



Architecture of streaming analytics pipelines

Data Engineering on Google Cloud Platform



©Google Inc. or its affiliates. All rights reserved. Do not distribute.
May only be taught by Google Cloud Platform Authorized Trainers.

1 hour

Agenda

What is streaming?

Stream data processing: Challenges

Handling variable data volumes

Dealing with unordered/late data

Derive insights from data + Lab

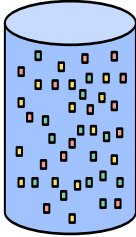
Notes:

This course discusses what stream processing is, how it fits into a big data architecture, when stream processing makes sense, and what Google Cloud technologies and products you can choose from to build a resilient streaming data processing solution.

Proprietary + Confidential

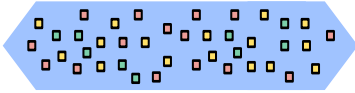
Streaming => data processing for unbounded data sets

Bounded



Finite data set
Is complete regardless of time
Typically at rest in a common durable store

Unbounded



Infinite data set
Is never complete, especially when considering time
Stored in multiple temporary, yet durable stores

Google Cloud Training and Certification 3

Notes:

Streaming is basically data processing on unbounded data. Bounded data is data at rest. Stream processing is how you deal with unbounded data.

Without streaming data....there'd be no stream processing. Streaming is an execution engine (system, service, runner) capable of processing unbounded data.

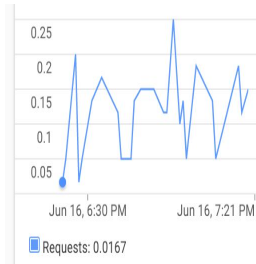
When designed correctly a streaming processing engine can provide: low latency, speculative or partial results, the ability to flexibly reason about time, controls for correctness, and the power to perform complex analysis.

Unbounded datasets are quite common

Proprietary + Confidential



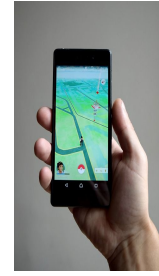
Traffic sensors
along highways



Usage information of
Cloud component by
every user with a GCP
project



Credit card
transactions



User moves in
multi-user
online gaming

Notes:

<https://pixabay.com/en/highway-night-traffic-light-motion-216090/> (cc0)

<https://pixabay.com/en/ledger-accounting-business-money-1428230/> (cc0)

<https://pixabay.com/en/pokemon-pokemon-go-mobile-trends-1581774/> (cc0)

In this course, we use the San Diego traffic data collected over a year. So this is data that is bounded. However, we are gonna simulate as if it were streaming real time.

Should We Replace the Term Big Data with “Unbounded Data”?

Clive Longbottom of Quocirca makes the case that “Big Data” is the wrong way to talk about the changes in the ways we store, manage and process data. The term certainly gets thrown around a lot, and in many cases for talking about managing data that is much smaller than the petabytes of data that arguably defines big data. Longbottom suggests the term “unbounded data”

<http://readwrite.com/2011/03/25/big-data-round-up-unbounded-da/>

The need for fast decisions leads to streaming



Massive data, from varied sources, that keeps growing over time



Need to derive insights immediately in the form of dashboards

HOW MANY SALES DID I MAKE IN THE LAST HOUR DUE TO ADVERTISING CONVERSION?

WHICH VERSION OF MY WEB PAGE DO PEOPLE LIKE BETTER?

WHICH TRANSACTIONS LOOK FRAUDULENT?

