Google Cloud Platform Data Pipeline Engineering

Develop a passion for learning.

Overview

Big Data Pipelines are just small data pipelines with massive amounts of data/information/knowledge/insight being processed. Big data pipeline engineering can become way more complex. There are lots of sources, lots of sinks (places where we put our data), lots of complex transformations, and lots of data. Just imagine having dozens of operational databases, a clickstream from your site coming through Kafka, hundreds of reports, OLAP cubes, and A/B experiments. Imagine, as well, having to store all the data in several ways, starting with raw data and ending with a layer of aggregated, cleaned, verified data suitable for building reports.

© 2021 by Innovation In Software Corporation

LOGISTICS

Class Hours:

- Start time is 9:00am PST
- End time is X:XXpm
- Class times may vary slightly for specific classes
- Breaks mid-morning and afternoon (15 minutes)

Lunch:

- Lunch is
- Yes, 1 hour and 15 minutes
- Extra time for email, phone calls, or simply a walk.

Telecommunication:

- Turn off or set electronic devices to vibrate
- Reading or attending to devices can be distracting to other students
- Try to delay until breaks or after class

Miscellaneous

- Courseware
- Bathroom
- Fire drills

© 2021 by Innovation In Software Corporation

DAY 1

Agility

Cloud SRE
- GCP Observabillity
- Stackdriver
  - Debugging
  - Error Reporting
  - Trace
  - Monitoring
- Visualization

Resiliency in Cloud
- RTO/RPO
- SLA/SLI/SLO
- Auto-scaling
- High-availability
- Disaster recovery
- Fault tolerance
- Technical account management

Dev/CI/CD transformation
- Containerization of service-oriented architecture
- Microservices architecture design principles
- Decentralized processing

- Business domain-Centric design
- Crossing process boundary is expensive!
- Designing for failureMicroservices and their relationship to the front-end
- Micro-frontends
- Service mesh
- API management/gateways
- GraphQL and data-driven microservices
- Building blocks of a fault-tolerant application
- Sample architecture for microservices
- Managing distributed services with Docker & Kubernetes

Data Engineering Foundation
- Containerization Explore the role of a data engineer.
- Analyze data engineering challenges.
- Intro to Data Pipelines
- Data Lakes
- Lakehouse Architecture
- Partner effectively with other data teams.
- Manage data access and governance.
- Build production-ready pipelines.

© 2021 by Innovation In Software Corporation

DAY 2

Velocity

Building Data Pipelines
- EI, ELT, ETL.
- Cloud Dataflow.
- Dataflow value proposition.
- Dataflow Pipelines.
- A Simple Dataflow Pipeline.
- MapReduce in Dataflow.
- Side Inputs
- Dataflow Templates.
- Dataflow SQL.
- Custom transforms.
- Composite transforms. Managing Data Pipelines
- Pipeline Components.
- Visualization.
- Orchestrating pipeline services.
- Data exploration.
- Workflow scheduling.
- Observability.

Cloud Composer
- Apache Airflow.
- Directed Acyclic Graphs.
- Workflow Management.
- DAGs and operators.
- Configuration.
- Use Cases.

Data Pipelines with Beam
- Planning the structure.
- Choosing the transforms.
- I/O methods.
- Pipeline driver.
- Pipeline definition.
- Pipeline execution.
- Runner-specific configurations.
- Testing and debugging.
- Use Cases.

Serverless messaging in pipelines
- Realtime analytics.
- Stream processing architecture.
- Reliable messaging systems.
- Publishers.
- Topics.
- Subscriptions.
- Message delivery.
- Snapshots.
- Message replay.

© 2021 by Innovation In Software Corporation

DAY 3

Observability

Integrating with Cloud Pub/Sub
- Topics
- Publishers.
- Subscriptions.
- Push/Pull.
- Pipeline Processing.

Google Cloud Functions
- Types of Functions
- Runtimes Support
- Triggers and Objects
- Setting up Google Cloud Platform Project
- Setting up local developer environment
- Example Google Cloud Function
- Use Cases.

Event-driven Processing
- Cloud Functions.
- Triggers.
- Background Functions
- Architecture.
- Observability.

Managing APIs
- Google Cloud Endpoints.
- Configuration.
- Security.
- Deployment.

Vertex AI API
- Explainable AI.
- Vizier.
- AI Platform SDK
- Feature Store.
- MLOps.
- Use Cases.

Vertex In Action
- Dataset Creation.
- Model Training.
- Versioning the Model.
- Evaluation.
- Deployment

© 2021 by Innovation In Software Corporation

DAY 4

Reliability

Cloud Functions and Stackdriver (Ops Suite)
- Monitoring Cloud Functions execution
- Logging
- Error Reporting

Cloud monitoring
- Logging and diagnostics.
- Dataflow integration with monitoring.
- Dataflow job metrics.
- Job Status.
- Element Count.
- System Lag (for streaming jobs).
- User Counters.
- Monitoring dashboards.
- Monitoring alerting capabilities.

AI, ML and Advanced Analytics on GCP
- What is AI?
- Ad-hoc data analysis.
- Data-driven decisions.
- ML models in GCP.
- Present and future.

ML, NLP and Unstructured Data
- Unstructured Data Processing.
- JSON
- ML APIs for Enriching Data.
- Natural Language API
- Classify Unstructured Text.

ML Pipelines with Kubeflow
- Ways to ML on GCP.
- Kubeflow.
- Kubeflow Pipelines
- Kubeflow Notebooks
- Pipelines Dashboard
- AI Hub.
- Running AI Models.

Serverless, autoscaling data pipelines
- Ingestion.
- Autoscaling.
- Processing & Storage
- Data Pipelines.
- Streaming Data.
- Reference Architecture.

© 2021 by Innovation In Software Corporation

DAY 5

Scalability

TensorFlow on GCP
- Getting Started.
- Concepts.a
- Types.
- ML datasetsi
- Use Cases.

ML Models withTensorFlow
- Graphs.
- Loops.
- tf.learn.
- Training the Model.
- Model Observability.
- Charting.

Cloud ML
- Model packaging.
- MLOps.
- Feature Engineering.
- Transformation

GCP Streaming
- Architecture.
- Watermarks.
- Triggers.
- Accumulation.

Building a Lakehouse
- Lakehouse Architecture.
- Data Storage and ETL options on GCP.
- Building a Data Lake using Cloud Storage.
- Cloud Functions in Lake.
- Securing Cloud Storage.
- Storing All Sorts of Data Types.
- Cloud SQL as a relational Data Lake.

Building a Cloud Data Warehouse
- Modernizing data warehouse.
- Getting Started.
- Loading Data.
- Exploring Schemas.
- Schema Design.
- Nested and Repeated Fields.
- Optimizing with Partitioning and Clustering.
- Transforming Batch and Streaming Data.

© 2021 by Innovation In Software Corporation

Meet The Instructor

George Niece AKA geo

Data Engineering Pipelines, Digital Transformation, DevSecOps, IoT Consultant with a Software Engineering, Digital Commerce, and IT operations background. Focused on cloud-native application modernization, datacenter divestiture, and cloud adoption.

Twitter
@georgeniece

LinkedIn
Linkedin.com/in/GeorgeNiece

mail
George.Niece@DigitalTransformationStrategies.net

Expertise

- Cloud-native
- XaaS
- AppDev
- IoT
- Automation
- CI/CD
- Microservices
- Agile

GTKY

Please let me know your

- Name
- Where you work from?
- Experience with
  - GCP
  - Data Engineering Pipelines
  - Serverless
  - DW/Lakehouse/Lake Formation

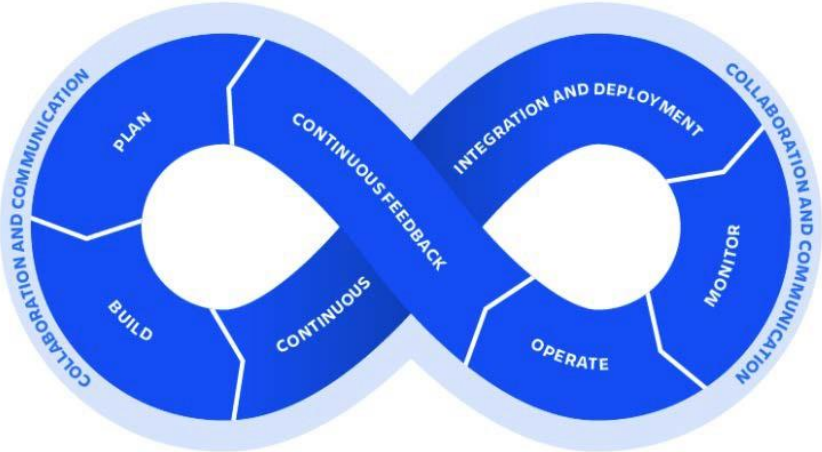
Please rate yourself (1-5) 1 = new joiner, 5 = SME

Lastly, what is your favorite dessert?

Cloud SRE

Pillars of SRE and DevSecOps

- Plan and build
- Continuous integration
- Continuous delivery
- Continuous deployment
- Continuous monitoring and operations
- Continuous feedback



Pipeline Plan and Build

In the **planning** phase, the core focus is to **understand the vision and convert** that vision into phases, otherwise known as **epics**

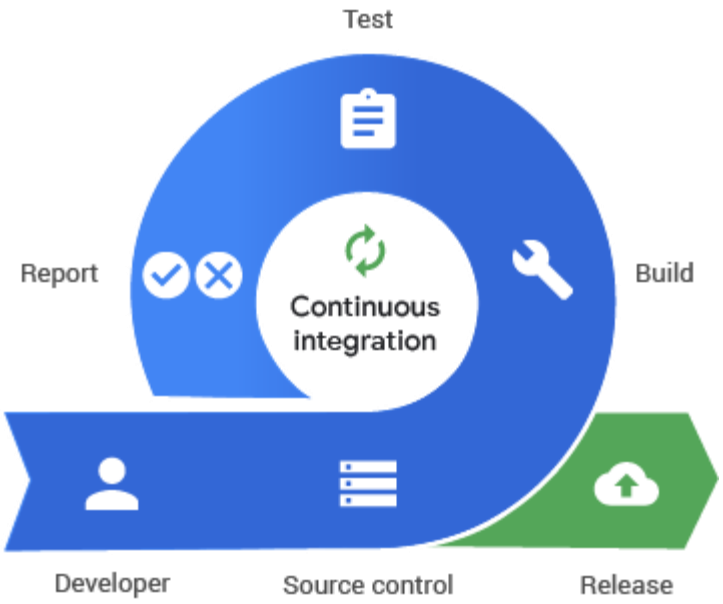
Each epic can be scoped to achieve a specific set of functionalities, which could be further **groomed** as one or **multiple user stories**.

In the **build** phase, the pipeline stages are written/built in the language(s) or tools of choice and appropriate build artifacts are created. **Code** is maintained in our **DVCS**

Continuous Integration (CI)

CI is a software development practice where developers **frequently integrate** their code changes to the main branch of a shared repository. This is done, preferably, several times in a day, leading to repeated smaller integration phases in the overall pipeline development.

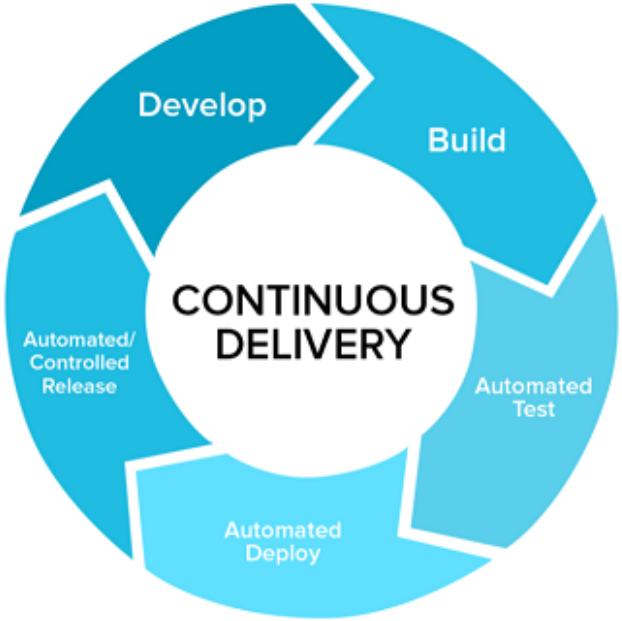
In the **build** phase, the pipeline stages are written/built in the language(s) or tools of choice and appropriate build artifacts are created. **Code** is maintained in our **DVCS**



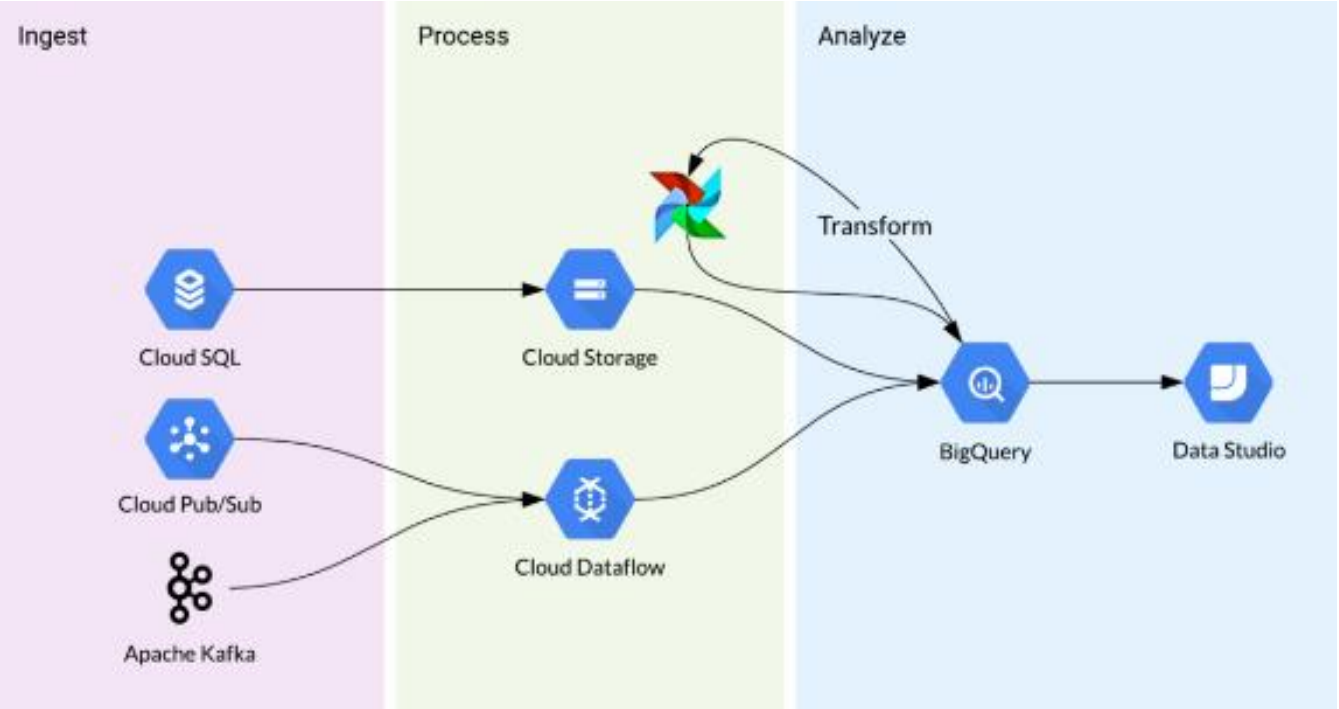
Continuous Delivery (CD)

CD is a development practice to build pipelines such that a set of changes can be delivered or released to production at any time.

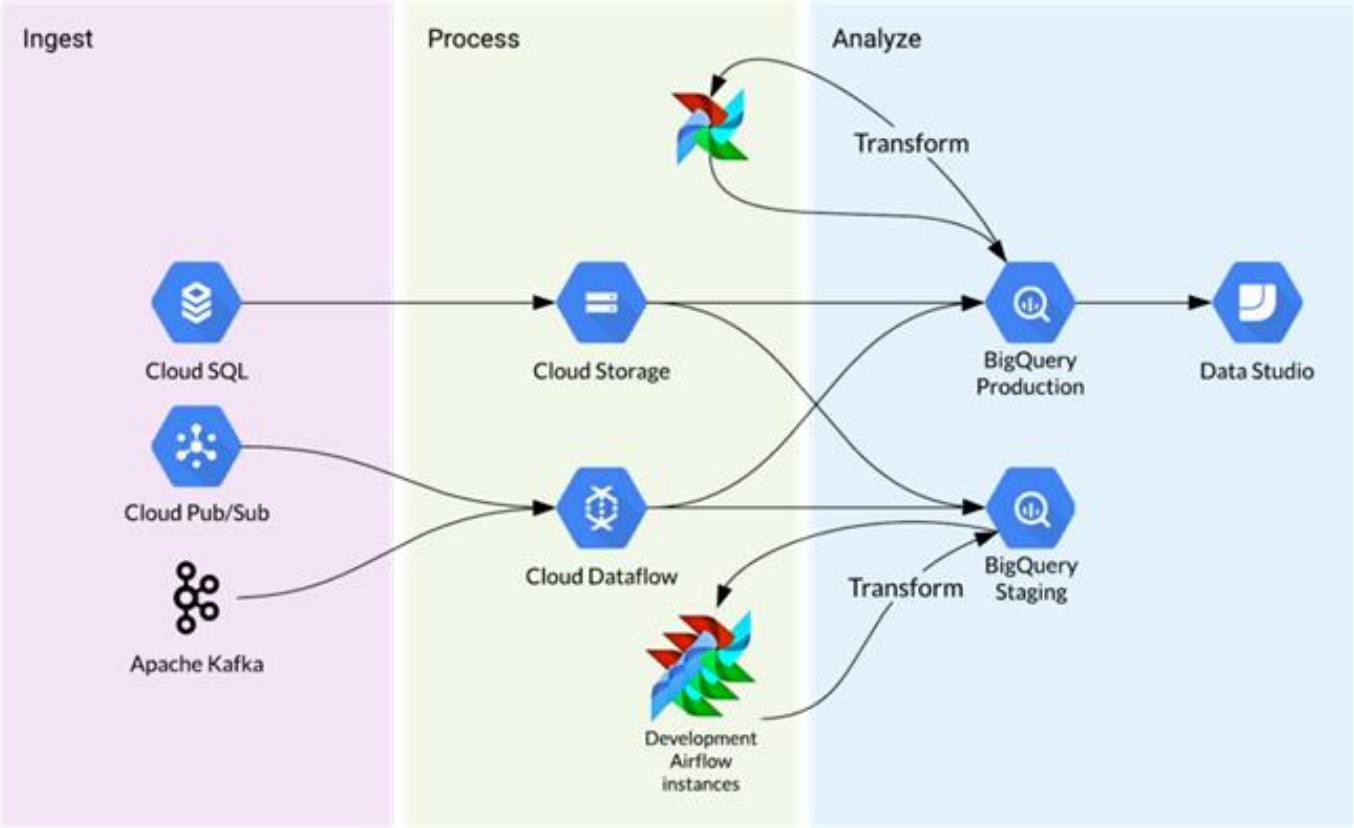
CD is considered an extension of CI and its core focus is on automating the release process to enable hands-free or single-click deployments.



Pipeline CI/CD – First Look



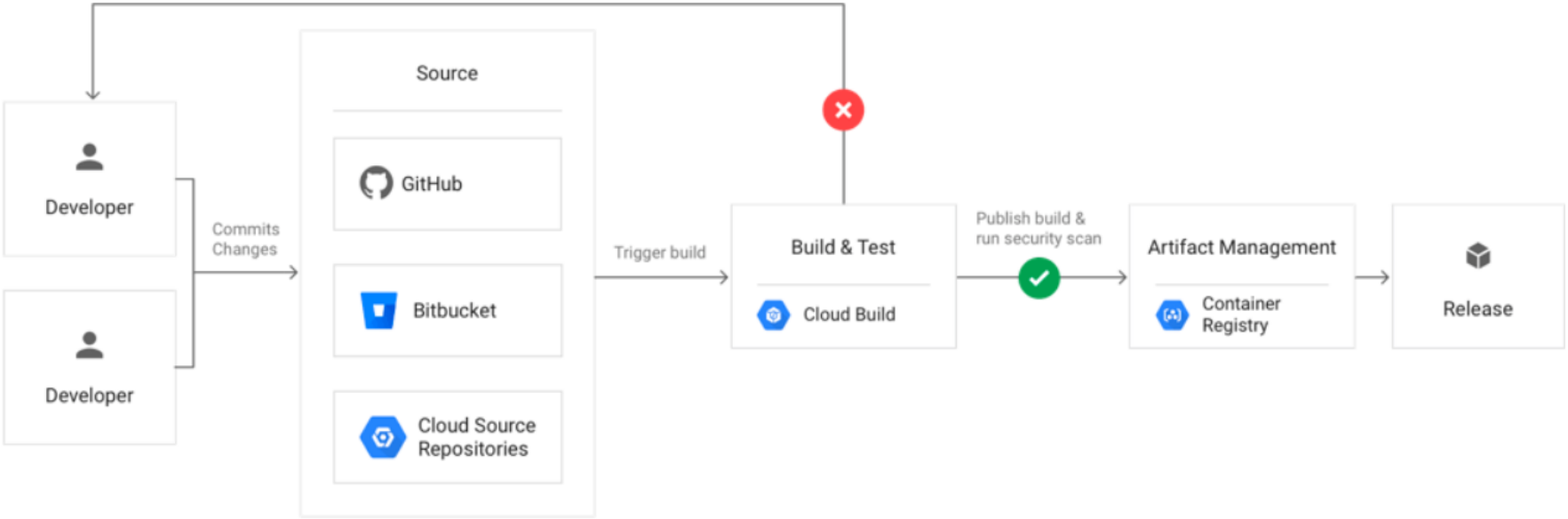
Pipeline CI/CD – Evolution



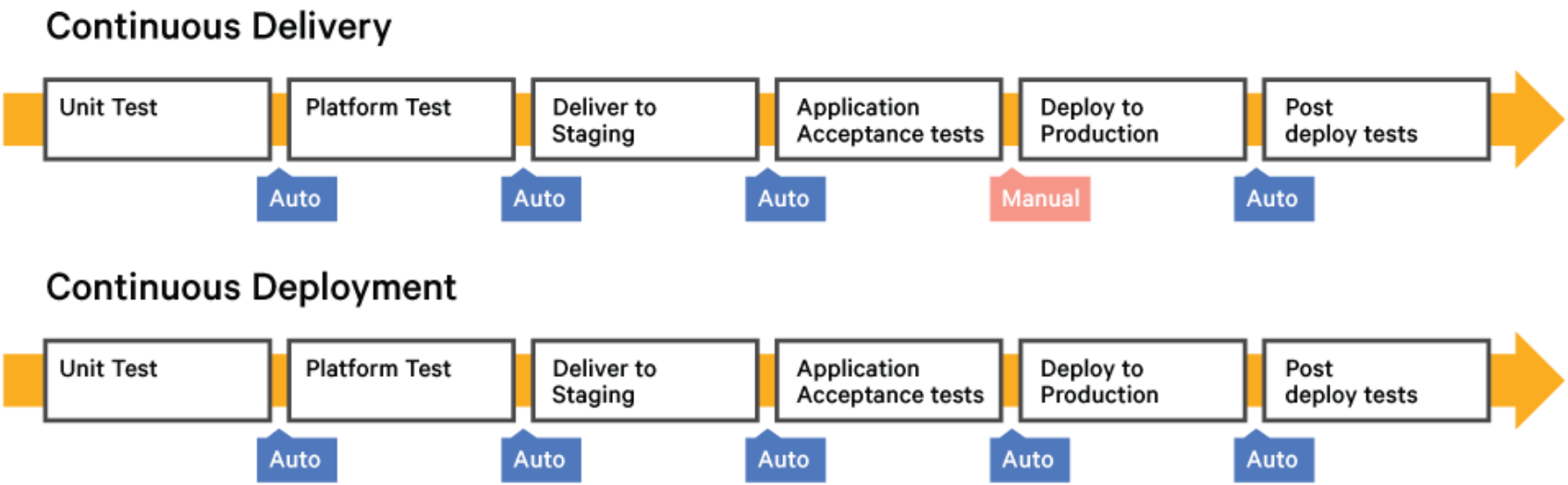
Continuous Deployment

Continuous deployment is a software development practice where the core focus is to release **automated deployments to production** without the user's intervention. It aims to **minimize the time elapsed** between developers writing new line(s) of code and this new code being used by end customers in production.

Continuous Deployment



CD vs Continuous Deployment



Continuous Monitoring & Operations

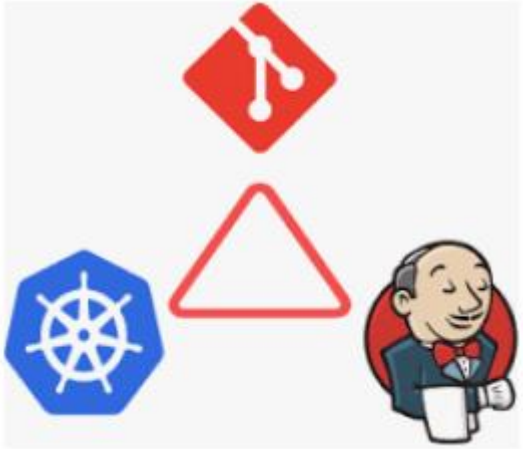
Continuous monitoring is a practice that uses analytical information to identify issues with the data engineering pipeline or its underlying SaaS/PaaS/IaaS.

Monitoring can be classified into two types: **data** monitoring and **pipeline** monitoring.

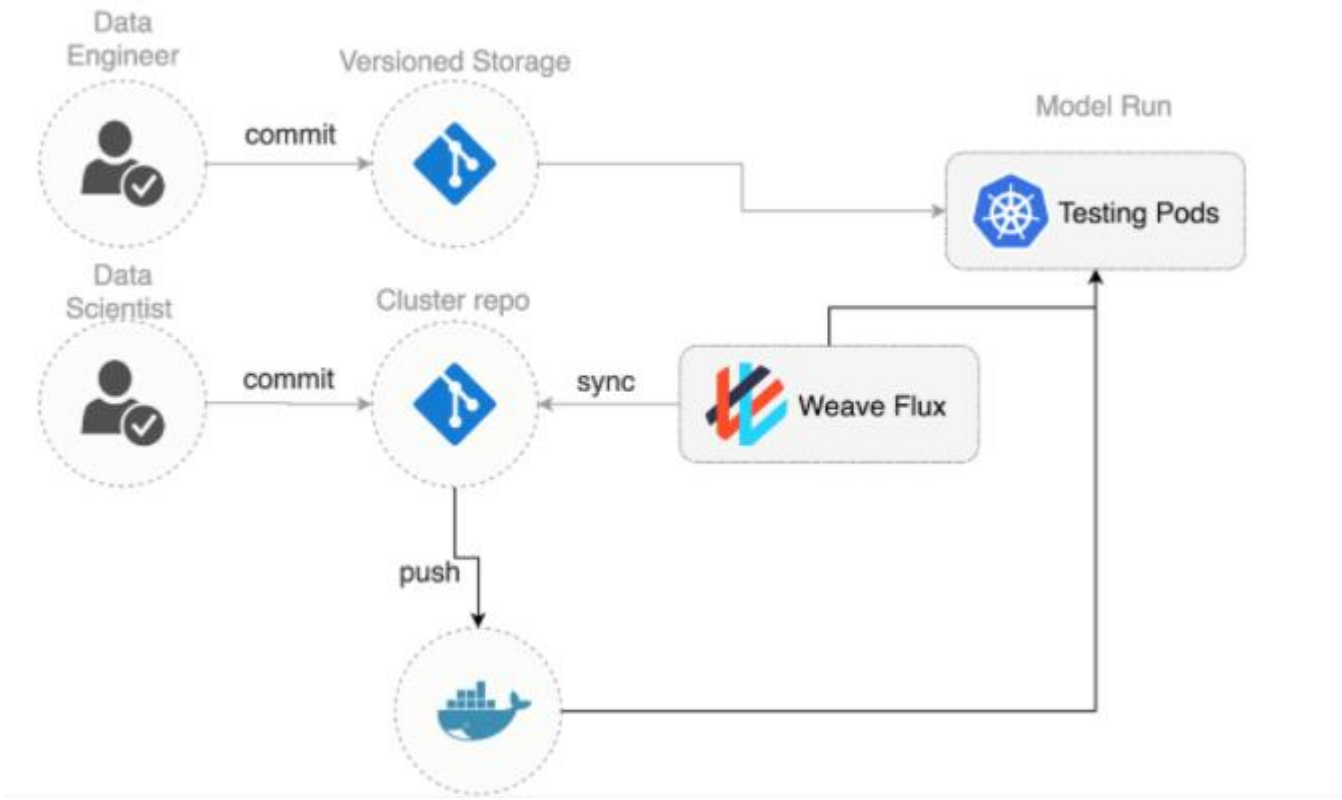
Continuous operations is a practice where the core focus is to mitigate, reduce, or eliminate the impact of planned downtime, such as scheduled maintenance, or in the case of unplanned downtime, such as an **incident**. The incident impact recovery is largely defined by **time, talent and treasure**.

Continuous Feedback

Continuous feedback is a practice where the core focus is to **collect feedback** that improves the data engineering pipeline **ingestion, operation, analysis, or insight**. A common misconception is that continuous feedback happens only as the last phase of the DevSecOps data engineering pipeline lifecycle.

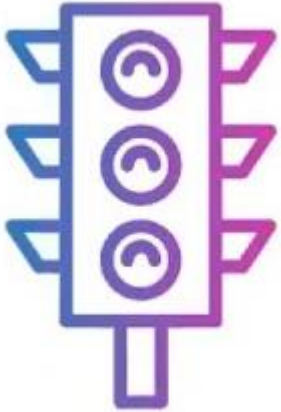


GitOps in Data Engineering Pipelines



SRE Observability

The four golden signals of monitoring are **latency**, **traffic**, **errors**, and **saturation**. If you can only measure four metrics of your user-facing system, focus on these four.

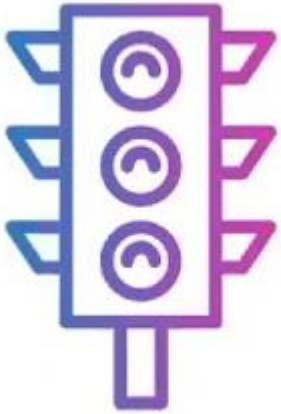


SRE Observability - Latency

The time it takes to service a request. It’s important to distinguish between the latency of successful requests and the latency of failed requests.

Slow errors are worse than fast errors normally! Track error latency, as opposed to just filtering out errors.

SRE Observability - Traffic



A measure of how much demand is being placed on your pipeline, measured in pipeline-specific metrics.

For a pipeline visualization web service, this measurement is usually HTTP requests per second, perhaps broken out by the nature of the requests (e.g., static versus dynamic content).

SRE Observability - Errors

The rate of requests that fail, either explicitly (e.g., HTTP 500s), implicitly (for example, an HTTP 200 success response, but coupled with the wrong content), or by policy (for example, "If you committed to one-second response times, any request over one second is an error").

SRE Observability - Saturation

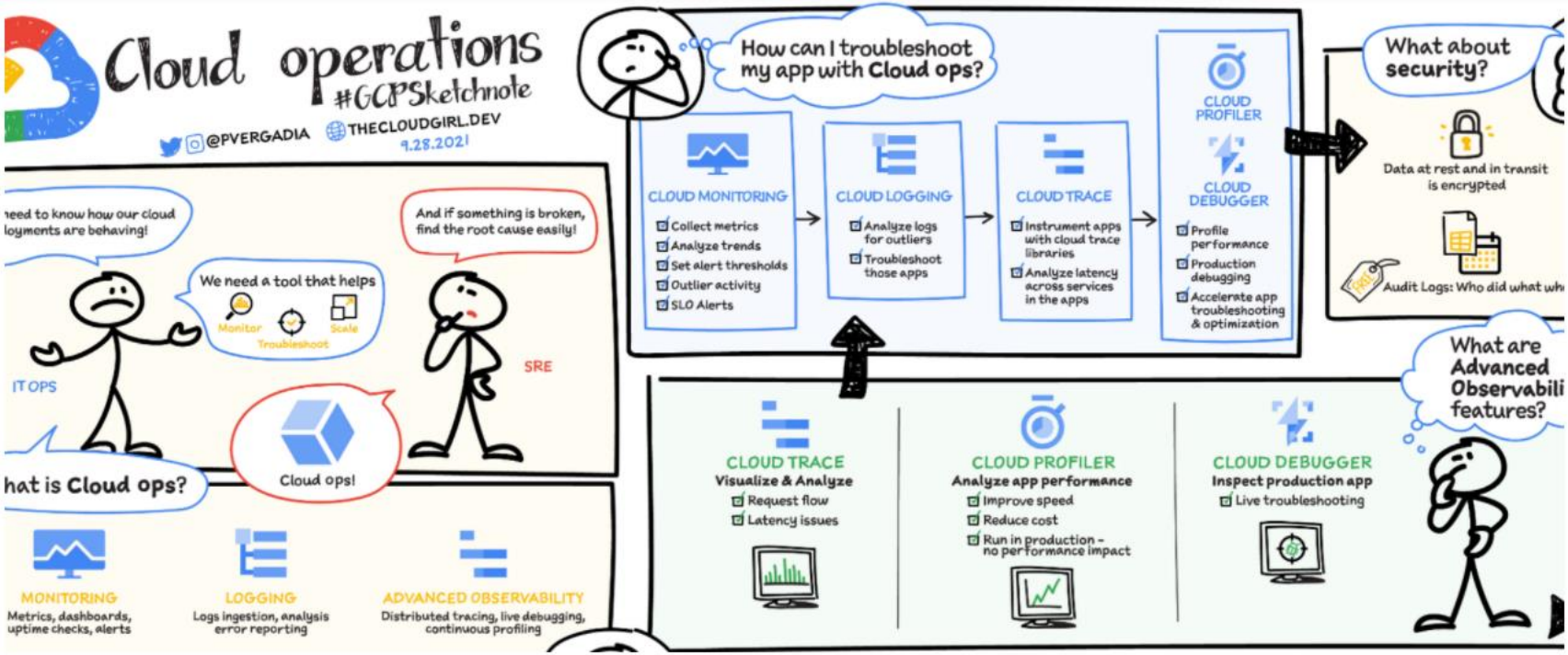
A measure of your holistic health, emphasizing key resources (e.g., in a memory-constrained system, show memory; in an I/O-constrained system, show I/O).

Many systems degrade in performance before they achieve 100% utilization, so utilization targets are key.

Stackdriver AKA GCP Ops Suite

Integrated monitoring, logging, and trace managed services for applications and systems running on Google Cloud. Extendable to poly-cloud or on-premise for pane-of-glass observability.

Google Cloud's Operations Suite



Google Cloud’s Operations Suite

- **Metrics, logs, and traces** are automatically collected and made available in the Console
- **End-to-end monitoring** with Cloud Monitoring and a managed service **(in Preview)** for Prometheus
- **Analyze logs** with Logs Explorer and perform advanced queries with Log Analytics **(in Preview)**
- Create and **monitor service-level objectives** (SLOs) as part of your SRE strategy
- **Alert on metrics** and logs to quickly take **corrective actions**

Google Cloud’s Operations Suite

- Cloud Logging
- Cloud Monitoring
- Cloud Trace
- Cloud Debugger
- Cloud Profiler

GCP Ops Suite – Cloud Logging

Fully managed, real-time log management with storage, search, analysis and alerting at exabyte scale.

- Use Logs Explorer to view your logs from Google Cloud services and others in the Console
- Get deep operational and business insights with Log Analytics (in Preview), powered by BigQuery
- Ingest custom log data from hybrid and multicloud environments
- Regional log buckets available to help you stay compliant with country and industry regulations

GCP Ops Suite – Cloud Monitoring

Gain visibility into the performance, availability, and health of your applications and infrastructure.

- View system metrics collected automatically from Google Cloud services
- Enable SRE best practices extensively used by Google including SLOs and SLIs
- Collect application metrics from workloads on GKE with the managed metrics-collection pipeline
- Use the Ops Agent to collect application metrics and deep system metrics from VMs
- Create custom dashboards and create alerts that integrate with incident management tools

GCP Ops Suite – Cloud Trace

Find performance bottlenecks in production

- Cloud Trace is a distributed tracing system that collects latency data from your applications and displays it in the Google Cloud Console.
- Track how requests propagate through your application and receive detailed near real-time performance insights.
- Automatic analysis of your application's traces to generate in-depth latency reports to surface performance degradations.
- Capture traces from VMs, containers, or App Engine projects.

GCP Ops Suite – Cloud Profiler

Continuous CPU and heap profiling to improve performance and reduce costs.

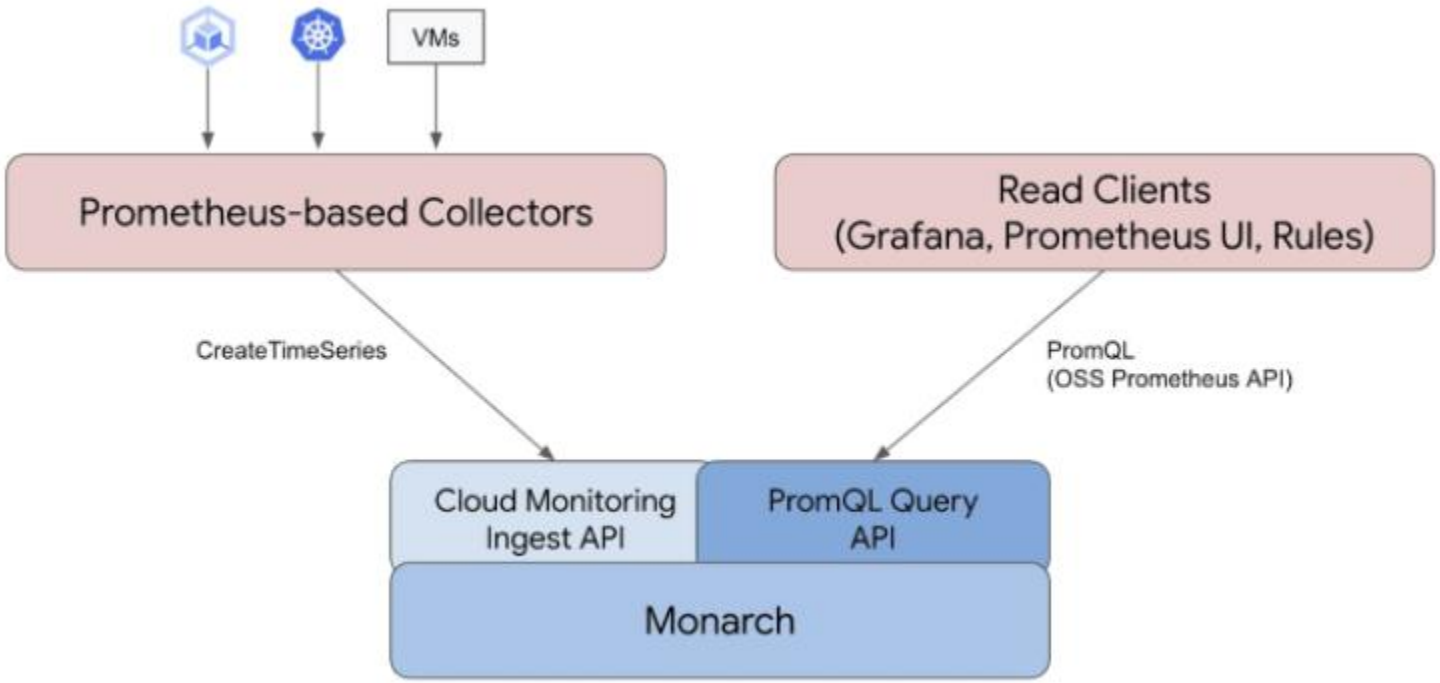
- Continuously analyze performance of CPU or memory-intensive functions executed across an application.
- Visualize the call hierarchy and resource consumption of the relevant function in an interactive flame graph.
- Help developers understand which paths consume the most resources.
- Identify different ways in which code is actually invoked.

GCP Ops Suite – Cloud Debugger

Investigate your code's behavior in production.

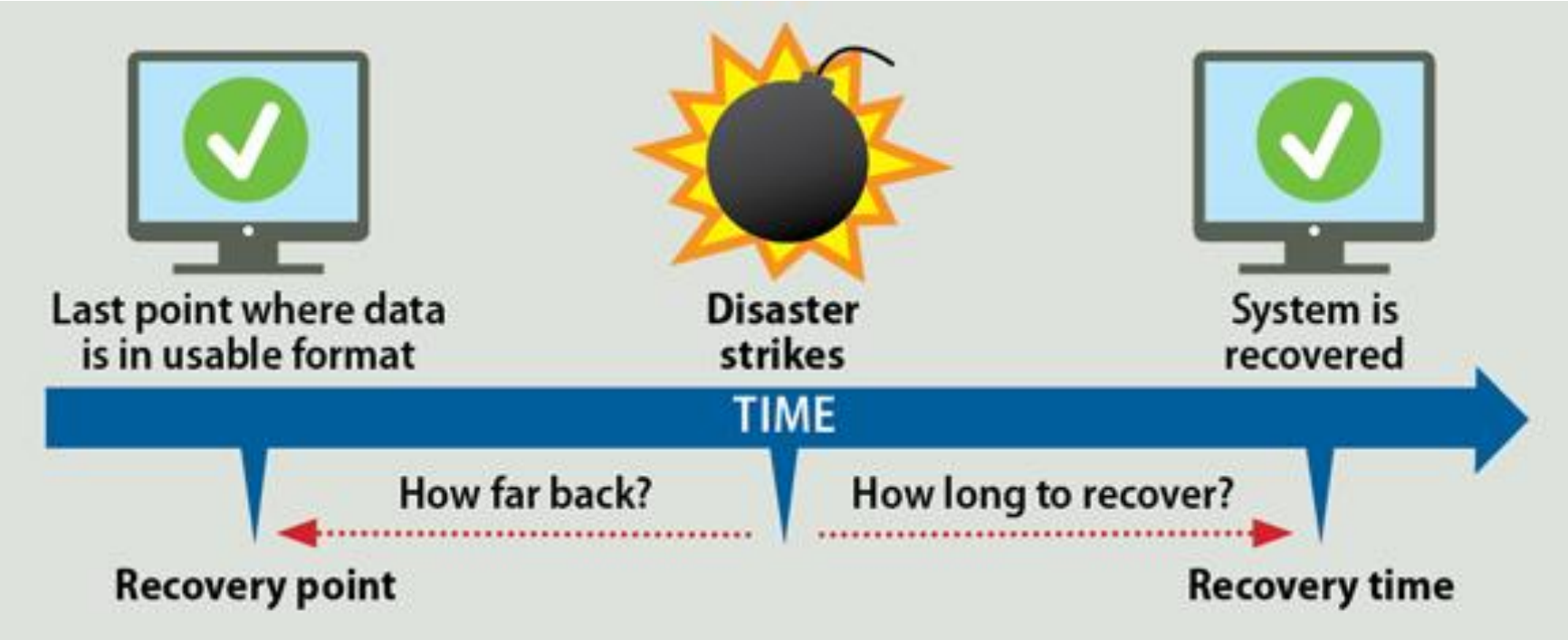
- Inspect the state of a running application in real time, without stopping or slowing it down.
- Capture the call stack and variables at any location in your source code.
- Understand the behavior of your code in production, as well as analyze its state to find those hard-to-find bugs.

Operations Suite Prometheus



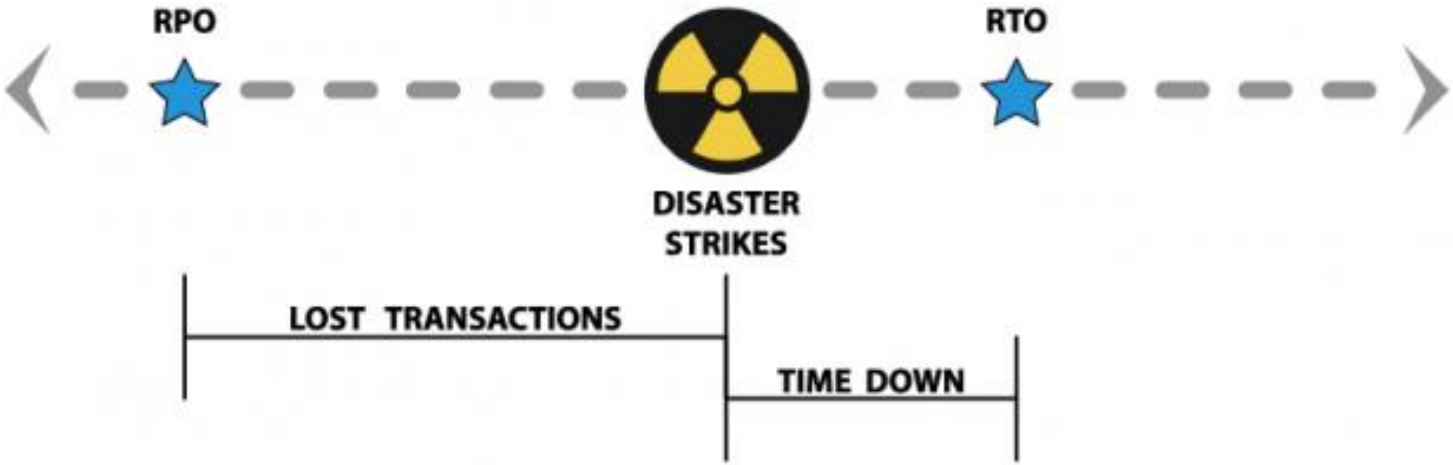
Resiliency in Cloud

RTO/RPO


Last point where data is in usable format
Disaster strikes
System is recovered
TIME
How far back?
How long to recover?
Recovery point
Recovery time

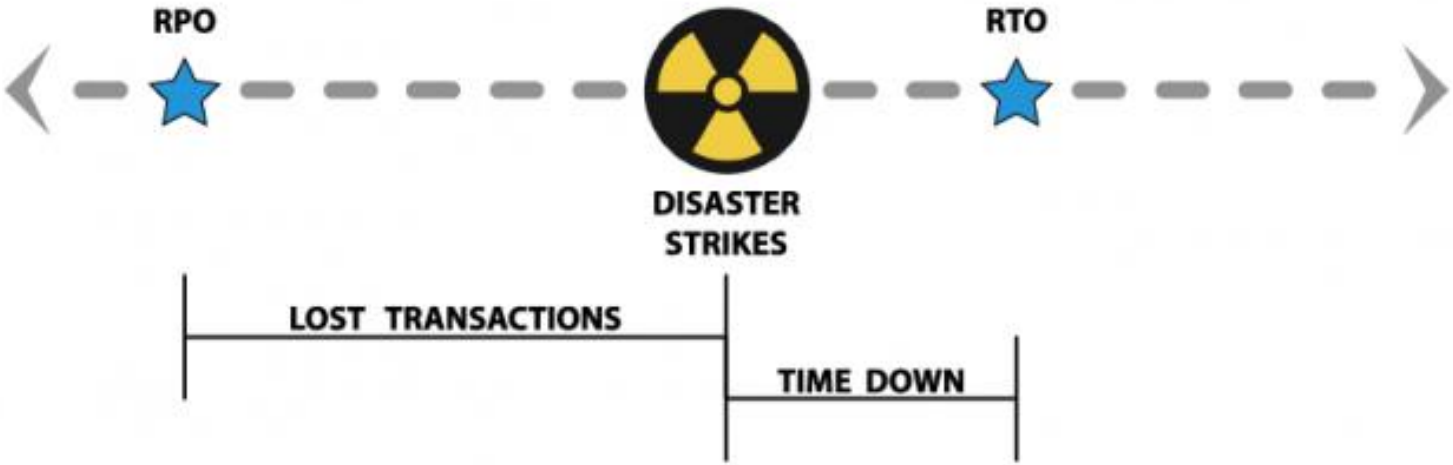
RTO

Recovery Time Objective (RTO) refers to the point in time in the future at which you will be up and running again.



RPO

Recovery Point Objective (RPO) refers to the point in time in the past to which you will recover.



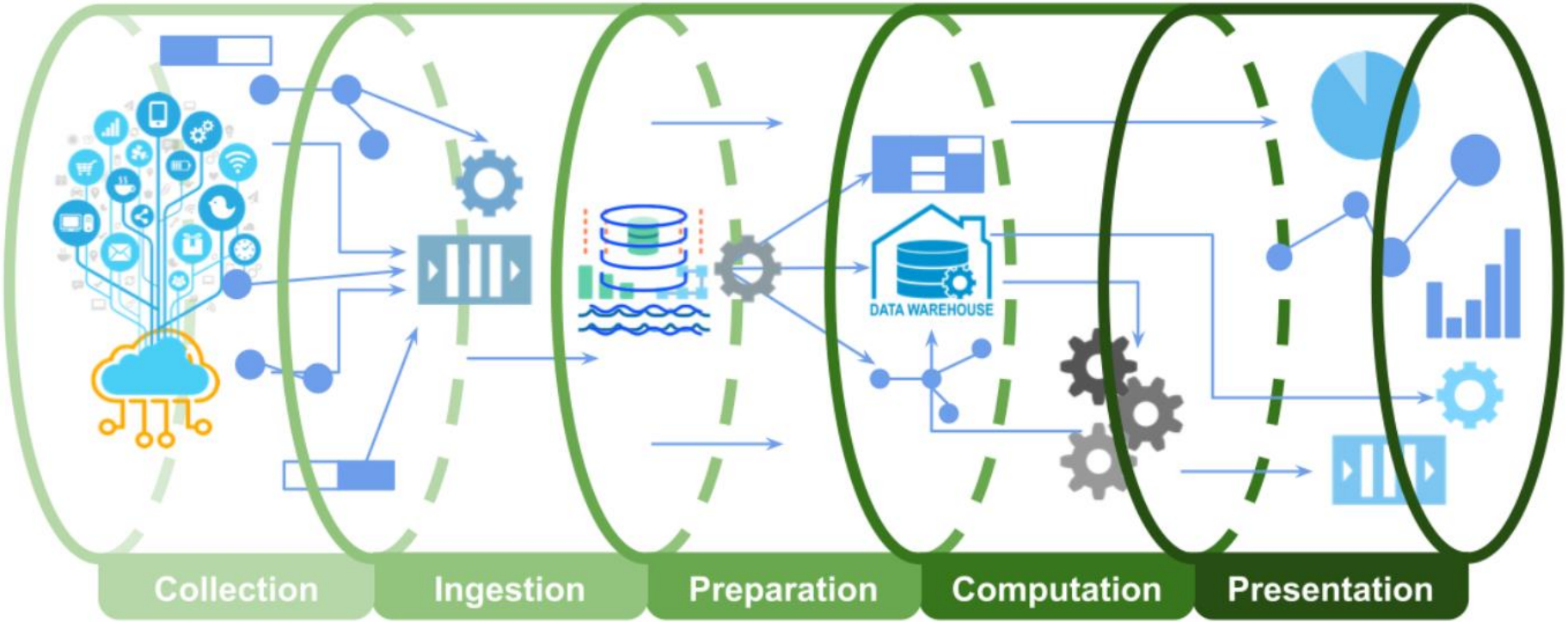
Pipeline SLA/SLO/SLI

Service Level Agreement (SLA) is the customer agreement on service characteristics, notifications and credits.

Service Level Objective (SLO) goals for your pipeline in terms of reliability, retention and throughput.

Service Level Indicator (SLI) refers to metrics for the pipeline, e.g., uptime, durability, freshness, lineage.

Data Pipeline Resiliency



© 2021 by Innovation In Software Corporation

Auto Scaling

Auto scaling refers to changing the number of machines/containers or the size of the machine/container depending on the size of the data to be processed. Increasing the number of machines or the size of the machine is called scaling up, and decreasing them is called scaling down.

The main reasons for scaling up are to

1. Increase the speed of data processing.
2. Handle large input data.

Scaling down is usually done to keep costs low.

© 2021 by Innovation In Software Corporation

Auto Scaling Types

Vertical scaling: Increasing memory and/or disk size of the data processing machine.

Horizontal scaling: Using multiple processes to process a large data set.

Independent processes: Used when independently processing multiple data sets in parallel. It involves one process per dataset.

Distributed systems: Cluster of machines that operate as one unit to processes a data set.

© 2021 by Innovation In Software Corporation

Auto Scaling Types

	Vertical Scaling	Horizontal Scaling - Independent processes	Horizontal Scaling - Distributed systems
Setup	Simple	Simple, if you are starting individual processes. If you are starting individual containers, you might need a service like Kubernetes, AWS Lambda, etc	Difficult to manage. Managed services that make this easier are available.
Autoscaling	No. Scaling requires a stop, scale, and restart.	Yes. Individual processes can be triggered asynchronously. Services like AWS Lambda make this much easier.	Yes, most OS (Spark, Flink) and vendor services (Snowflake) provides autoscaling support.
Data set size	Can handle data in the size of GBs. E.g. c5a.24xlarge has 192GB memory	Only bound by the individual processor size. Can handle data in the size of GBs.	Terabytes and above.

© 2021 by Innovation In Software Corporation

Scaling Decision Point

- What are the costs (service cost, developer time cost, code complexity cost)? Are they justifiable for the performance gain?
- Is this an over optimization ?
- Can this strategy be used for other data pipelines in your company?
- Does code complexity increase or decrease?
- Is it possible to have a short-term(e.g. increase machine size) and long-term(e.g. independent processes for higher scalability) solution for this?
- Is it possible to combine multiple scaling strategies?

© 2021 by Innovation In Software Corporation

Pipeline High Availability

- What are the costs (service cost, developer time cost, code complexity cost)? Are they justifiable for the performance gain?
- Is this an over optimization ?
- Can this strategy be used for other data pipelines in your company?
- Does code complexity increase or decrease?
- Is it possible to have a short-term(e.g. increase machine size) and long-term(e.g. independent processes for higher scalability) solution for this?
- Is it possible to combine multiple scaling strategies?

© 2021 by Innovation In Software Corporation

Pipeline Observability

ML-driven observability into your pipelines — making it easy to instrument, monitor, alert, troubleshoot, resolve, and collaborate on data issues — ultimately **reducing** your data **downtime** rates to begin and thereby **increasing** the **reliability**, **availability** and **resiliency** of your overall data pipelines.

© 2021 by Innovation In Software Corporation

Fault Tolerant Data Pipelines

"Failures are a given, and everything will eventually fail over time." — Werner Vogels

© 2021 by Innovation In Software Corporation

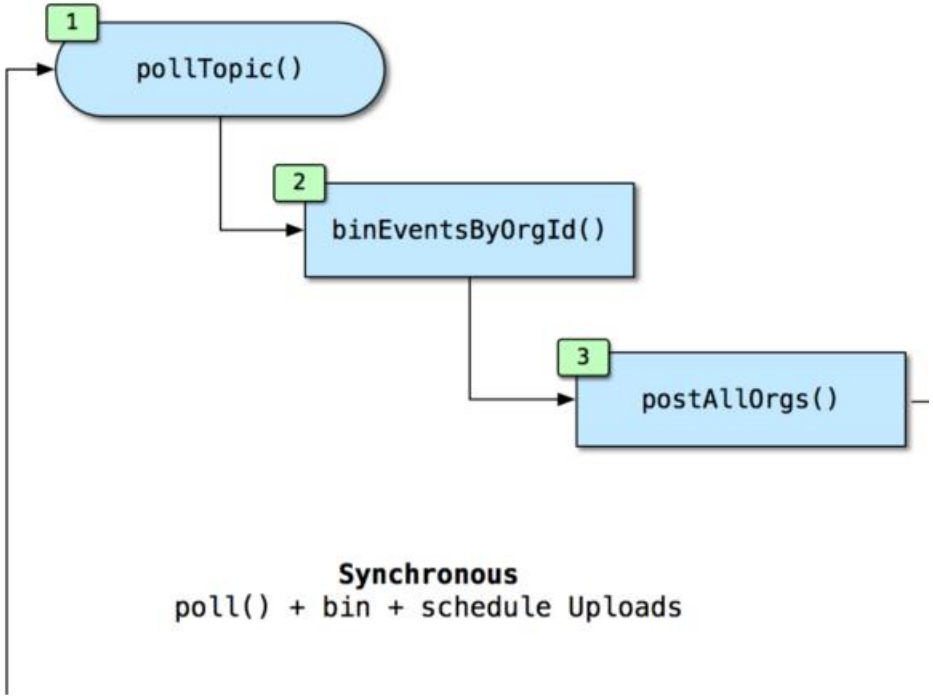
Data Pipeline Faults

At any scale, your data pipeline or service needs to anticipate and mitigate the fundamental and most common faults.

