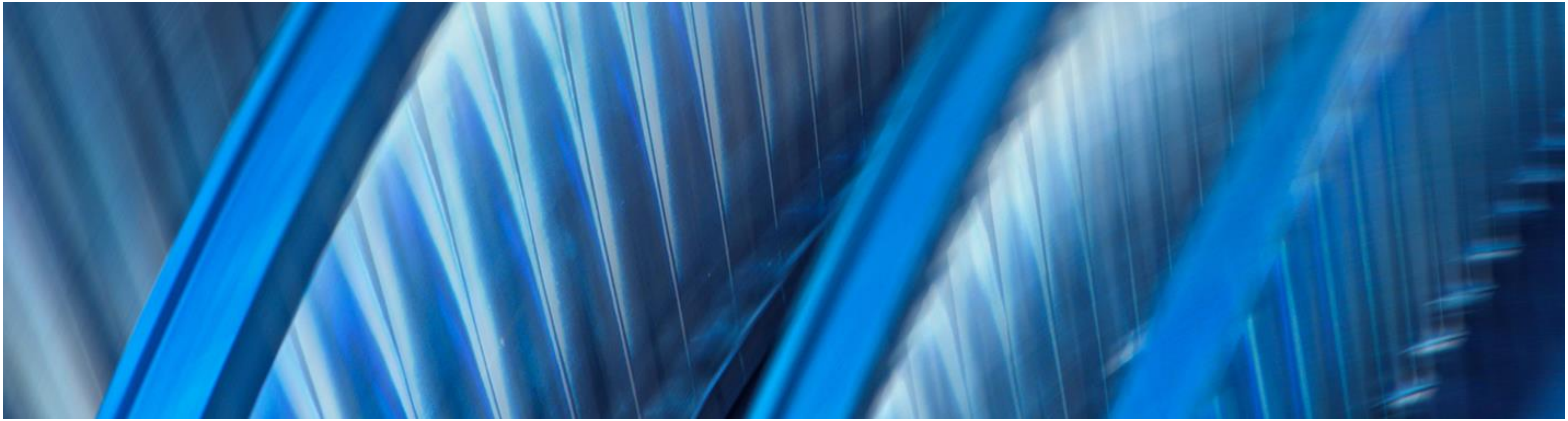
zühlke
empowering ideas



Data Analytics Platforms in the Cloud

Large Scale Data Platforms in the Cloud

Two marketing customers

zühlke
empowering ideas

60TB

Raw data on linear blob storage

55B

Number of rows in MPP Database

300

Number of nodes in transparent elastic scaling

800

Websites with simultanious click stream collection

Real-time processing of click stream data

Automated Machine Learning Lifecycle

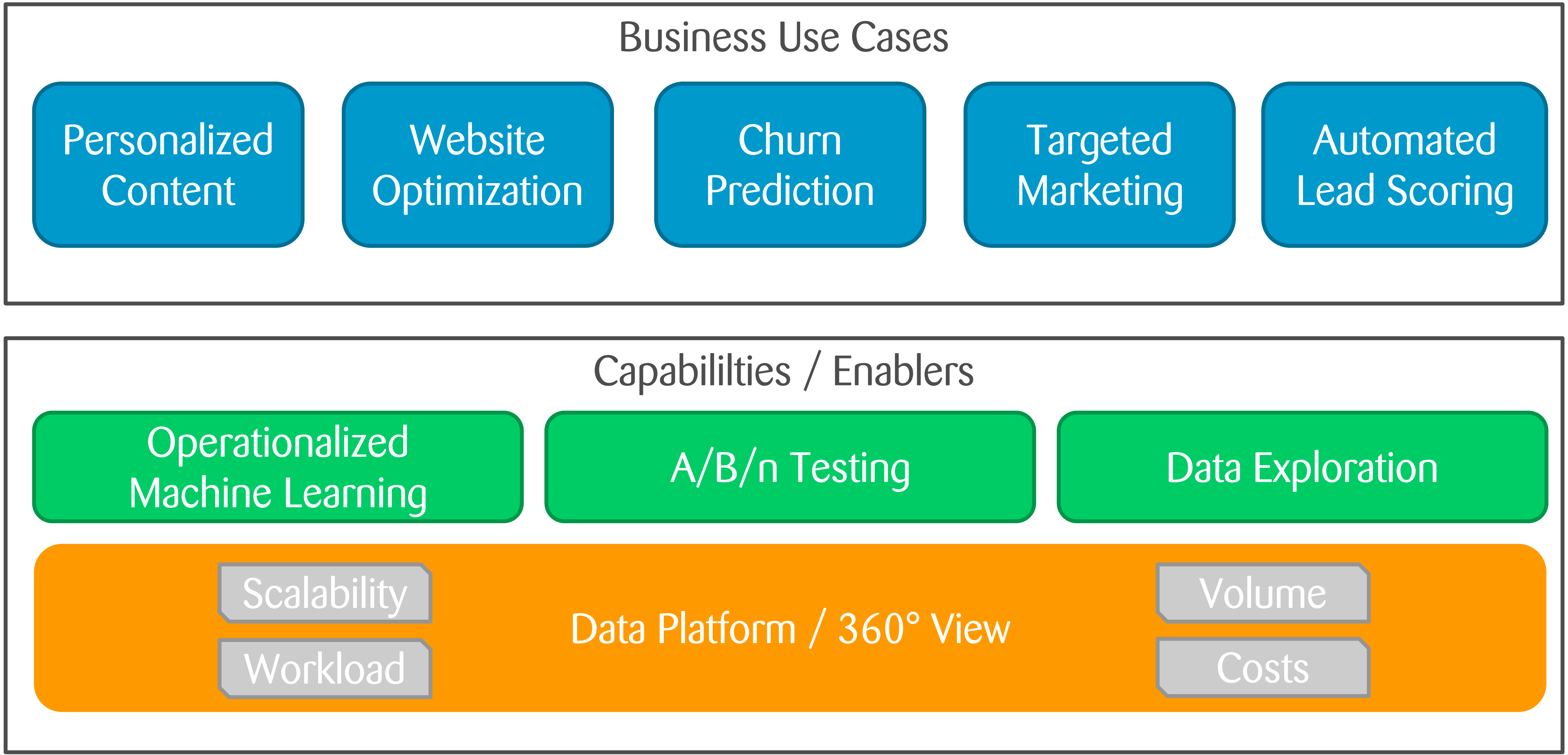
Collaboration with Google

Joel Akeret
Expert Data Scientist
Zühlke Engineering AG

Hendrik Schöneberg
Lead Software Architect
Zühlke Engineering AG

Why data analytics platforms?