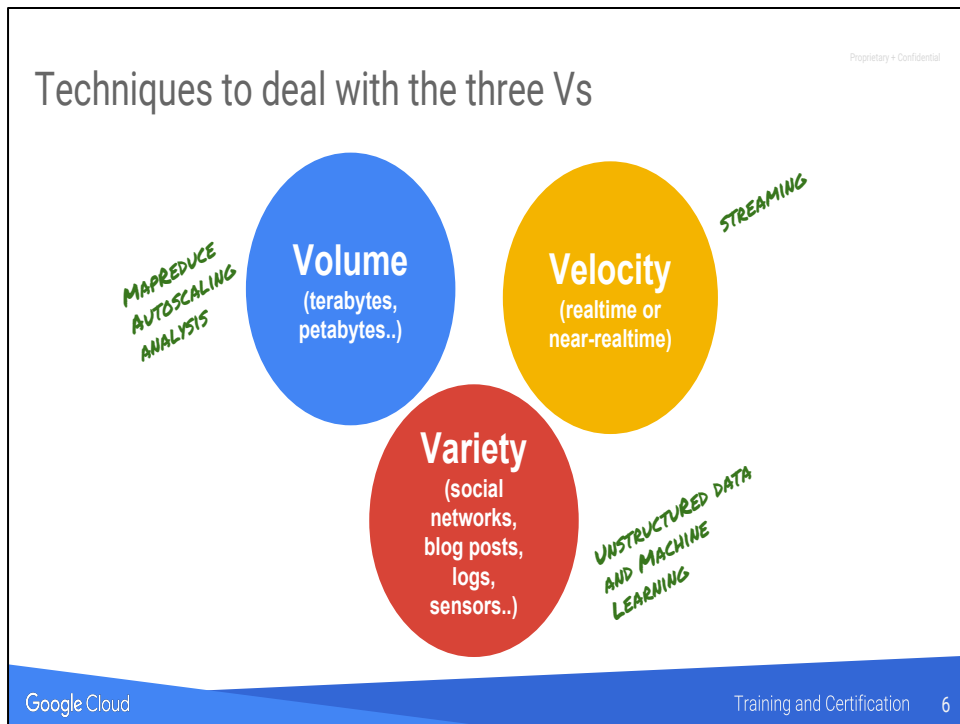
Need to make timely decisions

Notes:

- The demand for stream processing is increasing a lot these days.....processing big volumes of data is not enough
- Data has to be processed fast, so that a firm can react to changing business conditions in real time.
- Real world stream processing use cases include trading, fraud detection, system monitoring, smart order routing, transaction cost analysis....the list is long.

We talk about streaming unbounded data, we wanna derive insights. Which version of my web page ppl like...say ur running AB test..that's streaming.

Batch vs Stream: If you wanna look for fraud transactions 3 days ago....that's batch. If u wanna look at it as it happens...that's streaming.



Notes:

We have terabytes and petabytes of data...and how we deal with it....we talked about on day 1....mapreduce. We talked about batch processing over volumes of data. Variety is all sorts....audio video etc....and we talked about ML APIs. The third part is real real time.

A big data architecture contains several parts. Often, masses of structured and semi-structured historical data are stored in Cloud storage/PubSub/BigQuery (Volume + Variety). On the other side, stream processing is used for fast data requirements (Velocity + Variety). Both complement each other very well. Internet of Things has increased volume, variety and velocity of data, leading to a dramatic increase in the applications for stream processing technologies.

MApReduce -> volume
Velocity -> streaming
Variety - > ML

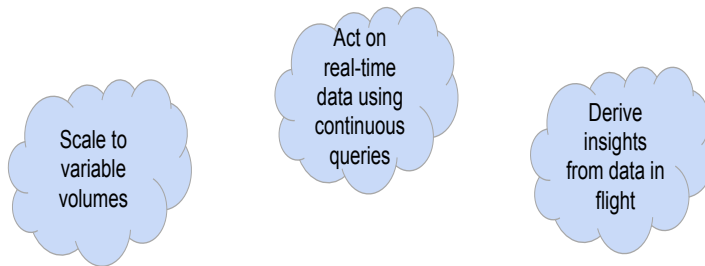
Agenda

Stream data processing: Challenges

Notes:

Stream processing makes it possible to derive real-time insights from growing data

So, a stream processing solution needs to be able to:



Notes:

Has to be able to scale...think of credit cards, there will be hundreds of transactions happening all the time. Continuous query....do the query on last hour of data.

