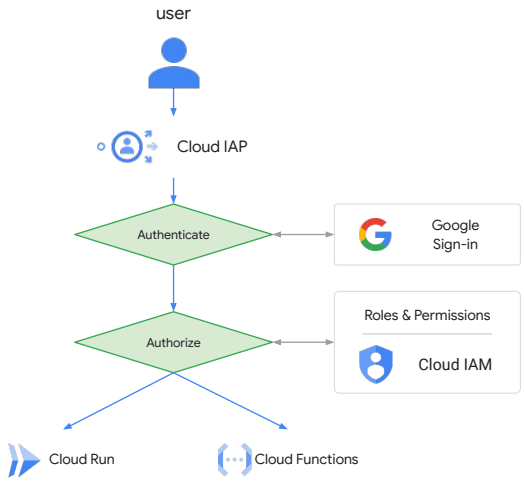
# Identity-Aware Proxy (IAP)

Identity-Aware Proxy (IAP) can be used to secure access to a **Cloud Run** service.

When enabled, a user accessing the service is redirected to OAuth 2.0 **Google Account sign-in flow for authentication**.*

**Authorization** is managed with **IAP** and **Cloud Run /Cloud Functions** IAM policy checks. If the user has the **IAP-secured Web App User** role (project or resource level), they are authorized to access the service. Cloud Run and Cloud Functions will perform their Invoker authorization normally.

**Note:** IAP is attached to the External HTTP(S) Load Balancer backend service resource and thus the Cloud Run service has to be exposed through the External HTTP(S) Load Balancer.




*Serveless solution*

# Connect to Linux VMs using Identity-Aware Proxy

You can connect to a **virtual machine (VM) instance** through its internal IP address, using **Identity-Aware Proxy (IAP) TCP forwarding**, using the `gcloud` SDK.

**IAP TCP forwarding** enables you to establish an encrypted tunnel over which you can forward **SSH** connections to **VMs**. When you connect to a **VM** that uses **IAP**, **IAP** wraps the **SSH** connection inside **HTTPS** before forwarding the connection to the **VM**. Then, IAP checks if the you have the required **IAM permissions** and if you do, grants access to the **VM**.

flag to use IAP

```
$ gcloud compute ssh <VM-NAME> --tunnel-through-iap
```

Your VM Instance name

# IAP Headers - JSON Web Tokens (JWT)

**Identity-Aware Proxy (IAP)** uses **JSON Web Tokens (JWT)** to make sure that a request to your app is authorized. This protects your app from the following kind of risks:

- IAP is accidentally disabled;
- Misconfigured firewalls;
- Access from within the project.

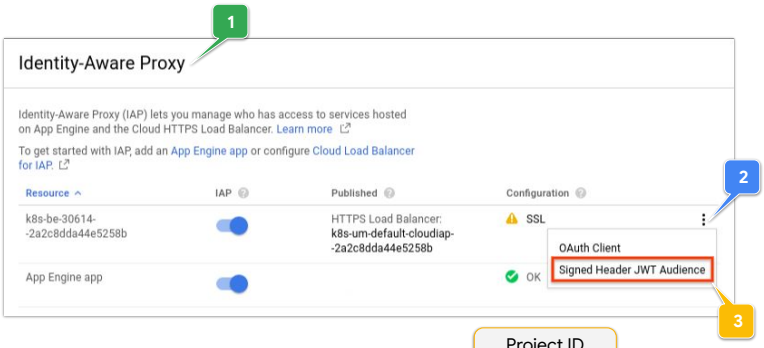
To secure your app with the **IAP JWT**, verify the header, payload, and signature of the **JWT**. The **JWT** is in the **HTTP** request header `x-goog-iap-jwt-assertion`. If an attacker bypasses **IAP**, they can forge the **IAP** unsigned identity headers, `x-goog-authenticated-user-{email,id}`.

The **IAP JWT** provides a more secure alternative.

# IAP Headers - JSON Web Tokens (JWT)

**Verifying the JWT header/payload**

1. Go to **IAP settings** for your project
2. **Click next to the Load Balancer**
3. **Signed Header JWT Audience.**




To get your **service ID** using the gcloud command-line tool, run the following command:

Project ID number

```
$ gcloud compute backend-services describe <SERVICE_NAME> --project=<PROJECT_ID> --global
```

Your Service Name

# IAP Headers - JSON Web Tokens (JWT)

**Retrieving the user identity**

The ID token's payload contains the `sub` and `email` user information.

Here is some sample code to secure an app with signed IAP headers.

```
let expectedAudience = null;
if (projectNumber && projectId) {
  expectedAudience = `/projects/${projectNumber}/apps/${projectId}`;
} else if (projectNumber && backendServiceId) {
  expectedAudience = `/projects/${projectNumber}/global/backendServices/${backendServiceId}`;
}

const oAuth2Client = new OAuth2Client();

async function verify() {
  const response = await oAuth2Client.getIapPublicKeys();
  const ticket = await oAuth2Client.verifySignedJwtWithCertsAsync(
    iapJwt,
    response.pubkeys,
    expectedAudience,
    ['https://cloud.google.com/iap']
  );
  console.log(ticket);
}
verify().catch(console.error);
```

Verify the id_token, and access the claims.

Return IAP ID token

# IAP Headers - JSON Web Tokens (JWT)

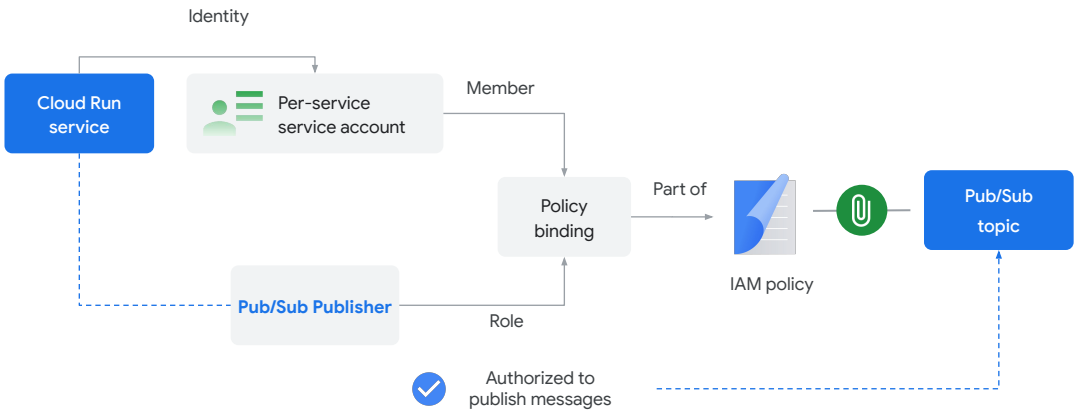
**Testing your validation code**

If you visit your app using the `secure_token_test` query parameters, **IAP** will include an **invalid JWT**. Use this to make sure your **JWT-validation** logic is handling all of the various failure cases, and to see how your app behaves when it receives an invalid **JWT**.

**Creating a health check exception**

Compute Engine and GKE health checks don't use **JWT headers** and **IAP** doesn't handle **health checks**. You'll need to configure your **health check** and app to allow the health check access.

## Service accounts - Calling Google API






Now in order to give the service account permissions on the Pub/Sub topic (our running example), this is what you need to do.

Add a policy binding with the role Pub/Sub publisher to the IAm policy that's attached to the topic.

# Identity token anatomy

**aud:** Audience "what service is this token for?"

**email/sub:** "Who owns this token?"

**iss: issuer** "Who signed this token"?

```
{
  "iss": "https://accounts.google.com",
  "aud": "https://hello-6w42z6vi3q-uc.a.run.app",
  "azp": "svc-2@my-sandbox-b377.iam.gserviceaccount.com",
  "sub": "100147106996764479085",
  "email": "svc-2@my-sandbox-b377.iam.gserviceaccount.com",
  "email_verified": true,
  "iat": 1574549410,
  "exp": 1574553010
}
```

Audience identifies the Cloud Run or Cloud Functions service that is called (for example, https://hello-6w42z6vi3q-uc.a.run.app). Please note that custom domains are **not** currently supported.