Drivers for Data Analytics Platforms



Business Use Cases

Personalized Content

Website Optimization

Churn Prediction

Targeted Marketing

Automated Lead Scoring

Capabililties / Enablers

Operationalized Machine Learning

A/B/n Testing

Data Exploration

Data Platform / 360° View

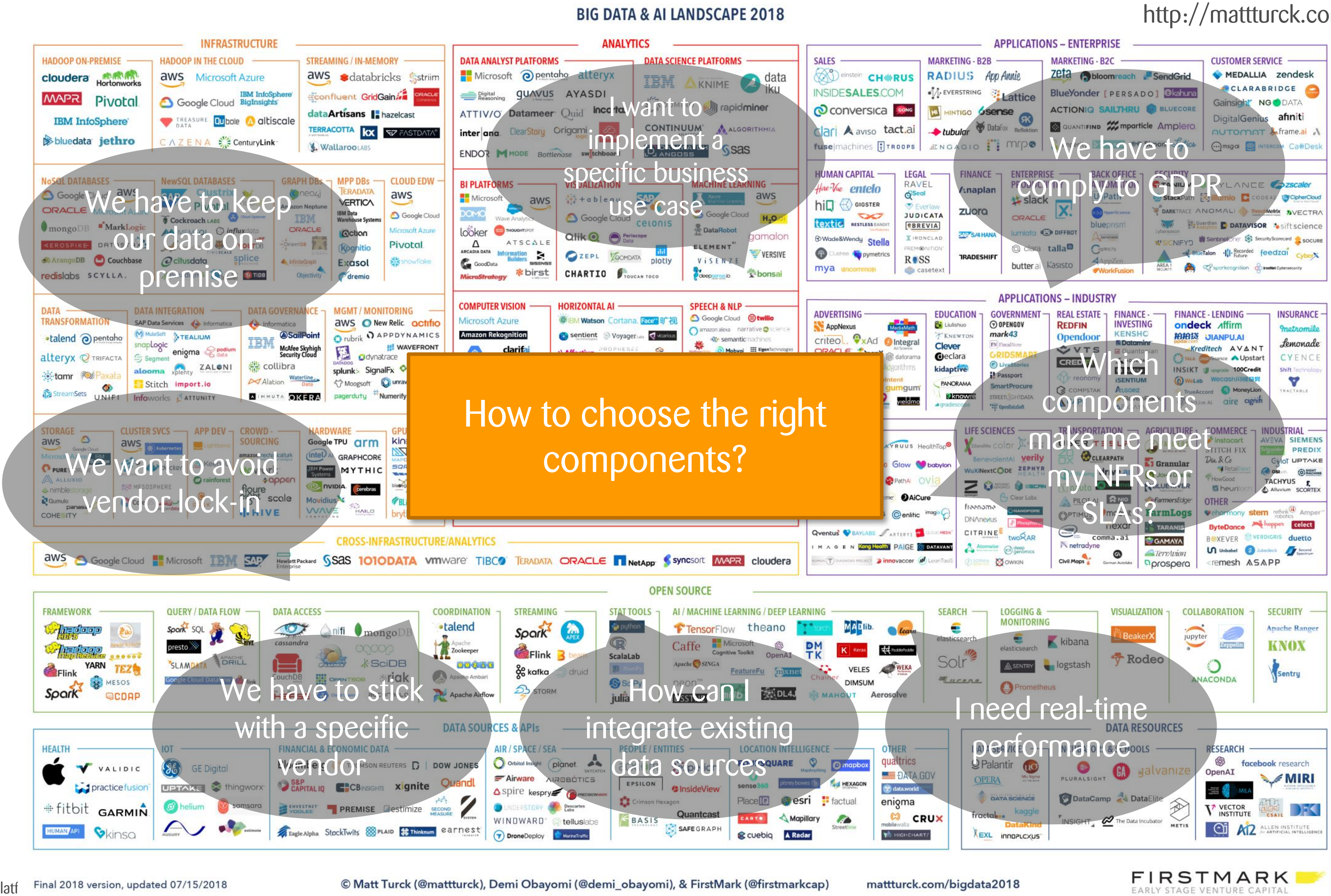
Scalability

Workload

Volume

Costs

http://mattturck.com/bigdata2018



We have to keep our data on-premise

I want to implement a specific business use case

We have to comply to GDPR

How to choose the right components?

Which components make me meet my NFRs or SLAs?