- If you are required to send data downstream via HTTP, it is inevitable that those endpoints will not be available or will be unresponsive at some point.
- In the case of a data pipeline, it is likely you will be required to reprocess data from a point in time

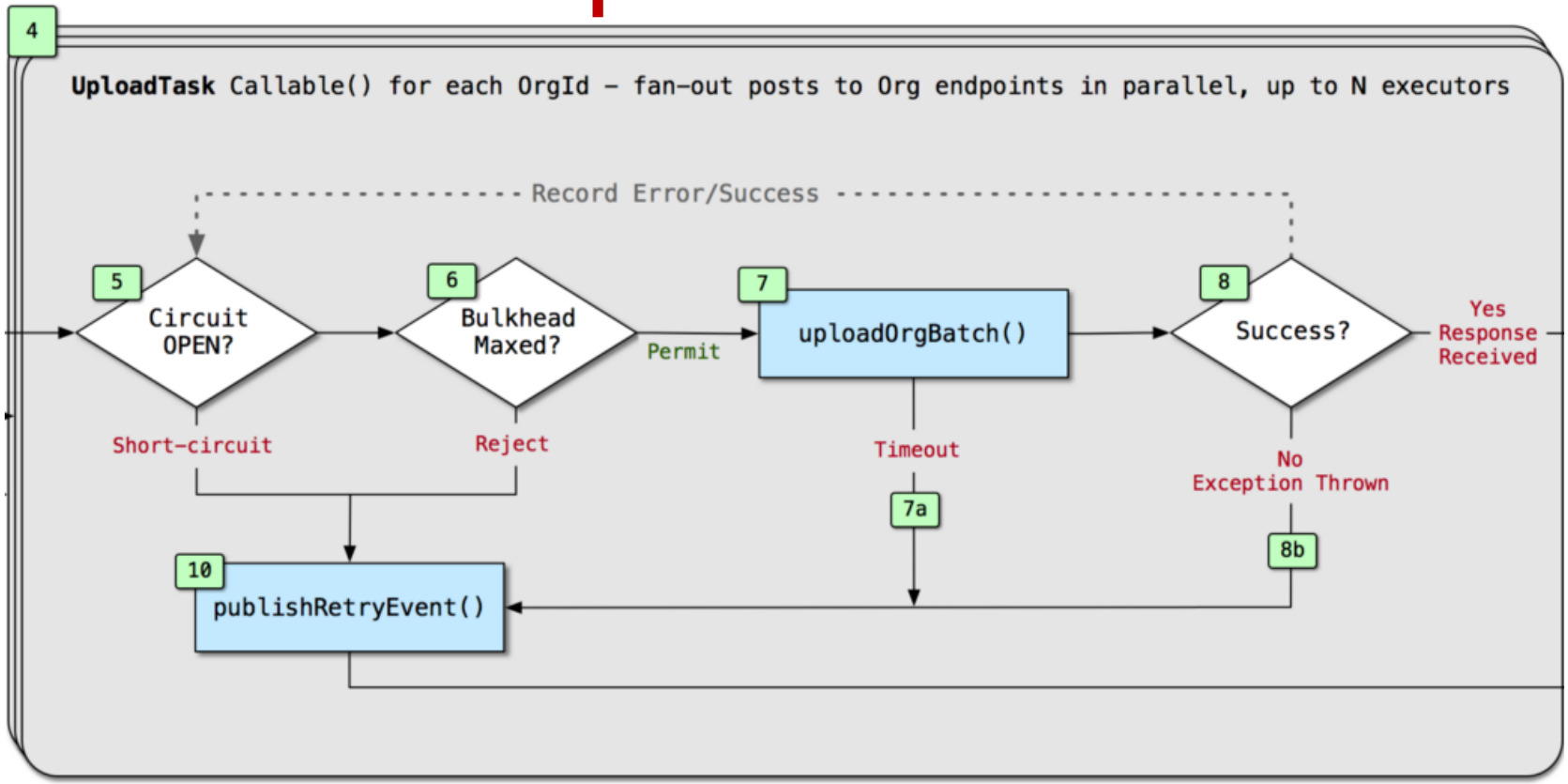
© 2021 by Innovation In Software Corporation

Data Pipeline Faults



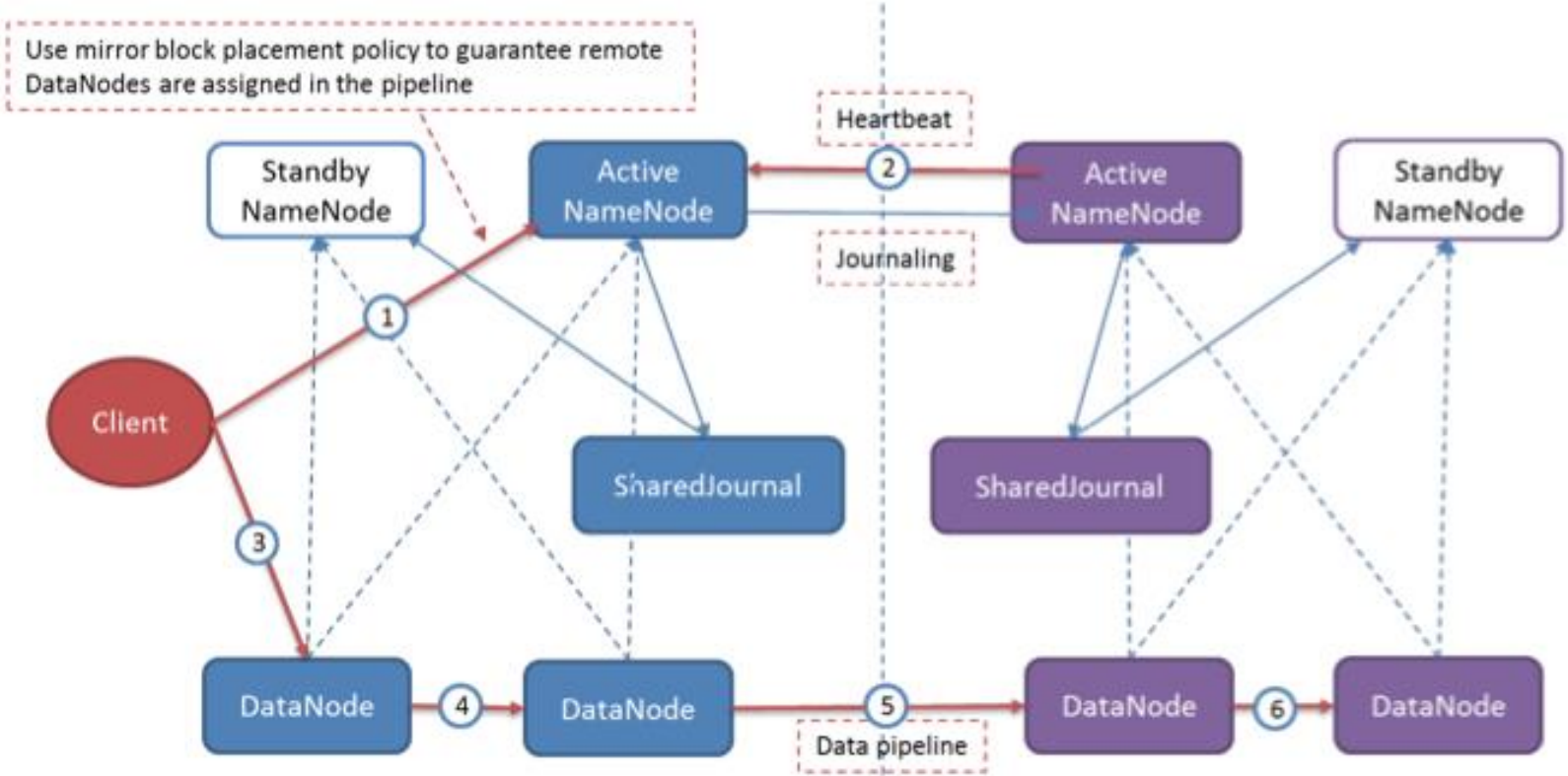
© 2021 by Innovation In Software Corporation

Data Pipeline Faults



© 2021 by Innovation In Software Corporation

Data Pipeline Sync DR



© 2021 by Innovation In Software Corporation

Data DR

Your disaster recovery plans must specify how you can avoid losing data during a disaster. The term *data* here covers two scenarios. Backing up and then recovering data flows, database, log data, checkpoint and other data types fits into one of the following scenarios:

- Data backups
- Database backups
- Message queue backups

© 2021 by Innovation In Software Corporation

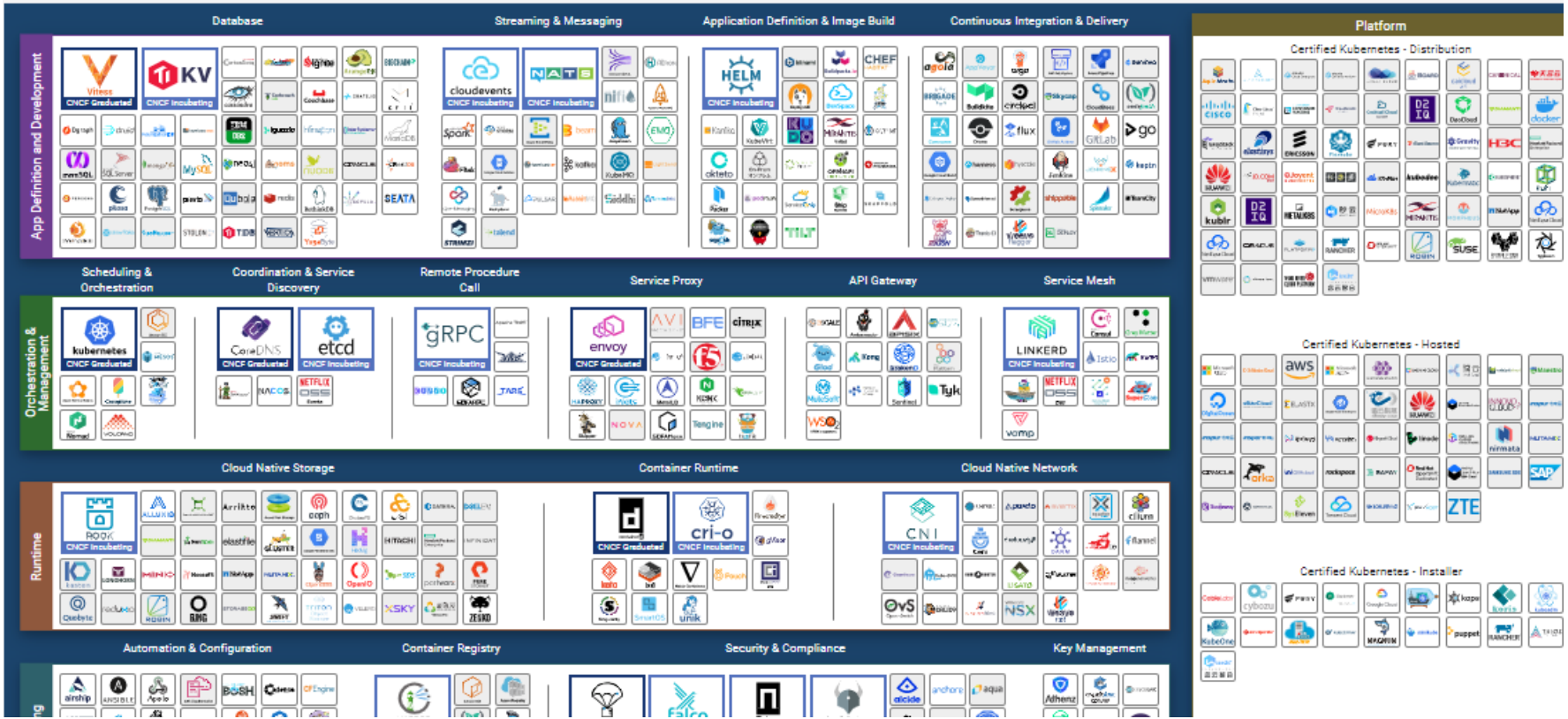
Data DR Building Blocks

DR building blocks:

- Cloud Storage
- Transfer service for on-premises data
- Cloud Interconnect
- Tiered storage
  - Nearline
  - Coldline
  - Archive

© 2021 by Innovation In Software Corporation

CNCF HA/FT/DR Landscape



© 2021 by Innovation In Software Corporation

Dev/CI/CD Transformation

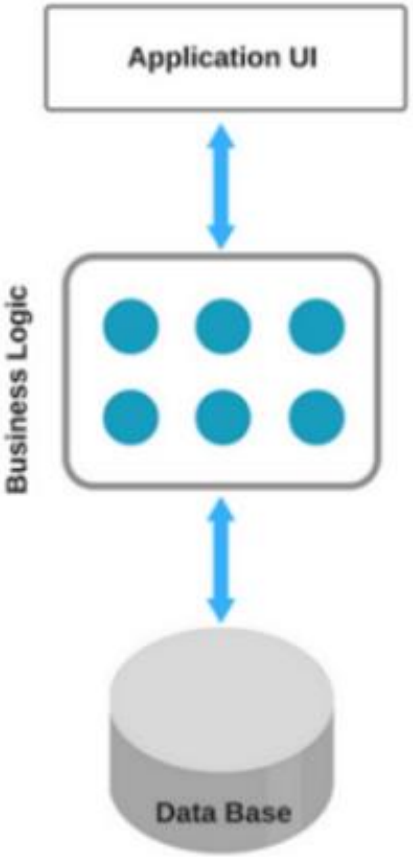
3

Monolithic pipelines

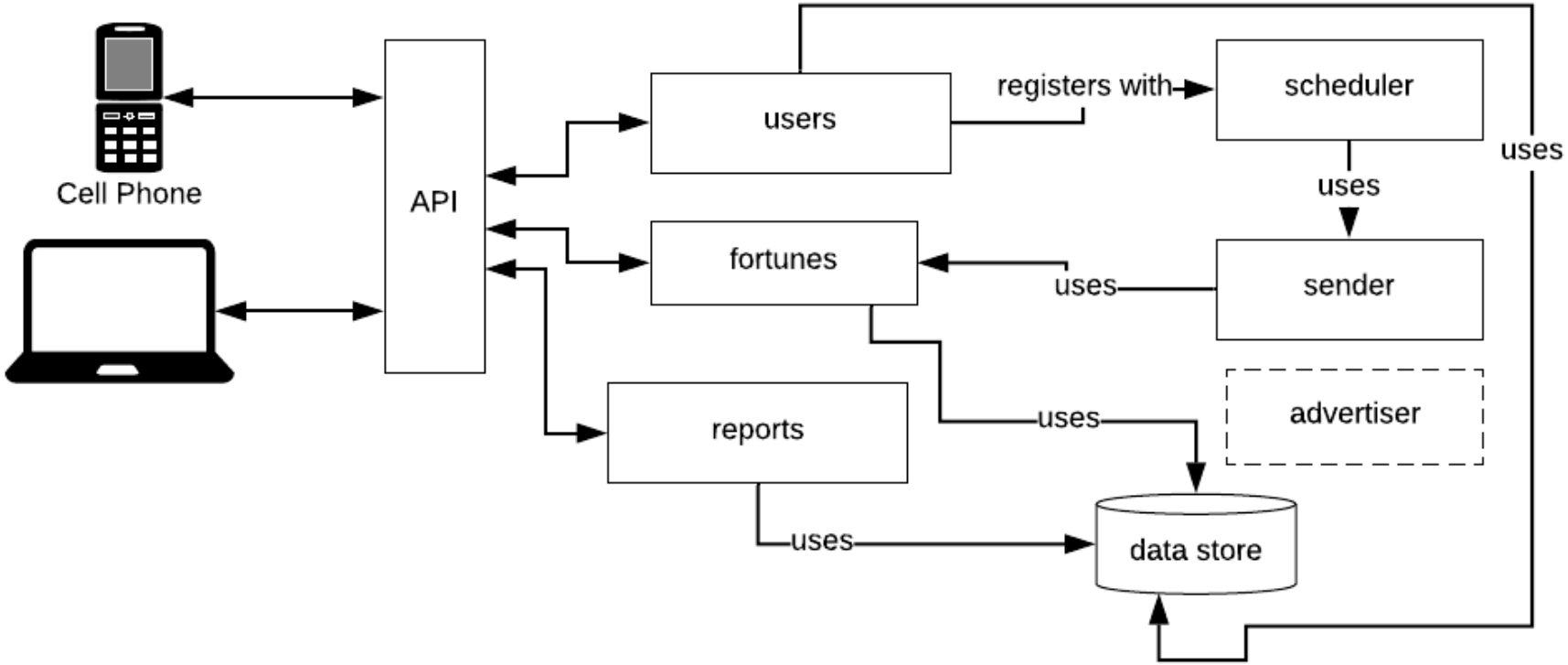
An pipeline in which the user interface, business rules, and data access code is combined into a single executable program and deployed on one platform.

A monolithic pipeline operates independently from other pipelines, performing every step of the process needed to complete the entire business function. It does not share any logic or data across system or organizational boundaries.

Databases are designed for access by single pipeline systems within a single agency, not for access by multiple pipeline systems in multiple agencies simultaneously.



Introduction to Microservices: Monolithic pipeline



Pros and Cons of Monolithic Pipeline

- Simple to develop
- Simple to test.
- Simple to deploy.
- Simple to scale horizontally by running multiple copies behind a load balancer.

- Maintenance
- The size of the pipeline can slow down the start-up time.
- You must redeploy the entire pipeline on each update.
- Monolithic pipelines can also be challenging to scale
- Reliability
- Regardless of how easy the initial stages may seem, Monolithic pipelines have difficulty to adopting new and advance technologies.

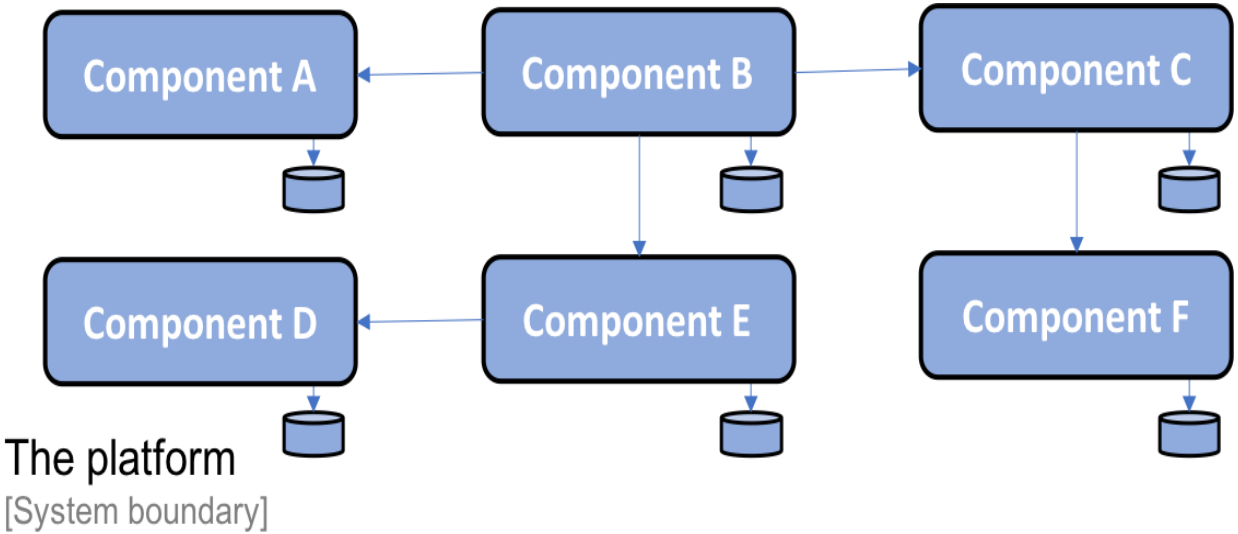
MICROSERVICE OVERVIEW

“The Microservice architectural style is an approach to developing a single pipeline as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.”

- Martin Fowler

© 2020 by Innovation In Software Corporation

Microservices



WHY MICROSERVICES?

The driving force behind microservices is to get the highest quality software possible delivered to customers as fast as possible, at web scale.

© 2021 by Innovation In Software Corporation

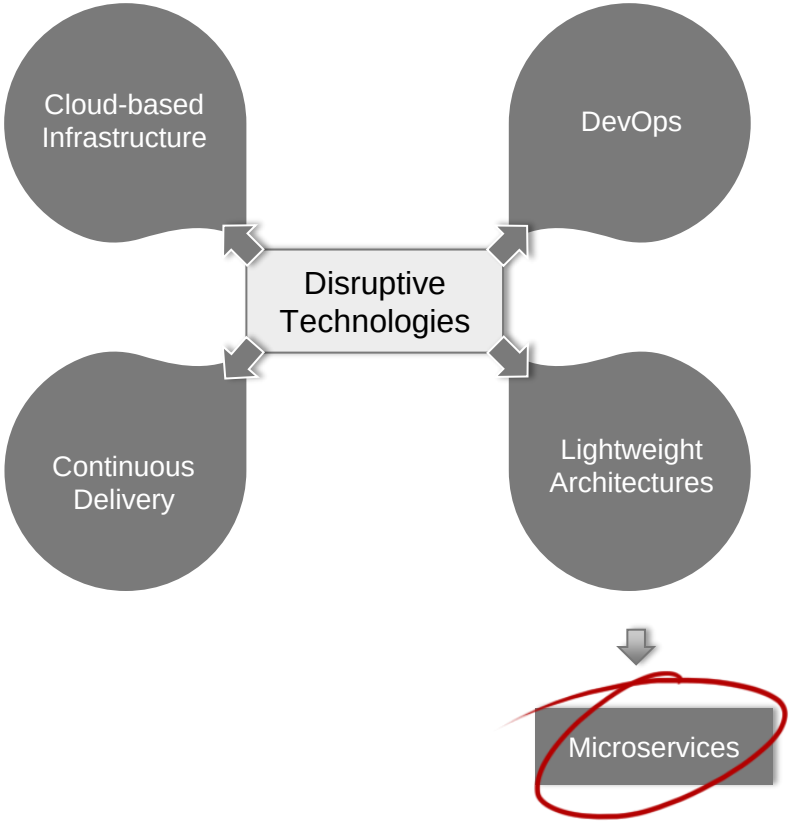
WHY MICROSERVICES?

As new disruptive technologies are introduced, it is critical for DevSecOps teams to evaluate their potential and facilitate experimentation together with the business.

These new disruptive technologies, force IT into a dramatic change and are designed to drive time to market and value for our business teams.

Enterprises who ignore them, will soon or later fall short and loose market share for their competitors.

When combined with Cloud-based Infrastructure, DevOps and Continuous Delivery, Microservices, as a lightweight Architecture approach, enables rapid Enterprise IT transformation and innovation.



© 2021 by Innovation In Software Corporation

WHY MICROSERVICES ARCHITECTURE?

Adopting this architecture increases the potential benefits of complex IT initiatives

Agility	**Agility** allows organizations to deliver new products, functions, and features **quicker**
Reduce time to market	Independent components move new features into production quicker and provide **more flexibility** for piloting and prototyping, thus **reducing time to market**
Composability	**Composability** reduces development time, provides a compound benefit through reusability over time, and guarantees **system scalability**
Comprehensibility	**Comprehensibility** of the software system **simplifies development planning**, increases accuracy, and **accelerates new resource integration**
Polyglotism	**Polyglotism** permits the use of the **right tools for the right task**, thus accelerating technology introduction and increasing solution options
Cost reduction	**Reduce cost** at operating speed by reducing the overall cost of designing, implementing, and deploying services

© 2021 by Innovation In Software Corporation

Microservices Principles

- Microservices are a set of small, independent, composable services.
- Each service can be accessed by way of a well-known API format such as REST, GraphQL, gRPC or in response to event notifications.
- This architectural style de-constructs business processes to their most basic level by creating small, separate processes that then replace large, single applications.

- Breaking a system into smaller parts allows each service to focus on a single business capability.
- Microservices do not live alone.
- They are a part of the larger system.

- They work alongside other services to accomplish the tasks normally handled by one large standalone monolith application.
- They communicate with each other synchronously or asynchronously.
- Microservices make applications easier to develop and maintain.
- BUT A LOT HARDER TO MANAGE

Characteristics of microservices

- Decentralized governance and data management.
- Automated and designed for failure.
- Infrastructure can be built, deployed and verified with CI/CD automation.

Business Motivations: Business Agility

- Speed of change is faster with a more modular architecture.
- React quicker to feature and enhancement requirements.
- Easy integration with new technology, processes (Mobile, Cloud, Continuous DevOps)

Business Motivations: Accelerate Digital Transformation

- Provide more flexibility through capability reuse.
- Helps reduce technical debt.

Business Motivations: Faster On-Boarding

- Facilitates fast on-boarding of new personnel provided the development teams have embraced the MOA philosophy
- Reduce learning curve by providing a more understandable architecture.
- Improves developer velocity.

Business Motivations: Enable PolyGLot Development

- Utilize the best technology choice for each service offering.
- Teams can code in the languages and framework within the scope of their expertise: e.g., Java, .NET, Node.js, Python, Go, Ruby, etc…
- Incremental approaches centered on smaller components radically reduces risk.

Pros & Cons

- Easy to develop
- Easy to understand
- Easy to deploy
- Easy to monitor each service
- Flexible Release Schedule
- Use standard APIs (JSON, XML)

- Requires embracing the philosophy of Microservice Oriented Architeture
- Requires more deployments
- Requires translation (JSON, XML)
- Requires more monitoring
- Operations configuration more complex
- System can be very complex

Microservice Reviewed

Independent services increase the speed of experimentation

Long standing autonomous teams responsible for each service

Ecosystem enables innovation through composing services

Services can be built with diverse set of technologies

Building small and independent services lowers the developer's cognitive load

Services have independent lifecycle

Ecosystem enables growth and scale of operation

Microservice Traps

- Message loss
- Duplicate messages
- Data corruption
- Resiliency
- Reliability
- Variability
- Authentication
- Monitoring
- Hardening

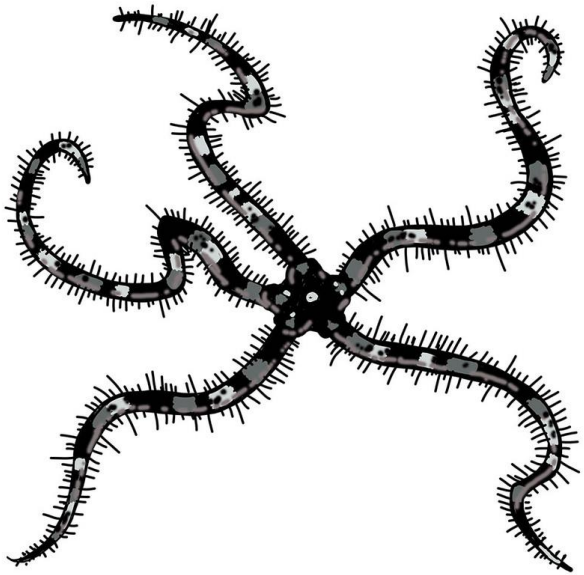
Nanoservices

The basic difference between the two frameworks is that nanoservices are considered smaller siblings of microservices. Nanoservices are designed to perform a single function, whose output is exposed through a specific API endpoint (command).

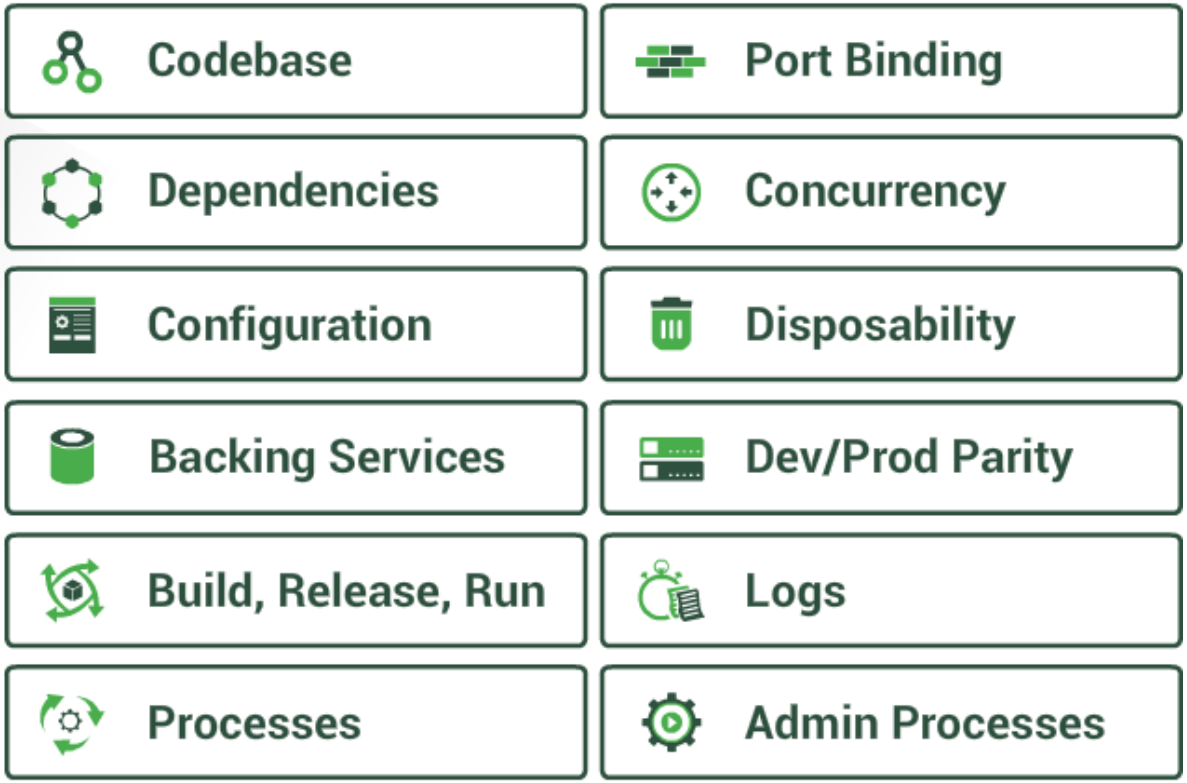
Nanoservices are fully discoverable among each other. Each one can link with other services to perform additional actions and extend functionality.

Brittle Services

- Brittle services are those that composite so many features and functions that they begin to become mini-monoliths.



12-Factor Application Building Blocks



Why 12 Factor?

The methodology was drafted by developers at Heroku and was first presented by Adam Wiggins circa 2011.

The twelve-factor app is a methodology for building software-as-a-service apps that:

Use **declarative formats** for setup automation, to minimize time and cost for new developers joining the project;

Have a **clean contract** with the underlying operating system, offering **maximum portability** between execution environments;

Are suitable for **deployment** on **modern cloud platforms**, obviating the need for servers and systems administration;

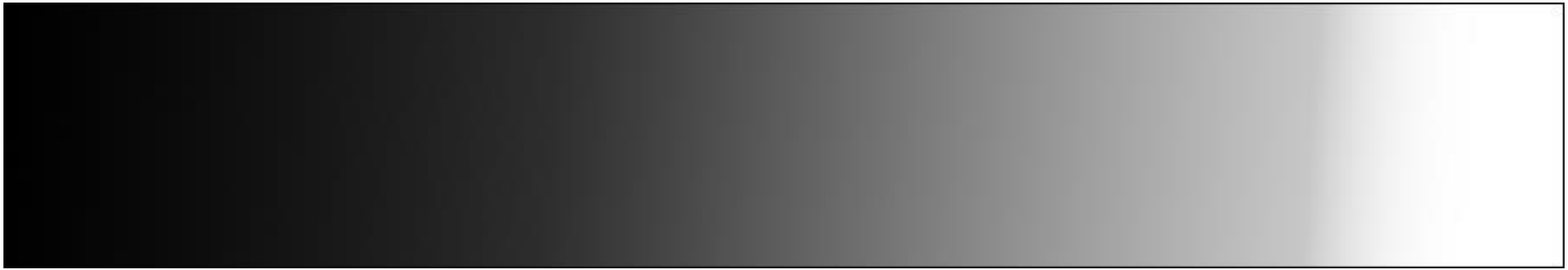
Minimize divergence between development and production, **enabling continuous deployment** for maximum agility;

And can **scale up** without significant changes to tooling, architecture, or development practices.

- Adam Wiggins, 12factor.net

A Word about Implementation

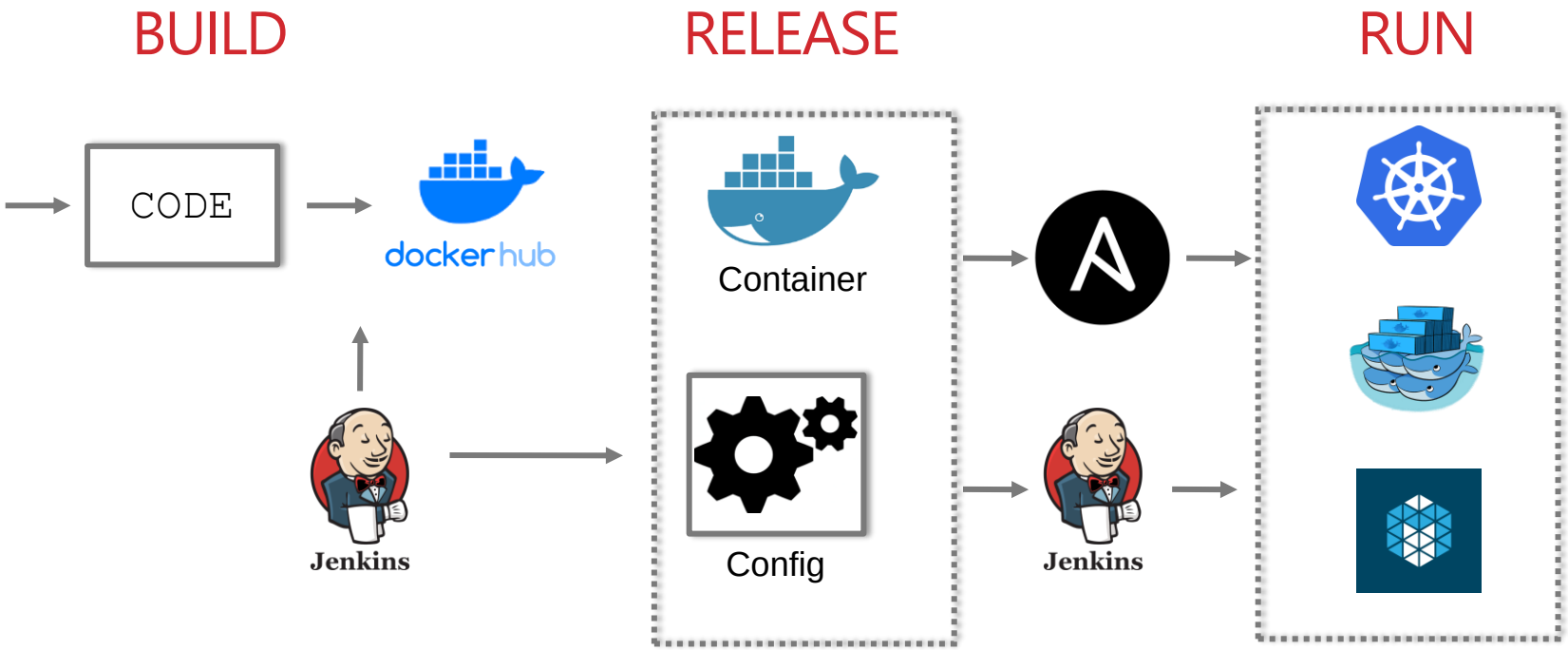
It’s shades of gray.



Adopting 12 Factor App is not an all or nothing thing.
Adopt what you can, when you can.

DEVELOPMENT, DEPLOYMENT, OPS

STRICTLY SEPARATE BUILD AND RUN STAGES

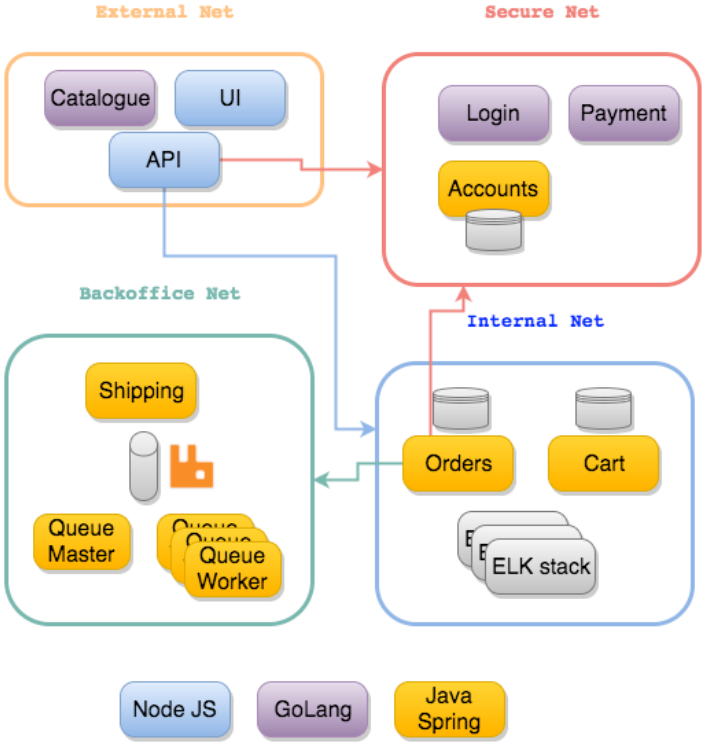


© 2021 by Innovation In Software Corporation

KUBERNETES:CLOUD-NATIVE APPLICATION DEPLOYMENT

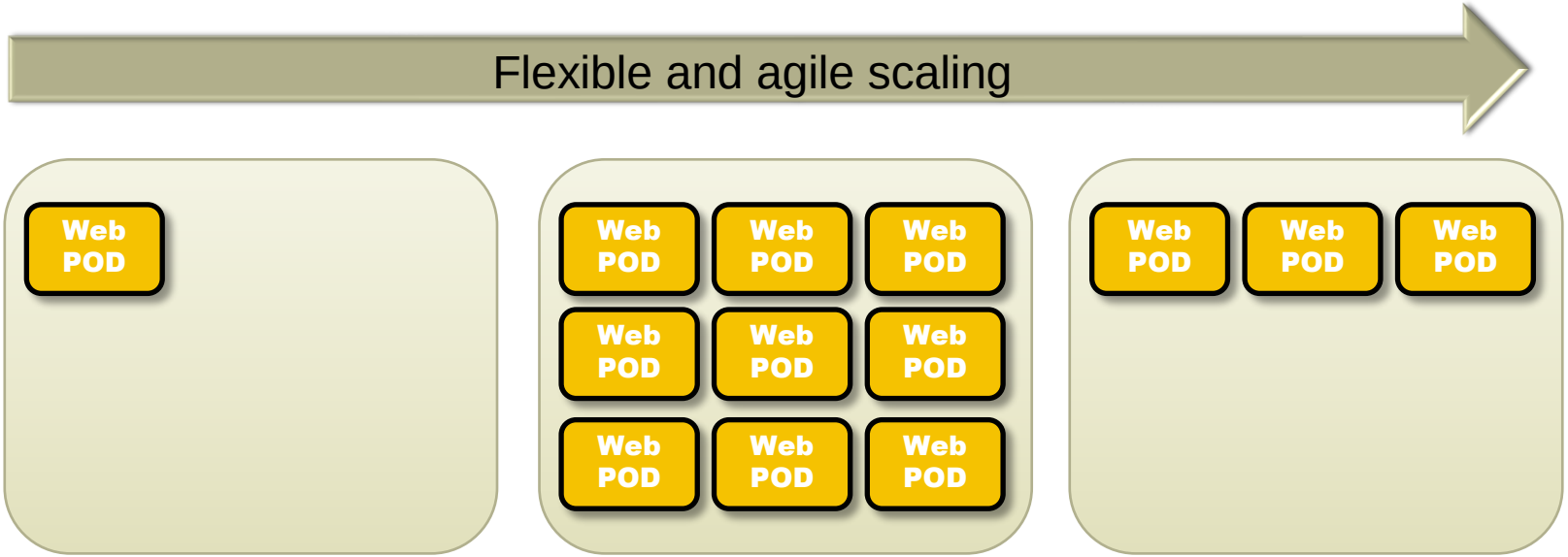
Cloud-native applications, aka microservice-based or 12-factor apps

- **Cloud-native applications are composed of small, independently-deployable, loosely-coupled services**
- **Kubernetes makes it easy to deploy, update, and coordinate operations between multiple containerized service components**
- **Kubernetes project actively enhancing features to better support and manage stateful applications**



KUBERNETES: ELASTIC SERVICES

Kubernetes supports manual and automated scaling of application services based on demand for resources



KUBERNETES: CI/CD PIPELINES

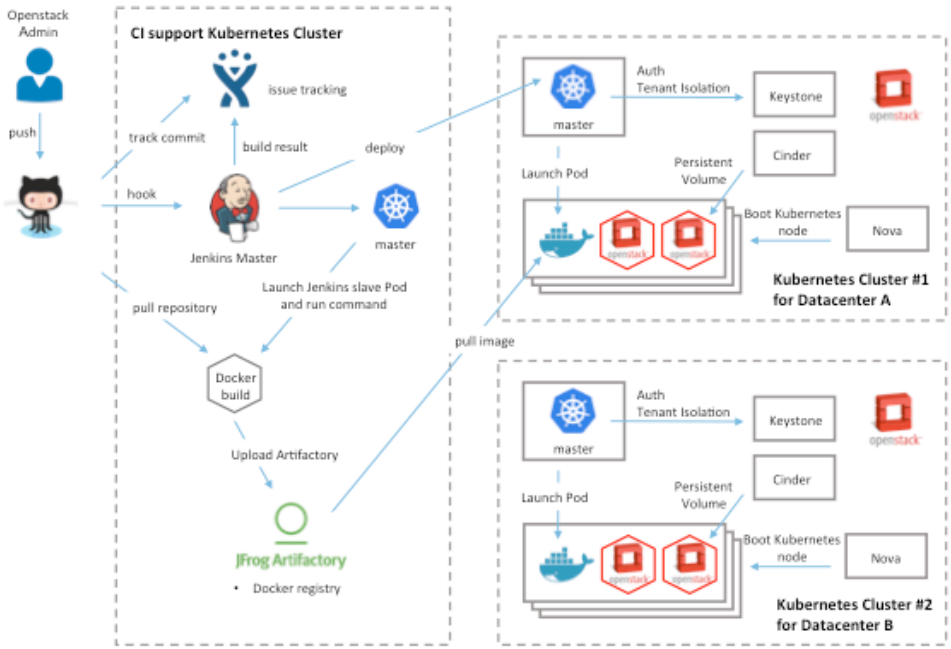
Managing execution environments in a CI/CD pipeline

- **Automated systems can push to logical environments in the same cluster, or to different clusters, using the same control plane API**
- **Kubernetes labels and annotations allow users to organize resources into separate environments without changing code or functionality**
- **Label selectors target specific sets of resources for control and exposure**

EXAMPLE: KUBERNETES CI IN PRODUCTION AT SCALE

YAHOO! JAPAN

- **Built automated system to build and deploy containerized applications on Kubernetes**
- **Kubernetes clusters built from VM's in private OpenStack environments**
  - 1,000 nodes
  - 42,000 Containers
  - ~2,500 Applications



KUBERNETES: PLATFORM AS A SERVICE (PAAS)

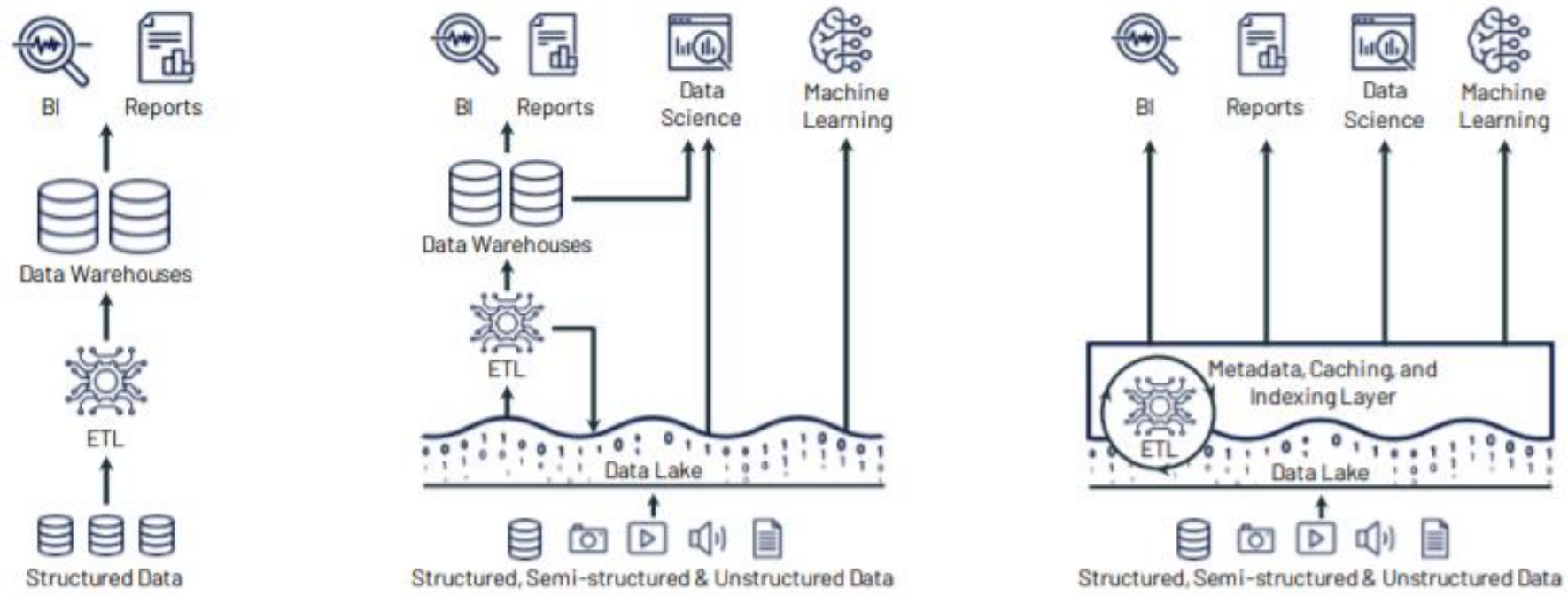
Many PaaS implementations run on Kubernetes

- **Kubernetes control plane built on open API – components like controllers and schedulers are pluggable and extensible**
- **Vendors such as Red Hat and Deis* build more advanced features and services on top of Kubernetes infrastructure**

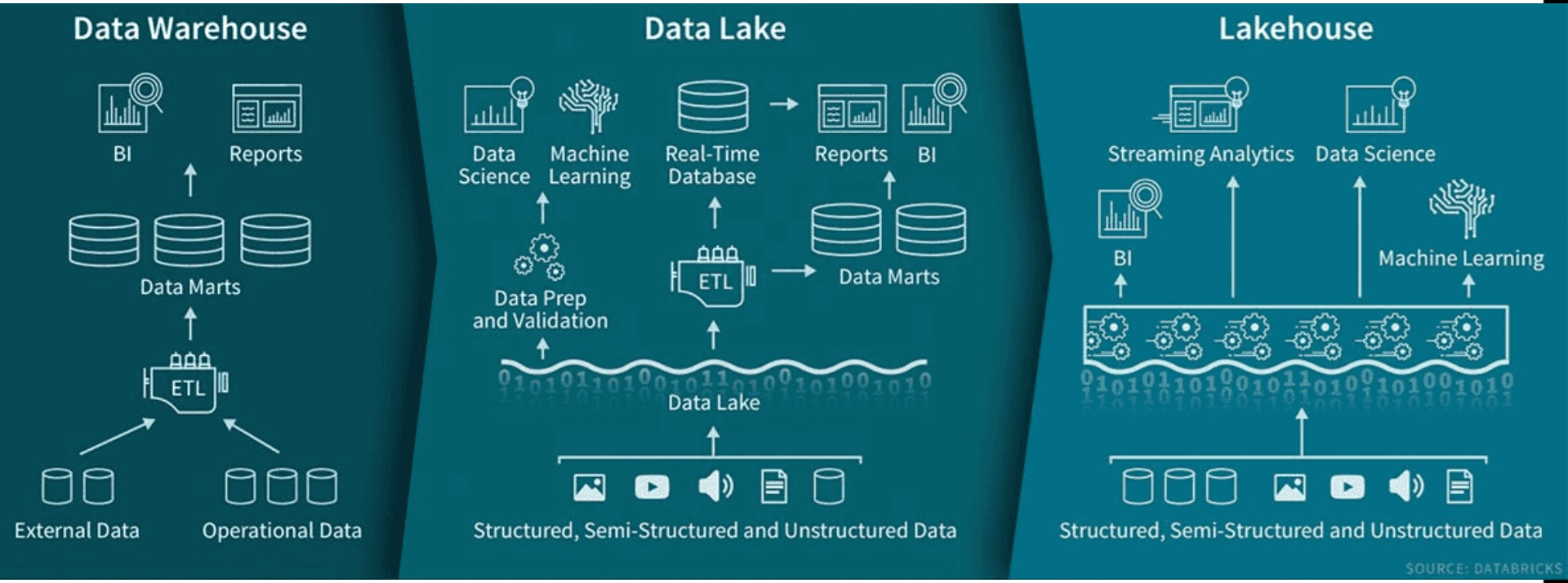
*Acquired by Microsoft March 2017

Data Engineering Foundation

Big Data Evolution



Big Data Evolution



Big Data Use Cases

Customer Sentiment Analysis

Behaviour Analytics

Customer Segmentation

Predictive Support

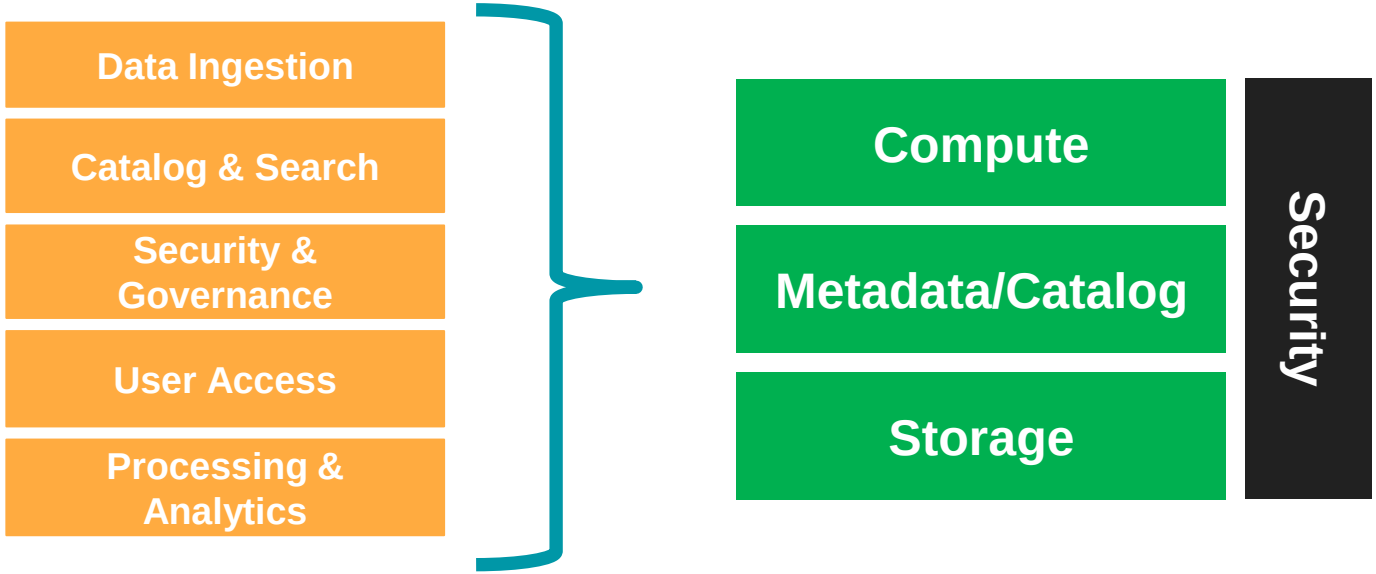
Fraud Detection

ETL Pipelines

Application Analytics

Data Warehouse
Data Lake
Lakehouse

Lakehouse Architecture



Key pillars of a lakehouse

Architectural layers of a lakehouse

Lakehouse Security

Identity & Access Management (application)

Application Security

Data Protection

Infrastructure Security

Network Security

Identity & Access Management (platform)

Multi-account Best Practices

GCP Data Engineering Foundation

Big Data Services

- Pub/Sub.
- Dataproc.
- Dataflow.
- Dataflow.
- Dataprep.
- Datalab.
- Data Studio.
- Cloud Composer.
- BigQuery.

ML Services

- Cloud ML Engine.
- Pretrained APIs
  - Google Cloud Video Intelligence
  - Google Cloud Speech Google Cloud Vision
  - Google Cloud Natural Language Google Cloud Translation.
- AutoML.
  - AutoML Translation
  - AutoML Natural Language and Vision
- Dialogflow.

Experiment –

Building Data Pipelines

Dataflow

#GCPSketchnote

@PVERGADIA THECLOUDGIRL.DEV

What is Dataflow?

SERVERLESS, DATA PROCESSING SERVICE FOR BOTH STREAMING AND BATCH DATA

BATCH DATA

STREAMING DATA

CREATE DATA PROCESSING PIPELINE

Apache Beam (SDK)

DEPLOY & EXECUTE

Dataflow

AUTOMATE Operations & Management

Data Pipeline

Dataflow Job

How does it WORK?

DATA PROCESSING PIPELINE

STEP 1 Read Data Source — Files, BigQuery, Bigtable, Pub/Sub, Custom Source

STEP 2 Transform Data — PCollection 1 → PCollection 2 → PCollection 3

STEP 3 Write Data Sink — Files, BigQuery, Bigtable, Pub/Sub, Custom Source

Dataflow — WORKER VIRTUAL MACHINES EXECUTE THE DATA PROCESSING

Worker Virtual Machines — Autoscale up or down dynamically as needed

Dataflow Streaming Engine — Separate Compute, Separate Storage

BATCH PIPELINES SCALE SEAMLESSLY, WITHOUT ANY TUNING REQUIRED.

Dataflow → FEATURE Service-based Dataflow Shuffle → Data Grouping & Joining → Scale to 100s of TBs

Dataflow PRICING

DATAFLOW — Per second billing

DATAFLOW JOB — Batch or Streaming — (FlexRS) flexible resource scheduling = low cost

AUTOSCALING — Worker(s) — Better price to performance

CUSTOMIZABLE — vCPU + Memory + Storage — Each billed on per second basis

SECURITY for Dataflow Pipelines

EXACTLY-ONCE PROCESSING

0 Avoid risk of data loss

1 Turn off public IPs

2 VPC controls

3 Customer managed encryption key (CMEK)

HOW TO USE Dataflow

1 DATAFLOW SQL VIA BIGQUERY
- Use SQL from BigQuery web UI
- Read from pub/sub, cloud storage or BigQuery
- Write to BigQuery

2 DATAFLOW TEMPLATES
- Share pipeline with teams
- Easy & repeatable Pre built templates

3 AI PLATFORM NOTEBOOKS
- Use latest data science and machine learning frameworks

Dataflow Use case example

Streaming analytics — Real-time AI

Trigger → Ingest → Enrich → Analyze → Activate

Architecture

Edge: Mobile, Web, Data Store, IoT → Pub/Sub → Apache Beam (SDK) / Dataflow Streaming: Analyze, Activate

BigQuery, AI Platform, Bigtable — Backfill/Reprocess — Dataflow Batch

Studio, Third-party BI, Cloud Functions

Creation Flow

Configure source to push event message to Pub/Sub Topic → Create Pub/Sub Topic and Subscription → Deploy streaming or batch Dataflow job using templates, CLI, or notebooks → Create dataset, tables, and models to receive stream → Build real-time dashboards and call external APIs

Google Cloud Dataflow definition

“A fully-managed cloud service and programming model for batch and streaming big data processing”

Main features

- Fully Managed
- Unified Programming Model
- Integrated & Open Source

Google Cloud Dataflow use cases

Both batch and streaming data processing

ETL (extract, transform, load) approach

Excels and high volume computation

High parallelism factor (“*embarrassingly parallel*”)

Cost effective

Dataflow Programming model

- Designed to simplify the mechanics of large-scale data processing
- It creates an optimized job to be executed as a unit by one of the Cloud Dataflow runner services
- You can focus on the logical composition of your data processing job, rather than the physical orchestratrion of parallel processing

Google Cloud Dataflow components

Two main components:

- A **set of SDK** used to define data processing jobs:
  - **Unified programming model**. "*One fits all" approach*
  - **Data programming model (**pipelines, collection, transformation, sources and sinks**)**

- A **Google Cloud Platform managed service** that ties together with the Google Cloud Platform, Google Compute Engine, Google Cloud Storage, Big Query, ….

Dataflow SDKs

- Each pipeline is an indepedent entity that reads some input data, transform it, and generates some output data. **A pipeline represents a directed graph of data processing transformation**
- Simple data representation.
  - Specialized collections called **Pcollection**
  - Pcollection can represent unlimited size dataset
  - Pcollections are the inputs and the ouputs for each step in your pipeline
- Dataflow provides abstractions to manipulate data
  - Transformation over data are known as **Ptransform**
  - Transformations can be linear or not
- I/O APIs for a variety of data formats like text or Avro files, Big Query table, Google Pub/Sub, …
- Dataflow SDK for Java available on Github. https://github.com/GoogleCloudPlatform/DataflowJavaSDK

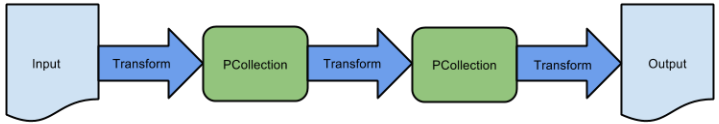
Pipeline Design Principles

Some question before building your Pipeline:

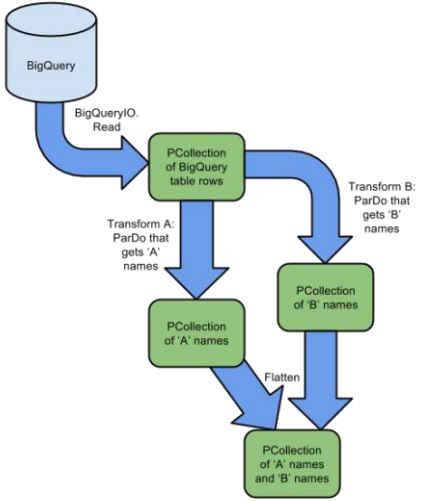
- Where is your input data stored? → **Read transformations**
- What does your data look like? → It defines your **Pcollections**
- What do you want to do with your data? → **Core or pre-written transforms**
- What does your output data look like, and where should it go? → **Write transformations**

Pipeline Shapes

Linear Pipeline



Branching Pipeline



Pipeline example

```
public static interface Options extends PipelineOptions {
  ...
}

public static void main(String[] args) {
  // Parse and validate command-line flags,
  // then create pipeline, passing it a user-defined Options object.
  Options options = PipelineOptionsFactory.fromArgs(args)
      .withValidation()
      .as(Options.class);

  Pipeline p = Pipeline.create(options);

  p.apply(TextIO.Read.from(input))   // SDK-provided PTransform for reading text data
   .apply(new CountWords())          // User-written subclass of PTransform for counting words
   .apply(TextIO.Write.to(output));  // SDK-provided PTransform for writing text data

  p.run();
}
```

Pipeline collections: **PCollections**

PCollection represents a potentially large, immutable “bag” of same-type elements

A PCollection can be of any type and it will be encoded based on the Dataflow SDK Data encoding or on your own.

PCollection requirements

- Immutable. Once created, you cannot add, remove or change individual objects.
- Does not support random access
- A PCollection belongs to one Pipeline (collections cannot be shared):

Bounded vs unbounded collections

- It depends on your source dataset.
- Bounded collections can be processed using batch jobs
- Unbounded collections must be processed using streaming jobs (Windowing and Timestamps)

Pipeline collections example

A collection created from individual lines of text

```
// Create a Java Collection, in this case a List of Strings.
  static final List<String> LINES = Arrays.asList(
      "To be, or not to be: that is the question: ",
      "Whether 'tis nobler in the mind to suffer ",
      "The slings and arrows of outrageous fortune, ",
      "Or to take arms against a sea of troubles, ");

  PipelineOptions options = PipelineOptionsFactory.create();
  Pipeline p = Pipeline.create(options);

  p.apply(Create.of(LINES)).setCoder(StringUtf8Coder.of())   // create the
PCollection
```

Pipeline collections types

Bounded PCollections. It represents a fixed data set from data sources/sinks as:

- TextIO
- BigQueryIO
- DataStoreIO
- Custom data sources using the Custom Source/Sink API

Unbounded PCollections. It represents a continuously updating data set, or streaming data sources/sinks as:

- PubsubIO
- BigQueryIO (only as a sink)

Each element in a **PCollection** has an associated timestamp. **NOTE**: it doesn't happen for all sources(e.g, TextIO)

Unbounded PCollections are processed as finite logical windows (**Windowing**). Windowing can also be applied to **Bounded PCollections** as a global window.

PCollections Windowing

Subdivide **PCollection** processing according to the timestamp.

Uses **Triggers** to determine when to close each finite window as unbounded data arrives.

Windowing functions

- Fixed Time Windows
- Sliding Time Windows. Two variables, windows size and period.
- Per-Session Windows. It relates to when actions are perform (e.g, mouse interactions)
- Single Global Window. By default window.
- Other windowing function as Calendar-based are found in com.google.cloud.dataflow.sdk.transforms.windowing

Time Skew, Data Lag, and Late Data. As each element is marked with a Timestamp it can be known if data arrives with some lag.

PCollections Windowing II

Adding Timestamp

```
PCollection<LogEntry> unstampedLogs = ...;
  PCollection<LogEntry> stampedLogs =
      unstampedLogs.apply(ParDo.of(new DoFn<LogEntry, LogEntry>() {
        public void processElement(ProcessContext c) {
          // Extract the timestamp from log entry we're currently
processing.
          Instant logTimeStamp =
extractTimeStampFromLogEntry(c.element());
          // Use outputWithTimestamp to emit the log entry with timestamp
attached.
          c.outputWithTimestamp(c.element(), logTimeStamp);
        }
      }));
```

PCollections Windowing II

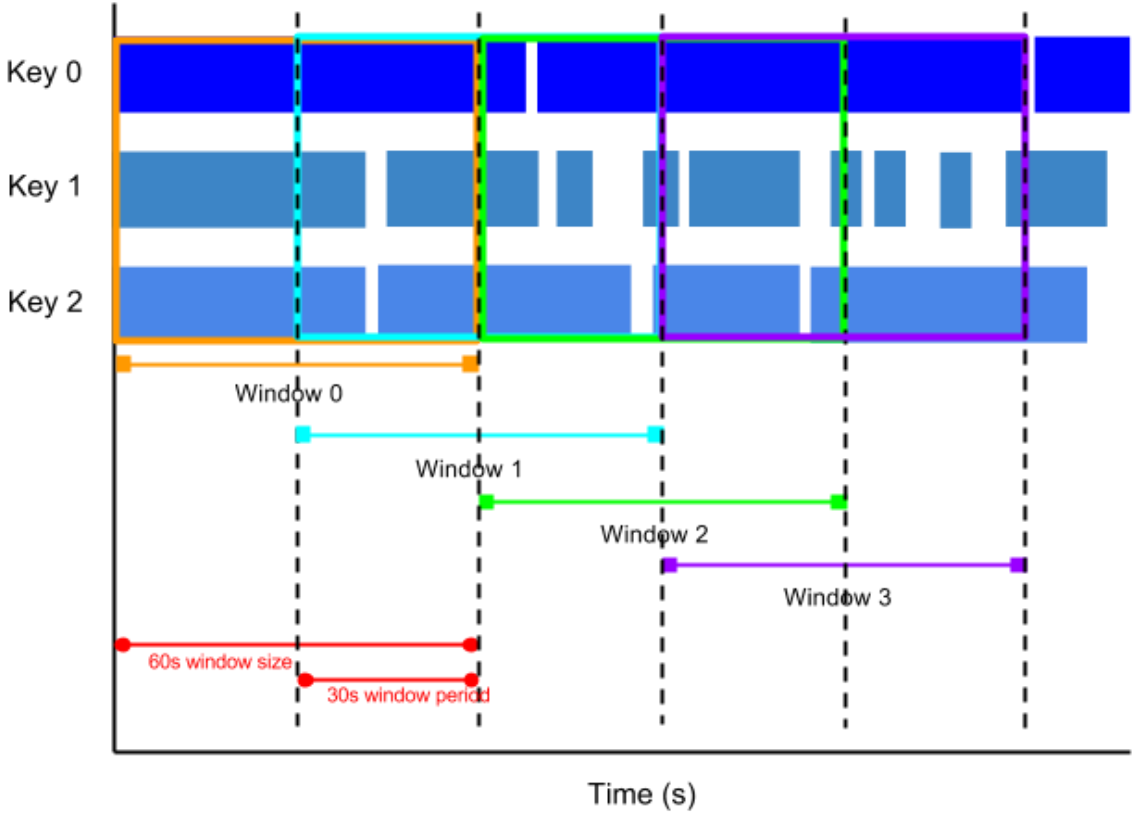
Time Skew and Late Data

```
PCollection<String> items = ...;
  PCollection<String> fixed_windowed_items = items.apply(
    Window.<String>into(FixedWindows.of(1, TimeUnit.MINUTES))
          .withAllowedLateness(Duration.standardDays(2)));
```

Sliding window

```
  PCollection<String> items = ...;
  PCollection<String> sliding_windowed_items = items.apply(
    Window.<String>into(SlidingWindows.of(Duration.standardMinutes(6
0)).every(Duration.standardSeconds(30))));
```

PCollection sliding Time Windows



Pipeline transforms: **PTransforms**

Represent a processing operation logic in a pipeline as a **function object.**
Processing operations

- Mathematical computations on data
- Converting data from one format to another
- Grouping data together
- Filtering data
- Combining data elements into single values

PTransforms requirements

- **Serializable**
- **Thread-compatible**. Functions are going to be accessed by a single thread on a worker instance.
- **Idempotent functions are recommended**: for any given input, a function provides the same ouput

How it works?. Call the apply method over the PCollection

Pipeline transforms types

Core transformation

- ParDo for generic parallel processing
- GroupByKey for Key-Grouping Key/Value pairs
- Combine for combining collections or grouped values
- Flatten for merging collections

Composite transform

- Built for multiple sub-transform in a modular way
- Examples, Count and Top composite transform.

Pre-Written Transform

- Proccessing logic as combining, splitting, manipulating and performing statistical analysis is already written.
- They are found in the com.google.cloud.dataflow.sdk.transforms package

Root Transforms for Reading and Writing Data

Pipeline transform example

A composite transform that count words

```
static class CountWords
    extends PTransform<PCollection<String>, PCollection<String>> {
  @Override
  public PCollection<String> apply(PCollection<String> lines) {
    PCollection<String> words = lines.apply(
        ParDo
        .named("ExtractWords")
        .of(new ExtractWordsFn()));

    PCollection<KV<String, Integer>> wordCounts =
        words.apply(Count.<String>perElement());

    PCollection<String> results = wordCounts.apply(
        ParDo
        .named("FormatCounts")
        .of(new DoFn<KV<String, Integer>, String>() {
            @Override
            public void processElement() {
              output(element().getKey() + ": " + element().getValue());
            }
          }));

    return results;
  }
}
```

Pipeline I/O

We need to read/write data from external sources like Google Cloud Storage or BigQuery table

Read/Write transforms are applied to sources to gather data

Read/Write data from Cloud Storage

```
p.apply(AvroIO.Read.named("ReadFromAvro")
        .from("gs://my_bucket/path/to/records-*.avro")
        .withSchema(schema));
```

```
records.apply(AvroIO.Write.named("WriteToAvro")
              .to("gs://my_bucket/path/to/numbers")
              .withSchema(schema)
              .withSuffix(".avro"));
```

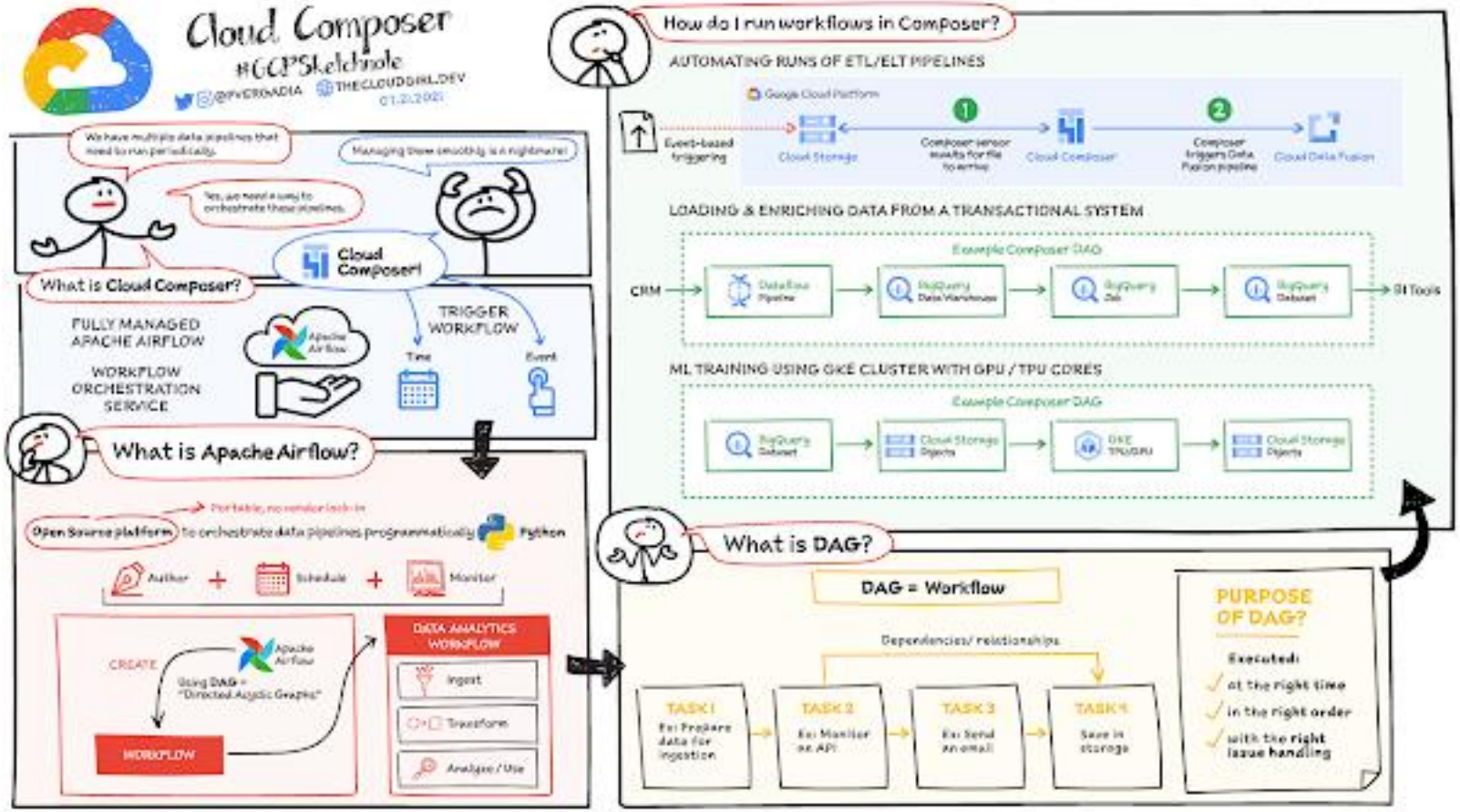
- Read and write Tranforms in the Dataflow SDKs
  - Text files
  - Big Query tables
  - Avro files
  - Pub/Sub
- Custom I/O sources and sink can be created.