# Identity token – How to get?

**Compute Engine, Google Kubernetes Engine, Cloud Run**

- "Just ask the metadata server"

```
$ curl -H "Metadata-Flavor: Google" \
  'http://metadata/computeMetadata/v1/instance/service-
accounts/default/identity?audience=
```

**gcloud**:

```
$ gcloud auth print-identity-token --audiences=
```

**ServiceAccount JSON**:

- Create a JWT with the audience
- Use the Service Account to sign JWT
- Exchange JWT with Google's /token endpoint
- Google returns OIDC token

**IAM API**

- Invoke IAMCredentials generateIdToken() API

  https://cloud.google.com/iam/docs/reference/credentials/rest/v1/projects.serviceAccounts/generateIdToken

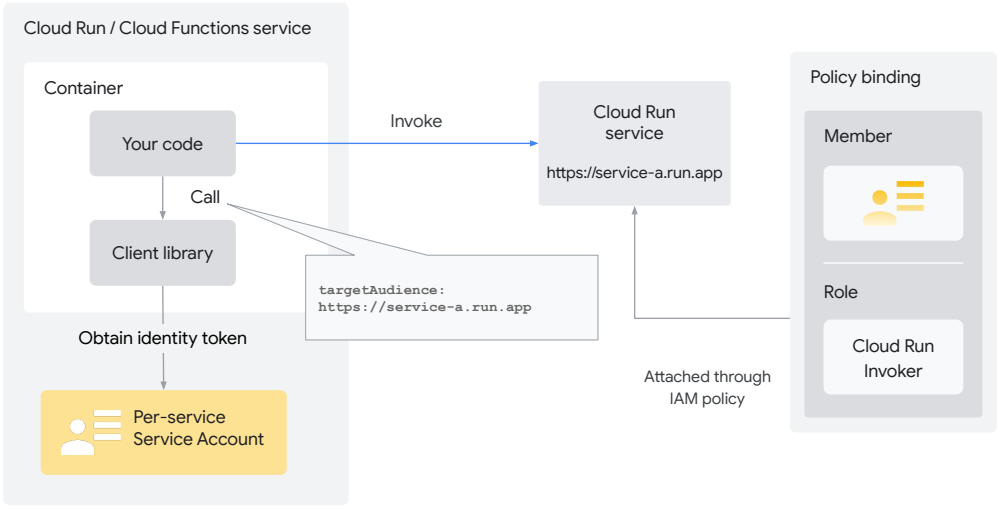
# Identity token – Or use Google client libraries

- The **easiest** and **most reliable** way to manage to generate and use this token.
- Works in any environment—even outside of Google Cloud—where the libraries can obtain authentication credentials by Application Default Credentials (ADC)
- Node.js example:

```
const {GoogleAuth} = require('google-auth-library');
const auth = new GoogleAuth();

async function request() {
  console.info(`request ${url} with target audience ${targetAudience}`);
  const client = await auth.getIdTokenClient(targetAudience);
  const res = await client.request({url});
  console.info(res.data);
}
```

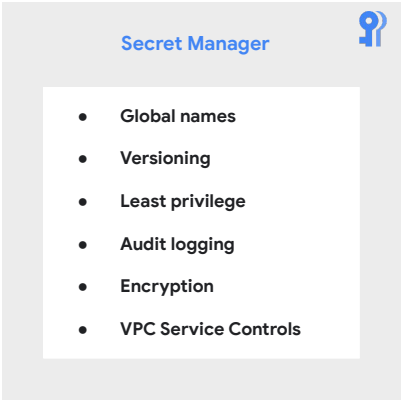
# Service-to-service communication



# Managing secrets

How to manage dependencies like API keys, passwords, and other sensitive information?

- Cloud Run and Cloud Functions are integrated with Secret Manager
- Two ways to inject into application:
  - Mount secret as a **volume**. Each read fetches the *latest* version of the secret.
  - Pass secret as an **environment variable**. Secret is resolved during at instance startup time.
- Secret access controlled using **Cloud IAM**
  - Grant role *Secret Manager Secret Accessor*



# Cloud IAP and App Engine Demo

This demo show how to host an application on **App Engine** using **IAP** to authorize and get users information of Google's authenticated users that access it.

Google Cloud
Partner Certification Academy


Professional Cloud Developer

v2309p

**Demo:** Cloud Identity Aware Proxy (IAP)

# Recommended demo

- **Code Labs: Secure Serverless Application with Identity Aware Proxy (IAP)**
  https://codelabs.developers.google.com/secure-serverless-application-with-identity-aware-proxy#0


This demo show how to host an application on **Cloud Run** using **IAP** to authorize access only to a list of approved users can access it.

https://www.youtube.com/watch?v=ayTGOuCaxuc



You can create your own http Load Balancer as a frontend of Cloud Run test deployment. After you can create an IAP to this frontend to all authenticated users as an example.

# Guide: Cloud IAP

Topics covered

- Identity-Aware Proxy overview
- Control access to with Identity-Aware Proxy
- IAP code samples
- IAP Use Case
- Configuring IAP to Protect a Project
- User Authentication
- Securing Compute Engine Applications and Resources using BeyondCorp Enterprise (BCE)

For more information:
https://cloud.google.com/iap/docs/signed-headers-howto

**Authenticating your users**

In this video, you learn how to configure Identity Aware Proxy (IAP) in App Engine, allowing you to easily and securely grant access to internal and external websites.


Centralize access to your organization's websites with Identity Aware Proxy (IAP)

**Recommended Course**

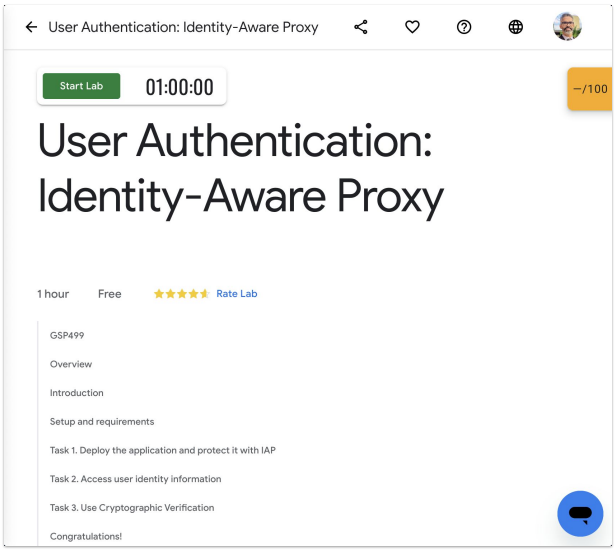
Complete the "Application Security: Techniques and Best Practices" section found in this course - watch the videos and do the hands-on labs.

Cloud IAP

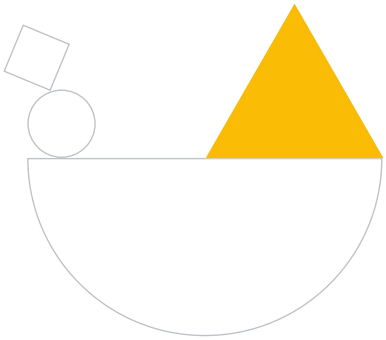
# Lab - User Authentication: Identity-Aware Proxy

In this lab, you build a minimal web application with Google App Engine, then explore various ways to use Identity-Aware Proxy (IAP) to restrict access to the application and provide user identity information to it.

https://partner.cloudskillsboost.google/focuses/11642?parent=catalog

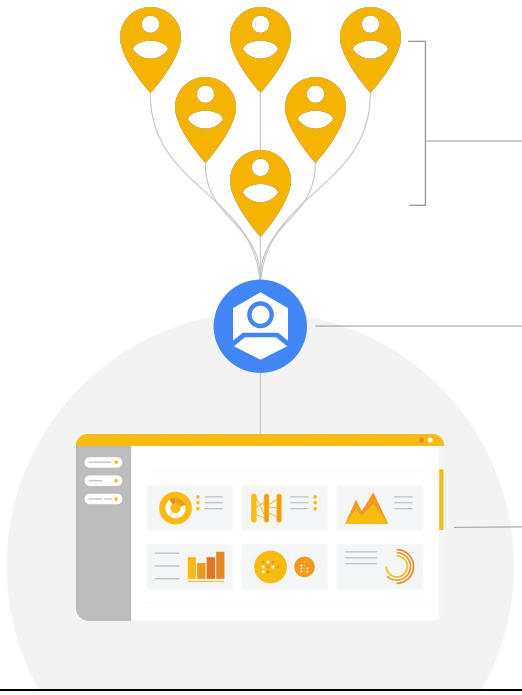


# Identity Platform



# Identity Platform

A customer identity and access management (CIAM) service that helps organizations add identity management functionality to their apps and services, protect user accounts, and scale with confidence.



# Identity Platform provides authentication as a service

- Provides federated login that integrates with many common providers.
- Used to provide sign-up and sign-in for your end users' applications.

