Pitt-Google Alert Broker

Rubin PCW 2023

Michael Wood-Vasey (Pitt),
Troy Raen (Caltech/IPAC),
Christopher Hernández (Pitt)

Pitt-Google Broker: Goals and Objectives

- Provide alert stream widely to large numbers of users across a variety of science cases and expected rates.
- Provide:
  a. The public alert data to **everyone**
  b. **Scalable replication** to N brokers
  c. **Semantically compressed streams** to 1,000 consumers
  d. Public analog of PPDB: “**AlertDB**”?
  e. **Transient Host/Cross-matching** across surveys and wavelength
  f. **Re-playable classification** photometric + redshifts
- Non-Goal:
  a. We do not aim to provide a data analysis platform.

Pitt-Google Broker Design

Running on Google Cloud Platform with **cloud-native architecture**:

- collection of loosely coupled modules (microservices)
- running on compute services (nearly serverless) Cloud Functions, Cloud Run
- communicating through public Pub/Sub message streams
- streaming inserts to BigQuery tables (SQL access) and Cloud Storage buckets

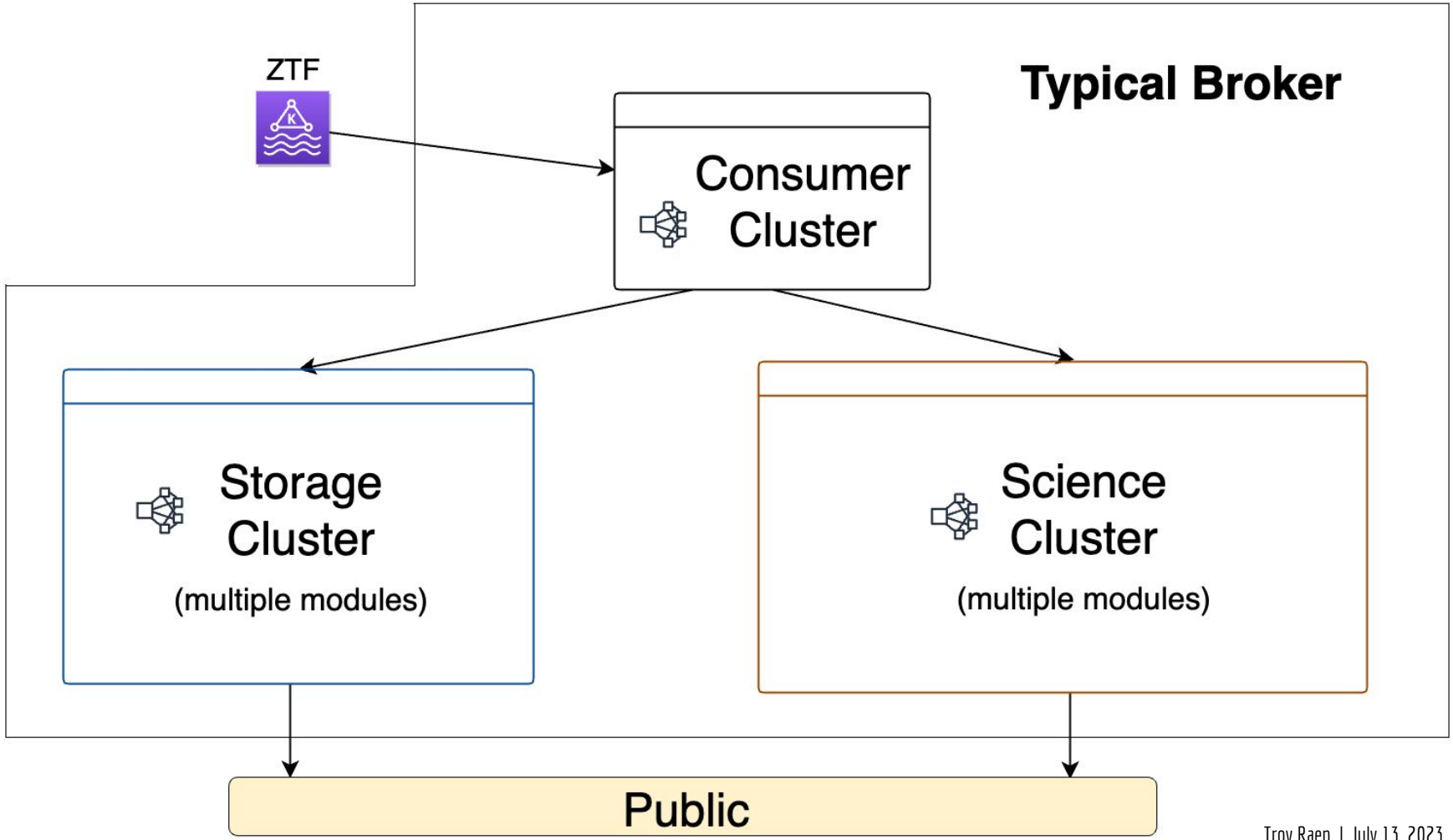
Pitt-Google Broker Design

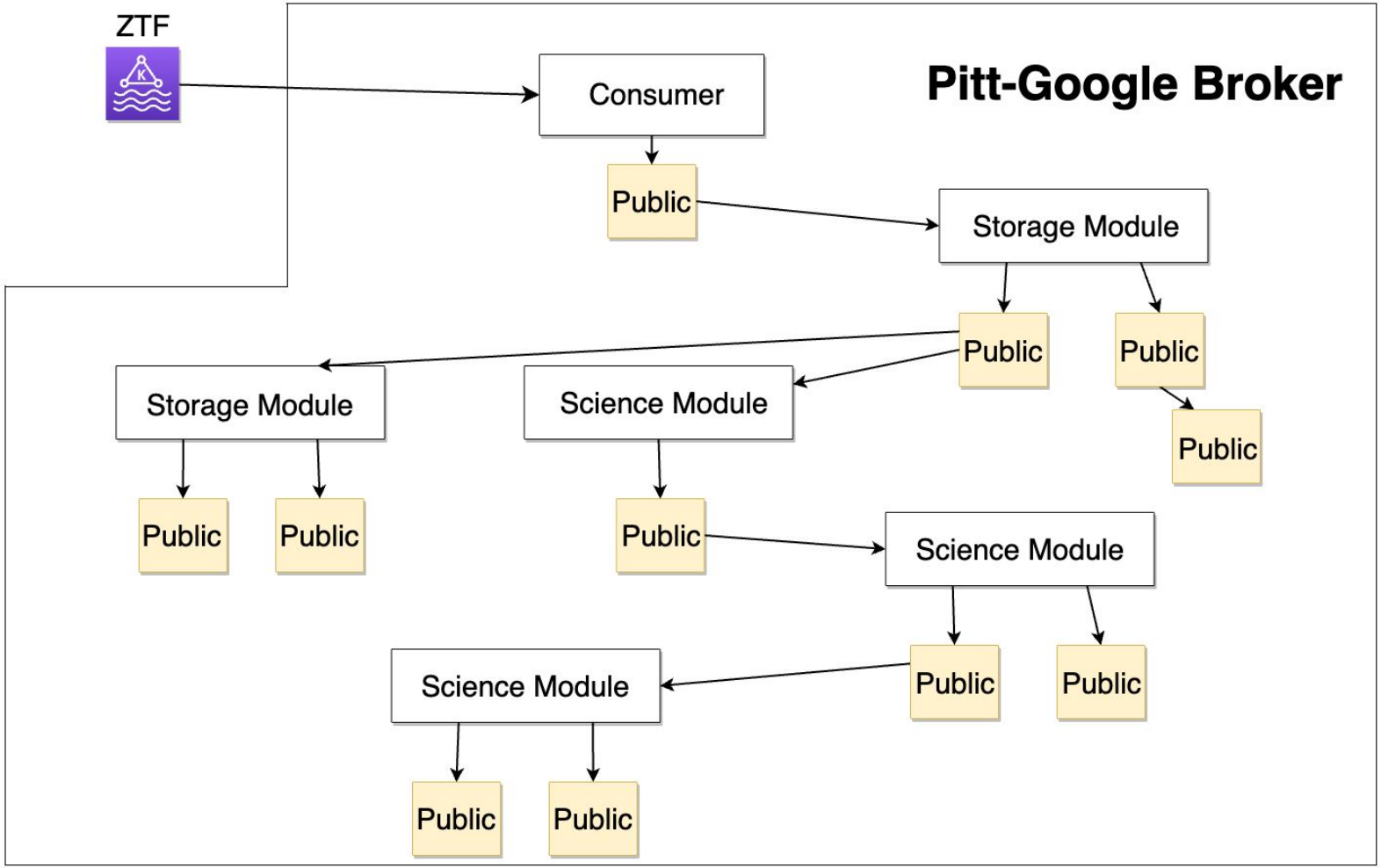
Running on Google Cloud Platform with **cloud-native architecture**:

- collection of loosely coupled modules (microservices)