Experiment –

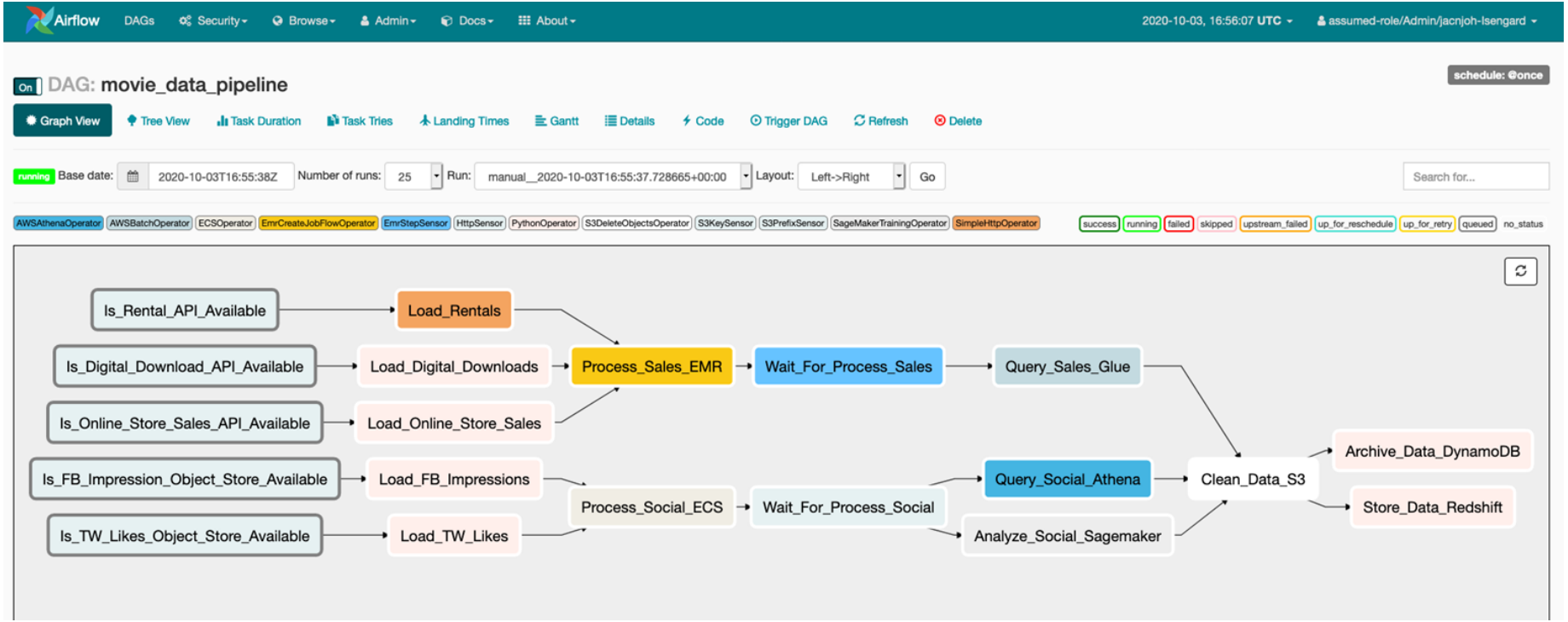
Cloud Composer

Apache Airflow

What is GCP Cloud Composer?



What is Apache Airflow?

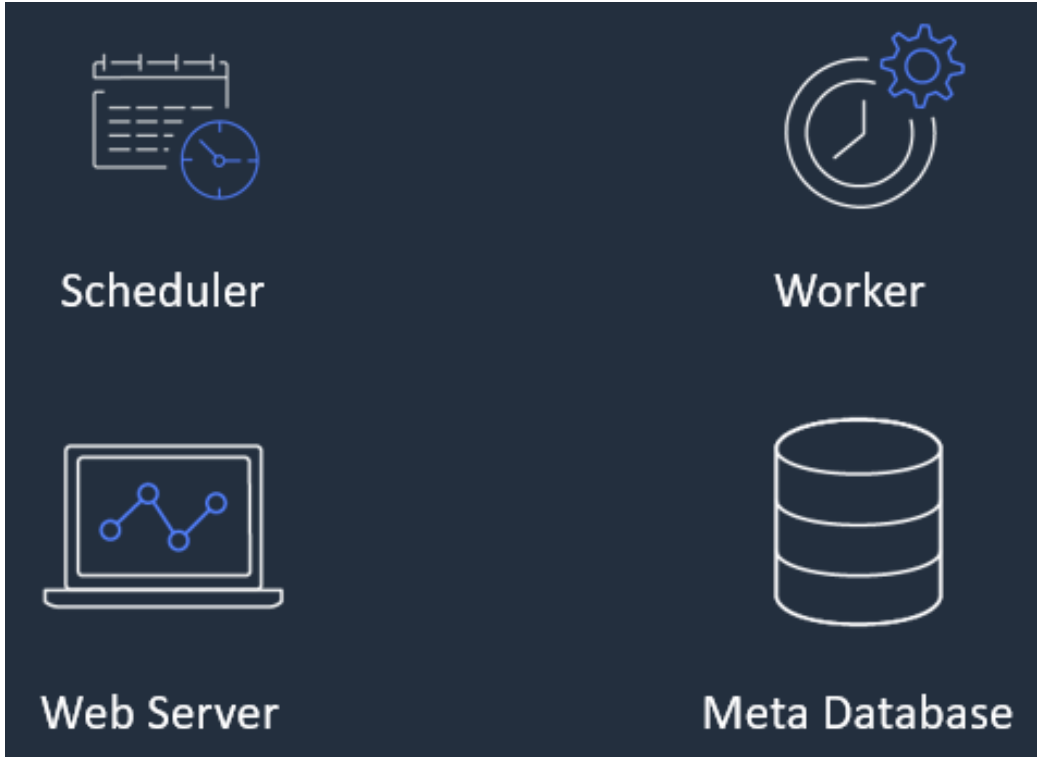


What is Apache Airflow?

- Apache Airflow is a workflow automation and scheduling system that can be used to author and manage data pipelines.
- Airflow uses workflows made of directed acyclic graphs (DAGs) of tasks.
- Open-source product running in GCP, Cloud Composer is a managed service for Apache Airflow.
- Can be installed locally, directly or in Docker, for local testing or development, if desired.
- Workflows are defined in python, but the steps themselves can be in preferred pattern, e.g. shell scripts, Python function, database function.

Apache Airflow Use Cases

Apache Airflow Components



Google Cloud Composer - GCP

GKE cluster: The Airflow schedulers, workers, and Redis Queue run as GKE workloads on a single cluster, and are responsible for processing and executing DAGs. The cluster also hosts other Cloud Composer components like Composer Agent and Airflow Monitoring, which help manage the Cloud Composer environment, gather logs to store in Cloud Logging, and gather metrics to upload to Cloud Monitoring.

Web server: The web server runs the Apache Airflow web interface, and Identity-Aware Proxy protects the interface. For more information, see Airflow Web Interface.

Database: The database holds the Apache Airflow metadata.

Cloud Storage bucket: Cloud Composer associates a Cloud Storage bucket with the environment. The associated bucket stores the DAGs, logs, custom plugins, and data for the environment.

Google Cloud Composer Versions

Cloud Composer has two major versions:

Cloud Composer 2. This version has autoscaling environments. Only Airflow 2.x

Cloud Composer 1. This version has manual scaling. Supports Airflow 1.x and 2.x

For our experiments we'll use Cloud Composer 1 environments on Airflow 2.x

Apache Airflow Component Workflow

- Airflow daemons are independent.
- Scheduler periodically polls to see if any DAGs which are registered need to be executed.
- Scheduler creates a new DagRun instance in the Metastore and starts to trigger the individual tasks in the DAG.
- Scheduler pushes messages into the queuing service.

Apache Airflow Vocabulary

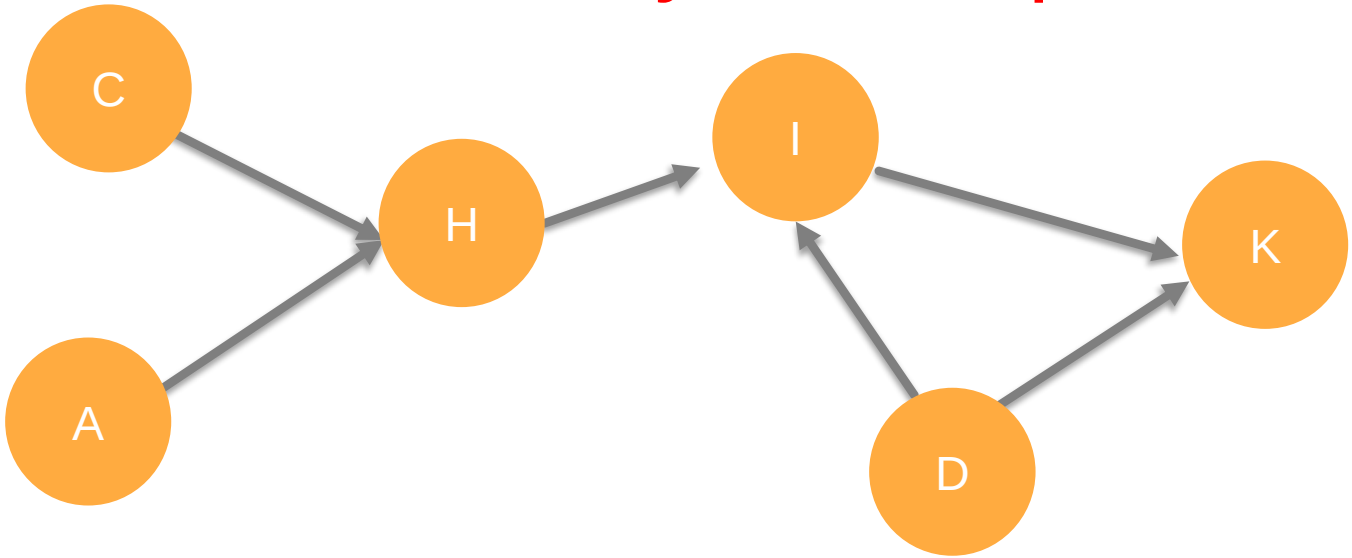
- DAG (Directed Acyclic Graph) – is a collection of all the tasks.
- DAG Runs – physical instance of a DAG
- Tasks – unit of work within a DAG
- Operators – determine what actually gets done by a task.
- Sensors – special purpose operations that act as triggers
- Hooks – interfaces to external sinks and sources
- Workflows – combinations of the above components executed at a certain timestamp (execution_date) with a task instance

Apache Airflow Workflow

DAG	Task	Operators and Sensors
Every workflow is defined as a directed acyclic graph.	A Task is a specific action or operation to be performed	Operators inform something to be performed (e.g. send email)
A DAG defines a series of tasks to perform something	For example, query a Sales table OR exporting data.	Sensor, trigger operation to be performed (e.g. when file exists)
The task can have directional dependencies.	The operators and sensors initiate the work	

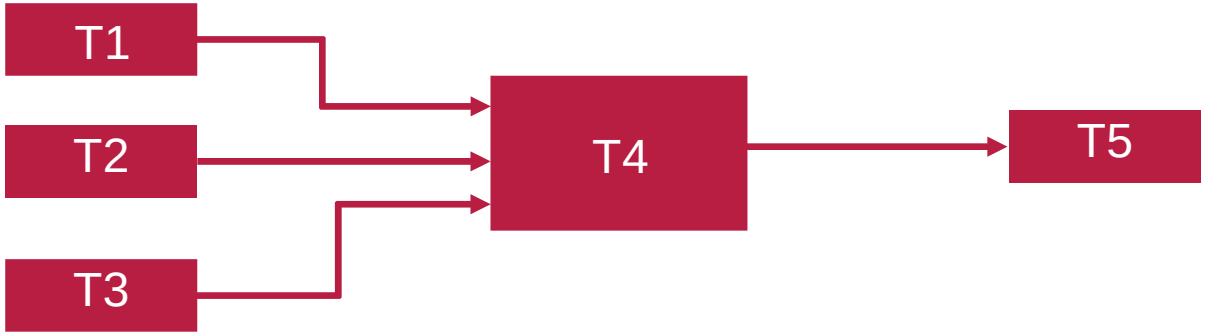
Apache Airflow Complexity

DAG – Directed Acyclic Graph



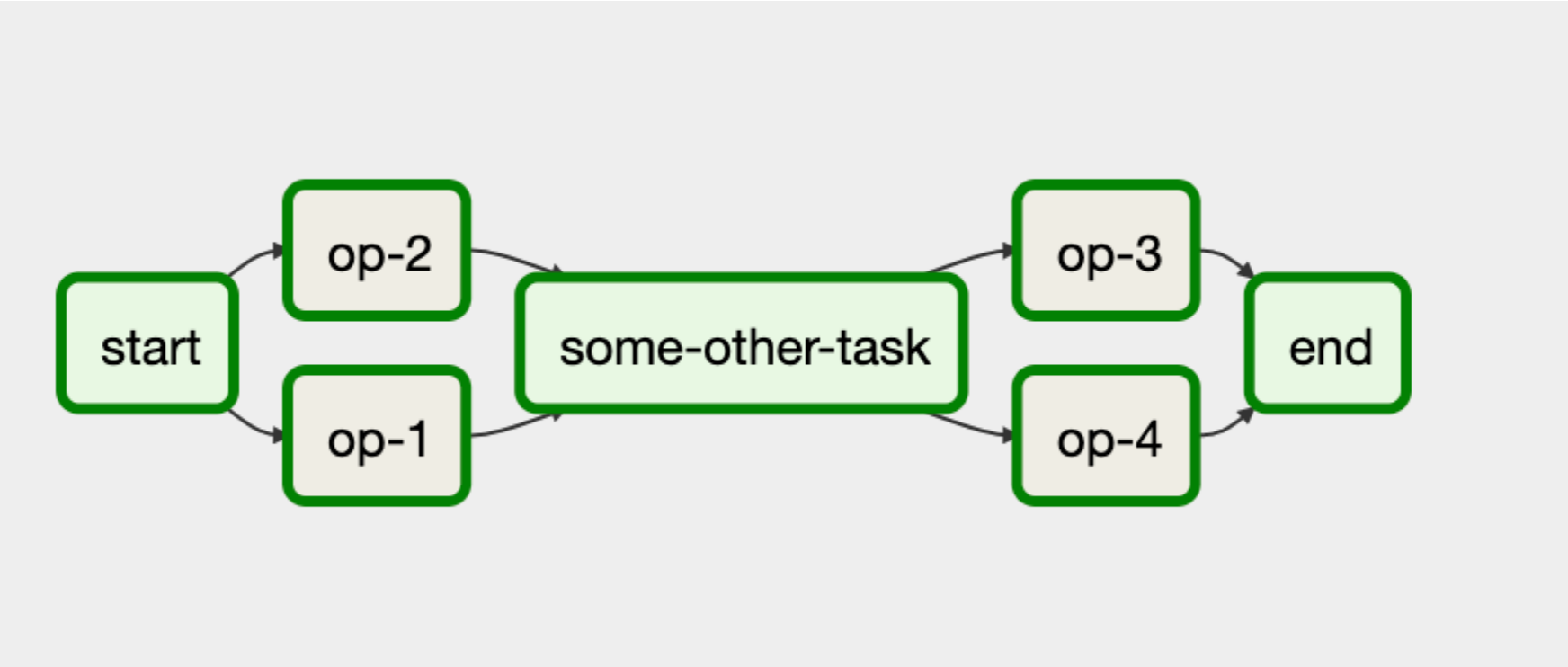
In mathematics, particularly graph theory, and computer science, a directed acyclic graph is a directed graph. The graph consists of vertices and edges, with each edge directed from one vertex to another, such that following those directions will never form a cyclic redundancy.

DAG – Directed Acyclic Graph

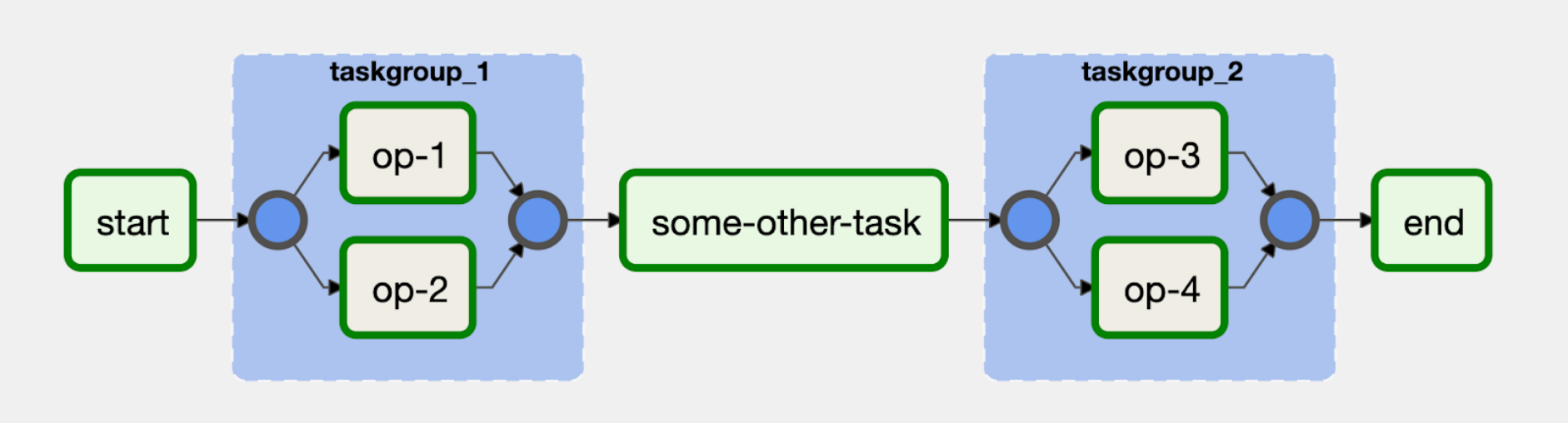


In Airflow all **workflows are DAGs**. A Dag consists of ***operators***. An operator defines an individual task that needs to be performed.There are different types of operators available. A DAG is **defined in a Python** script, which represents the DAGs structure (tasks and their dependencies) as code.

Tasks in a DAG



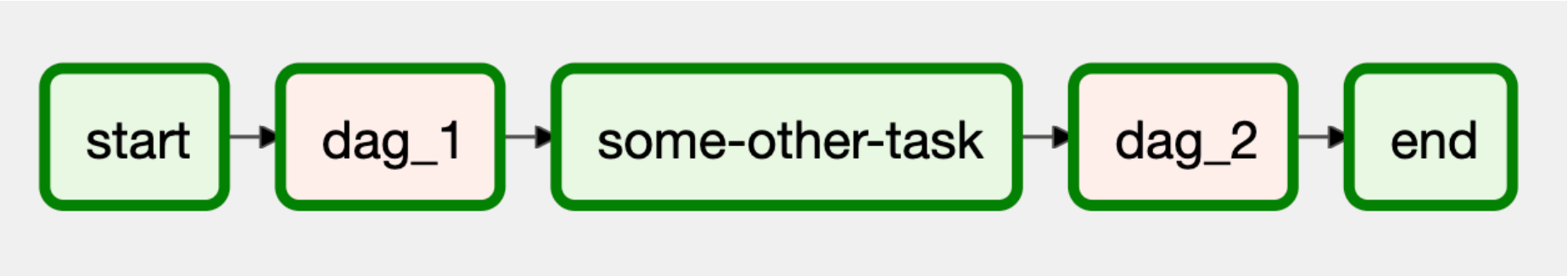
Grouping Tasks



In Airflow 2, we can use the TaskGroup operator to group tasks together in a DAG.

Tasks defined within a TaskGroup block are still part of the main DAG.

Triggering SubDAGs



For Airflow 1.10.*, use the airflow.operators.dagrun_operator module.

For Airflow 2, use the airflow.operators.trigger_dagrun module.

Airflow Operators

BashOperator - executes a bash command.

PythonOperator - calls an arbitrary Python function

EmailOperator - sends an email

SimpleHttpOperator - sends an HTTP request

Google Cloud operators

Database operators: MySqlOperator, SqliteOperator, PostgresOperator, MsSqlOperator, etc

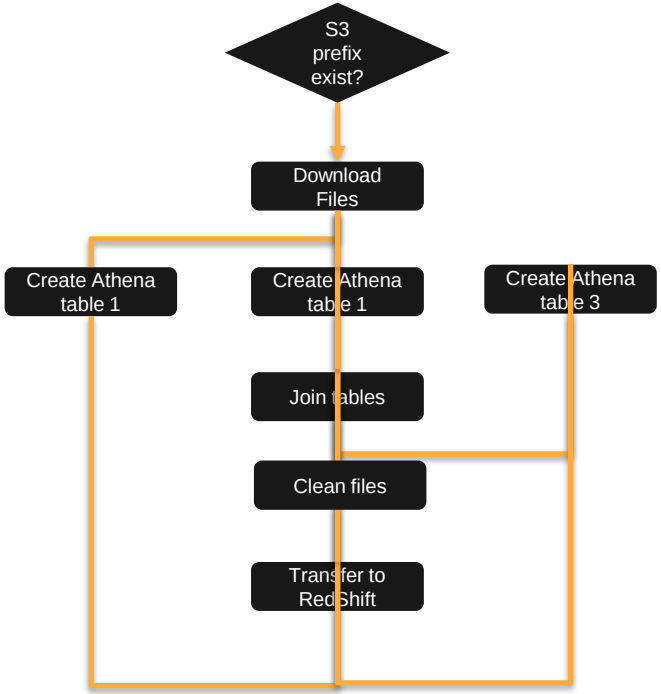
Airflow Sensors

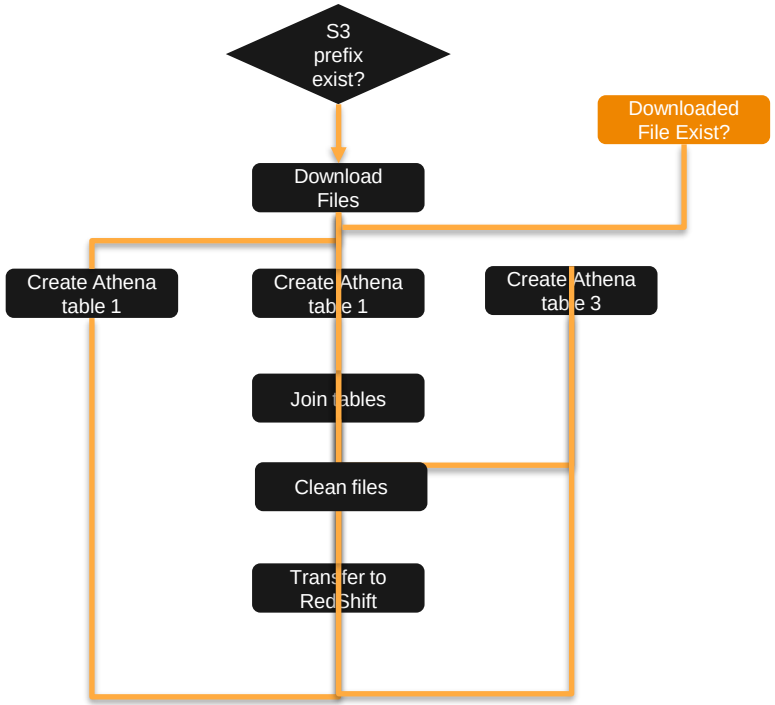
Sensors are a special type of Operator that are designed to do exactly one thing - wait for something to occur.

Sensors have three different modes of running so you can be a bit more efficient about using them:

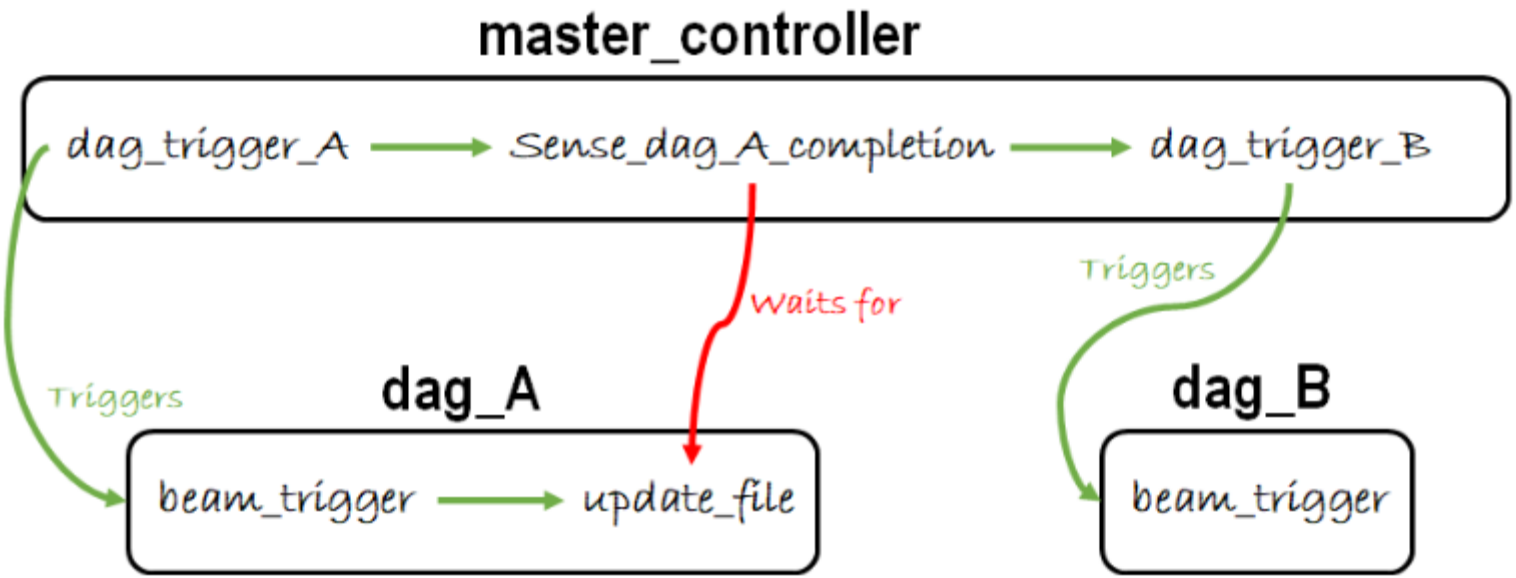
- **poke** (default): The Sensor takes up a worker slot for its entire runtime
- **reschedule**: The Sensor takes up a worker slot only when it is checking, and sleeps for a set duration between checks
- **smart sensor**: There is a single centralized version of a Sensor that batches all executions for this Sensor

More than one upstream tasks

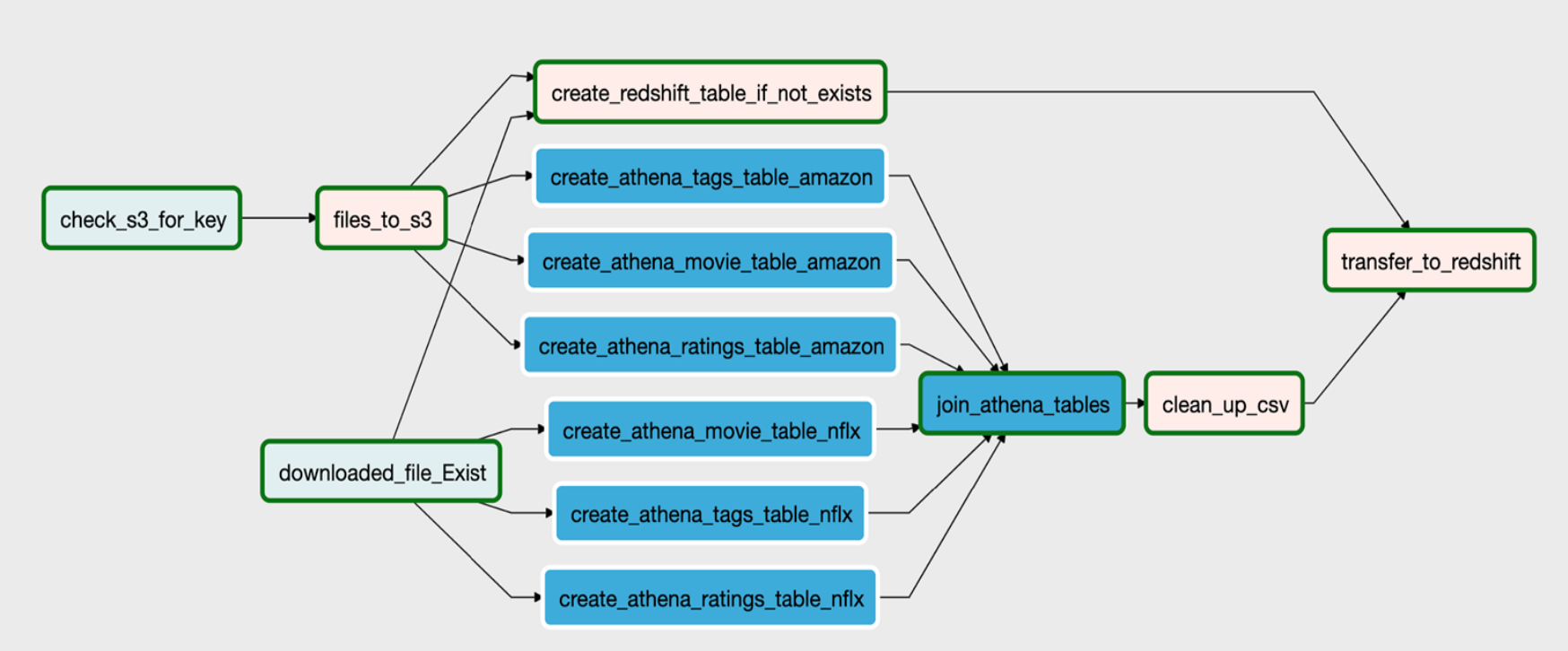




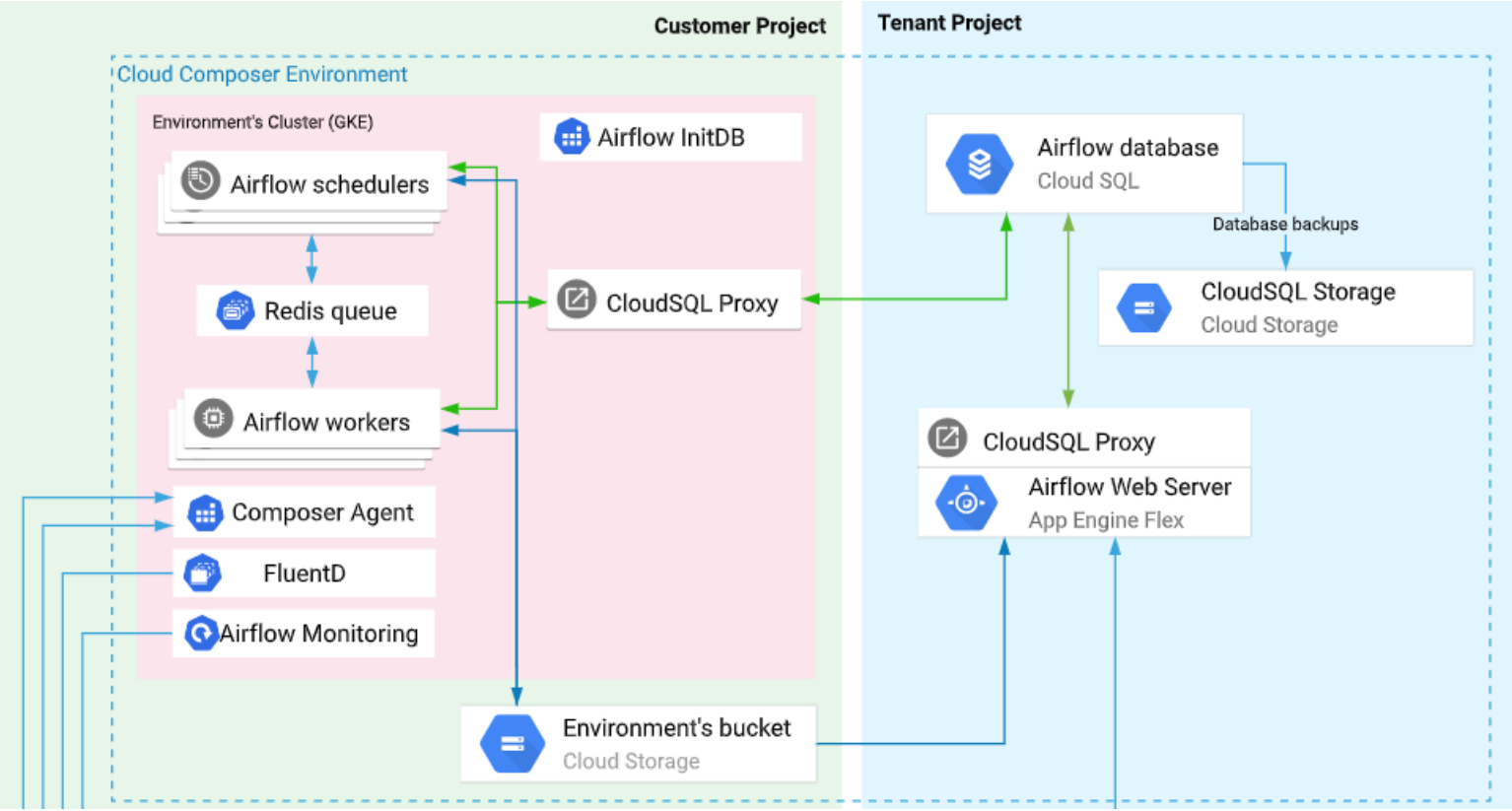
Cloud Composer Orchestration



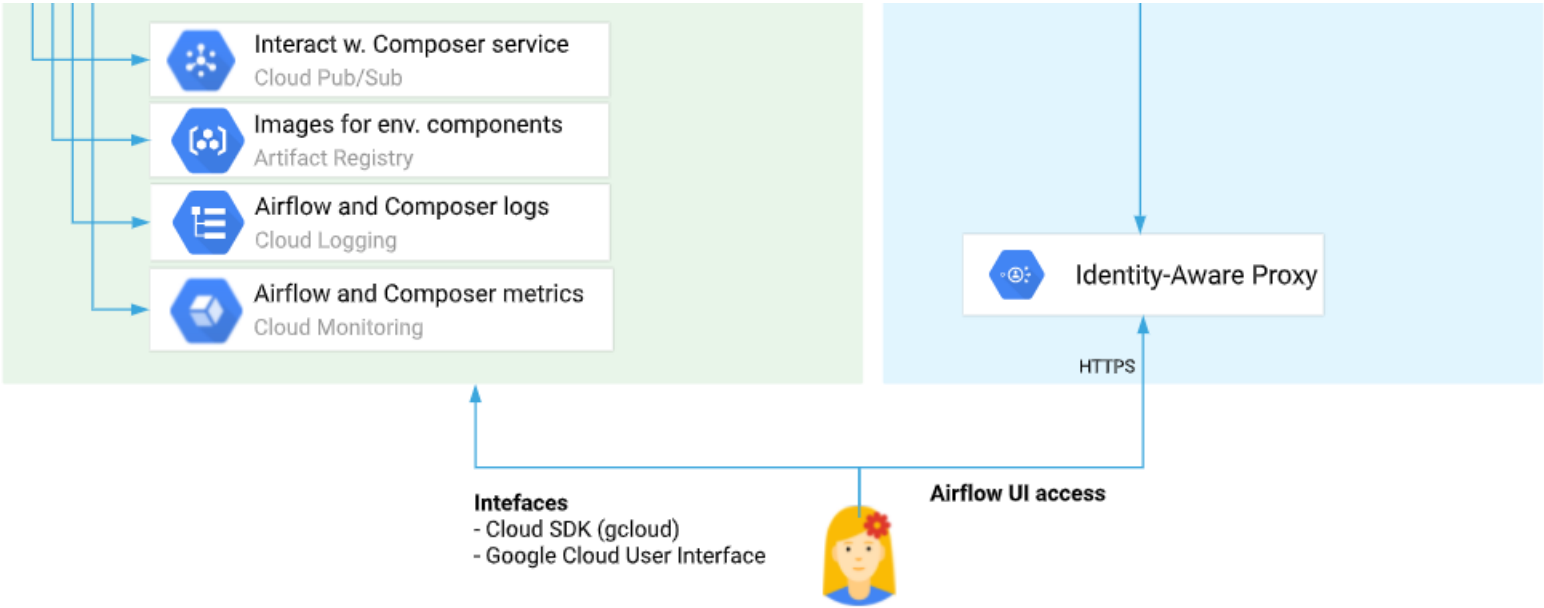
More than one upstream tasks



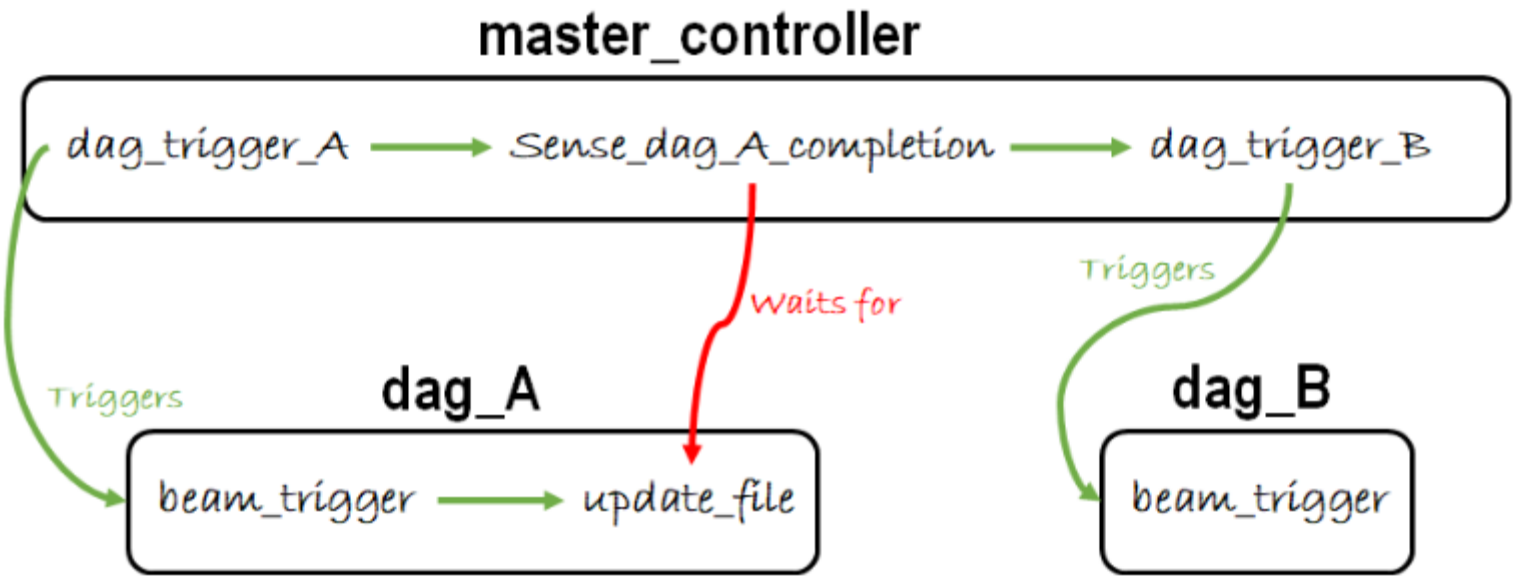
Google Cloud Composer Architecture



Google Cloud Composer Architecture



Cloud Composer Orchestration



POP QUIZ:

Kubernetes

Which of the following commands is used to create Kubernetes service?

A: Kubectl run

B: Kubectl deploy

C:: Kubectl expose

D: Java spring framework

© 2021 by Innovation In Software Corporation

POP QUIZ:

Kubernetes

Hich of the following commands is used to create Kubernetes service?

A: Kubectl run

B: Kubectl deploy

C:: Kubectl expose

D: Java spring framework

© 2021 by Innovation In Software Corporation

Experiment –

Data Pipelines with Beam

What is Apache Beam?

Unified
Use a single programming model for both batch and streaming use cases.

Extensible
Write and share new SDKs, IO connectors, and transformation libraries.

Portable
Execute pipelines on multiple execution environments.

Open Source
Community-based development and support to help evolve your application and use cases.

Why Apache Beam? Portability

Apache Beam is portable on several layers, namely:

- Portability of Apache Beam Pipelines between multiple **Runners** (by Runner we mean a technology, that actually executes distributed computation described by a Pipeline's author).
- Portability of Data Processing Model of Apache Beam to various programming languages.
- Portability of Data Processing logic between Bounded and Unbounded Data.

Runner Portability

- Run existing Pipelines against a data processing engine that can be chosen at runtime.
- Runner pipelines can be written in one of the supported programming languages, e.g. Python, Go, Scala or SQL.
- Runner could be Apache Flink, Apache Spark, or Google Dataflow.
- New Runners can be created as new technologies arise.

Model Portability

- Ability to provide support for multiple SDKs, independently of which language, or what technology a particular Runner uses.
- For example, one can code Beam Pipelines in Go language and run these against Apache Flink Runner, written in Java.

Data Processing Portability

- Bounded data is what was historically called batch processing
- Unbounded data represents real-time processing, a reliable application processing data as it arrives into the system, with low latency

Coding a Pipeline

Coding a Bounded Pipeline

- Identify the input
- Create the Pipeline
- Load data into a PCollection
- Use PTransform to do something with our data (transform)
- Configure Runner or use default
- Run the Pipeline
- Verify/validate the pipeline output/transformations

Programmatic Beam

- To use Beam, you create a driver program using the classes in one of the Beam SDK, i.e. Java, Python, or Go. Your driver program defines your pipeline, including all of the inputs, transforms, and outputs; it also sets execution options for your pipeline, including the Pipeline Runner, which determines what back-end your pipeline will run on.

- The Beam SDKs provide a number of abstractions that simplify the mechanics of large-scale distributed data processing. The same Beam abstractions work with both batch and streaming data sources.

Apache Beam in Context

Apache Beam Abstractions

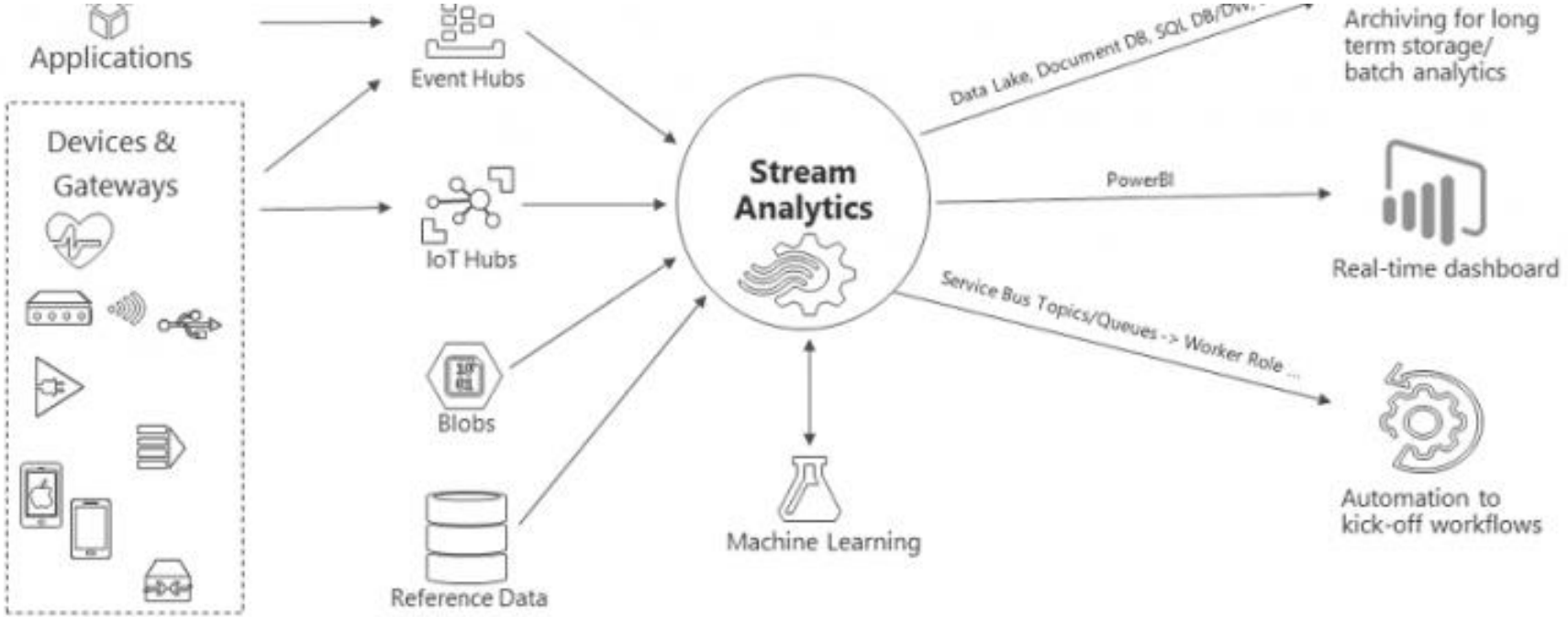
Pipelines - Encapsulated process of reading data from external source, transforming it and saving the output into some external storage sink.

Pcollections -Define the bounded or unbounded data on which pipeline operates. Pcollections are immutable objects created from any external system or in-memory data, and contain data of the same type.

Ptransforms - It takes Pcollection (input set of data) as an input, applies some processing function on it and produces another Pcollection (output set of data).

Pipeline IO - It enables you to read or write data from/to various external storage systems.

Sensor Data Application Processing



Time Series (Events)

Each element in a data set AKA PCollection has an associated intrinsic timestamp. The timestamp for each element is initially assigned by the Source that creates the data set.

Timestamps are useful for a data sets that contains elements with an inherent notion of time. If your pipeline is reading a stream of events, like Tweets or other social media messages, each element might use the time the event was posted as the element timestamp.

Data Timing

- In the real world we have multiple timings for data.
- The time at which we *process* our data and time at which the data *originated*.
- We usually call these two the **processing time** and the **event time**.
- The first one is the time we see on our clock when an event arrived to us, while the other is the time at which the event occurred.

Data Timing

Time	*Star Wars* episodes					
Processing	*Episode IV*	*Episode V*	*Episode VI*	*Episode I*	*Episode II*	*Episode III*
Event	*Episode I*	*Episode II*	*Episode III*	*Episode IV*	*Episode V*	*Episode VI*

Watermark

- Watermark is a (heuristic) algorithm, that gives us an estimation of how far we have got in the event time domain.
- A perfect watermark gives the highest possible time T, that guarantees that no further data arrives with event time < T.

Stateful Streams

- Every computation on a data stream which takes into account more than a single isolated event needs a state. The state holds (accumulates) values derived from the so-far processed stream elements.
- Long-running applications face disruptions caused by failing hardware, software, or necessary upgrades of the application itself.
- Every stream processing application has to be fault-tolerant by design.
- Ensuring fault-tolerance puts specific requirements on the state and on the stream itself.

Triggers

Capturing and emitting event state can required different triggers

- Processing-time:In fixed periods of processing time (for instance, every 5 seconds).
- Event-time: When the watermark reaches a certain time.
- Data-driven: When a specific condition is met in the data.

Timers

- **Event-time** and **Processing-time** triggers require an additional stream processing technique. This concept is a timer.
- A timer is a tool that lets an application specify a moment in either processing time or event time domain and, when that moment is reached, an **application-defined callback hook** is called.

Window

A window is a specific bounded range of data within a data stream. Beam has several types of pre-defined window functions, namely:

- Tumbling windows
- Sliding windows
- Session windows

Tumbling Window

Tumbling windows can each have exactly the same fixed length (for instance 1 hour or 1 day – these are called Fixed Windows), or different length (for example 1 month in case of Calendar Windows).

The common property is that event time of each element can be assigned to exactly one window, and that these windows cover a continuous (possibly) infinite time range without any gaps

Sliding Window

Sliding windows are windows that assign data elements into multiple windows, shifted by a time period, called a slide.

Sliding windows have the same fixed window length (for instance 1 hour) and the same fixed slide (for instance 10 minutes).

A sliding window of 1 hour with slide of 10 minutes assigns each event time into 6 distinct windows, each shifted by 10 minutes.

Session Window

Unlike tumbling and sliding, session windows are key unaligned.

Neither tumbling nor sliding windows depend on the data itself: each data element is assigned to a window (or several windows) based solely on element's timestamp.

Session windows split the stream into independent "sub-streams" based on a user provided key for each element in the stream. We can imagine the key as a color representing each stream element.

Global Window

This very special type of window assigns all data elements into a single window, regardless of their timestamp.

The window therefore spans complete time interval

from -infinity to +infinity.

This window is used as a default window before any other window is applied.

Allowed Lateness

This defines a timeout (in event time) after which the state in a window can be cleared and all remaining data can be cleared.

If we were to keep every window forever we would quickly run our application out of memory.

Without this measurement we could have data that is dropped or could produce an incorrect outcome.

Unifying Batch and Stream

Apache Beam is the first model to unify both of these evolving paths into a single model, targeted at both low latency and advanced programming models.

This was enabled by a simple, but very crucial insight: batch batch is a special case of streaming.

Code your Pipeline as a streaming Pipeline then we can run it in either batch and streaming modes.

Wrap Up

Apache Beam is the first model to unify both of these evolving paths into a single model, targeted at both low latency and advanced programming models.

This was enabled by a simple, but very crucial insight: batch batch is a special case of streaming.

Code your Pipeline as a streaming Pipeline then we can run it in either batch and streaming modes.

Beam SQL

Beam SQL allows a Beam user (currently only available in Beam Java and Python) to query bounded and unbounded PCollections with SQL statements.

Your SQL query is translated to a PTransform, an encapsulated segment of a Beam pipeline. You can freely mix SQL PTransforms and other PTransforms in your pipeline.

Beam SQL includes the following dialects:

- Beam Calcite SQL
- Beam ZetaSQL

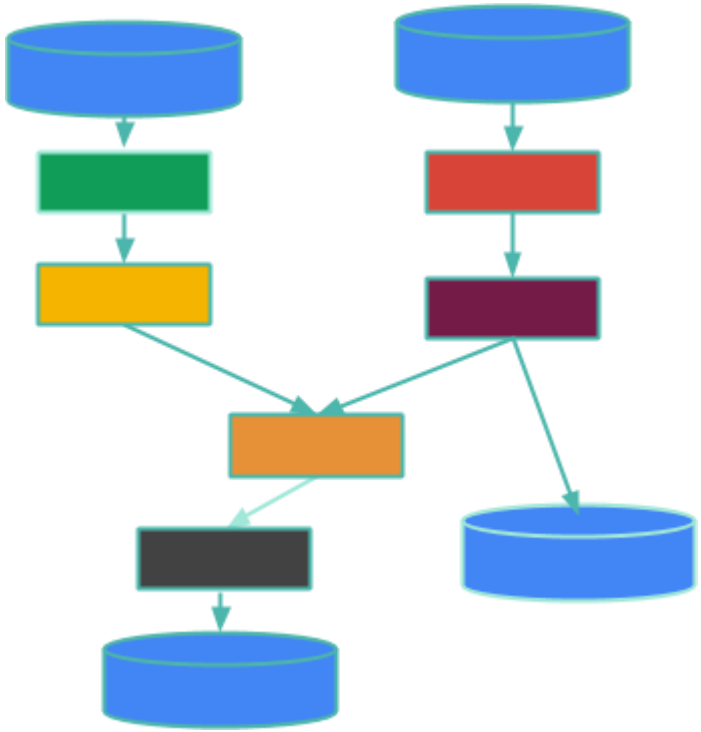
Beam SQL Dialects

Beam Calcite SQL is a variant of Apache Calcite, a dialect widespread in big data processing. Beam ZetaSQL is more compatible with BigQuery, so it’s especially useful in pipelines that write to or read from BigQuery tables.

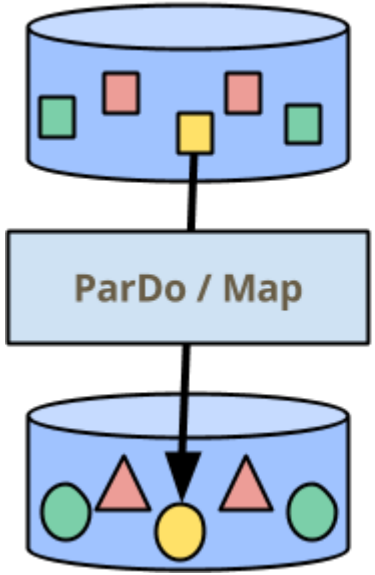
There are two additional concepts you need to know to use SQL in your pipeline:

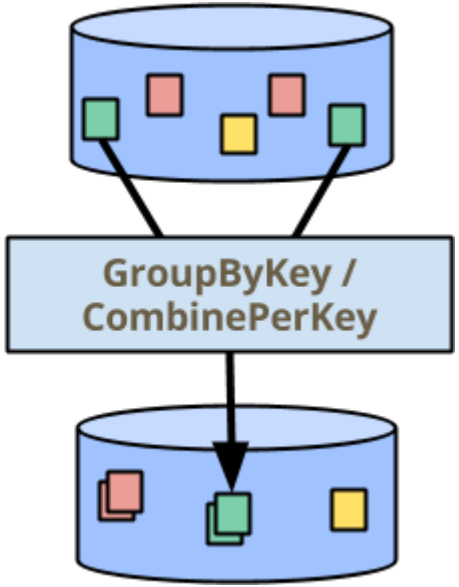
- SqlTransform: the interface for creating PTransforms from SQL queries.
- Row: the type of elements that Beam SQL operates on. A PCollection<Row> plays the role of a table.

Beam Processing DAG

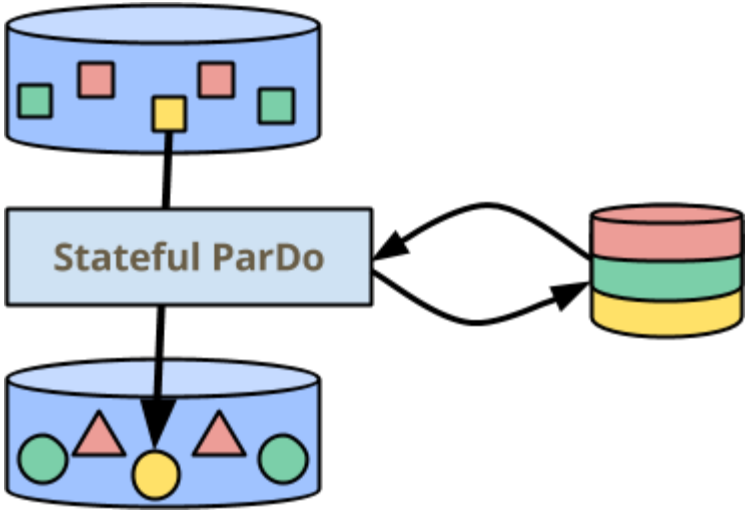


Beam Building Blocks

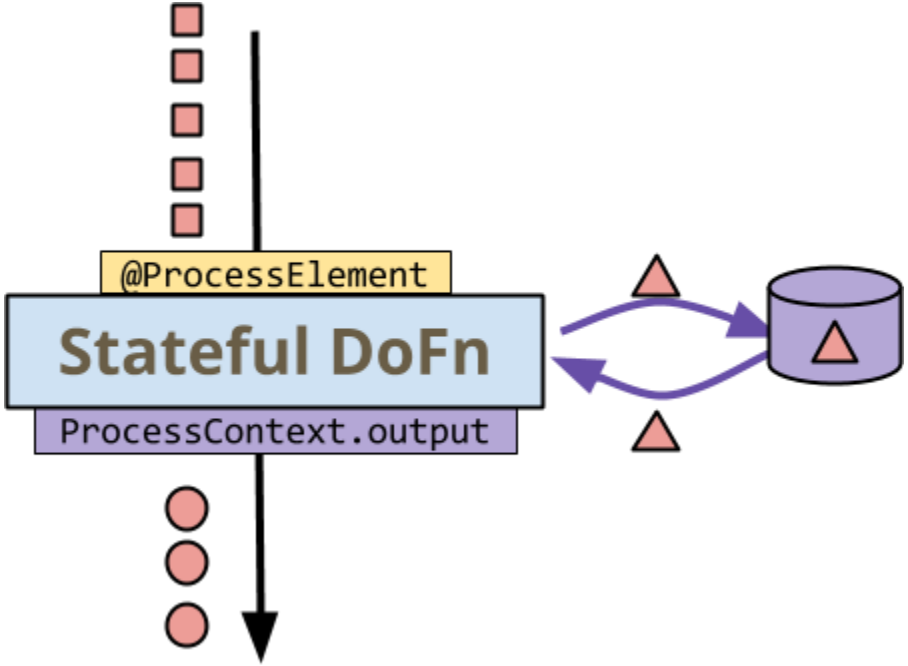




Stateful Pardo AKA Per Element Mutable State Beam



Stateful Processing in Action



Stateful Processing Example Requirements

- An Airport wants to have a dashboard that shows the status of all the rideshare/taxi drivers operating onsite with no more than 1 minute stale data.
- The company wants to know how many drivers are *available, busy,* or *idle.*
- This data would be represented as a bar chart that changes value with respect to how many taxis are in what state.

Stateful Processing Considerations

Source input – how is the input data provided e.g. ordered, unordered

Work distribution – which worker responsible for each data key

Data lateness – when is data no longer useful

Data lateness becomes very important in our requirements since we don’t want any driver to have the **wrong status skewing our actionable insight**.

Building a Stateful Pipeline

Stage 1: Reading from a Pub/Sub Subscription & Setting User Defined KeysWork distribution.

Stage 2: Stateful DoFn and Windowing.

Stage 3: Stateful Accumulation and Publishing Output to Cloud Pub/Sub

Building a Stateful Pipeline

Stage 1: Reading from a Pub/Sub Subscription & Setting User Defined KeysWork distribution.

Stage 2: Stateful DoFn and Windowing.

Stage 3: Stateful Accumulation and Publishing Output to Cloud Pub/Sub

Beam Code: Define your Pipeline

Java

```
// Start by defining the options for the pipeline.
PipelineOptions options = PipelineOptionsFactory.create();

// Then create the pipeline.
Pipeline p = Pipeline.create(options);
```

Python

```
import apache_beam as beam

with beam.Pipeline() as pipeline:
  pass  # build your pipeline here
```

Beam Code: Configuring Options

Java

```
PipelineOptions options =

PipelineOptionsFactory.fromArgs(arg
s).withValidation().create();

//This interprets command-line
arguments that follow the format:

--<option>=<value>
```

Python

```
from
apache_beam.options.pipeline_option
s import PipelineOptions

beam_options = PipelineOptions()

#This interprets command-line
arguments that follow the format:

--<option>=<value>
```

Beam Code: Reading a File

Java

```
public static void main(String[] args) {
    // Create the pipeline.
    // Create the PCollection 'lines' by applying a 'Read' transform.
    PCollection<String> lines = p.apply(
      "ReadMyFile", TextIO.read().from("gs://some/inputData.txt"));
}
```

Python

```
lines = pipeline | 'ReadMyFile' >> beam.io.ReadFromText(
    'gs://some/inputData.txt')
```

Beam Code: Using In-memory Data

Java

```
public static void main(String[] args) {
    // Create a List of Strings AKA Collection
    final List<String> LINES = Arrays.asList(
      "To be, or not to be: that is the question: ", "Whether 'tis nobler in the mind to suffer ");
    // Create the pipeline.
    // Apply Create, passing the list and the coder, to create the PCollection.
p.apply(Create.of(LINES)).setCoder(StringUtf8Coder.of());
}
```

Python

```
import apache_beam as beam

with beam.Pipeline() as pipeline:
  lines = (
      pipeline | beam.Create([
        'To be, or not to be: that is the question: ',
        "Whether 'tis nobler in the mind to suffer ",
      ]))
```

Beam Code: Apply Transforms

Java

[Output PCollection] = [Input PCollection].apply([Transform])

[Final Output PCollection] = [Initial Input PCollection].apply([First Transform])

.apply([Second Transform])

.apply([Third Transform])

Python

[Output PCollection] = [Input PCollection] | [Transform]

[Final Output PCollection] = ([Initial Input PCollection] | [First Transform]

| [Second Transform]

| [Third Transform])

Beam Code: Branching Pipeline

Java

[PCollection of database table rows] = [Database Table Reader].apply([Read Transform])

[PCollection of 'A' names] = [PCollection of database table rows].apply([Transform A])

[PCollection of 'B' names] = [PCollection of database table rows].apply([Transform B])

Python

[PCollection of database table rows] = [Database Table Reader] | [Read Transform]

[PCollection of 'A' names] = [PCollection of database table rows] | [Transform A]

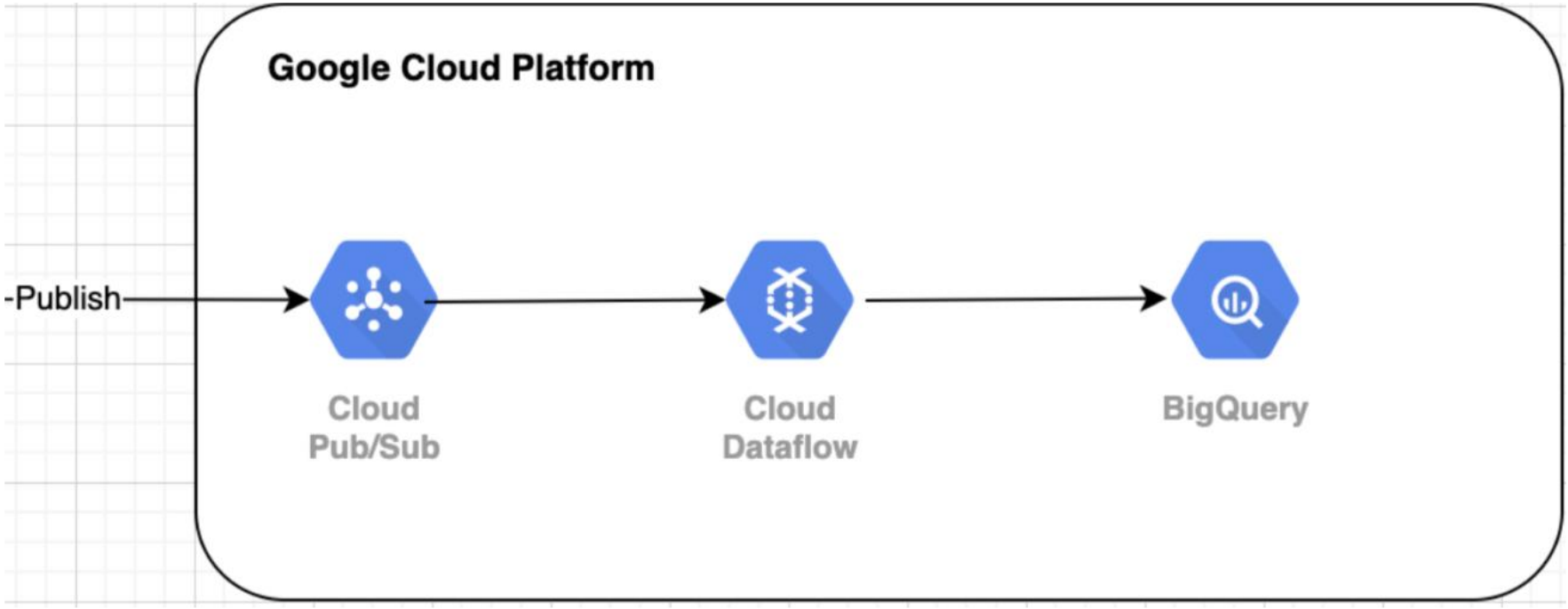
[PCollection of 'B' names] = [PCollection of database table rows] | [Transform B]

Serverless Messaging in Pipelines

Simple Data Pipelines in GCP

- Cloud Function: Our code will obtain data from an API/service and will load it into BigQuery, being subscribed to our execution topic of Pub/Sub.
- Cloud Pub/Sub: Our execution topic that will trigger our Cloud Function.
- BigQuery: The data obtained by the Cloud Function will be loaded here. It is our analytics engine, allowing us making simple or complex queries and data processes.