We want to avoid vendor lock-in

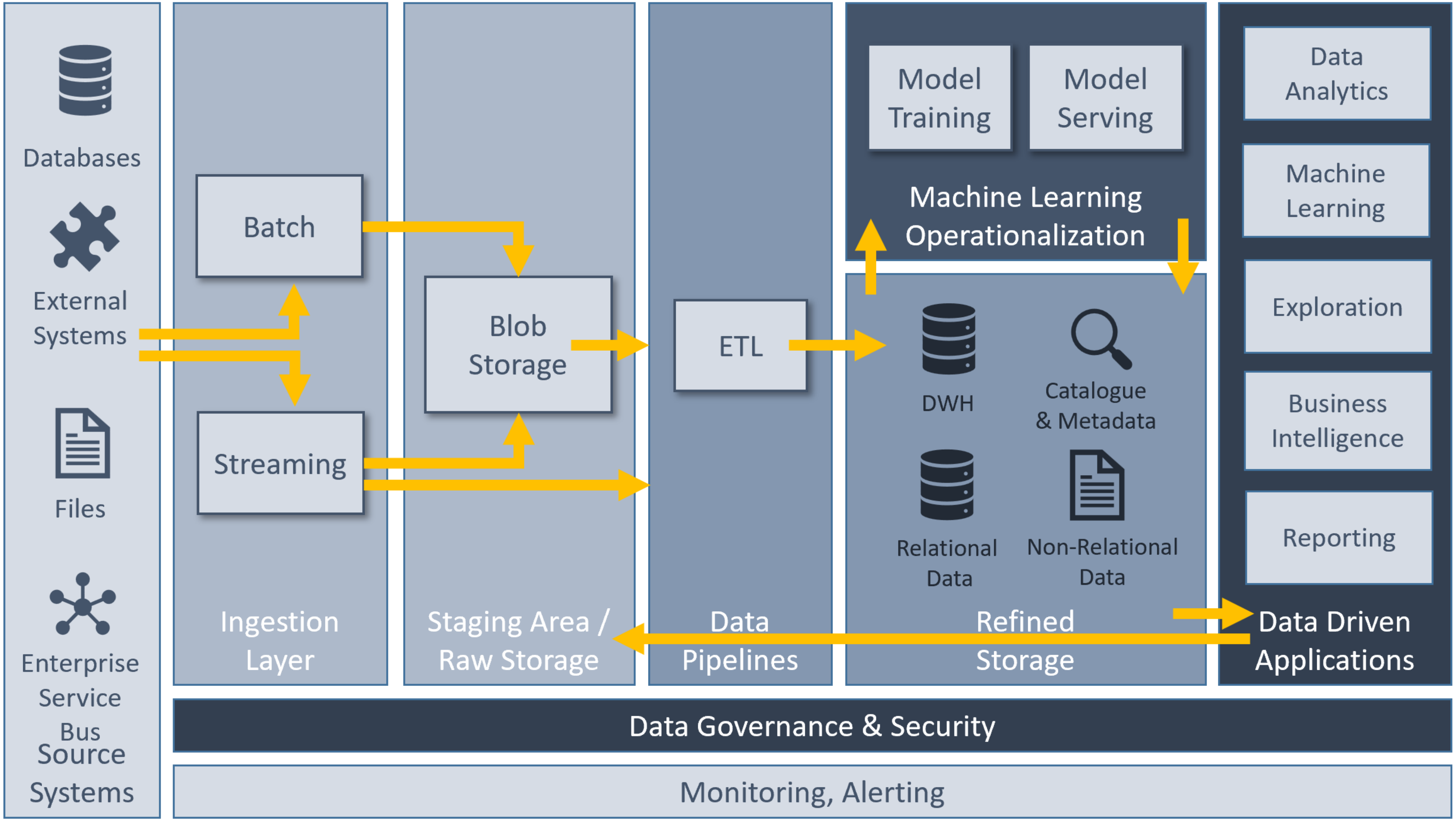
We have to stick with a specific vendor

How can I integrate existing data sources

I need real-time performance

Data platform architecture

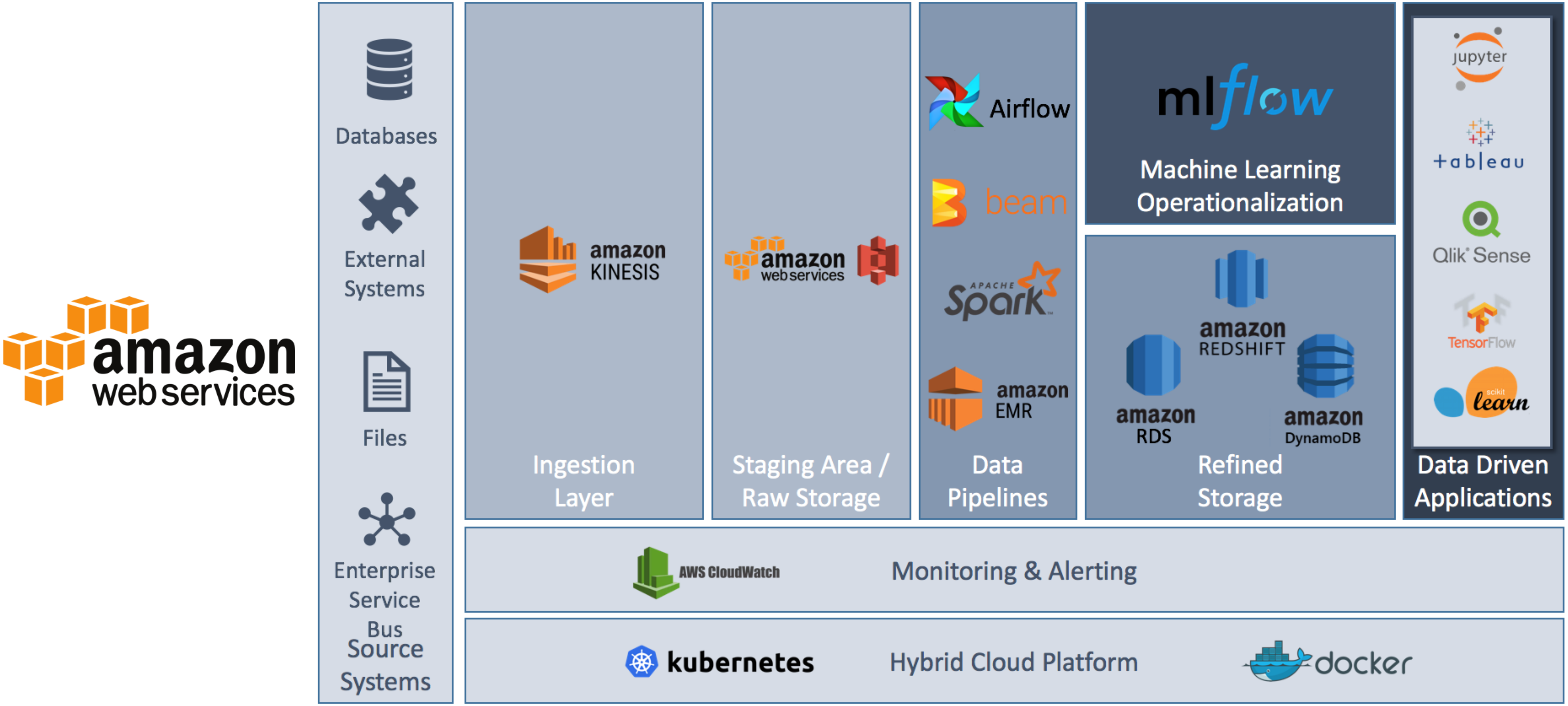
Platform Architecture Overview



Technological choices

Amazon Web Services

zühlke
empowering ideas



Databases

External Systems

Files

Enterprise Service Bus

Source Systems

Ingestion Layer

Staging Area / Raw Storage

Data Pipelines

Machine Learning Operationalization

Refined Storage

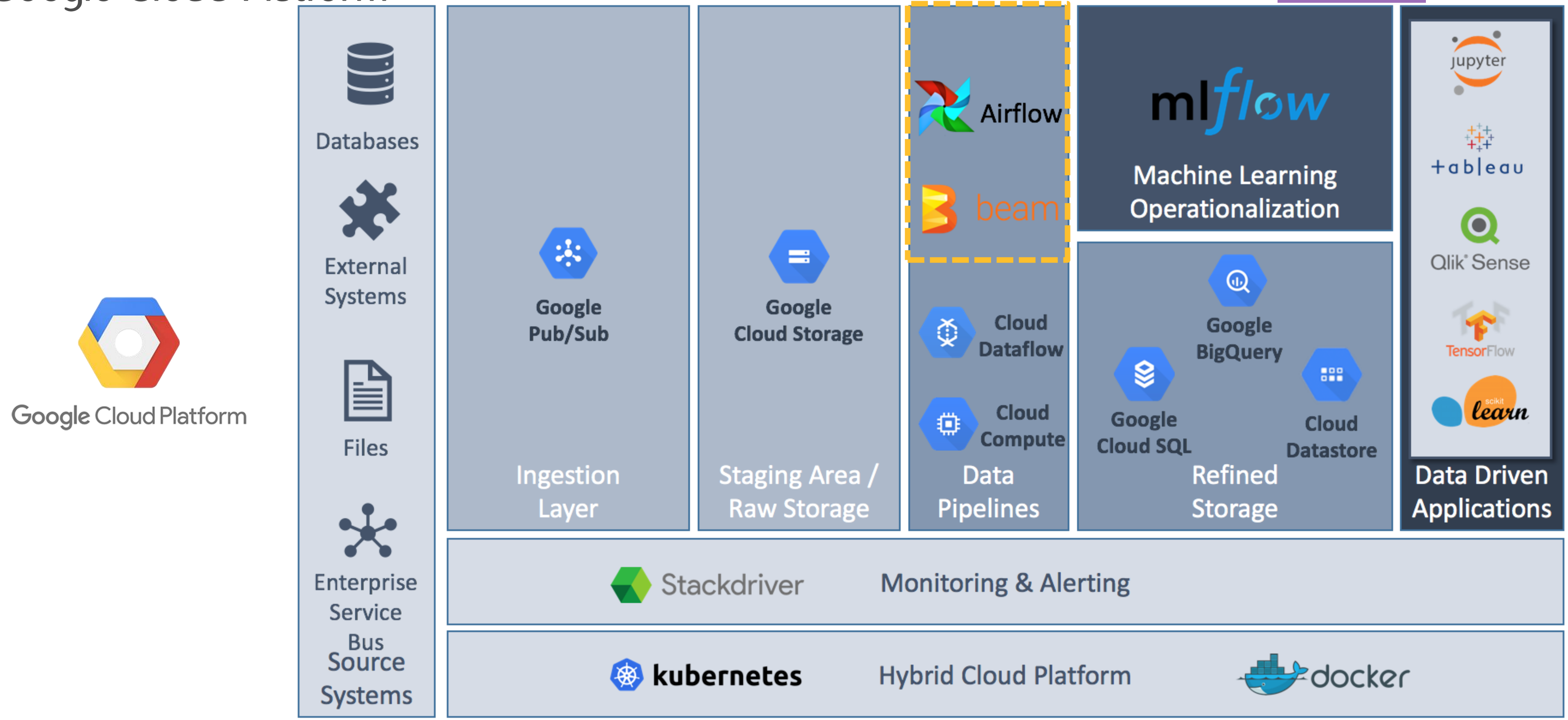
Data Driven Applications

Monitoring & Alerting

Hybrid Cloud Platform

Technological choices

Google Cloud Platform

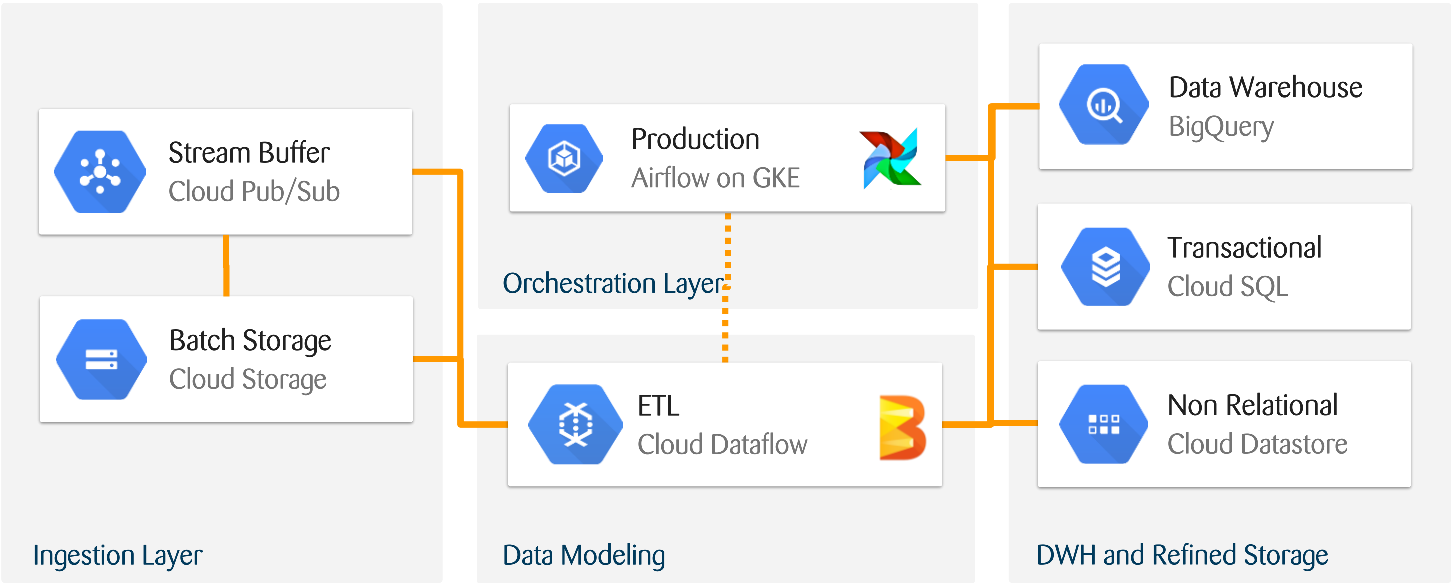


Deepdive
Data Processing Pipelines

Data Platform Architecture

Core Data Ingestion and Processing

- Workflows orchestrated by **Apache Airflow**
- Execution of parallelizable data transformations done with **Apache Beam** on **Google Cloud Dataflow**

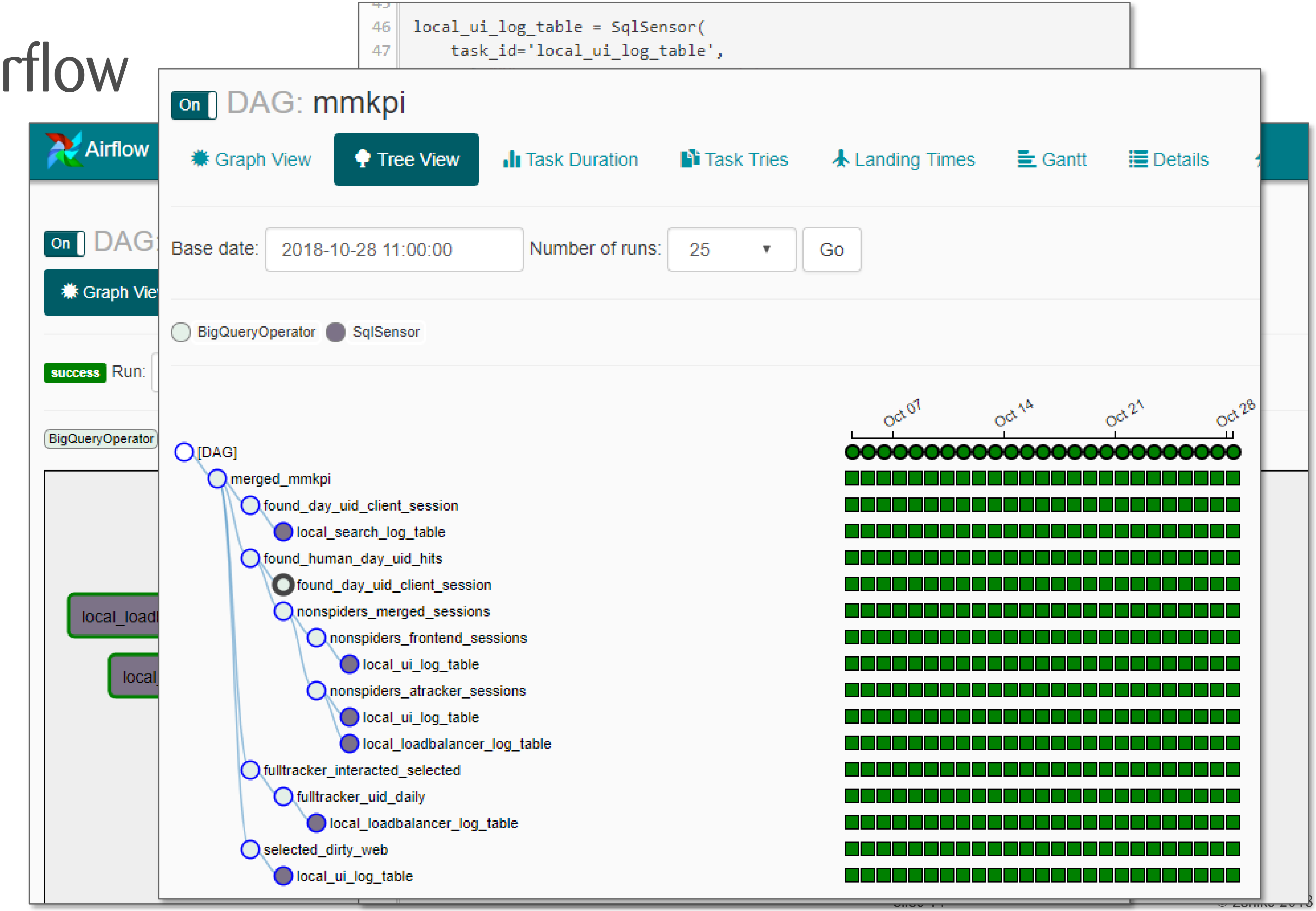


Apache Airflow

- Originally created at Airbnb, open sourced 2015
- Author, schedule and monitor workflows
- “Cron on steroids”
- Workflows are part of the codebase (Python)
