The Future of Data Engineering

Chris Riccomini / WePay / @criccomini / QCon SF / 2019-11-12

This talk

- Context
- Stages
- Architecture

Context

Me

- WePay, LinkedIn, PayPal
- Data infrastructure, data engineering, service infrastructure, data science
- Kafka, Airflow, BigQuery, Samza, Hadoop, Azkaban, Teradata

Me

- **WePay**, LinkedIn, PayPal
- **Data infrastructure, data engineering**, service infrastructure, data science
- **Airflow, BigQuery, Kafka**, Samza, Hadoop, Azkaban, Teradata

Me

- WePay, **LinkedIn**, PayPal
- **Data infrastructure, data engineering, service infrastructure, data science**
- Airflow, BigQuery, **Kafka, Samza, Hadoop, Azkaban**, Teradata

Me

- WePay, LinkedIn, **PayPal**
- Data infrastructure, data engineering, service infrastructure, **data science**
- Airflow, BigQuery, Kafka, Samza, Hadoop, Azkaban, **Teradata**

Data engineering?

A data engineer’s job is to help an organization **move** and **process** data

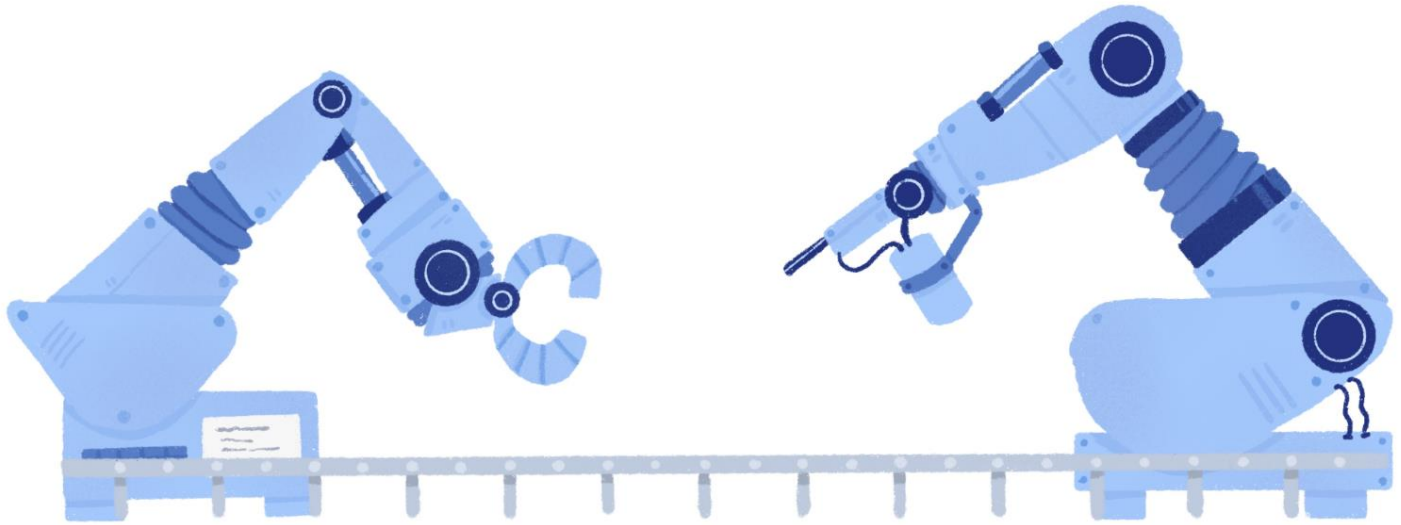
“…data engineers build tools, infrastructure, frameworks, and services.”

-- Maxime Beauchemin, The Rise of the Data Engineer

Why?

How we built analytics at Ada with Airflow and Redshift

Polina Slavskaya Follow
May 28 · 11 min read



At the beginning of 2018, when I was packing my bags to move from Moscow to Toronto, my dear colleagues, whom I didn't know yet, were receiving an increasing amount of requests for insights into how Ada's virtual assistant was doing. How many conversations did the virtual assistant start and received a response from the user? How many conversations happened on a mobile device? What are the most popular responses virtual assistant gives to French-speaking customers?

Both the Development and Customer Success teams rushed into answering those questions. Ada was barely a dozen people by that time, and all of them did their best to give clients their insights. But we had a huge feature

Chris Riccomini

Principal Software Engineer

Building WePay's data warehouse using BigQuery and Airflow

By Chris Riccomini on Jul 5, 2016

Over the coming weeks, we'll be writing a series of posts describing how we've built and run WePay's data warehouse. This series will cover our usage of Google Cloud Platform, BigQuery, and Apache Airflow (incubating), as well as how we handle security, data quality checks, and our plans for the future.

The beginning

We run MySQL as our primary OLTP database at WePay. Most of the data resides in a single monolithic database cluster that runs inside Google compute engine. When we started out, developers, analysts, product managers, and others were all running their analytic queries on a replica of this MySQL cluster. This replica was used to do things like train machine learning models, generate business and financial reports, build business intelligence dashboards, debug production issues, and so on.

This approach had the advantages of being convenient and having up-to-date data (since it was just a replica of the database). The disadvantage of this setup is that MySQL is not very good at handling large analytic-style queries. This is not surprising. It's meant to be used as a low latency OLTP database, not a data warehouse. As we (and our data) grew, we bumped up against some problems:

- Multi-tenancy issues, where one user would severely degrade the entire cluster.
- Performance problems, timeouts, and queries that never finish.
- Inability to easily run custom logic on the data (UDFs).

It was obvious that we needed to move to a real data warehouse. This post describes that journey, the lessons learned, and the technology that we used.

Picking the stack

WePay made the transition to Google Cloud Platform last year. One of the services that Google provides is BigQuery, a distributed data warehouse built on top of a bunch of great Google technology including Dremel, Borg, Colossus, and Jupiter. We did an evaluation of BigQuery, and decided to use it as our data warehousing solution.

After picking BigQuery, we needed to get data into it. A naive approach would be to run a Python script via CRON that wakes up periodically, selects rows from MySQL, and inserts them into BigQuery. We did this initially as a quick short-term solution. This approach has a number of problems, though:

- What happens if things fail? Should you retry?
- How is monitoring handled? Will anyone know if things failed?
- What if things need to happen in sequence, or depend on each other?
- What if the script runs slowly, and a second iteration of the script starts before the first finishes?

Most of these problems (and many others) are solved by workflow schedulers, so we opted to use one to run our ETL scripts. The four that we looked at were Oozie, Azkaban, Luigi, and Airflow.

Chris Riccomini on Twitter: "Th

twitter.com/criccomini/status/1149549248326692866

Incognito

Home Moments Search Twitter Have an account? Log in

Chris Ricco

@criccomini

Software engineer
PayPal. Co-autho
at CamelCase cor

Sunnyvale, CA

riccomini.nam

Joined April 20

Chris Riccomini
@criccomini

Follow

This post is almost verbatim the same path we went through in 2015. Next two iterations of this are to migrate to a real-time pipeline, and then build a completely self-serve (automated) pipeline.

How we built analytics at Ada with Airflow and Redshift

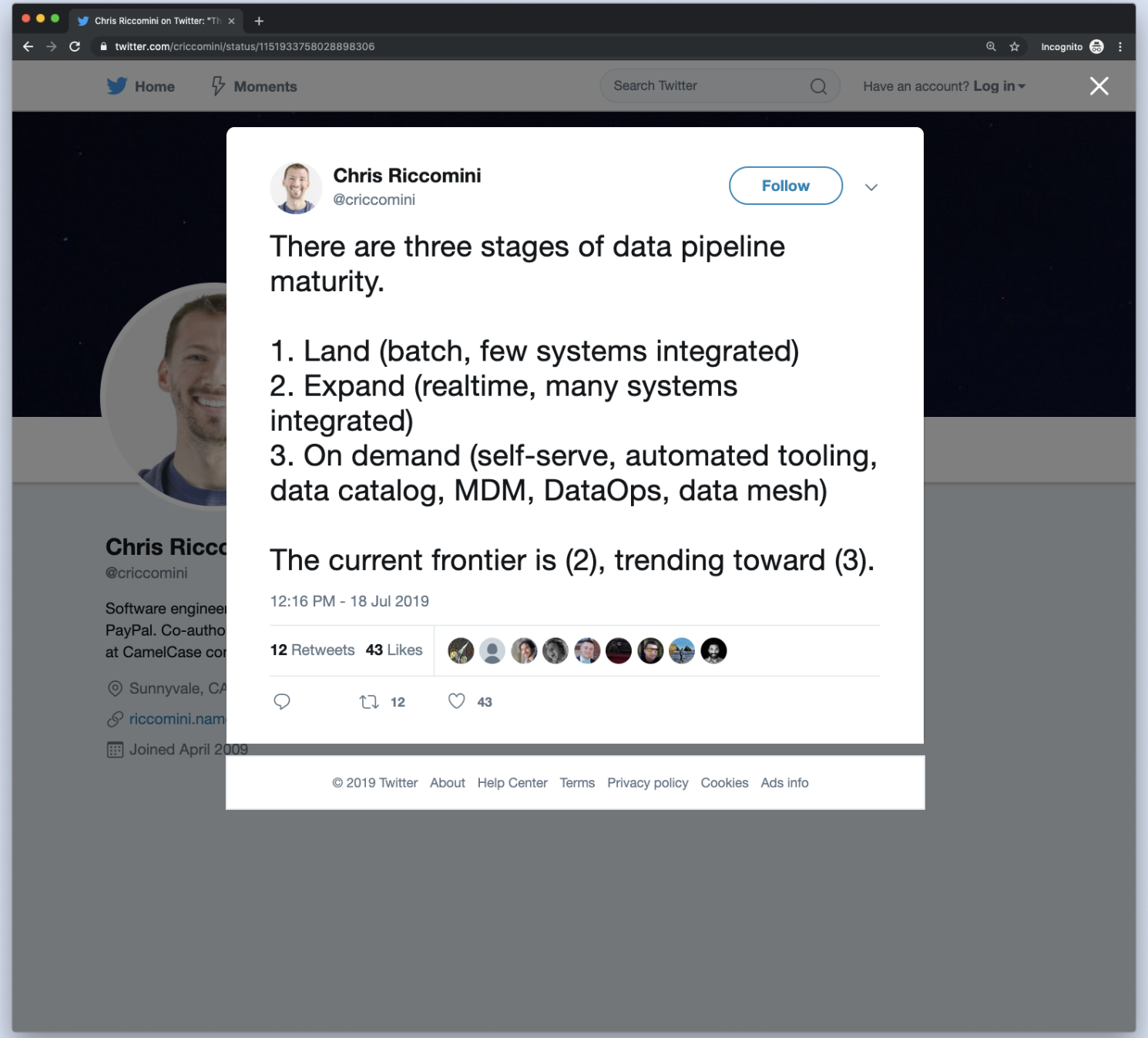
In this article I will try to succinctly describe how my team and I have build first version of data warehouse and analytics in Ada. There…

medium.com

10:21 PM - 11 Jul 2019

1 Retweet **4** Likes

1 4


Chris Riccomini on Twitter: "Th
twitter.com/criccomini/status/1151933758028898306
Incognito
Home
Moments
Search Twitter
Have an account? Log in
Chris Riccomini
@criccomini
Follow
There are three stages of data pipeline maturity.
1. Land (batch, few systems integrated)
2. Expand (realtime, many systems integrated)
3. On demand (self-serve, automated tooling, data catalog, MDM, DataOps, data mesh)
The current frontier is (2), trending toward (3).
12:16 PM - 18 Jul 2019
12 Retweets 43 Likes
12
43
Chris Ricco
@criccomini
Software engineer
PayPal. Co-autho
at CamelCase cor
Sunnyvale, CA
riccomini.nam
Joined April 2009
© 2019 Twitter About Help Center Terms Privacy policy Cookies Ads info

The Future of Data Engineering

riccomini.name/future-data-engineering

Incognito

Chris Riccomini

Software engineering stuff

Share

Chris Follow

Software engineer. WePay, LinkedIn, PayPal. Co-author Apache Samza, PMC Apache Airflow.