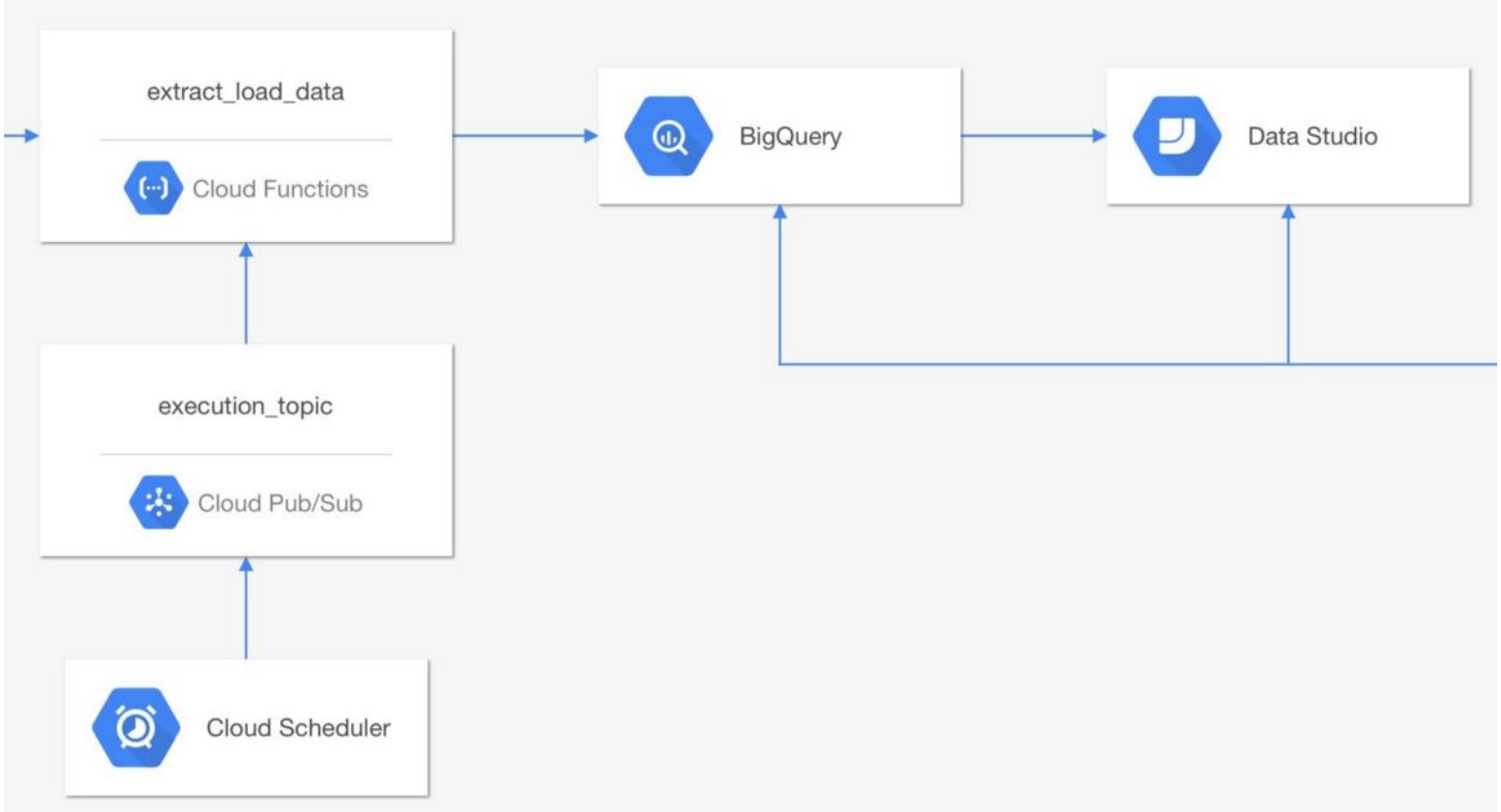
Simple Data Pipelines in GCP



Serverless in Data Pipelines in GCP

- Cloud Function: Our code will obtain data from an API/service and will load it into BigQuery, being subscribed to our execution topic of Pub/Sub.
- Cloud Pub/Sub: Our execution topic that will trigger our Cloud Function.
- Cloud Scheduler: This service will allow us to schedule the execution of our Cloud Function, invoking the Cloud Pub/Sub trigger.
- BigQuery: The data obtained by the Cloud Function will be loaded here. It is our analytics engine, allowing us making simple or complex queries and data processes.

Serverless in Data Pipelines in GCP



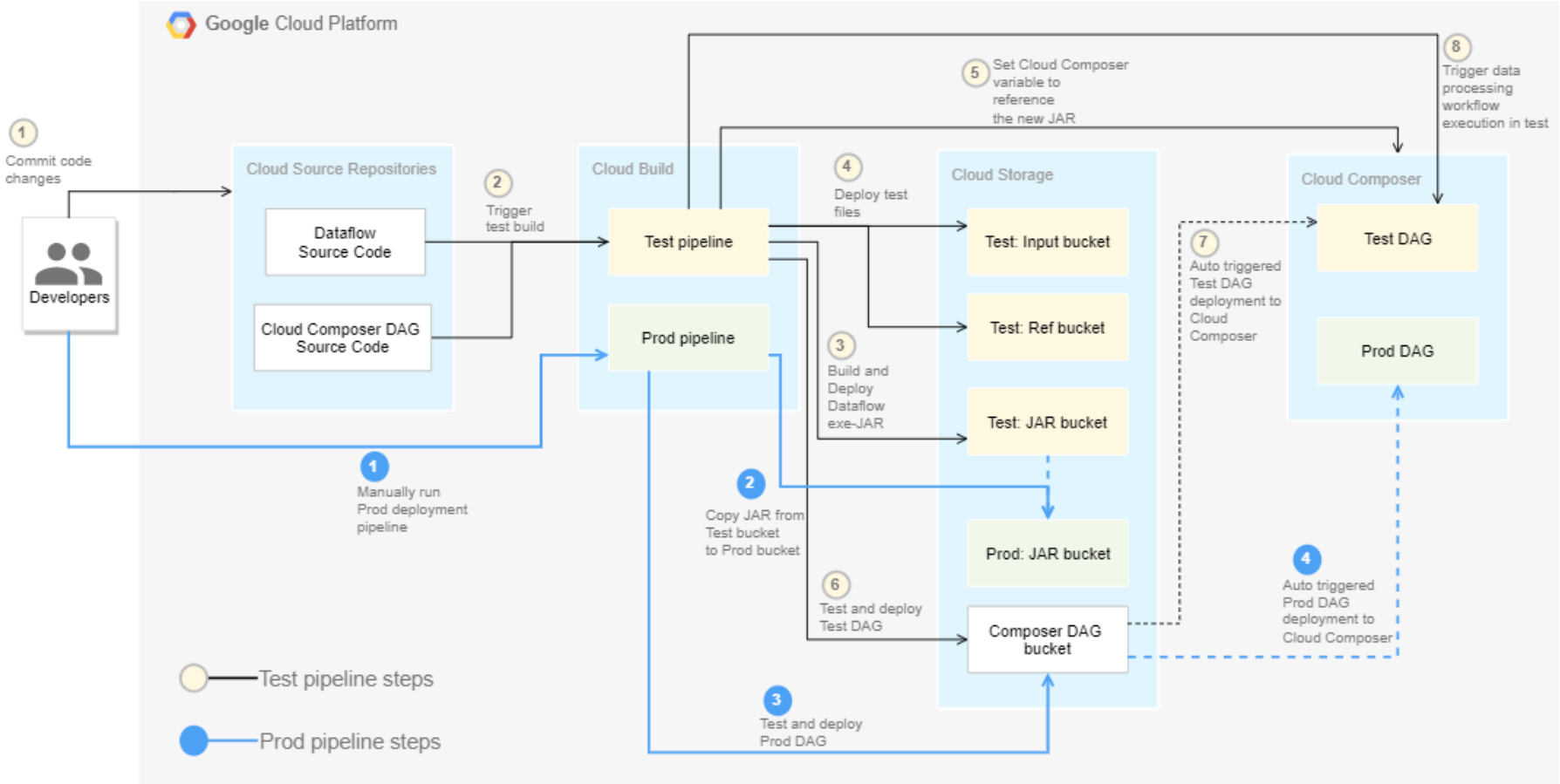
GCP Serverless Pre-reqs

- A GCP project with a linked billing account. The lower the frequency of execution, the lower the cost of the pipeline.
- Installed and initialized the Google Cloud SDK.
- An App Engine app activated in your project.
- Enabled the Cloud Functions, Cloud Scheduler, and APP Engine APIs.
- An API key from your data source.
- Unix based command line interpreter, e.g. the Cloud Shell on GCP

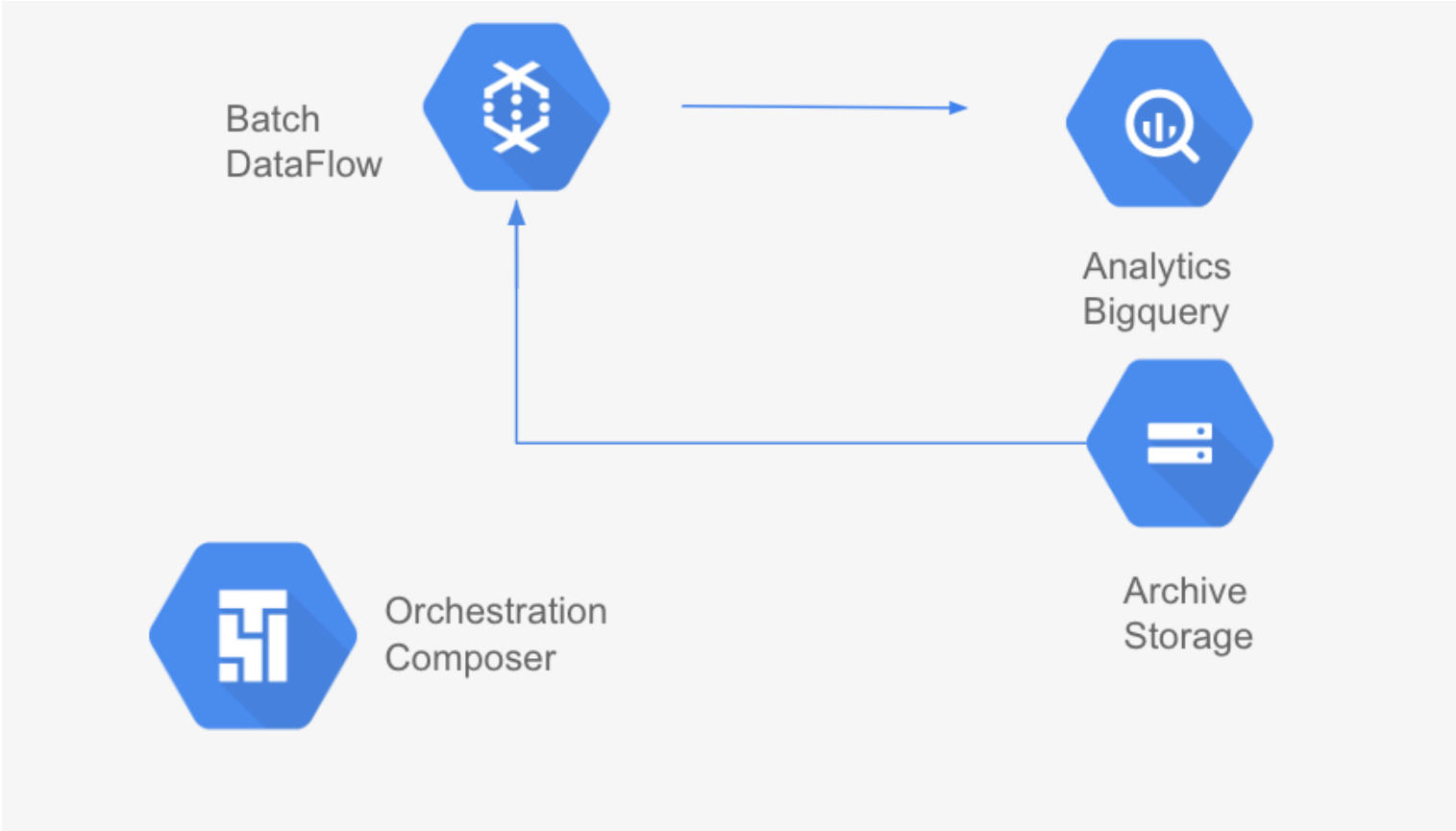
CI-CD Data Processing Pipeline

- Cloud Build to create a CI/CD pipeline for building, deploying, and testing a data-processing workflow, and the data processing itself. Cloud Build is a managed service that runs your build on Google Cloud.
- Cloud Composer to define and run the steps of the workflow, such as starting the data processing, testing and verifying results. Cloud Composer is a managed Apache Airflow service, which offers an environment where you can create, schedule, monitor, and manage complex workflows.
- Dataflow to run our data processing workflows. Unified stream and batch data processing that's serverless, fast, and cost-effective.

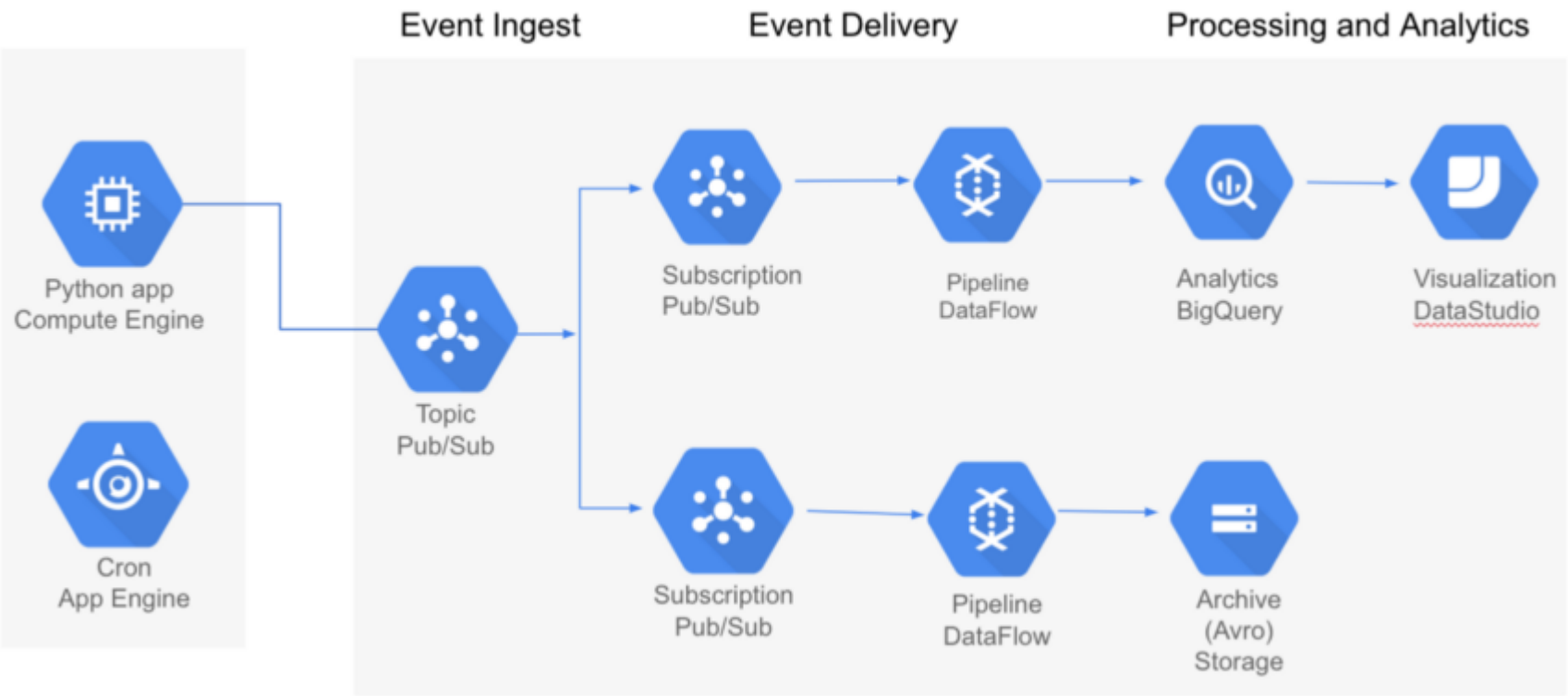
CI-CD Serverless Data Pipeline



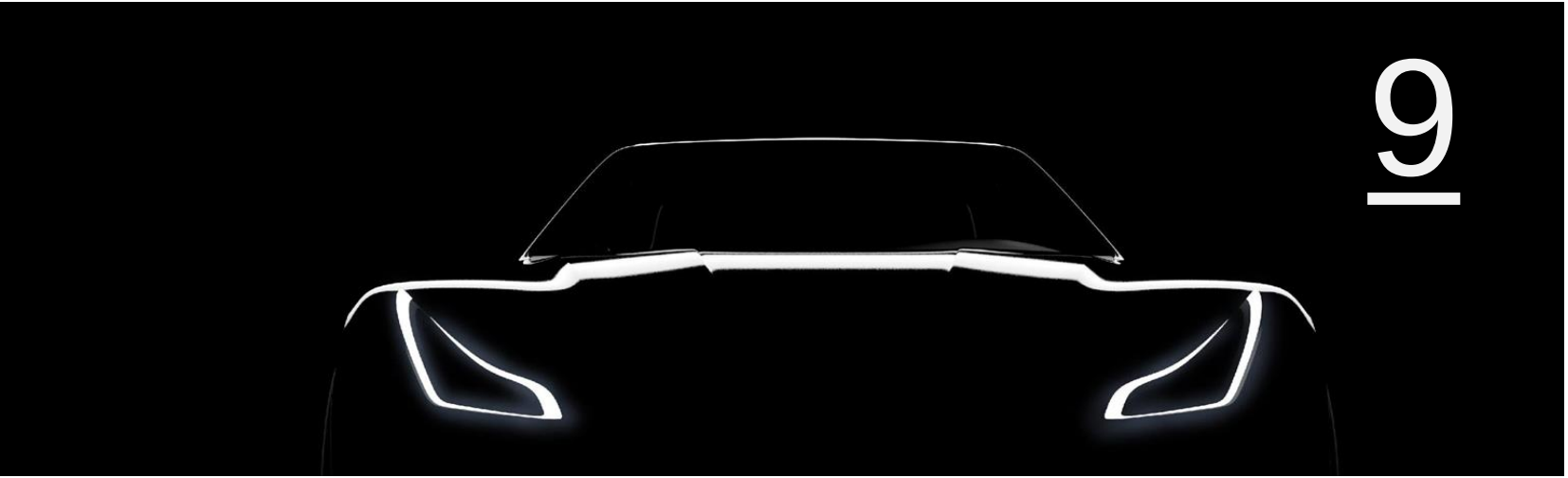
Batch Serverless Data Pipeline



Mixed Serverless Data Pipeline



Integrating with Cloud Pub/Sub



Cloud Pub/Sub

#GCPSketchnote

@PVERGADIA THECLOUDGIRL.DEV

10.19.2020

IT OPS SAM

"Event-driven architecture?" But our servers can't deal with the unpredictable traffic.

We have to make our systems real-time.

DEVELOPER ERIN

Pub/Sub!

What is Pub/Sub?

EVENT DRIVEN ASYNCHRONOUS MESSAGING SERVICE

Service (Produce event) → Pub/Sub → Service (Process event)

- In-order delivery at scale
- At-least-once delivery — In conjunction with Dataflow
- Pub/Sub Lite — Cost optimized delivery
- Exactly once processing
- Filtering — Filter messages based on attributes
- Dead letter topics — Offline message inspection
- Seek & Replay — Ability to reprocess & discard messages

How does Pub/Sub work?

We just set up Topic & Subscription, Pub/Sub takes care of the rest!

PUB/SUB MESSAGE FLOW

Publisher → Message (1) → Cloud Pub/Sub: Topic → Subscription (3) → Message (4) → Subscriber

Topic → Message Storage (2)

Subscriber → Ack (5) → Subscription

How does Pub/Sub help with Stream Analytics?

Trigger → Ingest → Enrich → Analyze → Activate

Architecture

- Edge: Mobile, Web, Data Store, IoT → Pub/Sub
- Enrich: Apache Beam (SDK), Dataflow Streaming, Analyze, Activate
- Analyze: BigQuery, AI Platform, Bigtable, Backfill/Reprocess, Dataflow Batch
- Activate: Studio, Third-party BI, Cloud Functions

Creation Flow

- Configure source to push event message to Pub/Sub Topic
- Create Pub/Sub Topic and Subscription
- Deploy streaming or batch Dataflow job using templates, CLI, or notebooks
- Create dataset, tables, and models to receive stream
- Build real-time dashboards and call external APIs

What are different Pub/Sub patterns?

MANY-TO-ONE (FAN-IN)

Publisher → Topic → Subscription → Subscriber

MANY-TO-MANY

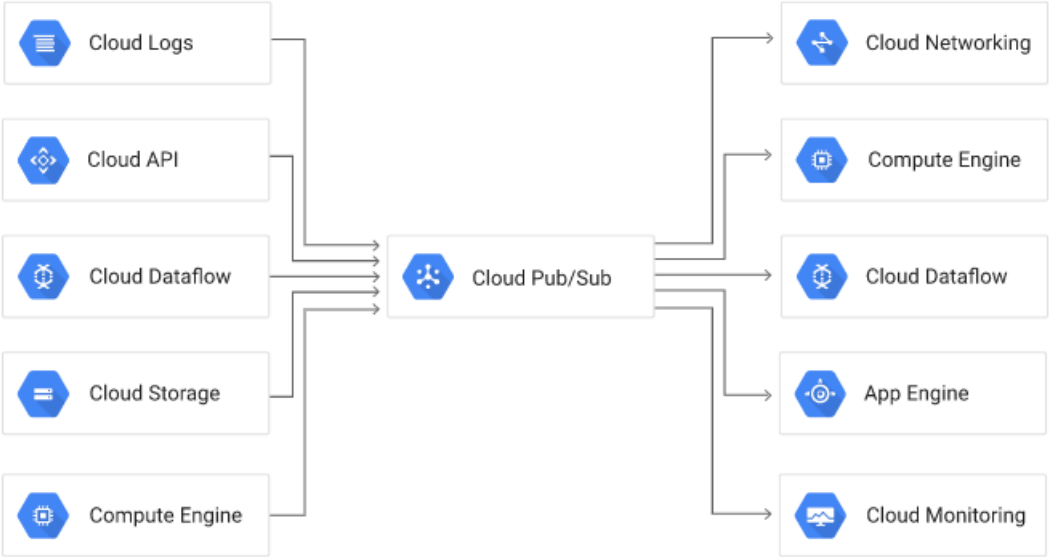
Publisher → Topic → Subscription → Subscriber

LOAD BALANCING

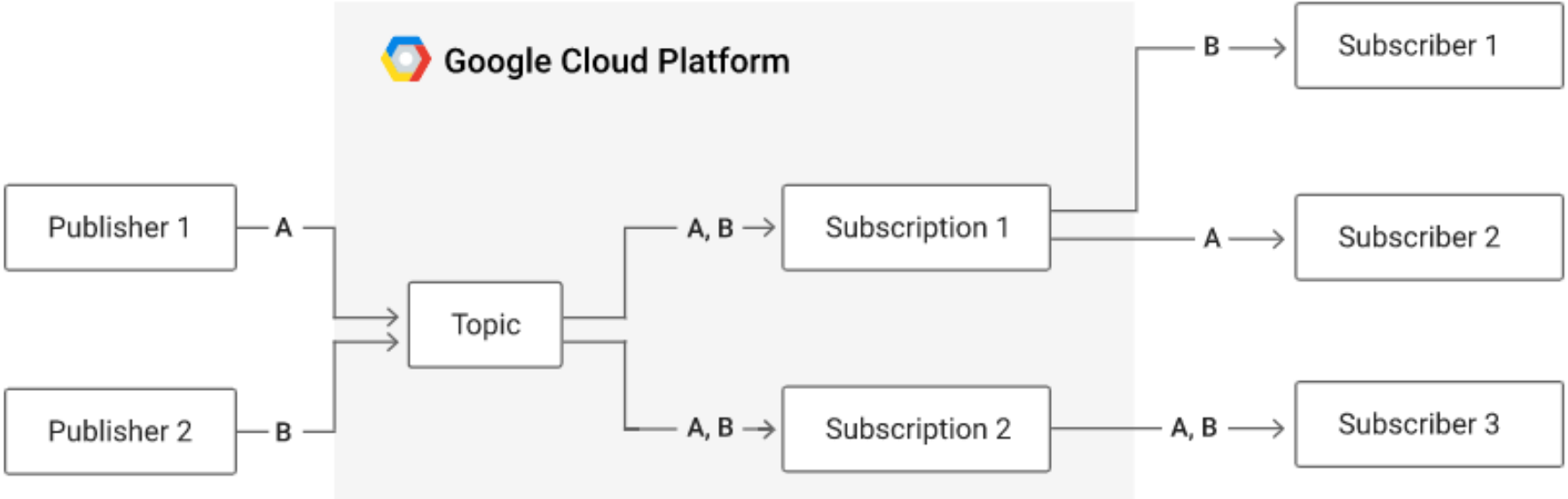
ONE-TO-MANY (FAN-OUT)

Publisher → Topic → Subscription → Subscriber

Cloud Pub/Sub In Context



Cloud Pub/Sub Flow



Cloud Pub/Sub Processing

1. A publisher sends a message.
2. The message is written to storage.
3. Pub/Sub sends an acknowledgement to the publisher that it has received the message and guarantees its delivery to all attached subscriptions.
4. At the same time as writing the message to storage, Pub/Sub delivers it to subscribers.
5. Subscribers send an acknowledgement to Pub/Sub that they have processed the message.
6. Once at least one subscriber for each subscription has acknowledged the message, Pub/Sub deletes the message from storage.

Cloud Pub/Sub Performance

A messaging service like Pub/Sub can be judged on its performance in three aspects: scalability, availability, and latency. These three factors are often at odds with each other, requiring compromises on one to improve the other two.

The terms "scalability," “availability,” and “latency” can refer to different properties of a system like Pub/Sub.

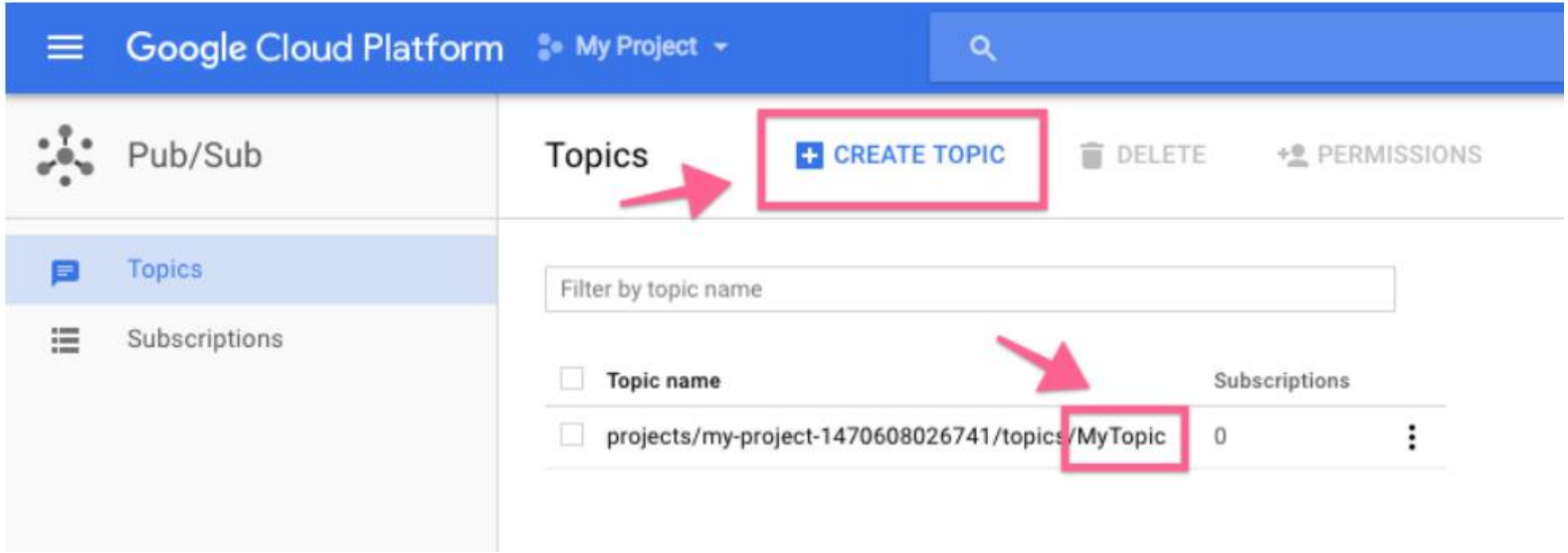
Building a Client

- Client for Pubsub can be any data producer like mobile app events, user behavior touchpoints, database changes (change data capture), sensor data, etc.
- Cloud PubSub provides many client libraries such as Java, C#, C++, Go, Python, Ruby, PHP & REST API.
- One can easily integrate client code into data producer to publish the data into Cloud PubSub.

Firing Data Pub/Sub from Scheduler

- GCP provides a simple scheduling tool called “Cloud Scheduler”. This is available from the GCP navigation menu on the top left as Cloud Scheduler. Assuming we already have a topic created, we move to the Scheduler.
- Select a region and wait for the initialization of Cloud Scheduler if not used previously in this region.
- Give your job a name, description, frequency and time zone. If you ever worked with any scheduler including Cron Jobs, you should feel pretty comfortable.
- Once you select Pub/Sub as the target you see options for topics and payload.

Creating a Topic



Firing Data Pub/Sub from Scheduler

- **Define the schedule**

Name *
data-pipeline-pub-sub-test

Must be unique across the jobs in the same region

Description
New Cloud Scheduler to fire a messaging data pipeline

Frequency *
0 9 * * 1

Schedules are specified using unix-cron format. E.g. every minute: "* * * * *", every 3 hours: "0 */3 * * *", every monday at 9:00: "0 9 * * 1". Learn more

Timezone *
Central Standard Time (CST)

Jobs in set in timezones affected by Daylight Saving Time can run outside of cadence during DST change. Using a UTC timezone can avoid the problem. Learn more

Firing Data Pub/Sub from Scheduler

- **Define the schedule**

Name *
data-pipeline-pub-sub-test

Must be unique across the jobs in the same region

Description
New Cloud Scheduler to fire a messaging data pipeline

Frequency *
0 9 * * 1

Schedules are specified using unix-cron format. E.g. every minute: "* * * * *", every 3 hours: "0 */3 * * *", every monday at 9:00: "0 9 * * 1". Learn more

Timezone *
Central Standard Time (CST)

Jobs in set in timezones affected by Daylight Saving Time can run outside of cadence during DST change. Using a UTC timezone can avoid the problem. Learn more

Firing Data Pub/Sub from Scheduler

- **Configure the execution**

Target type *
Pub/Sub

Select a Cloud Pub/Sub topic *

Message body

Message size cannot exceed 10MB.

Message attributes

Key * **Value ***

Key | Value

+ ADD AN ATTRIBUTE

Viewing Messages

Google Cloud Platform | project2 | Search products and resources

Pub/Sub

- Topics
- Subscriptions
- Snapshots
- Schemas
- Lite Topics
- Lite Subscriptions

test-1 | + PUBLISH MESSAGE | VIEW MESSAGES | + TRIGGER CLOUD FUNCTION | IMPORT | DELETE | HIDE INFO PANEL

Export options have moved to the **Create subscription** dropdown menu under the Subscriptions tab below.

GOT IT

Topic details

Topic name	projects/project2-301100/topics/test-1

Export to BigQuery

Create a Dataflow job to export data to a BigQuery table.

EXPORT TO BIGQUERY

Export to Cloud Storage

Create a Dataflow job to export data to a text or Avro file in Cloud Storage.

EXPORT TO TEXT | EXPORT TO AVRO

1 hour | 6 hours | 1 day | 7 days | 30 days | RESET ZOOM

Publish message request count

Requests/sec

3.60

3.55

test-1

PERMISSIONS | LABELS | STORAGE POLICY

Edit or delete permissions below or "Add Member" to grant new

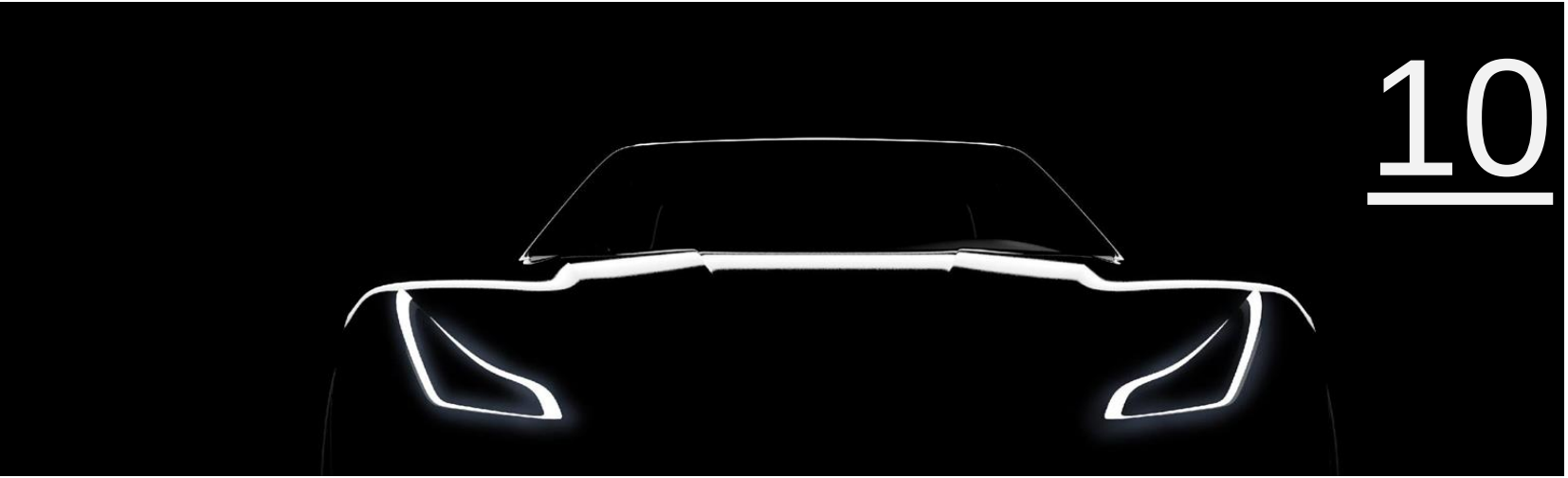
ADD MEMBER

Show inherited permissions

Filter | Enter property name or value

Role / Member	Inheritance
Cloud Build Service Account (1)	
Cloud Build Service Agent (1)	
Cloud Composer API Service Agent (1)	
Cloud Dataflow Service Agent (2)	
Cloud Scheduler Service Agent (1)	
Composer Worker (1)	
Container Registry Service Agent (1)	
DLP API Service Agent (2)	
Editor (2)	
Kubernetes Engine Service Agent (1)	
Owner (3)	

Google Cloud Functions



Cloud Functions

#GCPSketchnote

@PVERGADIA THECLOUDGIRL.DEV

8.17.2020

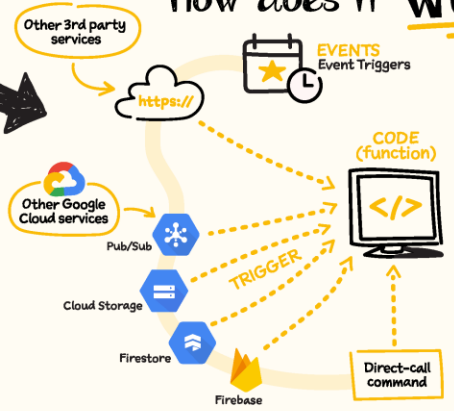
What is [Cloud Functions]?

EVENT-DRIVEN SERVERLESS

FUNCTION-AS-A-SERVICE (FAAS)

RUN CODE IN RESPONSE TO AN EVENT

How does it WORK?

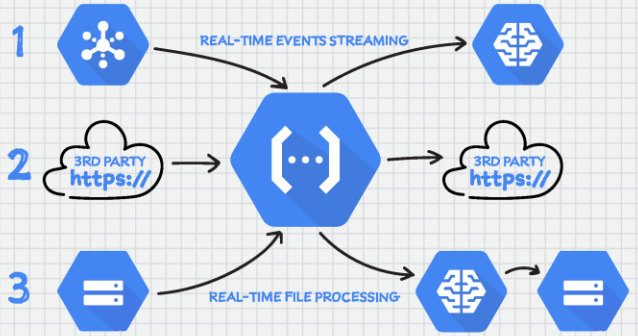


CLOUD FUNCTIONS FEATURES

- Based on open source FaaS framework (No vendor lock in)
- No server management
- Automatic scale (from 0 to millions)
- Just write and deploy your code
- Observability and diagnosis

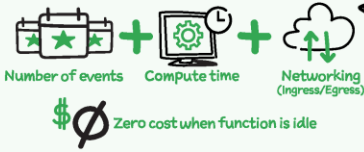
Cloud Functions Use case (example)

- Integration with third-party services and APIs
- IoT processing
- Real-time data processing - streaming
- Real-time file processing
- Serverless mobile back-ends
- Virtual assistance
- Real time ML



Cloud Functions PRICING

PAY AS YOU GO



Zero cost when function is idle

SECURITY for Cloud Functions

NETWORK-BASED

VPC controls

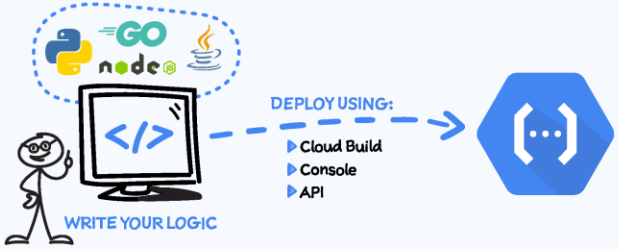
+

IDENTITY-BASED

Identity & Access Management (IAM)

Function Role

HOW TO USE Cloud Functions



Just write your logic in any of the supported languages and trigger it via HTTP request, file uploads to cloud storage, events in Pub/Sub or firebase.

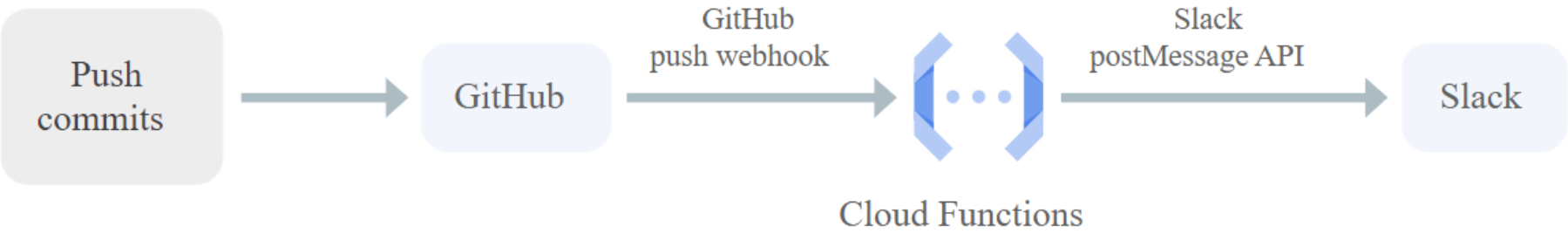
What are Cloud Functions?

- Cloud Functions is a Function as a Service (FaaS) offering.
- This means that you write a function in one of the languages supported by GCP and it can be triggered by an event or via HTTP.
- GCP takes care of provisioning and scaling the resources that are needed to run your functions.

How do Cloud Functions work?

- Cloud Functions are actually using containers to set an isolated environment for your function.
- These are called Cloud Functions instances.
- If multiple functions are executed in parallel, multiple instances are created.

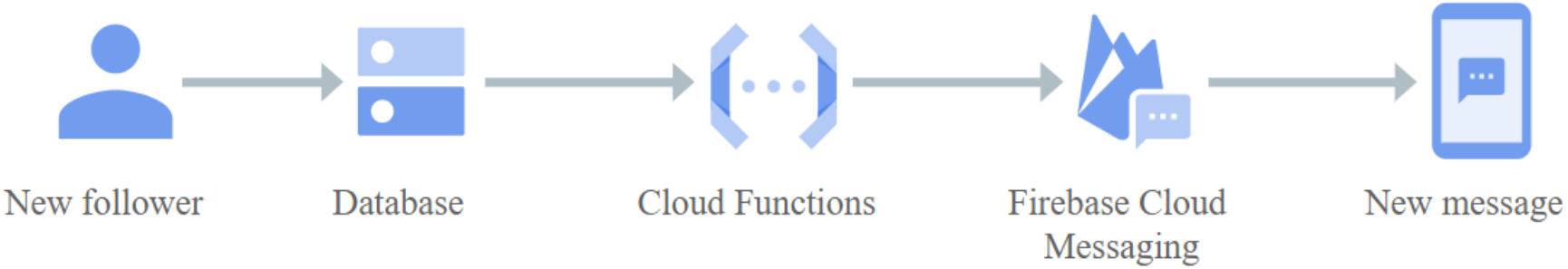
Integrating with APIs



Connect and Extend Servicd es

- Cloud Functions lets you treat all Google and third-party cloud services as building blocks.
- Connect and extend them with code, and rapidly move from concept to production with end-to-end solutions and complex workflows.
- Further, integrate with third-party services that offer webhook integrations to quickly extend your application with powerful capabilities.

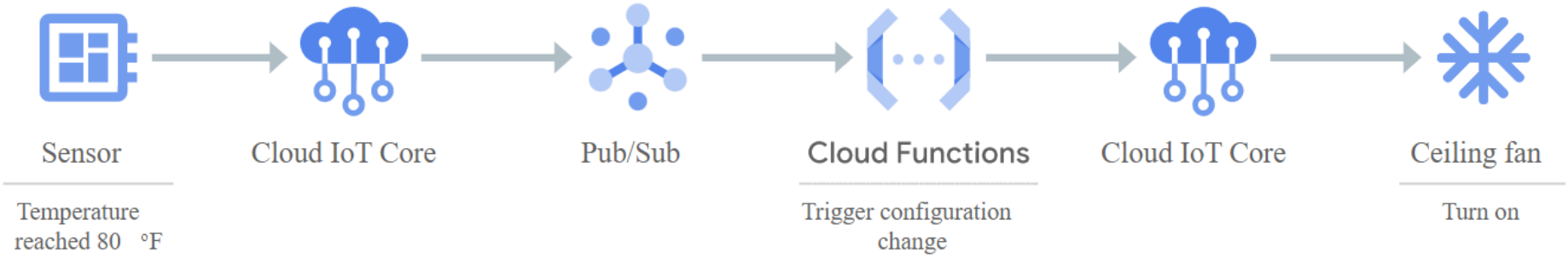
Streaming Data to Mobile



End to End Development and Diagnostic

- Go from code to deploy, with integrated monitoring.
- Get full observability and diagnosability for your application with Cloud Trace and Cloud Debugger.
- Support for local and disconnected development/debugging using open sourced functions framework.

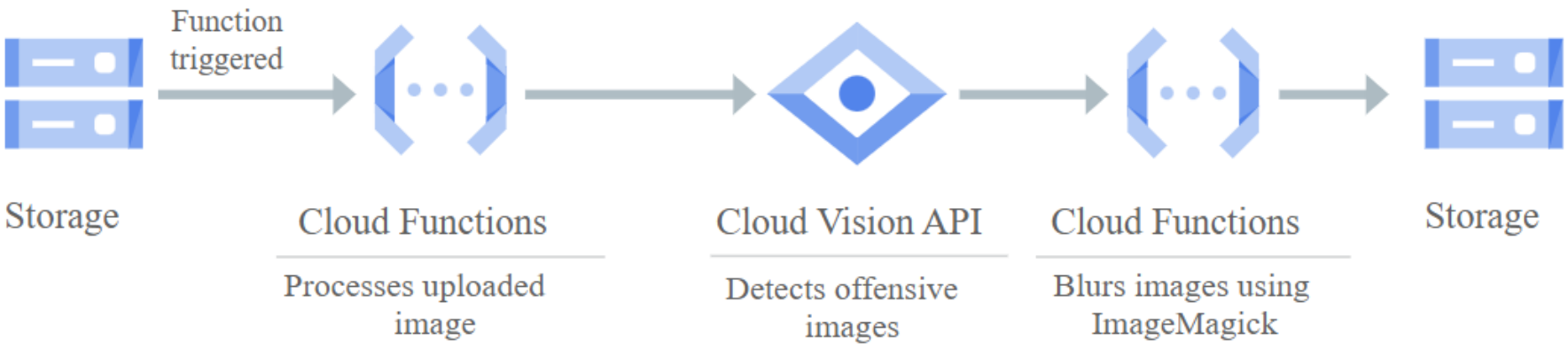
Iot Data Pipelines



Scale Automatically

- Serve users from zero to planet-scale without even thinking about any infrastructure.
- Cloud Functions automatically manages and scales underlying infrastructure with the size of workload.

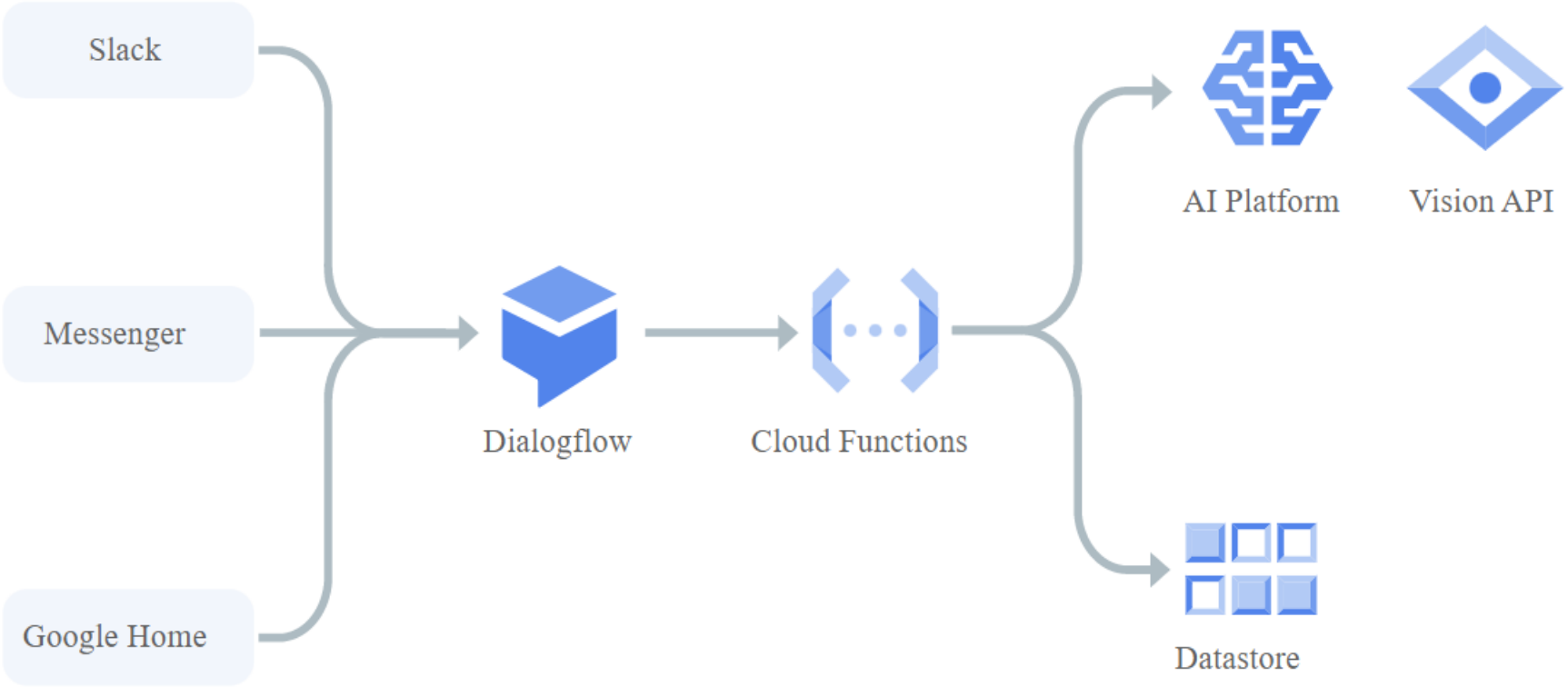
Real-time Data Processing Pipelines



No Server Management

- Deploy your code and let Google run and scale it for you.
- Cloud Functions abstracts away all the underlying infrastructure, so that you can focus on your code and build applications faster than ever before.

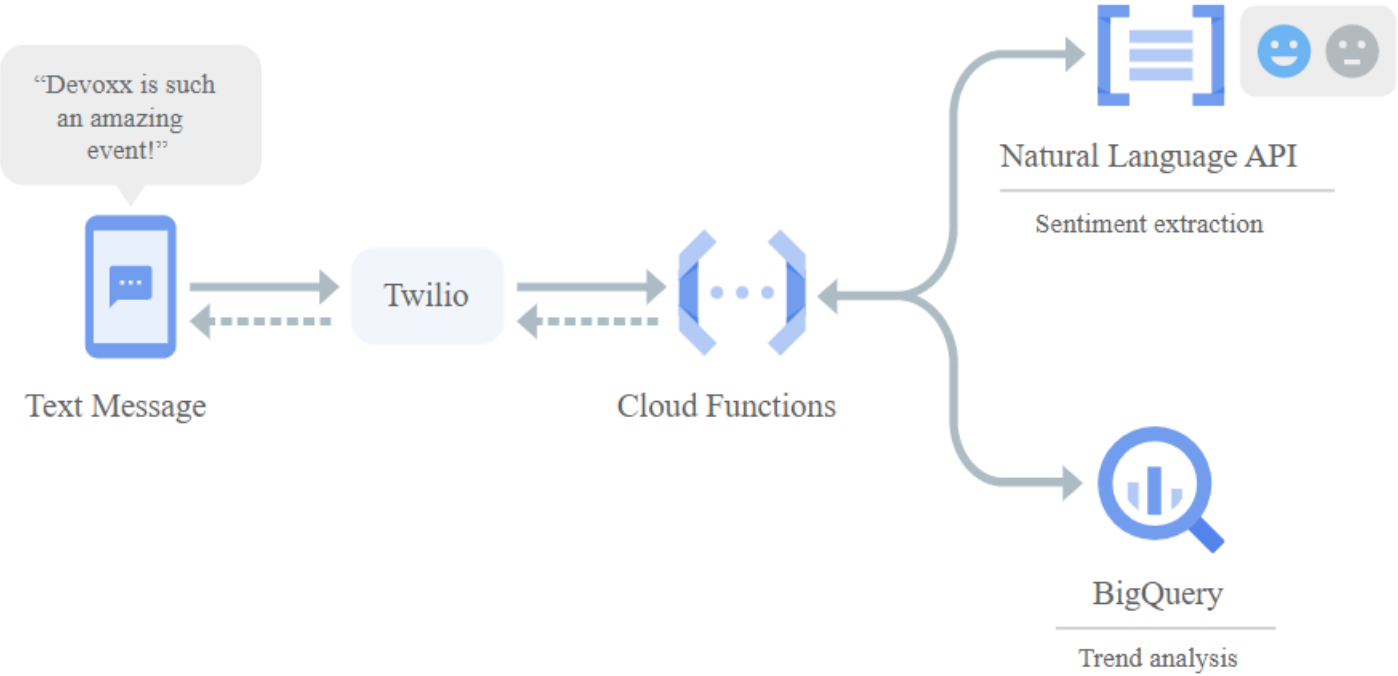
Conversational AI Pipelines



Run Code in Response to Events

- Cloud Functions allows you to trigger your code from Google Cloud, Firebase, and Google Assistant
- Call Cloud Functions directly from any web, mobile, or backend application via HTTP.

Sentiment Analysis Pipelines



Pay For What You Use

- You are only billed for your function’s execution time, metered to the nearest 100 milliseconds.
- You pay nothing when your function is idle.
- Cloud Functions automatically spins up and backs down in response to events.

Cloud Functions in a Nutshell

- **Serverless**: Cloud Functions are completely serverless. The underlying infrastructure is abstracted from the end user.
- **Event-driven**: Cloud Functions are event-driven. There are triggered in response to an event or HTTP request. This means they are invoked only when needed and do not produce any cost when inactive.
- **Stateless**: Cloud Functions do not store state nor data. This allows them to work independently and scale as needed
- **Autoscaling**: Cloud Functions scale from zero to the desired scale. Scaling is managed by GCP without any end user intervention. Autoscaling limits can be set to control the cost of execution.

Cloud Functions Pipeline Use Cases

When it comes to event-driven data processing, Cloud Functions can be triggered whenever a predefined event occurs. When this happens, it can preprocess the data that's passed for analysis with GCP big data services:

- **Real-time stream processing**: When messages arrive in the Pub/Subs, Cloud Functions can be triggered to analyze or enrich the messages to prepare them for further data processing steps in the pipeline.
- **Real-time files processing**: When files are uploaded to your Cloud Storage bucket, they can be immediately processed. For example, thumbnails can be created or analyzed using GCP AI APIs.

Cloud Functions Getting Started

Google Cloud Functions

Google Cloud Functions is a lightweight, event-based, asynchronous compute solution that allows you to create small, single-purpose functions that respond to cloud events without the need to manage a server or a runtime environment

CREATE FUNCTION

In the case of most services in GCP, once you access that area in the console the service must be initialized or enabled and then you can move forward to use that.

Cloud Functions Getting Started

Google Cloud Functions

Google Cloud Functions is a lightweight, event-based, asynchronous compute solution that allows you to create small, single-purpose functions that respond to cloud events without the need to manage a server or a runtime environment

CREATE FUNCTION

In the case of most services in GCP, once you access that area in the console the service must be initialized or enabled and then you can move forward to use that.

Cloud Functions Getting Started - Config

Cloud Functions | Create function

1 **Configuration** — 2 Code

Basics

Function name *

data-pipeline-function

Region

us-central1

Trigger

HTTP

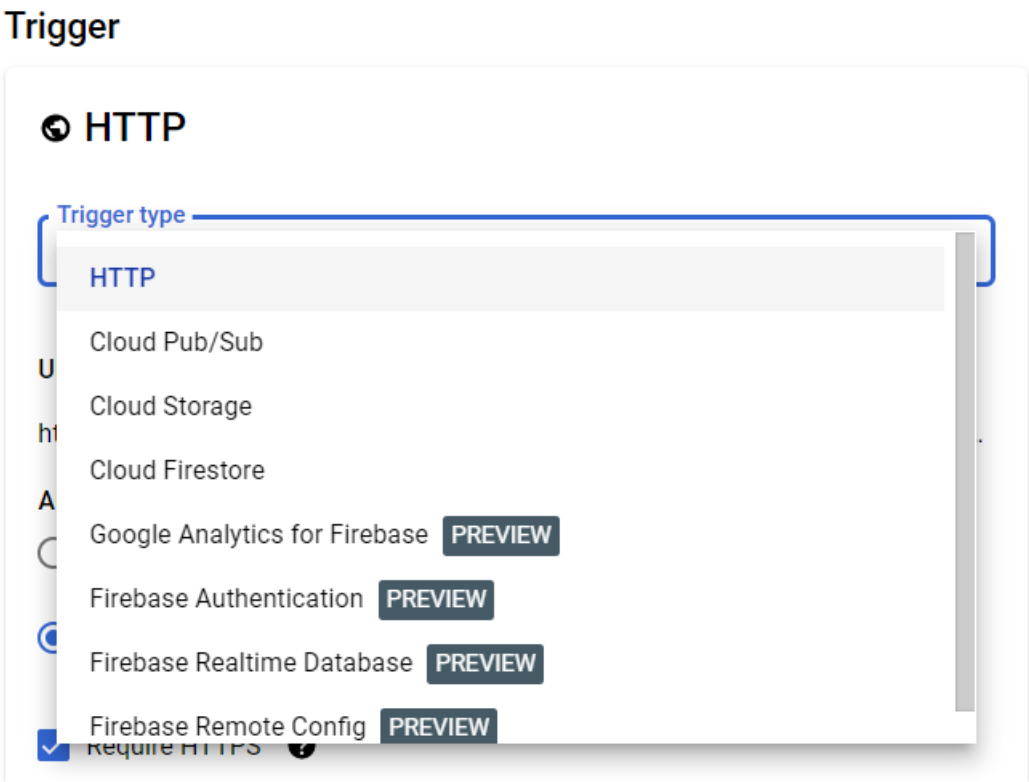
Trigger type

HTTP

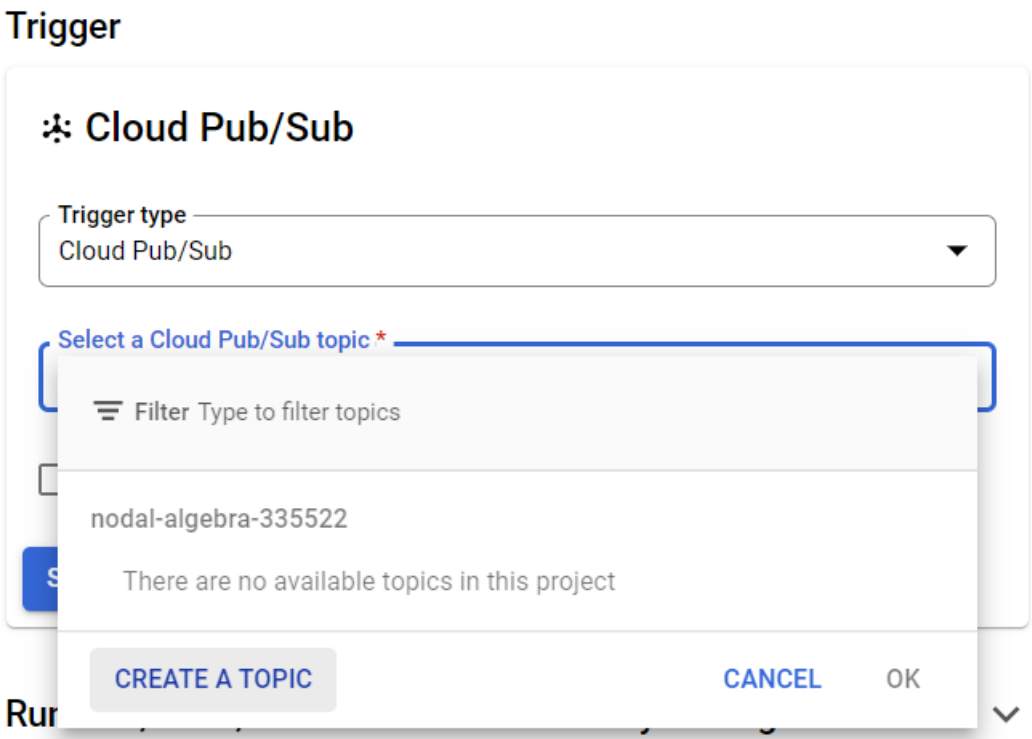
URL

https://us-central1-nodal-algebra-335522.cloudfunctions.net/data-pipeline-functi...

