

# Cloud Computing APIs

---

CIS437

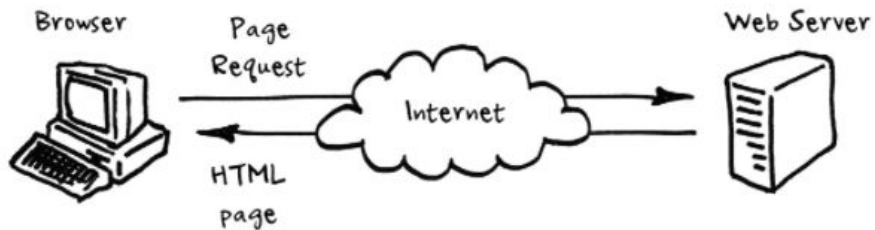
Erik Fredericks // [frederer@gvsu.edu](mailto:frederer@gvsu.edu)

*Adapted from Google Cloud Computing Foundations, Overview of Cloud Computing (Wufka & Canonico)*

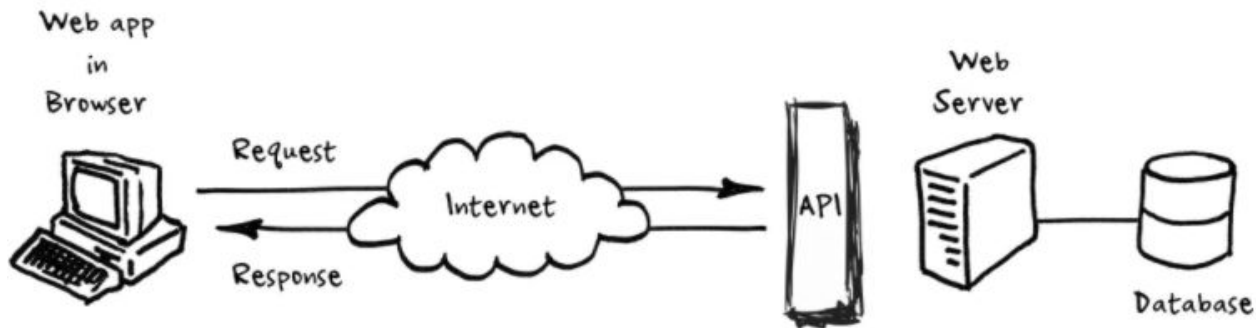
# Overview

Application programming interfaces (APIs)

Google options for APIs (i.e., so you don't have to write your own)



Web 1.0 – How the Internet was



The Web as a client-server application framework

## EXAMPLE: HTML

```
<form action="save.php" method="post">  
  <input type="text" name="email" value=""/>  
  <input type="submit" value="Post"/>  
</form>
```

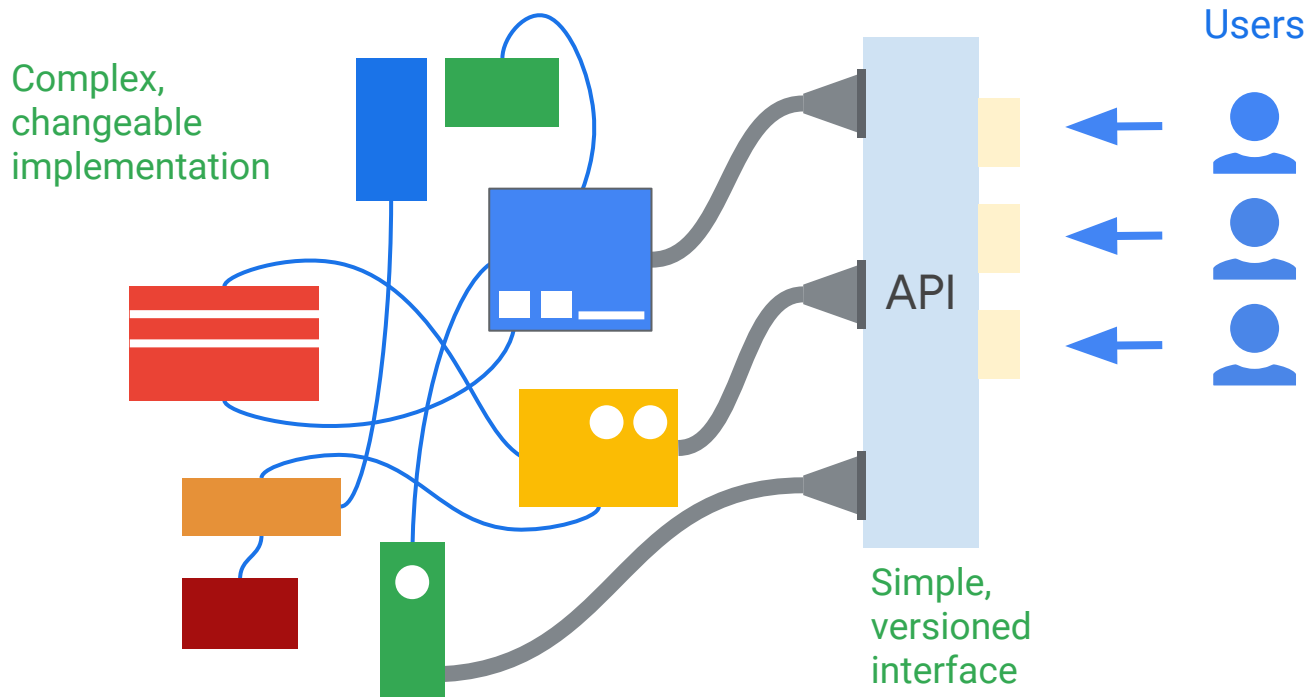
- Meaning: when submit button is pressed:
  - form the URL (<http://example.com/save.php>)
  - submit an HTTP **POST** with the value of the email text input field included in the body.
  - you might get back a redirect status code, follow it and render; replacing current page with new content.

## EXAMPLE: HTML

```
<link type="text/css" rel="stylesheet"  
href="css/fluid_grid.css" />
```

- Meaning: when link is encountered in the HTML
  - create URI from href ([http://example.com/css/fluid\\_grid.css](http://example.com/css/fluid_grid.css))
  - fetch via an HTTP **GET**.
  - resulting document should be CSS which contains rules that will guide the styling in the page render.