- running on compute services (nearly serverless)
- communicating through public Pub/Sub message streams
- streaming inserts to BigQuery tables (SQL access) and Cloud Storage buckets

means:

- Process the live stream. Query the past instantly.
- Our architecture is open and you can connect in anywhere.
- It is easy for users to use any of Google Cloud's tools and services.


ZTF
Typical Broker
Consumer Cluster
Storage Cluster
(multiple modules)
Science Cluster
(multiple modules)
Public


ZTF
Consumer
Pitt-Google Broker
Public
Storage Module
Public
Public
Storage Module
Science Module
Public
Public
Public
Public
Science Module
Science Module
Public
Public
Public
Public

Broker Design

No clusters in Pitt-Google broker.

Consumer:

- Single VM (ZTF): 1 vCPU, 4 GB memory
- Kafka → Pub/Sub Connector (Apache Kafka plugin, written by Google Pub/Sub maintainers)
  - Passes alert bytes and metadata straight through to a Pub/Sub topic. That's it.

Pub/Sub:

- Google's streaming message service
- Topics and Subscriptions are separate data resources in the Cloud
  - They can live in different Google Cloud projects
- Subscriptions:
  - Pull from anywhere via API
  - **Push to any HTTP endpoint like Cloud Functions or Cloud Run.** (Kafka doesn't support push)
  - BigQuery subscriptions