Stream processing systems were born nearly a decade ago, due to the need for low-latency processing of large volumes of highly dynamic, time-sensitive continuous streams of data from sensor-based monitoring devices and alike.

Stream processing found its first uses in the finance industry, as stock exchanges moved from floor-based trading to electronic trading. Today, it makes sense in almost every industry - anywhere where you generate stream data through human activities, machine data or sensors data.

Traditional "Store-and-Pull" model of traditional data processing systems was simply not suitable for the high performance needs of the streaming applications, where data was much more dynamic than queries and had to be processed on the fly. Consequently, this change of roles between data and queries led to an upside down architecture, and thus, to Stream Processing Engines

Scale to variable volumes, be highly available and fault tolerant

Act on real-time data using “continuous queries”

(i.e., SQL-type queries that operate over time and buffer windows)

Derive Insights from data in flight

(continuously calculate mathematical or statistical analytics on the fly within the stream while accounting for late, out-of-order data)

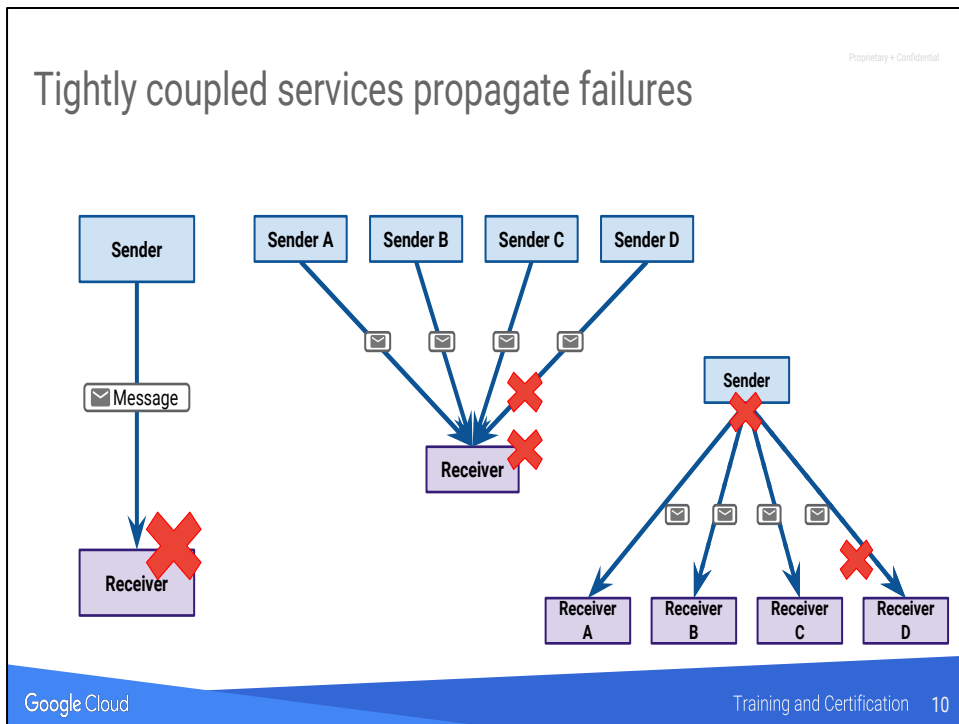
Challenge #1: Variable volumes require ability of ingest to scale and be fault-tolerant

Ingesting variable volumes

massive amounts of streaming events, handle spiky/bursty data, high availability and durability

Notes:

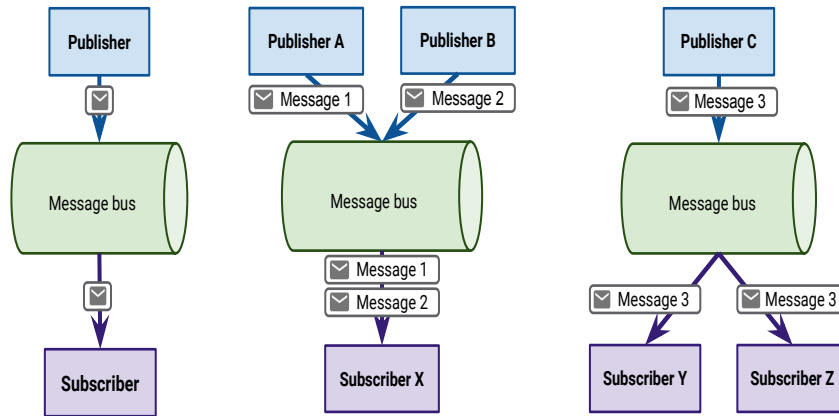
Scaling implies being able to deal with faults as clients/servers fail unexpectedly.



Notes:

One way is to send directly.....may not be able to handle volume.....single receiver is problem when receiver is down. Tight coupling means errors will propagate. Starts with one....then whole ecosystem gets buggy.

Loosely-coupled systems scale better



Notes:

Solution to this is buffering. Not send it directly...but sending it to a message bus...that can handle volumes. All scenarios fan-in, fan-out will work.

Challenge #2: Latency is to be expected

Ingesting variable volumes

massive amounts of streaming events, handle spiky/bursty data, high availability and durability



Cloud Pub/Sub
Ingest

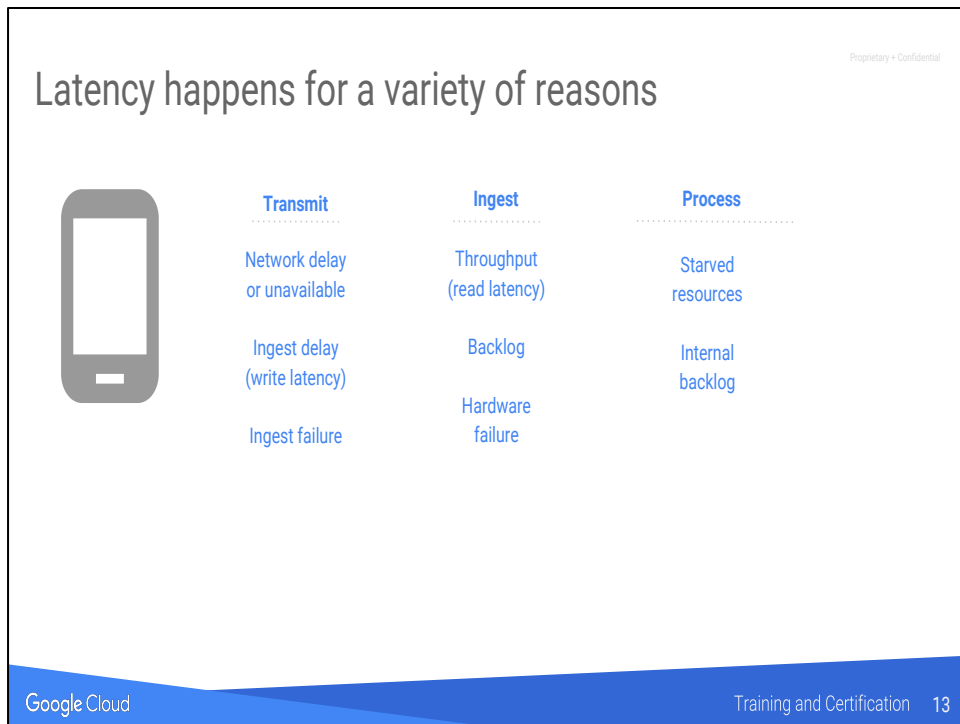
Late data, unordered data

How to deal with latency?
Late-arriving records? Speculative results?

Notes:

Streaming data can be late and unordered: Let's detail these challenges and how Cloud Dataflow can help address them

There is always danger with latency. If 2 windmills are sending in data...one may come before another. Or even out of order. You compute result.....then data that was supposed to be in result comes in.