# APIs hide the details and enforce contracts



---

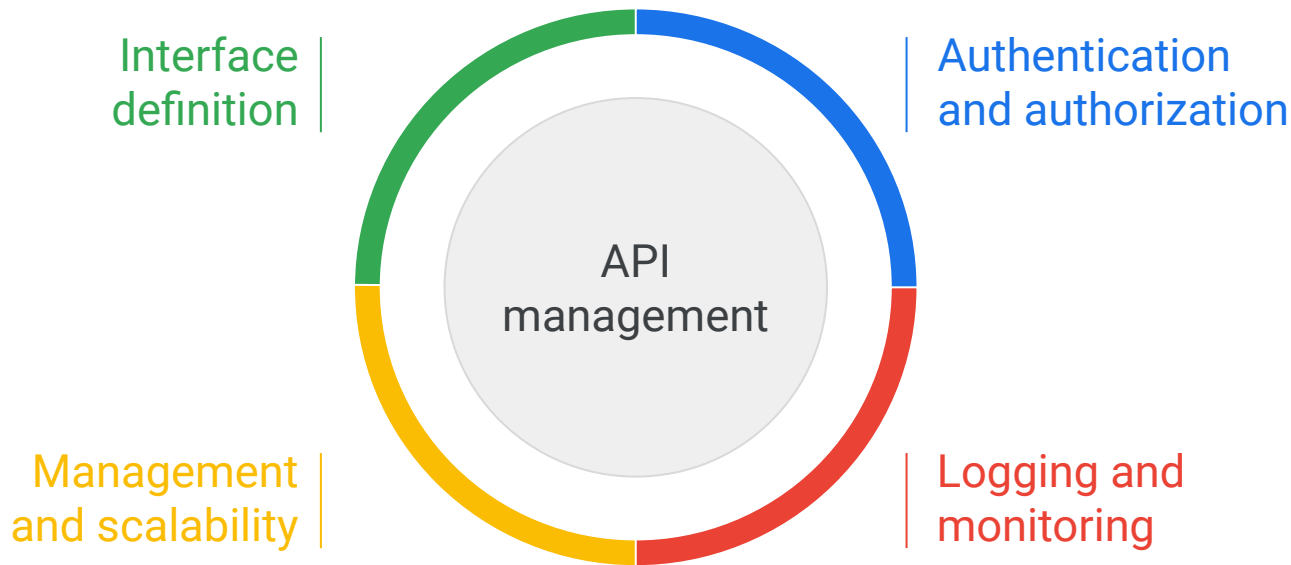
# What is a REST API?

- A set of constraints a service must comply with.
- An API that uses HTTP requests to GET, PUT, POST, and DELETE data.
- Designed to set up a format for applications to communicate.
- Great for cloud applications because they are stateless.
- Authentication via OAuth and security by leveraging tokens.



{REST}

# Deploying and managing APIs can be difficult



# APIs

<https://developer.twitter.com/en/docs/twitter-api/getting-started/about-twitter-api>

<https://discord.com/developers/docs/intro>

<https://www.reddit.com/dev/api>

...



# No real standard is in place

There are RESTful suggestions...

<https://www.w3.org/2001/sw/wiki/REST>

There are competing suggestions

<https://standards.rest/>

There are other things we can do:

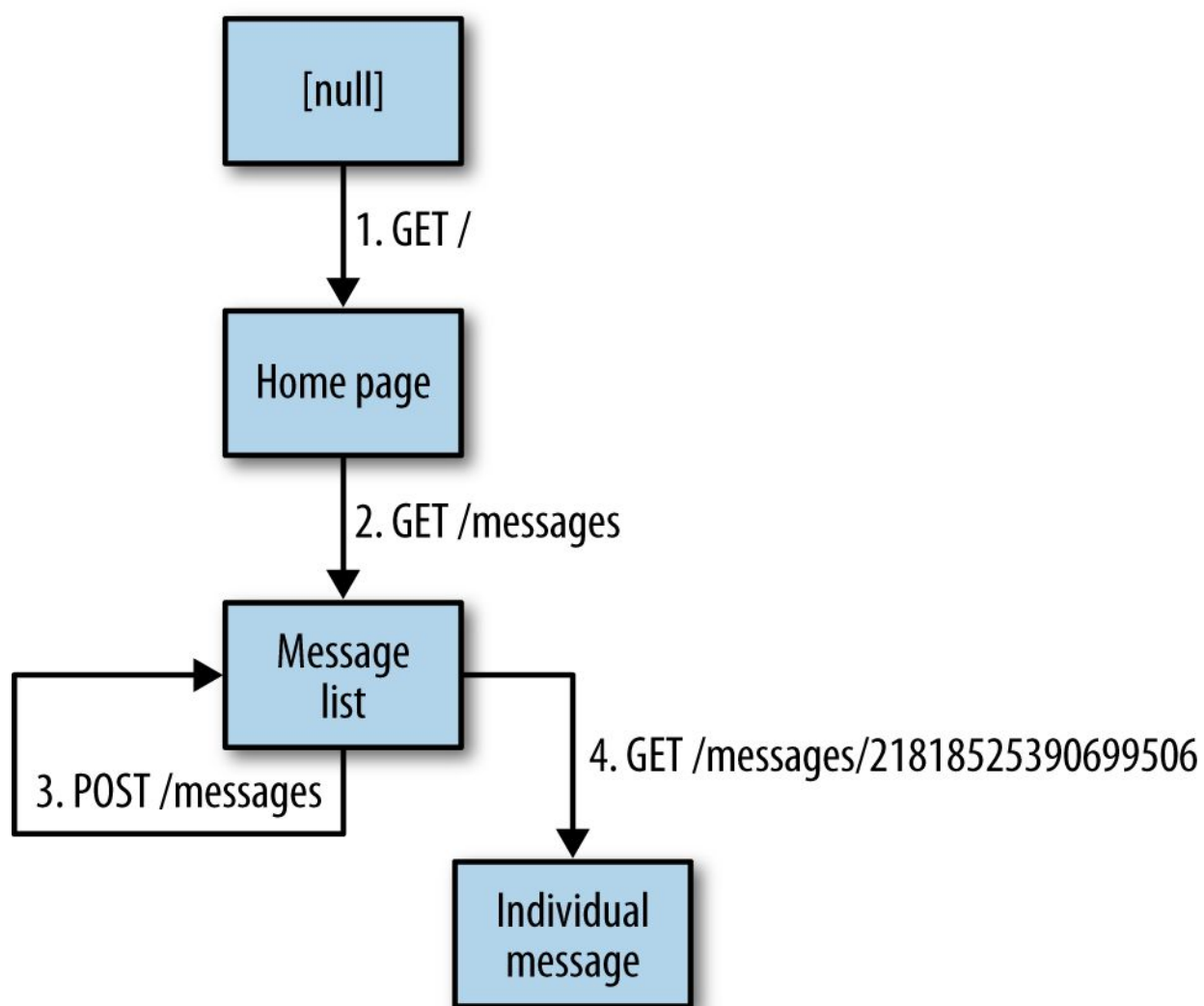
Microformats: <https://developer.mozilla.org/en-US/docs/Web/HTML/microformats>  
(i.e., embed in HTML)

Atom web standards: [https://en.wikipedia.org/wiki/Atom\\_%28Web\\_standard%29](https://en.wikipedia.org/wiki/Atom_%28Web_standard%29)

Or you could write your own...