Broker Design: Event-driven microservices model

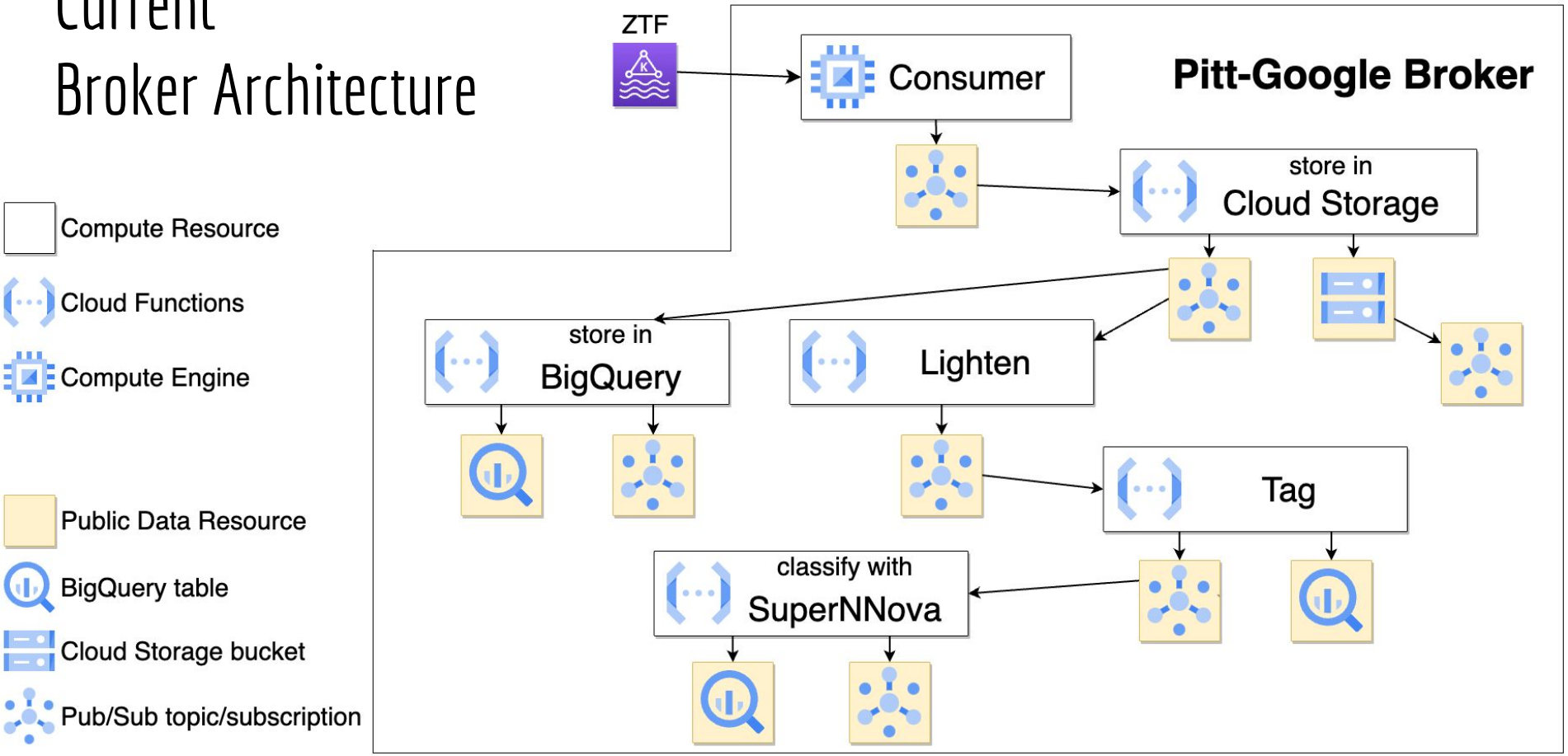
Each module (except Consumer):

- Runs independently on a serverless compute service
  - Cloud Functions or Cloud Run
- Per-alert processing. No batching.
- Streams data to Google's data services (Pub/Sub, BigQuery, Cloud Storage)

"Microservices is an architectural style that structures an application as a *collection of services* that are … *loosely coupled, independently deployable*, …" (https://microservices.io/)

(event = message = alert)

Current Broker Architecture



Event-driven microservices model

Pub/Sub + Cloud Functions or Cloud Run

Benefits:

- Lightweight. No cluster or VM management.
- Compute scales from zero, automatically and nearly instantaneously, to meet demand (alert rate).
- Per-alert processing is easy to think about and develop.
- Fault-tolerant
  - Failures can be re-delivered, dropped, or sent to a dead-letter topic.
  - No other message is affected
- Fast

Drawbacks: Per-alert processing is challenging in some cases...

Cost Structure

The idea behind Pitt-Google broker is to get the data into Google Cloud data services where it is widely available and easily accessible.

We can explore creating a public interface so that people can explore without authenticating to Google.

But in general, users will authenticate and access data directly through Google Cloud.
In Google Cloud, everyone pays for what they use.

Google Cloud pricing structure:

- Pay only for what you use
- Pay as you go
- Free tier
  - access a small but reasonable amount of each service for free
  - quotas renew monthly
- 90-day, $300 Free Trial
- Research credits available
- You cannot be charged unless you explicitly enable billing

Cost Structure Examples

Querying BigQuery

- Free 1 TB/month. After that, $6.25 per TB. (TB of data scanned by the query)

Pub/Sub topics and subscriptions

How it works:

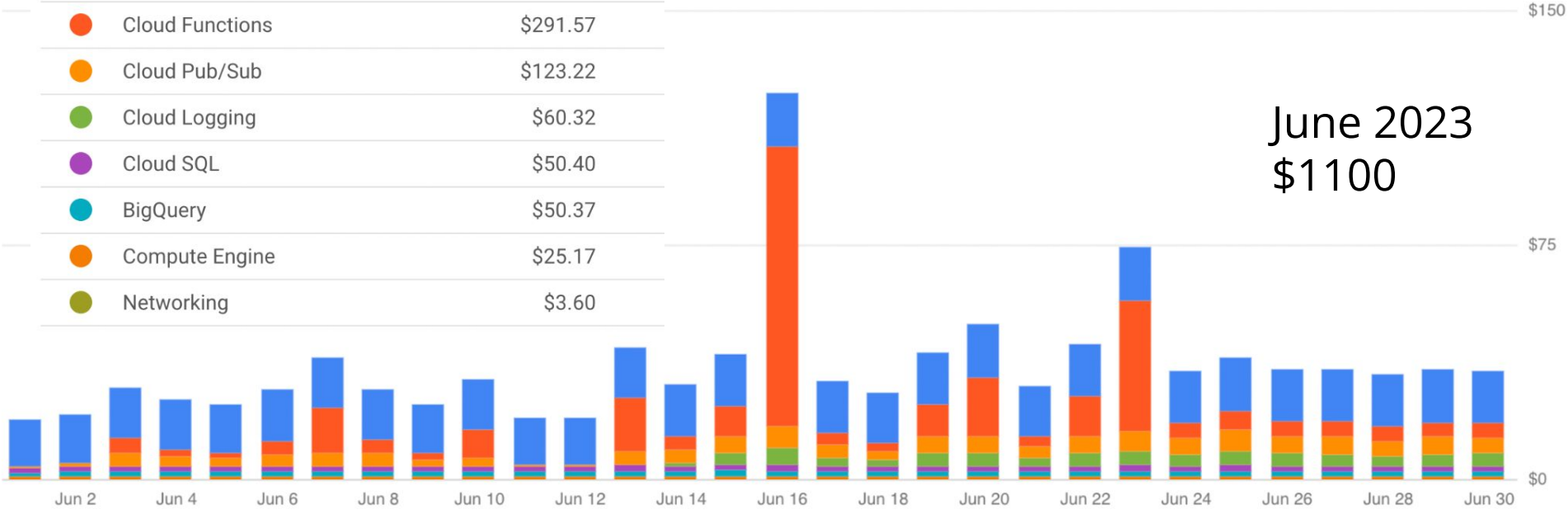
- Project that owns the topic pays to publish
- Project that owns the subscription pays to listen
- User must authenticate to the project owning the resource (topic or subscription)