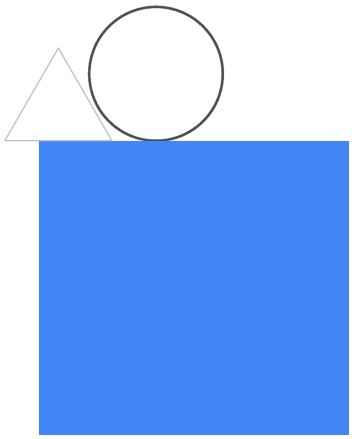
# Identity Platform features

- Provides **OpenID** Connect on top of **OAuth 2.0**
- Social sign-in capabilities offered by Firebase
- Cloud Identity supports many more features including:
  - Multi-factor authentication
  - Enhanced logging of user activity
  - Blocking functions that run custom code as a result of a user registration/sign-in
  - Support for SAML and OpenID Connect

# Compare Google CIAM products

| | Identity Platform | Firebase Authentication |
|---|---|---|
| **Authentication** | • Sign in with email, OAuth, phone, custom authentication<br>• Additionally, with OIDC, SAML<br>• Support multi-tenancy<br>• Support IAP integration | Sign in with email, OAuth, phone, custom authentication |
| **SDKs** | Web, iOS, Android, Admin SDKs | Web, iOS, Android, Admin SDKs |
| **Compliance** | • ISO 27001<br>• SSAE 18 SOC1, SSAE 18 SOC2, SSAE 18 SOC3<br>• TISAX, BAA coverage, PCI-DSS in-scope | • ISO 27001<br>• SSAE 18 SOC1, SSAE 18 SOC2, SSAE 18 SOC3 |
| **Enterprise SLA** | 99.95% uptime | N/A |

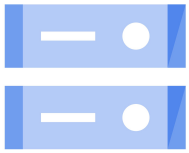
# Cloud Storage Authentication



# Cloud Storage

Most of the operations you perform in Cloud Storage must be authenticated. The only exceptions are operations on objects that allow **anonymous** access.

Objects are **anonymously** accessible if the `allUsers` group has **READ** permission. The `allUsers` group includes anyone on the Internet.



# Cloud Storage - OAuth 2.0

**Cloud Storage** uses **OAuth 2.0** for API authentication and authorization.

- A **server-centric** flow allows an application to directly hold the credentials of a service account to complete authentication.
- A **user-centric** flow allows an application to obtain credentials from an end user. The user signs in to complete authentication.

# Cloud Storage - Scopes

Authorization is the process of determining what permissions an authenticated identity has on a set of specified resources. OAuth 2.0 uses scopes to determine if an authenticated identity is authorized.

| Type | Description | Scope URL |
|---|---|---|
| `read-only` | Only allows access to read data, including listing buckets. | `https://www.googleapis.com/auth/devstorage.read_only` |
| `read-write` | Allows access to read and change data, but not metadata like IAM policies. | `https://www.googleapis.com/auth/devstorage.read_write` |
| `full-control` | Allows full control over data, including the ability to modify IAM policies. | `https://www.googleapis.com/auth/devstorage.full_control` |
| `cloud-platform.read-only` | View your data across Google Cloud services. For Cloud Storage, this is the same as `devstorage.read-only`. | `https://www.googleapis.com/auth/cloud-platform.read-only` |
| `cloud-platform` | View and manage data across all Google Cloud services. For Cloud Storage, this is the same as `devstorage.full-control`. | `https://www.googleapis.com/auth/cloud-platform` |

# Cloud Storage - Authentication

Client libraries can use Application Default Credentials (ADC) to easily authenticate with Google APIs and send requests to those APIs. Using **service accounts**.

**Google Cloud**

Compute Engine

App Engine

Cloud Run

Cloud Functions

The environment already provides a **service account** authentication information

**Other environments**

```
$ export GOOGLE_APPLICATION_CREDENTIALS=<path_to_service_account_file>
```

# Use Application Default Credentials (ADC) to authenticate between applications

ADC checks for credentials in the following order:

- Checks for `GOOGLE_APPLICATION_CREDENTIALS` environment variable
- Checks for default service accounts
- If 1 and 2 aren't found, an error is thrown

Make an authenticated API request

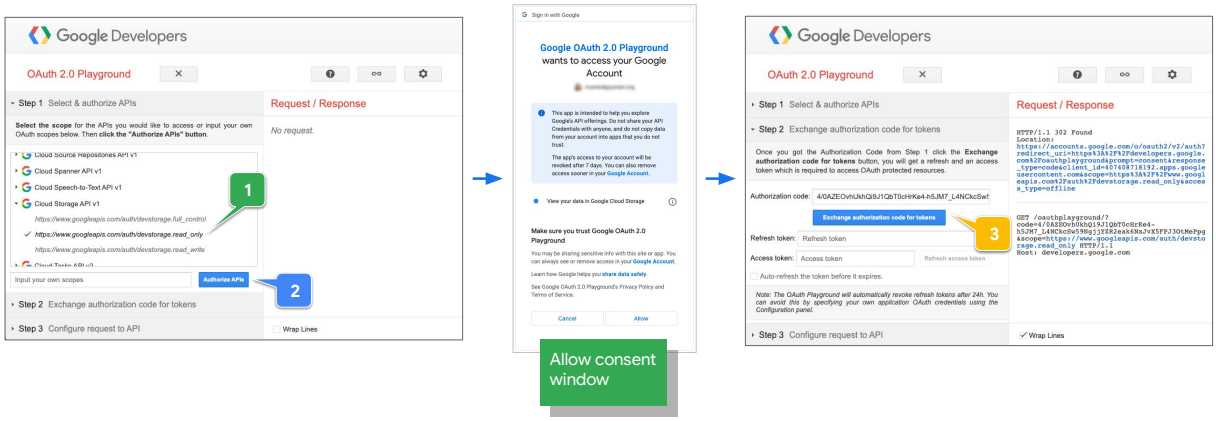
```
$ export GOOGLE_APPLICATION_CREDENTIALS=<path_to_service_account_file>
```

```
def implicit():
    from google.cloud import storage

    storage_client = storage.Client()
    buckets = list(storage_client.list_buckets())
    print(buckets)
```

If you don't specify credentials when constructing the client, the client library will look for credentials in the environment.

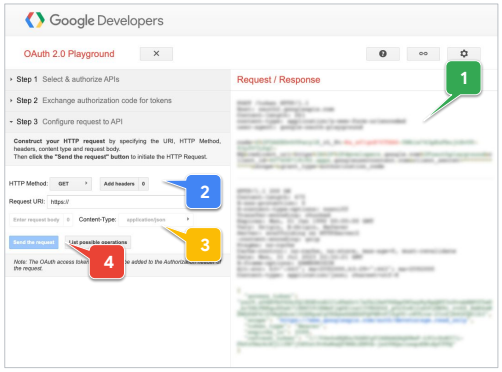
# Client Site API Authorization - Example 1/2




1. Select Cloud Storage API v1 access level
2. Authorize APIs
3. Exchange authorization code for tokens

https://cloud.google.com/storage/docs/authentication

# Client Site API Authorization - Example 2/2

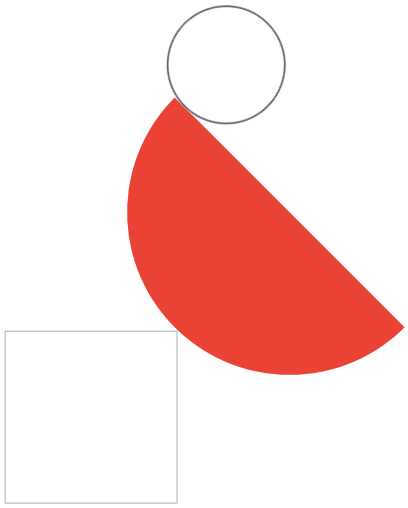



```
...
{
      ...
      "kind": "storage#object",
      "contentType": "text/plain",
      "name": "******/",
      "etag": "CLmU+P/F6P4CEAE=",
      ...
      "bucket": "bucket-name",
      "updated": "2023-05-09T15:26:10.410Z",
      "crc32c": "AAAAAA==",
      ...
      "mediaLink": "https://storage.googleapis.com/download/***...",
      ...
}
...
```

Response

1. Get the Token from the response
2. Set the HTTP Header
3. Set the request URI
4. Send the request

# API development



# API development

- REST / gRPC
- Cloud Endpoints
- Apigee

# API development

- REST / gRPC
- Cloud Endpoints
- Apigee

# Ways to use HTTP for APIs

**REST**

**Representational State Transfer**

Client requests a resource using a REST API, the server transfers back the current state of the resource in a standardized representation.