# Creating (your own) API (1/2)

1. List information clients want from API
  - a. Semantic descriptors (i.e., what your calls 'mean')
2. Draw a state diagram
  - a. Transitions = links
3. Use existing link descriptions / semantic descriptors where possible
  - a. [https://www.w3schools.com/html/html5\\_semantic\\_elements.asp](https://www.w3schools.com/html/html5_semantic_elements.asp)
  - b. <https://www.smartdatacollective.com/html5-and-semantic-web/>



# Creating (your own) API

## 4. Select a data format

- JSON, etc.

## 5. Write a profile

- RFC 6906 - <https://datatracker.ietf.org/doc/html/rfc6906>
- i.e., formalized descriptions of link relationships and semantic meanings
- XMDP, ALPS, JSON-LD
  - <https://json-ld.org/>

## 6. Publish!

### A Simple Example

```
{
  "@context": "https://json-ld.org/contexts/person.jsonld",
  "@id": "http://dbpedia.org/resource/John_Lennon",
  "name": "John Lennon",
  "born": "1940-10-09",
  "spouse": "http://dbpedia.org/resource/Cynthia_Lennon"
}
```

# So what does this all mean?

There are numerous ways people have implemented REST

There is the "correct" way by the author of the entire idea

- Then there are the numerous interpretations by the entirety of the Internet



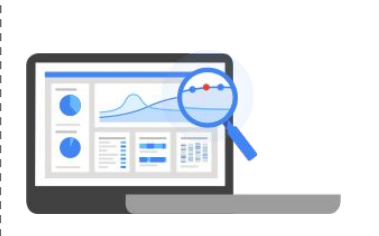
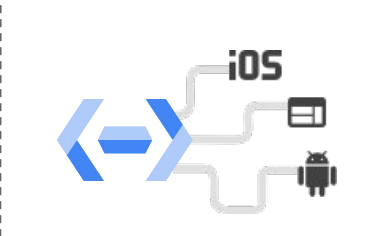
If you want some fun, read the reviews of the book on the subject:

<https://www.amazon.com/RESTful-Web-APIs-Services-Changing/dp/1449358063>

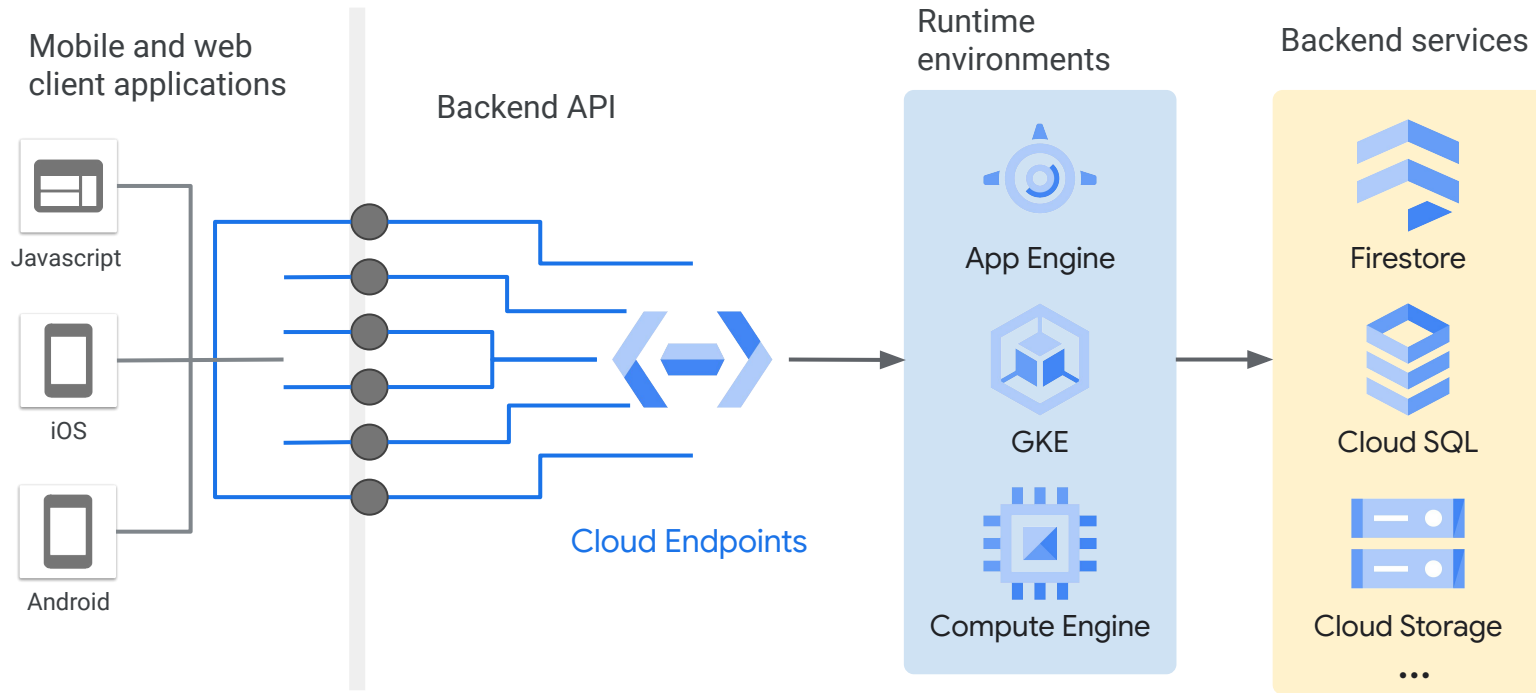
(they are not nice)

So back to Google...