The Future of Data Engineering

I have been thinking lately about where we've come in data engineering over the past few years, and about what the future holds for work in this area. Most of this thought has been framed in the context of what some of our teams are doing at WePay, but I believe the framework below applies more broadly, and is worth sharing.

Data engineering's job is to help an organization move and process data. This generally requires two different systems, broadly speaking: a data pipeline, and a data warehouse. The data pipeline is responsible for moving the data, and the data

Six stages of data pipeline maturity

- Stage 0: None
- Stage 1: Batch
- Stage 2: Realtime
- Stage 3: Integration
- Stage 4: Automation
- Stage 5: Decentralization

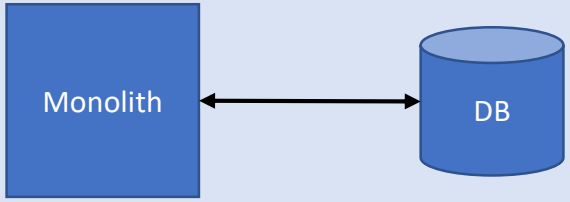
Six stages of data pipeline maturity

- **Stage 0: None**
- Stage 1: Batch
- Stage 2: Realtime
- Stage 3: Integration
- Stage 4: Automation
- Stage 5: Decentralization

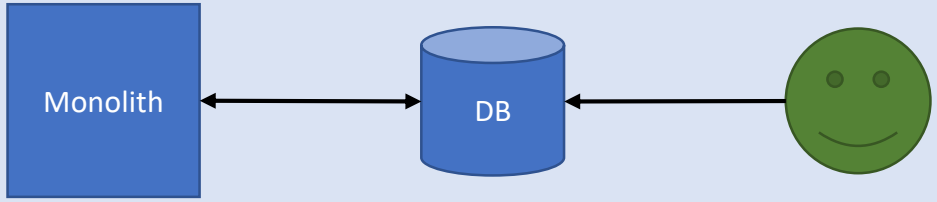
You might be ready for a data warehouse if…

- You have no data warehouse
- You have a monolithic architecture
- You need a data warehouse up and running ***yesterday***
- Data engineering isn’t your full time job

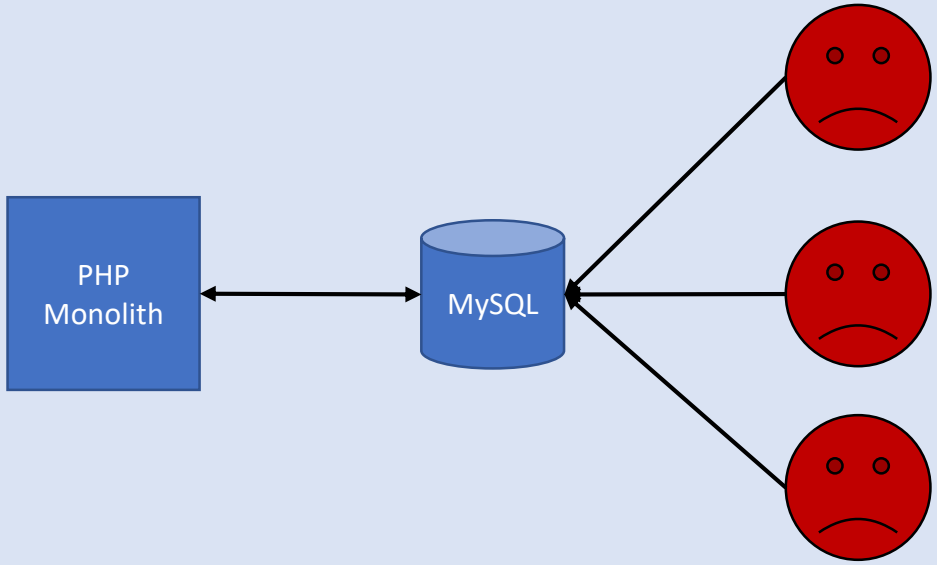
Stage 0: None



Stage 0: None



WePay circa 2014



Problems

- Queries began timing out
- Users were impacting each other
- MySQL was missing complex analytical SQL functions
- Report generation was breaking

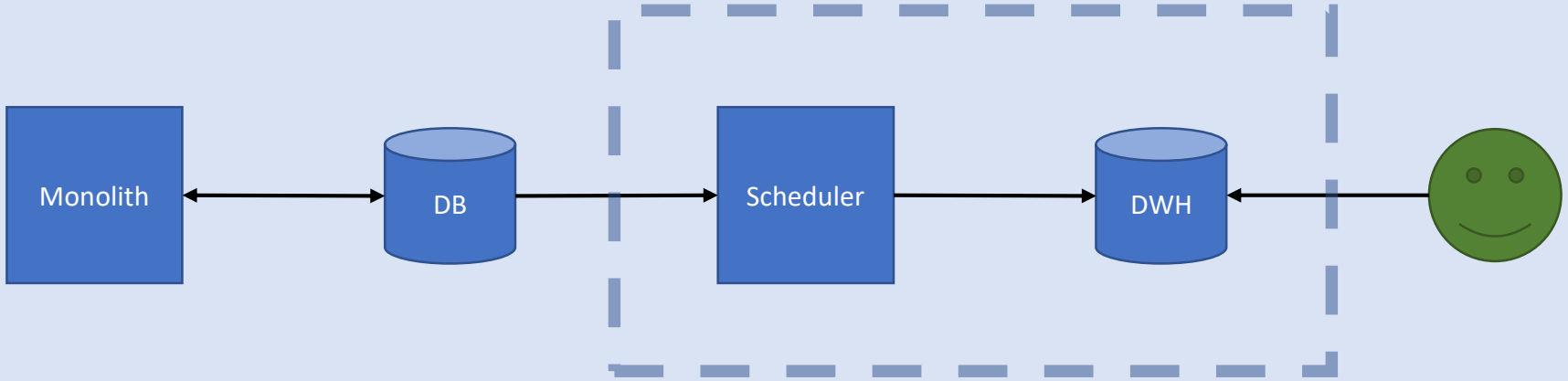
Six stages of data pipeline maturity

- Stage 0: None
- **Stage 1: Batch**
- Stage 2: Realtime
- Stage 3: Integration
- Stage 4: Automation
- Stage 5: Decentralization

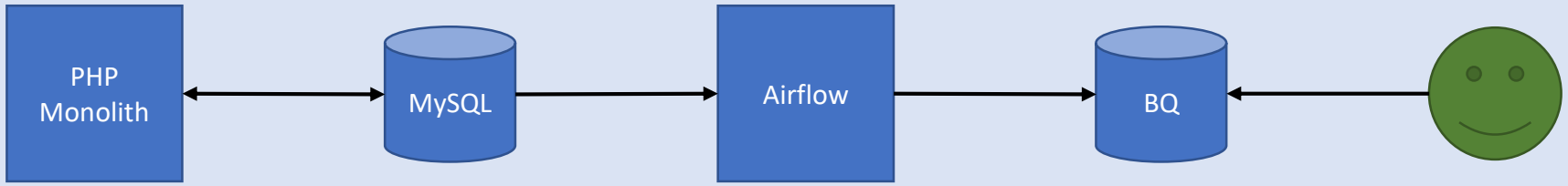
You might be ready for batch if…

- You have a monolithic architecture
- Data engineering is your part-time job
- Queries are timing out
- Exceeding DB capacity
- Need complex analytical SQL functions
- Need reports, charts, and business intelligence

Stage 1: Batch



WePay circa 2016



Problems

- Large number of Airflow jobs for loading all tables
- Missing and inaccurate create_time and modify_time
- DBA operations impacting pipeline
- Hard deletes weren’t propagating
- MySQL replication latency was causing data quality issues
- Periodic loads cause occasional MySQL timeouts

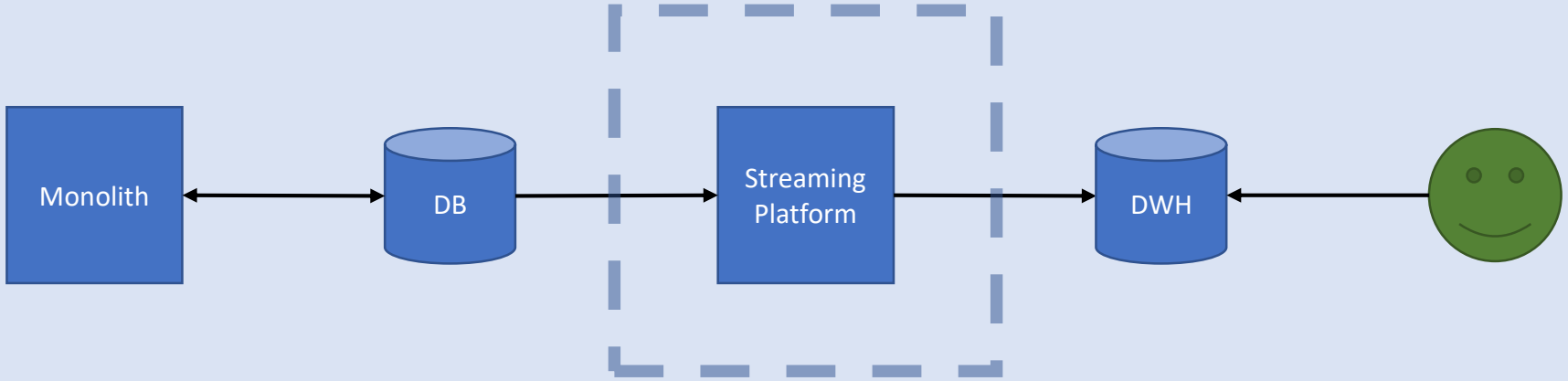
Six stages of data pipeline maturity

- Stage 0: None
- Stage 1: Batch
- **Stage 2: Realtime**
- Stage 3: Integration
- Stage 4: Automation
- Stage 5: Decentralization

You might be ready for realtime if…

- Loads are taking too long
- Pipeline is no longer stable
- Many complicated workflows
- Data latency is becoming an issue
- Data engineering is your fulltime job
- You already have Apache Kafka in your organization

Stage 2: Realtime



Chris Riccomini

Principal Software Engineer

Streaming databases in realtime with MySQL, Debezium, and Kafka

By Chris Riccomini on Feb 21, 2017

Change data capture has been around for a while, but some recent developments in technology have given it new life. Notably, using Kafka as a backbone to stream your database data in realtime has become increasingly common.

If you're wondering why you might want to stream database changes into Kafka, I highly suggest reading The Hardest Part About Microservices: Your Data. At WePay, we wanted to integrate our microservices and downstream datastores with each other, so every system could get access to the data that it needed. We use Kafka as our data integration layer, so we needed a way to get our database data into it.

Last year, Yelp's engineering team published an excellent series of posts on their data pipeline. These included a discussion on how they stream MySQL data into Kafka. Their architecture involves a series of homegrown pieces of software to accomplish the task, notably schematizer and MySQL streamer. The write-up triggered a thoughtful post on Debezium's blog about a proposed equivalent architecture using Kafka connect, Debezium, and Confluent's schema registry. This proposed architecture is what we've been implementing at WePay, and this post describes how we leverage Debezium and Kafka connect to stream our MySQL databases into Kafka.