Notes:

You have to operate assuming latency will happen.

2 school of thoughts: .one says let's harden our system so no error happens

Another says lets design so that errors will happen but we will be resilient.
Latency will happen...fact of life.

When you design processing pipeline....assume it may have unordered....latency and come up with ways to address it.

Beam/Dataflow model provides exactly once processing of events

- **What results are calculated?** Answered via transformations.
- **Where in event time are results calculated?** Answered via event-time windowing.
- **When in processing time are results materialized?** Answered via watermarks, triggers, and allowed lateness.
- **How do refinements of results relate?** Answered via accumulation modes

<https://cloud.google.com/blog/big-data/2017/05/after-lambda-exactly-once-processing-in-google-cloud-dataflow-part-1>

The Dataflow model lets you write clean, modular code that evolves beautifully over time as needs change and expand. The model maps directly onto the four questions that are relevant in any out-of-order data processing pipeline:

- **What results are calculated?** Answered via transformations.
- **Where in event time are results calculated?** Answered via event-time windowing.
- **When in processing time are results materialized?** Answered via watermarks, triggers, and allowed lateness.
- **How do refinements of results relate?** Answered via accumulation modes

In addition, the powerful out-of-order processing constructs in our model, such as watermarks and triggers, allow you to maintain eventual correctness of results within a single pipeline, while also offering low-latency speculation, the ability to refine results after the fact when upstream data change, and an easy way to cap the useful lifetime of data within your system.

Challenge #3: Need instant insights

Ingesting variable volumes

massive amounts of streaming events, handle spiky/bursty data, high availability and durability



Cloud Pub/Sub
Ingest

Late data, unordered data

How to deal with latency?

Late arriving records? Speculative results.

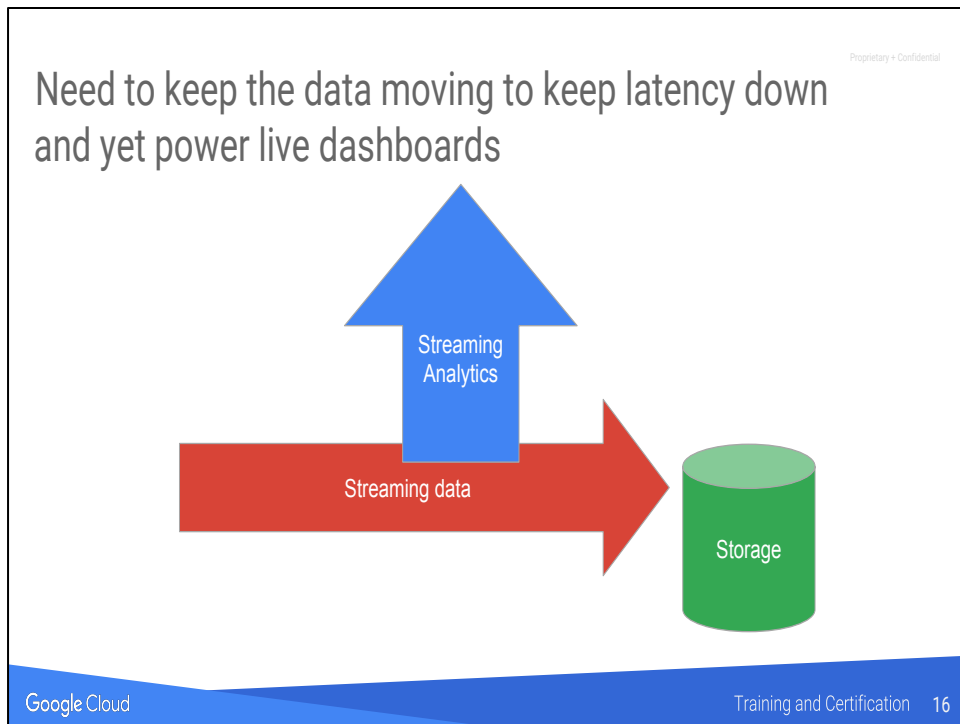


Cloud Dataflow
Processing and
Imperative Analysis

Real-time insights

Continuous query processing, Visualization, Analytics, etc.