- Workflows defined as DAGs of tasks
- Clear and transparent
- Easy to rerun or reproduce historical jobs by date →backfilling
- Thriving community

Apache Airflow

Example



```
local_ui_log_table = SqlSensor(
    task_id='local_ui_log_table',
```

On DAG: mmkpi

Base date: 2018-10-28 11:00:00 Number of runs: 25 ▼

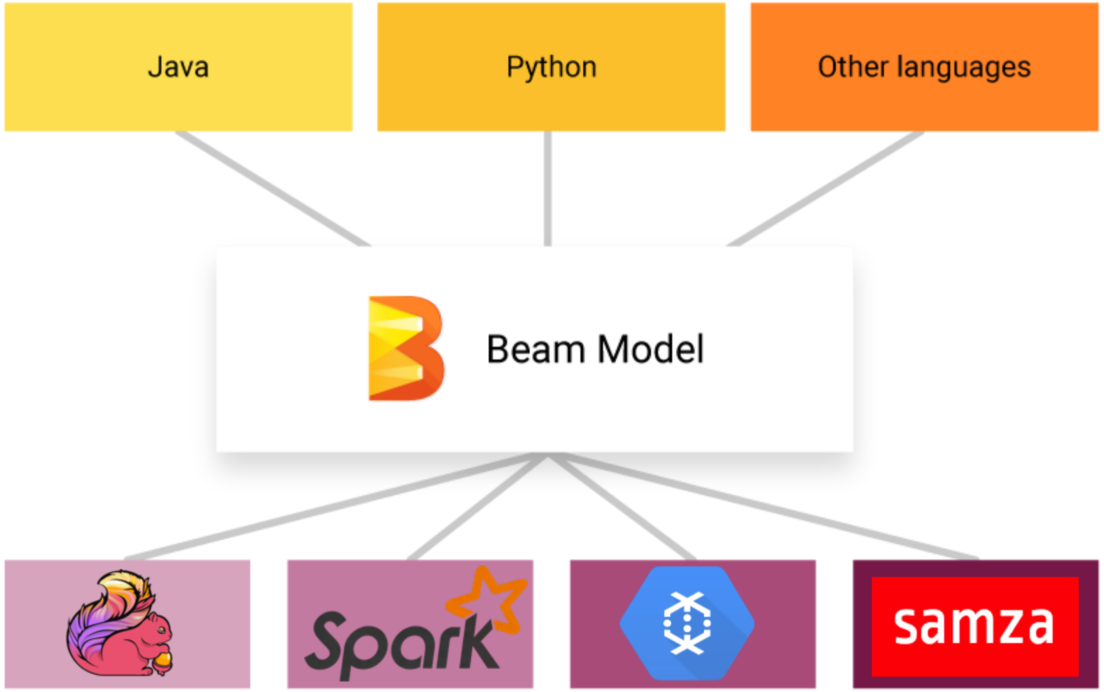
○ BigQueryOperator ● SqlSensor

```
○ [DAG]
  ○ merged_mmkpi
    ○ found_day_uid_client_session
      ● local_search_log_table
    ○ found_human_day_uid_hits
      ○ found_day_uid_client_session
      ○ nonspiders_merged_sessions
        ○ nonspiders_frontend_sessions
          ● local_ui_log_table
        ○ nonspiders_atracker_sessions
          ● local_ui_log_table
          ● local_loadbalancer_log_table
    ○ fulltracker_interacted_selected
      ○ fulltracker_uid_daily
        ● local_loadbalancer_log_table
    ○ selected_dirty_web
      ● local_ui_log_table
```

Apache Beam

- Unified **model** for batch and streaming
- Executes on a broad variety of runners (no vendor lock-in)
- Decouples data processing from the executor
- Comprehensive set of windowing, timing, lateness and triggering primitives

Choose your language...