**gRPC**

**Google Remote Procedure Call**

gRPC supports bidirectional streaming, enabling continuous data exchange between client and server

- gRPC supports bidirectional streaming, enabling continuous data exchange between client and server; REST is limited to request-response communication patterns.
- Both protocols have their strengths and weaknesses, and which one to choose depends on the specific requirements of your use case

## RESTful APIs

- REST is an architectural style, not a standard.
- REST APIs typically adhere to common web HTTP concepts.
- Message payloads generally use JSON (Javascript Object Notation).
- REST is currently the most common style of web API.
- APIs that follow the REST architectural style are called RESTful.



Before we get into the basics of designing REST APIs, let's explore what we mean by the term "REST." REST was defined in Roy Fielding's 2000 doctoral dissertation at the University of California, Irvine.

At this time, SOAP, or Simple Object Access Protocol, was a popular way to implement APIs. SOAP, however, is not very simple. In order to make an API call using the SOAP protocol, a developer needs to craft a complex XML payload by referencing an even more complex definition document called a WSDL, which stands for Web Services Description Language.

Fielding and his colleagues designed the REST architectural style while version 1.1 of HTTP was being designed. They saw an opportunity to use HTTP concepts to create a simpler and more familiar pattern for APIs. REST APIs leverage common web HTTP concepts like URLs, verbs, and status codes.

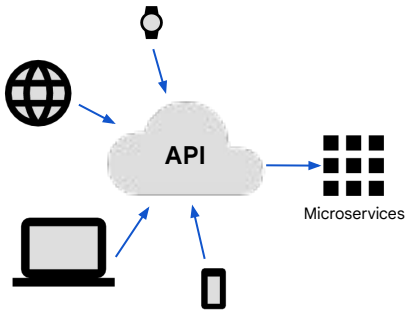
Message payloads for REST APIs are generally specified using simple JavaScript Object Notation, or JSON. The JSON payload of a well-written REST API is simpler and more human-readable than the complex XML used for SOAP services.

The use of common web patterns and simple payloads makes REST APIs easy for app developers to learn and use. Due to its simplicity and ease of use, REST has become the predominant style of web API today.

When an API is designed using the REST style, we call it a RESTful API.

# REST (Representational State Transfer) APIs

- Resources are identified by unique URIs (Uniform Resource Identifiers)
  - Customer record,  inventory item, etc.
- Clients perform operations on resources using standard HTTP methods GET, POST, PUT, and DELETE
- Key principle of REST API is statelessness
  - Each request from the client to the server should contain all the information necessary to complete the request
  - Each response from the server to the client should contain all information needed to satisfy the request
    - Makes the API more scalable
- Connections are not persisted between requests



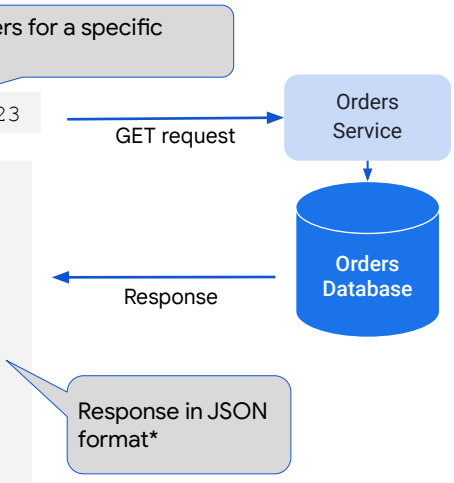
# Example REST API for the Orders service

```
GET https://api.example.com/orders?customer_id=123
```

```
200 OK
Content-Type: application/json
Cache-Control: max-age=3600

{
  "data": [
    {
      "id": "456",
      "order_date": "2023-03-20",
      "total_amount": 49.99,
      "customer_id": "123"
    }
  ]
}
```

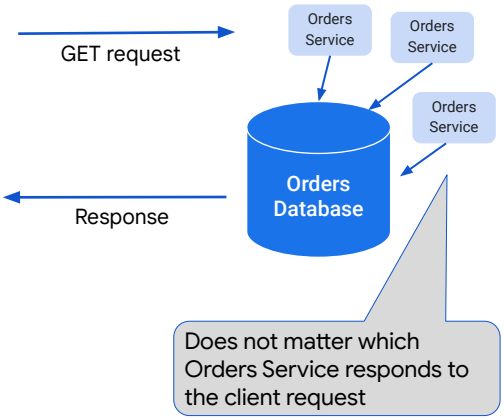
*Other response formats also supported in the standard, e.g., SOAP, XML



# Stateless microservices are easy to scale up/down as needed

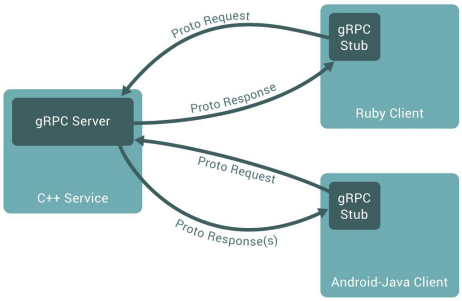
```
GET https://api.example.com/orders?customer_id=123
```

```
{
  "data": [
    {
      "id": "456",
      "order_date": "2023-03-20",
      "total_amount": 49.99,
      "customer_id": "123"
    }
  ]
}
```



# gRPC is a high-performance open-source remote procedure call (RPC) framework

- Originally developed by Google, now widely used by others
  - Netflix, Uber, and Square use gRPC as a foundation for their services and applications
- gRPC uses binary data serialization with Protocol Buffers* (protobuf)
  - More efficient than traditional text-based protocols like JSON or XML.
  - Results in faster and more efficient data transfer over the network, particularly in high-throughput scenarios
- gRPC libraries exist for many programming languages
  - Easy for developers to use gRPC in their preferred programming language and integrate it with their existing infrastructure




Introduction to gRPC

*Programming details is beyond the scope of this module



Introduction to gRPC
https://grpc.io/docs/what-is-grpc/introduction/

# Reasons to choose REST over gRPC

- Compatibility with existing systems
  - Based on the HTTP protocol, which is widely used and understood by developers
  - Makes it easier to integrate with existing systems, including web applications
- Simplicity and flexibility
  - REST is a simple and flexible protocol that can be used to build a wide range of applications
  - Supports a variety of data formats, including JSON, XML, and HTML.
- Caching
  - REST supports caching, which can improve performance by reducing the number of requests made to the server.
- Accessibility
  - REST can be accessed from any programming language, making it accessible to a wide range of developers



Introduction to gRPC
https://grpc.io/docs/what-is-grpc/introduction/

# Reasons to choose gRPC over REST

- Performance
  - gRPC is designed to be faster and more efficient than REST, especially for high-throughput and low-latency applications
- Strong typing and code generation
  - gRPC uses protocol buffers for message serialization and code generation
    - Easier to generate client and server code in multiple programming languages.
    - Protocol buffers are type-safe, i.e., data received is guaranteed to be of the correct data type, and the developer does not have to manually check or convert the data
- Streaming
  - gRPC supports bidirectional streaming, which allows for real-time communication between client and server.
- Authentication and encryption
  - gRPC supports authentication and encryption out of the box, making it easier to secure your application.



Introduction to gRPC
https://grpc.io/docs/what-is-grpc/introduction/

Type-safe remote server calls refer to a programming technique that allows developers to make remote calls to a server and receive data in a type-safe manner. In other words, the data received from the server is guaranteed to be of the correct data type, and the developer does not have to manually check or convert the data.

# Cloud Endpoints supports REST APIs and gRPC APIs

**Cloud Endpoints for REST APIs**

- JSON/REST API:
  - Popular
  - Easy to use
- API configuration: OpenAPI specification

**Cloud Endpoints for gRPC APIs**

- gRPC API:
  - Newer technology
  - Fast
  - Can generate client libraries for programming languages
  - Enables type safety
- API configuration:
  - Service definition: Protocol buffers
  - Service configuration: gRPC API specification



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 OpenAPI 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. Your clients can access your gRPC API while using plain old HTTP JSON calls. In the rest of this module, we'll focus on JSON-based REST APIs.