Costs (on both ends):

- Throughput: Free 10 GB/month. After that, $40 per TB (~$10^7$ LSST alerts)
- Egress for throughput that crosses Google Cloud regions
- No storage fees (unless you turn on special retention options)
- https://cloud.google.com/pubsub/pricing

Pitt-Google Broker's Cloud Costs (ZTF)

Service	Cost
Cloud Storage	$492.08
Cloud Functions	$291.57
Cloud Pub/Sub	$123.22
Cloud Logging	$60.32
Cloud SQL	$50.40
BigQuery	$50.37
Compute Engine	$25.17
Networking	$3.60



June 2023
$1100

Current Status of the Pitt-Google Broker

Pitt-Google broker has been running the ZTF stream consistently since late 2020

Offers all of the **standard technical services**:
- live message streams, SQL access, object storage, APIs

with **more flexibility**:
- open architecture. hook in anywhere
- use any/all of Google Cloud's tools (console, APIs, ...) and services (Pub/Sub → Cloud Run, ...)
- process the live stream. query the past instantly

and a **different cost structure**:
- Open access through Google Cloud. Everyone pays for what they use.

Scientific services like classification is an obvious area we could improve.
- anyone can add to our system and plug in.
- we hope to provide standard classifiers across a range of science and support their continued running in the Pitt-Google Broker system.
  - Microlensing (MicroLIA), variable star classifier (UPSILoN-T), re-training SN classifiers on LSST-expected data (SuperNNova, ...)

Hooking into the Pipeline

We intend to demonstrate both access routes in DESC's ELAsTiCC challenge.

Data all available in real time

- Pub/Sub: alert data plus classifications/categorizations.
- BigQuery: alert data plus classifications/categorizations.
  - Standard SQL, built-in analytics engine. (and potentially via a TAP service with ADQL)
- Cloud Storage: alert packets (Avro)

Hooking in from the Cloud

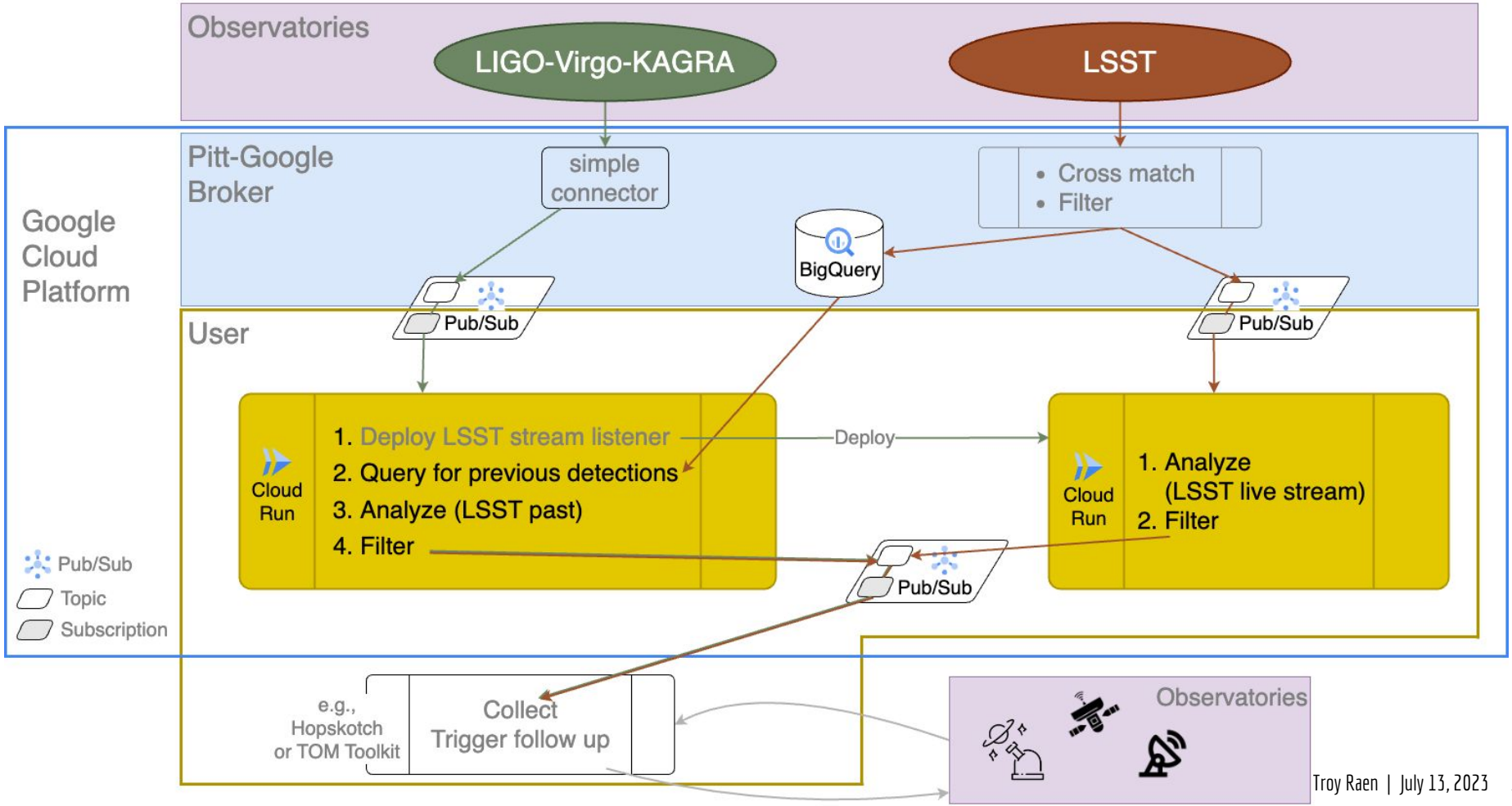
- Easy to connect Pub/Sub streams to serverless compute.
  - Write a function that processes a single alert. Deploy it to Cloud Functions or Run.
  - Store results in the Cloud and/or send to a local machine.