# Cloud Endpoints help you create and maintain APIs

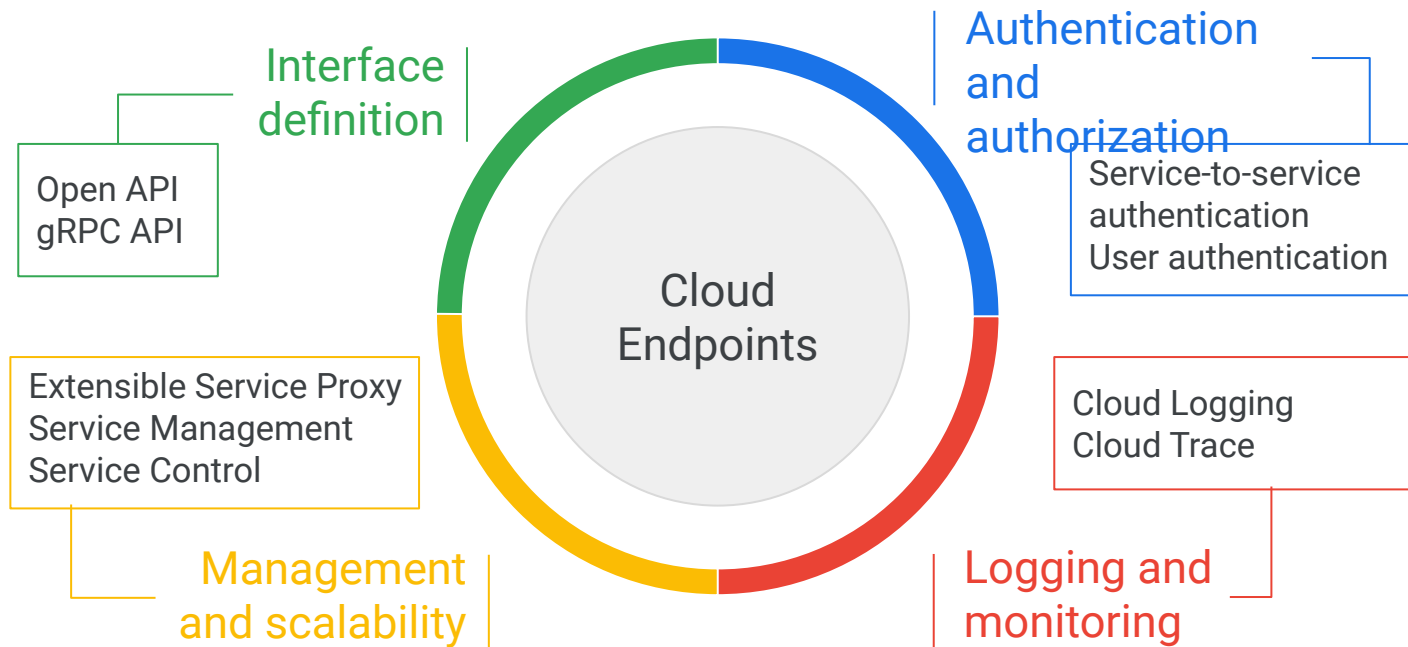
			
Protection	Speed	Monitoring and logging	Integration
<p>Generate and share API keys.</p> <p>Validate calls with JSON Web Tokens.</p> <p>Identify app users with Auth0 and Firebase Auth.</p>	<p>ESP provides security and insight &lt; 1ms.</p> <p>Automatic API deployment.</p>	<p>Inspect performance with Cloud Trace.</p> <p>Real-time log management with Cloud Logging.</p> <p>Analysis with BigQuery.</p>	<p>Choose your own framework and language.</p> <p>Upload an OpenAPI specification and deploy Google's containerized proxy.</p>

# Where Cloud Endpoints fit





# Cloud Endpoints makes it easier to deploy and manage APIs



---

# Lab Intro

## Cloud Endpoints: Qwik Start

Deploy a sample API with Google Cloud Endpoints and view the Cloud Endpoints Activity Graphs and Logs.

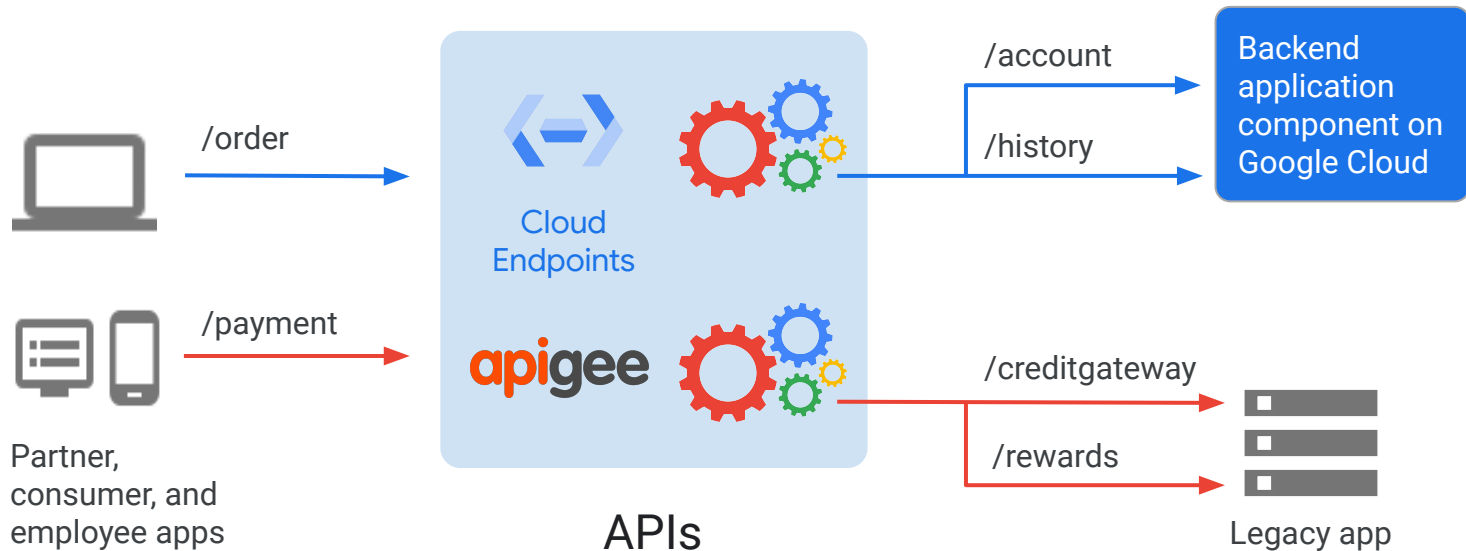
The lab can be found [here](#).



