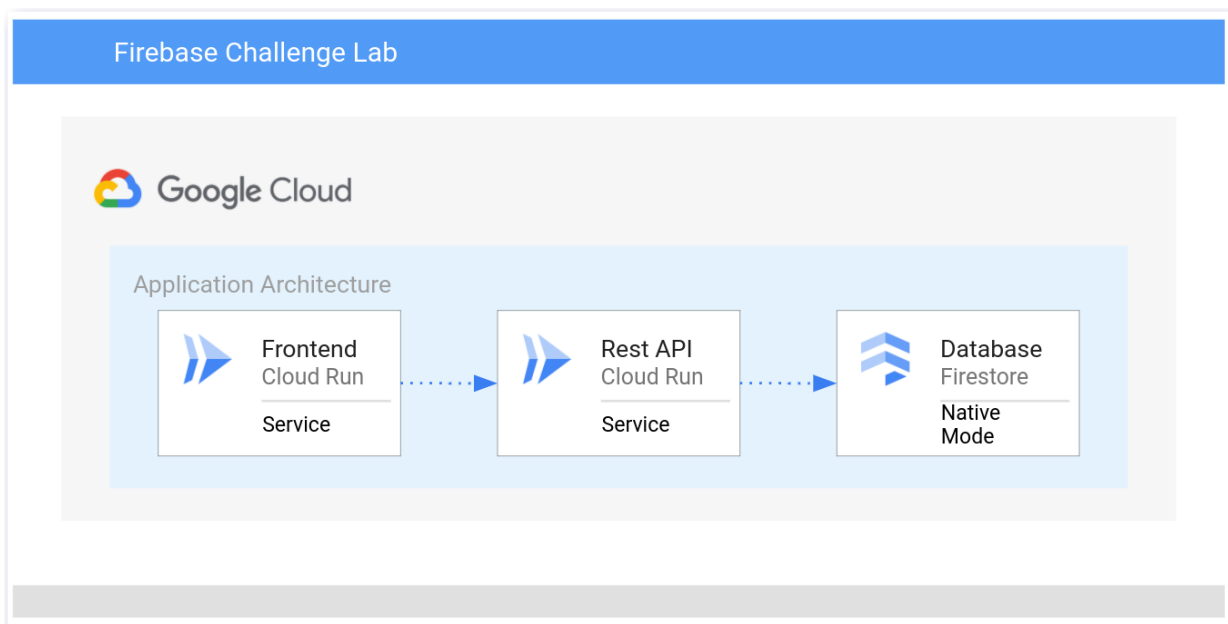


Develop Serverless Apps with Firebase: Challenge Lab

Challenge scenario

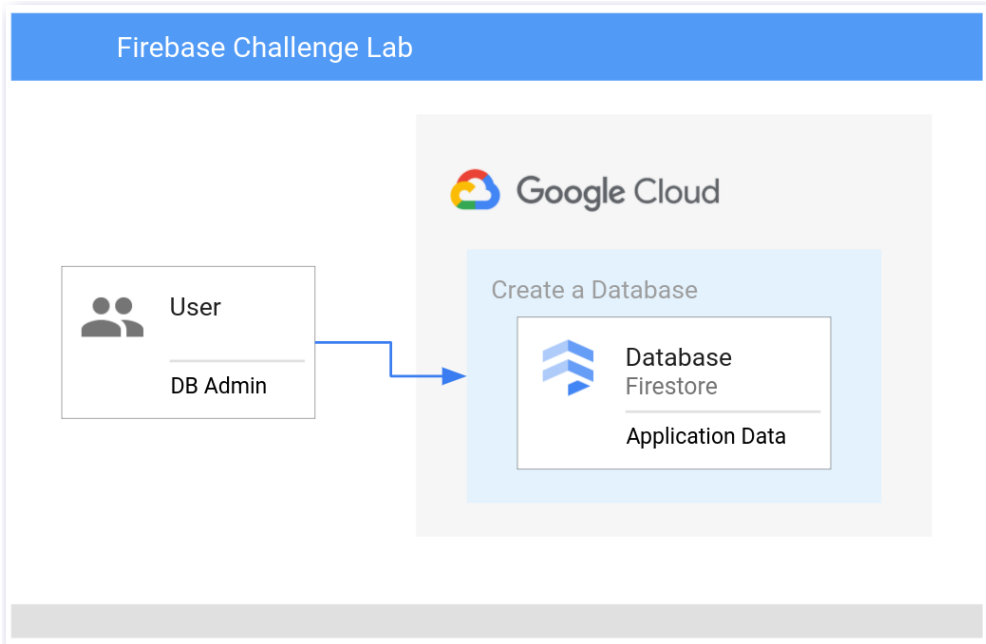
In this lab you will create a frontend solution using a Rest API and Firestore database. Cloud Firestore is a NoSQL document database that is part of the Firebase platform where you can store, sync, and query data for your mobile and web apps at scale. Lab content is based on resolving a real world scenario through the use of Google Cloud serverless infrastructure.