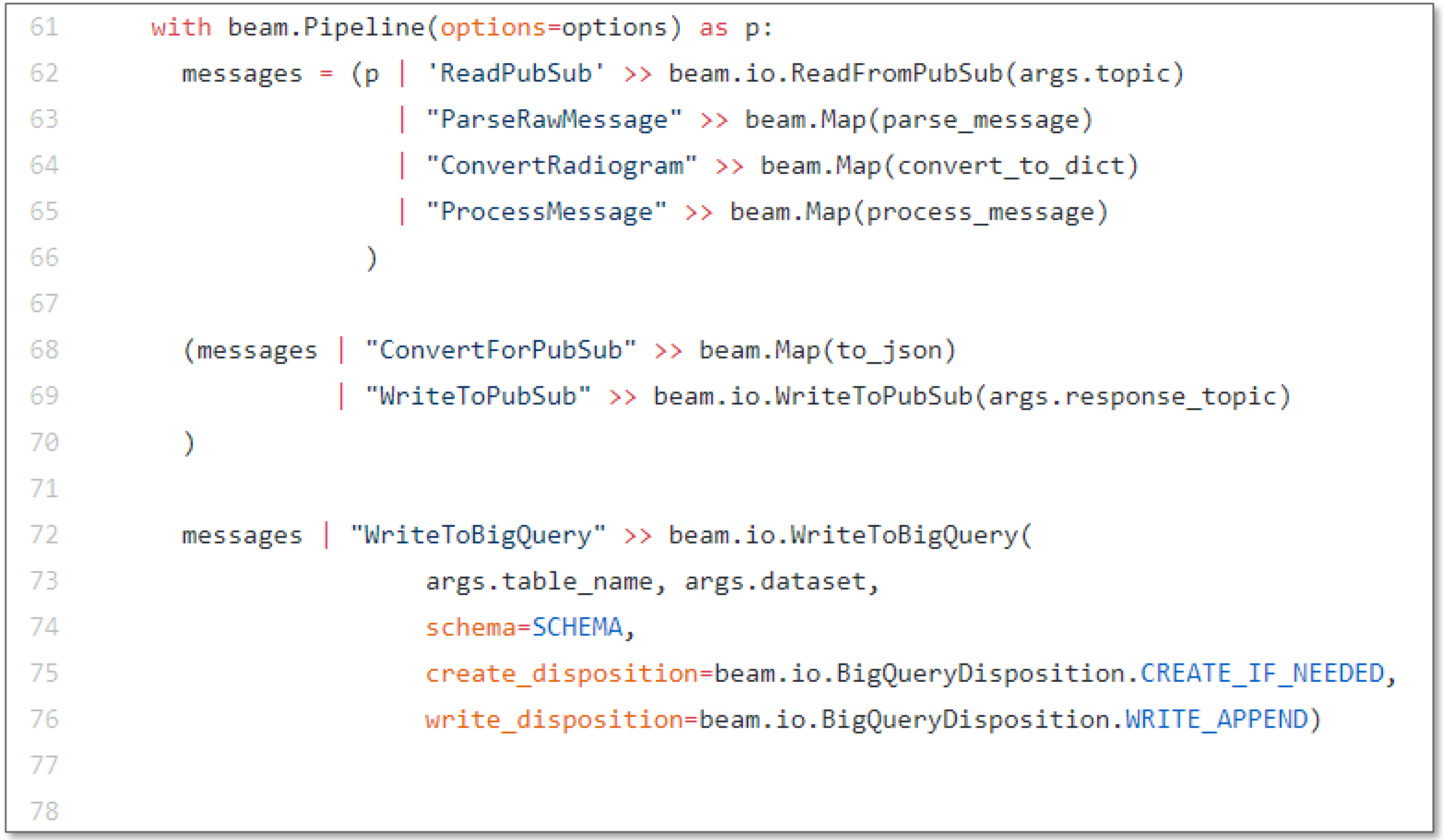
...and your runtime.

Source

- http://beam.apache.org

Apache Beam

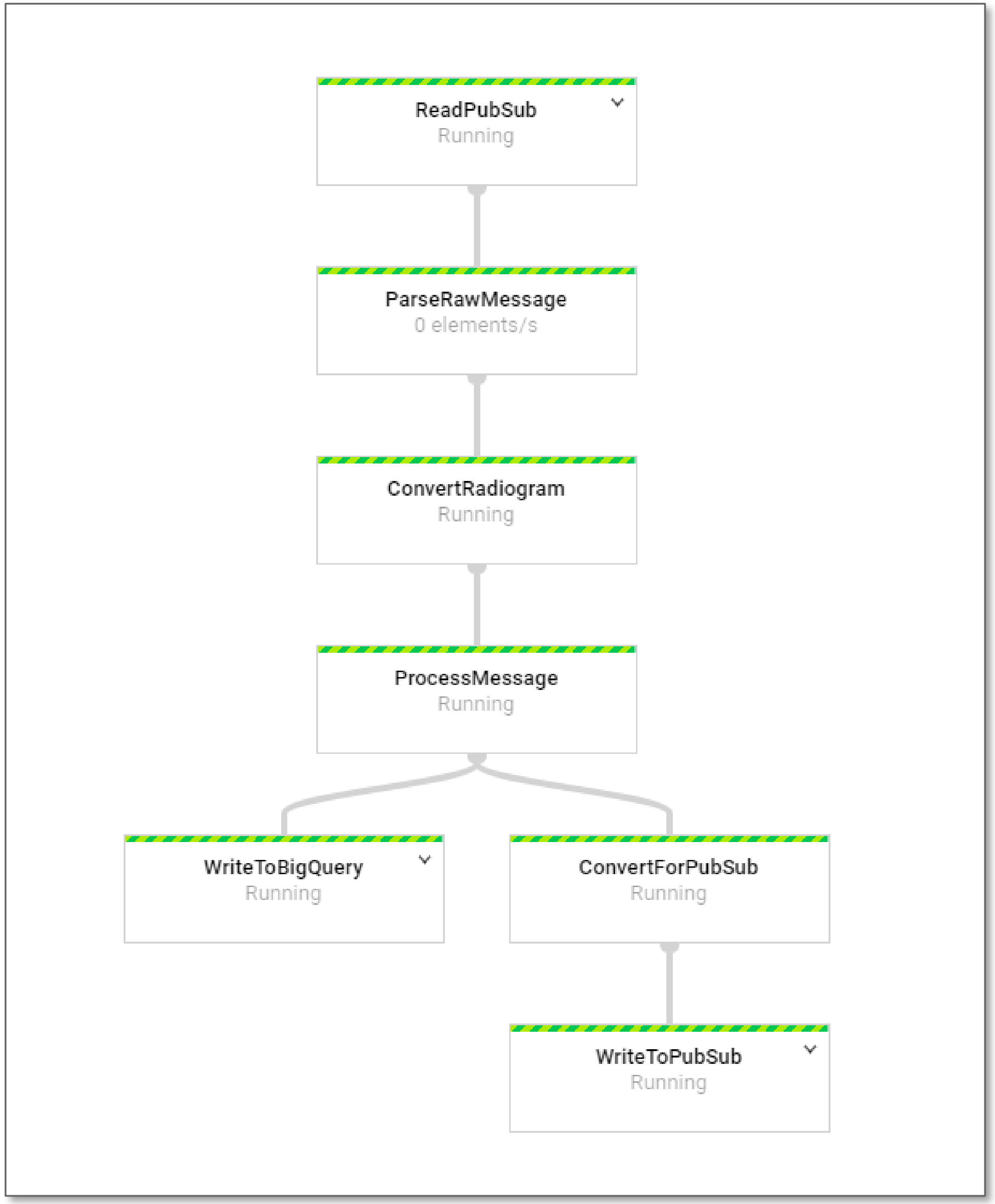
Python SDK



```python
    with beam.Pipeline(options=options) as p:
      messages = (p | 'ReadPubSub' >> beam.io.ReadFromPubSub(args.topic)
                    | "ParseRawMessage" >> beam.Map(parse_message)
                    | "ConvertRadiogram" >> beam.Map(convert_to_dict)
                    | "ProcessMessage" >> beam.Map(process_message)
                  )

      (messages | "ConvertForPubSub" >> beam.Map(to_json)
                | "WriteToPubSub" >> beam.io.WriteToPubSub(args.response_topic)
      )

      messages | "WriteToBigQuery" >> beam.io.WriteToBigQuery(
                      args.table_name, args.dataset,
                      schema=SCHEMA,
                      create_disposition=beam.io.BigQueryDisposition.CREATE_IF_NEEDED,
                      write_disposition=beam.io.BigQueryDisposition.WRITE_APPEND)