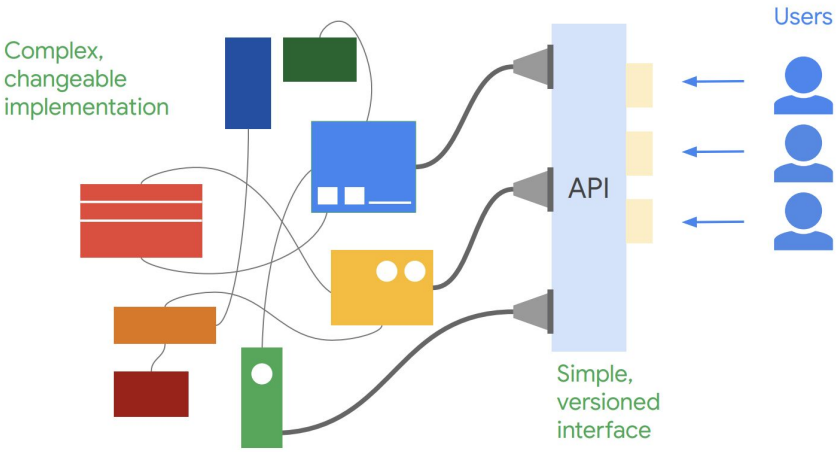
For more information about Cloud Endpoints for REST APIs, see https://cloud.google.com/endpoints/docs/openapi/open-api-spec.

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

# Application Programming Interfaces hide detail, enforce contracts



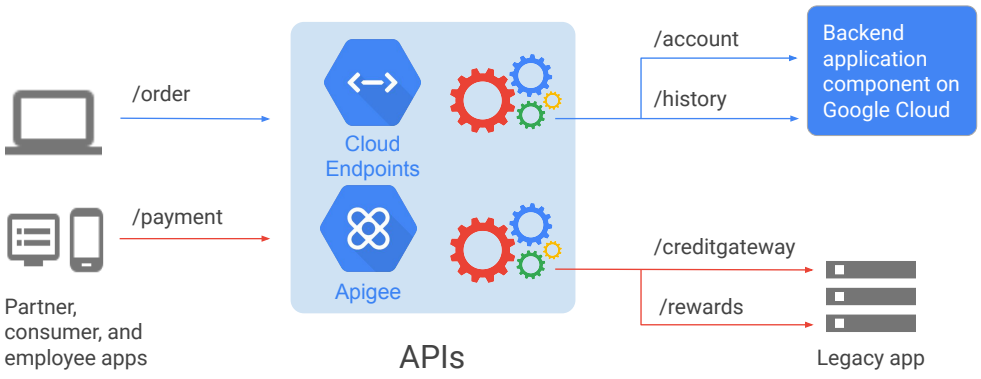
# API development

REST / gRPC

Cloud Endpoints

Apigee

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



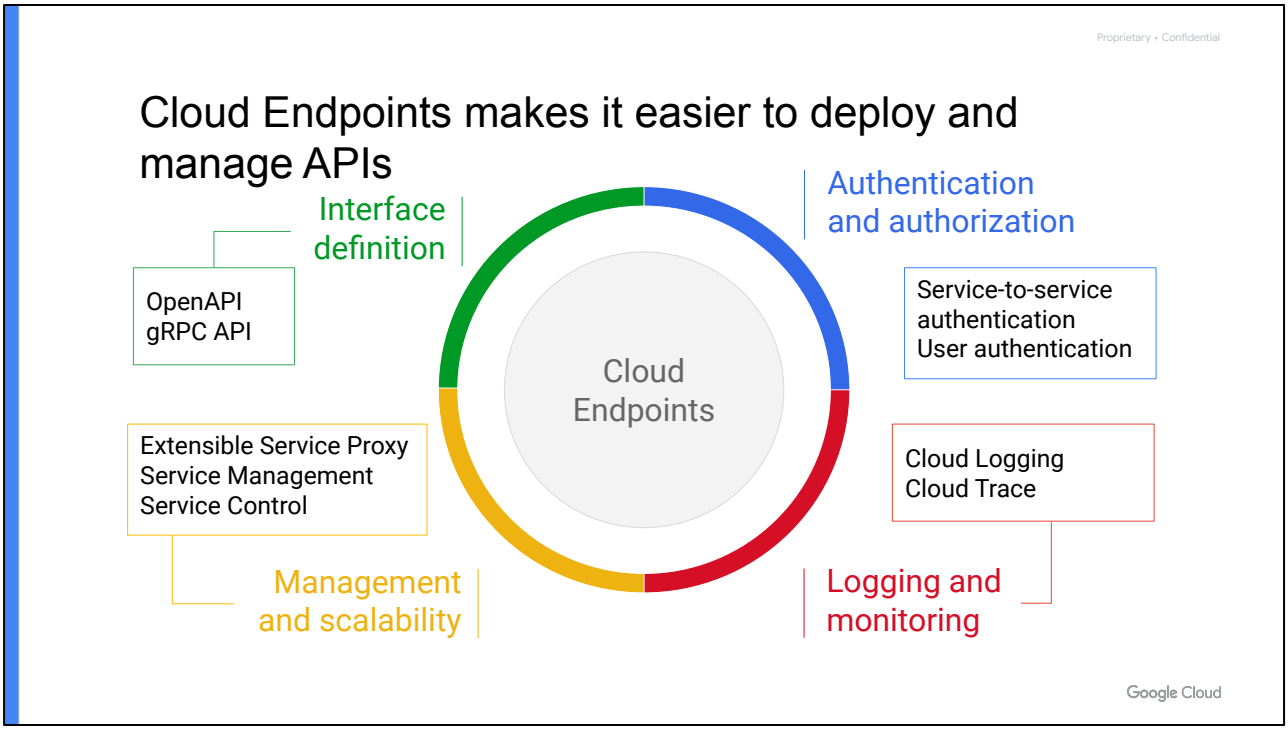



An API gateway enables clients to retrieve data from multiple services with a single request. An API gateway creates a layer of abstraction and insulates the clients from the partitioning of the application into microservices.

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

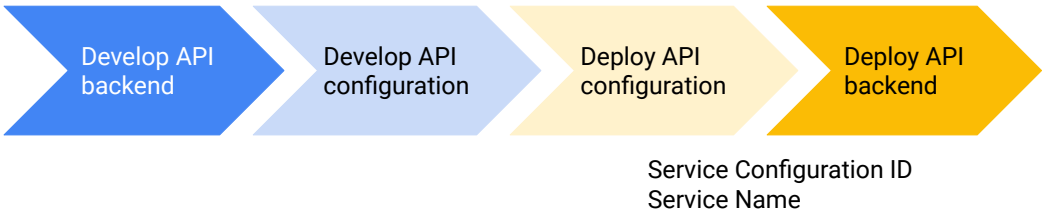
If you have legacy applications that cannot be re-factored and moved to the cloud, consider implementing APIs as an adaptor layer or façade. Each consumer can then invoke these modern APIs, instead of invoking outdated APIs using old protocols and disparate interfaces.

Using the Apigee API platform, you can design, secure, analyze, and scale your APIs for legacy backends.




Cloud Endpoints provides the infrastructure support that you need to deploy and manage robust, secure, and scalable APIs. Cloud Endpoints supports the OpenAPI specification and gRPC API specification. Cloud Endpoints supports service-to-service authentication and user authentication with Firebase, Auto zero, and Google authentication. The Extensible Service Proxy, Service Management, and Service Control APIs together validate requests, log data, and handle high volumes of traffic. Using Cloud Endpoints, Cloud Logging, and Cloud Trace, you can view detailed logs, trace lists, and metrics related to traffic volume, latency, size of requests and responses, and errors.

# Develop and deploy your API configuration and API backend






Let's dive into Cloud Endpoints for REST APIs.

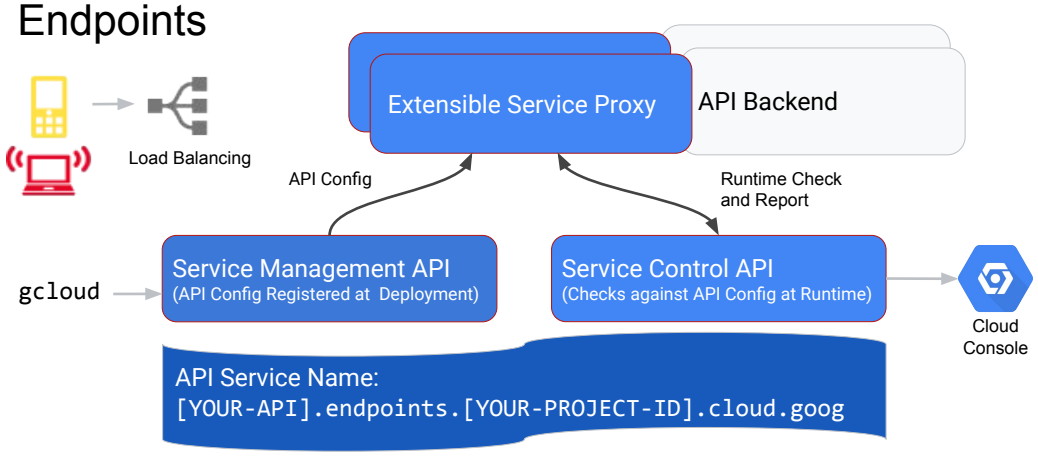
This diagram illustrates the high-level steps to make your API backend available as Cloud Endpoints API.