Hooking in locally

- web console
- command line, python, etc.
- pittgoogle-client (python). Wrappers around Google API to facilitate connection to our resources and simplify some tasks (e.g., Pub/Sub → local database).

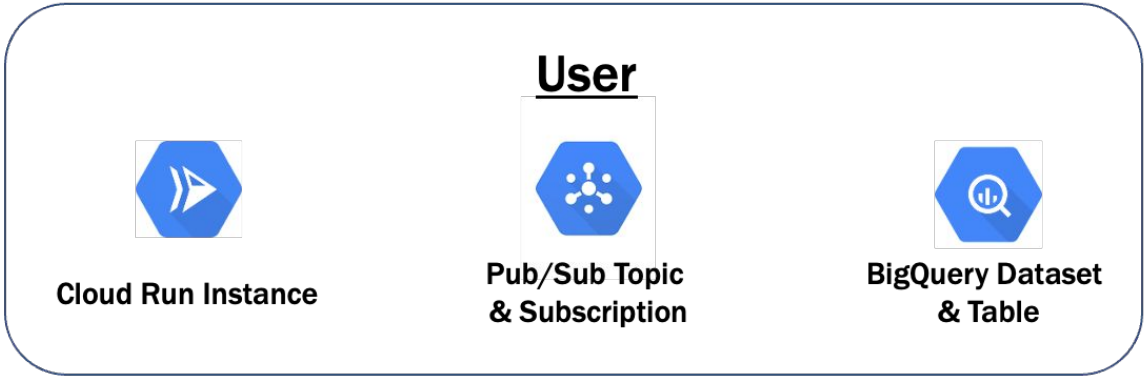
Lightweight way to explore the live streams. Doable by a single grad student.

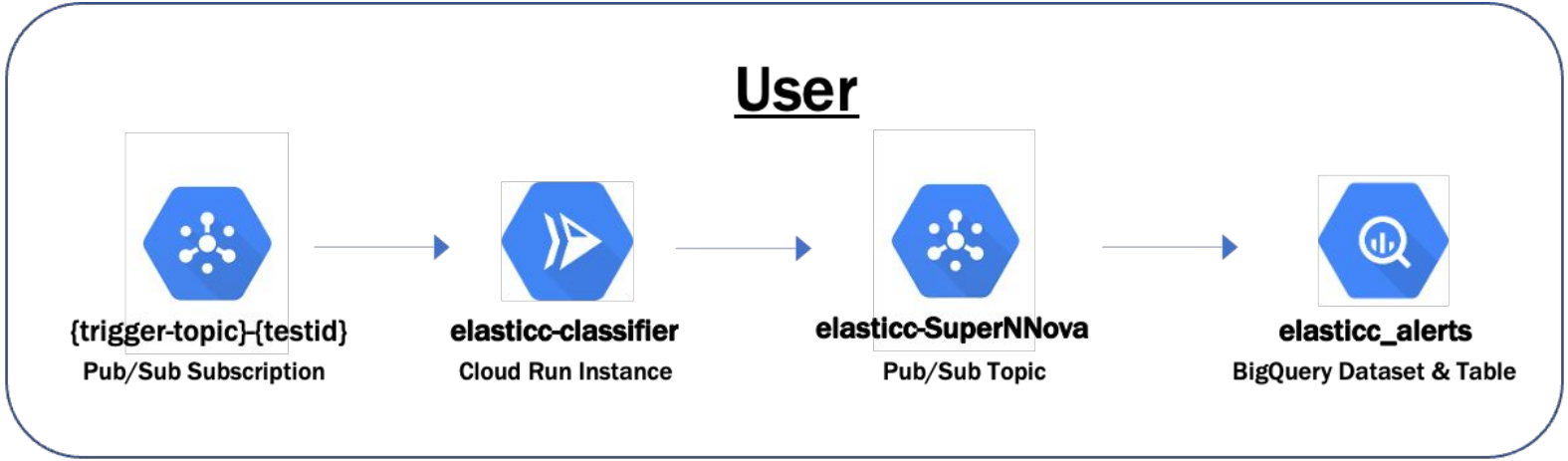
MMA follow up using Pitt-Google broker and Google Cloud Platform