# Apigee Edge is a platform for developing and managing APIs



# An API gateway enables clients to retrieve data from multiple services with a single request



# Agenda

The Purpose of APIs

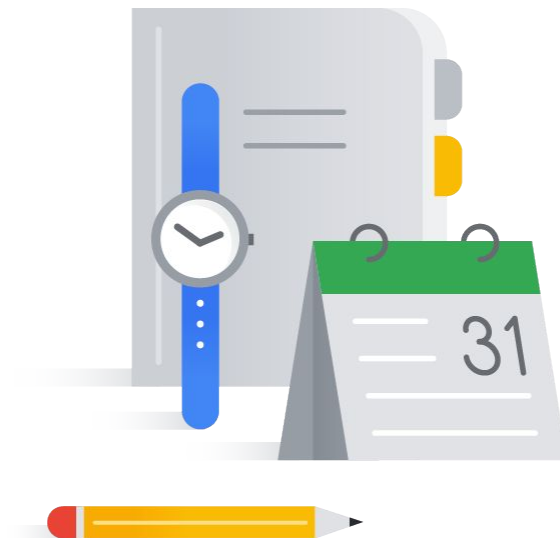
Cloud Endpoints

Lab: Cloud Endpoints: Qwik Start

Using Apigee Edge

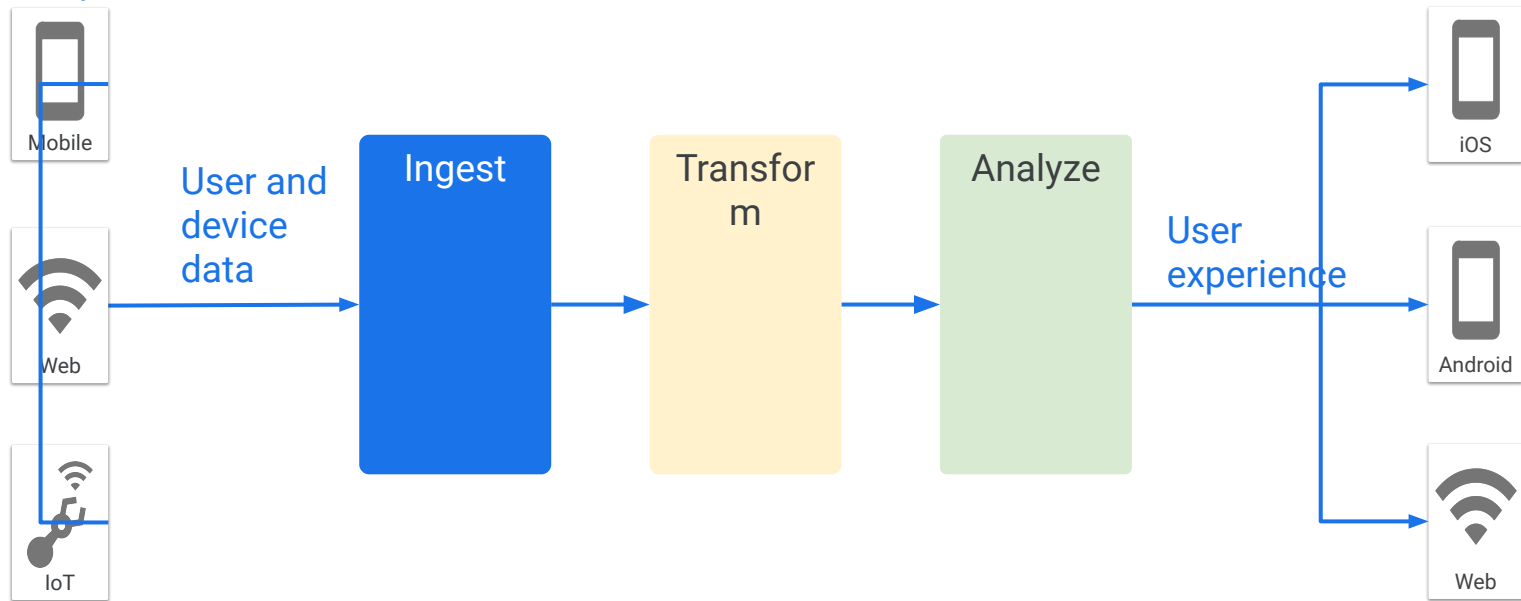
Managed Message Services

Pub/Sub

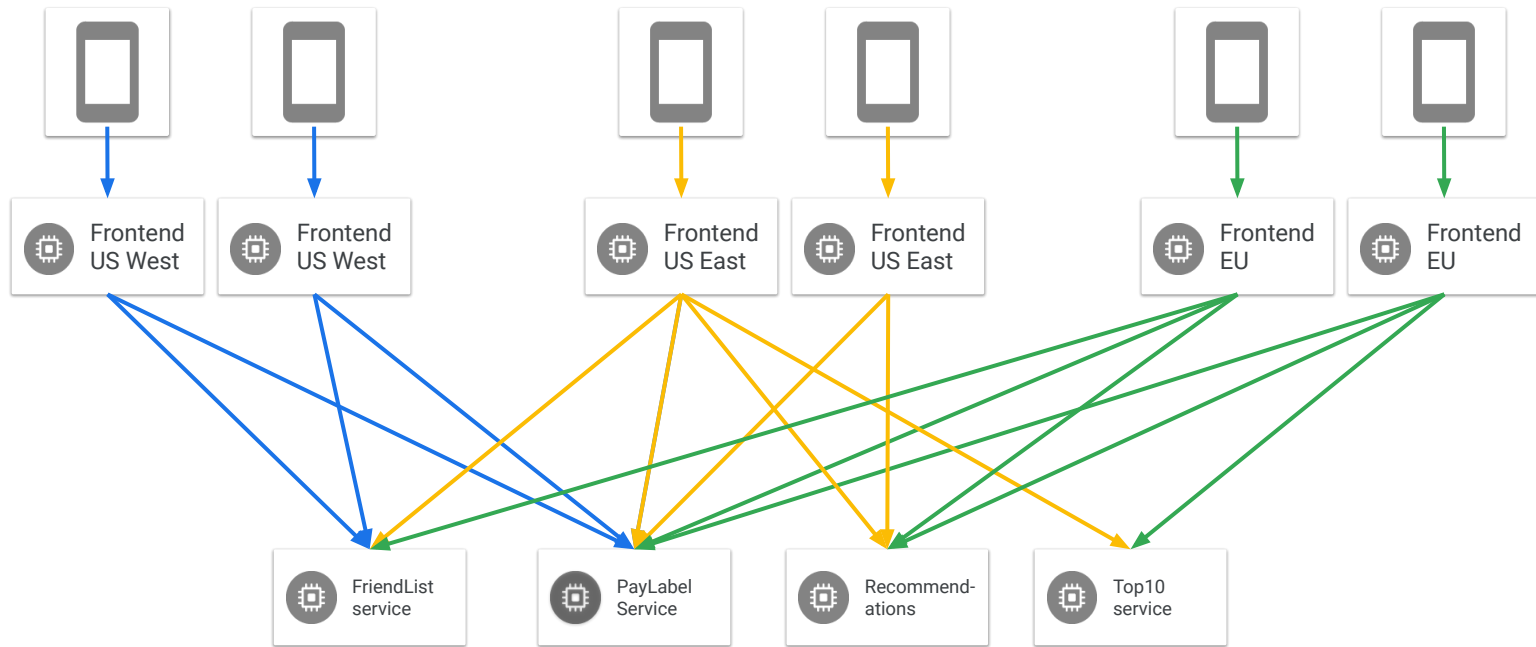


# Organizations have to rapidly ingest, transform, and analyze massive amounts of data

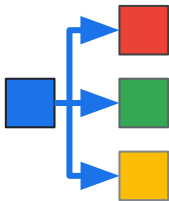
## Endpoint clients



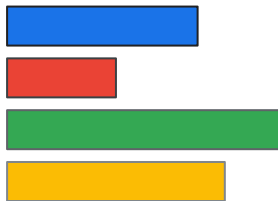
# Organizations have to orchestrate complex business processes



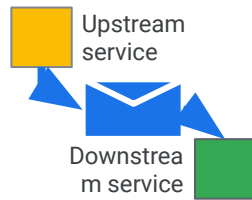
# Use cases for a managed messaging system



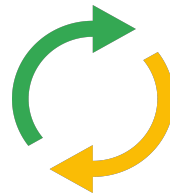
Balance workloads in network clusters



Implement asynchronous workflows



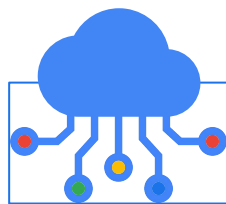
Distribute event notifications



Refresh distributed caches



Log to multiple systems