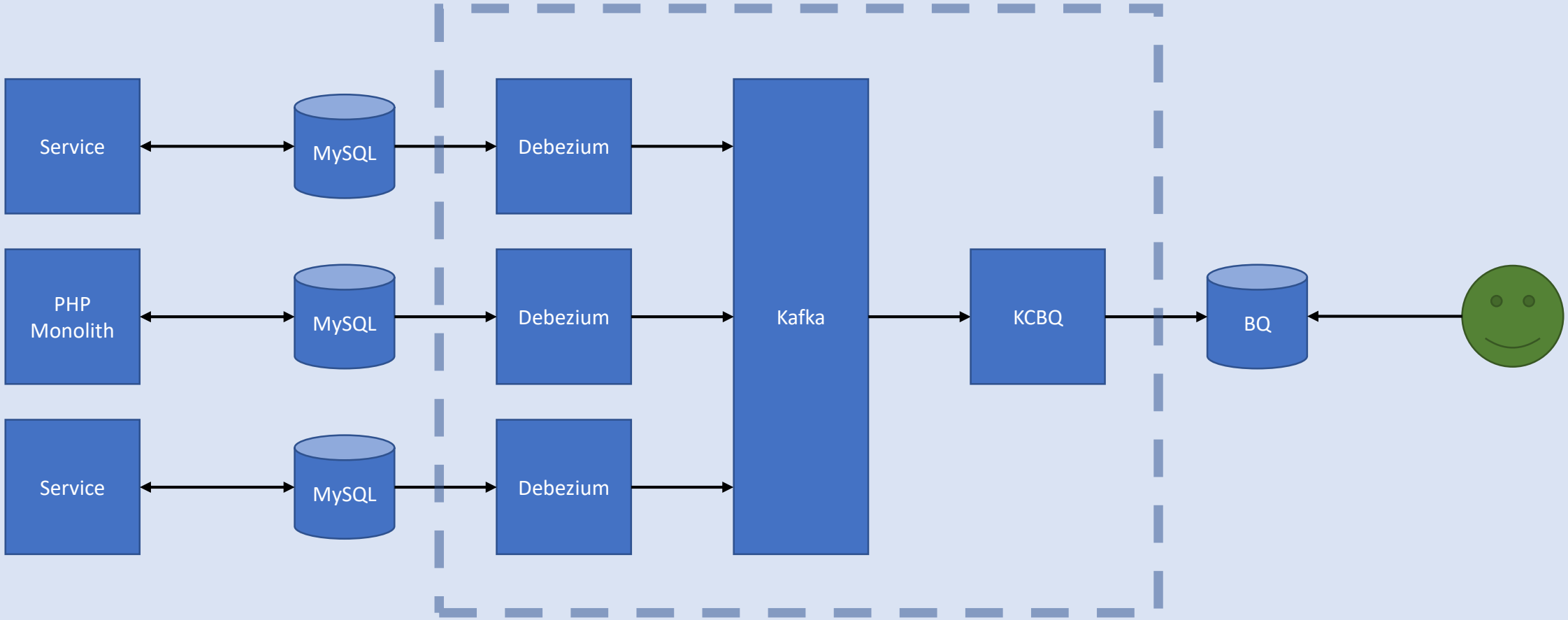
Architecture

The flow of data starts with each microservice's MySQL database. These databases run in Google Cloud as CloudSQL MySQL instances with GTIDs enabled. We've set up a downstream MySQL cluster specifically for Debezium. Each CloudSQL instance replicates its data into the Debezium cluster, which consists of two MySQL machines: a primary (active) server and secondary (passive) server. This single Debezium cluster is an operational trick to make it easier for us to operate Debezium. Rather than having Debezium connect to dozens of microservice databases directly, we can connect to just a single database. This also isolates Debezium from impacting the production OLTP workload that the master CloudSQL instances are handling.

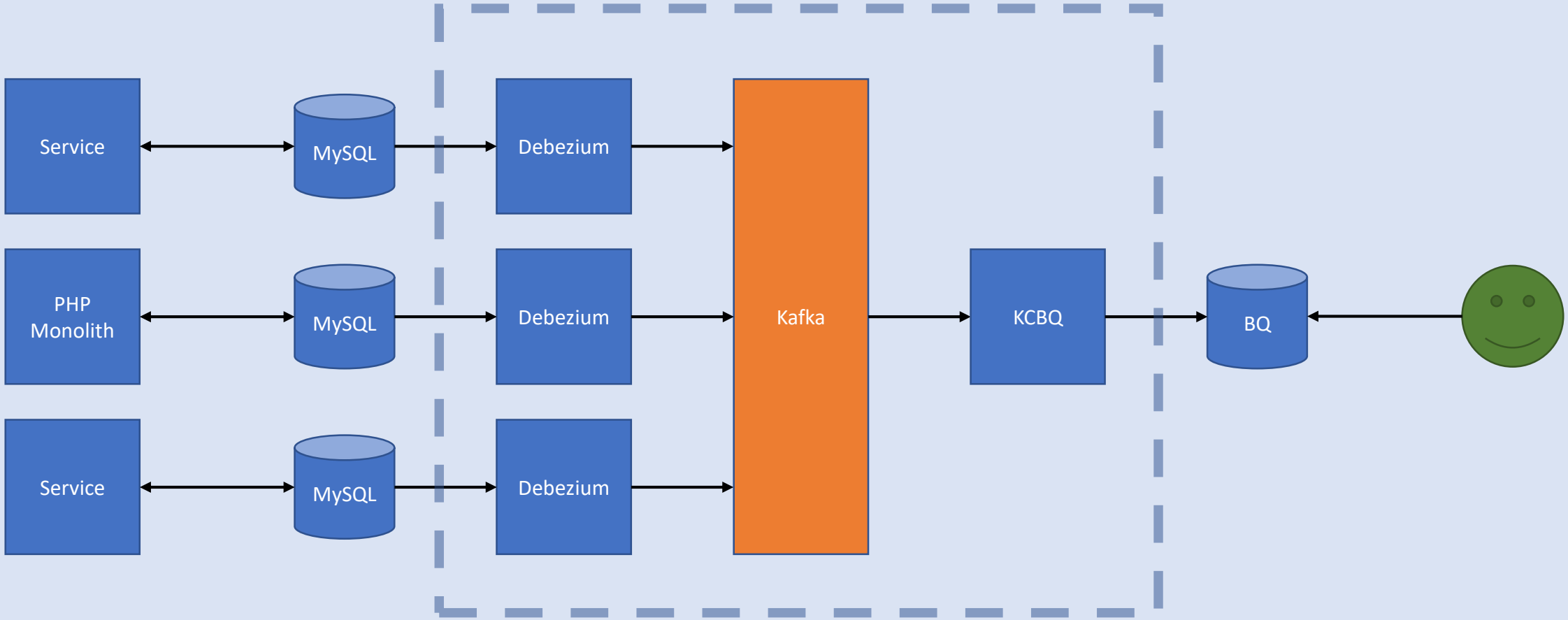
We run one Debezium connector (in distributed mode on the Kafka connect framework) for each microservice database. Again, the goal here is isolation. Theoretically, we could run a single Debezium connector that produces messages for all databases (since all microservice databases are in the Debezium cluster). This approach would actually be more resource efficient since each Debezium connector has to read MySQL's entire binlog anyway. We opted not to do this because we wanted to be able to bring Debezium connectors up and down, and configure them differently for each microservice DB.

The Debezium connectors feed the MySQL messages into Kafka (and add their schemas to the Confluent schema registry), where downstream systems can consume them. We use our Kafka connect BigQuery connector to load the MySQL data into BigQuery using BigQuery's streaming API. This gives us a data warehouse in BigQuery that is usually less than 30 seconds behind the data that's in production. Other microservices, stream processors, and data infrastructure consume the feeds as well.

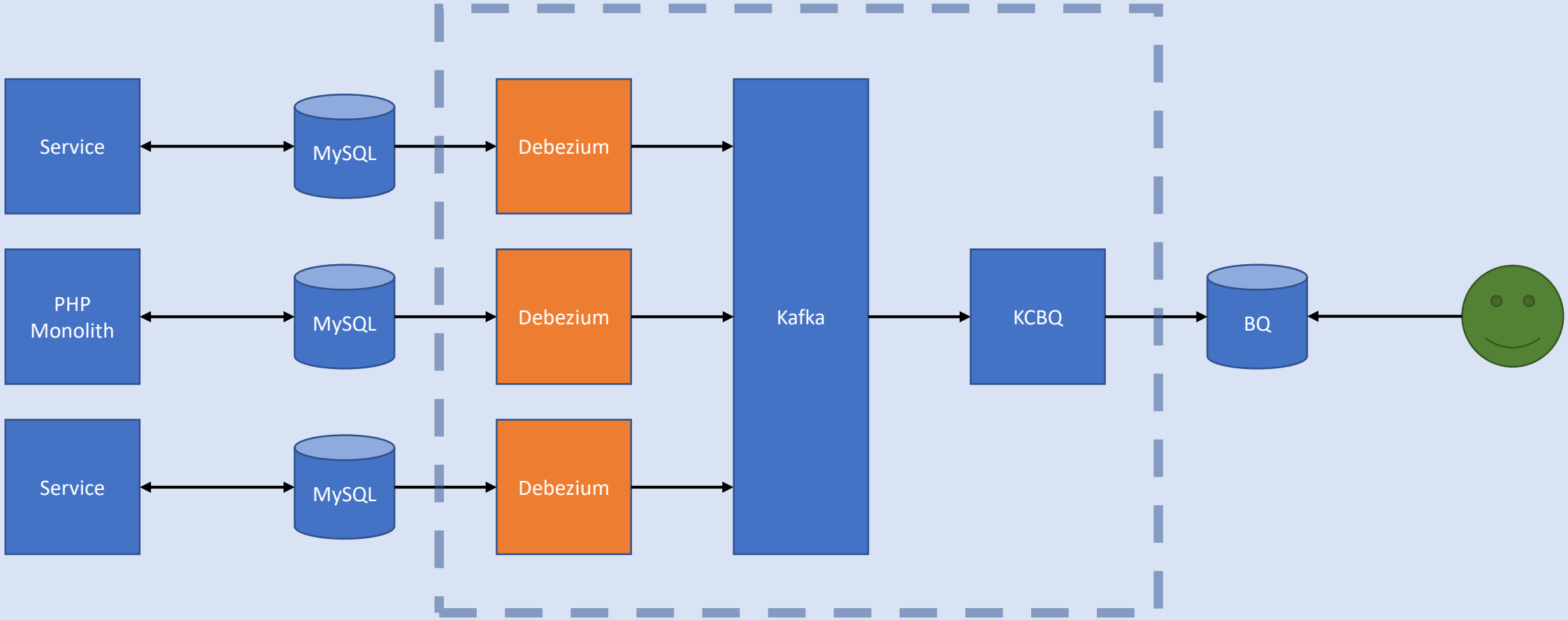
WePay circa 2017



WePay circa 2017



WePay circa 2017



Change data capture?

…an approach to data integration that is based on the identification, capture and delivery of the changes made to enterprise data sources.

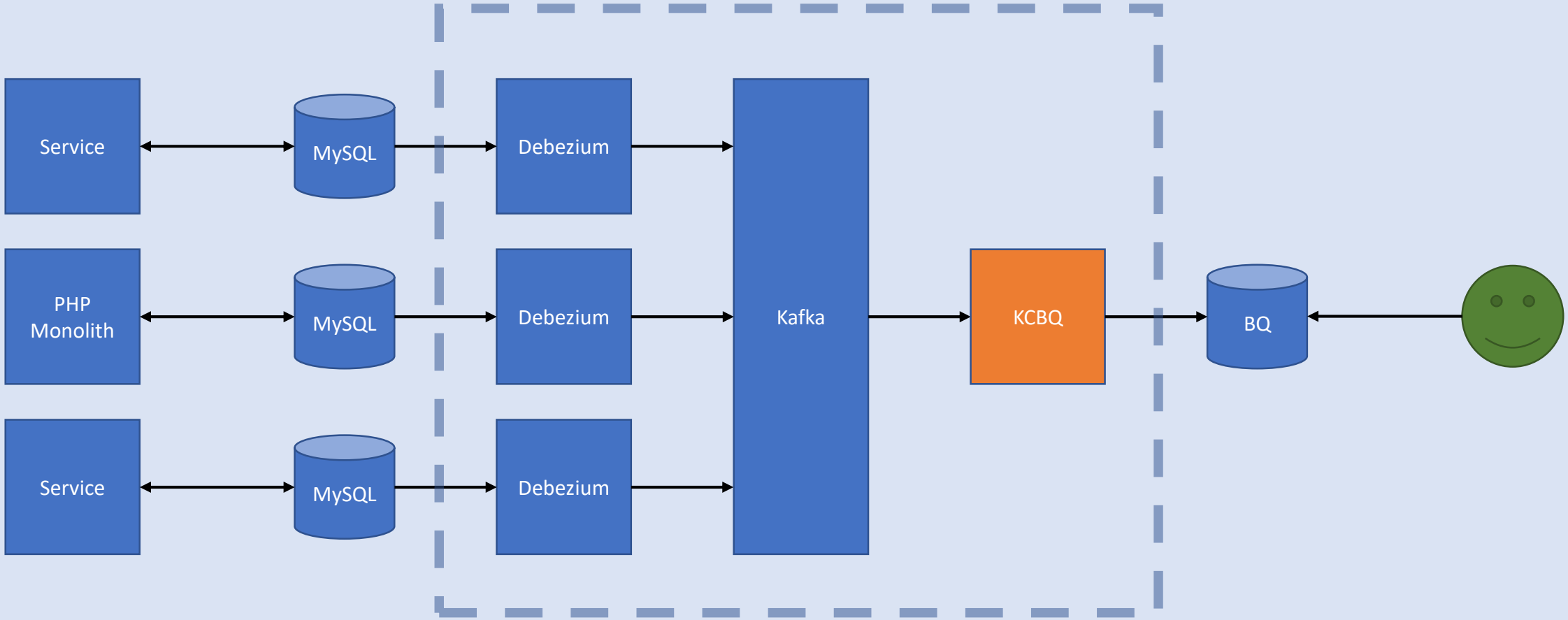
https://en.wikipedia.org/wiki/Change_data_capture