```

Apache Beam

Resulting pipeline

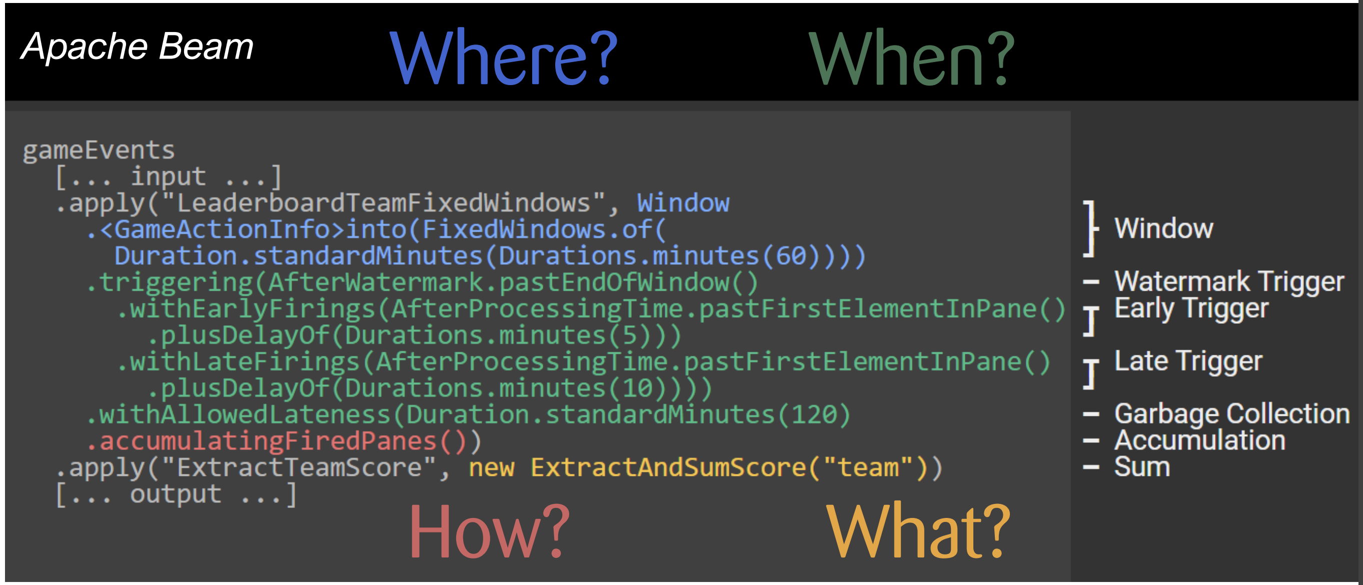


Apache Beam

Unified model for stream and batch processing

zühlke
empowering ideas

Example: Calculate Team scores by the hour

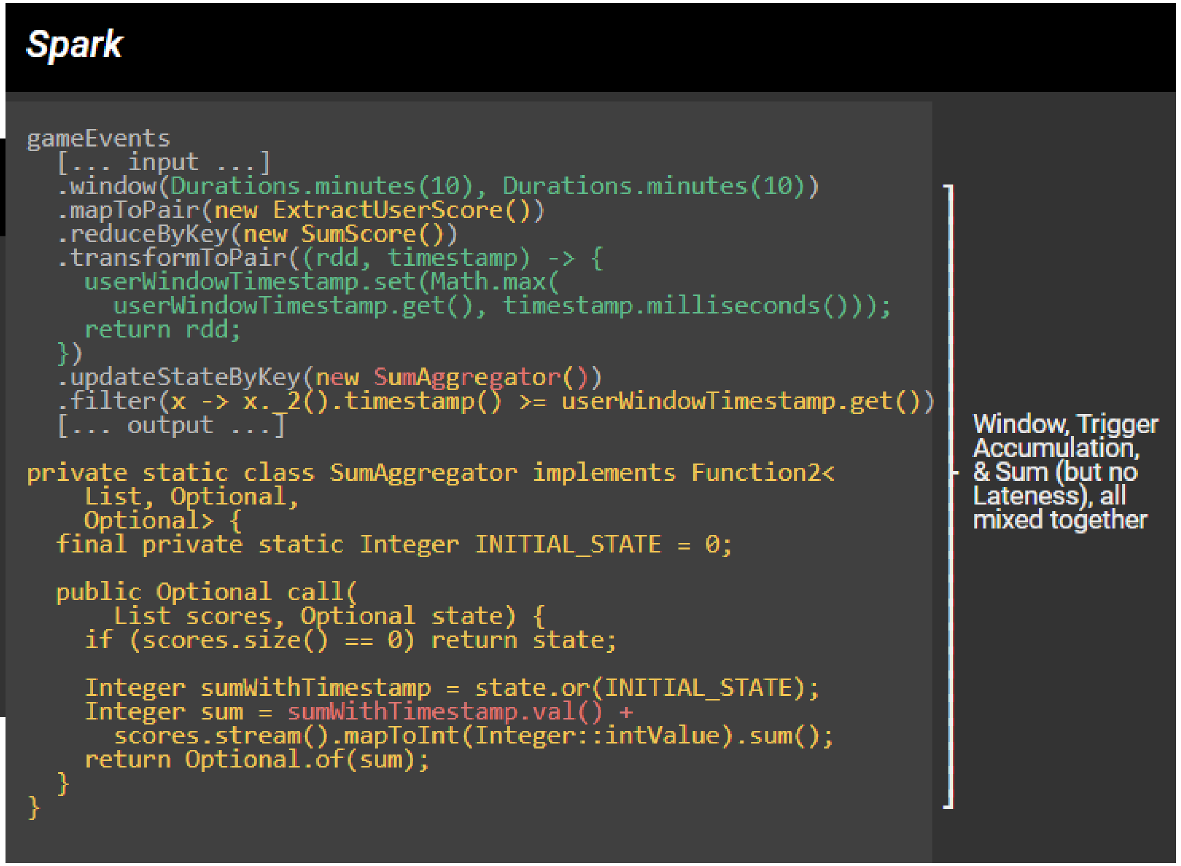


Apache Beam Where? When?

```
gameEvents
  [... input ...]
  .apply("LeaderboardTeamFixedWindows", Window
    .<GameActionInfo>into(FixedWindows.of(
      Duration.standardMinutes(Durations.minutes(60))))
    .triggering(AfterWatermark.pastEndOfWindow()
      .withEarlyFirings(AfterProcessingTime.pastFirstElementInPane()
        .plusDelayOf(Durations.minutes(5)))
      .withLateFirings(AfterProcessingTime.pastFirstElementInPane()
        .plusDelayOf(Durations.minutes(10))))
    .withAllowedLateness(Duration.standardMinutes(120)
    .accumulatingFiredPanes())
  .apply("ExtractTeamScore", new ExtractAndSumScore("team"))
  [... output ...]
```

How? What?