Deployment of a Classifier Module

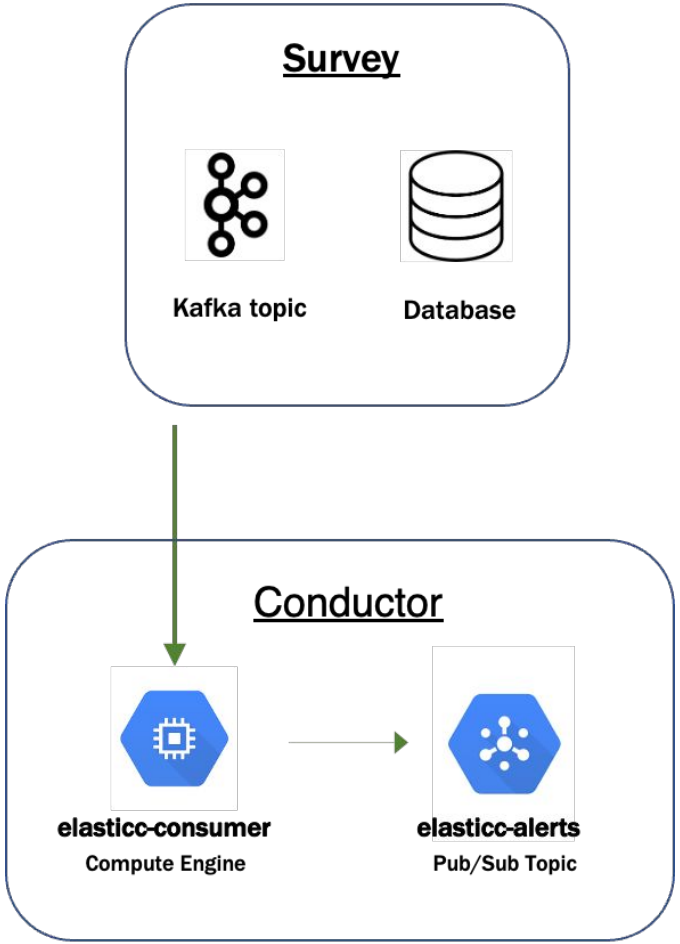
Deploying a supernova classifier using Google Cloud Products





Independent Google Cloud Project: elasticc-challenge

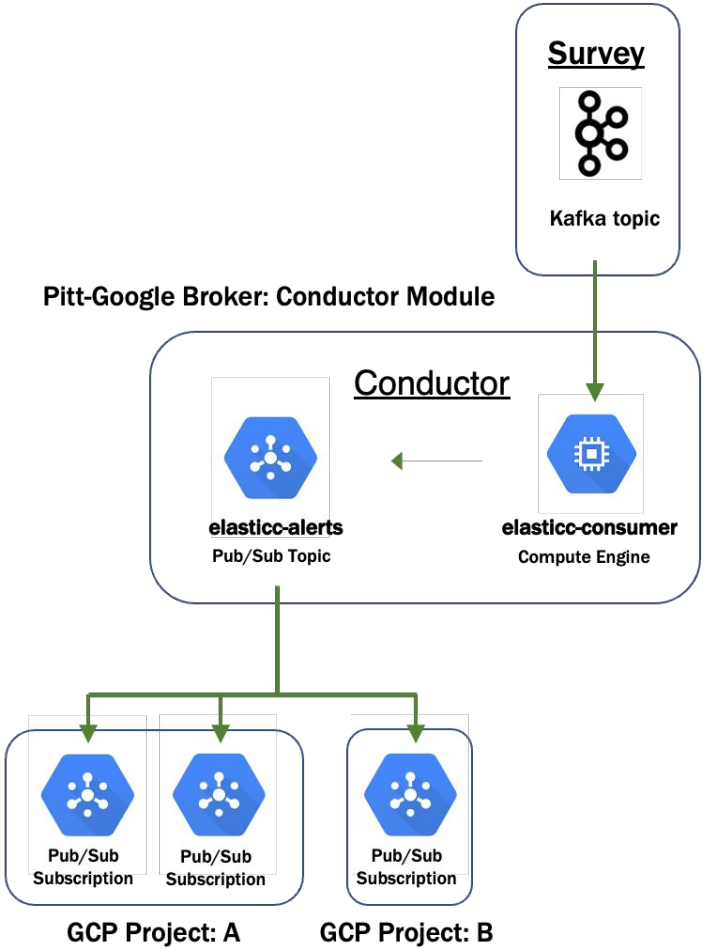
1. How do I receive alerts for processing?
2. How can I deploy my classifier (and trigger it)?
3. Where do I store my classifications?