Debezium sources

- MongoDB
- MySQL
- PostgreSQL
- SQL Server
- Oracle (Incubating)
- Cassandra (Incubating)

7:00AM	**WOMEN & ALLIES IN TECH COMMUNITY BREAKFAST CO-SPONSORED BY NETFLIX - REGISTRATION REQUIRED**							
9:00AM	**It Really is a Series of Tubes** Molly Wright Steenson - AI, Ethics & Design: Author, Designer, Professor, Research Leader							
10:10AM	**BREAK**							
	Microservices Patterns & Practices Host: Colin Breck	**Modern Data Architectures** Host: Gwen Shapira	**Modern CS in the Real World** Host: Werner Schuster	**Practices of DevOps & Lean Thinking** Host: John Willis	**Pushing the Web Forward: JavaScript, Frameworks, Transpilers, and WebAssembly** Host: Dylan Schiemann Host: Phil Haack	**Socially Conscious Software** Host: Alex Qin	**SPONSORED SOLUTIONS TRACK III** Host: Nitin Bharti	**SPONSORED SOLUTIONS TRACK IV** Host: Nitin Bharti
10:35AM	**Managing Failure Modes in Microservice Architectures** Adrian Cockcroft - Amazon Web Services Ballroom A	**Future of Data Engineering** Chris Riccomini - WePay Ballroom BC	**Anti-Entropy Using CRDTs on HA Datastores @Netflix** Sailesh Mukil - Netflix Pacific DEKJ	**DevOps & Lean Thinking Panel** Bayview AB	**MakeCode: Types, Games and Machine Code** Michal Moskal - Microsoft Pacific LMNO	**How Do We Heal?** Alex Qin - the code cooperative Seacliff ABC	**Real-time Log Analytics at Scale with Kafka** Kresten Thorup - Humio Pacific BC	**Java on Kubernetes 101 Hands-On Lab!** Pratik Patel - IBM & William Korando - IBM Marina
11:25AM	**BREAK**							
11:50AM	**User & Device Identity for Microservices @ Netflix Scale** Satyajit Thadeshwar - Netflix Ballroom A	**Data Mesh Paradigm Shift in Data Platform Architecture** Zhamak Dehghani - ThoughtWorks Ballroom BC	**The Talk You've Been Await-Ing For** Steve Klabnik - Rust Core Team Pacific DEKJ	**Mapping the Evolution of Socio-Technical Systems** Cat Swetel - Ticketmaster Bayview AB	**Javascript Open Space** Facilitated Peer Sharing Seacliff D	**Holistic EdTech & Diversity** Antoine Patton - Unlock.Academy Seacliff ABC	**The Next Era in Engineering Velocity** Bryan Helmkamp - Code Climate Pacific BC	**Session Continues**
12:40PM	**LUNCH**							
1:40PM	**Stateful Programming Models in Serverless Functions** Chris Gillum - Microsoft Ballroom A	**Taming Large State: Lessons From Building Stream Processing** Sonali Sharma - Netflix & Shriya Arora - Netflix Ballroom BC	**Swift 4 TF** Paige Bailey - Google Pacific DEKJ	**DevOps & Lean Thinking Open Space** Facilitated Peer Sharing Seacliff D	**Goodbye View Source: Hello JavaScript in the Age of Compilers** Amal Hussein - npm Pacific LMNO	**Socially Conscious Software** Jabari Bell - giya.us Seacliff ABC	**The Case for Human-Centric Observability** Austin Parker - LightStep Pacific BC	**Java on Kubernetes 101 Hands-On Lab!** Pratik Patel - IBM & William Korando - IBM Marina
2:30PM	**BREAK**							
2:55PM	**Controlled Chaos: Taming Organic, Federated Growth of Microservices** Tobias Kunze - Glasnostic Ballroom A	**Modern Data Architectures Open Space** Facilitated Peer Sharing Seacliff D	**ML/AI Panel** Amy Unruh - Google & Chris Albon - Devoted Health & June Andrews - Stitch Fix & Paige Bailey - Google Pacific DEKJ	**The System of Profound Knowledge** Ben Rockwood - Packet Bayview AB	**Build Your Own WebAssembly Compiler** Colin Eberhardt - Scott Logic Pacific LMNO	**Socially Conscious Software** Tiffani Ashley Bell Seacliff ABC	**My Code Is Broken but So Is the Debugging Experience** Matt Robenolt - Sentry Pacific BC	**Session Continues**
3:45PM	**BREAK**							
4:10PM	**Beyond Microservices: Streams, State and Scalability** Gwen Shapira - Confluent Ballroom A	**Kafka Needs No Keeper** Colin McCabe - Confluent Ballroom BC	**Sorbet: Why and How We Built a Typechecker for Ruby** Dmitry Petrashko - Stripe Pacific DEKJ	**Incident Management in the Age of DevOps & SRE** Damon Edwards - Rundeck Bayview AB	**Make Your Electron App Feel at Home Everywhere** Kilian Valkhof - Polypane Pacific LMNO	**Socially Conscious Software Open Space** Facilitated Peer Sharing Seacliff D	**Azul** Gil Tene - Azul Pacific BC	**Java on Kubernetes 101 Hands-On Lab!** Pratik Patel - IBM & William Korando - IBM Marina
5:00PM	**BREAK**							
5:25PM	**Microservices Open Space** Facilitated Peer Sharing Seacliff D	**Practical Change Data Streaming Use Cases With Apache Kafka & Debezium** Gunnar Morling - Red Hat Ballroom BC	**Probabilistic programming for software engineers** Michael Tingley - Facebook Pacific DEKJ	**Balancing Priorities: Revenue Generation vs. Revenue Protection** Dominica DeGrandis - Tasktop Bayview AB	**Javascript** Pacific LMNO	**Impact Starts With You** Julia Nguyen - Mailchimp Seacliff ABC	**Checkr** Pacific BC	**Session Continues**
6:15PM	**EXHIBITOR RECEPTION IN EXHIBITOR AREAS ON MARKET STREET LEVEL AND PACIFIC LEVEL**							

WePay circa 2017

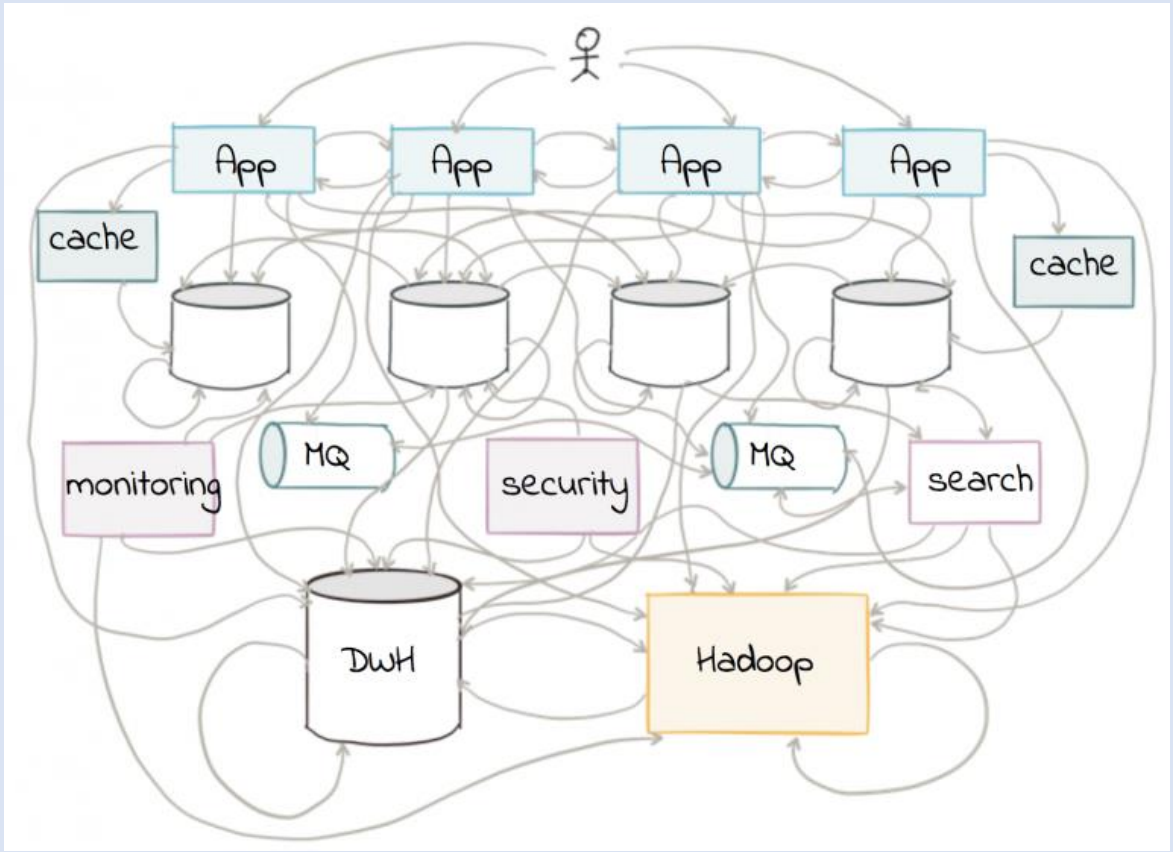


Kafka Connect BigQuery

- Open source connector that WePay wrote
- Stream data from Apache Kafka to Google BigQuery
- Supports GCS loads
- Supports realtime streaming inserts
- Automatic table schema updates

Problems

- Pipeline for Datastore was still on Airflow
- No pipeline at all for Cassandra or Bigtable
- BigQuery needed logging data
- Elastic search needed data
- Graph DB needed data



https://www.confluent.io/blog/building-real-time-streaming-etl-pipeline-20-minutes/