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’s backend configuration files, such as the “app.yaml” file for App Engine flexible environment deployments.

Finally, deploy the API backend.

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






When you deploy the API configuration, it is registered with the 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 a name in the format shown here, that is “your API name.endpoints.yourprojectid.cloud.goog”. This API service name is created, if it doesn't already exist, and then Cloud Endpoints configures the service according to your OpenAPI configuration file. Cloud Endpoints uses DNS-compatible names to uniquely identify services. Because projects in Google Cloud are guaranteed to have a globally unique name, you can use your project name to create a unique API service name, such as “`quizapi.endpoints.myprojectid.cloud.goog` ”. You can also map your own DNS name to your API.

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

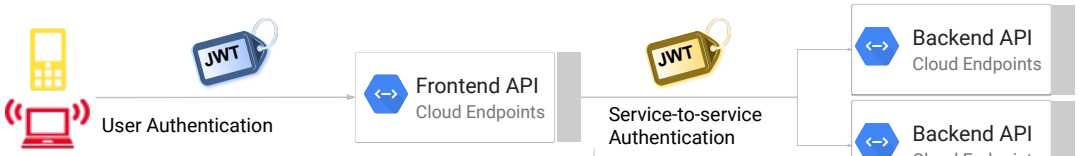
At runtime, Cloud Endpoints can receive calls from any source such as mobile applications, web applications and other services. The calls are load balanced and routed to the Extensible Service Proxy. The Extensible Service Proxy works with the Service Control API to check the request against the API 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/metad
ata/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.



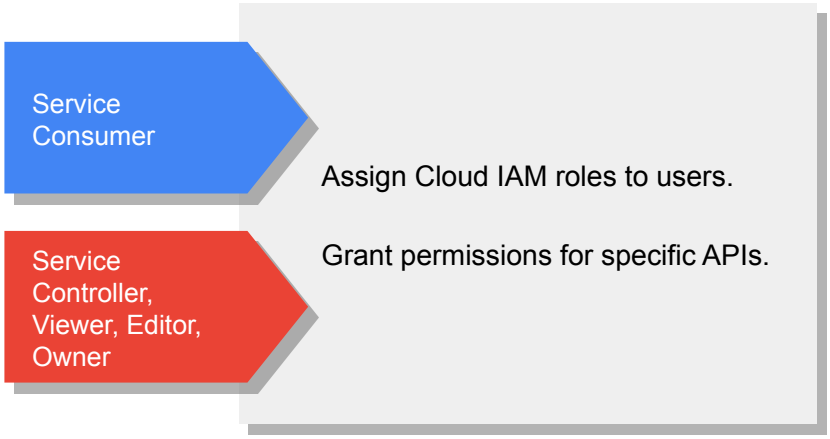
With Cloud Endpoints API, you can authenticate users who are attempting to invoke your frontend APIs. Cloud Endpoints supports user authentication with Firebase, Auth0 *[auth zero]*, Google authentication, and other custom authentication methods. After the user signs in, the authentication providers send a signed JSON Web Token (or 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 request.

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

## You can restrict API access

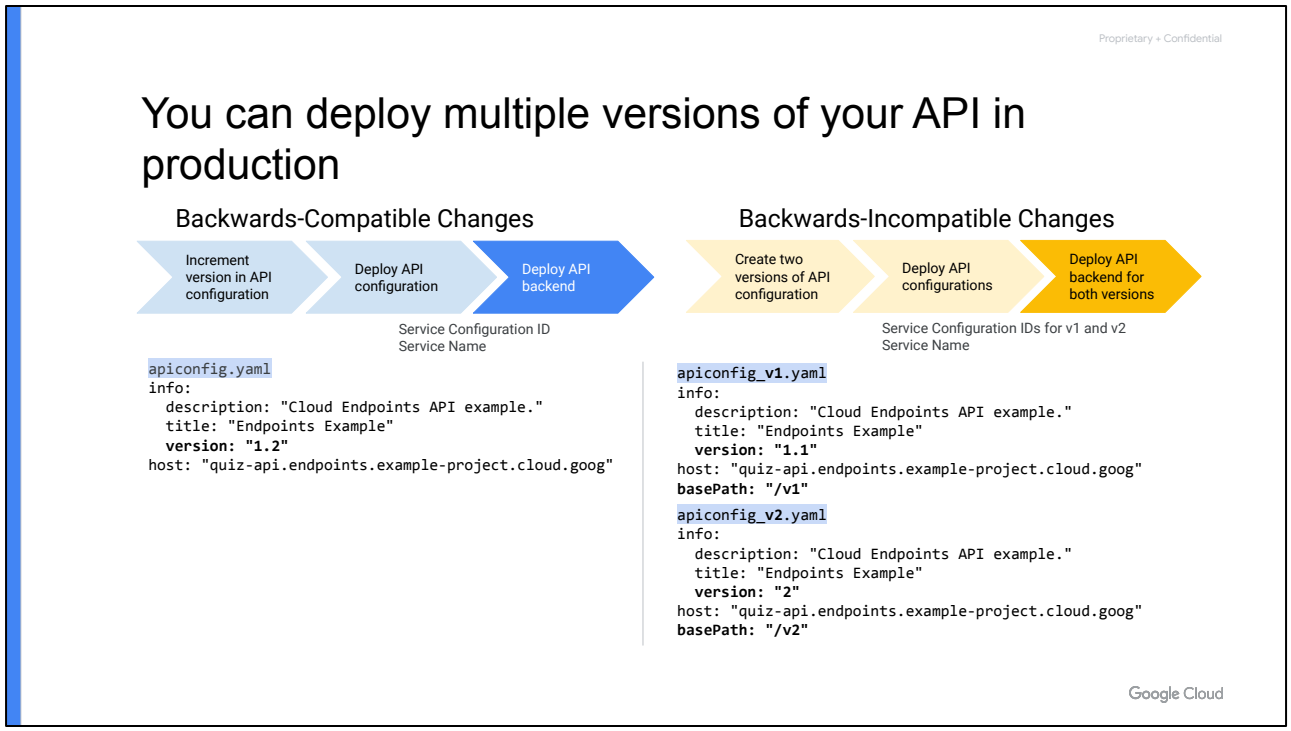





To restrict access to your Cloud Endpoints API, you can assign Cloud Identity and Access Management (Cloud IAM) roles to specific users. You can grant access to a specific API or to the entire Cloud project.

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.