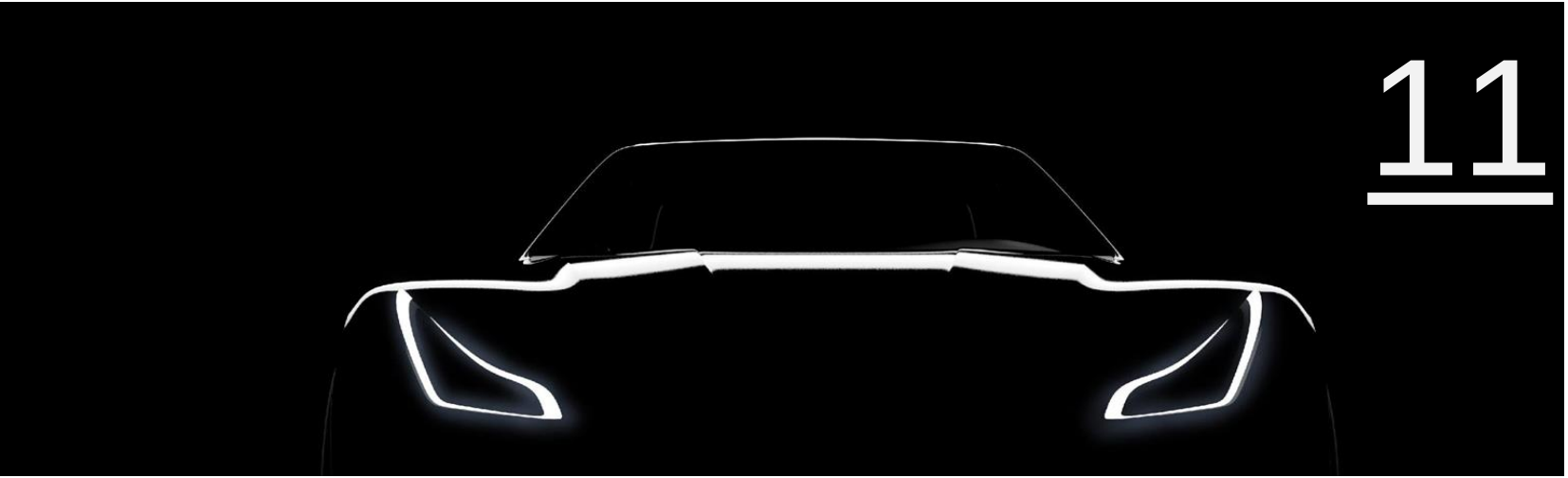
Cloud Functions Getting Started - Triggers

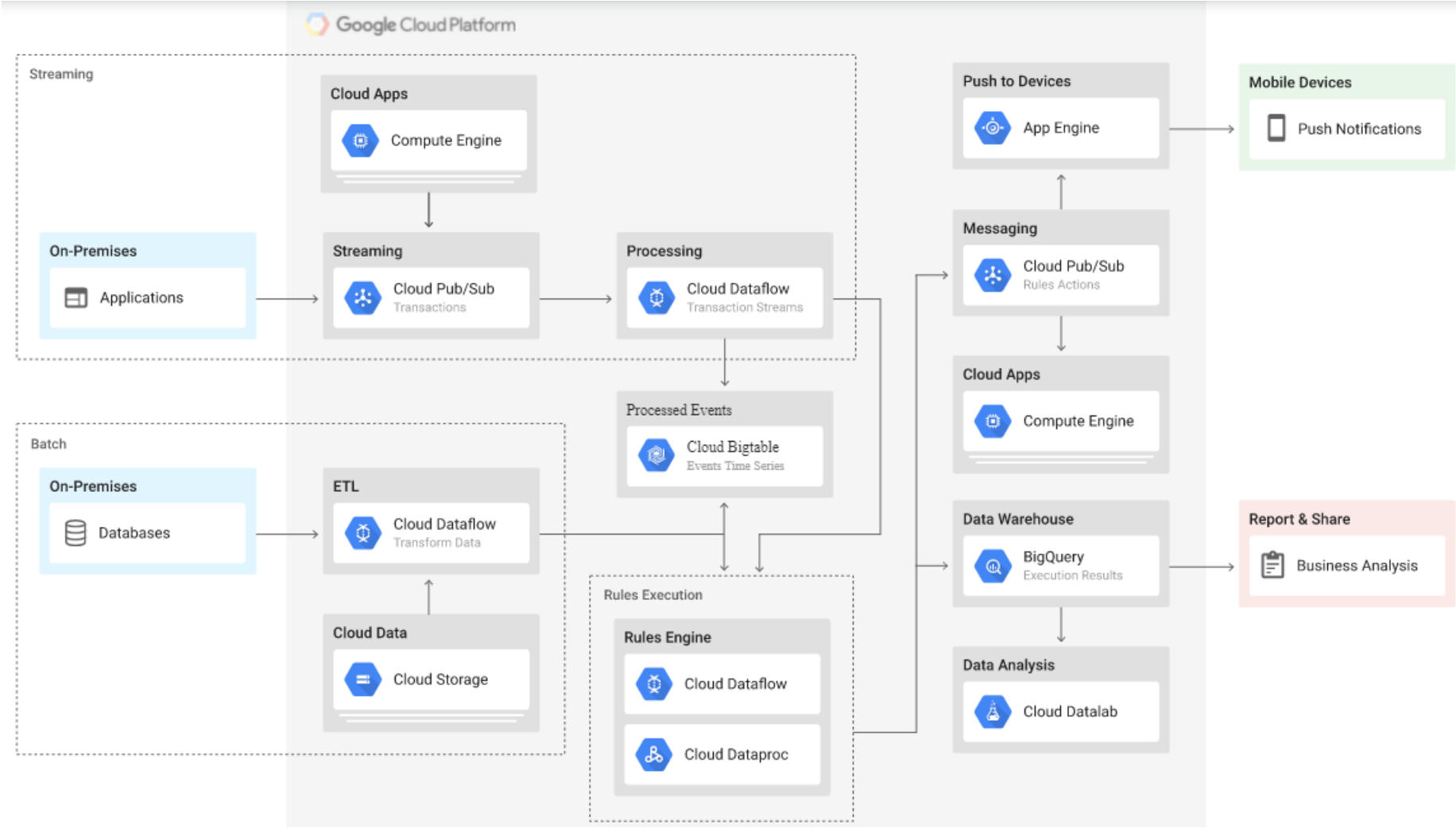


Cloud Functions Getting Started – Pub/Sub



Event-driven Processing




Google Cloud Platform
Streaming
Cloud Apps
Compute Engine
On-Premises
Applications
Streaming
Cloud Pub/Sub
Transactions
Processing
Cloud Dataflow
Transaction Streams
Push to Devices
App Engine
Mobile Devices
Push Notifications
Messaging
Cloud Pub/Sub
Rules Actions
Cloud Apps
Compute Engine
Processed Events
Cloud Bigtable
Events Time Series
Batch
On-Premises
Databases
ETL
Cloud Dataflow
Transform Data
Cloud Data
Cloud Storage
Rules Execution
Rules Engine
Cloud Dataflow
Cloud Dataproc
Data Warehouse
BigQuery
Execution Results
Report & Share
Business Analysis
Data Analysis
Cloud Datalab

Event Sources

Data sources that contain events to be processed can be located on-premises or on Google Cloud.

For on-premises data sources, you can export data and subsequently copy it to Cloud Storage for downstream processing.

There are several different platforms you can use to store unprocessed event data on Google Cloud:

- Cloud Storage provides object storage for flat files or unstructured data.
- Cloud SQL stores relational data.
- Datastore is a NoSQL database that supports dynamic schemas and full data-indexing.

Event Data Processing

- As event data streams in from applications through Pub/Sub or is loaded into other cloud data sources, such as Cloud Storage, Cloud SQL, Datastore, you might need to transform, enrich, aggregate, or apply general purpose computations across the data.
- Dataflow is built to perform these tasks on data that is streaming or batched, by using a simple pipeline-based programming model.
- Dataflow executes these pipelines on a managed service that elastically scales, as needed.
- In Complex Event Processing (CEP) workloads, you can use Dataflow to normalize data from multiple heterogeneous sources and format it into a single, consistent representation.

Event Storage

- Dataflow can be used to process event data and normalize it into a single, consistent, time-series representation, and then store in a SQL Database, Messaging, or Cloud Bigtable, a fully managed, NoSQL database service.

- Databases, and Messaging can provide consistent, low-latency, high-throughput data storage, useful for downstream processing and analytical workloads.

Rules Execution

- At the heart of CEP systems are rules that must be executed across incoming events.
- Rules engines can vary from simple, custom, domain-specific languages to complex business rules management systems.
- Regardless of the rules engine's complexity, it can be deployed in a distributed fashion using either Dataflow or Dataproc, a managed service for running Hadoop or Spark clusters.
- Using either of these systems, you can execute complex rules across large amounts of time series events, and then use the results of these rules evaluations to drive outbound actions.

Transport

- Whether applications are hosted on-premises or on Google Cloud, the events they generate must travel through a loosely coupled, highly available messaging layer.
- Pub/Sub, a globally durable message-transport service, supports native connectivity to other Google Cloud services, making it the glue between applications and downstream processing.

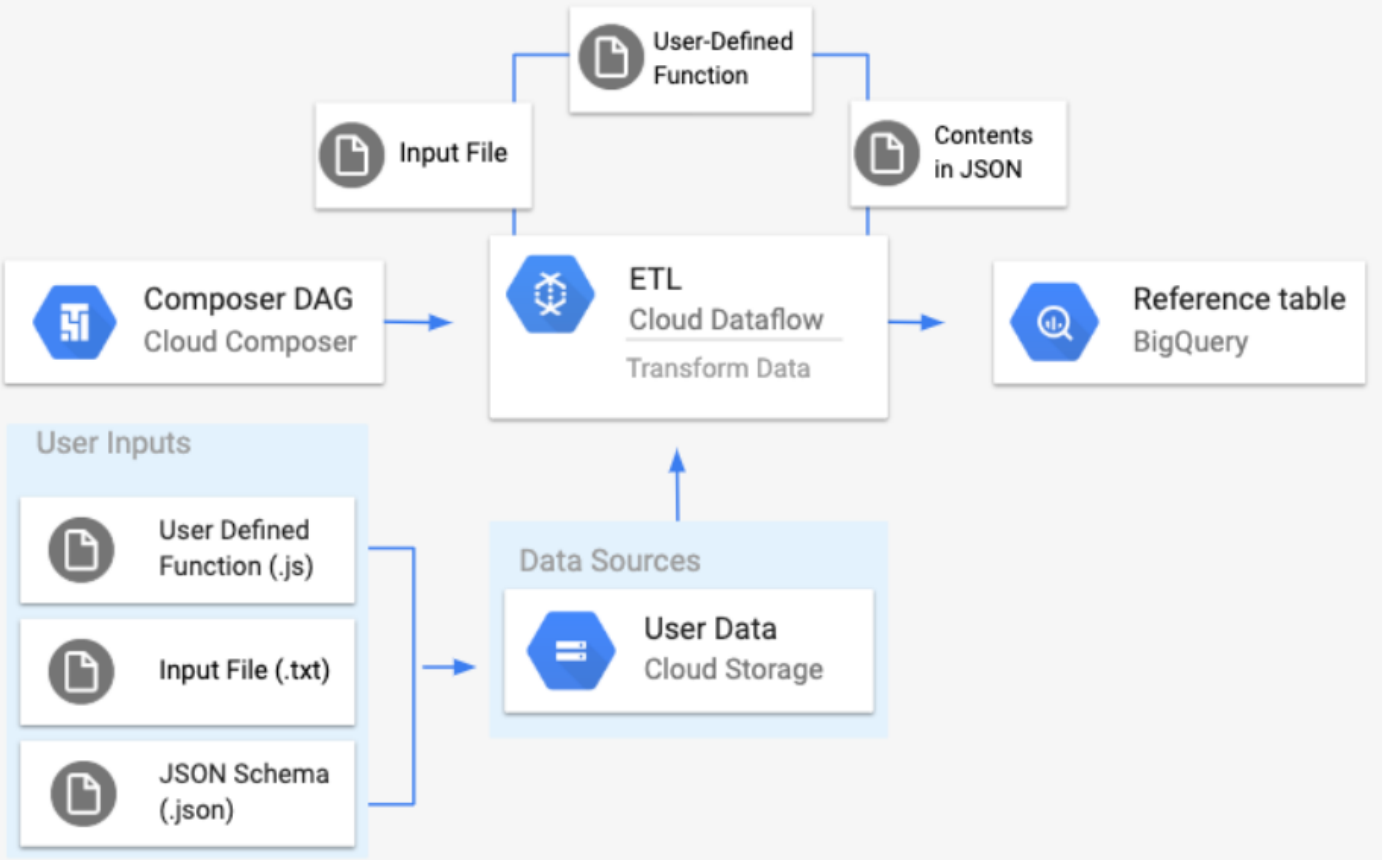
Outbound Events

- After inbound events have been processed by the rules execution mechanism, results can be turned into a series of outbound actions.
- These actions can include push notifications for mobile devices, notifications for other applications, or sending email to users.
- Deliver these actions, you can send messages through Pub/Sub to applications hosted on-premises or on Google Cloud, whether running on App Engine, GKE, or Compute Engine.

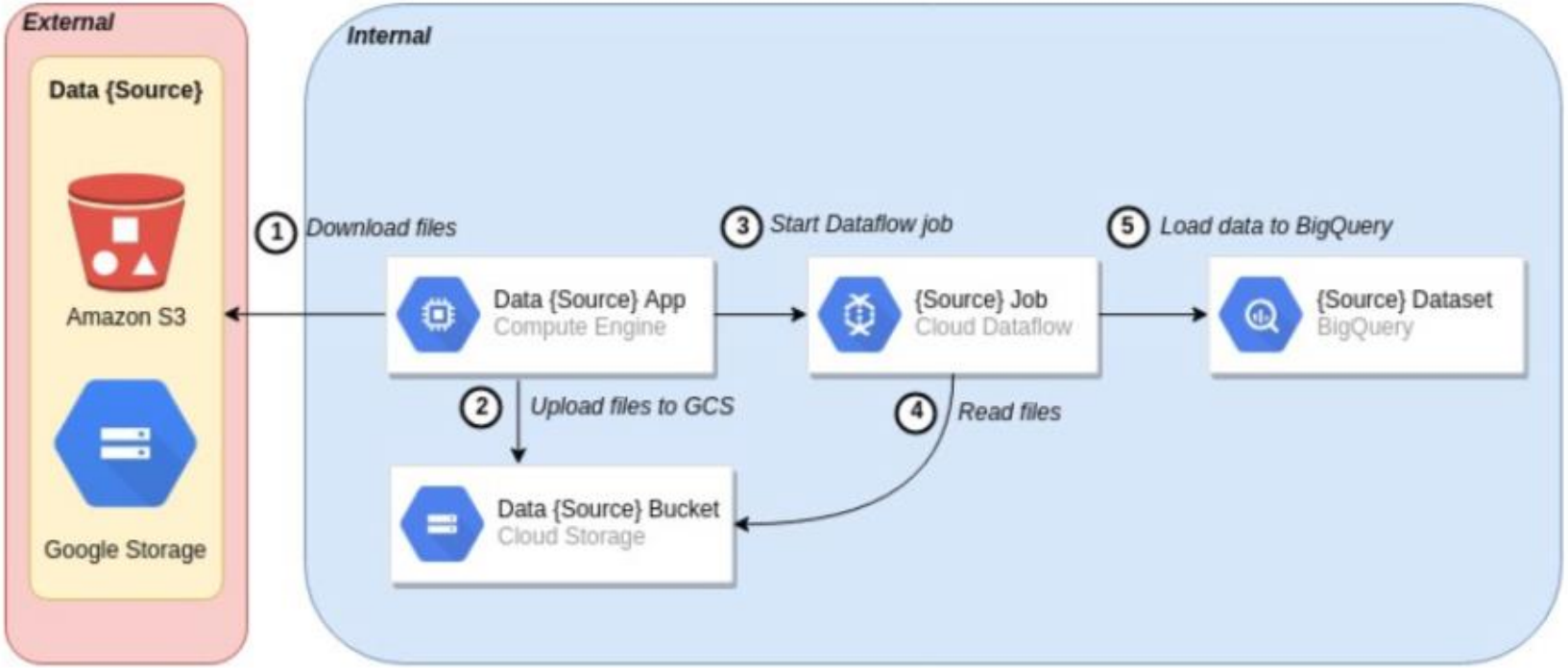
Event Analysis

- You can store the results of rules execution across inbound events in BigQuery, or another event sink, to perform exploration and analysis for business intelligence or reporting purposes.
- BigQuery is a managed, petabyte-scale data warehouse that supports SQL-like queries and has connectivity to additional downstream analytical tools.
- or data-science-oriented analysis, Datalab offers powerful data exploration capabilities based on Jupyter notebooks.
- For reporting or dashboard use cases, Google Data Studio supports creating customizable and shareable reports, backed by the query and storage capabilities of BigQuery.

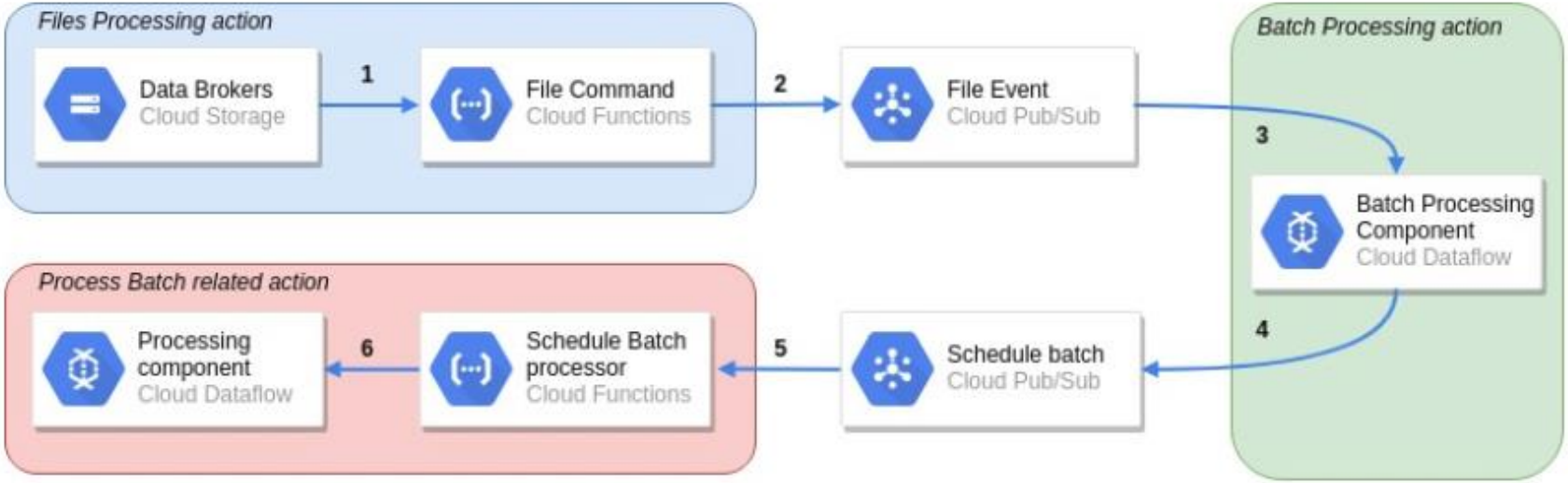
Triggering Data Pipeline Events



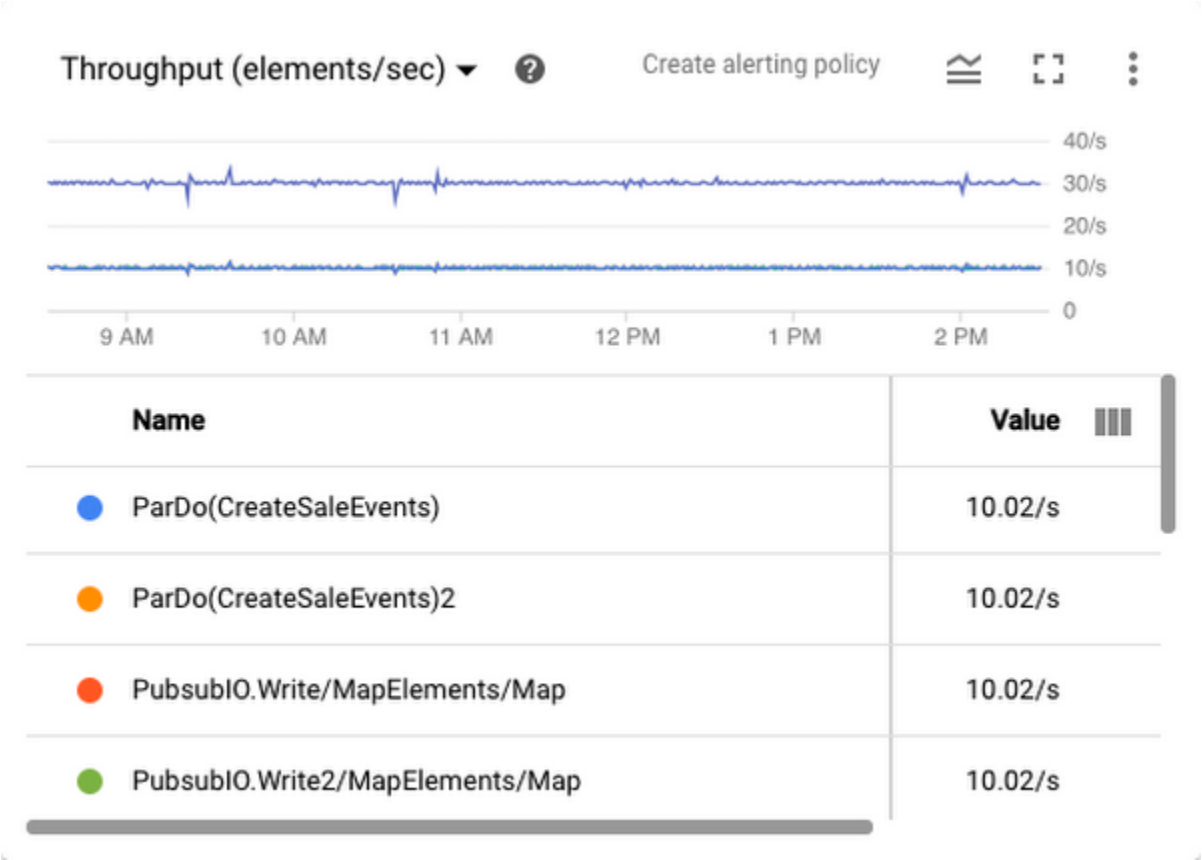
Historical Data Pipeline Triggering



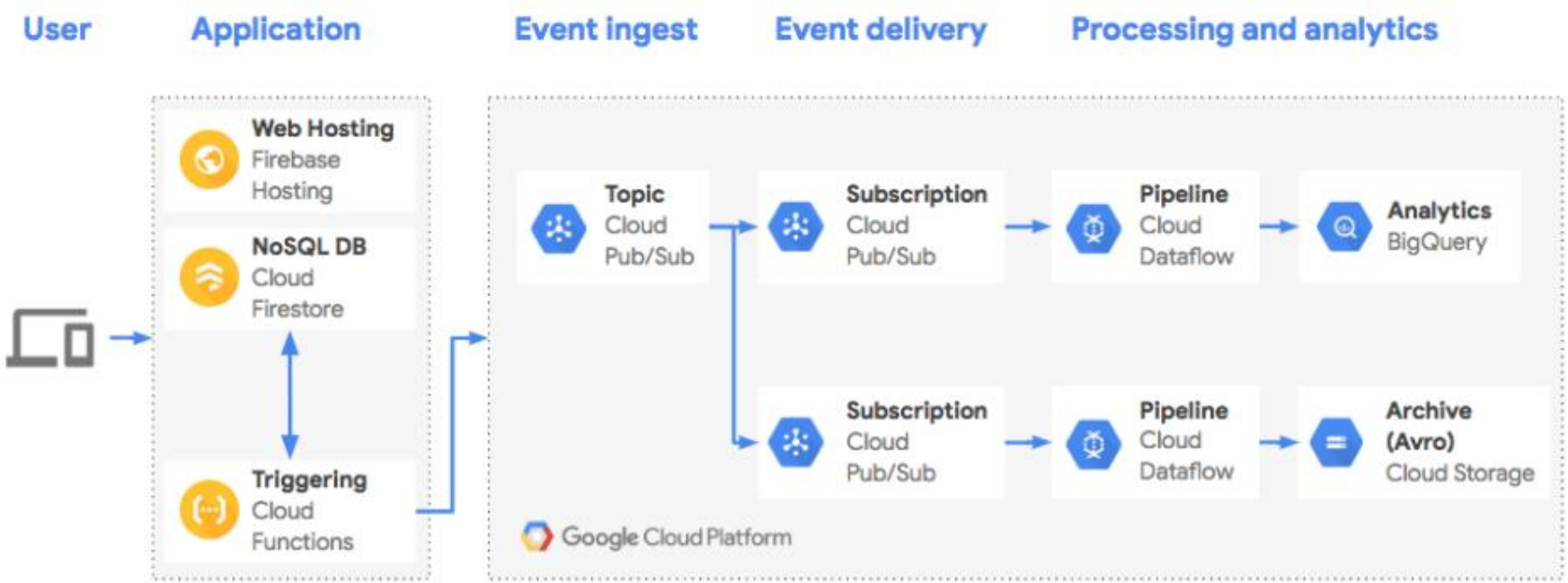
Functions Data Pipeline Triggering

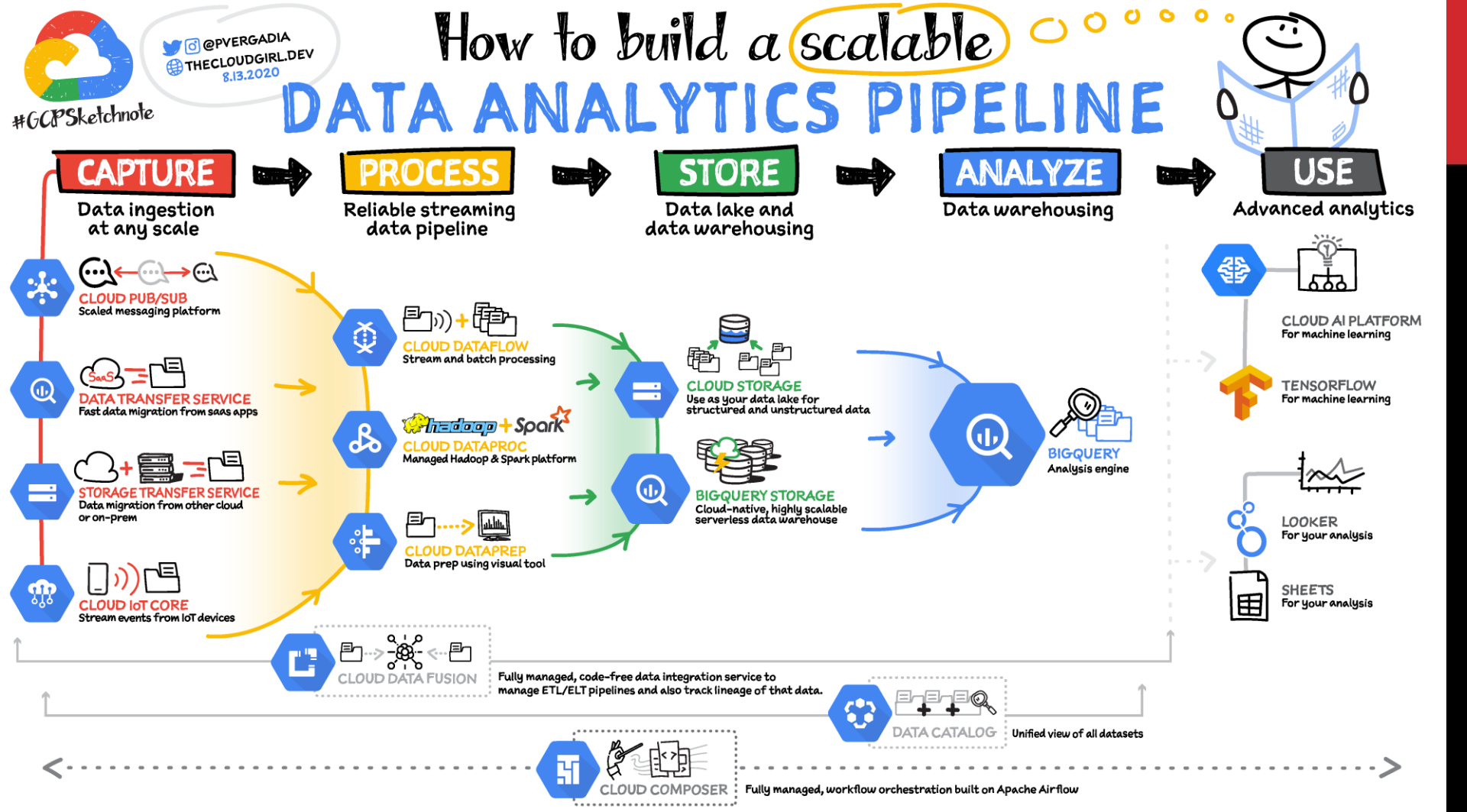


Data Pipeline Event Observability



Observability in Complex Event Pipelines




#GCPSketchnote
@PVERGADIA
THECLOUDGIRL.DEV
8.13.2020
How to build a scalable
DATA ANALYTICS PIPELINE
CAPTURE
Data ingestion at any scale
PROCESS
Reliable streaming data pipeline
STORE
Data lake and data warehousing
ANALYZE
Data warehousing
USE
Advanced analytics
CLOUD PUB/SUB
Scaled messaging platform
DATA TRANSFER SERVICE
Fast data migration from saas apps
STORAGE TRANSFER SERVICE
Data migration from other cloud or on-prem
CLOUD IoT CORE
Stream events from IoT devices
CLOUD DATAFLOW
Stream and batch processing
hadoop + Spark
CLOUD DATAPROC
Managed Hadoop & Spark platform
CLOUD DATAPREP
Data prep using visual tool
CLOUD STORAGE
Use as your data lake for structured and unstructured data
BIGQUERY STORAGE
Cloud-native, highly scalable serverless data warehouse
BIGQUERY
Analysis engine
CLOUD AI PLATFORM
For machine learning
TENSORFLOW
For machine learning
LOOKER
For your analysis
SHEETS
For your analysis
CLOUD DATA FUSION
Fully managed, code-free data integration service to manage ETL/ELT pipelines and also track lineage of that data.
DATA CATALOG
Unified view of all datasets
CLOUD COMPOSER
Fully managed, workflow orchestration built on Apache Airflow

Managing APIs

12

Managing APIs

- Public APIs are now considered to be the most effective mechanism to integrate two applications.
- Typically, the APIs are hosted in a server-side application and exposed to the clients to consume.
- The clients could be other server-side applications, native mobile applications or even browser applications (mobile or desktop).

Common API Characteristics

When we speak about a APIs, few things are common to them:

- They are available over a popular and well defined protocol, usually HTTP
- They follow a request/response pattern e.g. REST, gRPC, or GraphQL.
- They typically use JSON or XML as the data format.

The above are not absolutely binding but the majority of APIs have these in common.

Google Cloud Endpoints

- Google Cloud Endpoints is a solution from GCP to help create a Public API for your App Engine,Cloud Functions or other GCP-hosted applications.
- Cloud Endpoints also provides facilities to generate client libraries
- Libraries include those both application and mobile integration, thus easing your task to integrate your backend application functionality into your application and mobile data engineering pipeline consumers.

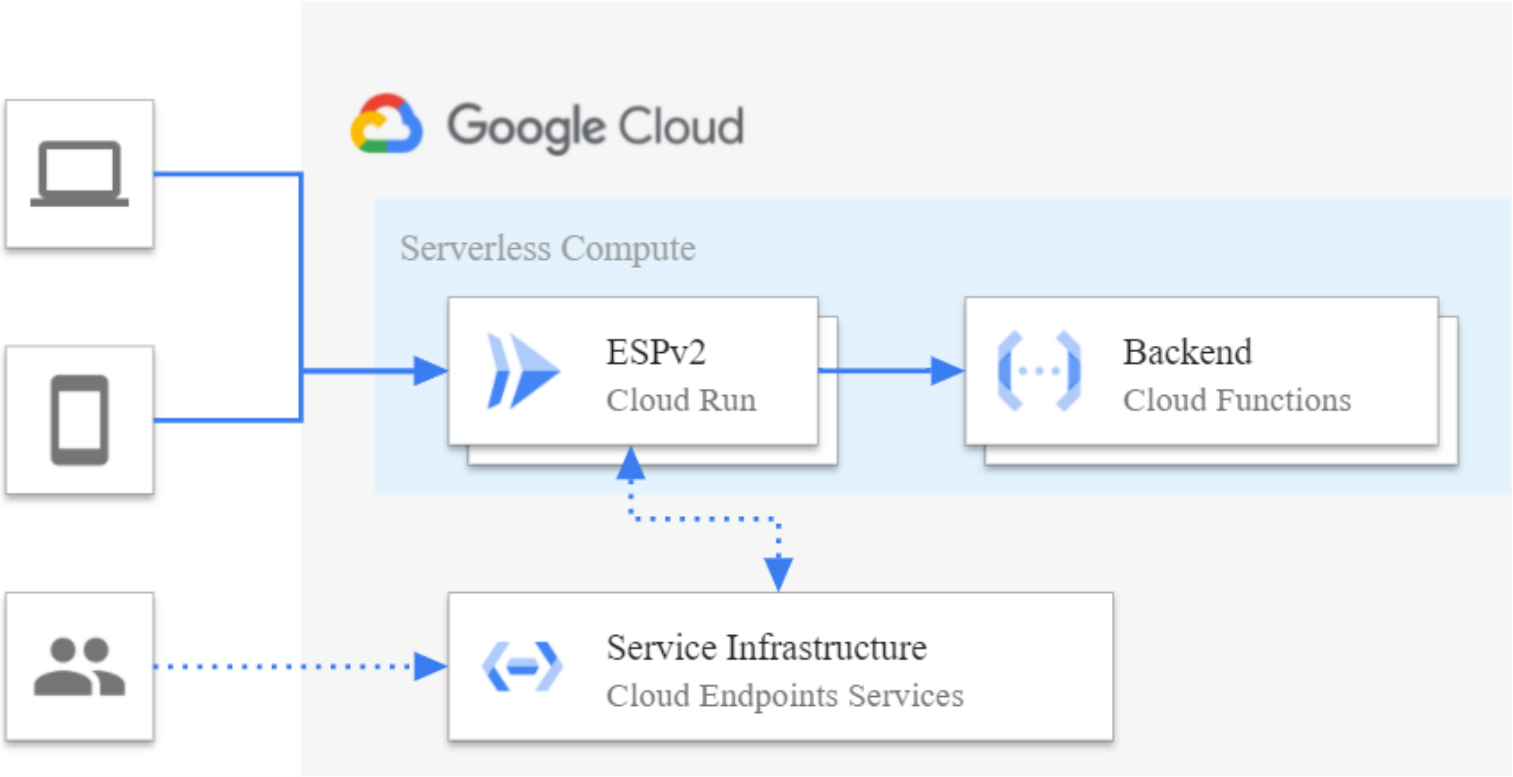
Client Libraries

You can use Google Cloud APIs by making direct HTTP requests to the server (or RPC calls where available).

GCP provides client library code for all our Cloud APIs that makes it easier to access them from your code.

You can find out more about the available libraries for your product or language of choice in the product or language's documentation.

Serverless with Cloud Endpoints

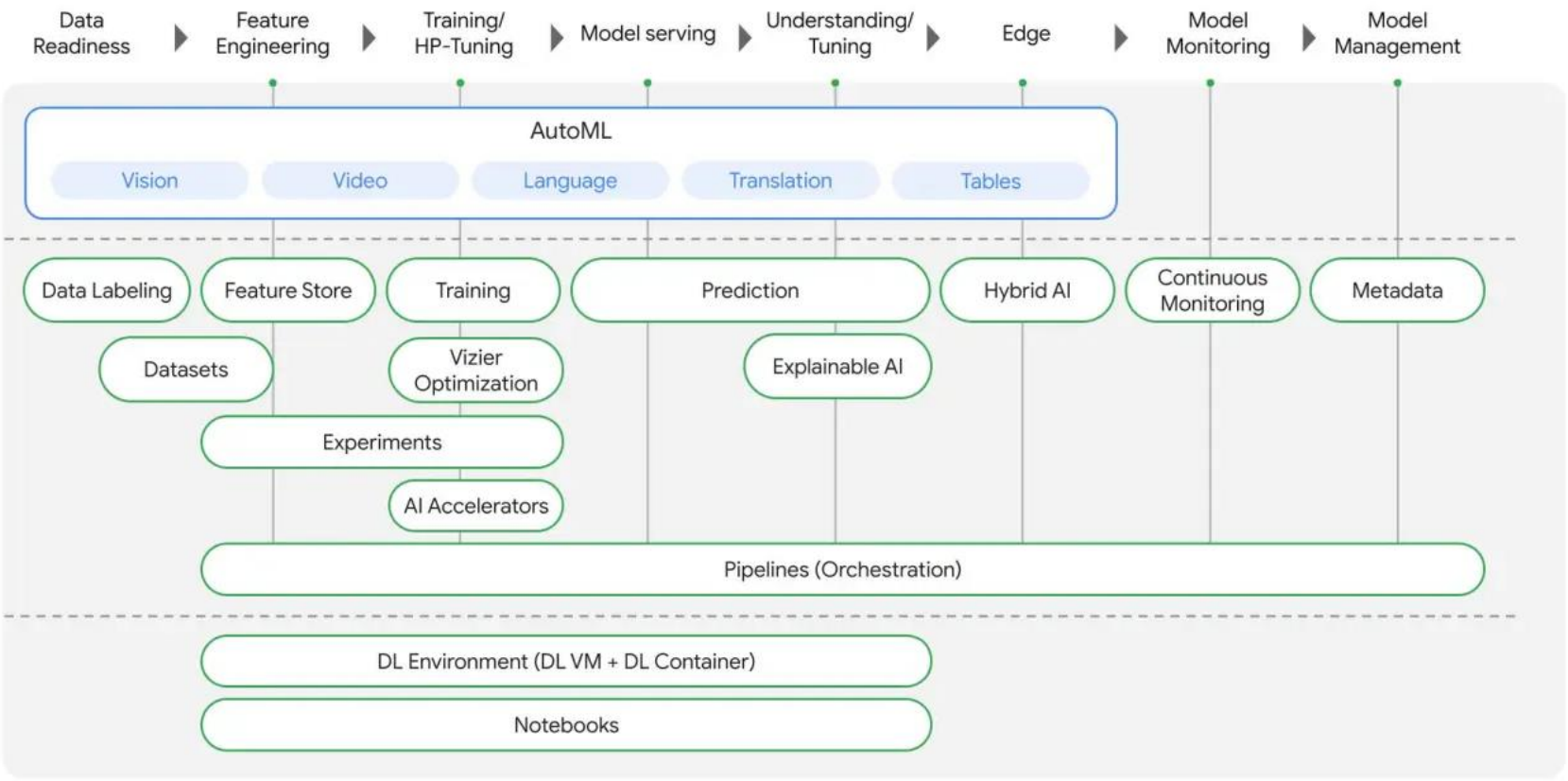


Flow Using Cloud Endpoints with FaaS

- Create a Google Cloud project.
- Deploy Cloud Function.
- Reserve a Cloud Run hostname.
- Create an OpenAPI document that describes your API
- Configure the routes to your Cloud Functions.
- Deploy the OpenAPI document to create a managed service.
- Build a new ESPv2 Docker image with your Endpoints service configuration.
- Deploy the ESPv2 container onto Cloud Run.
- Grant ESPv2 IAM permissions.
- Invoke a function.
- Track activity to your functions.
- Clean up.

Vertex AI API

What is Vertex AI?



What is Vertex AI?

- Vertex AI brings together the Google Cloud services for building ML under one, unified UI and API.
- In Vertex AI, you can now easily train and compare models using AutoML or custom code training and all your models are stored in one central model repository.
- These models can now be deployed to the same endpoints on Vertex AI.

Vertex AI Components

Machine learning workflow step	Recommended tools and products
ML environment setup	•Vertex AI Workbench user-managed notebooks •Vertex SDK for Python
Data processing	•BigQuery •Dataflow •Dataproc •Managed datasets •TensorFlow Extended
Model monitoring	•Vertex Explainable AI •Vertex Model Monitoring

Vertex AI Components

Machine learning workflow step	Recommended tools and products
ML development	•BigQuery •Cloud Storage •Vertex AI Workbench user-managed notebooks •Vertex Data Labeling •Vertex Explainable AI •Vertex AI Feature Store •Vertex AI TensorBoard •Vertex Training
Artifact organization	•Artifact Registry

Vertex AI Components

Machine learning workflow step	Recommended tools and products
Operationalized training	•Cloud Storage •PyTorch •TensorFlow Core •Vertex AI Feature Store •Vertex AI Pipelines •Vertex Training
Model deployment and serving	•Vertex Prediction
ML workflow orchestration	•Kubeflow Pipelines •TensorFlow Extended •Vertex AI Pipelines

Vertex AI

- Vertex AI integrates with widely used open source frameworks such as TensorFlow, PyTorch, and scikit-learn, along with supporting all ML frameworks and artificial intelligence branches via custom containers for training and prediction.

Vertex AI

Vertex AI unifies definitions/implementations of four concepts:

- A dataset can be structured or unstructured. It has managed metadata including annotations, and can be stored anywhere on GCP.
- A training pipeline is a series of containerized steps that can be used to train an ML model using a dataset. The containerization helps with generalization, reproducibility, and auditability.
- A model is an ML model with metadata that was built with a Training Pipeline or directly loaded
- An endpoint can be invoked by users for online predictions and explanations. It can have one or more models, and one or more versions of those models, with disambiguation carried out based on the request.

End to End Data w/ Vertex AI Workbench

- Through Vertex AI Workbench, Vertex AI is natively integrated with BigQuery, Dataproc and Spark.
- You can use BigQuery ML to create and execute machine learning models in BigQuery using standard SQL queries on existing business intelligence tools and spreadsheets.
- Export datasets from BigQuery directly into Vertex AI Workbench and run your models from there.
- Use Vertex Data Labeling to generate highly accurate labels for your data collection.

Vertex AI Workbench

Overview of Vertex AI Workbench

Introduction to Vertex AI Workbench

Choose a notebook solution

REST API reference

RPC API reference

Pricing

Managed notebooks

Use the Cloud Console to create a managed notebooks instance

Use the Cloud Console to schedule a managed notebooks run

Run a managed notebooks instance on a Dataproc cluster

User-managed notebooks

Use the Cloud Console to create a user-managed notebooks instance

Manage hardware accelerators for a user-managed notebooks instance

Use a user-managed notebooks instance within a service perimeter

Monitor health status

Vertex Explainable AI

Vertex Explainable AI helps you understand your model's outputs for classification and regression tasks.

Vertex AI supports Vertex Explainable AI for the following types of models:

- AutoML image models (classification models only)
- AutoML tabular models (classification and regression models only)
- Custom-trained models based on tabular data
- Custom-trained models based on image data

Vertex AI Model Monitoring

Models used in production require continuous monitoring to ensure that they continue to perform as expected. After you deploy a model in production, the input data provided to the model for predictions often changes. When the prediction input data deviates from the data that the model was trained on, the performance of the model can deteriorate, even though the model itself hasn't changed.

Model Monitoring supports feature skew and drift detection for categorical and numerical features.

Vertex AI Feature Store

Vertex AI Feature Store provides a centralized repository for organizing, storing, and serving ML features. Using a central featurestore, enables an organization to efficiently share, discover, and re-use ML features at scale, which can increase the velocity of developing and deploying new ML applications.

Vertex AI Feature Store is a fully managed solution, which manages and scales the underlying infrastructure such as storage and compute resources. This solution means that data scientists can focus on the feature computation logic instead of worrying about the challenges of deploying features into production.

Vertex AI SDK for Python

The Python client library for Vertex AI is now called the Vertex AI SDK for Python.

With the release of version 0.7 (Preview), the Vertex AI SDK for Python provides two levels of support:

- High-level aiplatform library is designed to simplify common data science workflows by using wrapper classes and opinionated defaults.
- Lower-level aiplatform.gapic library remains available for those times when you need more flexibility or control, or for those methods that are not supported by the high-level library.

Vertex AI Vizier

Vertex AI Vizier is a black-box optimization service that helps you tune hyperparameters in complex machine learning (ML) models. When ML models have many different hyperparameters, it can be difficult and time consuming to tune them manually. Vertex AI Vizier optimizes your model's output by tuning the hyperparameters for you.

Black-box optimization is the optimization of a system that meets either of the following criteria:

- Doesn't have a known objective function to evaluate.
- Is too costly to evaluate by using the objective function, usually due to the complexity of the system.

Vertex AI Vizier Differentiator

Vertex AI Vizier is an independent service for optimizing complex models with many parameters.

- It can be used for both ML and non-ML use cases.
- It can be used with Training jobs or with other systems (even multi-cloud). Hyperparameter tuning for custom training is a built-in feature that uses Vizier for training jobs.
- It helps determine the best hyperparameter settings for an ML model.

Vertex AI Vizier Use Cases

In the following scenarios, Vertex AI Vizier helps tune hyperparameters to optimize a model or tune parameters to optimize an outcome:

- Optimize the learning rate, batch size, and other hyperparameters of a neural network recommendation engine.
- Optimize usability of an application by testing different arrangements of user interface elements.
- Minimize computing resources for a job by identifying an ideal buffer size and thread count.
- Optimize the amounts of ingredients in a recipe to produce the most delicious version.

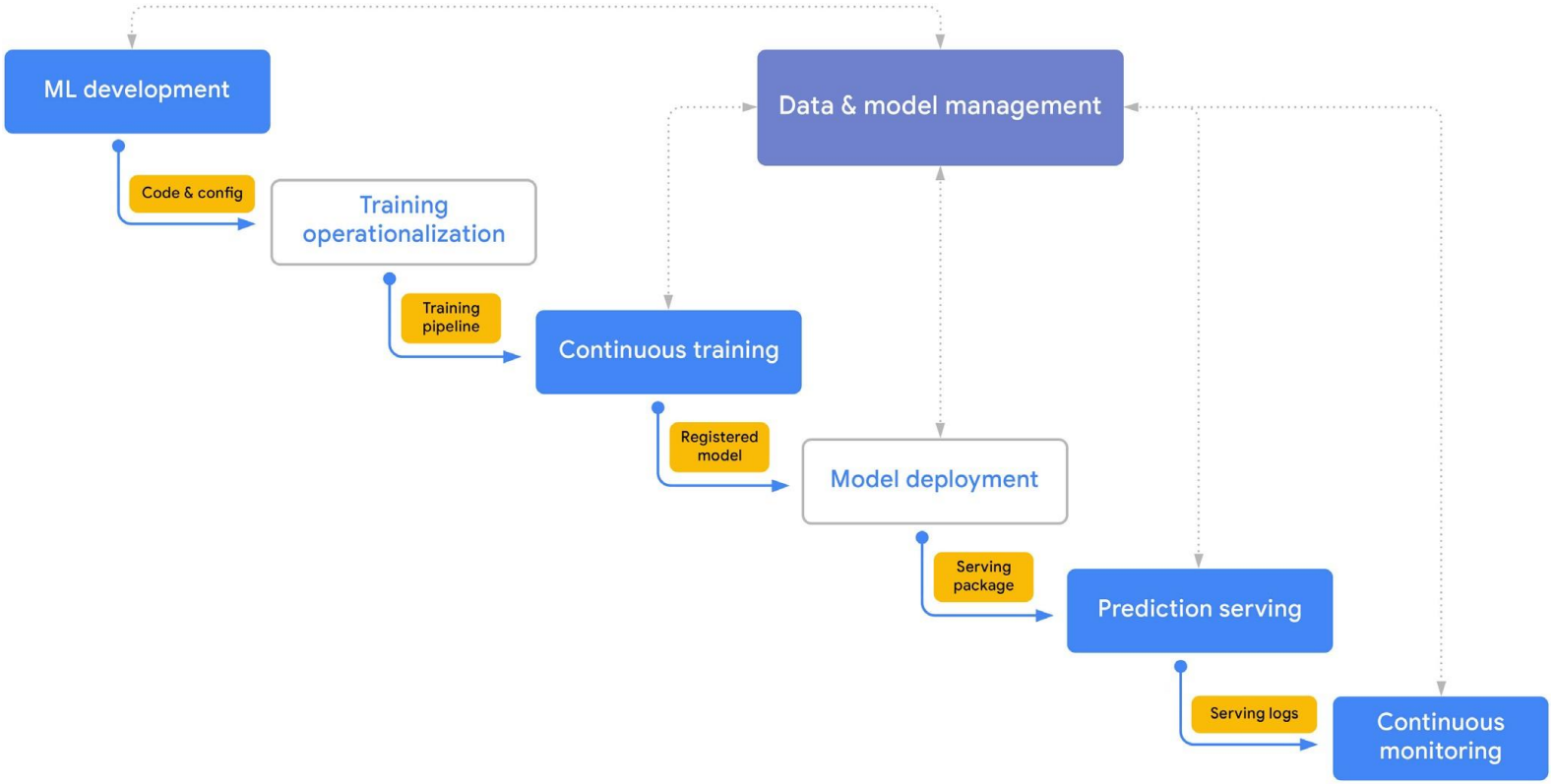
ML Ops

ML development
Training operationalization
Continuous training
Model deployment
Prediction serving
Continuous monitoring
Data & model management

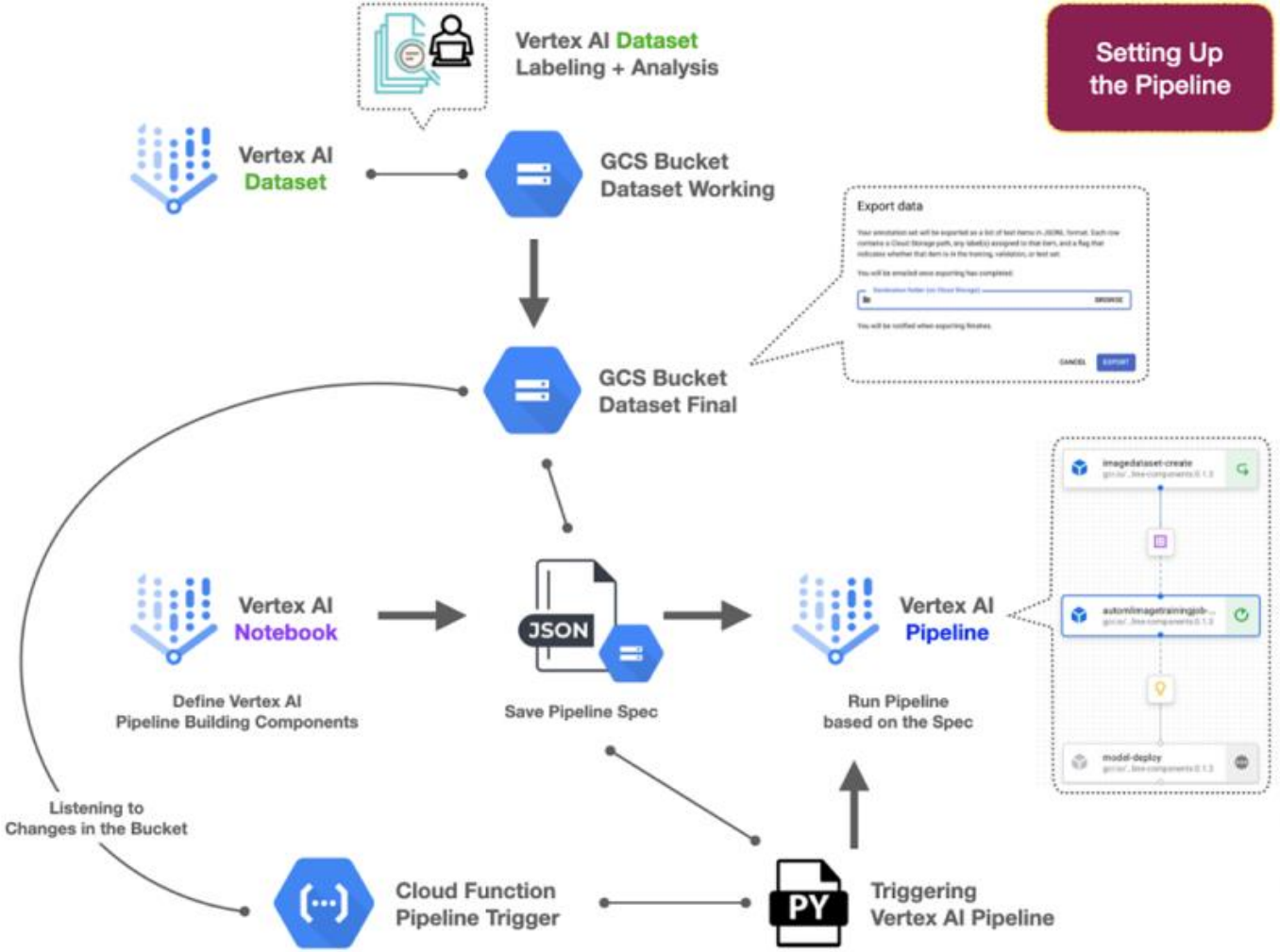
ML Ops Processes

- **ML development** concerns experimenting and developing a robust and reproducible model training procedure (training pipeline code)
- **Training operationalization** concerns automating the process of packaging, testing, and deploying training pipelines.
- **Continuous training** concerns repeatedly executing the training pipeline in response to new data or to code changes, or on a schedule, potentially with new training settings.
- **Model deployment** concerns packaging, testing, and deploying a model to a serving environment for online experimentation and production serving.
- **Prediction serving** is about serving the model that is deployed in production for inference.
- **Continuous monitoring** is about monitoring the effectiveness and efficiency of a deployed model.
- **Data and model management** is a central, cross-cutting function for governing ML artifacts to support audit- ability, traceability, and compliance.

ML Ops End-to-End Flow



MLOps Pipeline



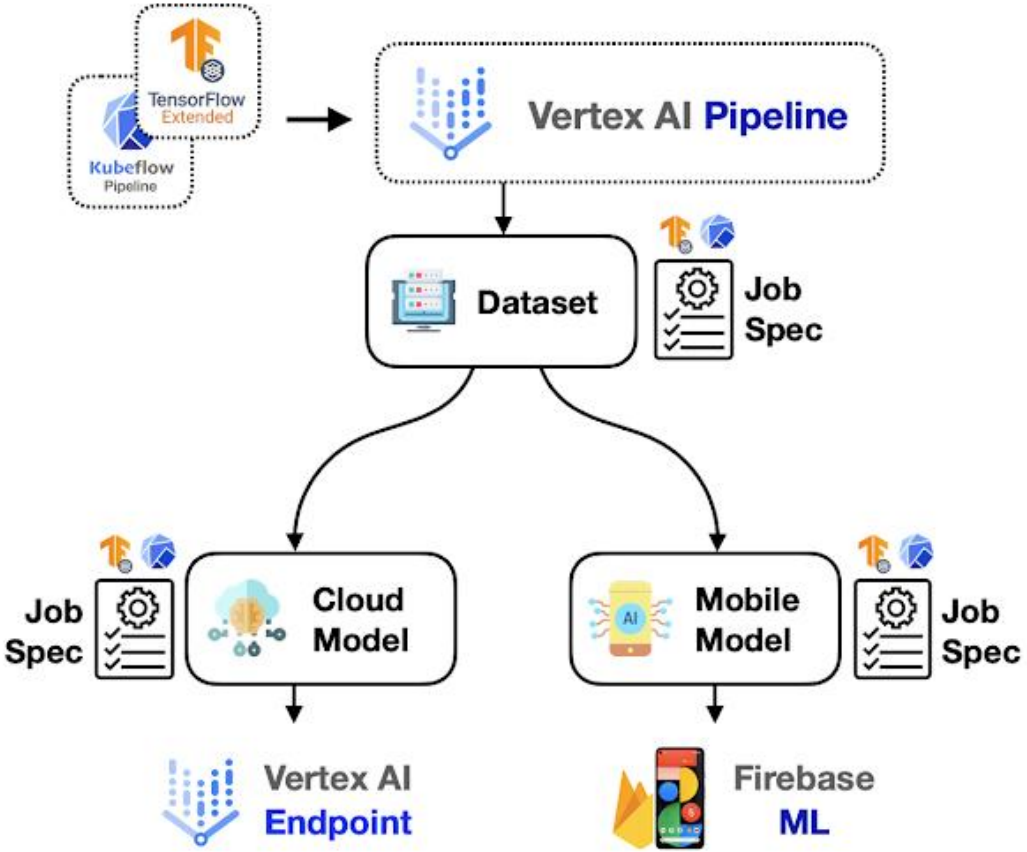
Vertex In Action

Actionable Insight

When you run a pipeline using Vertex AI Pipelines, the artifacts and metadata of your pipeline run are stored using Vertex ML Metadata. You can use this metadata help answer questions like the following:

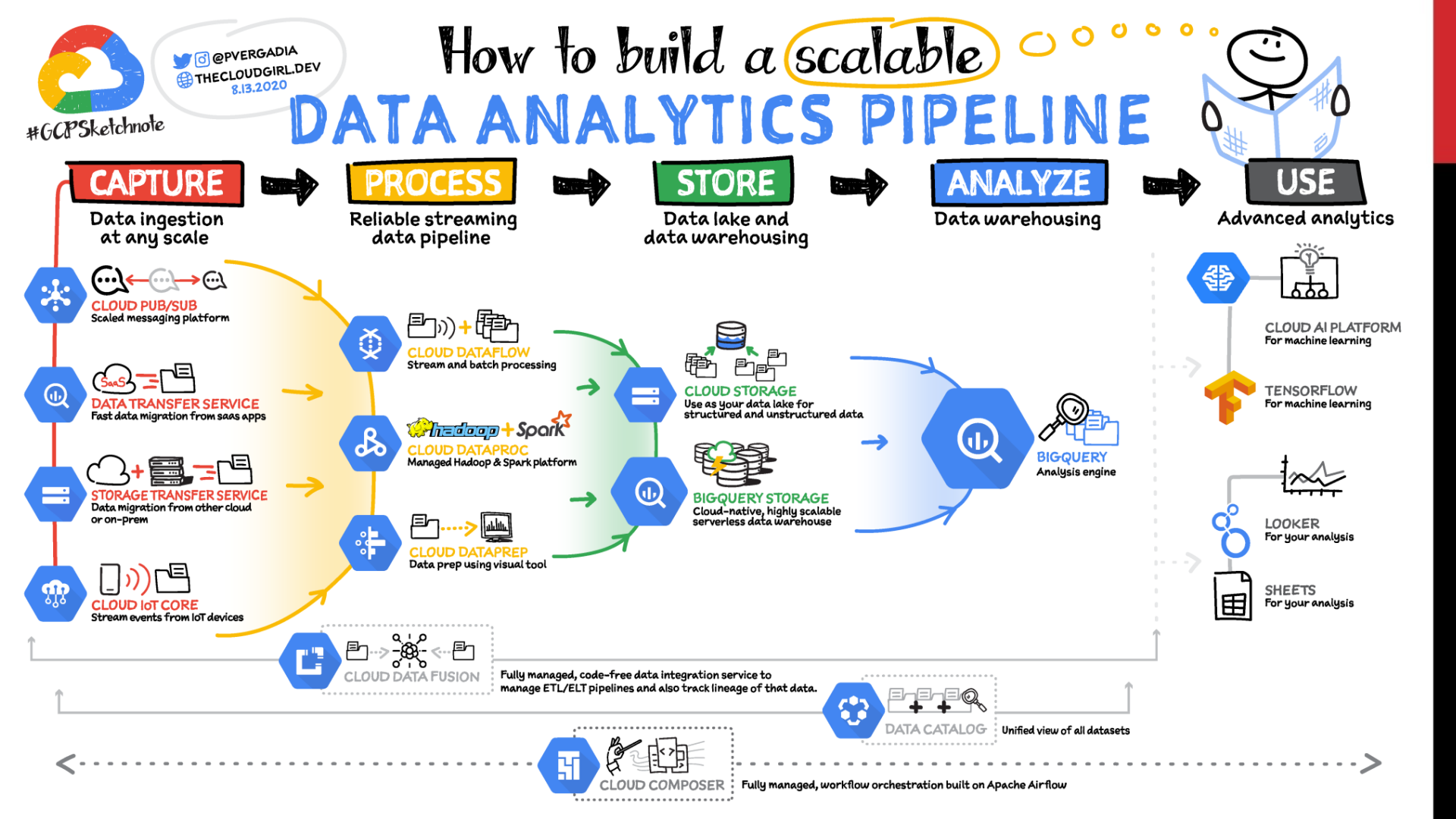
- Why did a certain pipeline run produce an especially accurate model?
- Which pipeline run produced the most accurate model, and what hyperparameters were used to train the model?
- Depending on the steps in your pipeline, you may be able to use Vertex ML Metadata to answer system governance questions. For example, you could use metadata to determine which version of your model was in production at a given point in time.

Vertex AI Pipeline in Action



Understanding Vertex AI Pipelines

- Pipelines allow you to automate, monitor, and experiment with interdependent parts of a ML workflow.
- ML Pipelines are portable, scalable, and based on containers.
- Each individual part of your pipeline workflow (for example, creating a dataset or training a model) is defined by code. This code is referred to as a component.
- Each instance of a component is called a step.


#GCPSketchnote
@PVERGADIA
THECLOUDGIRL.DEV
8.13.2020
How to build a scalable
DATA ANALYTICS PIPELINE
CAPTURE
Data ingestion at any scale
PROCESS
Reliable streaming data pipeline
STORE
Data lake and data warehousing
ANALYZE
Data warehousing
USE
Advanced analytics
CLOUD PUB/SUB
Scaled messaging platform
DATA TRANSFER SERVICE
Fast data migration from saas apps
STORAGE TRANSFER SERVICE
Data migration from other cloud or on-prem
CLOUD IoT CORE
Stream events from IoT devices
CLOUD DATAFLOW
Stream and batch processing
hadoop + Spark
CLOUD DATAPROC
Managed Hadoop & Spark platform
CLOUD DATAPREP
Data prep using visual tool
CLOUD STORAGE
Use as your data lake for structured and unstructured data
BIGQUERY STORAGE
Cloud-native, highly scalable serverless data warehouse
BIGQUERY
Analysis engine
CLOUD AI PLATFORM
For machine learning
TENSORFLOW
For machine learning
LOOKER
For your analysis
SHEETS
For your analysis
CLOUD DATA FUSION
Fully managed, code-free data integration service to manage ETL/ELT pipelines and also track lineage of that data.
DATA CATALOG
Unified view of all datasets
CLOUD COMPOSER
Fully managed, workflow orchestration built on Apache Airflow

Orchestration of ML Pipelines

- To orchestrate your ML workflow on Vertex AI Pipelines, you must first describe your workflow as a pipeline.
- ML pipelines are portable and scalable ML workflows that are based on containers.
- ML pipelines are composed of a set of input parameters and a list of steps. Each step is an instance of a pipeline component.

Utilizing Vertex AI Pipelines

You can use ML pipelines to:

- Apply MLOps strategies to automate and monitor repeatable processes.
- Experiment by running an ML workflow with different sets of hyperparameters, number of training steps or iterations, etc.
- Reuse a pipeline's workflow to train a new model.

You can use Vertex AI Pipelines to run pipelines that were built using the Kubeflow Pipelines SDK or TensorFlow Extended.

Understanding Pipeline Components

- Pipeline components are self-contained sets of code that perform one part of a pipeline's workflow, such as data preprocessing, data transformation, and training a model.
- Components are composed of a set of inputs, a set of outputs, and the location of a container image. A component's container image is a package that includes the component's executable code and a definition of the environment that the code runs in.
- You can build custom components or you can reuse prebuilt components. To use features of Vertex AI like AutoML in your pipeline, use the Google Cloud pipeline components.

Understanding Pipeline Workflow

- Each pipeline step (component instance) has inputs, outputs, and an associated container image. These inputs and outputs may depend on other component steps.
- Input and output dependencies between pipeline workflow steps create a directed acyclic graph (DAG).
- The pipelines SDK you use determines this workflow graph based on these dependencies and runs steps when inputs and outputs are available (concurrently or sequentially).

Sample Pipeline Workflow

- **Ingest data**: This step loads training data into the pipeline.
- **Preprocess data**: This step preprocesses the ingested training data.
- **Train model**: This step uses the preprocessed training data to train a model.
- **Evaluate model**: This step evaluates the trained model.
- **Deploy**: This step deploys the trained model for predictions.