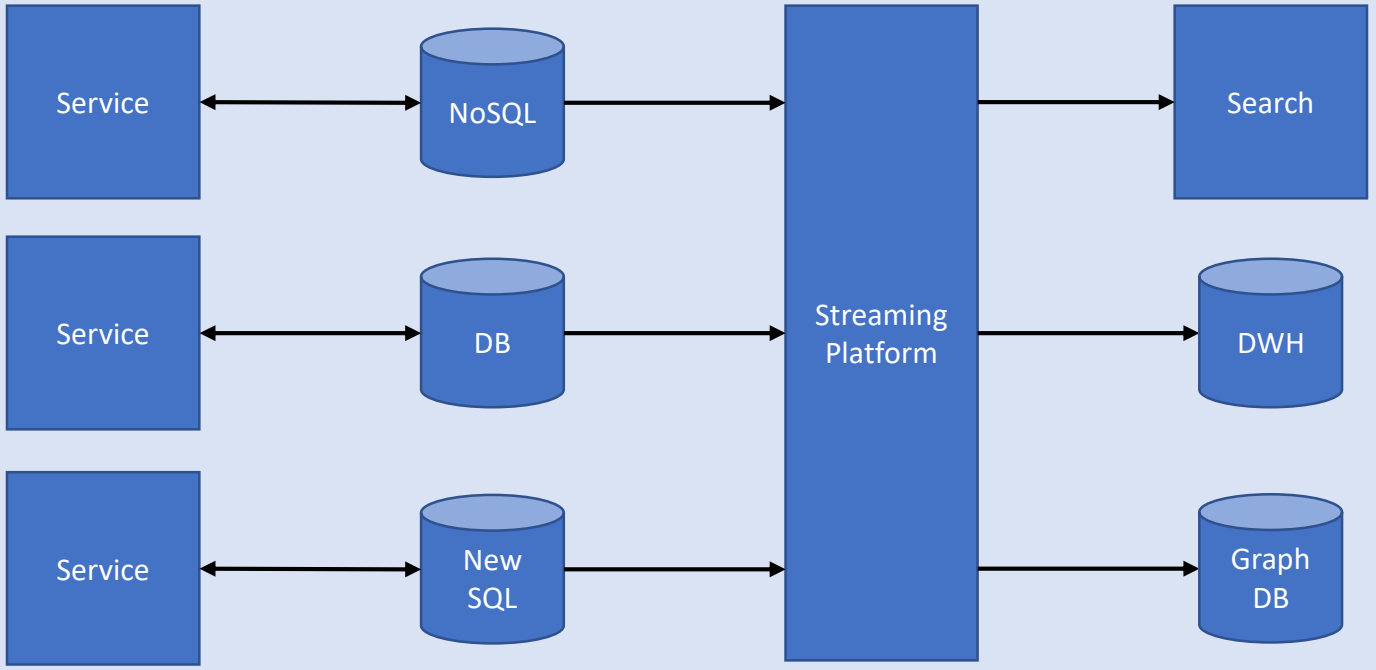
Six stages of data pipeline maturity

- Stage 0: None
- Stage 1: Batch
- Stage 2: Realtime
- **Stage 3: Integration**
- Stage 4: Automation
- Stage 5: Decentralization

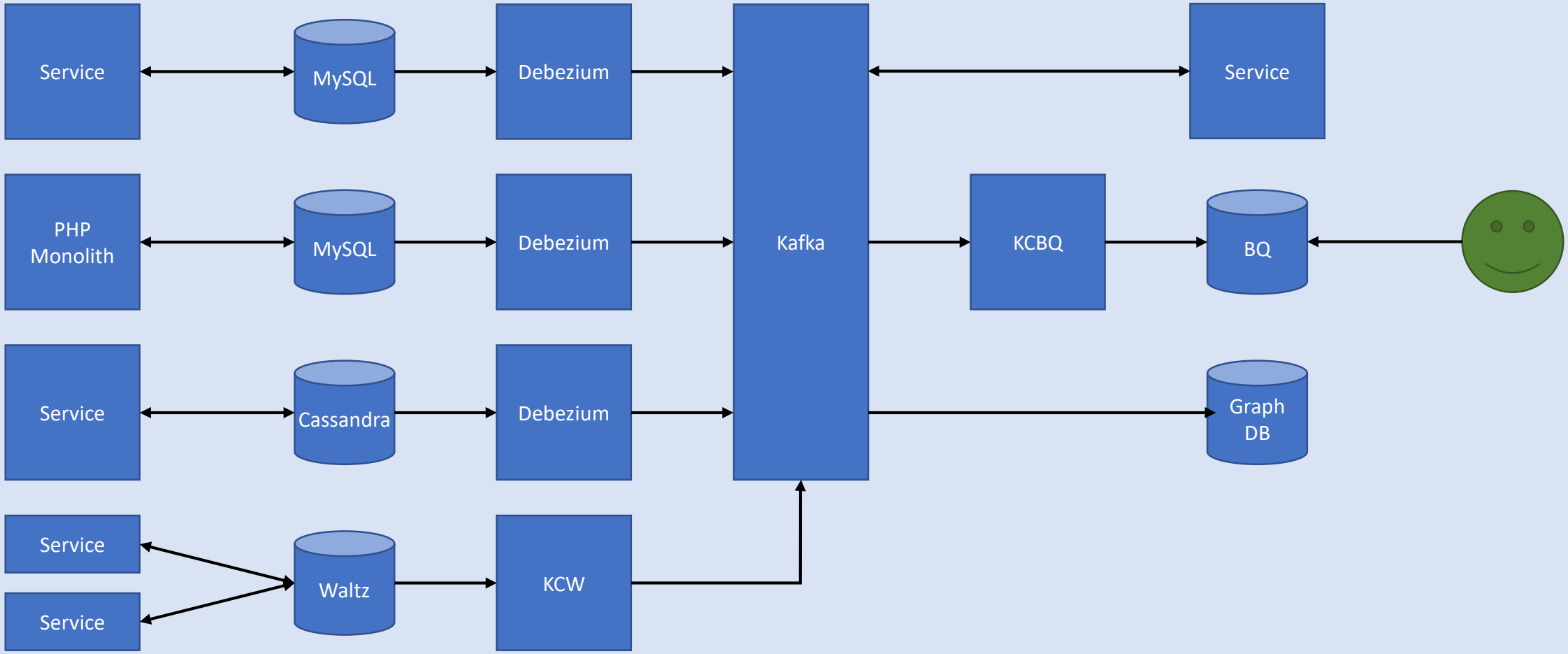
You might be ready for integration if…

- You have microservices
- You have a diverse database ecosystem
- You have many specialized derived data systems
- You have a team of data engineers
- You have a mature SRE organization

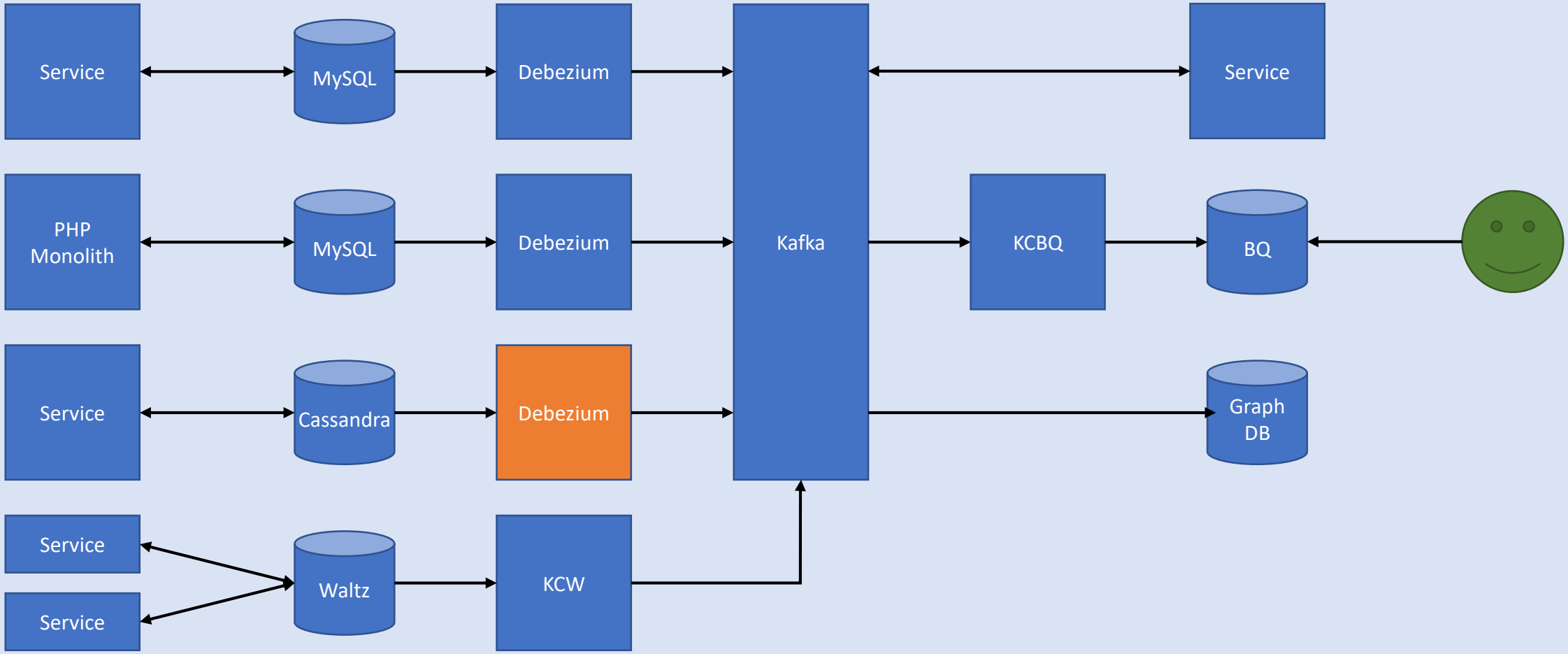
Stage 3: Integration



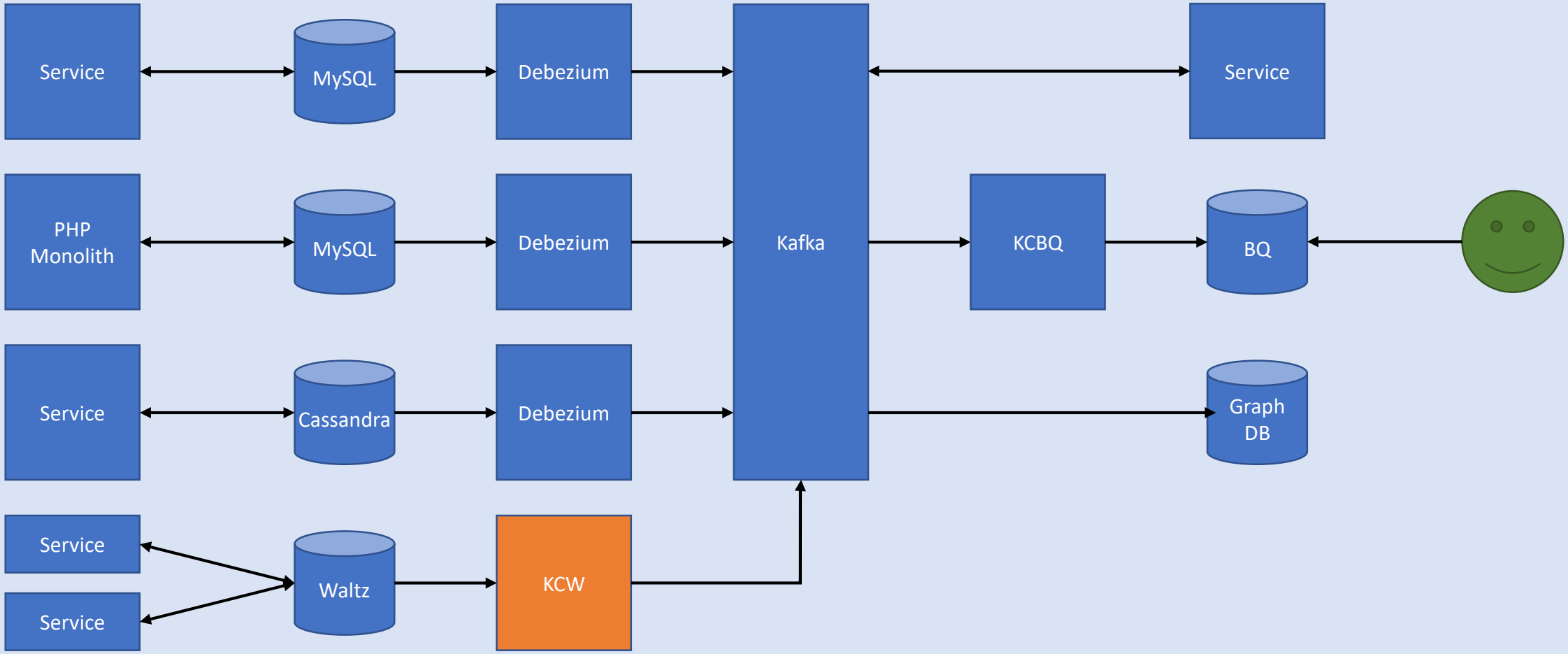
WePay circa 2019



WePay circa 2019



WePay circa 2019



Yasuhiro Matsuda

Principal Software Engineer

Waltz: A Distributed Write-Ahead Log

By Yasuhiro Matsuda on Sep 9, 2019

We are happy to announce the open source release of Waltz. Waltz is a distributed write-ahead log. It was initially designed to be the ledger of money transactions on the WePay system and was generalized for broader use cases of distributed systems that require serializable consistency. Waltz is similar to existing log systems like Kafka in that it accepts/persists/propagates transaction data produced/consumed by many services. However, unlike other systems, Waltz provides a machinery that facilitates a serializable consistency in distributed applications. It detects conflicting transactions before they are committed to the log. Waltz is regarded as the single source of truth rather than the database, and it enables a highly reliable log-centric system architecture.