Stream data from various processes or devices



Reliability improvement



---

# Agenda

The Purpose of APIs

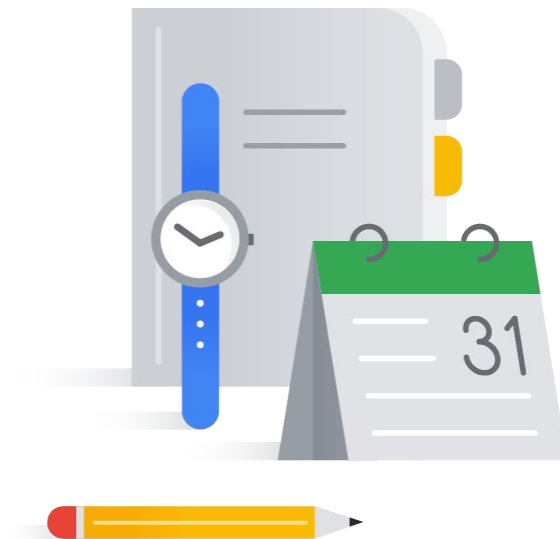
Cloud Endpoints

Lab: Cloud Endpoints: Qwik Start

Using Apigee Edge

Managed Message Services

Pub/Sub



# Pub/Sub simplifies global messaging and event ingestion



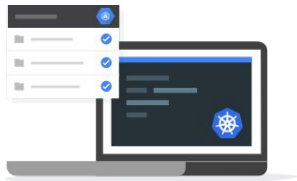
## Ingest events at any scale

No provisioning, partitioning, or load isolation worries.

Expand with global topics.

Enrich, deduplicate, order, aggregate, and land events using Dataflow.

Durable storage.



## Simple development of event-driven microservices

Reliable delivery of each event to the services that must react to it.

Push subs deliver the event to serverless apps.

Pull subs make events available to more complex stateful services.

Multi-region environments operate seamlessly.



## Be production ready from day one

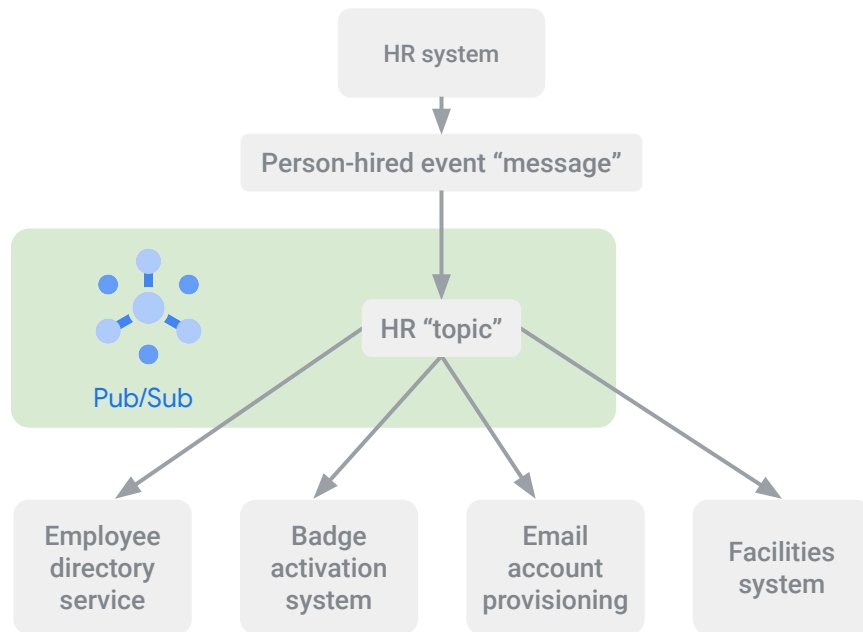
End-to-end encryption, IAM, and audit logging.

NoOps, fully automated scaling and provisioning.