ML Artifact Lineage

- Each pipeline run produces metadata and ML artifacts, such as models or datasets. Artifact lineage describes all the factors that resulted in an artifact, such as training data or hyperparameters used for model training.
- By using artifact lineage you can understand differences in performance or accuracy over several pipeline runs.
- Vertex ML Metadata stores artifacts and metadata for pipelines run using Vertex AI Pipelines.

Sample ML Artifact Lineage

A model's lineage could include the following:

- The training, test, and evaluation data used to create the model.
- The hyperparameters used during model training.
- The code that was used to train the model.
- Metadata recorded from the training and evaluation process, such as the model's accuracy.
- Artifacts that descend from this model, such as the results of batch predictions.

Versioning the Model

- This model is versioned and rebuilt when there is new data, on a schedule, or when the code changes (ML Ops).
- Machine learning (ML) is about rapid experimentation and iteration, and without keeping track of your modeling history you won't be able to learn much.
- Versioning lets you keep track of all of your models, how well they've done, and what hyperparameters you used to get there.
- Versioning is also an important component of AI/ML governance.

Evaluation

- You should always evaluate a model to determine if it will do a good job of predicting the target on new and future data.
- Because future instances have unknown target values, you need to check the accuracy metric of the ML model on data for which you already know the target answer, and use this assessment as a proxy for predictive accuracy on future data.
- To properly evaluate a model, you hold out a sample of data that has been labeled with the target (ground truth) from the training datasource.
- We have to be aware of rewards models that can “remember” training data

Deployment

- Deployment is the method by which you integrate a machine learning model into an existing production environment to make practical business decisions based on data.
- It is one of the last stages in the machine learning life cycle and can be one of the most cumbersome.
- Often, an organization’s IT systems are incompatible with traditional model-building languages, forcing data scientists and programmers to spend valuable time and brainpower rewriting them.
- Model deployment is one of the most difficult processes of gaining value from machine learning.

Cloud Functions and Operations Suite

Cloud Functions & Operations Suite

Writing and viewing logs

- Cloud Logging captures, stores, and analyzes Cloud Functions logs.

Reporting errors

- Error Reporting captures specially formatted error logs and displays them in the Error Reporting dashboard.

Viewing monitored metrics

- Cloud Monitoring records metrics regarding the execution of Cloud Functions.

Writing Runtime Logs

Cloud Functions includes simple runtime logging by default. Logs written to stdout or stderr will appear automatically in the Cloud Console. For more advanced logging, use the Cloud Logging Client Libraries.

Note: There is typically a slight delay between when log entries are created and when they show up in Cloud Logging.

As a default, the log payload is a simple text string, as shown in the following snippets. This string is stored in textPayload.

Writing Structured Logs

- The default text logs described do not have an associated log level.
- If you want to include log levels, or other specific fields, in your log entries, you can use structured data instead, in the form of a single line of serialized JSON.
- This line is picked up and parsed by Cloud Logging and is placed into jsonPayload instead of textPayload.
- The data is constructed as a JSON dictionary.

Viewing Cloud Function Logs

Logs for Cloud Functions are viewable in the Cloud Logging UI, and via the gcloud command-line tool.

- To view logs with the gcloud tool, use the logs read command:
  gcloud functions logs read

- To view the logs for a specific function, provide the function name as an argument:
  gcloud functions logs read FUNCTION_NAME

Error Reporting

- Errors can be emitted from functions or caught depending on their intended use.
- If you would like more fine-grained error reporting, you can use the Error Reporting client libraries.
- You can view the reported errors in Error Reporting in the Cloud Console. You can also see the errors reported from a particular function when you select it from the list of functions in the Cloud Console.
- Any uncaught exception will appear in the Error Reporting

Metric Viewing

- You can view your Cloud Functions with their execution times, execution counts, and memory usage in the Cloud Console. These metrics are also available in Cloud Monitoring, where you can set up custom alerting on these metrics. See the Cloud Monitoring documentation for more information.

- You can view the metrics for API calls in the API overview page of the Cloud Console.

- You can set up alerting on quota errors in Cloud Monitoring by filtering executions that have an out of quota value for the STATUS metric label.

Cloud Audit Logging

- Google Cloud services write audit logs to help you answer the questions, "Who did what, where, and when?" within your Google Cloud resources.

- Your Google Cloud projects contain only the audit logs for resources that are directly within the Cloud project.

- Other Google Cloud resources, such as folders, organizations, and billing accounts, contain the audit logs for the entity itself.

Cloud Function Audit Logs

The following types of audit logs are available for Cloud Functions:

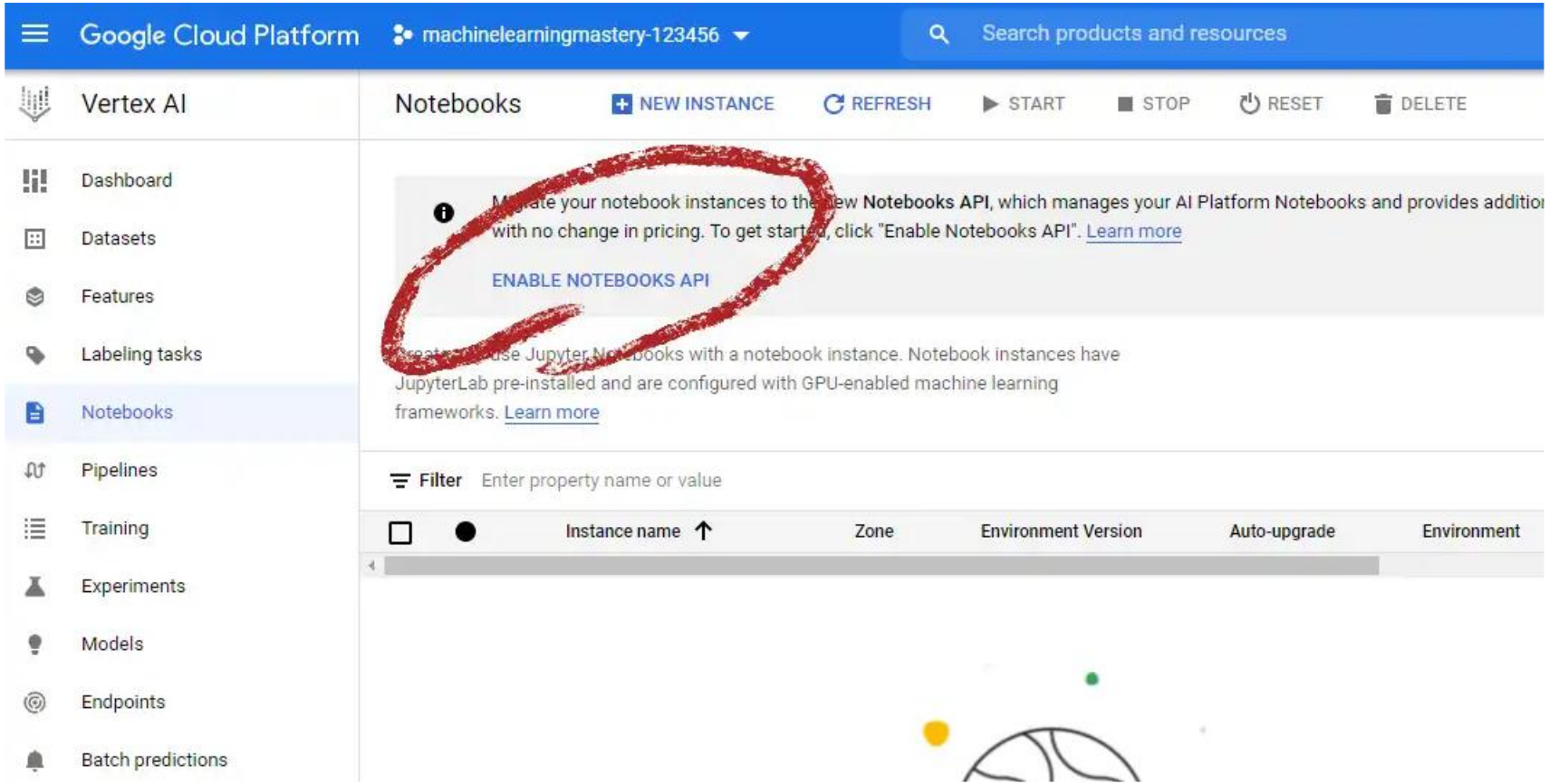
Admin Activity audit logs

- Includes "admin write" operations that write metadata or configuration information.
- You can't disable Admin Activity audit logs.

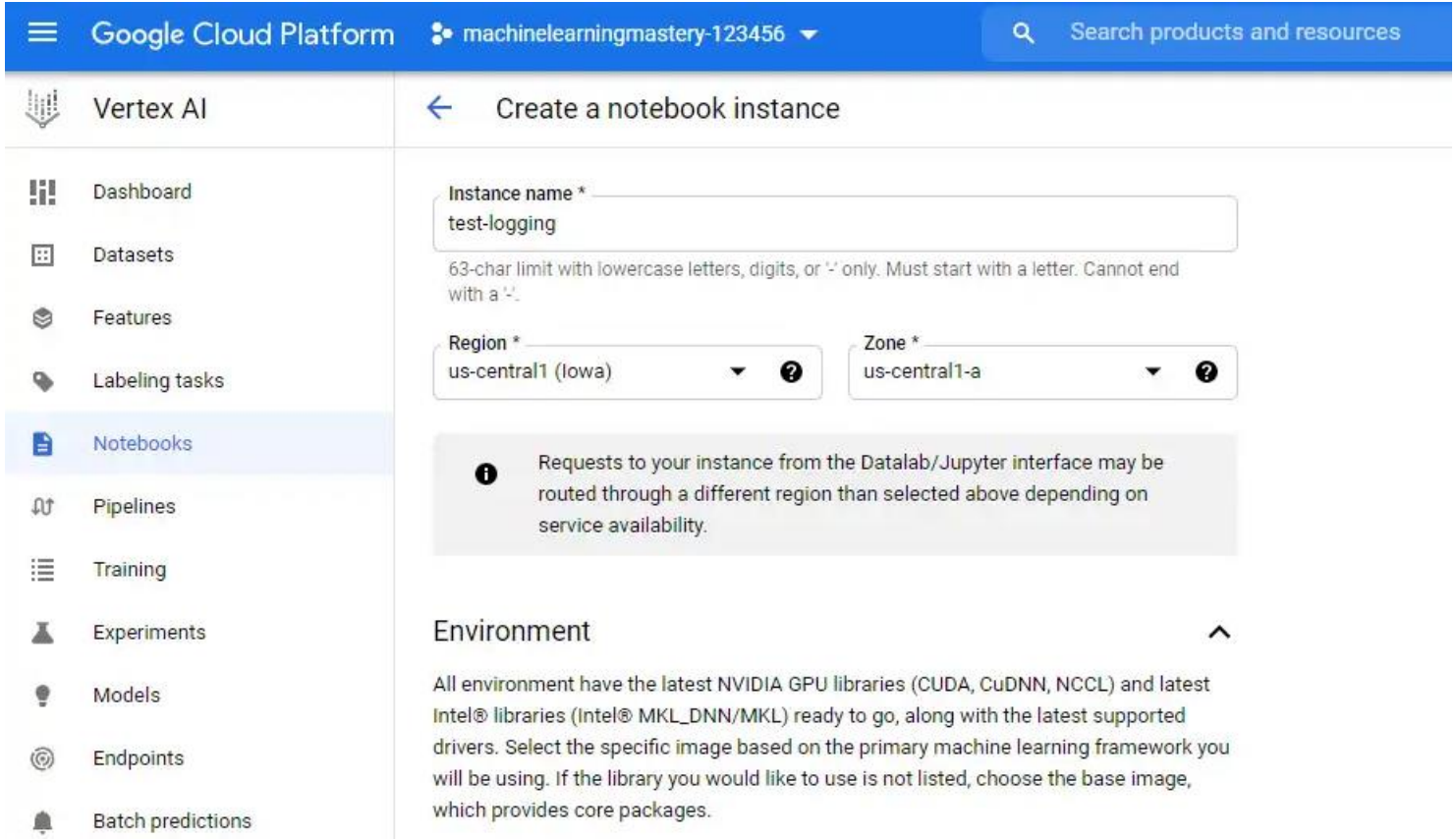
Data Access audit logs

- Includes "admin read" operations that read metadata or configuration information. Also includes "data read" and "data write" operations that read or write user-provided data.
- To receive Data Access audit logs, you must explicitly enable them.

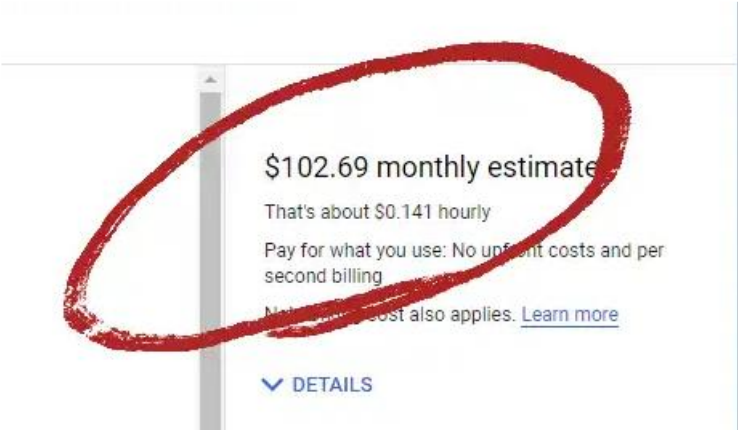
Logging in Vertex AI: Enable Notebooks



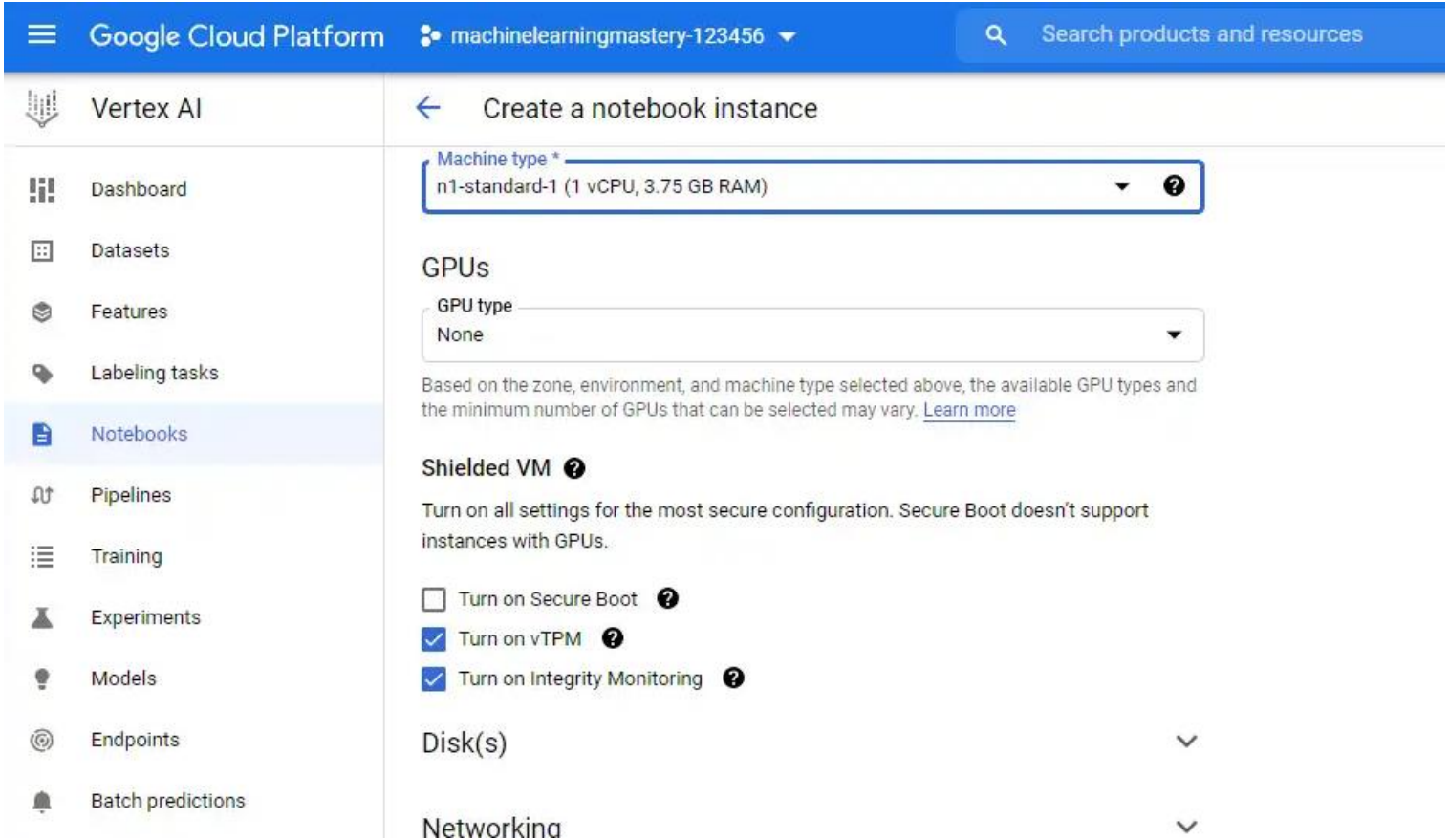
Logging in Vertex AI: Create Notebook



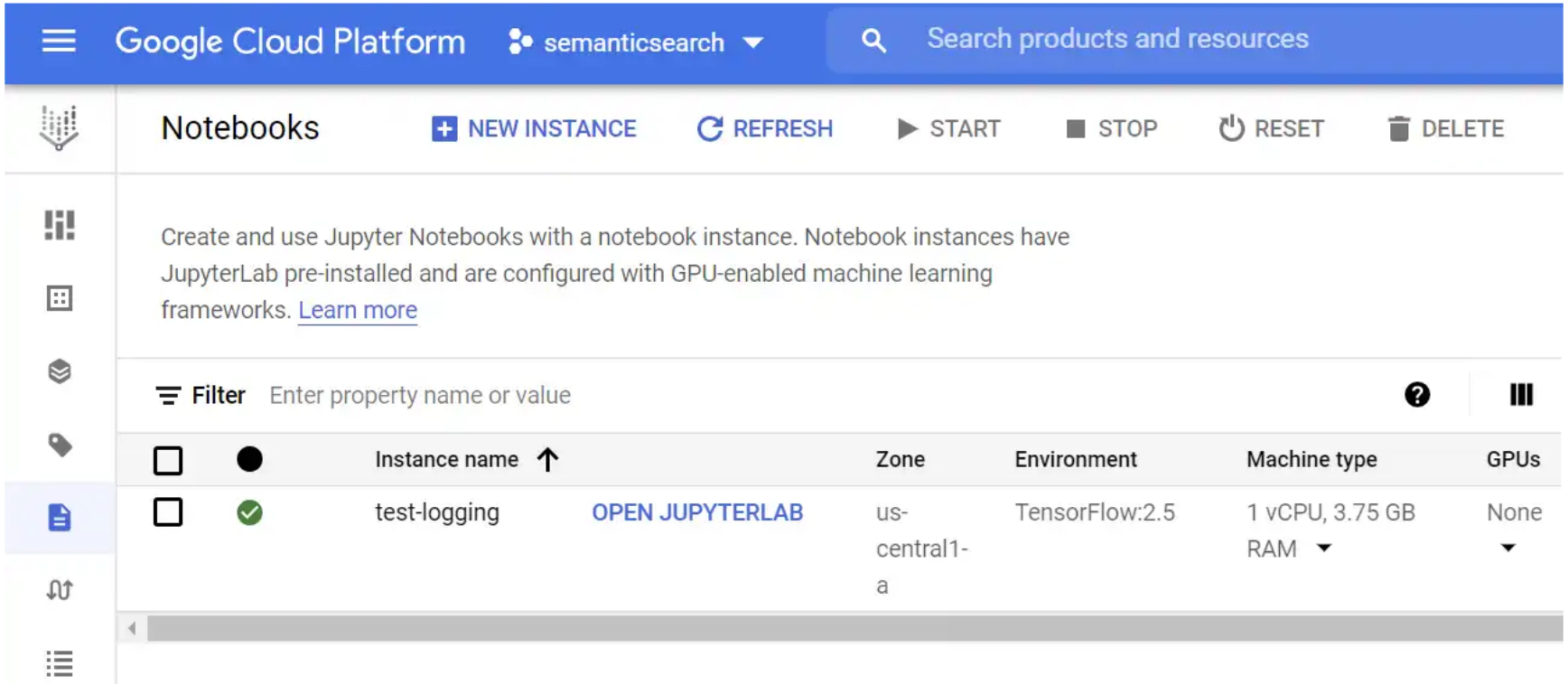
Logging in Vertex AI: Resource Costs



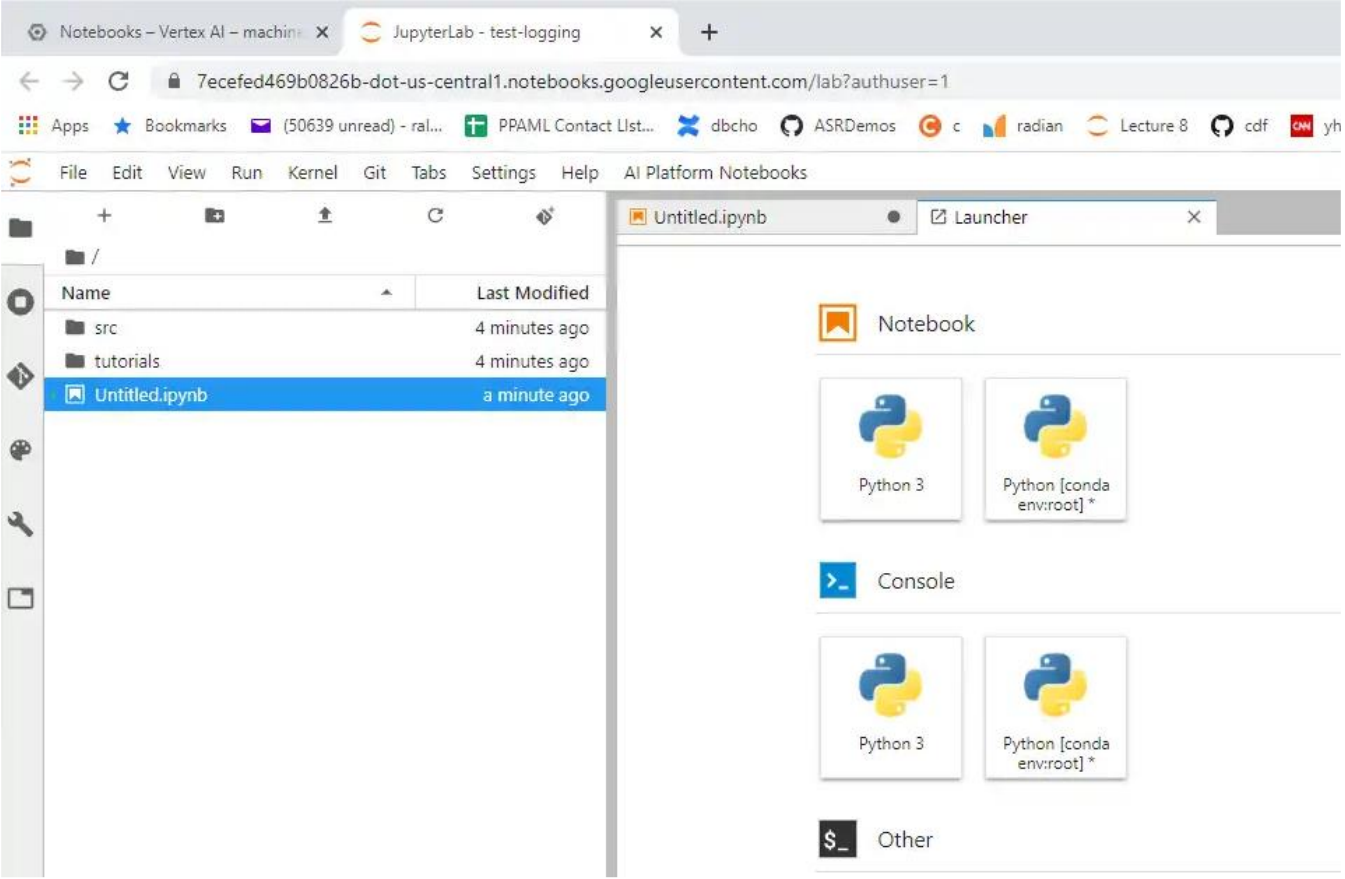
Logging in Vertex AI: Select Options



Logging in Vertex AI: Environment



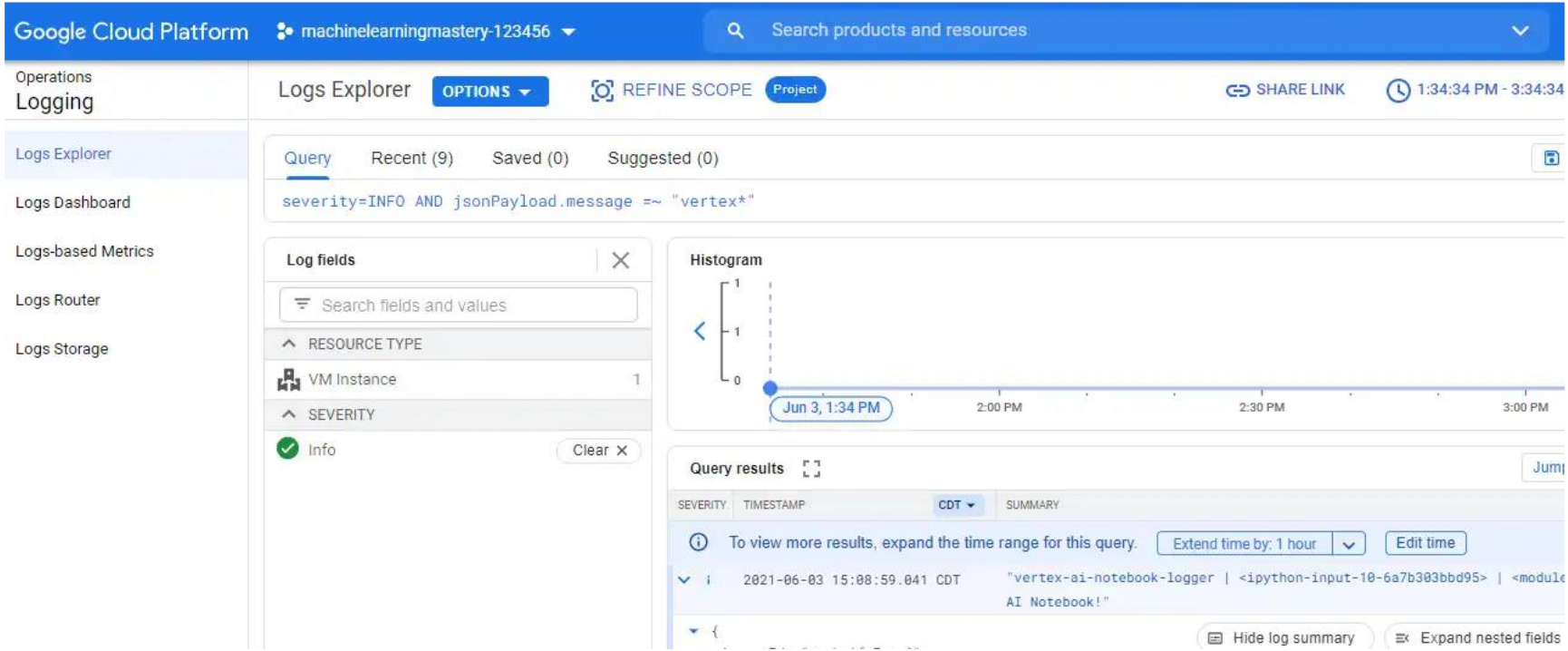
Logging in Vertex AI: Environment



Logging in Vertex AI: Code

```
# Code that sets logging
import logging
import google.cloud.logging_v2 as logging_v2
from os import environ

client = logging_v2.client.Client()

# set the format for the log
google_log_format= logging.Formatter(
fmt='%(name)s | %(module)s | %(funcName)s | %(message)s',
datefmt='%Y-%m-$dT%H:%M:%S')


handler = client.get_default_handler()
handler.setFormatter(google_log_format)

cloud_logger = logging.getLogger("vertex-ai-notebook-logger")
cloud_logger.setLevel("INFO")
```

Logging in Vertex AI: Log Explorer



Logging in Vertex AI: Cleanup

Finally, remember to go back into the notebook console within Vertex AI to:

- Select the "instance name" that you created (such as "test-logging")
- Click the delete icon at the top of the console in order to delete the instance

When successful, you should see something on the console that states " You don't have any notebook instances in this project yet."

Cloud Monitor

Cloud Monitoring

- View system metrics collected automatically from Google Cloud services
- Enable SRE best practices extensively used by Google including SLOs and SLIs
- Collect application metrics from workloads on GKE with the managed metrics-collection pipeline
- Use the Ops Agent to collect application metrics and deep system metrics from VMs
- Create custom dashboards and create alerts that integrate with incident management tools

Alerting Policies and Uptime Checks

To be notified when the performance of a data service doesn't meet criteria you define, **create an alerting policy**.

To be notified when a deployed service isn't accessible or when it isn't responding correctly, **configure an uptime check** and attach an alerting policy:

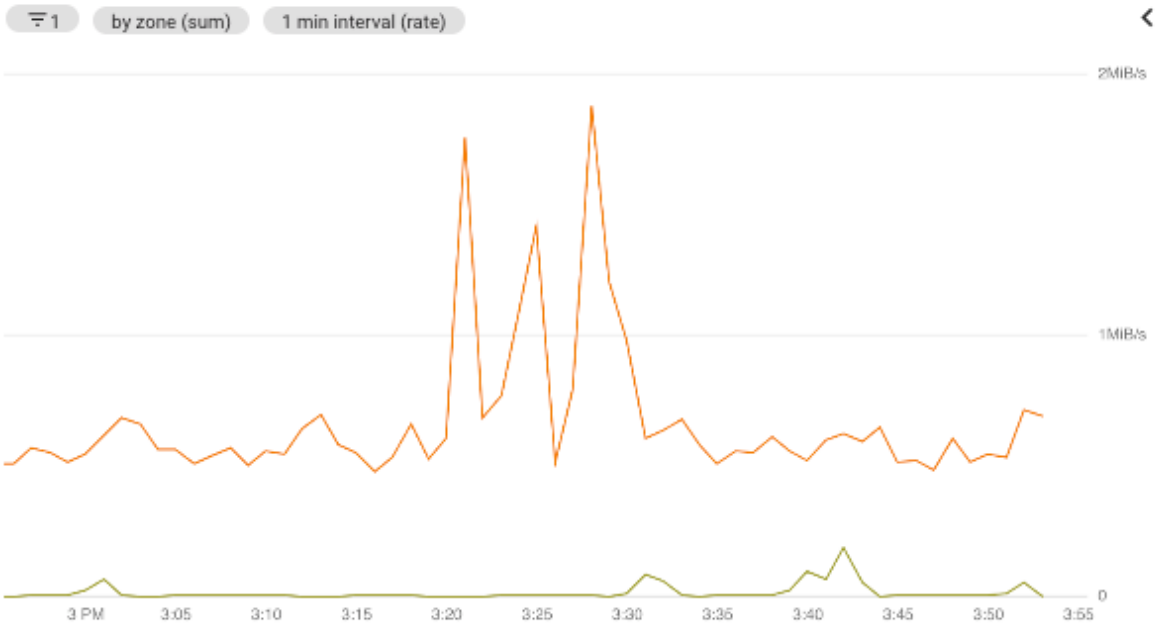
- The **uptime check** periodically probes your service and stores the success and latency of that probe as metric data.
- The **alerting policy** monitors the success status of the uptime check and notifies you when a probe fails.

Cloud Monitoring Visualization

You can chart and monitor any (numeric) metric data that your Google Cloud project collects, including the following:

- System metrics generated by Google Cloud services.
- System and application metrics
- Custom metrics with the CM API or OpenTelemetry.
- Logs-based metrics, which collect numeric information about the logs

Cloud Monitoring Visualization



Metric	zone	Value
Disk read bytes	us-central1-a	0
Disk write bytes	us-central1-a	703.260KiB/s

Time Series & Metrics

A **metric** describes something that is measured. Examples of metrics include a VM's CPU utilization and the percentage of a disk that is used.

A **time series** is a data structure that contains time-stamped measurements of a metric and information about the source and meaning of those measurements.

Time Services Visualization

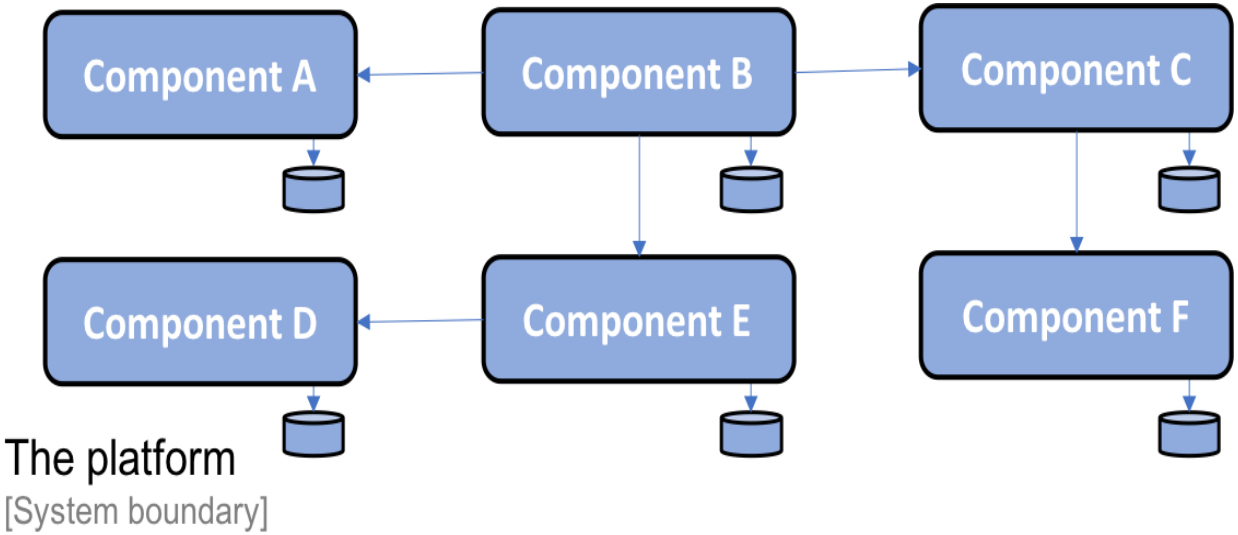
Cloud Monitoring provides you with multiple ways to visualize your time-series data:

- Google Cloud dashboards: Cloud Monitoring automatically creates these dashboards based on the resources used by your Google Cloud project.
- Custom dashboards: You create or install these dashboards.
- Charts: You can add charts to a custom dashboard or you can use Metrics Explorer

Cloud Monitoring & Vertex AI

- Vertex AI exports metrics to Cloud Monitoring. Vertex AI also shows some of these metrics in the Vertex AI Cloud Console.
- You can use Cloud Monitoring to create dashboards or configure alerts based on the metrics.
- For example, you can receive alerts if a model's prediction latency in Vertex AI gets too high.

Title


Component A
Component B
Component C
Component D
Component E
Component F
The platform
[System boundary]

AI, ML and Advanced Analytics on GCP

17

Making AI Reality

Making AI Reality

“We need to be an AI-enabled company.” Replace the “AI” with any technology from history and this comment becomes a common refrain across businesses lured by the promises of new technology and fueled by FOMO (a fear of missing out).

To paraphrase Roy Amara, we overestimate the impact of a new technology early on. When it falls short of our expectations, our disappointment means we are less willing to adopt it when it is truly ready.

The reality is that often it is only when we change how we operate and organise that many new technologies come into their own.

GCP ML Options

- TensorFlow (for data scientist): This is an option for those who want to work with ML from scratch. It is a software library that's developed and open-sourced by Google.
- ML Engine (for data scientist): This is an option for those who want to train their own models, but who use Google for training and predictions.
- Pretrained ML models (for developer): This is an option for those who want to leverage ML without having any knowledge of it. It allows Google-developed models to be used to perform predictions.
- AutoML (for developer): This is an option for those who want to leverage ML without having any knowledge of it, and where the pretrained models are not fit for purpose.

Upskill

- Making a concerted effort to upskill
- Share what you learn
- Find highly skilled mentors in AI, in or out of the organization
- Understand the opportunity
- Five whys?
- Build a community
- Recognize that with anything new there is a constant cycle of change
- Rinse and repeat

AI & Agile & DevSecOps

- Successfully adopting AI hinges on your ability to be an agile organisation.
- In other words, your organization must be willing and able to experiment and learn fast using cross-functional teams.
- Like any effective technology or initiative, the power of AI comes through its application to real-world problems and the turning of insights into action.
- Ultimately this requires changes to business processes, operating norms, and even the roles employees fill.

Ubiquitous Data Across the Org

- Foundational to sophisticated AI/ML/Analytics workstreams is the need to collect, clean, and make accessible the underlying data.
- Improving data quality at the source or appointing data stewards is helpful, but foundational elements are equally as critical.
- Having a cost-effective highly scalable data lake is a good starting point.
- Increasing data literacy and data accessibility across your organisation.

Mythbusters: Doing AI/ML/Analtics right means doing it fast.

- Effectively scaling AI typically doesn't happen in a matter of hours, days, weeks, or even months.
- Successfully scaling AI means doing it purposefully...and that can take years.
- Accenture conducted a global study—surveying 1,500 executives across 16 industries—and found that companies that have had the most success scaling their AI efforts are 65% more likely than their less successful peers to set timelines of one to two years.

Mythbusters: Money is No Object with AI Investment

- Companies that scale AI successfully actually spend less than their counterparts. They're more careful and thoughtful in their approach to AI. They rely on clearly defined strategies and operating models to make enterprise-wide AI a reality. But they don’t spend more.

- Scaling AI effectively costs less than you might think because the technology itself is already being delivered on a massive scale today by major technology providers. For example, chatbots are fairly inexpensive to implement yet can result in major ROI.

Mythbusters: More Data is Better

- Companies often get off track when they focus on data issues before determining whether they have the right business processes in place.
- When they nail the business processes first, it becomes easier to approach data issues strategically, identifying and structuring the information necessary for AI initiatives.

Mythbusters: IT Should Own AI/ML/Analytics

- Companies that scale AI successfully have Business Leaders who actively champion the technology themselves, focus on a specific business outcome, and make clear that AI is a priority for the entire organization.

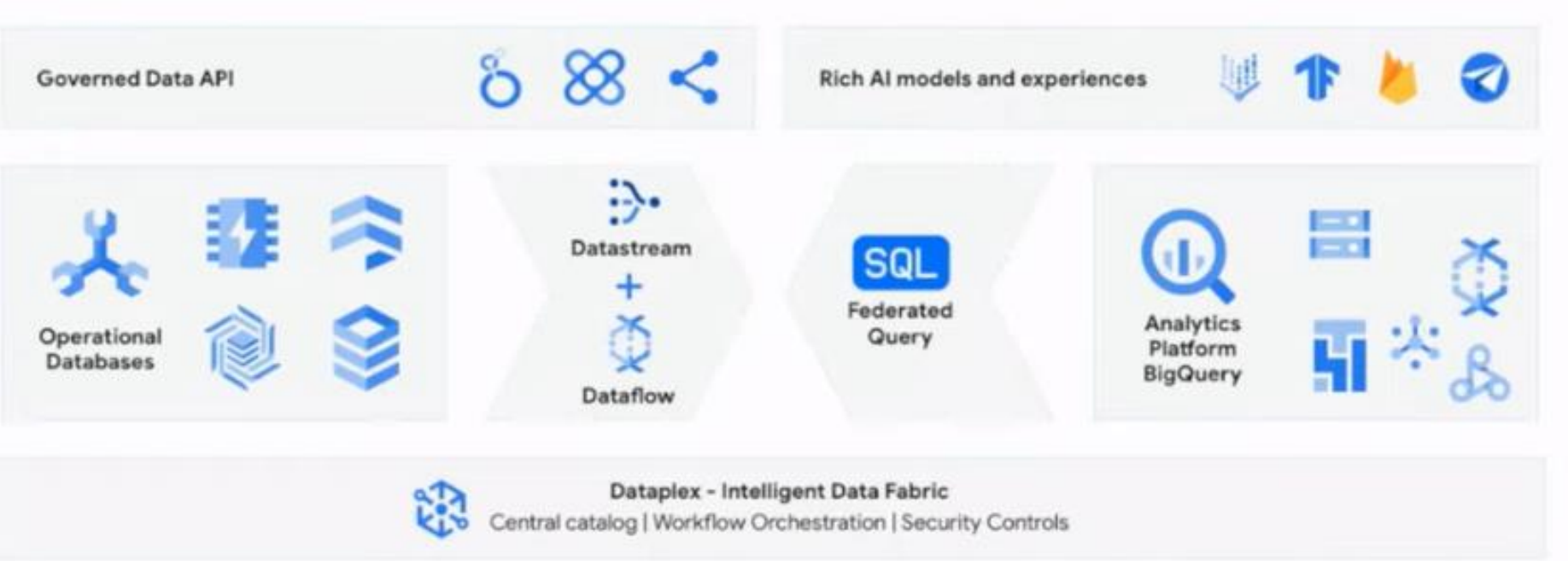
- What's more, successful AI efforts are often undertaken by an interdisciplinary team. They involve not just engineers and IT professionals, but also data scientists and business experts to ensure that the technical architecture of the initiative meets the company's needs.

Mythbusters: Big Companies Stink at AI/ML/Advanced Analytics

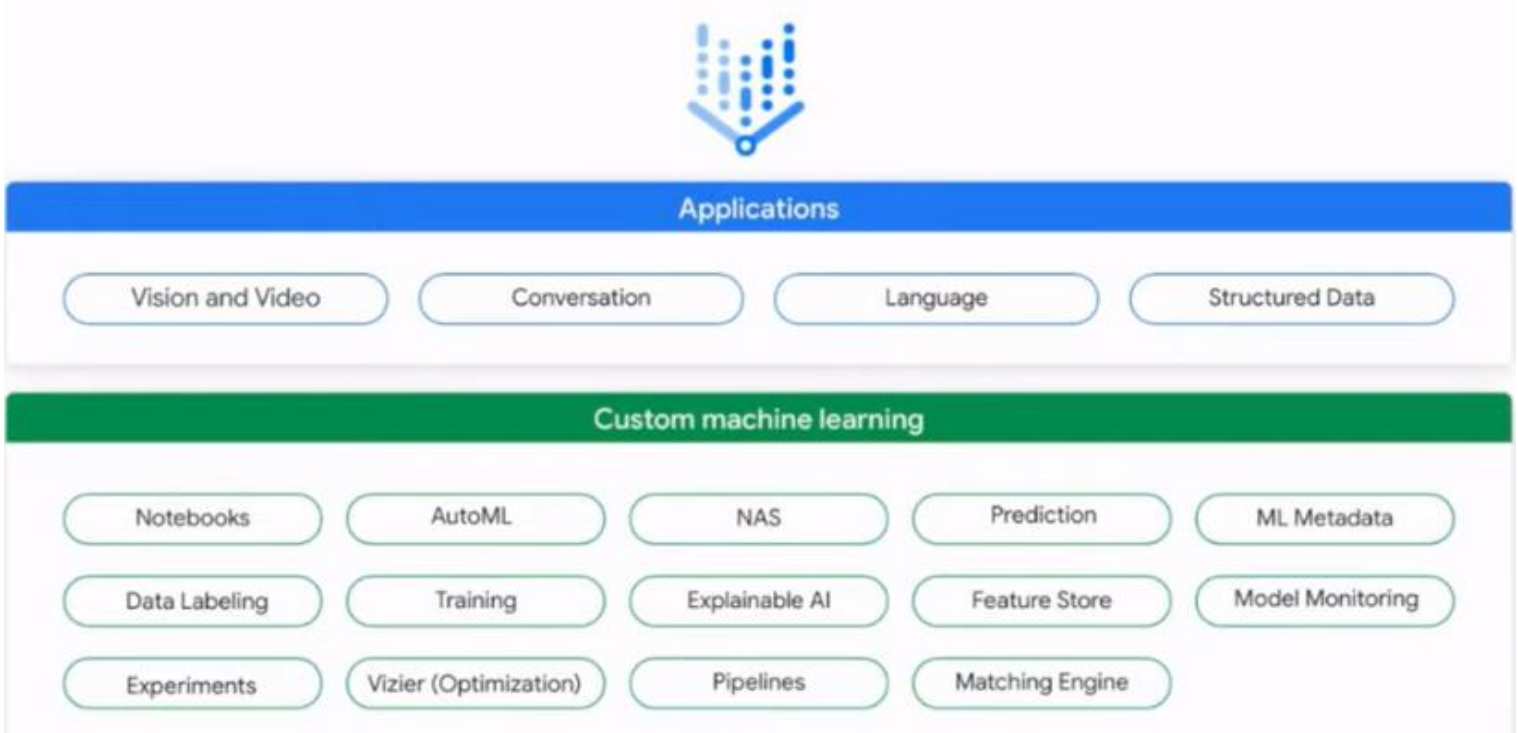
- When scaling AI, giant entities are no less successful than small, agile companies. There is no significant relationship proven between scaling success rates or return on investment.

- Large companies have grown to their current size for a reason. They know how to scale, and they're good at it. They apply those capabilities to their AI initiatives.

Googles AI Infrastructure



Vertex AI


Applications
Vision and Video
Conversation
Language
Structured Data
Custom machine learning
Notebooks
AutoML
NAS
Prediction
ML Metadata
Data Labeling
Training
Explainable AI
Feature Store
Model Monitoring
Experiments
Vizier (Optimization)
Pipelines
Matching Engine

Vertex AI Workbench

Fully managed compute

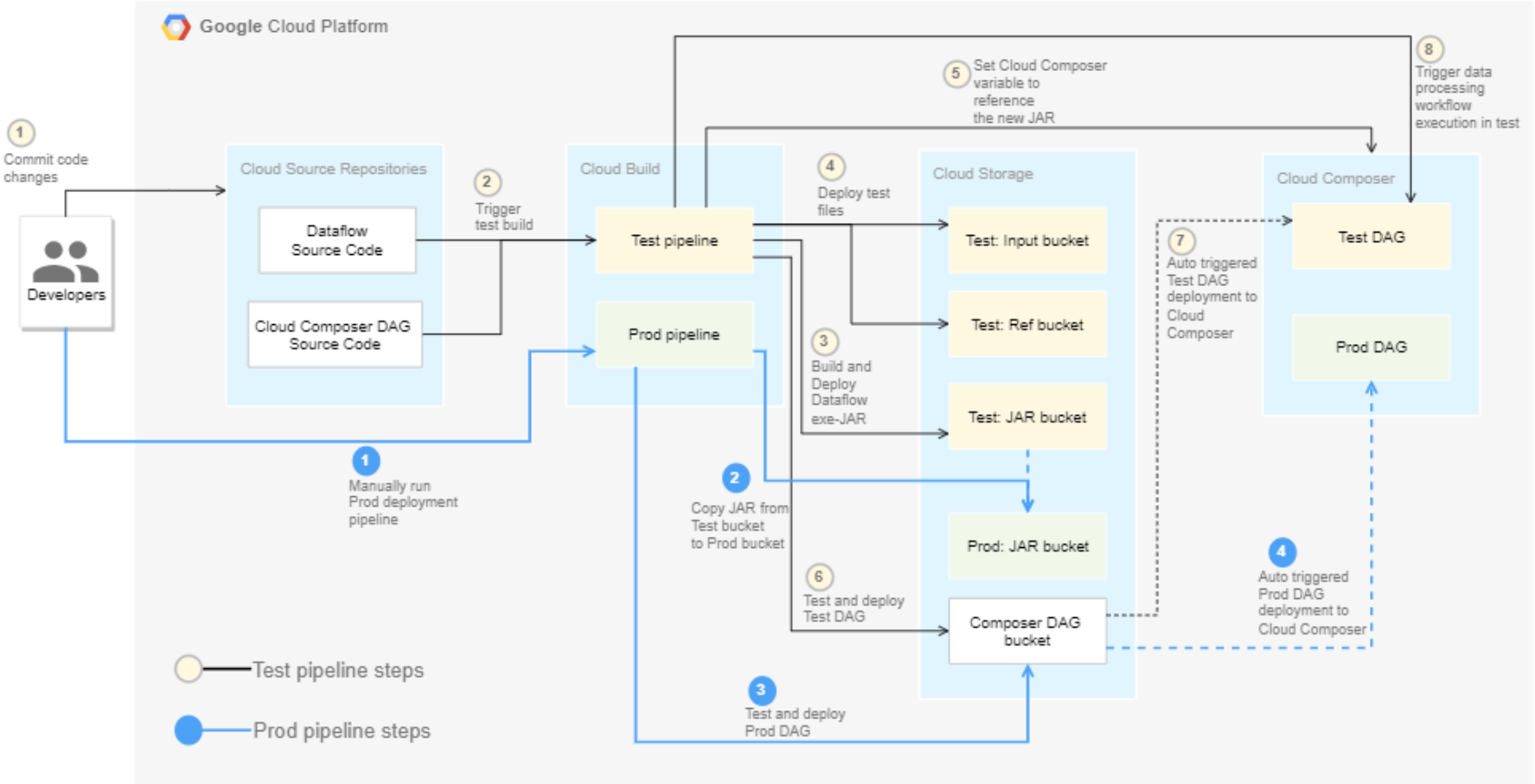
- A Jupyter-based fully managed, scalable, enterprise-ready compute infrastructure with security controls and user management capabilities.

Interactive data and ML experience

- Explore data and train ML models with easy connections to Google Cloud's big data solutions.

Portal to complete end-to-end ML training

- Develop and deploy AI solutions on Vertex AI with minimal transition.


Google Cloud Platform
1
Commit code changes
Developers
Cloud Source Repositories
Dataflow Source Code
Cloud Composer DAG Source Code
2
Trigger test build
Cloud Build
Test pipeline
Prod pipeline
4
Deploy test files
Cloud Storage
Test: Input bucket
Test: Ref bucket
Test: JAR bucket
Prod: JAR bucket
Composer DAG bucket
3
Build and Deploy Dataflow exe-JAR
5
Set Cloud Composer variable to reference the new JAR
8
Trigger data processing workflow execution in test
Cloud Composer
Test DAG
Prod DAG
7
Auto triggered Test DAG deployment to Cloud Composer
1
Manually run Prod deployment pipeline
2
Copy JAR from Test bucket to Prod bucket
6
Test and deploy Test DAG
4
Auto triggered Prod DAG deployment to Cloud Composer
3
Test and deploy Prod DAG
Test pipeline steps
Prod pipeline steps

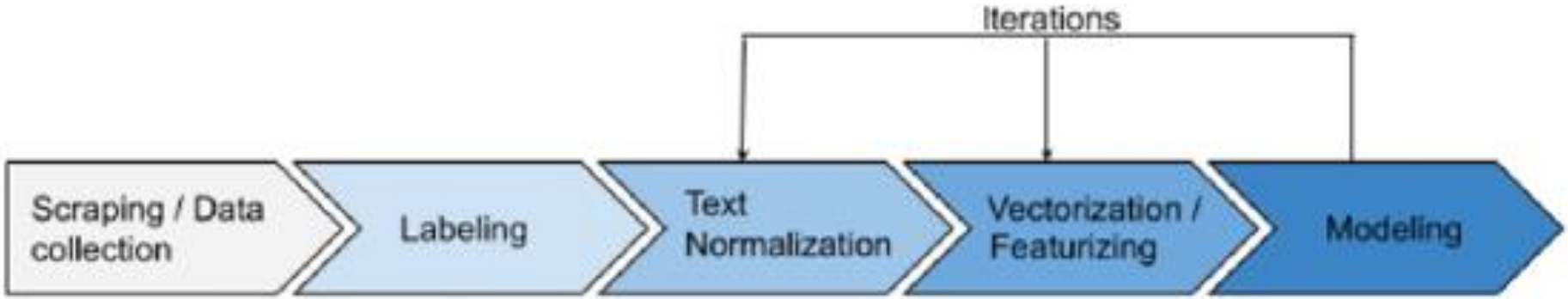
ML, NLP and Unstructured Data

18

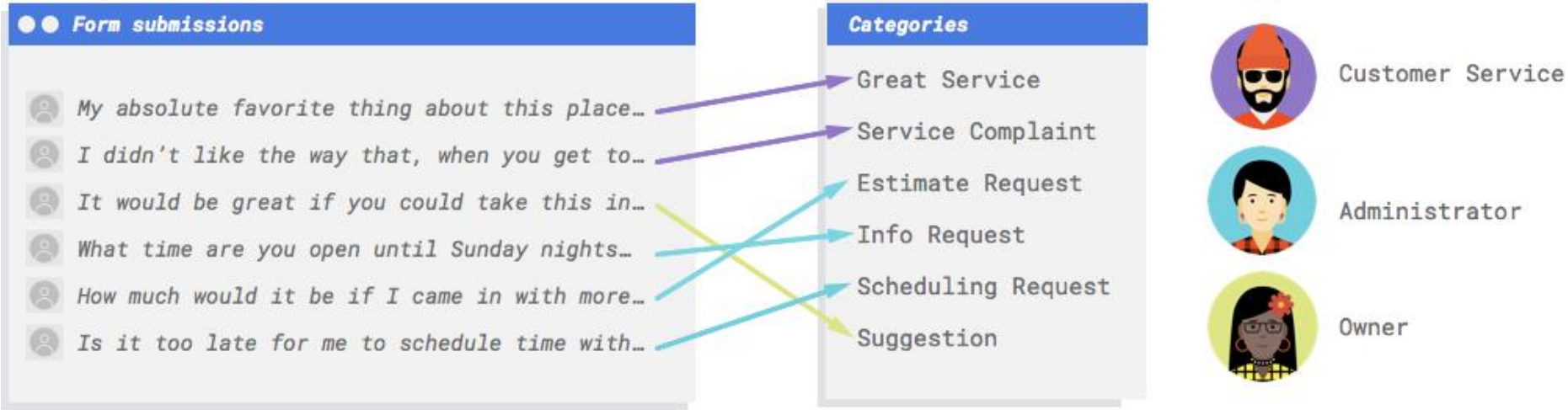
Why NLP and ML?

- Imagine your business has a contact form on its website.
- Every day you get many messages from the form, many of which are actionable in some way, but they all come in together and it's easy to fall behind on dealing with them since different employees handle different message types.
- It would be great if an automated system could categorize them so that the right person sees the right comments.

NLP and Text Processing



Why NLP and ML?



Framing NLPs Complexity

- Classical programming requires the programmer to specify step-by-step instructions for the computer to follow. But this approach quickly gets infeasible with NLP.
- Customer comments use a broad and varied vocabulary and structure — too diverse to be captured by a simple set of rules. If you tried to build manual filters, you'd quickly find that you weren't able to categorize the vast majority of your customer comments.

Cloud NL API vs AutoML NL

- The Natural Language API discovers syntax, entities, and sentiment in text, and classifies text into a predefined set of categories.
- If your text consists of news articles or other content you want to categorize, or if you want to discover the sentiment of your examples, the Natural Language API is worth trying.
- But if your text examples don't fit neatly into the sentiment-based or vertical-topic-based classification scheme available in the Natural Language API, and you want to use your own labels instead, it's worth experimenting with a custom classifier to see if it fits your needs.

Cloud NL API vs AutoML NL

Natural Language API

Review

I had a great experience. The grocer was really helpful. One thing I would recommend is putting the price of the market price items online so I can look them up as they change.

Food & Grocery Retailers	0.53
Hospitality Industry > Food Service	0.53

AutoML Content Classification

(with your own data and labels)

Review

I had a great experience. The grocer was really helpful. One thing I would recommend is putting the price of the market price items online so I can look them up as they change.

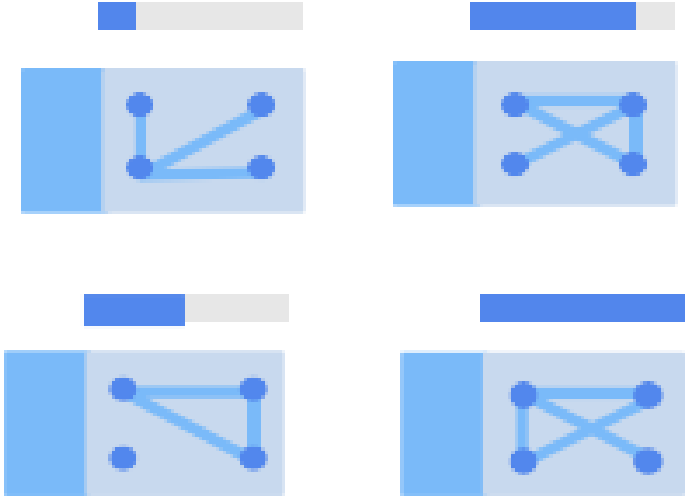
Great Service	0.88
Suggestion	0.84
Info Request	0.79

AutoML NL

- Machine learning involves using data to train algorithms to achieve a desired outcome. The specifics of the algorithm and training methods change based on the use case.
- There are many different subcategories of machine learning, all of which solve different problems and work within different constraints.
- AutoML Natural Language enables you to perform supervised learning, which involves training a computer to recognize patterns from labeled data.
- Using supervised learning, you can train a custom model to recognize content that you care about in text.

Data Preparation

- To train a custom model with AutoML Natural Language, you supply labeled examples of the kinds of text items (inputs) that you want to classify and the categories or labels (the answer) that you want the ML systems to predict.



Assess Your Use Case

While putting together the dataset, always start with the use case. You can begin with the following questions:

- What outcome are you trying to achieve?
- What kinds of categories do you need to recognize to achieve this outcome?
- Is it possible for humans to recognize those categories? Although AutoML Natural Language can handle more categories than humans can remember and assign at any one time, if a human can't recognize a specific category, then AutoML Natural Language will have a hard time as well.
- What kinds of examples would best reflect the type and range of data your system will classify?

Human-centered ML

A core principle underpinning Google's ML products is human-centered machine learning, an approach that foregrounds responsible AI practices, including fairness.

The goal of fairness in ML is to understand and prevent unjust or prejudicial treatment of people related to race, income, sexual orientation, religion, gender, and other characteristics historically associated with discrimination and marginalization,

ML Principle: ML Won’t Figure Out What Problems to Solve

Machine learning and artificial intelligence have a lot of hype around them right now.

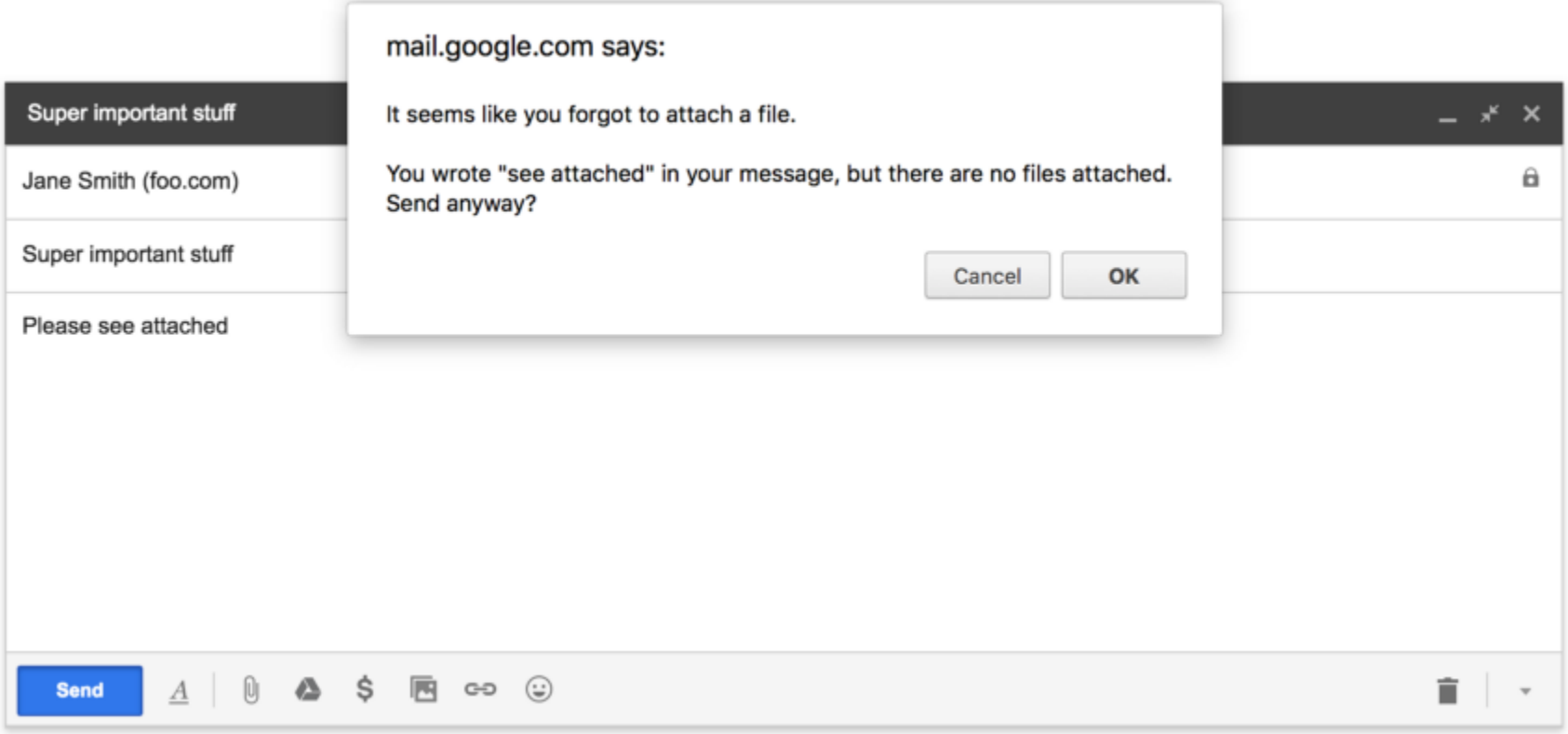
Many companies and product teams are jumping right into product strategies that start with ML as a solution and skip over focusing on a meaningful problem to solve.

ML Principle: Does This Problem Require ML?

Once you've identified the need or needs you want to address, you'll want to assess whether ML can solve these needs in unique ways.

There are plenty of legitimate problems that don't require ML solutions.

ML Principle: To ML or not to ML



ML Principle: Does This Problem Require ML?

Describe the way a theoretical human “expert” might perform the task today.

If your human expert were to perform this task, how would you respond to them so they improved for the next time? Do this for all four phases of the confusion matrix.

If a human were to perform this task, what assumptions would the user want them to make?

ML Principle: Weigh Costs of False Negatives and False Positives

Your ML system will make mistakes. It’s important to understand what these errors look like and how they might affect the user’s experience of the product.

In one of the questions in point 2 we mentioned something called the confusion matrix. This is a key concept in ML and describes what it looks like when an ML system gets it right and gets it wrong.

ML Pipelines with Kubeflow

19

What are Kubeflow Pipelines?