You will build the following architecture:



Task 1. Create a Firestore database

In this scenario you create a Firestore Database in Google Cloud. The high level architecture diagram below summarizes the general architecture.



Requirements:

Field	Value
Cloud Firestore	Native Mode
Location	Lab Region

Create a Firestore database

To complete this section successfully, you are required to implement the following:

- Cloud Firestore Database
- Use Firestore Native Mode
- Add location Lab Region



Firestore

All databases

Firestore is an enterprise-grade, serverless document database service—now with MongoDB compatibility

Easily store and read semi-structured JSON data on a differentiated Firestore database service with virtually unlimited scalability, industry-leading availability with up to 99.999% SLA, and single digit milliseconds read latency using flexible Firestore, Datastore, or MongoDB-compatible interfaces.

[CREATE A FIRESTORE DATABASE](#)

[VIEW FLEET HEALTH](#)



☐ Show deleted

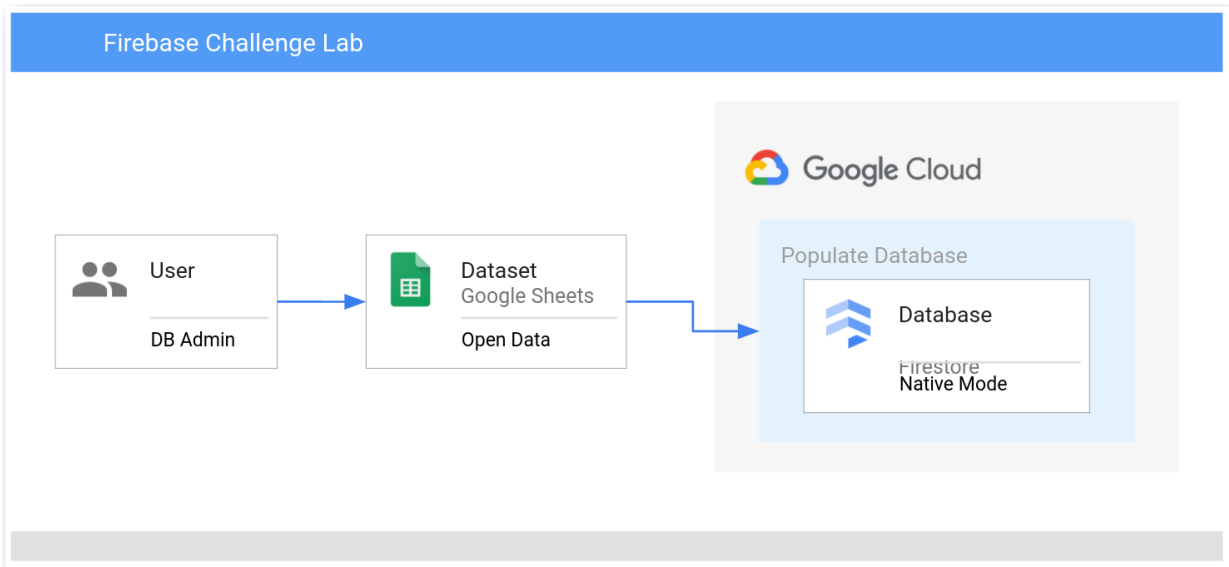
Filter

Database ID	Edition	Configuration	Location	Creation time (UTC+5:30)	Scheduled backups	Encryption type	Encryption key	Actions
(default)	Standard	Firestore Native	us-east4 (Northern Virginia)	Jun 30, 2025, 5:52:44 PM	Disabled Edit settings	Google-managed		

Task 2. Populate the database

In this scenario, populate the database using test data.

A high level architecture diagram below summarizes the general architecture.



Populate the database

Example Firestore schema:

Collection	Document	Field
data	70234439	[dataset]

The [Netflix Shows Dataset](#) includes the following information:

Field	Description
show_id:	Unique ID for every Movie / Tv Show
type:	Identifier - A Movie or TV Show
title:	Title of the Movie / Tv Show
director:	Director of the Movie
cast:	Actors involved in the movie / show
country:	Country where the movie / show was produced
date_added:	Date it was added on Netflix
release_year:	Actual Release year of the move / show
rating:	TV Rating of the movie / show
duration:	Total Duration - in minutes or number of seasons

The screenshot shows the Firebase Firestore console. On the left, the 'Database' sidebar is visible with options like 'Firestore Studio', 'Indexes', 'Import/Export', 'Disaster Recovery', 'Time-to-live (TTL)', 'Security Rules', 'Insights', 'Usage', 'Query insights', 'Monitoring', and 'Release Notes'. The main area displays the 'data' collection with a list of documents. The document with ID '1005494' is selected, and its details are shown on the right. The fields for this document are: cast, country, date_added, description, director, duration, listed_in, rating, release_year, and show_id.