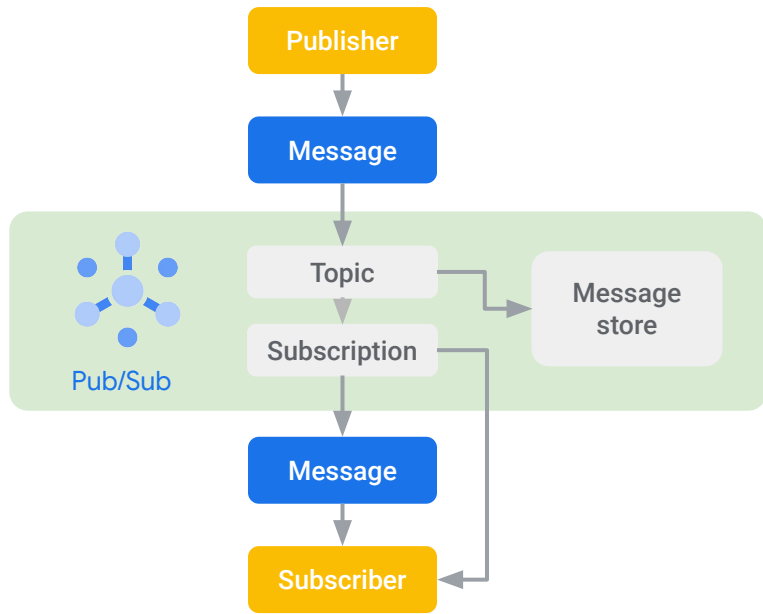
Extreme data durability and availability.

Native client libraries and an open-service API

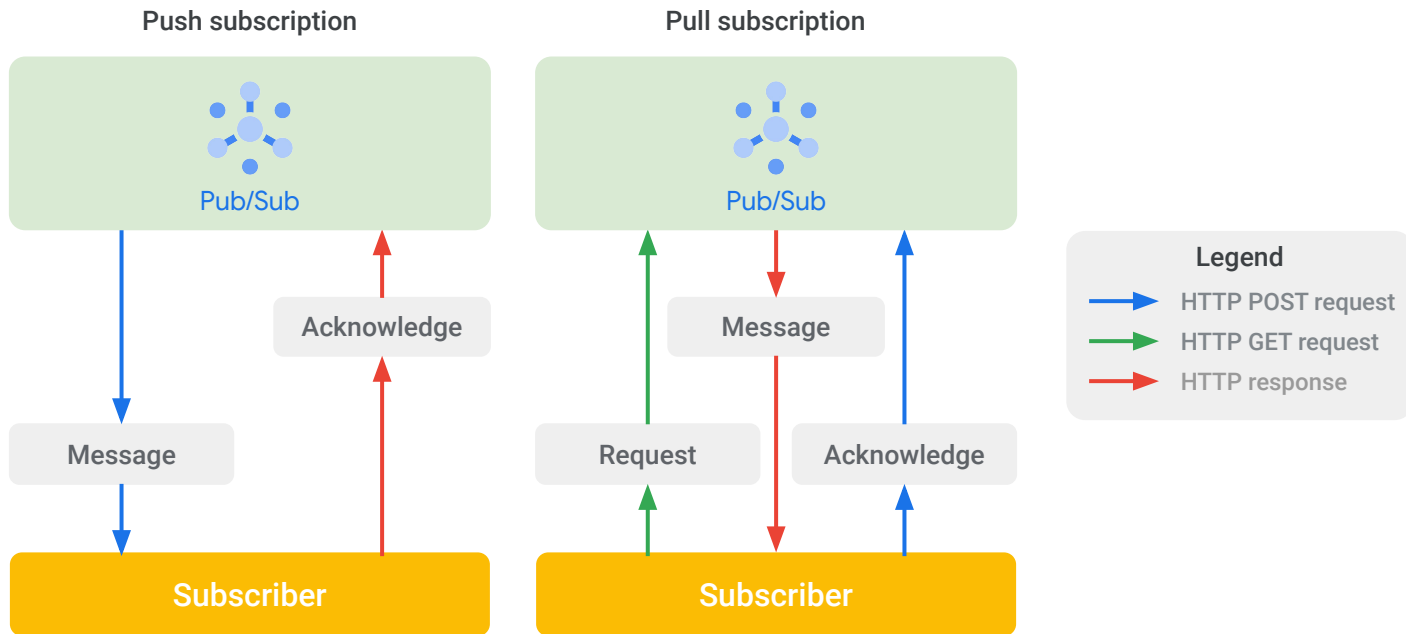
# Pub/Sub is used between data gathering and processing systems



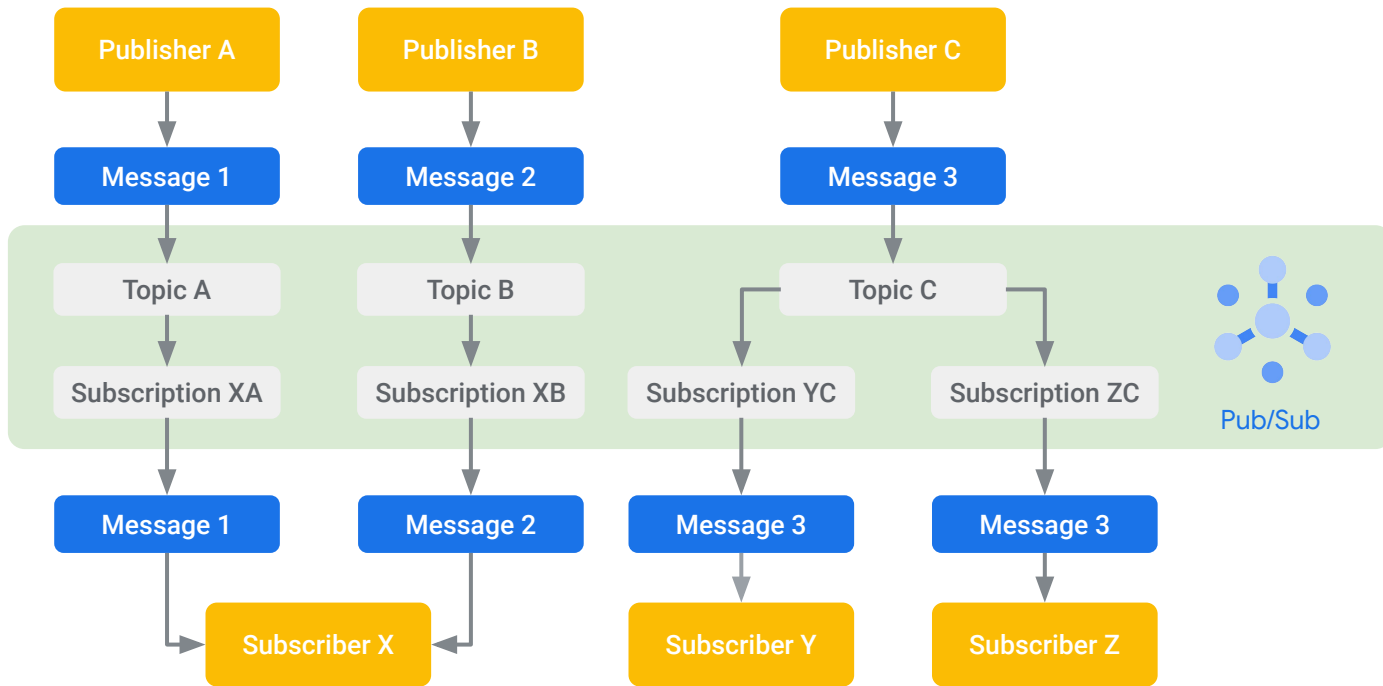
# Pub/Sub uses the Publish-Subscribe pattern