- Kubeflow Pipelines are a new component of Kubeflow that can help you compose, deploy, and manage end-to-end (optionally hybrid) machine learning workflows.
- Because they are a useful component of Kubeflow, they give you a no lock-in way to advance from prototyping to production.
- Kubeflow Pipelines also support rapid and reliable experimentation, so users can try many ML techniques to identify what works best for their application.

Running Kubeflow Pipelines

- We’re running the examples on Google Kubernetes Engine (GKE).
- GKE allows easy integration of GCP services that are relevant for ML workflows, including Cloud Dataflow, BigQuery, and Cloud ML Engine.
- If you need to keep your ML workflows on-premises, you may also be interested in GKE On-Prem[ALPHA].

Kubeflow Pipelines & TensorFlow

- Many Kubeflow pipelines make use of theTensorFlow Transform (TFT) for data preprocessing and to avoid training (or serving) skew
- Kubeflow's TFJob CRD (Custom Resource Definitions library) for supporting distributed training
- TensorFlow Model Analysis (TFMA) for analysis of learned models in conjunction with Kubeflow's JupyterHub notebooks installation.

Kubeflow Components

Kubeflow's components include:

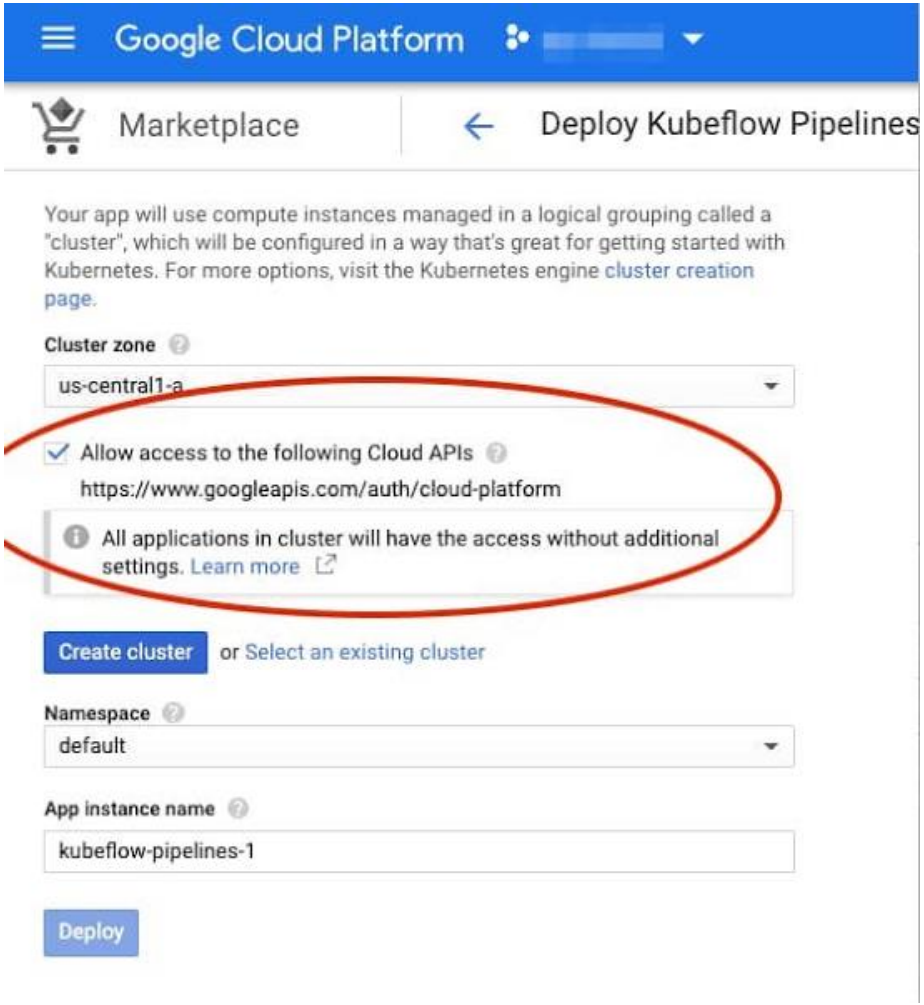
- Support for distributed TensorFlow training via the TFJob CRD
- The ability to serve trained models using TensorFlow Serving
- A JupyterHub installation with many commonly-required libraries and widgets included in the notebook installation, included those needed for TensorFlow Model Analysis (TFMA) and TensorFlow Transform (TFT)
- Kubeflow Pipelines

What are Cloud AI Platform Pipelines?

Cloud AI Platform Pipelines, now in Beta, provides a way to deploy robust, repeatable machine learning pipelines along with monitoring, auditing, version tracking, and reproducibility.

It also delivers an enterprise-ready, easy to install, secure execution environment for your ML workflows.

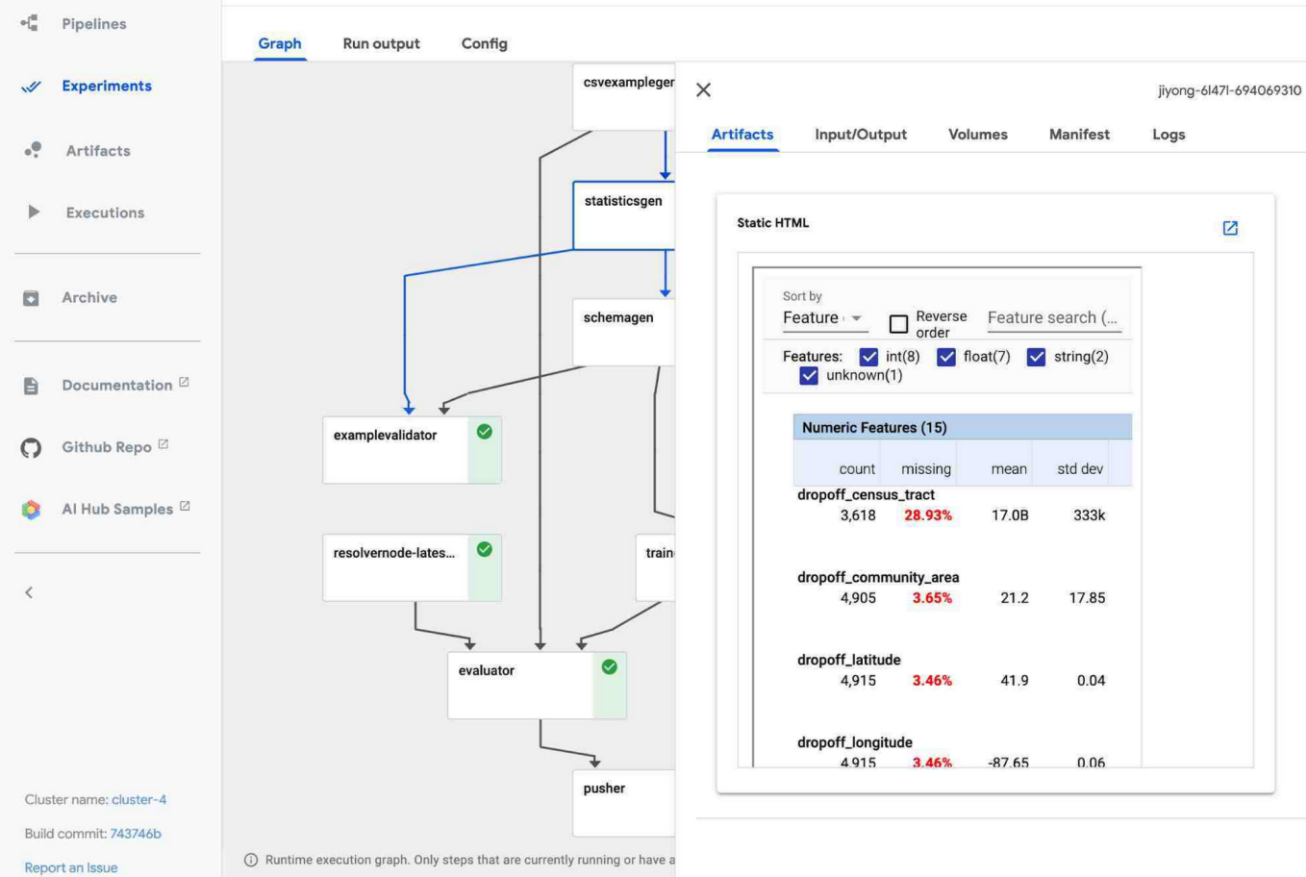
Cloud AI Platform Pipelines Quick Start



Cloud AI Platform Pipelines Lineage

- AI Platform Pipelines is based on Kubeflow Pipelines (KFP) installed on a Google Kubernetes Engine (GKE) cluster, and can run pipelines specified via both the KFP and TFX SDKs.

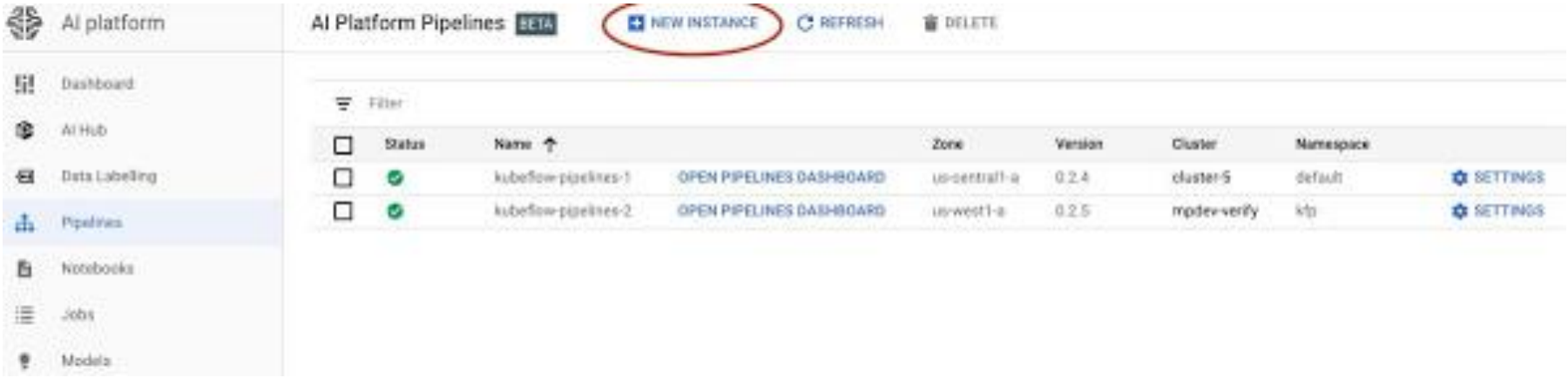
Cloud AI Platform Pipelines Scaffold



Why Cloud AI Platform Pipelines?

- A machine learning workflow can involve many steps with dependencies on each other, from data preparation and analysis, to training, to evaluation, to deployment, and more.
- It’s hard to compose and track these processes in an ad-hoc manner—for example, in a set of notebooks or scripts—and things like auditing and reproducibility become increasingly problematic.

Cloud AI Platform Pipelines



Cloud AI Platform Pipelines Features

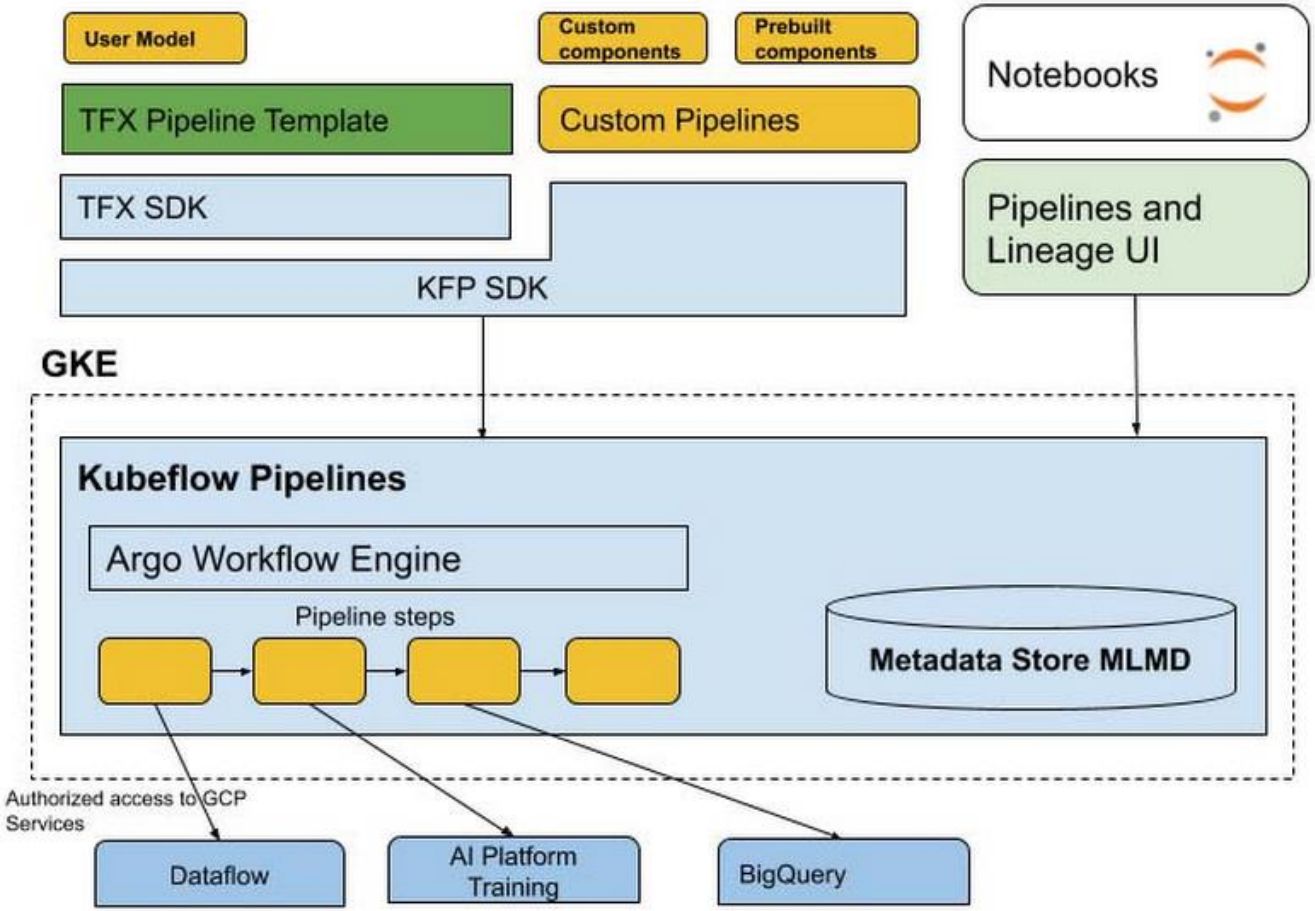
- Push-button installation via the Google Cloud Console
- Enterprise features for running ML workloads, including pipeline versioning, automatic metadata tracking of artifacts and executions, Cloud Logging, visualization tools, and more
- Seamless integration with Google Cloud managed services like BigQuery, Dataflow, AI Platform Training and Serving, Cloud Functions, and many others
- Many prebuilt pipeline components (pipeline steps) for ML workflows, with easy construction of your own custom components

Cloud AI Platform Pipelines IAM

- AI Platform Pipelines gives you secure and authenticated access to the Pipelines UI via the Cloud AI Platform UI, with no need to set up port-forwarding. You can also give access to other members of your team.

- It is straightforward to programmatically access a Pipelines cluster via its REST API service. This makes it easy to use the Pipelines SDK from Cloud AI Platform notebooks, for example, to perform tasks like defining pipelines or scheduling pipeline run jobs.

Cloud AI Platform Pipelines Arch



Cloud AI Platform Pipelines SDKs

Cloud AI Platform pipelines supports two SDKs to author ML pipelines: the Kubeflow Pipelines SDK—part of the Kubeflow OSS project—and the TFX SDK.

Over time, these two SDK experiences will merge. The TFX SDK will support framework-agnostic operations available in the KFP SDK. Google will provide transition paths that make it easy for existing KFP SDK users to upgrade to the merged SDK.

ML Metadata

AI Platform Pipeline runs include automatic metadata tracking, using ML Metadata (MLMD), which is a library for recording and retrieving metadata associated with ML developer and data scientist workflows. It’s part of TensorFlow Extended (TFX), but it’s designed to also be used independently.

The automatic metadata tracking logs the artifacts used in each pipeline step, pipeline parameters, and the linkage across the input/output artifacts, as well as the pipeline steps that created and consumed them.

Pipeline Versioning

Pipelines

+ Upload pipeline | Refresh | Delete

Filter pipelines

Pipeline name	Description	Uploaded on
[Tutorial] DSL - Control structures	source code Shows how to use conditional execution and exit handlers. This pipeline will randomly fail to demonstra...	2/20/2020, 3:28:12 PM
[Tutorial] Data passing in python com...	source code Shows how to pass data between python components.	2/20/2020, 3:28:11 PM
[Demo] TFX - Taxi Tip Prediction Mod...	source code GCP Permission requirements. Example pipeline that does classification with model analysis based on ...	2/20/2020, 3:28:10 PM

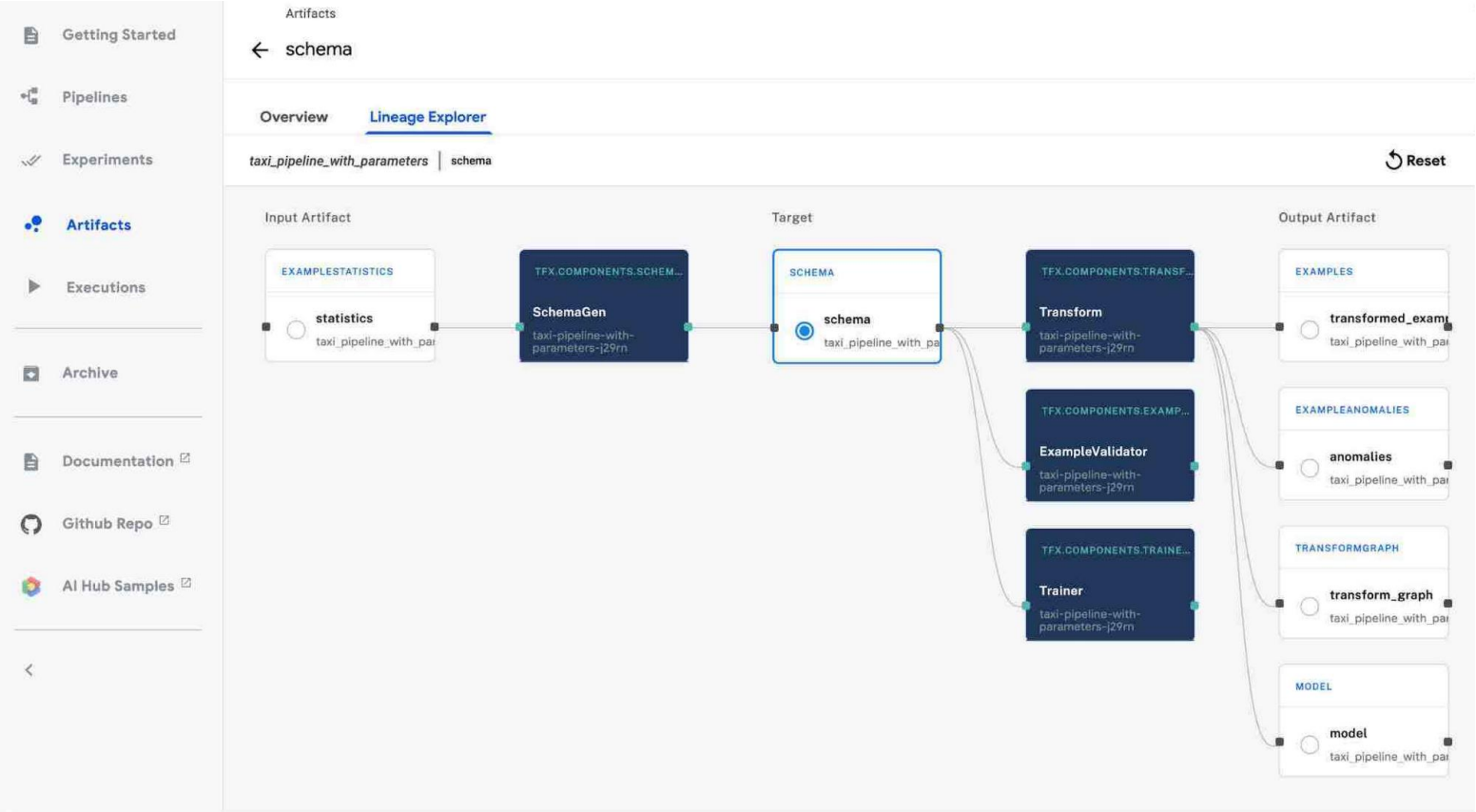
Version name	Uploaded on
TFX - Taxi Tip Prediction Model Trainer_version_at_2020-03-03T15:44:30.197Z	3/3/2020, 7:55:03 AM
[Demo] TFX - Taxi Tip Prediction Model Trainer	2/20/2020, 3:28:10 PM

Rows per page: 10

[Demo] XGBoost - Training with Confu...	source code GCP Permission requirements. A trainer that does end-to-end distributed training for XGBoost models.	2/20/2020, 3:28:09 PM

Rows per page: 10

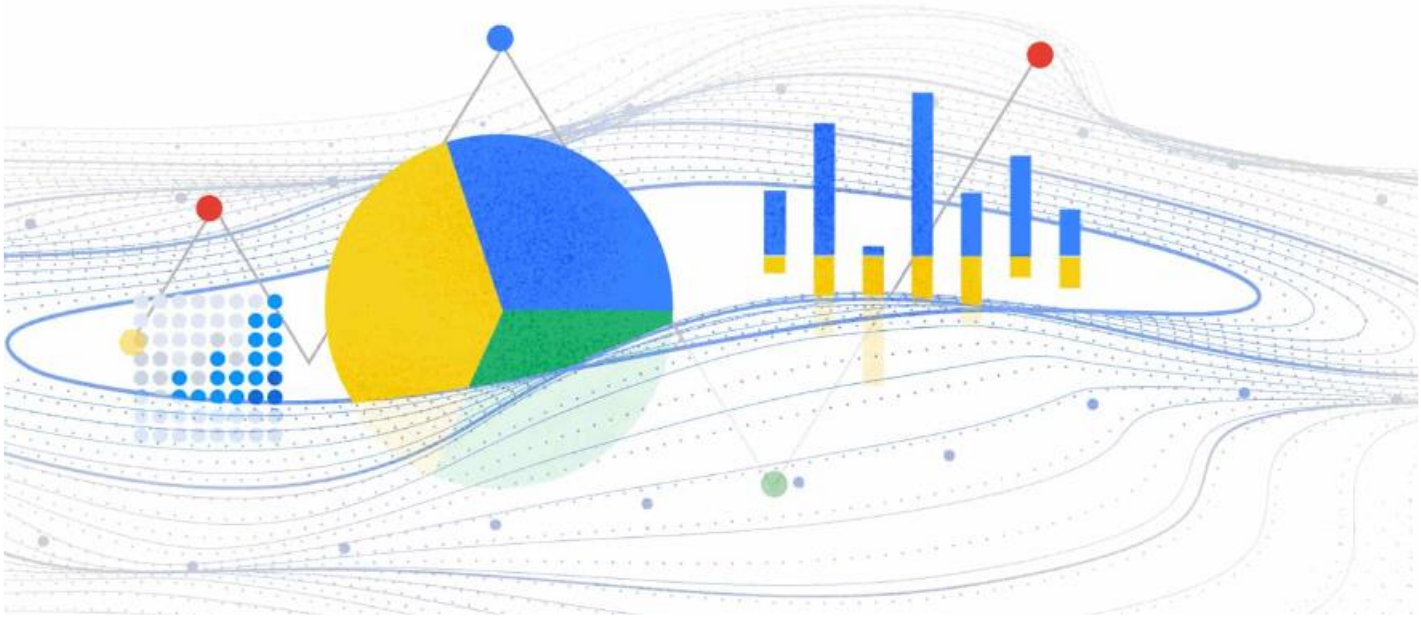
Pipeline Lineage



Serverless, Autoscaling Data Pipelines

20

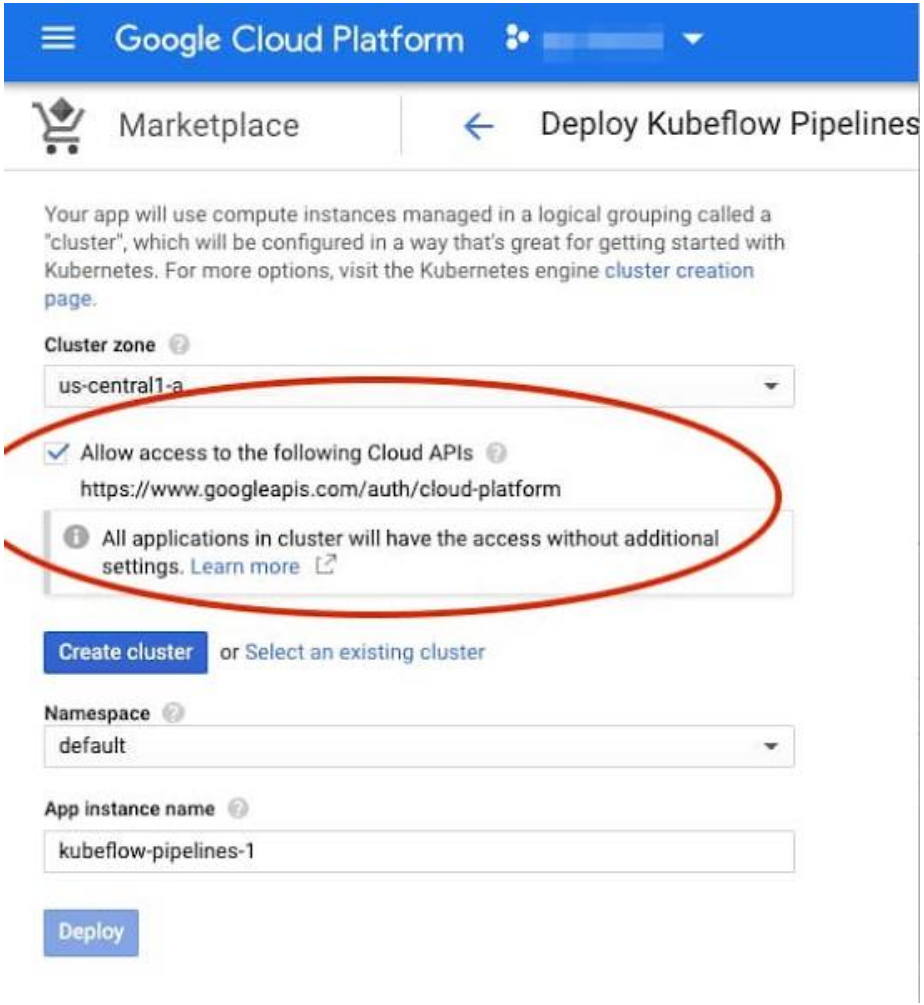
Varied Pipeline Mechanisms in GCP



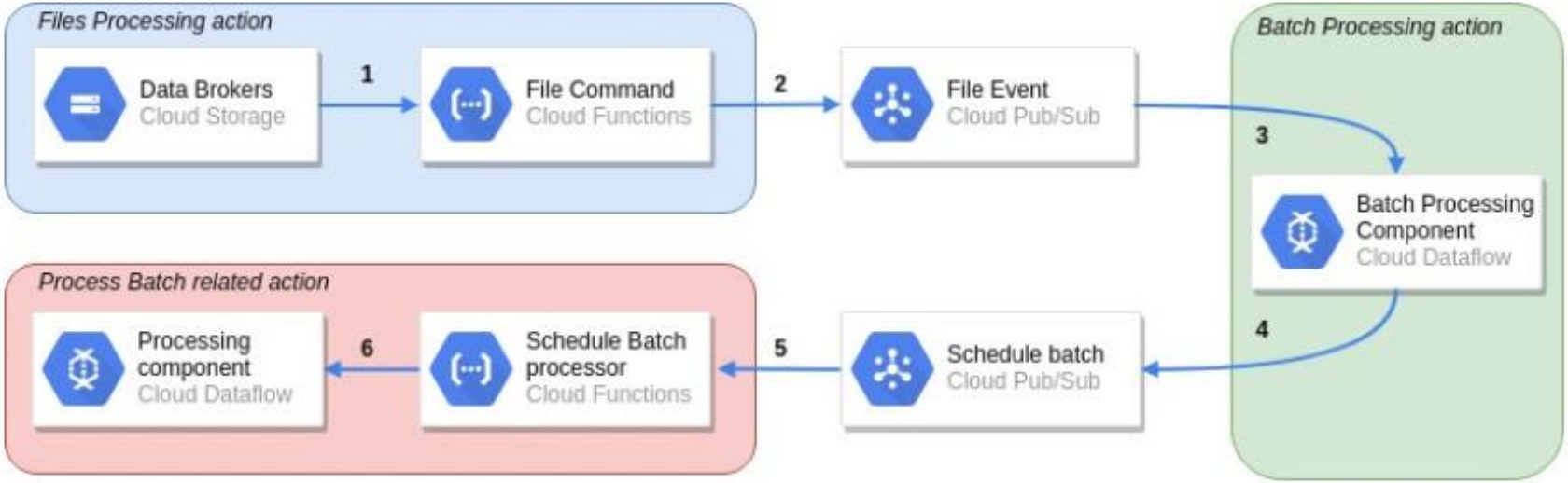
Spark on Google Cloud

- Eliminate time spent managing Spark clusters: With serverless Spark, users submit their Spark jobs, and let them do auto-provision, and autoscale to finish.
- Enable data users of all levels: Connect, analyze, and execute Spark jobs from the interface of users' choice including BigQuery, Vertex AI or Dataplex, in 2 clicks, without any custom integrations.
- Retain flexibility of consumption: No one size fits all. Use Spark as serverless, deploy on Google Kubernetes Engine (GKE), or on compute clusters based on the requirements.

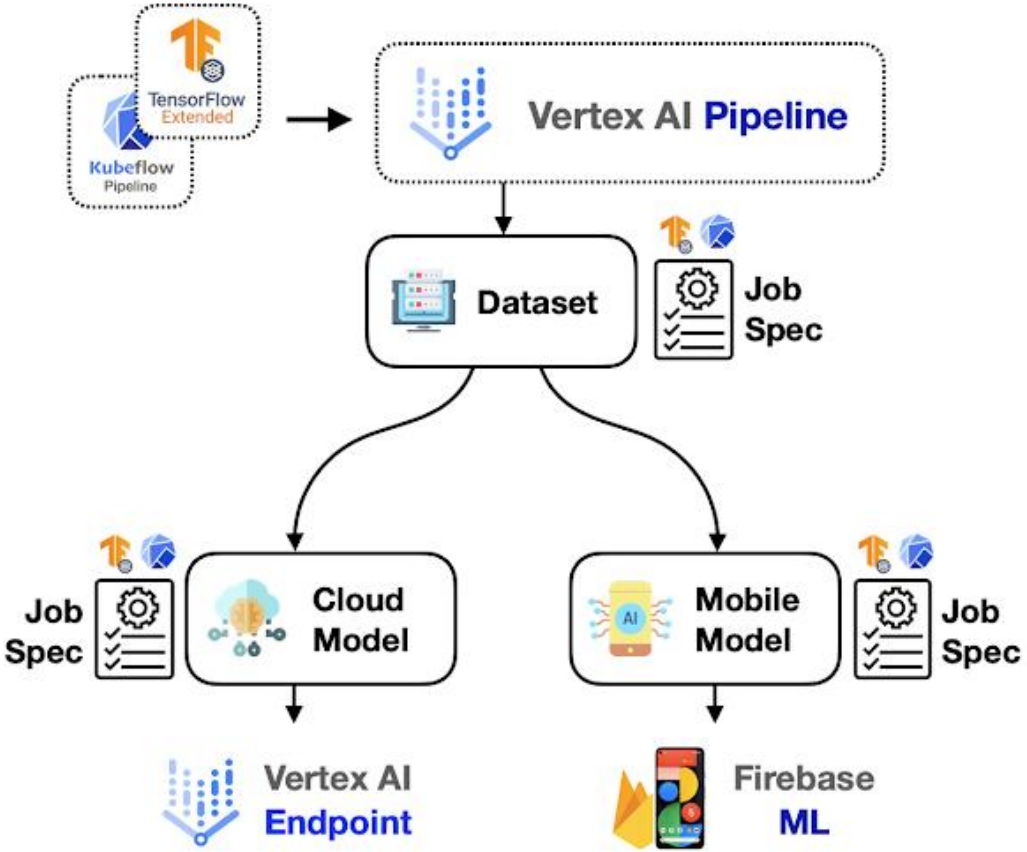
Cloud AI Platform Pipelines Quick Start



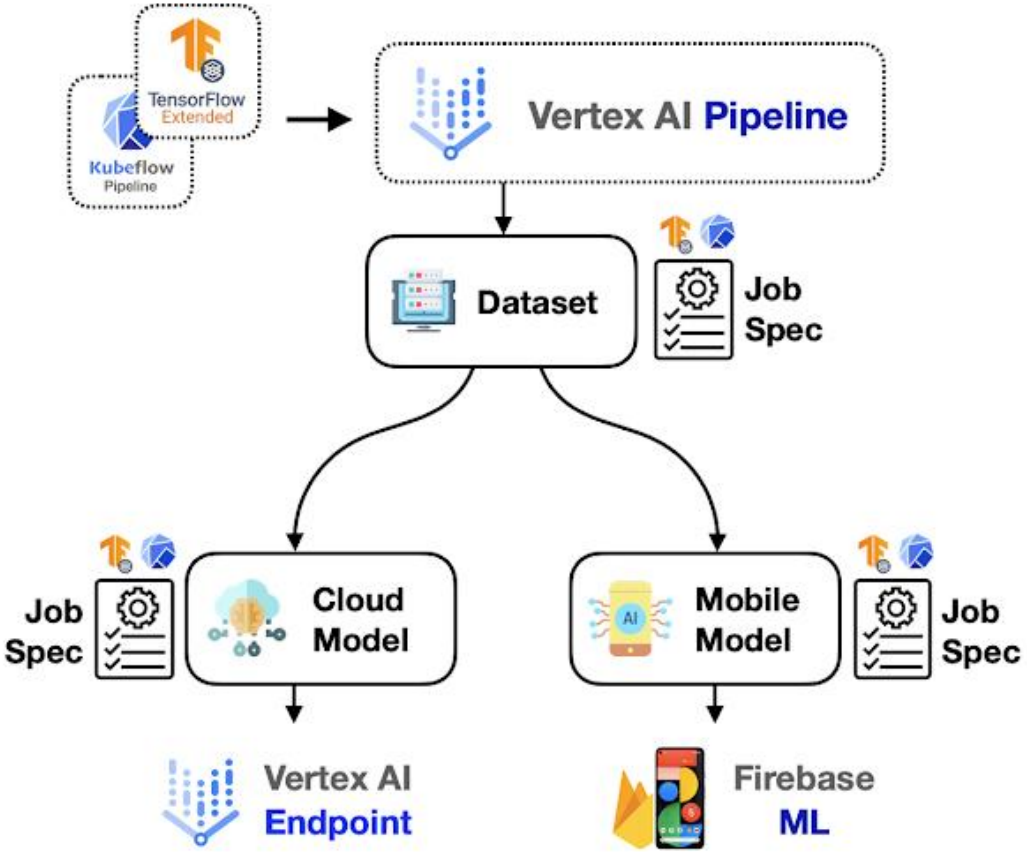
Functions Data Pipeline Triggering



Vertex AI Pipeline in Action



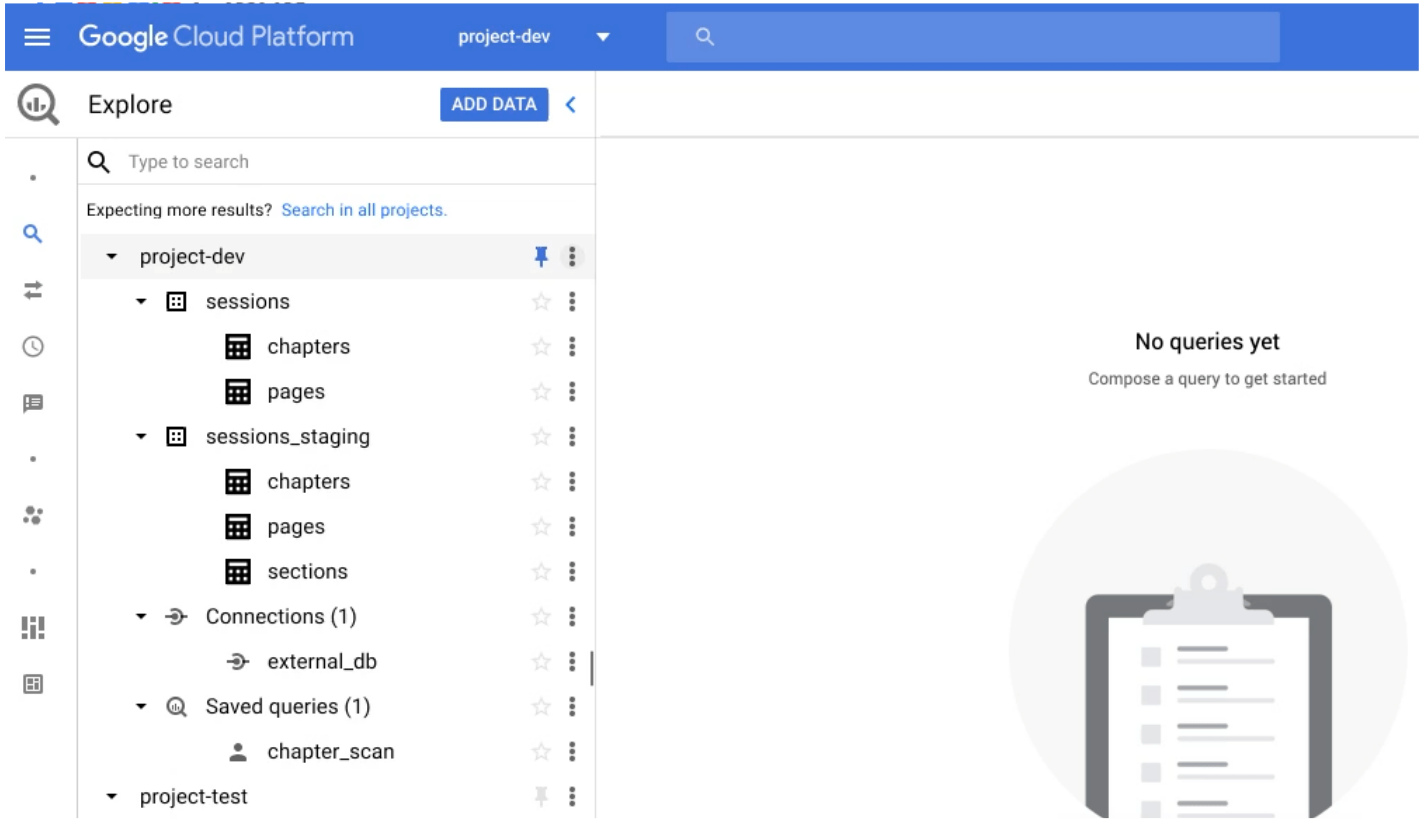
Vertex AI Pipeline in Action



Spark on Google Cloud

- Eliminate time spent managing Spark clusters: With serverless Spark, users submit their Spark jobs, and let them do auto-provision, and autoscale to finish.
- Enable data users of all levels: Connect, analyze, and execute Spark jobs from the interface of users' choice including BigQuery, Vertex AI or Dataplex, in 2 clicks, without any custom integrations.
- Retain flexibility of consumption: No one size fits all. Use Spark as serverless, deploy on Google Kubernetes Engine (GKE), or on compute clusters based on the requirements.

Spark on BigQuery



Spark on Vertex

Google Cloud Platform | abhi | Search products and resources

AI Platform

- Dashboard
- Data Labeling
- Notebooks
- Pipelines
- Jobs
- Models

Managed notebooks

NEW NOTEBOOK | REFRESH | START | STOP | RESET | DELETE

MANAGED NOTEBOOKS PREVIEW | INSTANCES | EXECUTIONS PREVIEW | SCHEDULES PREVIEW | SCHEDULED RUNS

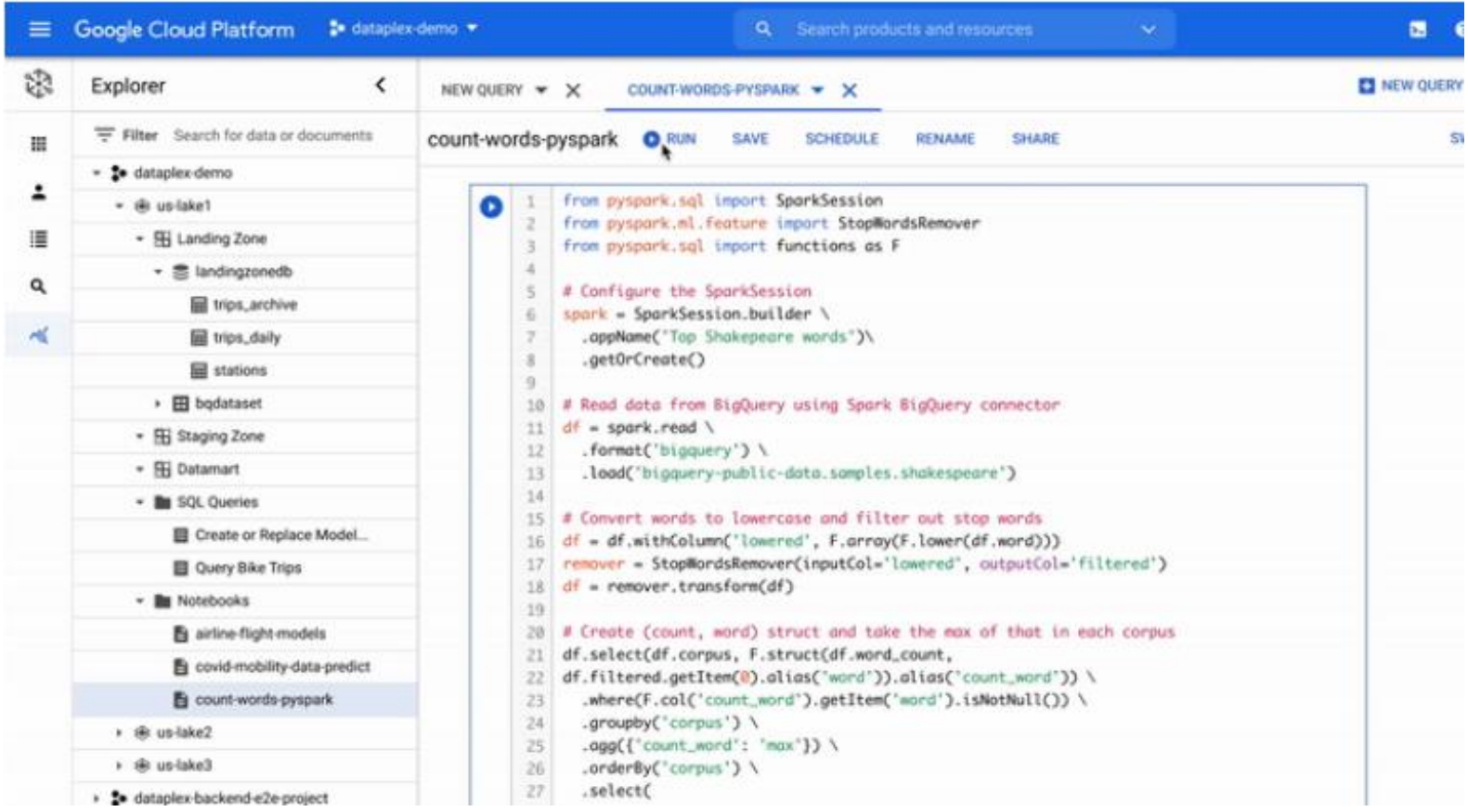
Managed notebooks provide JupyterLab services and flexible computing resources integrated with Google Cloud services. Learn more

Region: us-central1 (Iowa)

Filter: Enter property name or value

		Notebook name		Location
☐	●	managed-notebook-1628014966	OPEN JUPYTERLAB	us-central1-f
☐	●	managed-notebook-1629781423	OPEN JUPYTERLAB	us-central1-c

Spark through DataPlex



Timing is everything

Dataproc Serverless for Spark will be Generally Available within a few weeks.

BigQuery and Dataplex integration is in Private Preview.

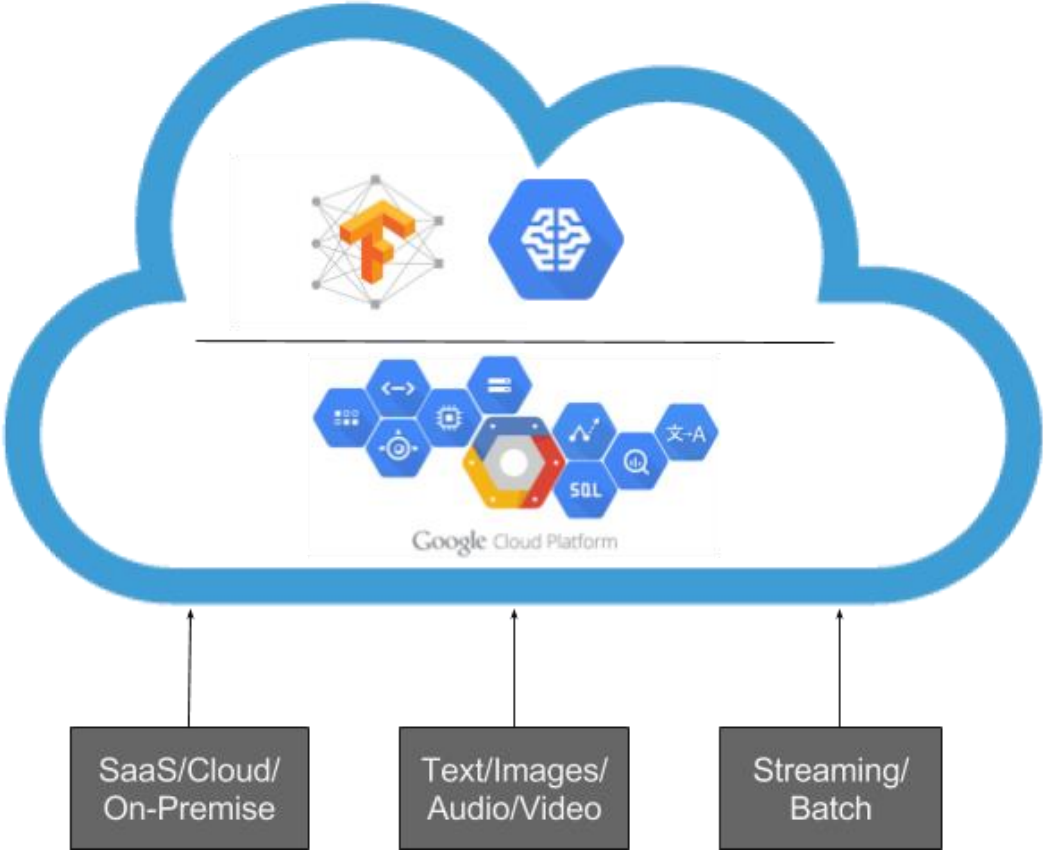
Vertex AI workbench is available in Public Preview, you can get started here.

For all capabilities, you can request for Preview access through the GCP preview form.

TensorFlow on GCP

21

Tensorflow ML

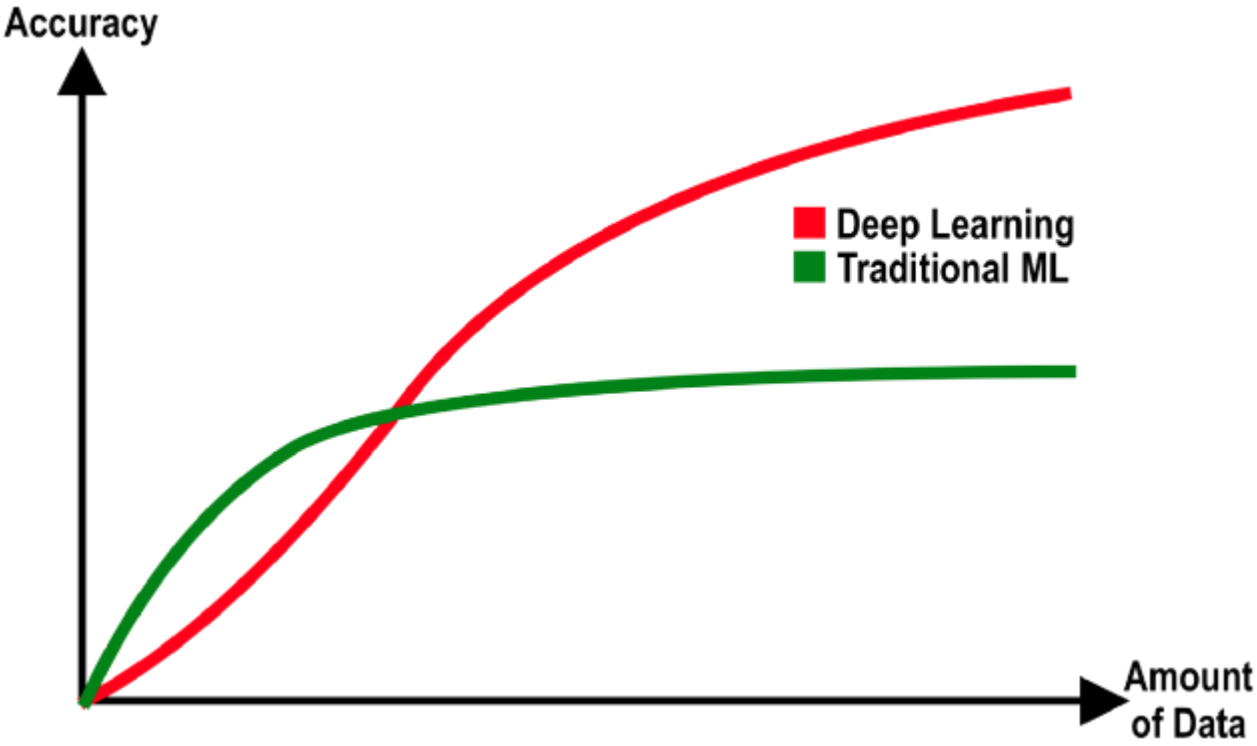


Tensorflow

- TensorFlow is an end-to-end open source platform for machine learning.
- TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular
- TensorFlow API to develop and train machine learning models.

TensorFlow

Tensorflow Deep Learning



Deep Learning Options

PyTorch

TensorFlow

Distributed High Performance

- Data in TensorFlow is represented as tensors which are multi-dimensional arrays. In some sense they are similar to NumPy arrays in Python but with a few important differences especially around distributed computing.
- TensorFlow programs are structured into a construction phase that creates a computational graph followed by an execution phase.
- TensorFlow supports distributed computing allowing parts of the graph to be executed on different processes. In addition it also supports execution on GPU that allows for much faster computation compared to CPU based processing.

End to End Deep Learning Platform

- TensorFlow models can be deployed and made available as a prediction service. This makes the model available to clients with either real time or batch needs.
- For instance, a mobile application needing to get a prediction can make a call out via REST to the model to get a real time prediction.
- The process of training the model on the cloud, deploying and then making it available as a prediction service is simple – this is possible because of the fully integrated stack that Cloud/AutoML offers.
- Cloud/AutoML provides API based services such as Speech, Vision and Translation which can be easily integrated without need of complex custom development.

Easy to Get Started

- Though TensorFlow provides complex deep-learning models and non-trivial distributed computing features, it is fairly easy to get started.
- Google has provided copious documentation, many examples and code labs that make it possible to get started and understand how various components work together.
- There’s also a tool called TensorBoard that makes it easier to visualize, debug and tune the models.
- Cloud/AutoML is integrated to Google Big Data product suite e.g BigQuery and Google Storage which makes it easier for data access and running machine learning models quicker.

Open Source

- TensorFlow was originally developed by Google Brain team but later made open source and is getting widespread acceptance.
- Github repository lists over 600 contributors and TensorFlow Dev summit highlights many use cases in such diverse areas as agriculture, retail, genomics and healthcare.
- We believe open-source support and collaboration with various companies helps to build a comprehensive open-source machine learning library.

Tensorflow Programming

- The two programming languages with stable and official TensorFlow APIs are Python and C.
- C++, Java, JavaScript, Go, and Swift are other programming languages where developers may find limited-to-extensive TensorFlow compatibility.
- Most developers end up using Python since Python has compelling data libraries such as NumPy, pandas, and Matplotlib.

Tensorflow Features

- Reduced time to build and train models;
- Useful data processing tools;
- Compatibility with other popular data libraries such as NumPy, matplotlib, and pandas;
- A rich catalog of pre-trained models with TF Hub;
- Tools to deploy trained models across different devices such as iOS, Android, Windows, macOS, and Web;
- Great community support;
- A desirable skill by tech companies.

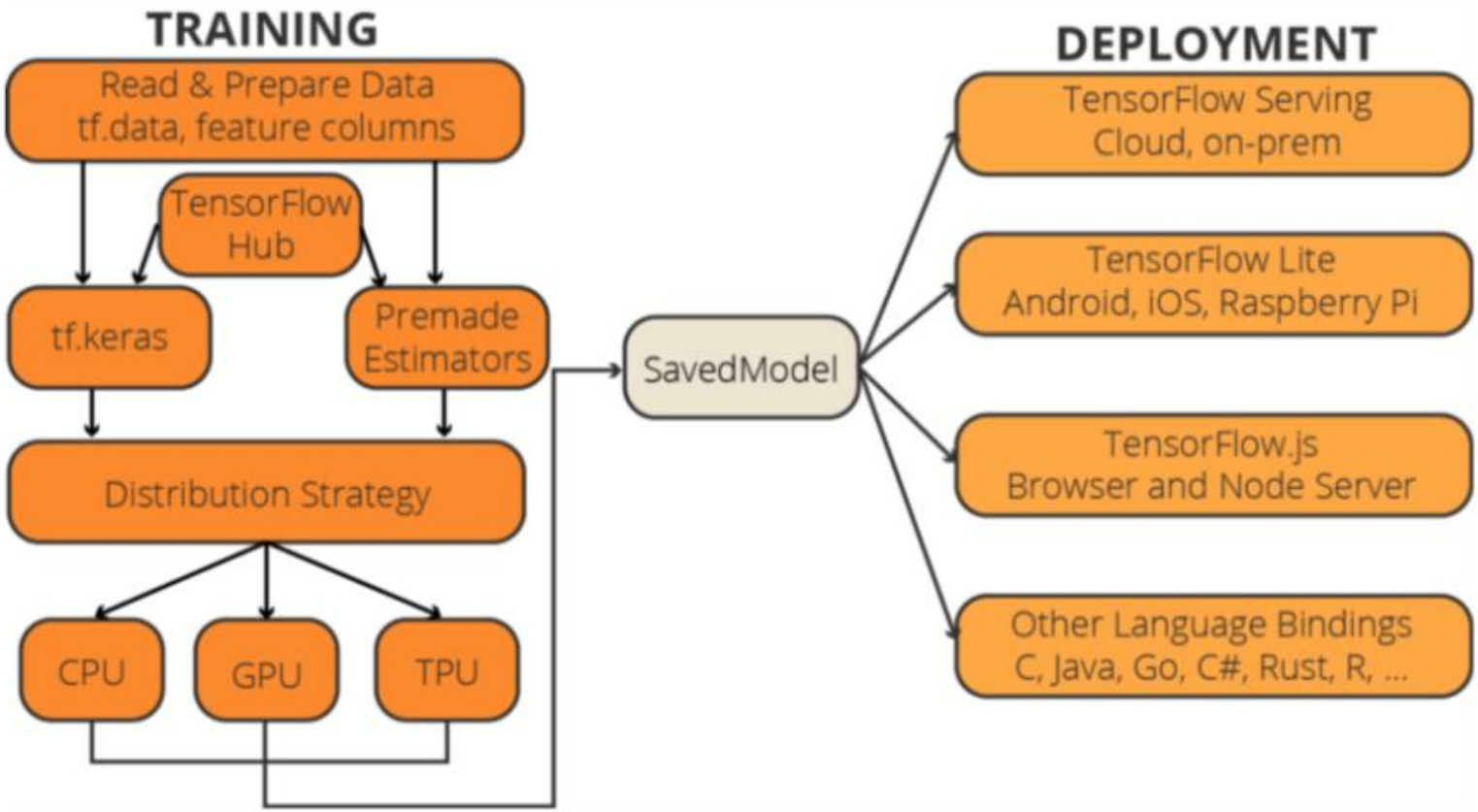
Tensorflow History

- Currently, we are using the second major version of TensorFlow: TensorFlow 2.x.
- It took almost nine years to achieve this level of maturity.
- Tensorflow is still in the beginning phase of becoming the ultimate deep learning platform.
- Current trends indicate that deep learning processes will be much more streamlined in the future.

Tensorflow 2.0

- In September 2019, the TensorFlow team released TensorFlow 2.0, the current major version, which streamlined many of the inconveniencies of building neural networks.
- With version 2.0, TensorFlow finally embraced Keras as the official main High-level API to build, train, evaluate neural networks.
- TensorFlow 2.0 streamlined the data loading and processing tools and provided newly added features.
- Eager Execution was made the default option, replacing Graph execution. This strategy was adopted because PyTorch has attracted many researchers with eager execution.

Tensorflow 2.0

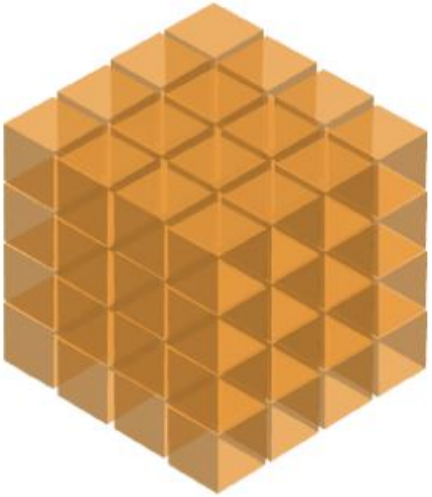


Tensors

- **Definition of Tensors** → What is a Tensor?
- **Creation of Tensors** → Functions to Create Tensor Objects
- **Qualifications of Tensors** → Characteristics and Features of Tensor Objects
- **Operations with Tensors** → Indexing, Basic Tensor Operations, Shape Manipulation, and Broadcasting
- **Special Types of Tensors** → Special Tensor Types Other than Regular Tensors

What is aTensor?

- Tensors are TensorFlow's multi-dimensional arrays with uniform type. They are very similar to NumPy arrays, and they are immutable, which means that they cannot be altered once created. You can only create a new copy with the edits.



Tensor Creation

- # You can create Tensor objects with the `tf.constant` function:
- x = tf.constant([[1, 2, 3, 4 ,5]])

- # You can create Tensor objects only consisting of 1s with the `tf.ones` function:
- y = tf.ones((1,5))

- # You can create Tensor objects only consisting of 0s with the `tf.zeros` function:
- z = tf.zeros((1,5))

- # You can use the `tf.range()` function to create Tensor objects:
- q = tf.range(start=1, limit=6, delta=1)

Tensor Characteristics

TensorFlow Tensors are created as tf.Tensor objects, and they have several characteristic features:

1. Tensors have a rank based on the number of dimensions they have. Secondly, they have a shape, a list that consists of the lengths of all their dimensions.
2. All tensors have a size, which is the total number of elements within a Tensor.
3. Finally, their elements are all recorded in a uniform Dtype (data type).

Tensor Indexes

An index is a numerical representation of an item's position in a sequence. This sequence can refer to many things: a list, a string of characters, or any arbitrary sequence of values.

TensorFlow also follows standard Python indexing rules, which is similar to list indexing or NumPy array indexing.

A few rules about indexing:

- Indices start at zero (0).
- Negative index ("-n") value means backward counting from the end.
- Colons (":") are used for slicing: start:stop:step.
- Commas (",") are used to reach deeper levels.

Tensor Math Operations

You can easily do basic math operations on tensors such as:

- Addition
- Element-wise Multiplication
- Matrix Multiplication
- Finding the Maximum or Minimum
- Finding the Index of the Max Element
- Computing Softmax Value

Special Types of Tensors

We tend to generate Tensors in a rectangular shape and store numerical values as elements. However, TensorFlow also supports irregular, or specialized, Tensor types, which are:

- Ragged - tensors with different numbers of elements along the size axis.
- String – tensors, which stores string objects.
- Sparse – rectangular Tensors for sparse data.

Tensorflow Variables

Definition of Variables →A Brief Introduction, Comparison with Tensors

Creation of Variables → Instantiating tf.Variable Objects

Qualifications of Variables → Characteristics and Features

Operations with Variables → Basic Tensor Operations, Indexing, Shape Manipulation, and Broadcasting

Hardware Selection for Variables → GPUs, CPUs, TPUs

Tensorflow Variables

- A TensorFlow Variable is the preferred object type representing a shared and persistent state that you can manipulate with any operation, including TensorFlow models.
- Manipulation refers to any value or parameter update. This characteristic is the most distinguishing feature of Variables compared to tf.Tensor objects.
- TensorFlow Variables are recorded as tf.Variable objects.