1. Use the sample code from [pet-theory/lab06/firebase-import-csv/solution](#):

npm install

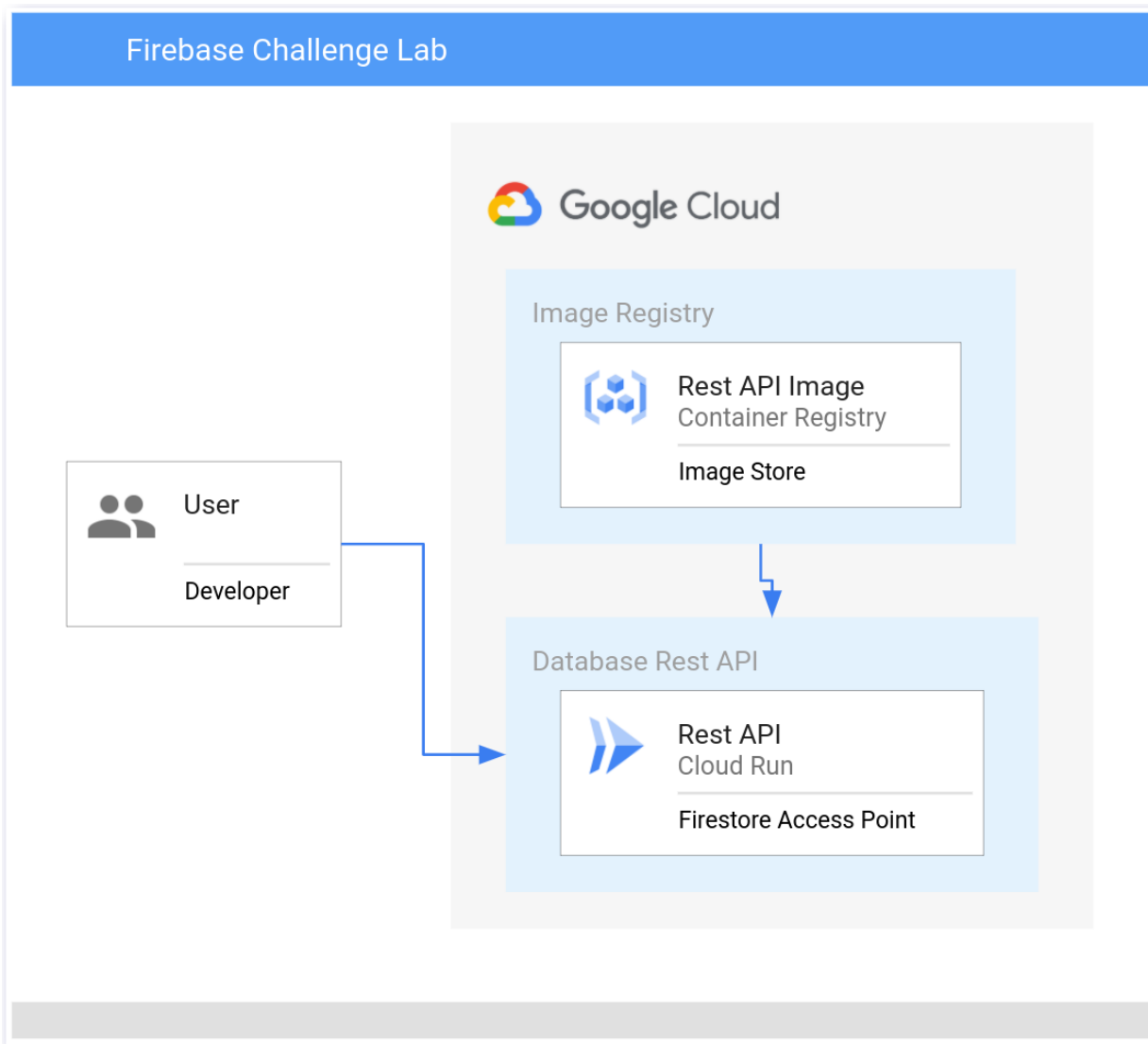
2. To import CSV use the node pet-theory/lab06/firebase-import-csv/solution/index.js:

```
node index.js netflix_titles_original.csv
```

Task 3. Create a REST API