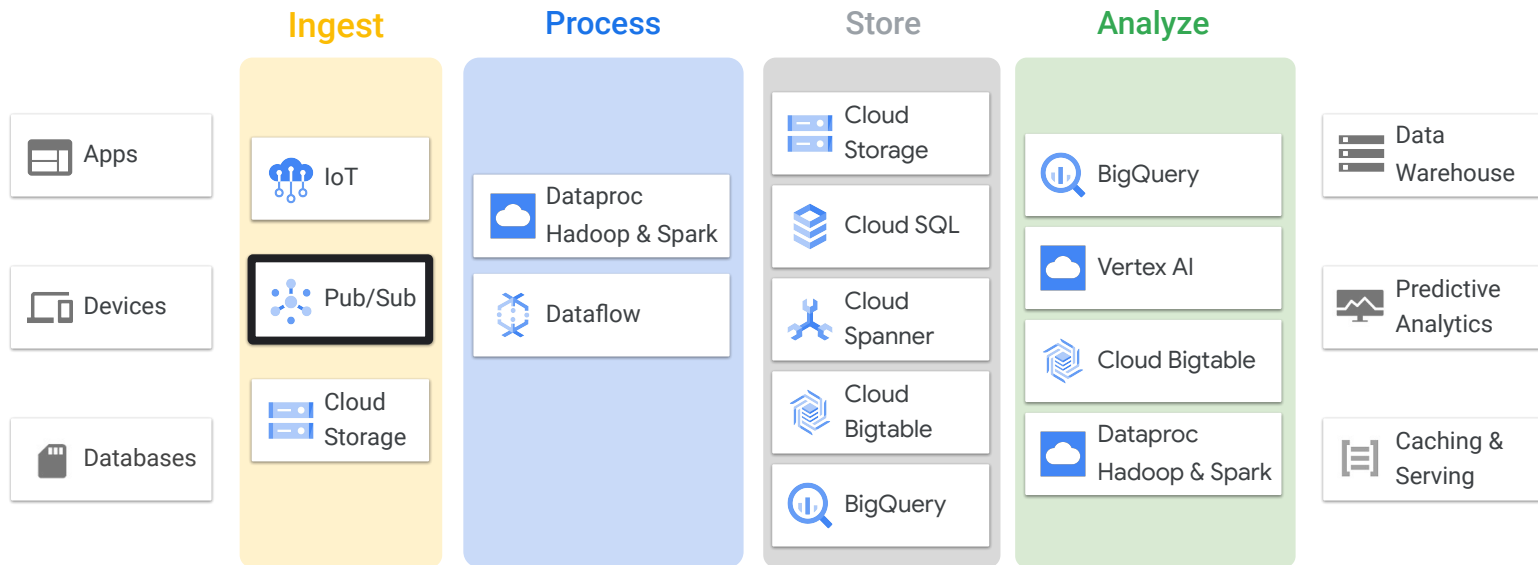
# Pub/Sub acts as a buffer between sending and receiving across software applications



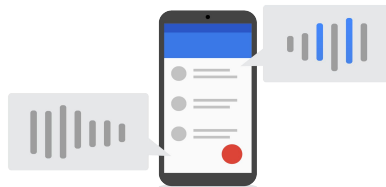
## A slightly more complex setup of Pub/Sub



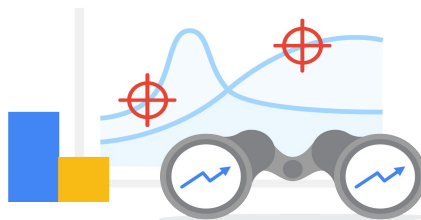
# The position of Pub/Sub within the big data processing model



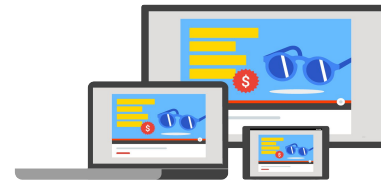
# Examples of Pub/Sub working



Chat and mobile



Instant search



Ads and budgets



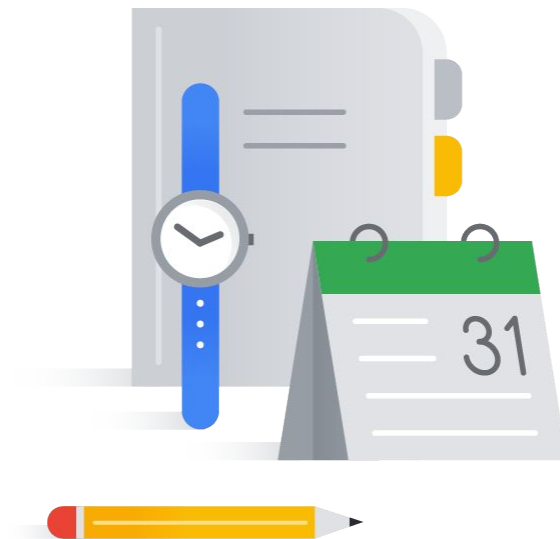
---

# Agenda

Lab: Cloud Pub/Sub: Qwik Start -  
Python

Quiz

Summary



---

# Lab Intro

## Google Cloud Pub/Sub: Qwik Start - Python

Publish messages with Pub/Sub using the  
Python client library.

You can find the lab [here](#).