Creation of Variables

- We can instantiate (i.e., create) tf.Variableobjects with the tf.Variable() function.
- The tf.Variable() function accepts different data types as parameter such as integers, floats, strings, lists, and tf.Constant objects.

```
e = ["Hello, World!",
     "This is an",
     "example of",
     "TensorFlow Variable"]
var_e = tf.Variable(e)
print(var_e)
```

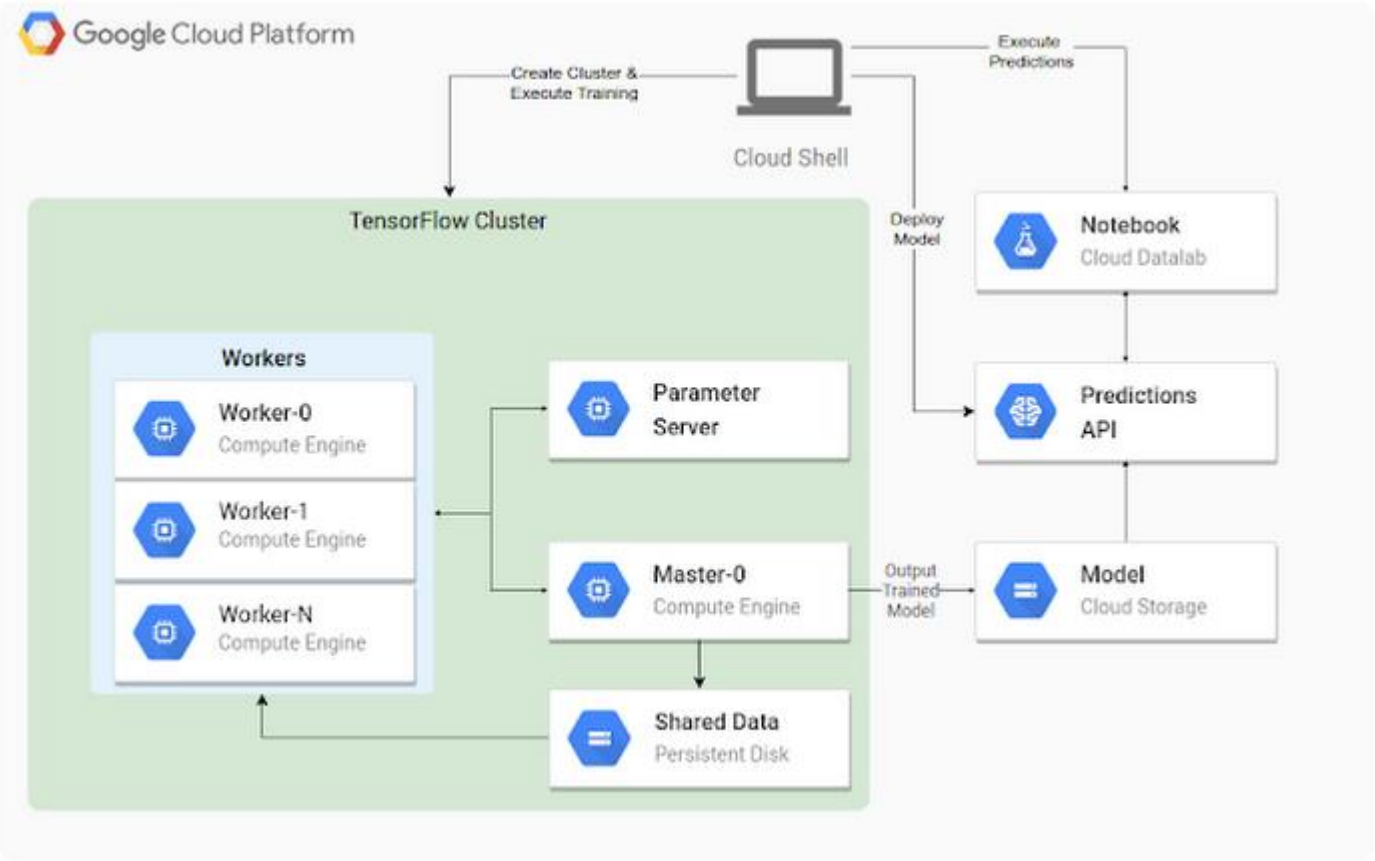
```
a = tf.constant([[0.0, 1.0],
                 [2.0, 3.0]])
var_a = tf.Variable(a)
print(var_a)
```

Variable Qualifications

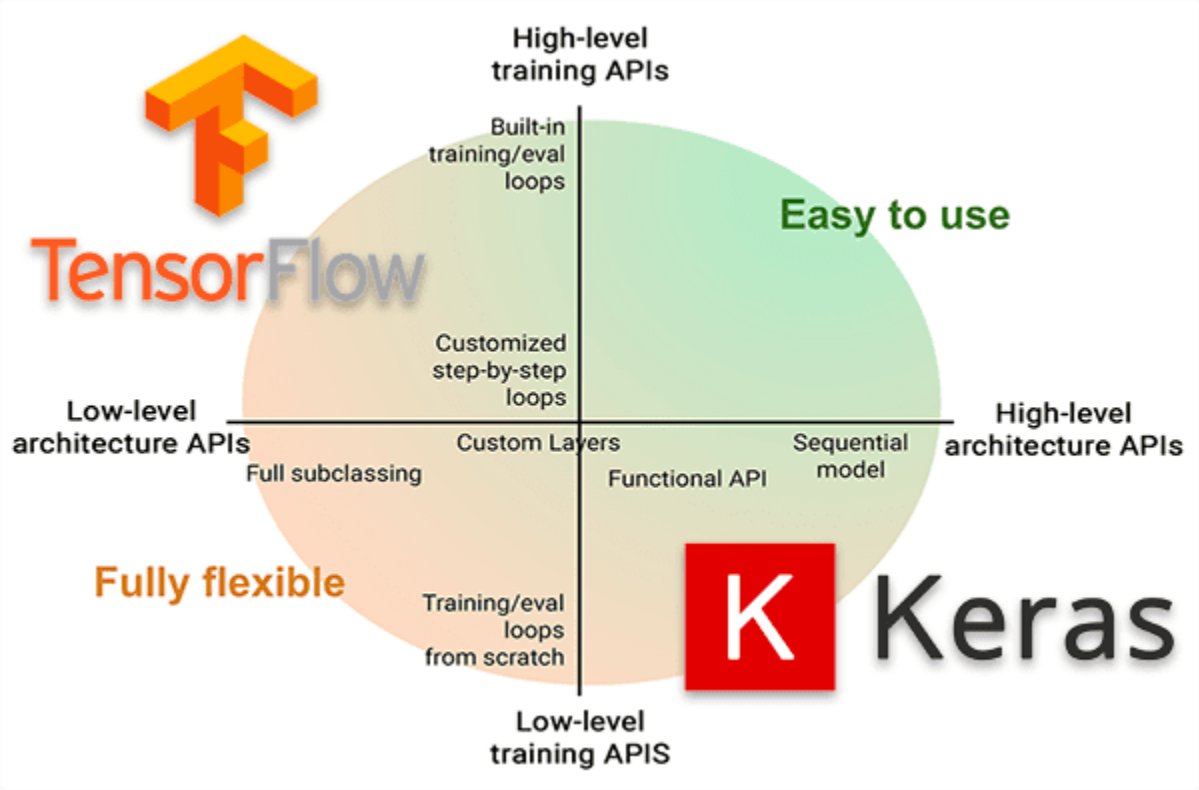
Every Variable must have some properties such as

- Value
- Name
- uniform data type
- Shape
- Rank
- Size

Tensorflow ML Cluster



Tensorflow & Keras



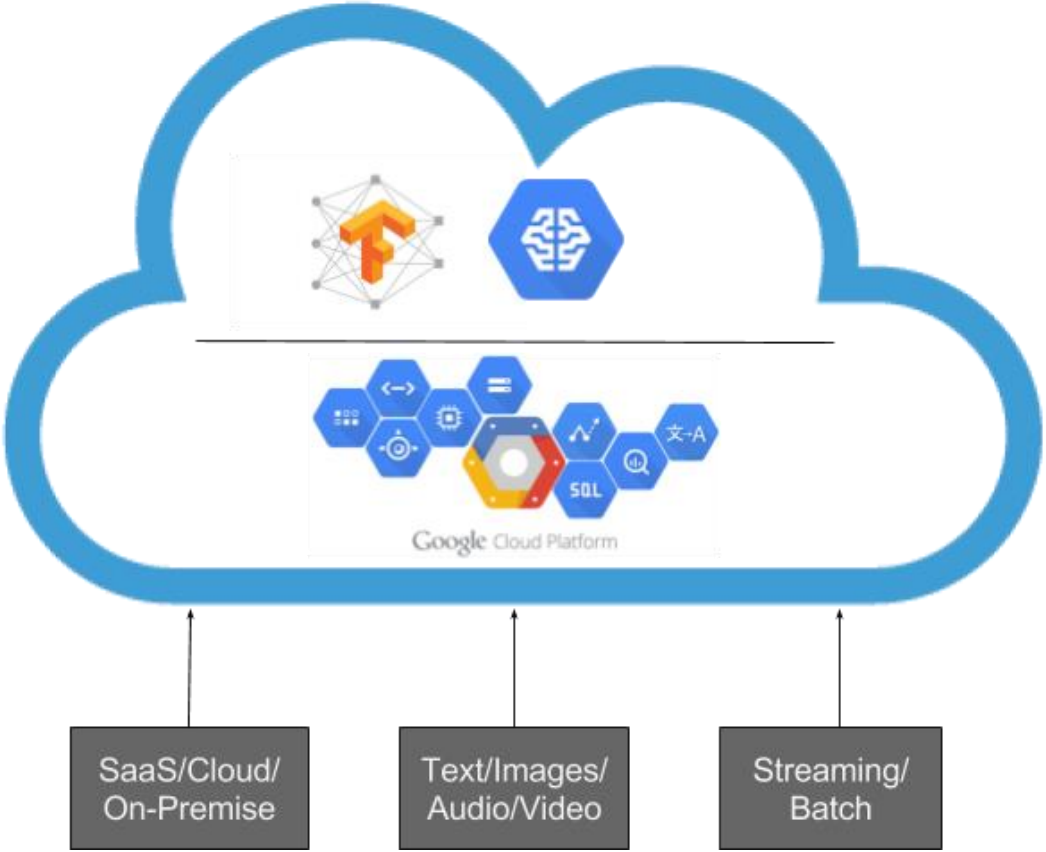
ML Models withTensorFlow

22

Tensorflow Graphs

- Graph execution enables portability outside Python and tends to offer better performance than eager execution. Graph execution means that tensor computations are executed as a TensorFlow graph, sometimes referred to as a tf.Graph or simply a "graph."
- Graphs are data structures that contain a set of tf.Operation objects, which represent units of computation; and tf.Tensor objects, which represent the units of data that flow between operations. They are defined in a tf.Graph context. Since these graphs are data structures, they can be saved, run, and restored all without the original Python code.

Tensorflow ML



Graph Benefits

TensorFlow uses graphs as the format for saved models when it exports them from Python.

Graphs optimizations:

- Statically infer the value of tensors by folding constant nodes in your computation ("constant folding").
- Separate sub-parts of a computation that are independent and split them between threads or devices.
- Simplify arithmetic operations by eliminating common subexpressions.

There is an entire optimization system, Grappler, to perform this and other speedups.

Functions

You create and run a graph in TensorFlow by using tf.function, either as a direct call or as a decorator.

tf.function takes a regular function as input and returns a Function.

A Function is a Python callable that builds TensorFlow graphs from the Python function.

You use a Function in the same way as its Python equivalent.

tf.learn

Deep learning library featuring a higher-level API for TensorFlow.

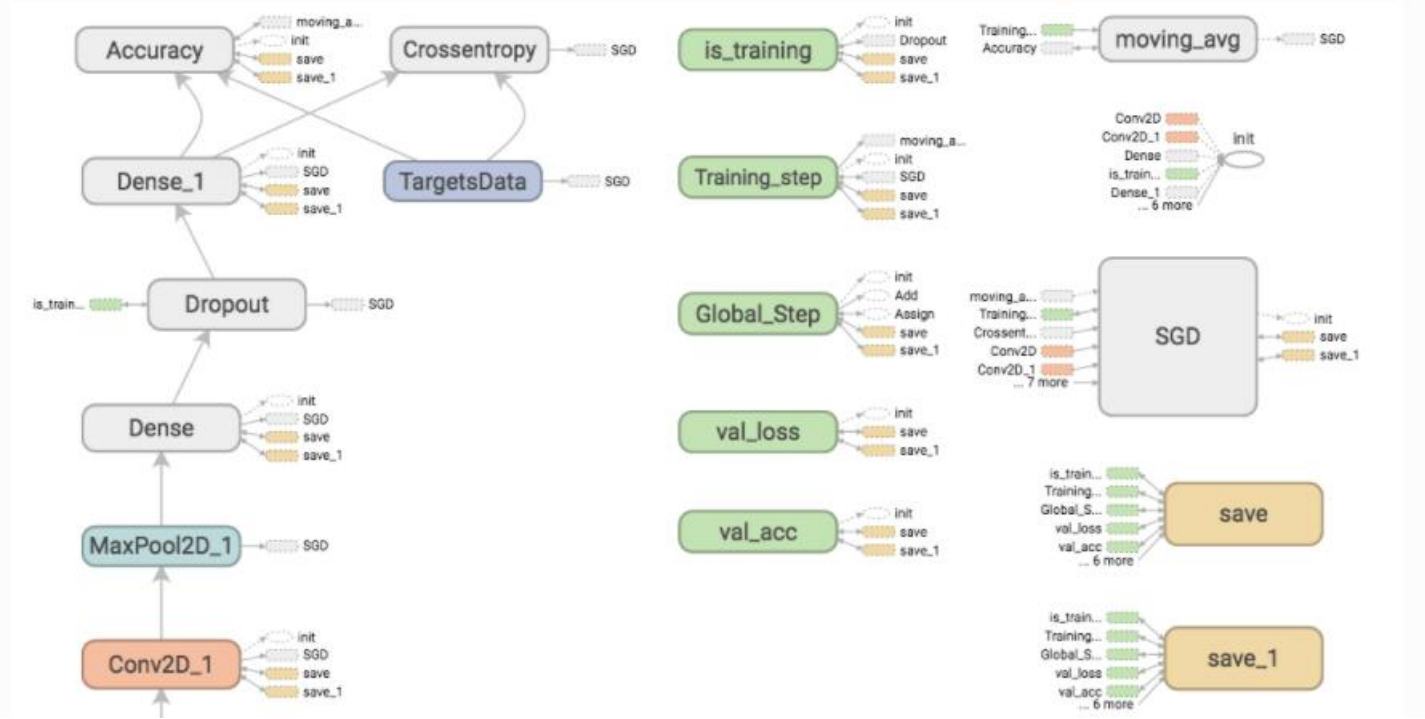
TFlearn is a modular and transparent deep learning library built on top of Tensorflow.

It was designed to provide a higher-level API to TensorFlow in order to facilitate and speed-up experimentations, while remaining fully transparent and compatible with it.

TFLearn Features

- Easy-to-use and understand high-level API for implementing deep neural networks, with tutorial and examples.
- Fast prototyping through highly modular built-in neural network layers, regularizers, optimizers, metrics...
- Full transparency over Tensorflow. All functions are built over tensors and can be used independently of TFLearn.
- Powerful helper functions to train any TensorFlow graph, with support of multiple inputs, outputs and optimizers.
- Easy and beautiful graph visualization, with details about weights, gradients, activations and more...
- Effortless device placement for using multiple CPU/GPU.

Model Visualization



TFLearn Features

- Easy-to-use and understand high-level API for implementing deep neural networks, with tutorial and examples.
- Fast prototyping through highly modular built-in neural network layers, regularizers, optimizers, metrics...
- Full transparency over Tensorflow. All functions are built over tensors and can be used independently of TFLearn.
- Powerful helper functions to train any TensorFlow graph, with support of multiple inputs, outputs and optimizers.
- Easy and beautiful graph visualization, with details about weights, gradients, activations and more...
- Effortless device placement for using multiple CPU/GPU.

Training the Model

We can use machine learning to categorize objects:

- Build a model,
- Train this model on example data, and
- Use the model to make predictions about unknown data.

What is a Model?

- A model is a relationship between features and the label.
- For a classification problem, the model defines the relationship between the data measurements and the predicted categorization.
- Some simple models can be described with a few lines of algebra, but complex machine learning models have a large number of parameters that are difficult to summarize.

What is a Model?

- A model is a relationship between features and the label.
- For a classification problem, the model defines the relationship between the data measurements and the predicted categorization.
- Some simple models can be described with a few lines of algebra, but complex machine learning models have a large number of parameters that are difficult to summarize.

What is a Model?

The representation of what a machine learning system has learned from the training data. Within TensorFlow, model is an overloaded term, which can have either of the following two related meanings:

- The TensorFlow graph that expresses the structure of how a prediction will be computed.
- The particular weights and biases of that TensorFlow graph, which are determined by training.

Models Require Data

For a supervised learning task, you must have at least some labeled data to train a model.

For example, if you want to build a model to predict financial stock data for a given company, you will first need historical training data.

If the company in question has not been public for very long, there may not be adequate training data.

Advanced Neural Networks (ANNs) require a significant amount of data to train the model.

Cloud ML

23

Cloud ML -> AutoML -> Vertex AI

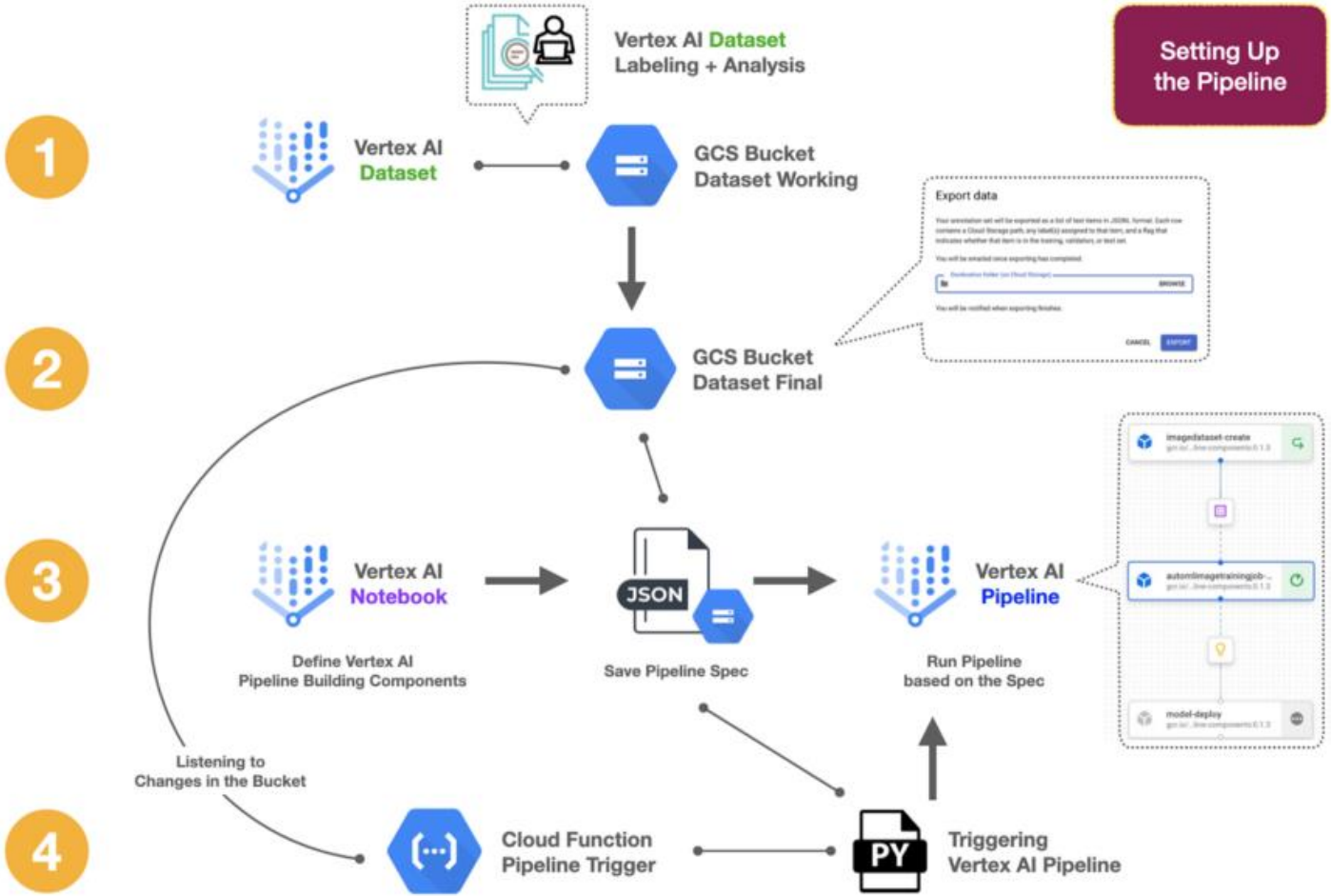
- Vertex AI supports all features and models available in AutoML and AI Platform. However, the client libraries do not support client integration backward compatibility. In other words, you must plan to migrate your resources to benefit from Vertex AI features.

- If you are planning a new project, you should build your code, job, dataset, or model with Vertex AI. This lets you take advantage of the new features and service improvements as they become available. AutoML and AI Platform remain available, but future improvements will be implemented on Vertex AI.

MLOps Pipelines

- We need Vertex AI, Google Cloud Storage, and Cloud Function to build a machine learning pipeline and a trigger.
- Vertex AI is not a single service, but a combination of lots of different AI related services.
- A presently crafted GCP data pipeline will leverage Vertex AI's Dataset, AutoML, Notebook, Pipeline, Model, and Endpoint features altogether.

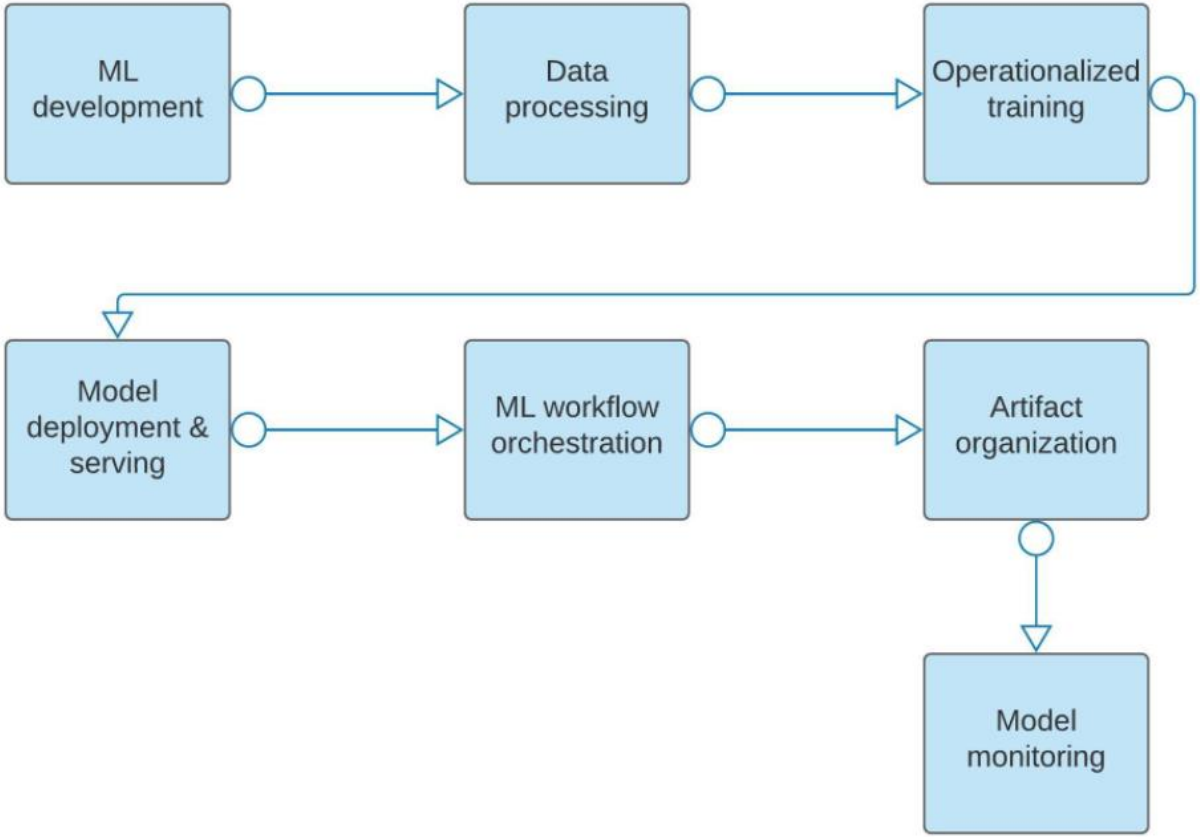
MLOps Pipelines



MLOps Pipelines

- ML development
- Data processing
- Operationalized training
- Model deployment and serving
- ML workflow orchestration
- Artifact organization
- Model monitoring

MLOps Pipelines

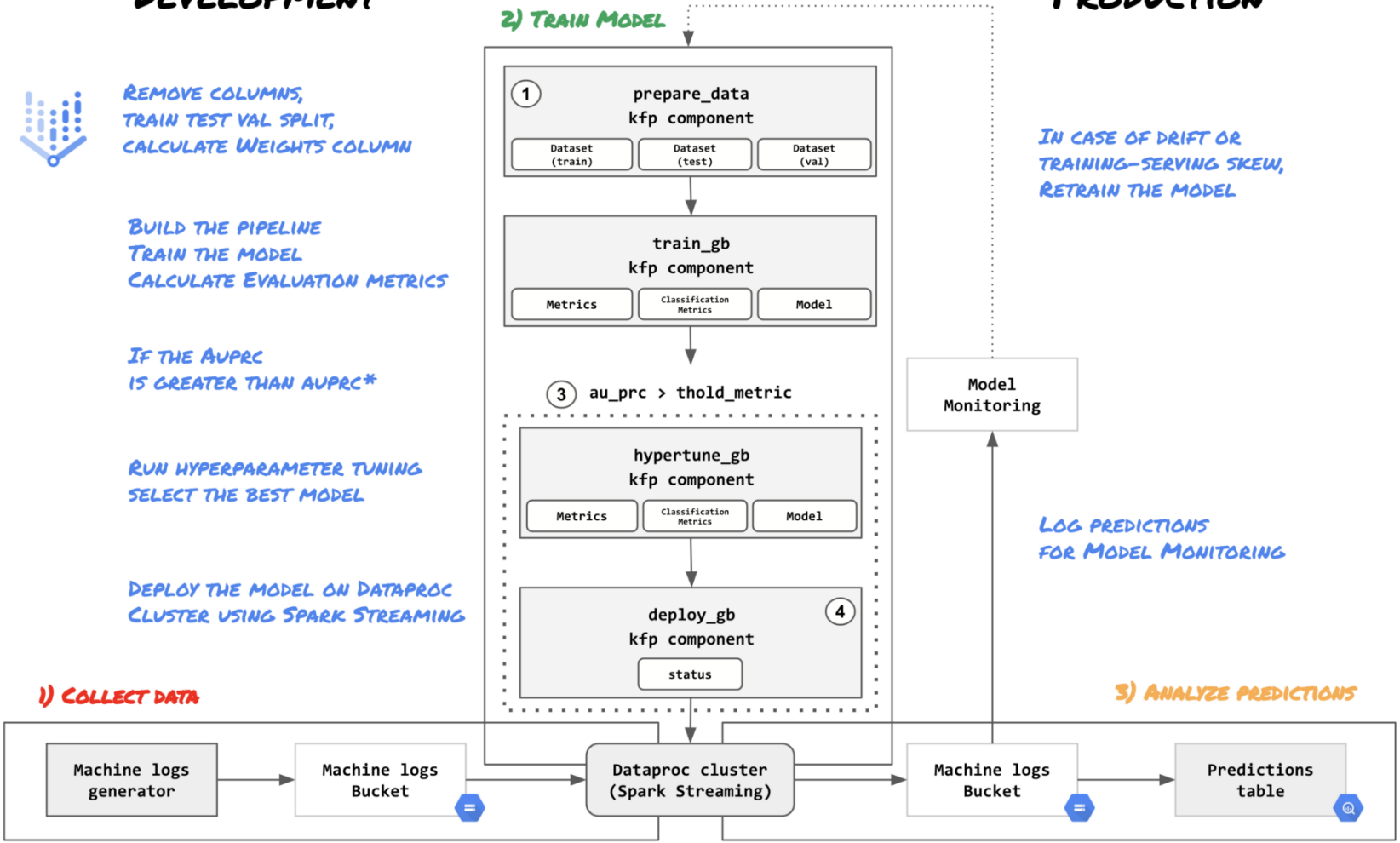


GCP Streaming

24

Development

Production



Dataproc #GCPSketchnote

@PVERGADIA THECLOUDGIRL.DEV

12.07.2020

ERIN: Managing Apache Hadoop cluster is hard!

ERIN: We need a tool that make it easy to gain insights from our data.

SAM: Yes, it is slow, difficult & expensive!

SAM: Dataproc!

How is a typical Hadoop/Spark cluster installation different from Dataproc?

Typical Cluster Activities:
- Setup
- Configure
- Optimize
- Utilize
- Monitor
- Shutdown

Dataproc

90 seconds from start to finish

Cluster Dataproc Activities:
- **Simple** deployment
- **Rapid** development
- **Cost-effective** use

Dataproc

- Familiar open source tools: Crunch, Flink, Hive, Jupyter, druid, Oozie, Hadoop, HBase, presto, Spark
- Customizable machines — Easily autoscale up or down
- Ephemeral clusters on-demand — Save cost
- Tightly Integrated with Google Cloud Platform
- Rapid cluster creation — Spin up a cluster in 90 seconds

How do I move Spark/Hadoop jobs to Dataproc?

Copy Data → Update File Path → Go

GCS

from hdfs:// to gs://

Run your Spark, Hadoop, Hive, or Pig script

Can I right-size clusters per use case?

Storage — Cloud Storage

Hive Metastore — Cloud SQL

Dataproc	Dataproc	Dataproc	Dataproc
375GB local SSD	100GB PD	500GB PD-SSD	150GB PD
m 8, 20	m 16, 30	m 16, 50	m 8, 8
MapReduce ETL	Machine learning	Hive analytics	Business reporting

How do I use ephemeral & long-lived clusters to save cost?

Clusters per job:
- Cluster — Dataproc
- Cluster — Dataproc
- Cluster — Dataproc

Edge Nodes — Compute Engine

Storage — Cloud Storage

Semi-long-lived clusters - group and select by label

Production (1.2):
- Prod 1 — Dataproc
- Prod 2 — Dataproc

Development (Preview):
- Dev cluster — Dataproc

Client, Client, Client

Clients

How does it work?

Disaggregates storage & compute

External Applications
- Application Logs

Data Sources
- Storage — Cloud Storage
- Storage — BigQuery
- Storage — Cloud Bigtable

Production — Dataproc

External Applications
- Test — Dataproc
- Development — Dataproc

Cluster monitoring
- Monitor — Stackdriver
- Logs — Logging

Data Sources
- Storage — Cloud Storage
- Storage — BigQuery
- Storage — Cloud Bigtable

Data science
- Analysis — Notebook

Streaming Workflow Components

- **prepare_data** component to ingest data, do some simple data preparation and creates Vertex Datasets to train model and evaluate the model
- **train_model** component to train a one simple GBTClassifier using PySpark MLlib library, calculate several classification metrics such as areaUnderPRC, confusion matrix, accuracy and log them using Vertex Metadata[2]

Then if the area under the precision-recall curve is greater than a predefined metric threshold:

- **hypertune_gb** component finds the best model and parameters using ParamGridBuilder and CrossValidator methods. All metrics and hypertuning metadata are stored in the pipeline lineage again using metadata service.
- **deploy_gb** component deploys the final model on Google Cloud Dataproc cluster using Spark Streaming

Streaming Workflow Components

- Create the Dataproc cluster
- Check the Cluster state
- Submit the Job
- Check the Job state
- Delete the cluster

DECISION TREE → Apache Hadoop to

#GCPSketchnote

@PVERGADIA THECLOUDGIRL.DEV

01.29.2021

QUESTION #1

Got NoSQL workloads?

→ APACHE HBASE → Need to use co-processors or SQL with APACHE PHOENIX

- YES → DATAPROC
- NO → CLOUD BIGTABLE

QUESTION #2

Processing streaming data?

- Apache Beam → CLOUD DATAFLOW
- Apache Spark or kafka → DATAPROC

QUESTION #3

Doing interactive data analysis or ad-hoc querying?

- SQL with HIVE presto or similar → Want to keep the SQL queries as they are?
  - YES → DATAPROC
  - NO Interested in managed solution → BIGQUERY
- Apache Spark with interactive web notebooks → DATAPROC

in combination with Jupyter or Zeppelin (Optional)

QUESTION #4

Doing ETL/ELT or batch processing?

- Apache Airflow OOZIE for workflow orchestration
  - YES → DATAPROC — Run Oozie on Dataproc
  - NO Interested in managed solution → CLOUD COMPOSER — Fully Managed Apache Airflow
- MapReduce, Apache Spark, Apache Pig, HIVE → DATAPROC

How do I migrate Apache Hadoop cluster to ?

IT DEPENDS...

Streaming Pipelines

- Unbounded PCollections, or unbounded collections, represent data in streaming pipelines. An unbounded collection contains data from a continuously updating data source such as Pub/Sub.
- You cannot use only a key to group elements in an unbounded collection. There might be infinitely many elements for a given key in streaming data because the data source constantly adds new elements. You can use windows, watermarks, and triggers to aggregate elements in unbounded collections.
- The concept of windows also applies to bounded PCollections that represent data in batch pipelines.

Windows

Windowing functions divide unbounded collections into logical components, or windows. Windowing functions group unbounded collections by the timestamps of the individual elements. Each window contains a finite number of elements.

You set the following windows with the Apache Beam SDK or Dataflow SQL streaming extensions:

- Tumbling windows (called fixed windows in Apache Beam)
- Hopping windows (called sliding windows in Apache Beam)
- Session windows

Watermarks

A watermark is a threshold that indicates when Dataflow expects all of the data in a window to have arrived. If new data arrives with a timestamp that's in the window but older than the watermark, the data is considered late data.

Dataflow tracks watermarks because of the following:

Data is not guaranteed to arrive in time order or at predictable intervals. Data events are not guaranteed to appear in pipelines in the same order that they were generated.

The data source determines the watermark. You can allow late data with the Apache Beam SDK. Dataflow SQL does not process late data.

Triggers

Triggers determine when to emit aggregated results as data arrives. By default, results are emitted when the watermark passes the end of the window.

You can use the Apache Beam SDK to create or modify triggers for each collection in a streaming pipeline. You cannot set triggers with Dataflow SQL.

The Apache Beam SDK can set triggers that operate on any combination of the following conditions:

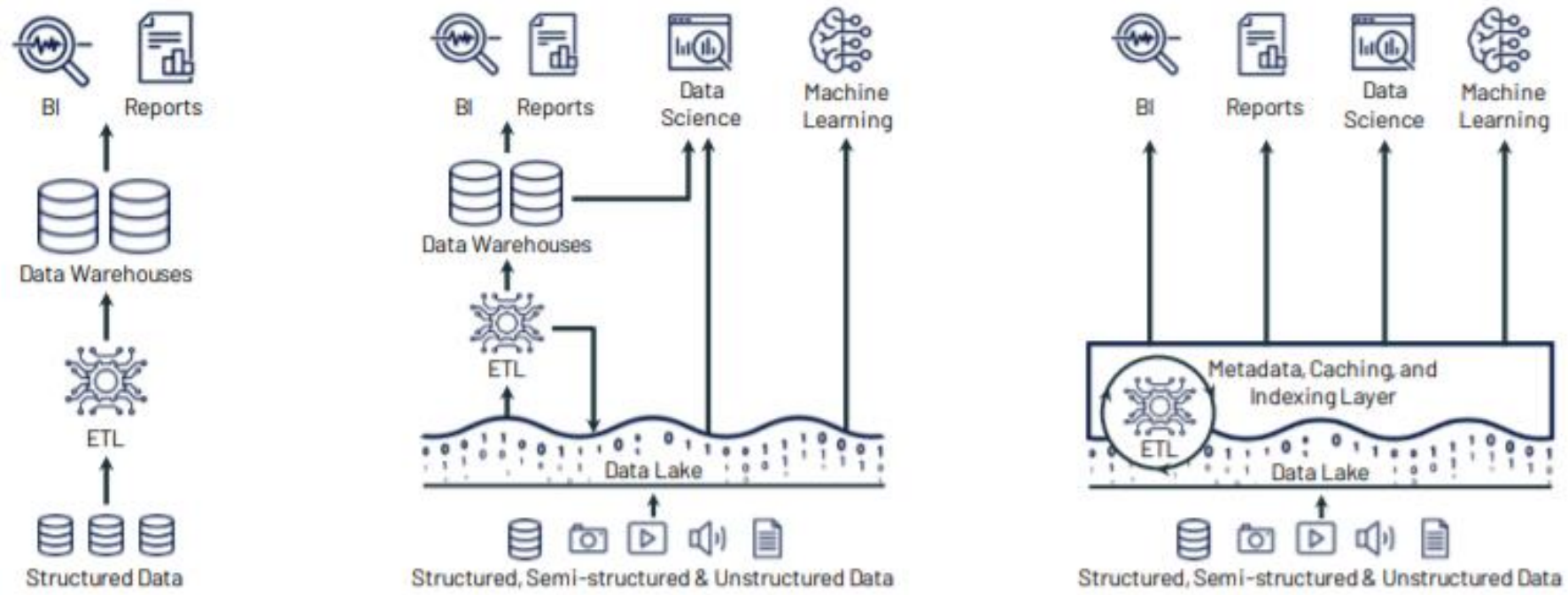
- Event time, as indicated by the timestamp on each data element.
- Processing time, which is the time that the data element is processed at any given stage in the pipeline.
- The number of data elements in a collection.

Building a Lakehouse

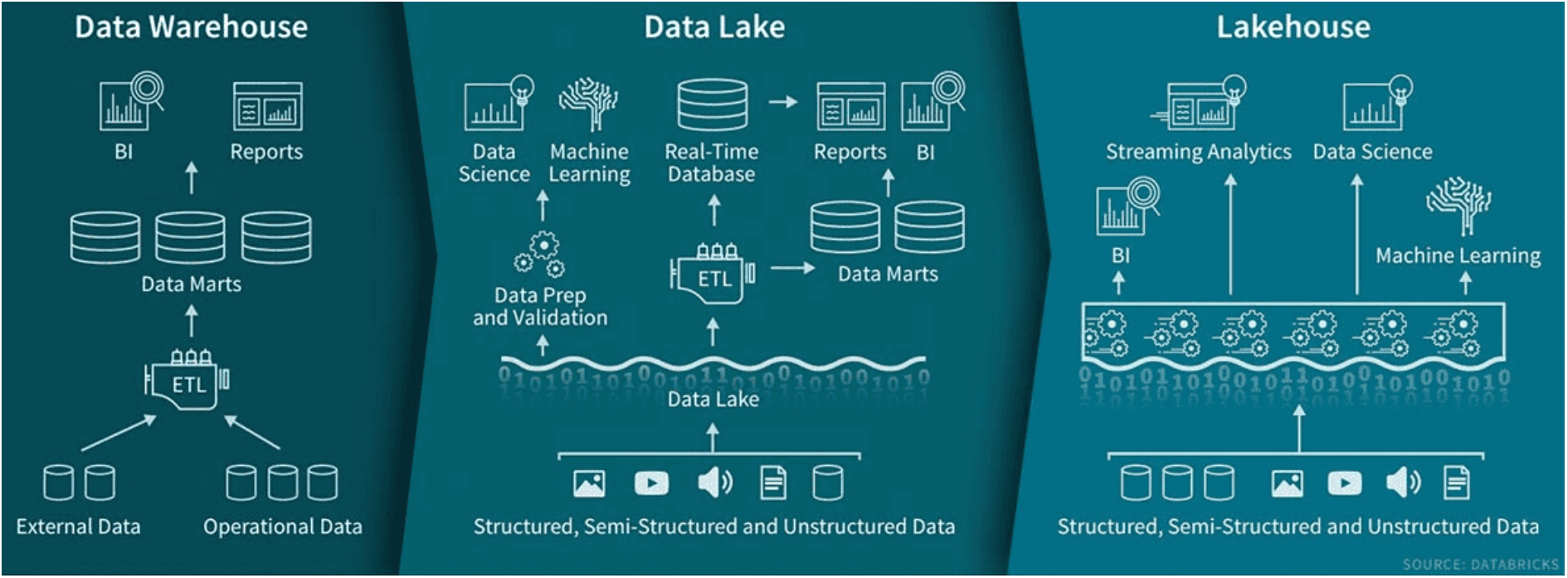
25

Data Warehouse
Data Lake
Lakehouse

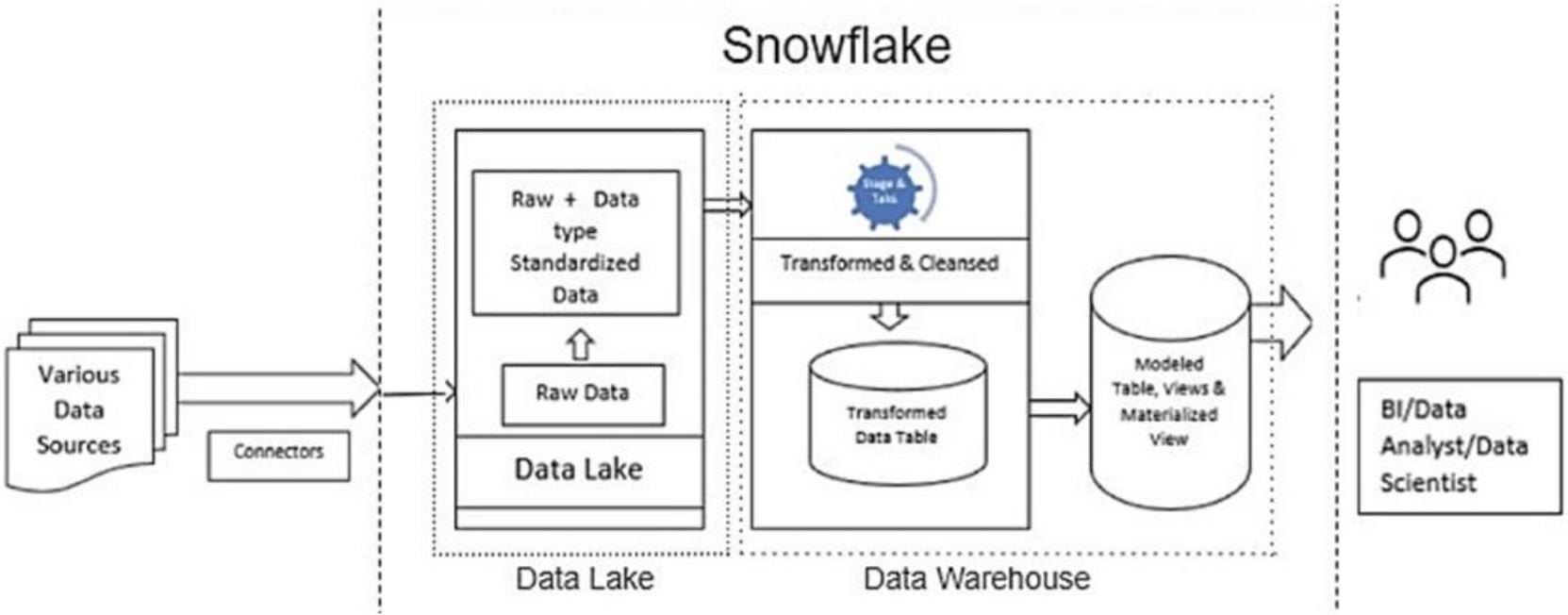
Big Data Evolution



Big Data Evolution



DATA WAREHOUSE, LAKE, LAKEHOUSE



LAKEHOUSE CHARACTERISTICS

- Transactional reliability
- Horizontal scalability
- Single source of truth
- ACID guarantees
- Scalable data and metadata handling:
- Audit History and Time travel
- The Lakehouse transaction log records details about Schema enforcement and schema evolution
- Support for deletes updates, and merge
- Streaming and batch unification

ACID GUARANTEES

- Lakehouse should ensure that all data changes written to storage are committed for durability and made visible to readers atomically.
- In other words, no more partial or corrupted files!
- Any discussion of acid guarantees requires advanced log handling

SCALABLE DATA AND METADATA HANDLING

- Lakehouse ensures that all data changes written to storage are committed for durability and made visible to readers atomically.
- In other words, no more partial or corrupted files!
- Any discussion of acid guarantees requires advanced log handling

AUDIT HISTORY AND TIME TRAVEL

- The Lakehouse transaction log should record details about every change made to data providing a full audit trail of the changes.
- These data snapshots enable developers to access and revert to earlier versions of data for audits, rollbacks, or to reproduce experiments.

SCHEMA ENFORCEMENT AND EVOLUTION

- Lakehouse should automatically prevent the insertion of data with an incorrect schema, i.e. not matching the table schema.
- When needed, it allows the table schema to be explicitly and safely evolved to accommodate ever-change data.

DELETE, UPDATE AND MERGE SUPPORT

- Most distributed processing frameworks do not support atomic data modification operations on data lakes.
- Lakehouse should support merge, update, and delete operations to enable complex use cases including but not limited to change-data- capture (CDC), slowly-changing-dimension (SCD) operations, and streaming upserts.

STREAMING AND BATCH UNIFICATION

- A Lakehouse table has the ability to work both in batch and as a streaming source and sink.
- The ability to work across a wide variety of latencies ranging from streaming data ingest to batch historic backfill to interactive queries all just work out of the box.

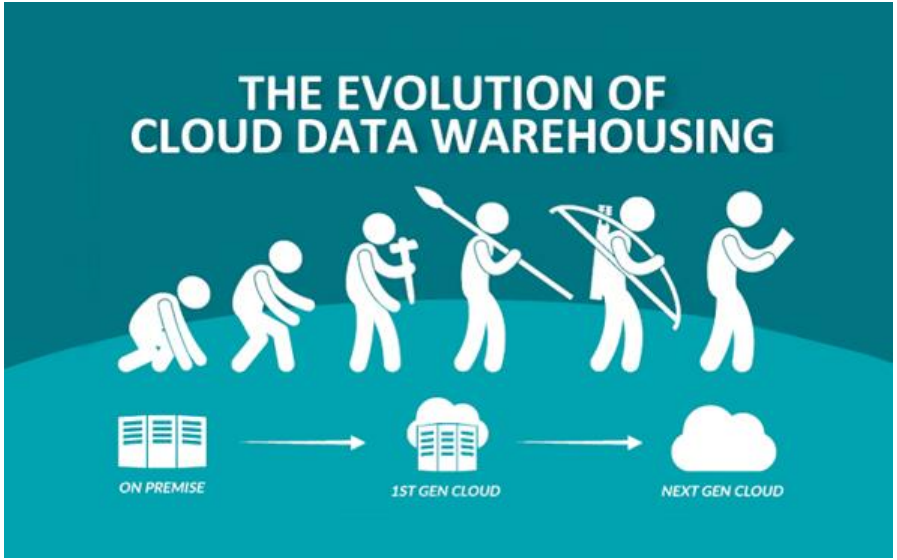
SINGLE SOURCE OF TRUTH

- The concept of a single source of truth is important because, over time, processing jobs will fail in your data lake.
- The result is that processing jobs or tasks will leave partial files that do not get cleaned up.
- Subsequent processing or queries will not be able to ascertain which files should or should not be included.
- Lakehouse should ensure that any user's version of a table is synchronized with the master record as of the most recent query.

Building a Cloud Data Warehouse

26

DATA WAREHOUSING EVOLUTION



DATA WAREHOUSE CHARACTERISTICS

Warehouses are required for queries, as well as all DML operations, including loading data into tables. A warehouse is defined by its size, as well as the other properties that can be set to help control and automate warehouse activity.

Warehouses can be started and stopped at any time. They can also be resized at any time, even while running, to accommodate the need for more or less compute resources, based on the type of operations being performed by the warehouse.

SNOWFLAKE DATA WAREHOUSE

A **data warehouse** is a relational database that is designed for analytical rather than transactional work.

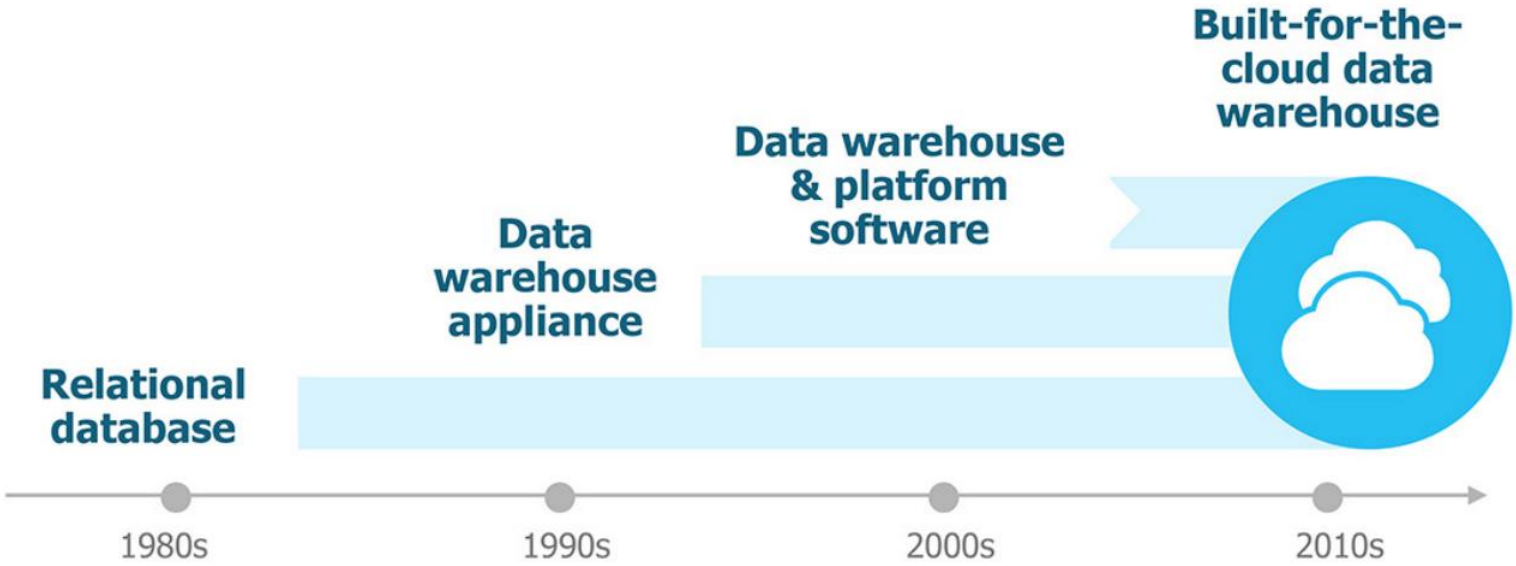
It collects and aggregates data from one or many sources so it can be analyzed to produce business insights.

It serves as a federated repository for all or certain data sets collected by a business's operational systems. **Snowflake is a cloud data warehouse**

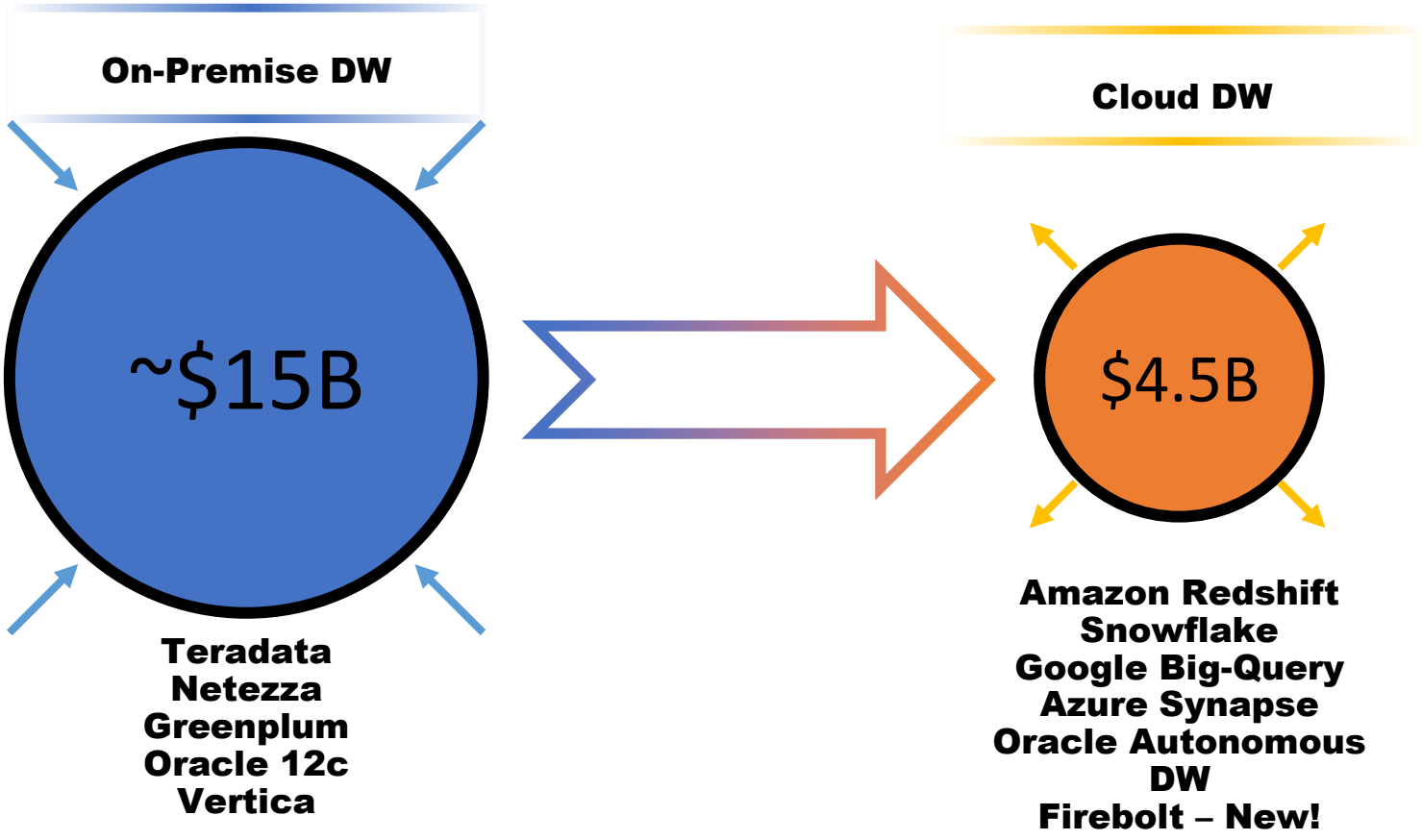
SNOWFLAKE DATA WAREHOUSE

- Founded in 2012 – product launched in 2015
- Runs on AWS, GCP & Azure
- Raised $1.4B prior to IPO
- Snowflake had 4,532 customers as of May’2021
- Completed IPO in September
- Largest software IPO
- Currently valued at ~75B$
- ~3,300 employees

CLOUD DATA WAREHOUSING

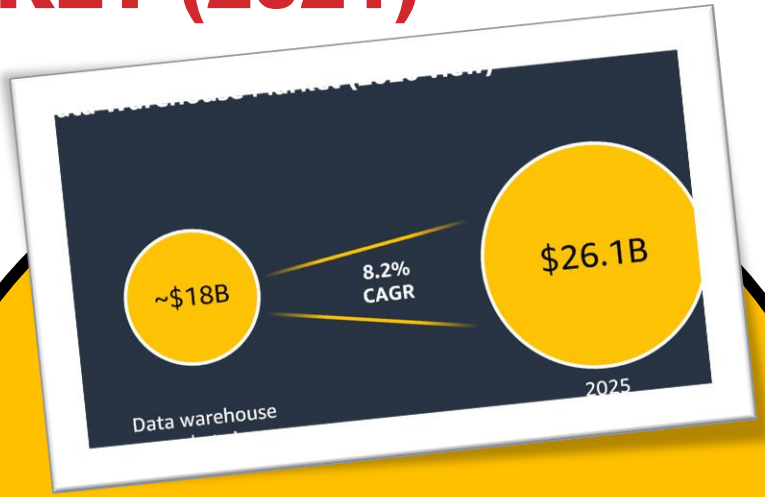


CLOUD DATA WAREHOUSING



DATA WAREHOUSE MARKET (2021)

Source: Allied Market Research



~$21.2B

2021

Data warehouse market size

10.7% CAGR

$51.2B

2028

ADAPTING TO INCREASING DEMANDS FOR DATA ACCESS

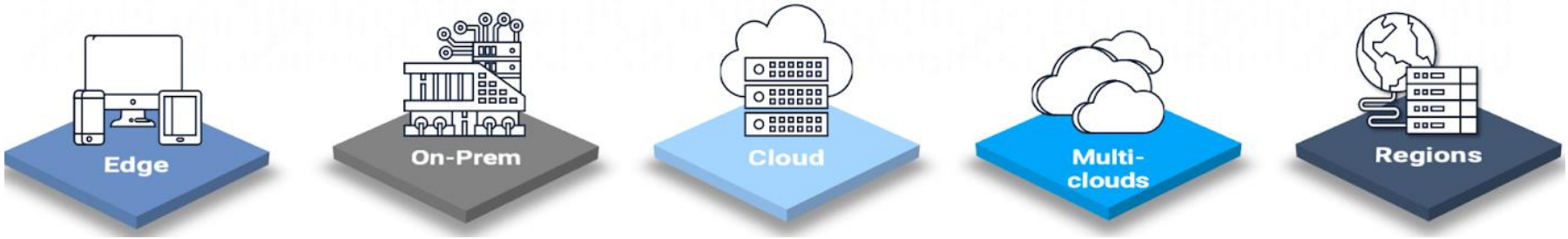
5.8B
endpoints in use in 2020 at 21% growth
- Gartner

$200B
data center market in 2021 at 6% growth
- Gartner

$335B
global public cloud market by 2022 at 12% growth
- Gartner

40%
of companies in the cloud will use 2+ providers
- Gartner

70%
think cloud is more complex due to local privacy regulations
- Privacera



WHAT CUSTOMERS DO WITH CLOUD DW

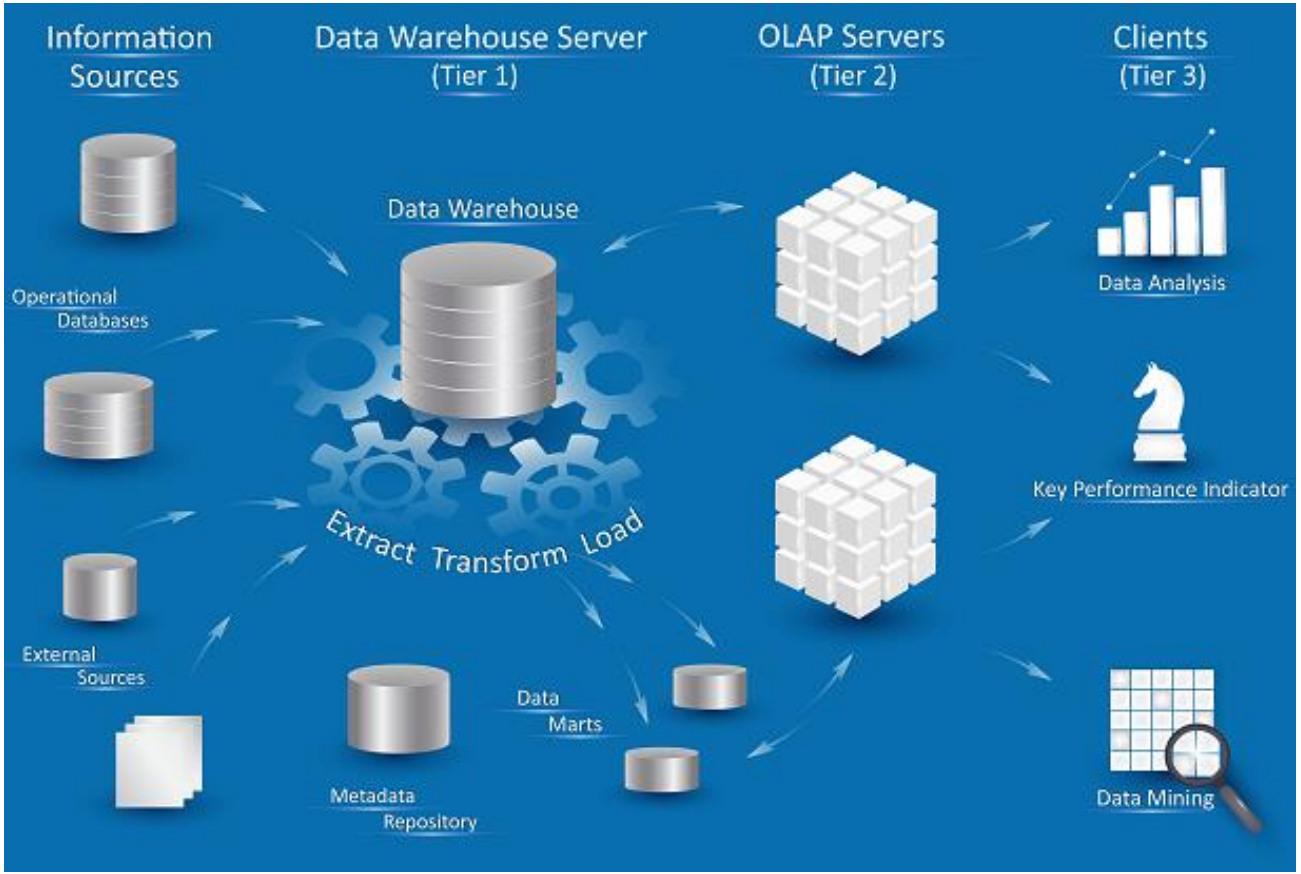
- Gaming company **replaced Hadoop + SQL database with Snowflake**
- Consumer retailer modernizing DW by **replacing the legacy appliance with Snowflake**
- Market research company **consolidated data marts** to reduce costs and data silos
- Mobile analytics **company shares live data** with clients

DATA WAREHOUSE SELECTION CRITERIA

When you're in the market for a **data warehouse**, a **checklist of criteria** will help determine which alternative best meets your needs.

- Supports Existing Skills, Tools, and Expertise
- Saves Your Organization Money
- Provides Data Resiliency and Recovery
- Secures Data at Rest and in Transit
- Streamlining the data pipeline
- Optimizes Your Time to Value

BUSINESS INTELLIGENCE



DATA WAREHOUSE VS. DATABASE

A data warehouse focuses on collecting data from multiple sources to facilitate broad access and analysis. They specialize in data aggregation and providing a longer view of an organization's data over time.

A data warehouse is optimized to store large volumes of historical data and enables fast and complex querying of that data. Standard operational databases focus on transactional functions such as real-time data updates for ongoing business processes.

DATA WAREHOUSE KEY FUNCTIONS

Data warehousing has two key functions.

First, it serves as a **historical repository** for integrating the information and data that is needed by the business, which may come from a variety of different sources.

Second, it serves as a **query execution and processing engine** for that data, enabling end users to interact with the data that is stored in the database.

COMPLEX QUERY EXECUTION ENGINE

Complex queries are very difficult to run without a temporary pause of database update operations. A frequently paused transactional database will inevitably lead to data errors and gaps.

Therefore, a data warehouse serves as a **separate platform for aggregation** across multiple sources and then for analytics tasks across those diverse sources. This separation of roles allows databases to remain focused on purely transactional jobs without interruption.

DIGITAL DATA SUPPLY CHAIN

DATA WAREHOUSE COMPONENTS

A data warehouse usually consists of data sources from operational and transactional systems (ERP, CRM, finance apps, IoT devices, mobile and online systems) as well as:

- A data staging area for aggregation and cleaning
- A presentation/access area where data is warehoused for analytics (querying, reporting) and sharing
- A range of data tool integrations or APIs (BI software, ingestion and ETL tools, etc.).

DECISION TREE → Apache Hadoop to

#GCPSketchnote

@PVERGADIA THECLOUDGIRL.DEV

01.29.2021

QUESTION #1

Got NoSQL workloads?

→ APACHE HBASE → Need to use co-processors or SQL with APACHE PHOENIX

YES → DATAPROC

NO → CLOUD BIGTABLE

QUESTION #2

Processing streaming data?

Apache Beam → CLOUD DATAFLOW

Spark or kafka → DATAPROC

QUESTION #3

Doing interactive data analysis or ad-hoc querying?

SQL with hive presto or similar → Want to keep the SQL queries as they are?

YES → DATAPROC

NO Interested in managed solution → BIGQUERY

Spark with interactive web notebooks → DATAPROC

in combination with Jupyter or Zeppelin (Optional)

QUESTION #4

Doing ETL/ELT or batch processing?

Apache Airflow OOZIE for workflow orchestration

YES → DATAPROC

Run Oozie on Dataproc

NO Interested in managed solution → CLOUD COMPOSER

Fully Managed Apache Airflow

MapReduce Spark Apache Pig hive → DATAPROC

How do I migrate Apache Hadoop cluster to ?

IT DEPENDS...

Best Practices

27

Next Steps

28