In this scenario, create an example REST API.

A high level architecture diagram below summarizes the general architecture



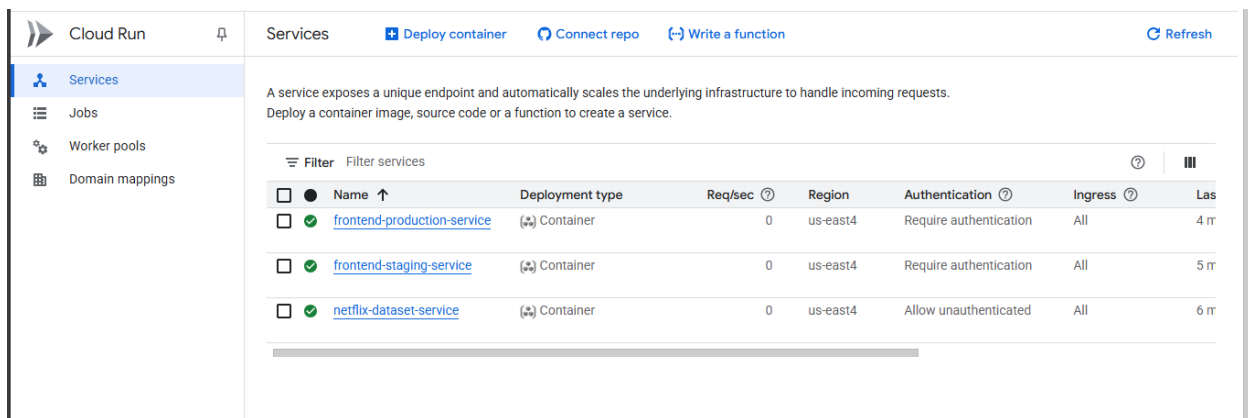
Cloud Run development

Field	Value
-------	-------

Container Registry Image	rest-api:0.1
Cloud Run Service	netflix-dataset-service
Permission	--allow-unauthenticated