Notes:



Notes:

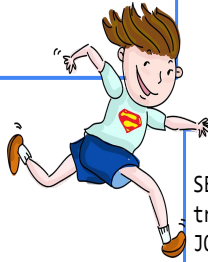
To achieve low latency, a system must be able to perform message processing without having a costly storage operation in the critical processing path. A storage operation adds a great deal of unnecessary latency to the process (e.g., committing a database record requires a disk write of a log record). For many stream processing applications, it is neither acceptable nor necessary to require such a time-intensive operation before message processing can occur. Instead, messages should be processed “in-stream” as they fly by

Source: <http://cs.brown.edu/~ugur/8rulesSigRec.pdf>

BigQuery lets you ingest streaming data and run queries as the data arrives

Dataflow streaming into BigQuery

```
.apply(BigQueryIO.writeTableRows().to(trafficTable))  
.....  
....
```



```
SELECT avg_speed FROM  
traffic_conditions  
JOIN ( ..... ) ON ....
```

Notes:

We looked at bigquery as a batch system....here we look at its streaming capability

Provided by the APIs `tabledata().insertAll()` method

Works for partitioned and standard tables

Image icon: <https://pixabay.com/en/man-running-exercise-jogging-2034538/>
(cc0)

Proprietary + Confidential

Need to use unified language when querying historic and streaming data for seamless integration

The diagram illustrates the need for a unified language for querying historic and streaming data for fraud detection. It features two input sources on the left: a ledger representing 'historical data' and a transaction between two computer monitors representing 'live data'. Two orange arrows point from these sources to a central figure of a police officer. The top arrow is labeled 'historical data' and the bottom arrow is labeled 'live data'. Above the police officer is a red starburst shape containing the text 'Fraud Detection'. The police officer is wearing a blue uniform and has handcuffs on his wrists.

Google Cloud

Training and Certification 19

Notes:

A system that can deal with reference data and live streaming data if you want real time insight

For example, in on-line data mining applications (such as detecting credit card or other transactional fraud), identifying whether an activity is “unusual” requires, by definition, gathering the usual activity patterns over time, summarizing them as a “signature”, and comparing them to the present activity in real time. To realize this task, both historical and live data need to be integrated within the same application for comparison.

For low-latency streaming data applications, interfacing with a client-server database connection to efficiently store and access persistent state will add excessive latency and overhead to the application. Therefore, state must be stored in the same operating system address space as the application using an embedded database system.

[https://pixabay.com/en/ecommerce-selling-online-2140603/\(cc0\)](https://pixabay.com/en/ecommerce-selling-online-2140603/(cc0))
[https://pixabay.com/en/comic-characters-crisis-disaster-2024758/\(cc0\)](https://pixabay.com/en/comic-characters-crisis-disaster-2024758/(cc0))
[https://pixabay.com/en/ledger-accounting-business-money-1428230/\(cc0\)](https://pixabay.com/en/ledger-accounting-business-money-1428230/(cc0))

Stream processing on GCP

Proprietary + Confidential

Ingesting variable volumes

massive amounts of streaming events, handle spiky/bursty data, high availability and durability



Cloud Pub/Sub
Ingest

Late data, unordered data

How to deal with latency?
Windows, watermarks



Cloud Dataflow
Processing and
Imperative Analysis

Real-time insights

Continuous query processing,
Visualization, Analytics, etc.