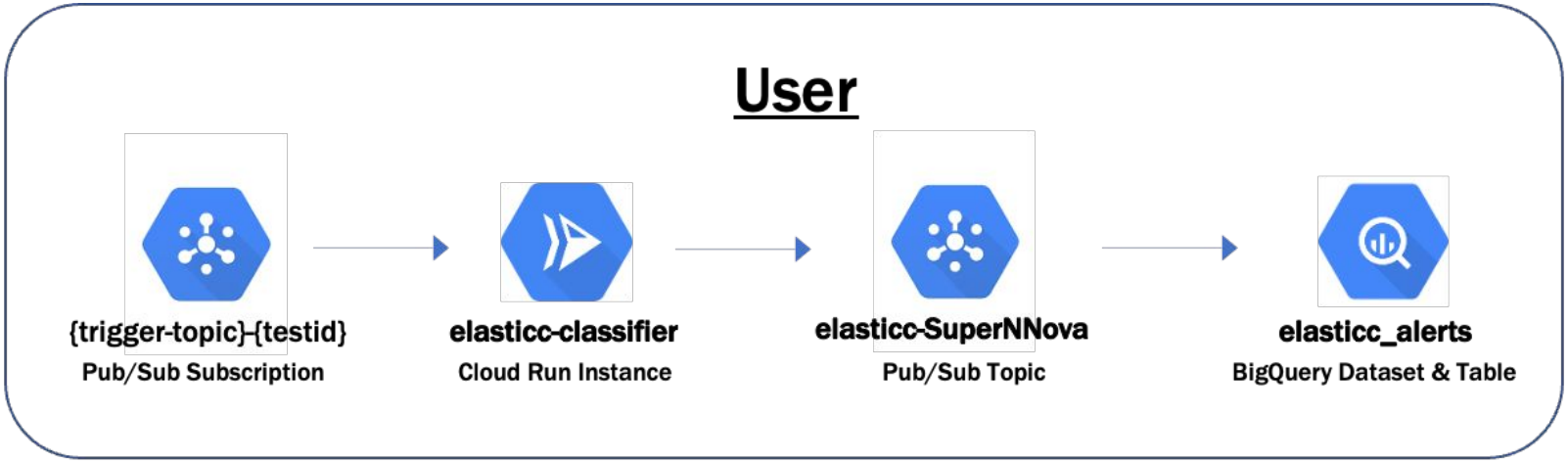


Pitt-Google Broker: Conductor Module

Pub/Sub Subscriptions

- Pub/Sub subscriptions subscribe to Pub/Sub topics in order to receive messages
- Pub/Sub offers at-least-once delivery
- A topic can have multiple subscriptions, but a given subscription belongs to a single topic
- Pub/Sub subscriptions are used to trigger Cloud Run instances

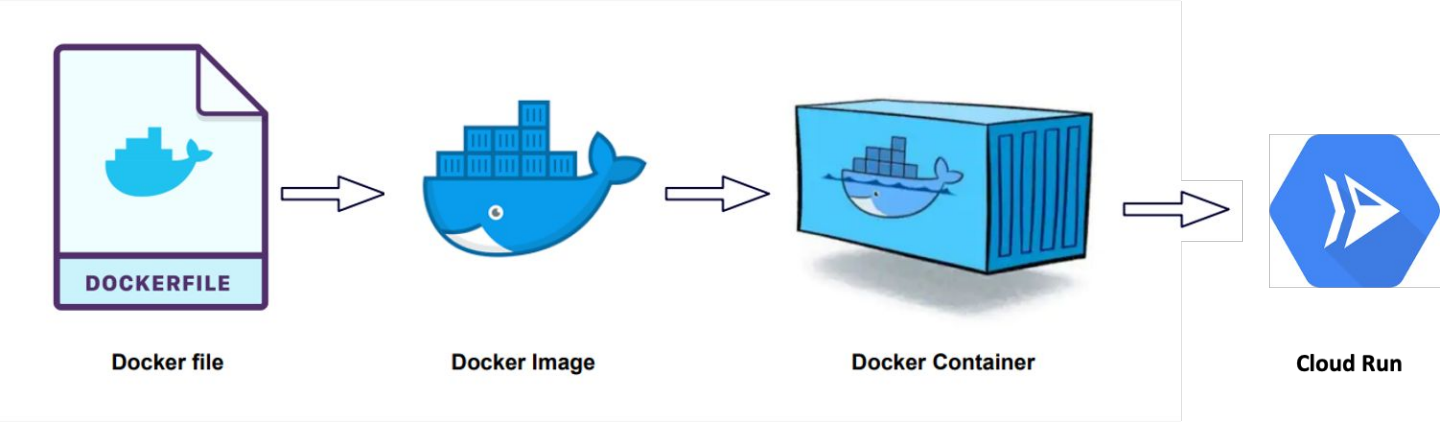




Independent Google Cloud Project: elasticc-challenge

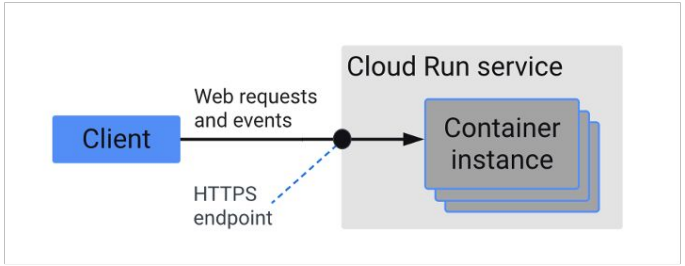
1. How do I receive alerts for processing?
2. How can I deploy my classifier (and trigger it)?
3. Where do I store my classifications?

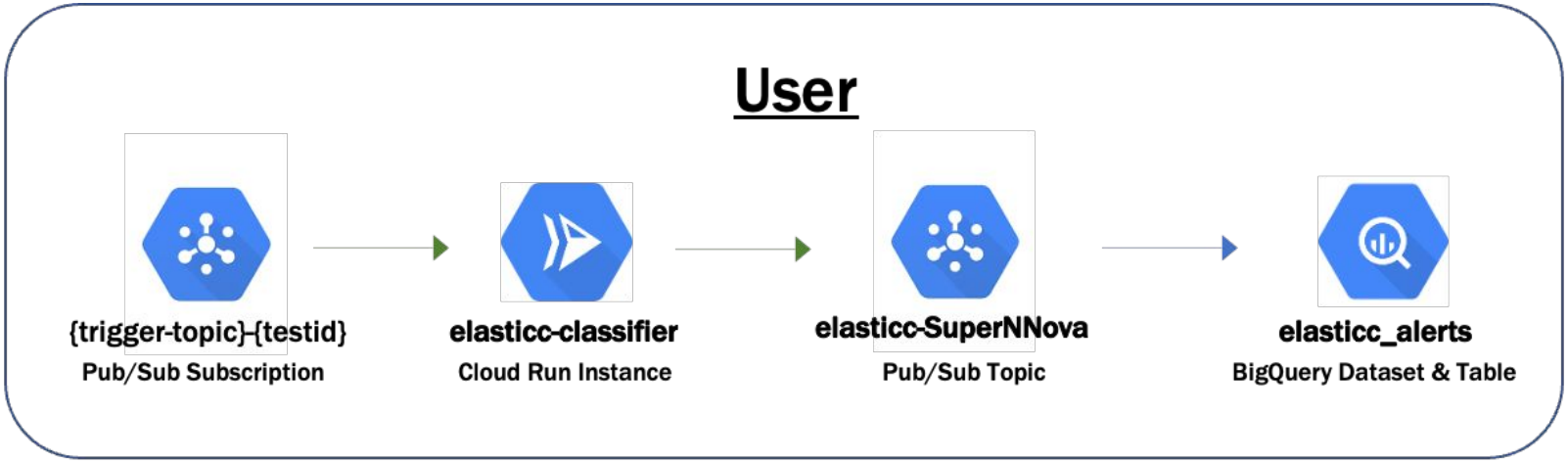
SuperNNova
open source photometric classification