To complete this section successfully, you are required to implement the following tasks:

1. Access pet-theory/lab06/firebase-rest-api/solution-01.
2. Build and Deploy the code to Google Container Registry.
3. Deploy the image as a Cloud Run service.
4. Go to Cloud Run and click **netflix-dataset-service** then copy the service URL:
 - SERVICE_URL=copy url from your netflix-dataset-service
 - curl -X GET \$SERVICE_URL should respond with: {"status":"Netflix Dataset! Make a query."}



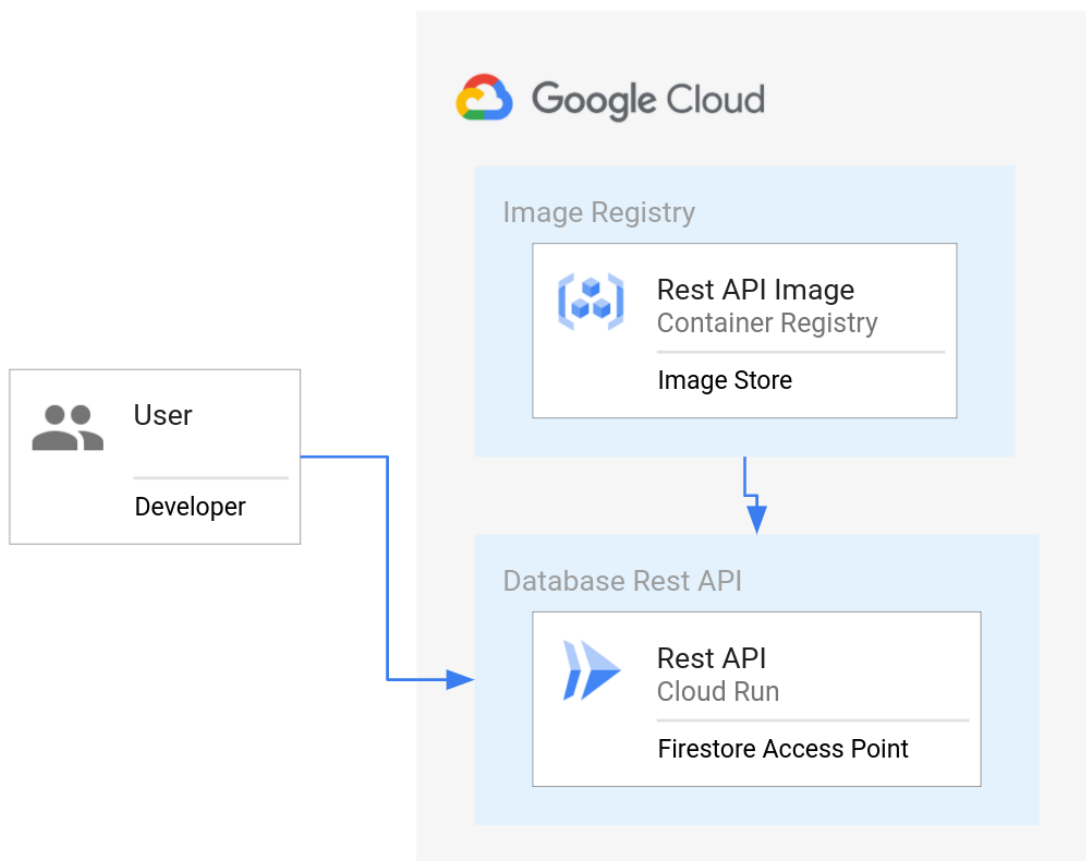
	Name	Deployment type	Req/sec	Region	Authentication	Ingress	Last updated
<input type="checkbox"/>	frontend-production-service	Container	0	us-east4	Require authentication	All	4 m
<input type="checkbox"/>	frontend-staging-service	Container	0	us-east4	Require authentication	All	5 m
<input checked="" type="checkbox"/>	netflix-dataset-service	Container	0	us-east4	Allow unauthenticated	All	6 m

Task 4. Firestore API access

In this scenario, deploy an updated revision of the code to access the Firestore DB.