Background

Databases

The WePay system has been constantly growing to handle more traffic and more functionalities. We split a large service into smaller services to keep the system manageable when it makes sense. Each service typically has its own database. For better isolation it is not shared with other services.

It is not trivial to keep all databases consistent when there are faults such as network failures, process failures, and machine failures. Services interact with each other over the network. Interactions often result in database updates on both sides. Faults may cause inconsistencies between the databases. Most such inconsistencies are fixed by daemon threads that perform check-and-repair operations periodically. But not every repair can be automated. Sometimes manual operations are required.

On top of this, databases are replicated for fault tolerance. We use MySQL async replication. When the primary region goes down, a region failover will happen, and the backup region will take over the processing so that we can continue processing payments. Multi-region replication has its own issues. A database update in the master database will not appear in a slave database instantly. There is always a latency, and replication lags are the norm. There is no guarantee that new master databases have all up-to-date data nor that they are in sync with each other.

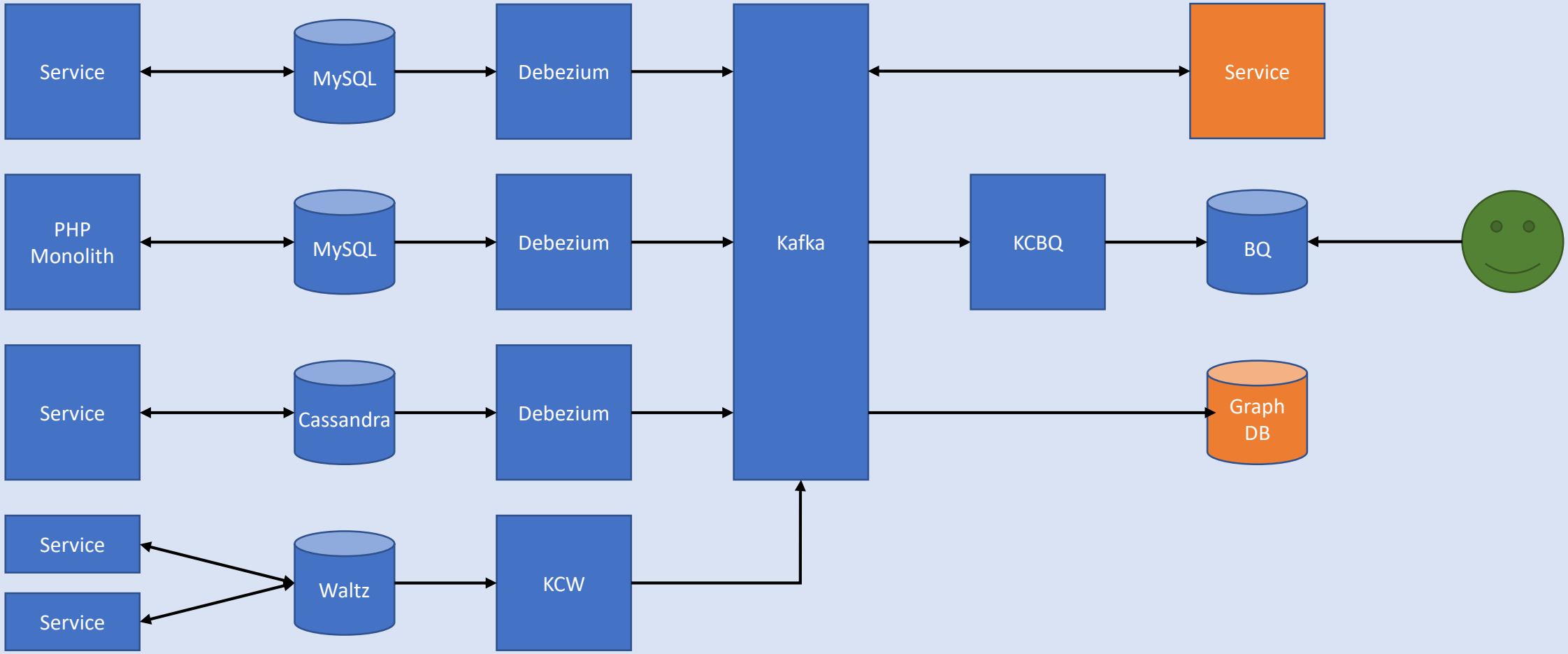
Stream Oriented Processing

We employ asynchronous processing in many places. We want to defer updates that don't require immediate consistency. This makes the main transactions lighter and improves the response and throughput. We do so by using a stream oriented processing with Kafka. A service updates its own database and writes messages to Kafka at the same time. The same service or a different service asynchronously performs another database update when it consumes Kafka messages. This works well, but the drawback is that a service has to write to two separate storage systems, a database and Kafka. We still need check-and-repair.

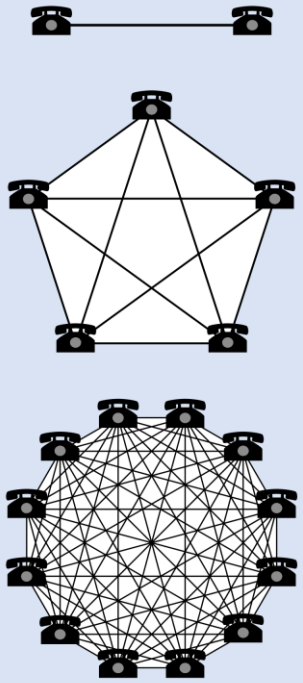
Basic Idea

Waltz is what we describe as a write-ahead log. This recorded log is neither the output of a change-data-capture from a database nor a secondary output from an application. It is the primary information of the

WePay circa 2019



Metcalfe’s law



Chris Riccomini

Software engineering stuff

Chris Follow

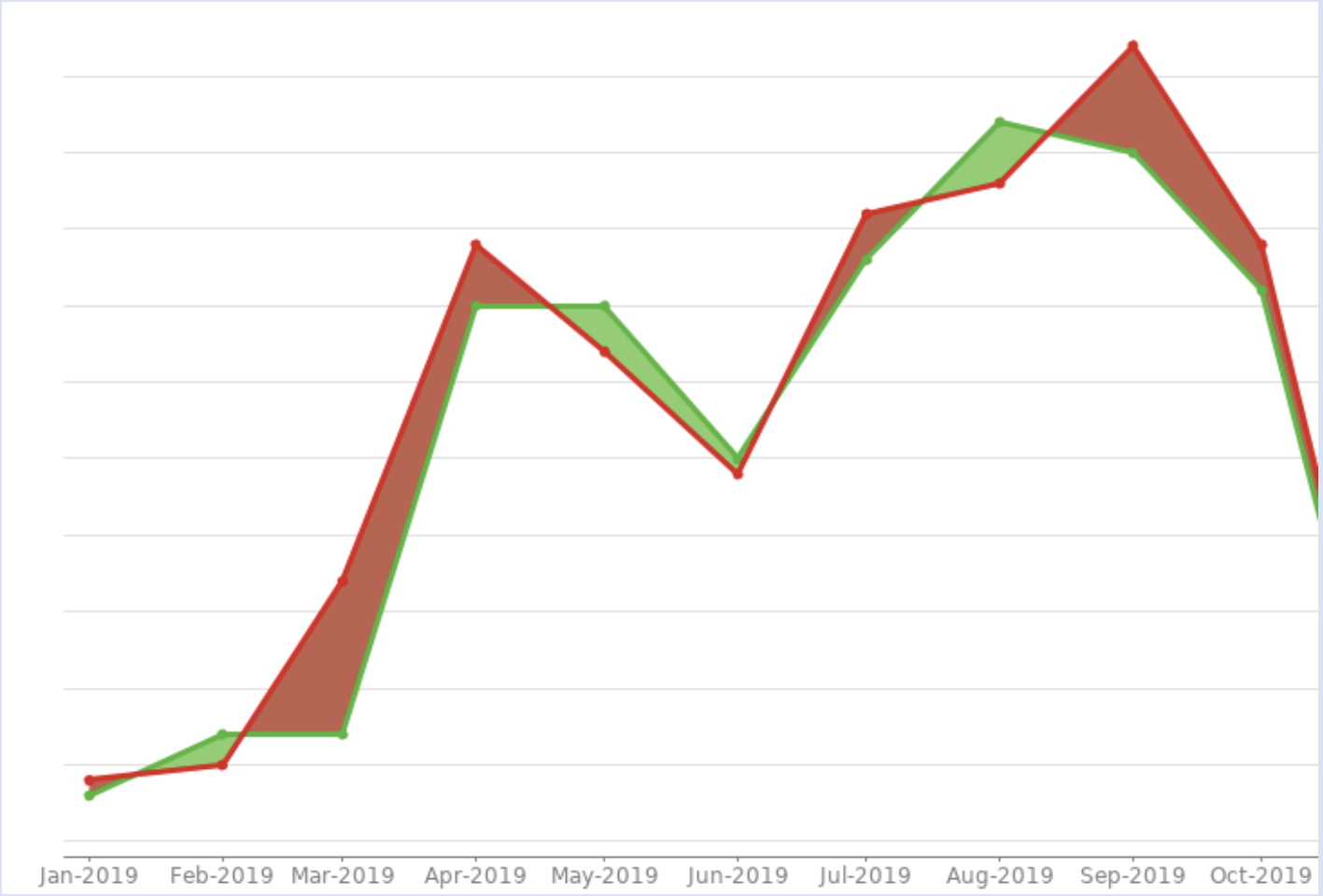
Software engineer. WePay, LinkedIn, PayPal. Co-author Apache Samza, PMC Apache Airflow.

Kafka is your escape hatch

I've become much more comfortable with the idea of vendor lock-in. Or rather, I don't feel as locked in as I used to. The odd thing is, I'm using more proprietary systems than I ever have before (thanks to the cloud). Apache Kafka is what's making me comfortable. Specifically, Kafka connect.