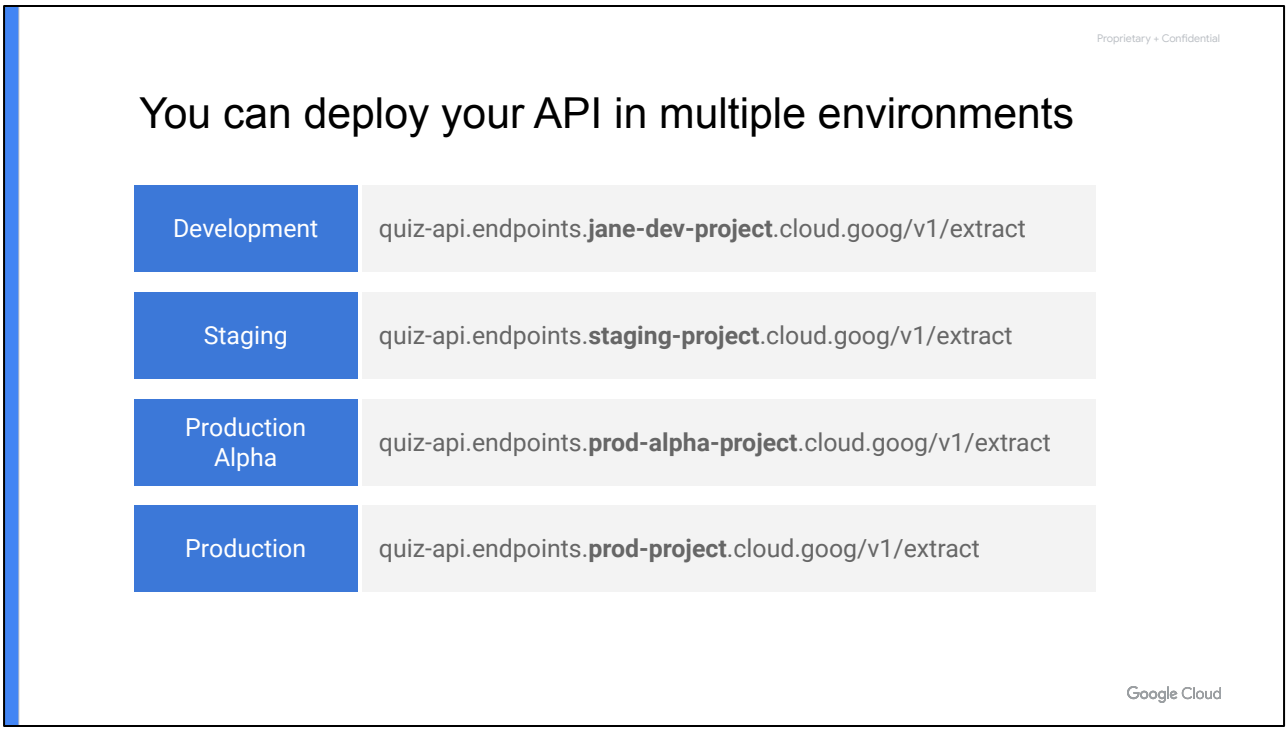

You might make small changes, bug fixes and performance enhancements to your API backend. These changes are usually backwards compatible. In such cases, it's a good practice to increment the version attribute in the Cloud Endpoints API configuration and then redeploy the API configuration and the API backend. In the example shown, we've increased the version number from 1.1 to 1.2 because this is a backwards-compatible change.

If you make changes that are not backwards compatible and will break functionality for consumers of your API, deploy two versions of your Cloud Endpoints API by creating a separate API configuration for each version. The “g-cloud service management deploy” command will return a different service configuration ID for each version. Update the API backend configuration of each version with a corresponding service configuration ID. In the example shown, we have different API configurations for version one of the API and version two of the API.

You can also delete versions of your API when they're 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

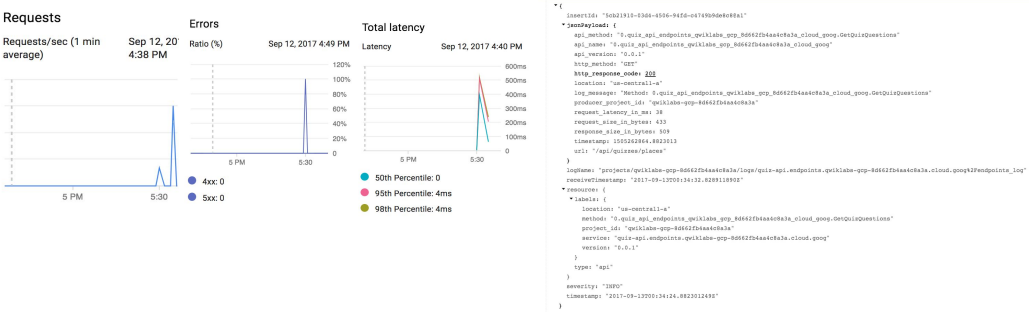



When developing and deploying APIs, you will have separate environments for developing and staging limited releases, such as private alphas, and for production.

You can create separate projects for each environment and deploy a 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 higher 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



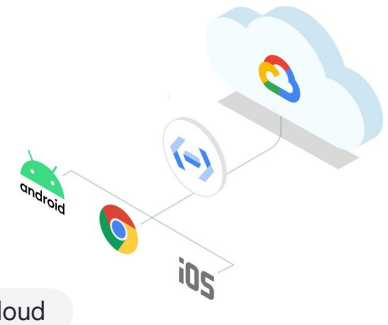

| Method | Requests | Avg req size | Avg resp size | 4xx | 5xx | Latency 98% | Latency 95% | Latency median |
|---|---|---|---|---|---|---|---|---|
| GET /api/quizzes/* | 10 | 430 B | 509 B | 0 | 0 | 472 ms | 393 ms | 66 ms |



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

## Cloud Endpoints

- Full control over the API gateway
- Uses gRPC
- You must manage the lifecycle
- You pay for Compute Engine VMs
- Service proxy required for services outside Google Cloud





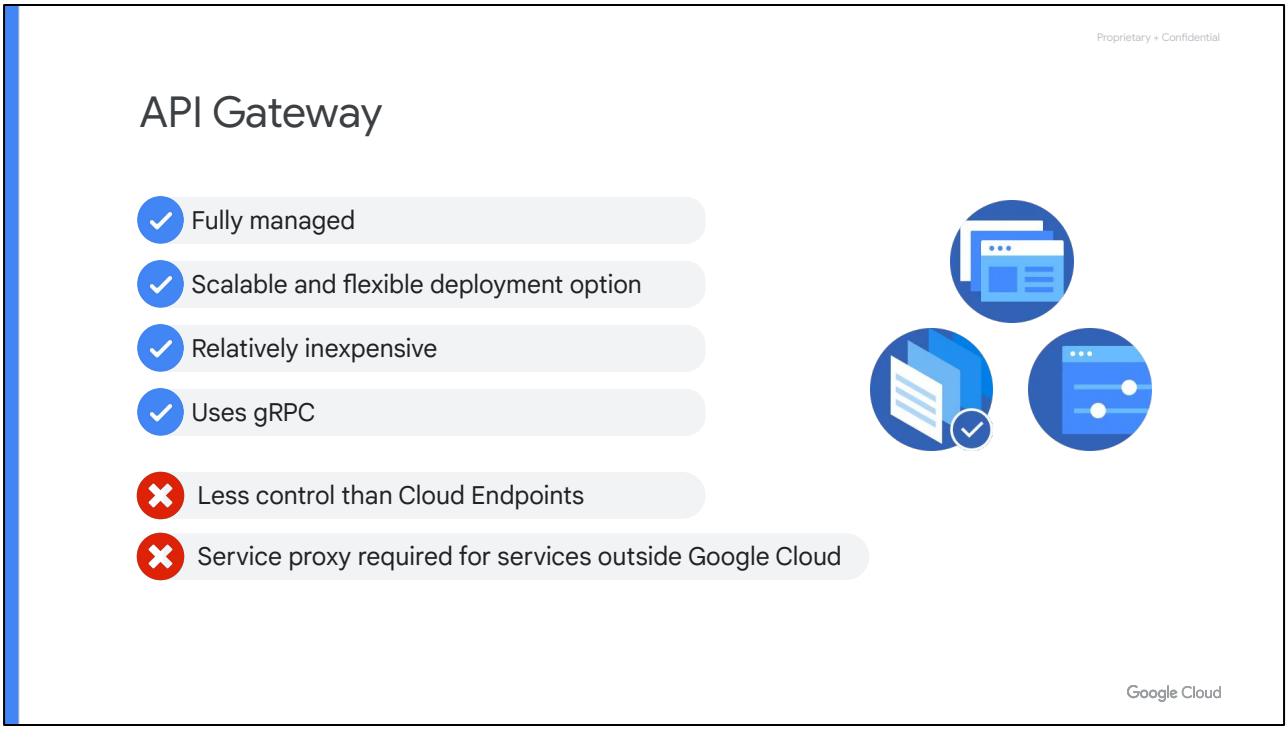

Let's explore some of the reasons why you might choose one platform over another. First, let's look at some of the reasons why you might choose Cloud Endpoints.

Cloud Endpoints is a Google Cloud API management solution that gives you full control over your API gateway because it is deployed, configured, and managed by you. It supports gRPC, a modern, open-source, high-performance Remote Procedure Call (RPC) framework that can run in any environment.

On the negative side, Cloud Endpoints forces you to manage the entire lifecycle of the NGINX/Envoy-based gateway, including security updates, traffic migration, and managing the compute platform. You also pay for Compute Engine VMs.

In addition, proxied services outside Google Cloud require a service proxy in Google Cloud.

Now, let’s look at some of the reasons why you might choose API Gateway.




Google Cloud's API Gateway enables you to develop, deploy, secure, and manage your APIs using a fully-managed gateway. You can provide secure access to your backend services through a well-defined REST API that is consistent across all of your services, regardless of service implementation. Why might you choose API Gateway?

API Gateway is fully managed by Google, which allows you to take advantage of all the operational benefits of serverless technology. Google handles management operations like security and configuration updates, while providing you flexible deployment and scalability. API Gateway is often a better solution when using serverless app platforms like Cloud Functions, Cloud Run, and App Engine. It's relatively inexpensive to run as you you don't pay for and manage Compute VMs. API Gateway also supports gRPC.