A high level architecture diagram below summarizes the general architecture.

Firestore Challenge Lab



Deploy Cloud Run revision 0.2

Field	Value
Container Registry Image	rest-api:0.2
Cloud Run Service	netflix-dataset-service
Permission	--allow-unauthenticated

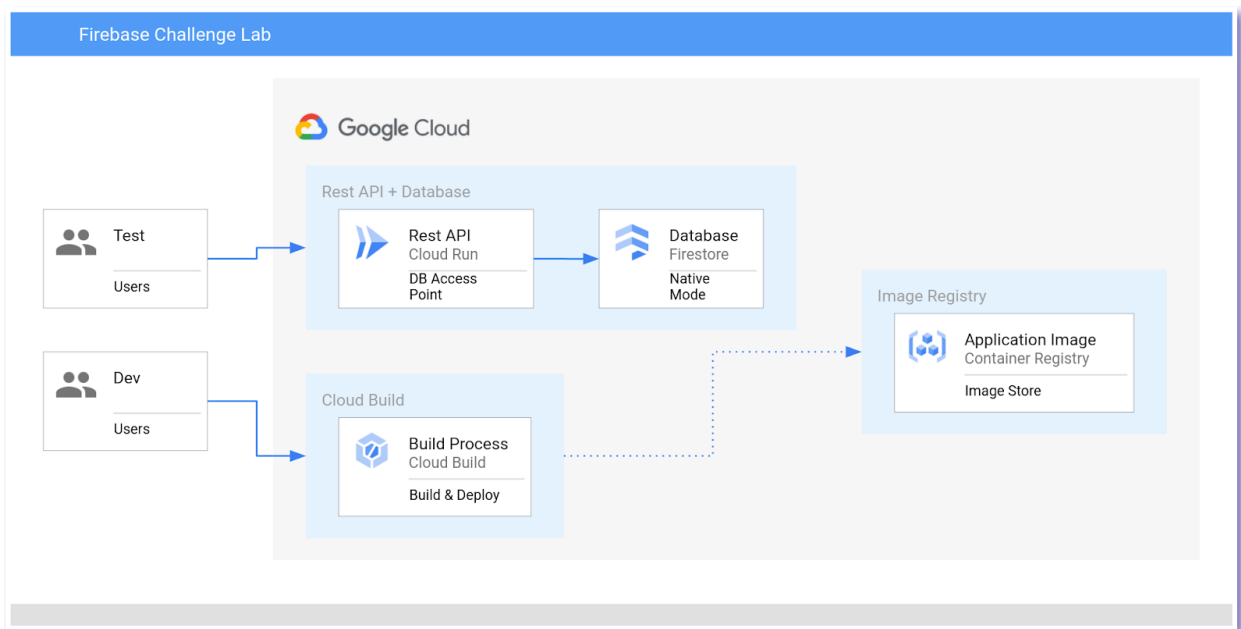
To complete this section successfully, you are required to implement the following tasks:

1. Access [pet-theory/lab06/firebase-rest-api/solution-02](#).
2. Build the updated application.
3. Use Cloud Build to tag and deploy image revision to Container Registry.
4. Deploy the new image a Cloud Run service.
5. Go to Cloud Run and click **netflix-dataset-service** then copy the service URL:
 - SERVICE_URL=copy url from your netflix-dataset-service
 - curl -X GET \$SERVICE_URL/2019 should respond with json dataset

Task 5. Deploy the Staging Frontend

In this scenario, deploy the Staging Frontend.

A high level architecture diagram below summarizes the general architecture.