Problems

- Add new channel to replica MySQL DB
- Create and configure Kafka topics
- Add new Debezium connector to Kafka connect
- Create destination dataset in BigQuery
- Add new KCBQ connector to Kafka connect
- Create BigQuery views
- Configure data quality checks for new tables
- Grant access to BigQuery dataset
- Deploy stream processors or workflows


Jan-2019
Feb-2019
Mar-2019
Apr-2019
May-2019
Jun-2019
Jul-2019
Aug-2019
Sep-2019
Oct-2019

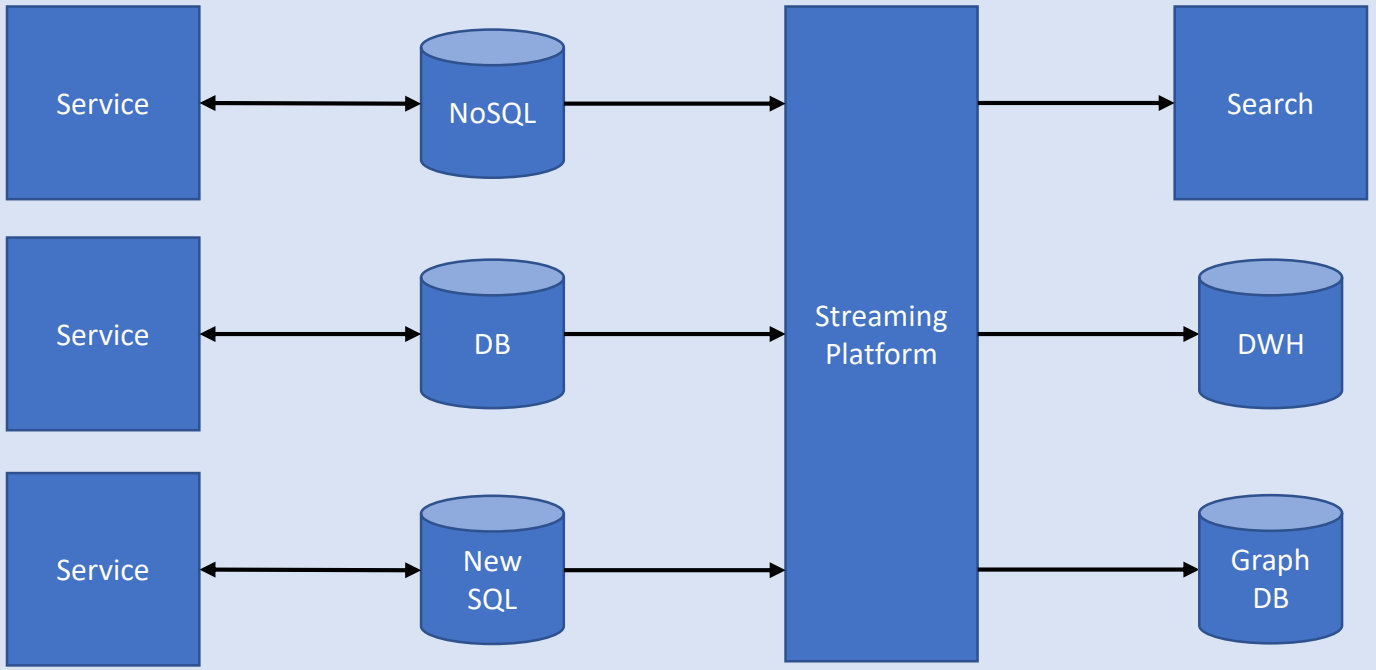
Six stages of data pipeline maturity

- Stage 0: None
- Stage 1: Batch
- Stage 2: Realtime
- Stage 3: Integration
- **Stage 4: Automation**
- Stage 5: Decentralization

You might be ready for automation if…

- Your SREs can’t keep up
- You’re spending a lot of time on manual toil
- You don’t have time for the fun stuff

Stage 4: Automation



Automated Data Management

Data Catalog | RBAC/IAM/ACL | DLP | ...

Automated Operations

Orchestration | Monitoring | Configuration | ...

Automated Operations

*“If a human operator needs to touch your system during **normal operations**, you have a bug.”*

-- Carla Geisser, Google SRE

Normal operations?

- Add new channel to replica MySQL DB
- Create and configure Kafka topics
- Add new Debezium connector to Kafka connect
- Create destination dataset in BigQuery
- Add new KCBQ connector to Kafka connect
- Create BigQuery views
- Configure data quality checks for new tables
- Granting access
- Deploying stream processors or workflows

Automated operations

- Terraform
- Ansible
- Helm
- Salt
- CloudFormation
- Chef
- Puppet
- Spinnaker

Terraform

```
provider "kafka" {
  bootstrap_servers = ["localhost:9092"]
}

resource "kafka_topic" "logs" {
  name               = "systemd_logs"
  replication_factor = 2
  partitions         = 100

  config = {
    "segment.ms"     = "20000"
    "cleanup.policy" = "compact"
  }
}
```

Terraform

```
provider "kafka-connect" {
  url = "http://localhost:8083"
}

resource "kafka-connect_connector" "sqlite-sink" {
  name = "test-sink"

  config = {
    "name"            = "test-sink"
    "connector.class" = "io.confluent.connect.jdbc.JdbcSinkConnector"
    "tasks.max"       = "1"
    "topics"          = "orders"
    "connection.url"  = "jdbc:sqlite:test.db"
    "auto.create"     = "true"
  }
}
```

But we were doing this… why so much toil?

- We had Terraform and Ansible
- We were on the cloud
- We had BigQuery scripts and tooling

Spending time on data management

- Who gets access to this data?
- How long can this data be persisted?
- Is this data allowed in this system?
- Which geographies must data be persisted in?
- Should columns be masked?

Photo by Darren Halstead

Regulation is ~~coming~~ here

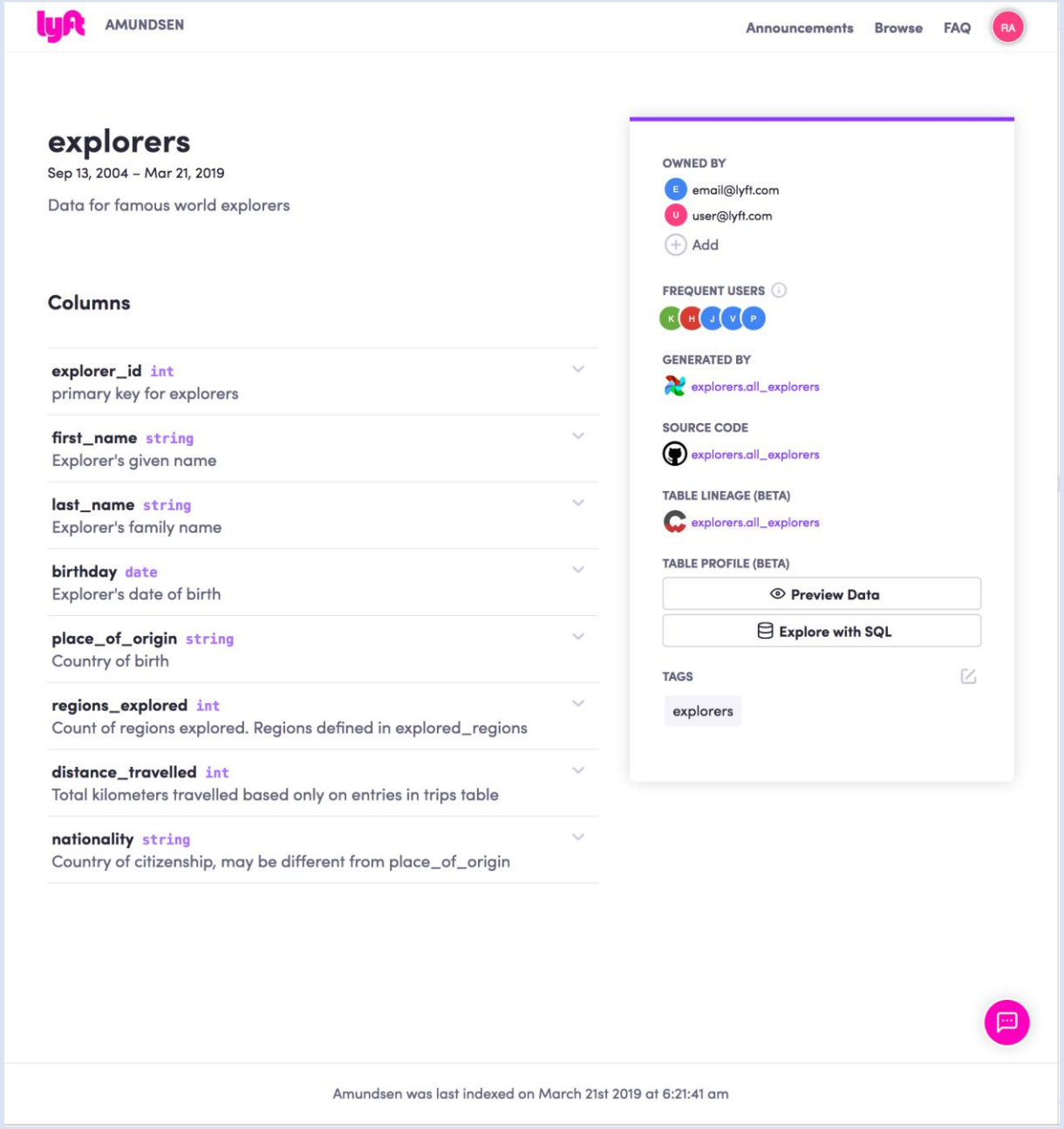
GDPR, CCPA, PCI, HIPAA, SOX, SHIELD, …

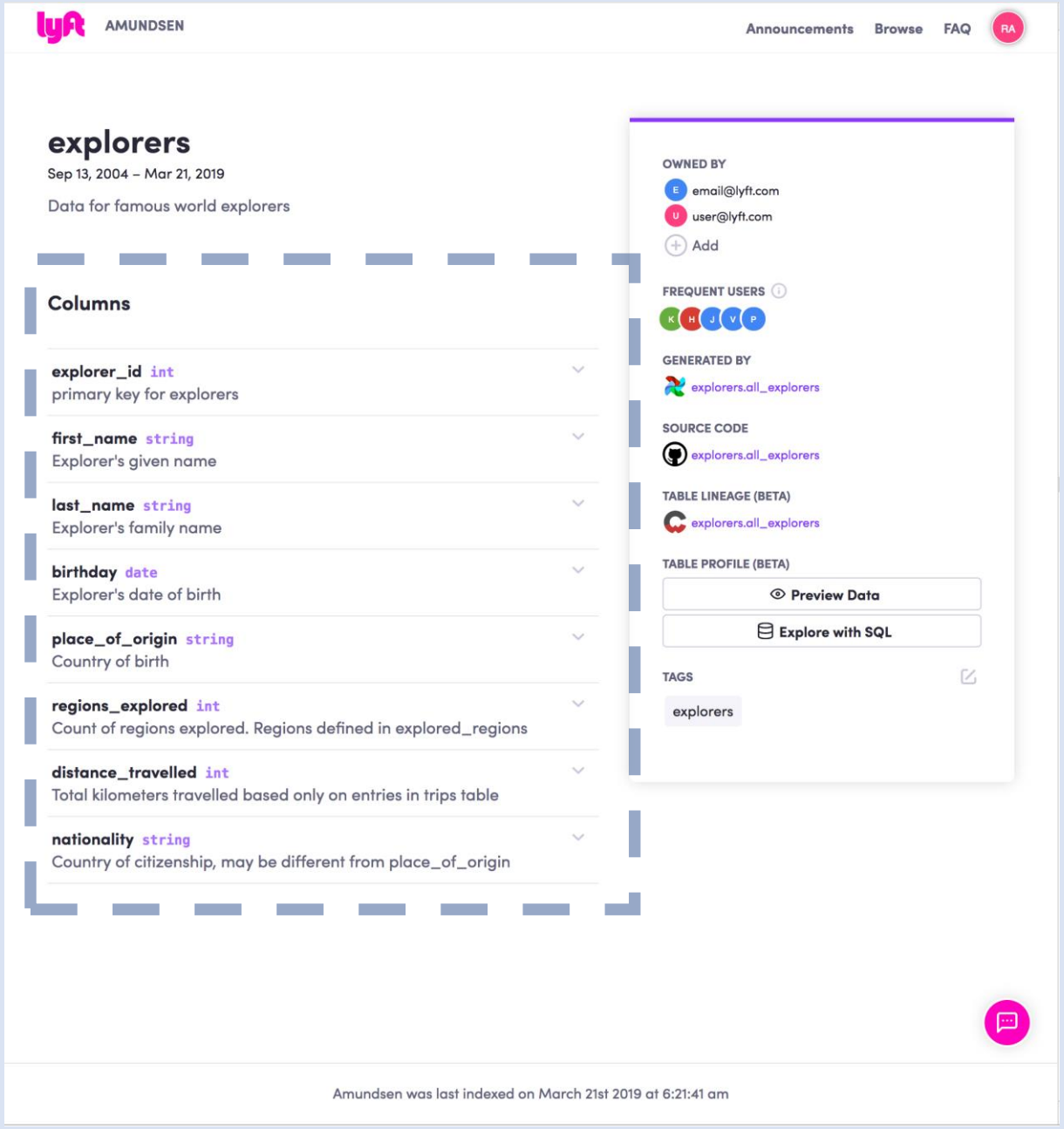
Photo by Darren Halstead

Automated Data Management

Set up a data catalog

- Location
- Schema
- Ownership
- Lineage
- Encryption
- Versioning


AMUNDSEN
Announcements
Browse
FAQ
RA
explorers
Sep 13, 2004 – Mar 21, 2019
Data for famous world explorers
Columns
explorer_id int
primary key for explorers
first_name string
Explorer's given name
last_name string
Explorer's family name
birthday date
Explorer's date of birth
place_of_origin string
Country of birth
regions_explored int
Count of regions explored. Regions defined in explored_regions
distance_travelled int
Total kilometers travelled based only on entries in trips table
nationality string
Country of citizenship, may be different from place_of_origin
OWNED BY
email@lyft.com
user@lyft.com
Add
FREQUENT USERS
GENERATED BY
explorers.all_explorers
SOURCE CODE
explorers.all_explorers
TABLE LINEAGE (BETA)
explorers.all_explorers
TABLE PROFILE (BETA)
Preview Data
Explore with SQL
TAGS
explorers
Amundsen was last indexed on March 21st 2019 at 6:21:41 am