- Window
- Watermark Trigger
- Early Trigger
- Late Trigger
- Garbage Collection
- Accumulation
- Sum



Spark

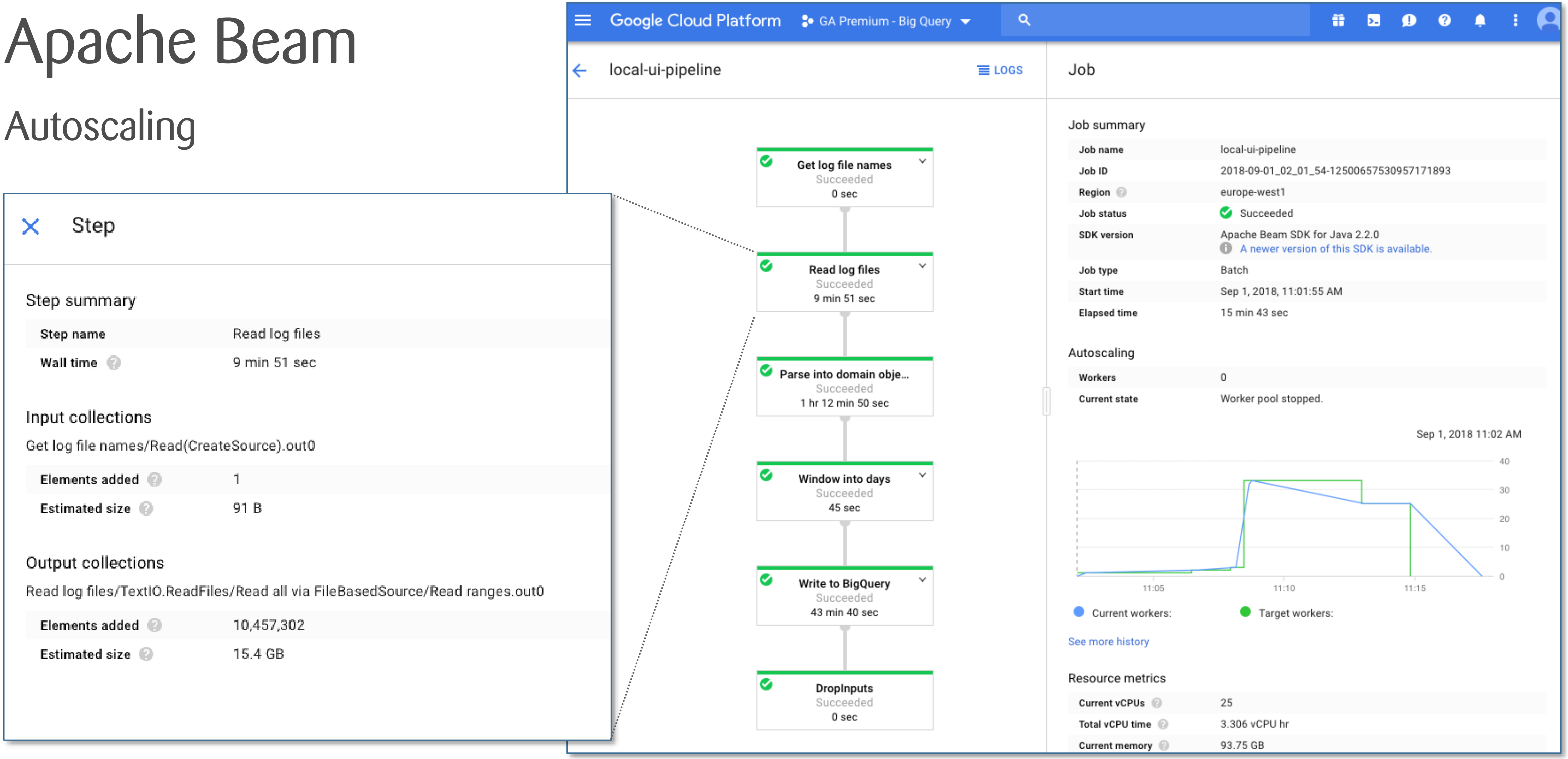
```
gameEvents
  [... input ...]
  .window(Durations.minutes(10), Durations.minutes(10))
  .mapToPair(new ExtractUserScore())
  .reduceByKey(new SumScore())
  .transformToPair((rdd, timestamp) -> {
    userWindowTimestamp.set(Math.max(
      userWindowTimestamp.get(), timestamp.milliseconds()));
    return rdd;
  })
  .updateStateByKey(new SumAggregator())
  .filter(x -> x._2().timestamp() >= userWindowTimestamp.get())
  [... output ...]

private static class SumAggregator implements Function2<
    List, Optional,
    Optional> {
  final private static Integer INITIAL_STATE = 0;

  public Optional call(
      List scores, Optional state) {
    if (scores.size() == 0) return state;

    Integer sumWithTimestamp = state.or(INITIAL_STATE);
    Integer sum = sumWithTimestamp.val() +
      scores.stream().mapToInt(Integer::intValue).sum();
    return Optional.of(sum);
  }
}
```

Window, Trigger Accumulation, & Sum (but no Lateness), all mixed together

Source:
https://cloud.google.com/dataflow/blog/dataflow-beam-and-spark-comparison

Apache Beam

Autoscaling



Google Cloud Platform GA Premium - Big Query

local-ui-pipeline LOGS

Get log file names
Succeeded
0 sec

Read log files
Succeeded
9 min 51 sec

Parse into domain obje...
Succeeded
1 hr 12 min 50 sec

Window into days
Succeeded
45 sec

Write to BigQuery
Succeeded
43 min 40 sec

DropInputs
Succeeded
0 sec

Job

Job summary

Job name	local-ui-pipeline
Job ID	2018-09-01_02_01_54-12500657530957171893
Region	europe-west1
Job status	✓ Succeeded
SDK version	Apache Beam SDK for Java 2.2.0 A newer version of this SDK is available.
Job type	Batch
Start time	Sep 1, 2018, 11:01:55 AM
Elapsed time	15 min 43 sec

Autoscaling

Workers	0
Current state	Worker pool stopped.