Deploy Frontend

Field	Value
REST_API_SERVICE	REST API SERVICE URL
Container Registry Image	frontend-staging:0.1

Cloud Run Service

frontend-staging-service

To complete this section successfully, you are required to implement the following tasks:

1. Access pet-theory/lab06/firebase-frontend.
2. Build the frontend staging application.
3. Use Cloud Build to tag and deploy image revision to Container Registry.
4. Deploy the new image as a Cloud Run service.
5. Frontend access to Rest API and Firestore Database.
6. Access the Frontend Service URL.



INTRODUCTION TO SERVERLESS

A Serverless Night at the Movies

Sample application to demonstrate how to develop using Firestore.

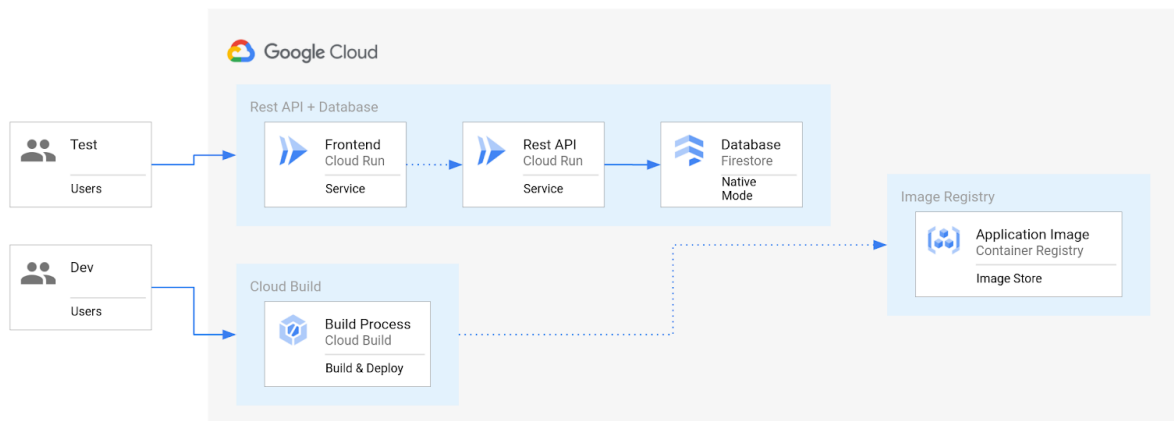
The dataset uses a Kaggle open dataset and both Rest API and Website use Google Cloud Run.

Title	Type	Rating	Director	Duration	Date
Norm of the North: King Sized Adventure	Movie	TV-PG	Richard Finn, Tim Maltby	90 min	September 9, 2019

Task 6. Deploy the Production Frontend

In this scenario, update the Staging Frontend to use the Firestore database.

A high level architecture diagram below summarizes the general architecture.



1. Access pet-theory/lab06/firebase-frontend/public.
2. Update the frontend application i.e. `app.js` to use the REST API.
3. Don't forget to append the year to the `SERVICE_URL`.
4. Use Cloud Build to tag and deploy image revision to Container Registry.
5. Deploy the new image a Cloud Run service
6. Frontend access to Rest API and Firestore Database.



A Serverless Night at the Movies

Sample application to demonstrate how to develop using Firestore.

The dataset uses a Kaggle open dataset and both Rest API and Website use Google Cloud Run.

Title	Type	Rating	Director	Duration	Date
Messiah	TV Show	TV-MA		1 Season	January 1, 2020
Nisman: The Prosecutor, the President, and the Spy	TV Show	TV-MA	Justin Webster	1 Season	January 1, 2020
Spinning Out	TV Show	TV-MA		1 Season	January 1, 2020
Kipo and the Age of Wonderbeasts	TV Show	TV-Y7- FV		1 Season	January 14, 2020
Live Twice, Love Once	Movie	TV-MA	Maria Ripoll	102 min	January 7, 2020
AJ and the Queen	TV Show	TV-14		1 Season	January 10, 2020
Go! Go! Cory Carson	TV Show	TV-Y		1 Season	January 4, 2020