Google BigQuery
Durable storage and
Interactive Analysis

Notes:

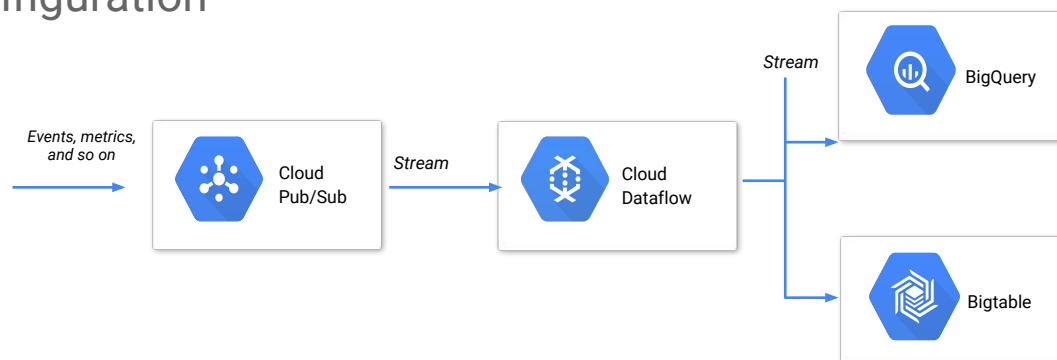
PubSub is your global message bus.

Dataflow is capable of doing batch and streamingcode does not change.

BigQuery gives you power of analytics.

BigTable when you get overwhelming volume.

Stream data processing on GCP: A common configuration



Fully managed, No-ops services

Notes:

PubSub is your global message bus

Dataflow is capable of doing batch and streamingcode does not change

BigQuery gives you power of analytics

BigTable when you get overwhelming volume

Bigquery is order of seconds, sql,

While bigtable is order of ms, nosql

Data sources – source of data streams. This is the raw data and so the source of truth. A data source could be a sensor network, or a mobile application, or a web client, or a log from a server, or even a thing from Internet of Things.

Message bus – reliable, high-throughput and low latency messaging system.

Stream processing system – a computational framework capable of doing computations on data streams. There are a few stream processing frameworks out there.

Analytic Environment – processed data is of no use if it does not serve end applications or users with real-time insights

End applications – Dashboards, Reports, Decision Makers

Lab:

Discuss some streaming scenarios

Scenario	What data is sent	What processing does this data need	What analytics/queries do you run	What dashboard/reporting would be most useful
San Diego Department of Transport wants to bring live updates on highway lanes to commuters on the highway	San Diego highway traffic data collected over many miles of highway for all lanes, and transmitted at 5-min intervals	<ul style="list-style-type: none"> Current traffic conditions to power re-routing decisions for example Average speed and anomaly detection to understand which lanes are slower 	<ul style="list-style-type: none"> Results with min, max and averages Comparing averages between lanes to detect anomaly 	Near real-time update on lane slowdowns Map with markers showing traffic congestion
PR department of airline wants to handle negative tweets				
Real time Credit card fraud detection feature that will inform users of suspicious transactions as they happen over the counter				

Notes:

Talk through the first scenario since labs are based on it. Then have them jot down answers individually. Then discuss as a group. This looks trivial, but turns out to be quite hard, especially if you are not used to thinking about the full journey, from scenario to data pipeline to dashboards. Suggest ~10 minutes to fill out the table (2 minutes per row).

PR example:

Tweets that mention airline name or hashtag

Sentiment analysis, entity detection for topics ("delay", "weather", "baggage", "flight attendant", etc.) and location

dashboard top 10 negative topics segregated different ways.

Fraud example: data-transactions from every terminal in the world, processing-segregate by user, country, store, who to notify, queries-are we seeing increase in fraudulent activity in London, Dashboards - today's query is tomorrow's dashboard

Scenario	What data is sent	What processing does this data need	What analytics/queries do you run	What dashboard/reporting would be most useful
Smart vending machines that keep track of inventory and send refill requests proactively				
Real-time user targeting based on segment and preferences				
Real-time charging and billing based on customer usage, ability to populate up-to-date usage dashboards for users to be able to manage resource allocation				

Vending machine:

Data: purchases processing: time between successive purchases, analytics: estimate how long inventory will last dashboard:



cloud.google.com