Sep 1, 2018 11:02 AM

See more history

Resource metrics

Current vCPUs	25
Total vCPU time	3.306 vCPU hr
Current memory	93.75 GB

Step

Step summary

Step name	Read log files
Wall time	9 min 51 sec

Input collections

Get log file names/Read(CreateSource).out0

Elements added	1
Estimated size	91 B

Output collections

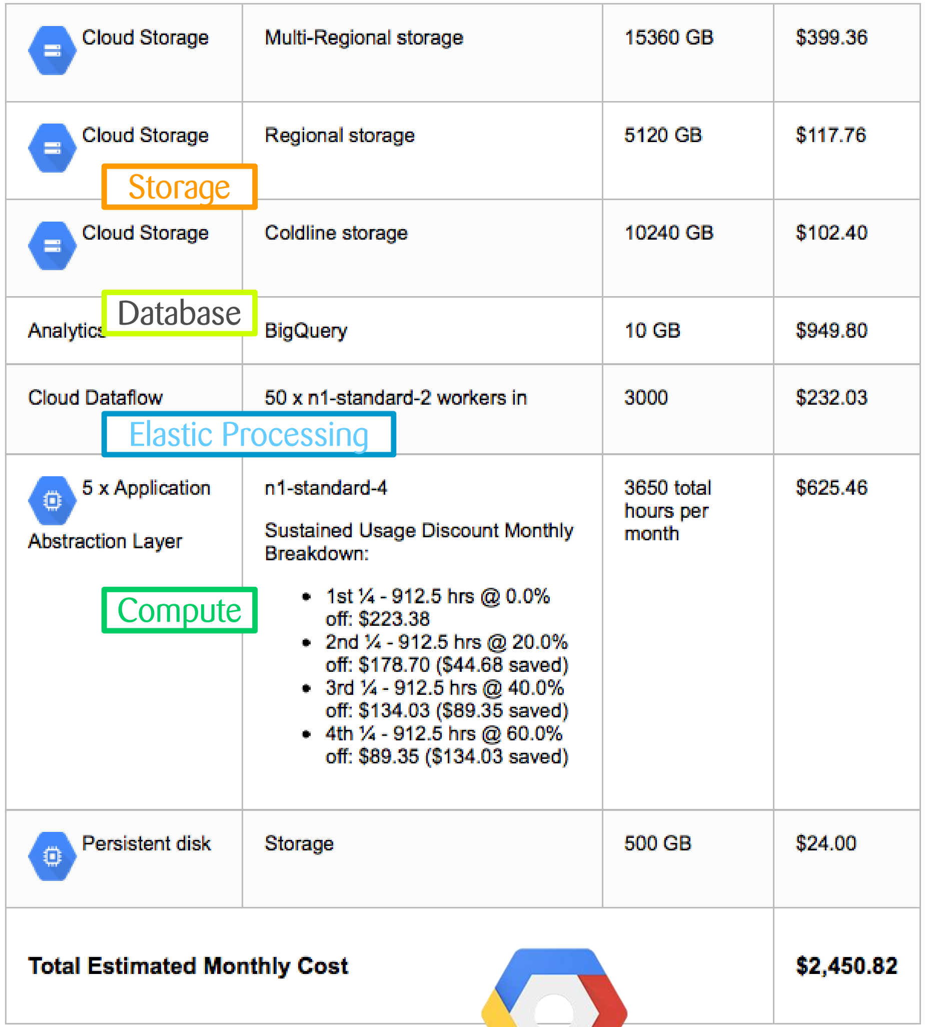
Read log files/TextIO.ReadFiles/Read all via FileBasedSource/Read ranges.out0

Elements added	10,457,302
Estimated size	15.4 GB

Cloud Vendor comparison

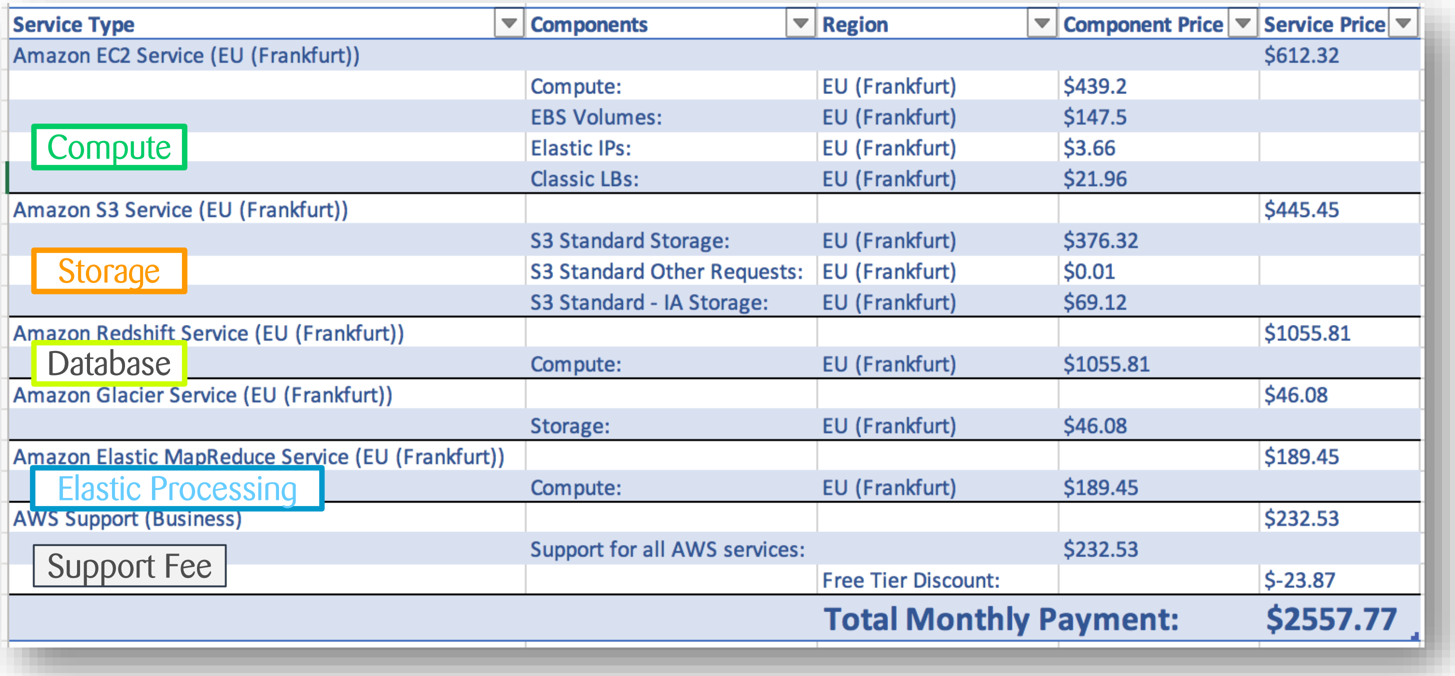
Data Platform Monthly Breakdown

zühlke
empowering ideas



Cloud Storage	Multi-Regional storage	15360 GB	$399.36
Cloud Storage Storage	Regional storage	5120 GB	$117.76
Cloud Storage	Coldline storage	10240 GB	$102.40
Analytics Database	BigQuery	10 GB	$949.80
Cloud Dataflow Elastic Processing	50 x n1-standard-2 workers in	3000	$232.03
5 x Application Abstraction Layer Compute	n1-standard-4 Sustained Usage Discount Monthly Breakdown: • 1st ¼ - 912.5 hrs @ 0.0% off: $223.38 • 2nd ¼ - 912.5 hrs @ 20.0% off: $178.70 ($44.68 saved) • 3rd ¼ - 912.5 hrs @ 40.0% off: $134.03 ($89.35 saved) • 4th ¼ - 912.5 hrs @ 60.0% off: $89.35 ($134.03 saved)	3650 total hours per month	$625.46
Persistent disk	Storage	500 GB	$24.00
Total Estimated Monthly Cost			**$2,450.82**

Google Cloud Platform



Service Type	Components	Region	Component Price	Service Price
Amazon EC2 Service (EU (Frankfurt))				$612.32
	Compute:	EU (Frankfurt)	$439.2	
	EBS Volumes:	EU (Frankfurt)	$147.5	
Compute	Elastic IPs:	EU (Frankfurt)	$3.66	
	Classic LBs:	EU (Frankfurt)	$21.96	
Amazon S3 Service (EU (Frankfurt))				$445.45
	S3 Standard Storage:	EU (Frankfurt)	$376.32	
Storage	S3 Standard Other Requests:	EU (Frankfurt)	$0.01	
	S3 Standard - IA Storage:	EU (Frankfurt)	$69.12	
Amazon Redshift Service (EU (Frankfurt))				$1055.81
Database	Compute:	EU (Frankfurt)	$1055.81	
Amazon Glacier Service (EU (Frankfurt))				$46.08
	Storage:	EU (Frankfurt)	$46.08	
Amazon Elastic MapReduce Service (EU (Frankfurt))				$189.45
Elastic Processing	Compute:	EU (Frankfurt)	$189.45	
AWS Support (Business)				$232.53
Support Fee	Support for all AWS services:		$232.53	
		Free Tier Discount:		$-23.87
		Total Monthly Payment:		**$2557.77**

amazon web services

Conclusion

- We created **reproducible** and **traceable** data pipelines using Apache Airflow and Apache Beam
- We established an essential building block to **incrementally** create **trustworthy** data-driven applications
- The tech stack relies on OSS and avoids a vendor lock-in (runs on own laptop, on-prem, AWS, GCP, …)
- Great collaboration with Google

Thank you