DOCKERFILE
Docker file
Docker Image
Docker Container
Cloud Run

Cloud Run

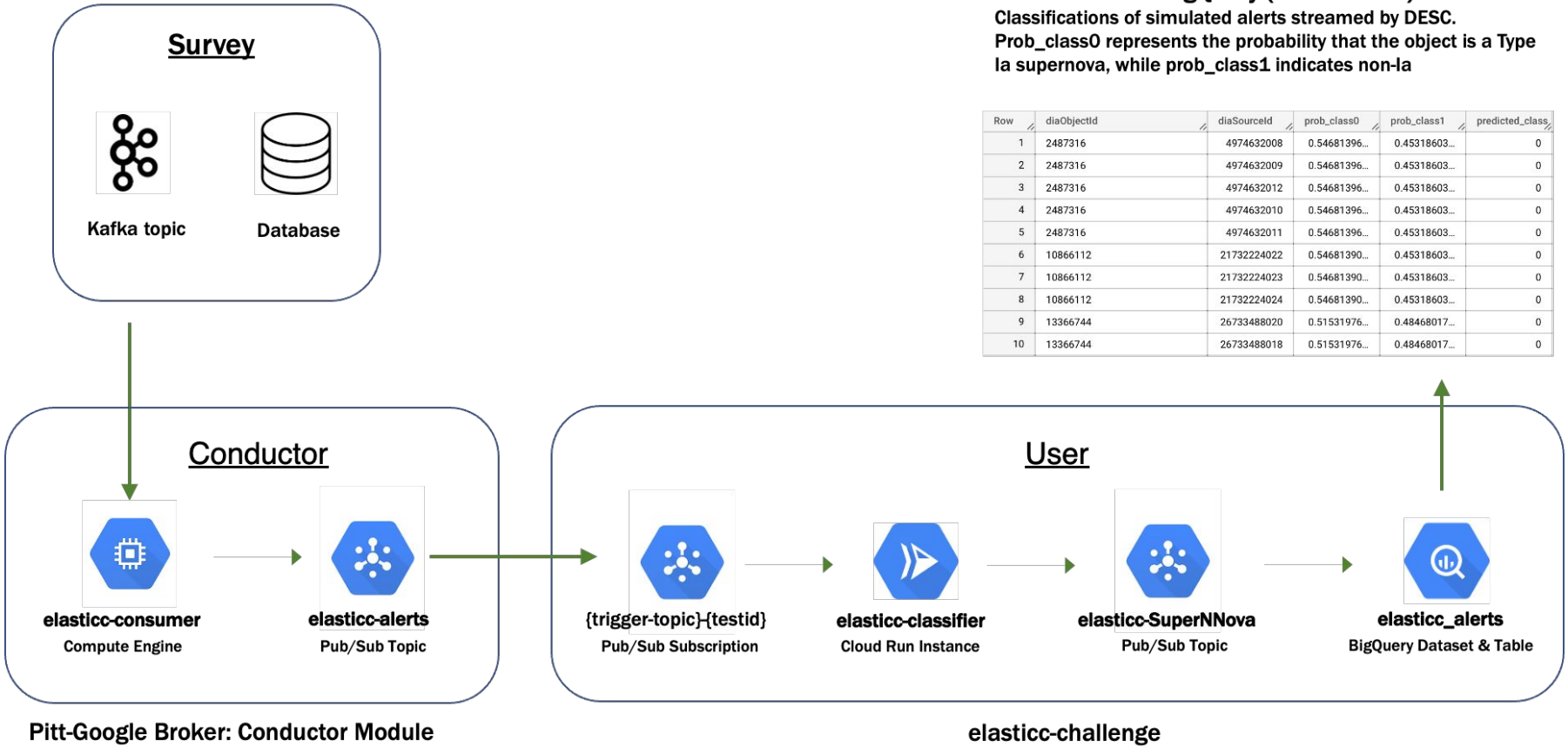
- Is a scalable, serverless product designed to deploy applications written in any programming language (if able to build a container image from it)
- Cloud Run services use scalable container instances to respond to web requests and events





Independent Google Cloud Project: elasticc-challenge

1. How do I receive alerts for processing?
2. How can I deploy my classifier (and trigger it)?
3. Where do I store my classifications?



Data written to BigQuery (visualization):

Classifications of simulated alerts streamed by DESC. Prob_class0 represents the probability that the object is a Type Ia supernova, while prob_class1 indicates non-Ia

Row	diaObjectId	diaSourceId	prob_class0	prob_class1	predicted_class
1	2487316	4974632008	0.54681396...	0.45318603...	0
2	2487316	4974632009	0.54681396...	0.45318603...	0
3	2487316	4974632012	0.54681396...	0.45318603...	0
4	2487316	4974632010	0.54681396...	0.45318603...	0
5	2487316	4974632011	0.54681396...	0.45318603...	0
6	10866112	21732224022	0.54681390...	0.45318603...	0
7	10866112	21732224023	0.54681390...	0.45318603...	0
8	10866112	21732224024	0.54681390...	0.45318603...	0
9	13366744	26733488020	0.51531976...	0.48468017...	0
10	13366744	26733488018	0.51531976...	0.48468017...	0