However, since API Gateway is managed by Google, it allows less control than Cloud Endpoints. Also, like Cloud Endpoints, proxied services outside Google Cloud require a service proxy in Google Cloud.

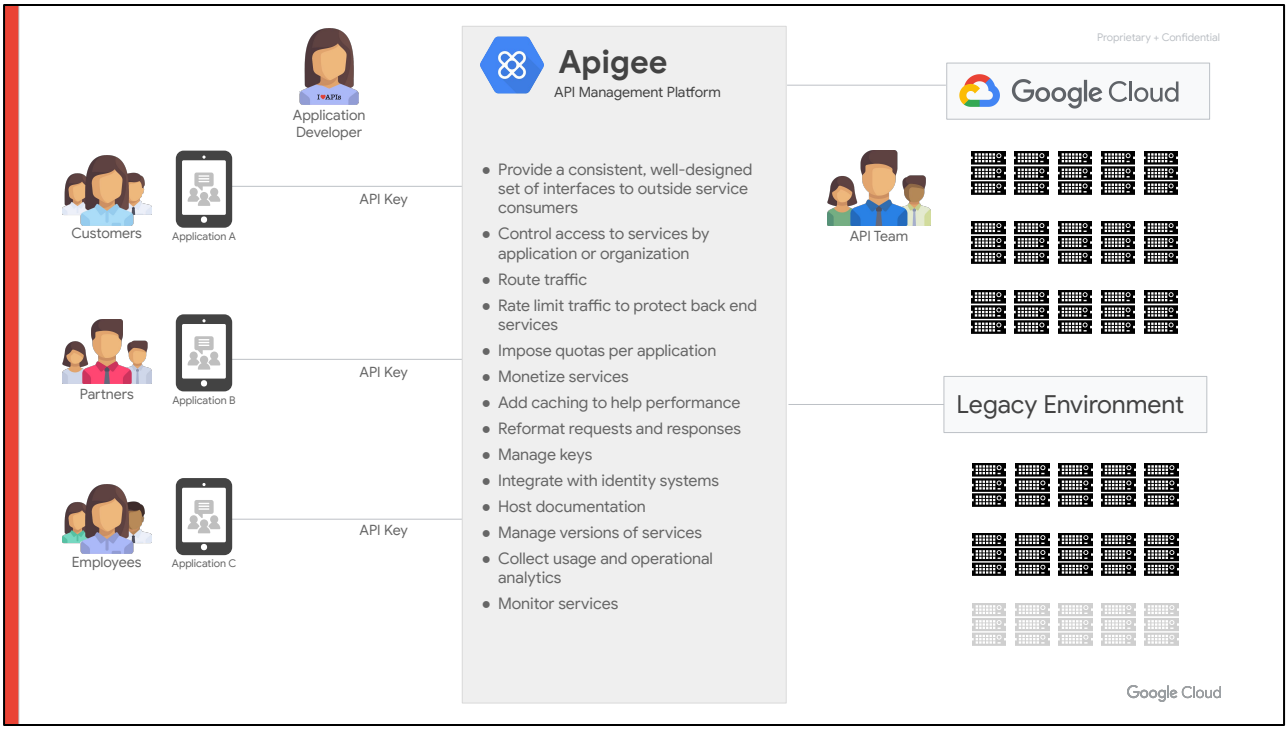
Now let's look at some of the reasons why you might choose the Apigee API Platform.

# API development

REST / gRPC

Cloud Endpoints

Apigee

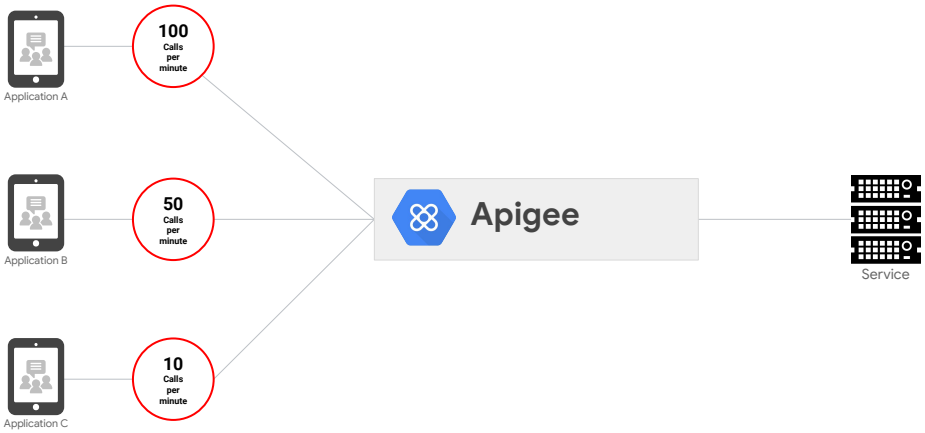



Apigee provides a consistent, well-designed set of interfaces to the outside consumers.

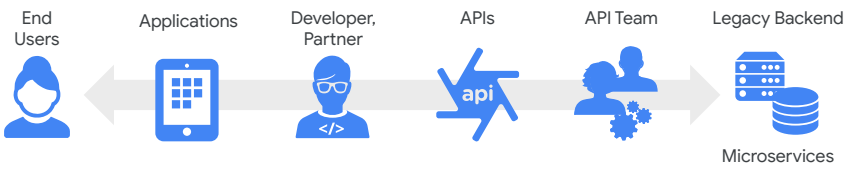
It can:

- Provide a consistent, well-designed set of interfaces to outside service consumers
- Control access to services by application or organization
- Route traffic
- Rate limit traffic to protect back end services
- Impose quotas per application
- Monetize services
- Add caching to help performance
- Reformat requests and responses
- Manage keys
- Integrate with identity systems
- Host documentation
- Manage versions of services
- Collect usage and operational analytics
- Monitor services

# Manage Traffic



# API Value Chain






This is the API Value Chain. It's a common pattern of success as we talk to our customers. Customers who have well managed APIs see better business results. They're able to deliver new experiences faster and take advantage of new opportunities in the market.

Success is driven by how they think about their APIs. It requires a shift in thinking away from the traditional view of APIs as just an integration tool as opposed to a product used to empower developers to create great applications for end users.

Let's start from the end users on the left side:

End users: we want to deliver a great experience to the end user… <click>
Applications: this means we need great applications… <click>
Developer: so we need to empower the developers and make it easy for them to create great applications… <click>
APIs: we can do this by providing great APIs that are consistent and easy to use… <click>

API team: we will need an API team to manage the APIs

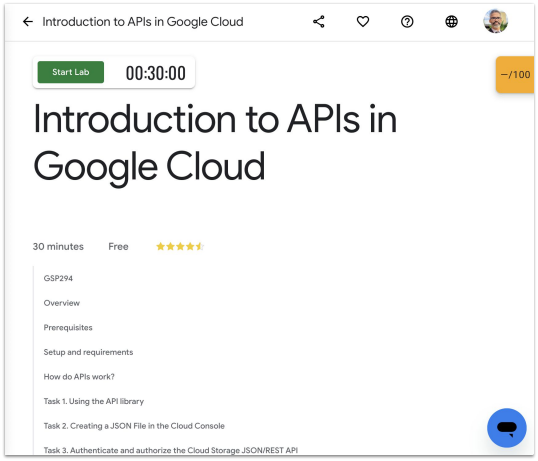
This is the modern way to expose legacy backend services and new microservices.

Thinking “outside in”, as described in this API value chain, is the key to success.

# Lab - Introduction to APIs in Google Cloud

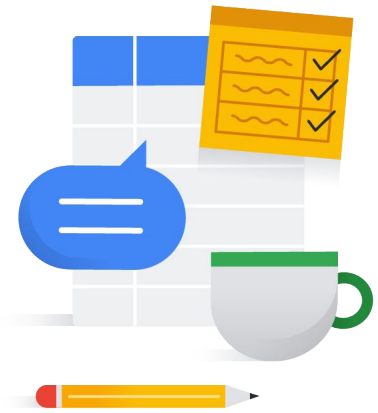
This lab reviews the architecture and basic functioning of APIs. This also provides hands-on practice, where you configure and run Cloud Storage API methods in Google Cloud Shell. Oauth 2.0 Explorer

https://partner.cloudskillsboost.google/focuses/11595?parent=catalog



# Program issues or concerns?

- Problems with **accessing** Cloud Skills Boost for Partners
  - partner-training@google.com
- Problems with **a lab** (locked out, etc.)
  - support@qwiklabs.com
- Problems with accessing Partner Advantage
  - https://support.google.com/googlecloud/topic/9198654




Google Cloud