AMUNDSEN
Announcements
Browse
FAQ
RA
explorers
Sep 13, 2004 – Mar 21, 2019
Data for famous world explorers
Columns
explorer_id int
primary key for explorers
first_name string
Explorer's given name
last_name string
Explorer's family name
birthday date
Explorer's date of birth
place_of_origin string
Country of birth
regions_explored int
Count of regions explored. Regions defined in explored_regions
distance_travelled int
Total kilometers travelled based only on entries in trips table
nationality string
Country of citizenship, may be different from place_of_origin
OWNED BY
email@lyft.com
user@lyft.com
Add
FREQUENT USERS
GENERATED BY
explorers.all_explorers
SOURCE CODE
explorers.all_explorers
TABLE LINEAGE (BETA)
explorers.all_explorers
TABLE PROFILE (BETA)
Preview Data
Explore with SQL
TAGS
explorers
Amundsen was last indexed on March 21st 2019 at 6:21:41 am

explorers

Sep 13, 2004 – Mar 21, 2019

Data for famous world explorers

Columns

explorer_id `int`
primary key for explorers

first_name `string`
Explorer's given name

last_name `string`
Explorer's family name

birthday `date`
Explorer's date of birth

place_of_origin `string`
Country of birth

regions_explored `int`
Count of regions explored. Regions defined in explored_regions

distance_travelled `int`
Total kilometers travelled based only on entries in trips table

nationality `string`
Country of citizenship, may be different from place_of_origin

OWNED BY
email@lyft.com
user@lyft.com
Add

FREQUENT USERS

GENERATED BY
explorers.all_explorers

SOURCE CODE
explorers.all_explorers

TABLE LINEAGE (BETA)
explorers.all_explorers

TABLE PROFILE (BETA)
Preview Data
Explore with SQL

TAGS
explorers

explorers

Sep 13, 2004 – Mar 21, 2019

Data for famous world explorers

Columns

explorer_id `int`
primary key for explorers

first_name `string`
Explorer's given name

last_name `string`
Explorer's family name

birthday `date`
Explorer's date of birth

place_of_origin `string`
Country of birth

regions_explored `int`
Count of regions explored. Regions defined in explored_regions

distance_travelled `int`
Total kilometers travelled based only on entries in trips table

nationality `string`
Country of citizenship, may be different from place_of_origin

OWNED BY
email@lyft.com
user@lyft.com
Add

FREQUENT USERS

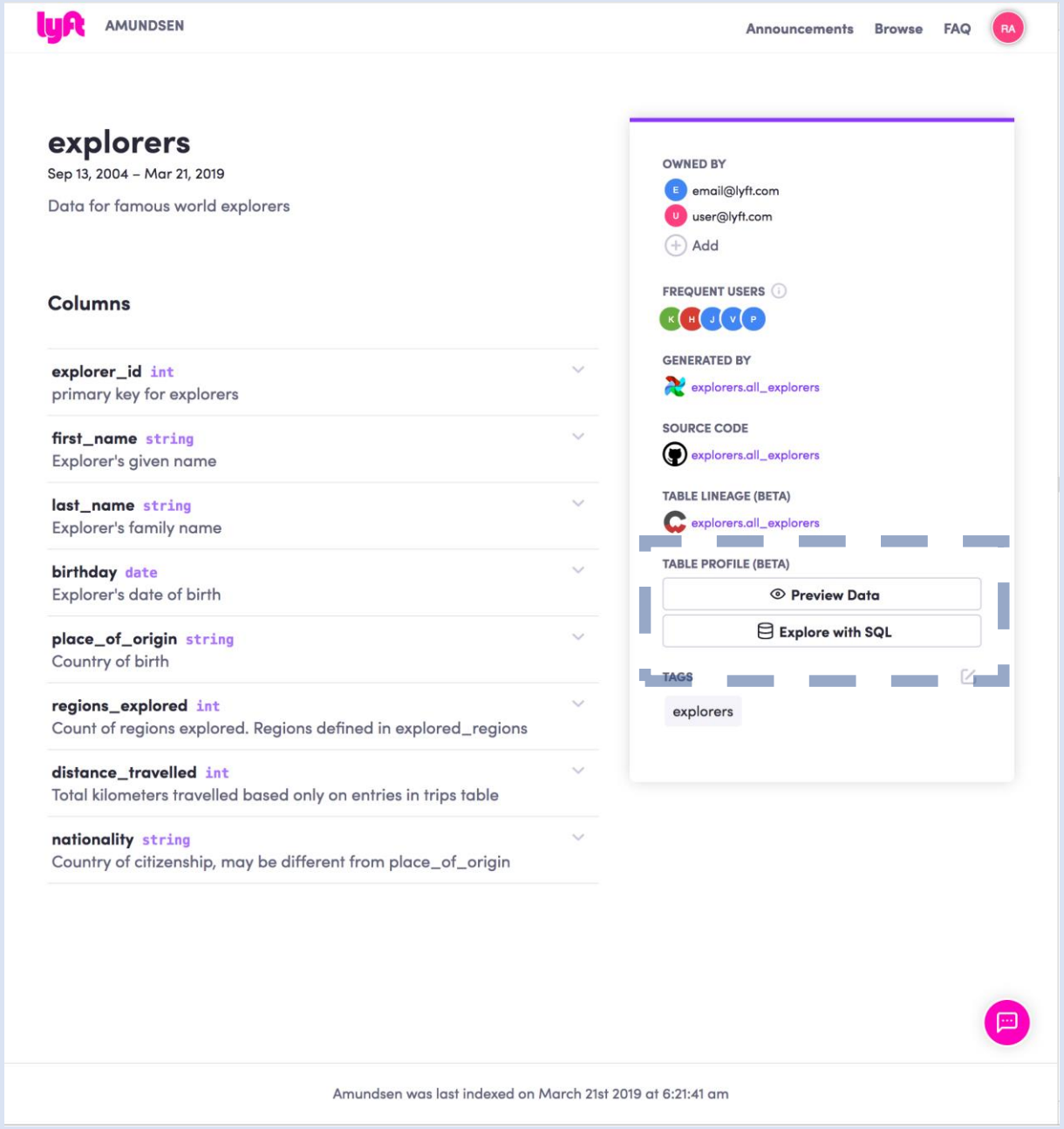
GENERATED BY
explorers.all_explorers

SOURCE CODE
explorers.all_explorers

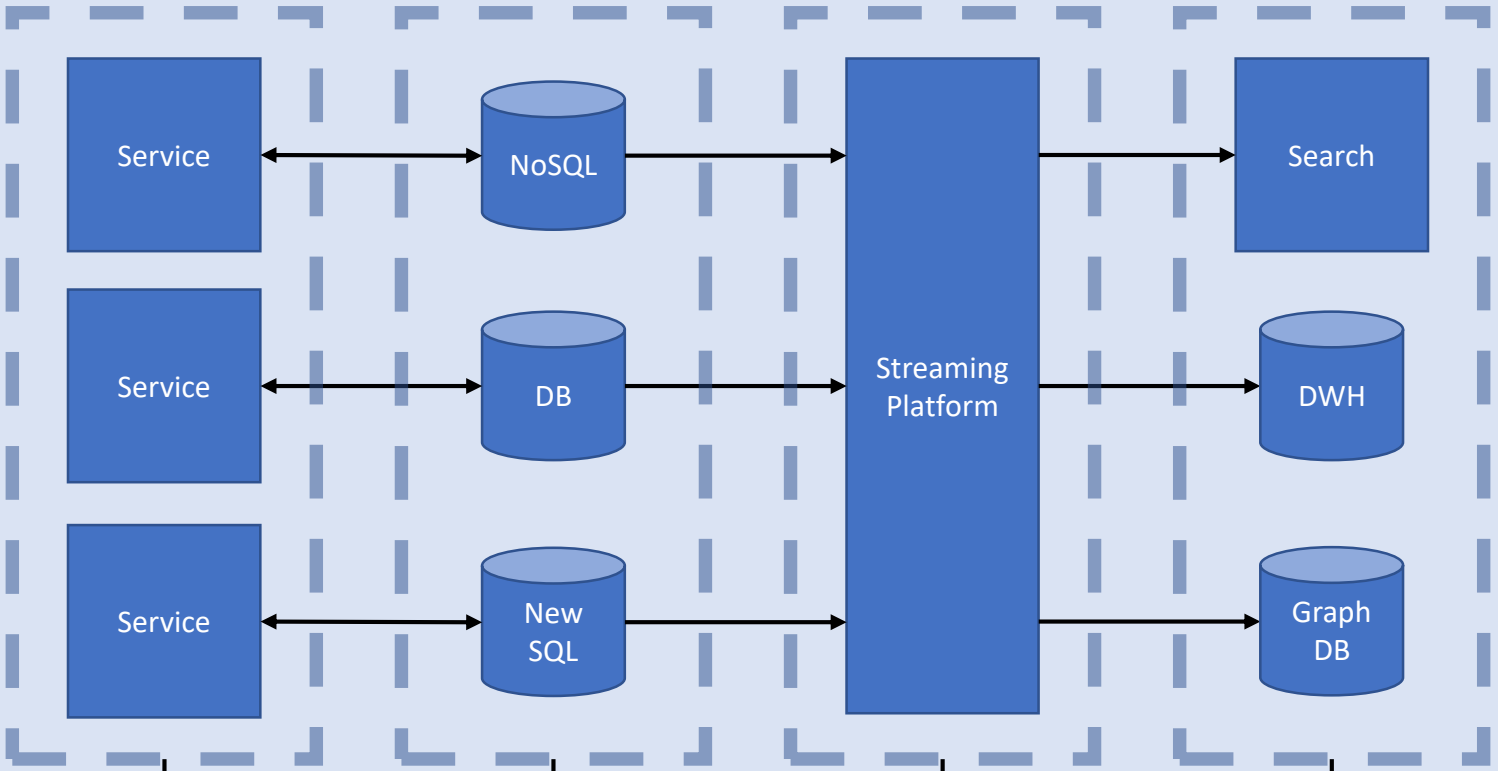
TABLE LINEAGE (BETA)
explorers.all_explorers

TABLE PROFILE (BETA)
Preview Data
Explore with SQL

TAGS
explorers


AMUNDSEN
Announcements
Browse
FAQ
RA
explorers
Sep 13, 2004 – Mar 21, 2019
Data for famous world explorers
Columns
explorer_id int
primary key for explorers
first_name string
Explorer's given name
last_name string
Explorer's family name
birthday date
Explorer's date of birth
place_of_origin string
Country of birth
regions_explored int
Count of regions explored. Regions defined in explored_regions
distance_travelled int
Total kilometers travelled based only on entries in trips table
nationality string
Country of citizenship, may be different from place_of_origin
OWNED BY
email@lyft.com
user@lyft.com
Add
FREQUENT USERS
GENERATED BY
explorers.all_explorers
SOURCE CODE
explorers.all_explorers
TABLE LINEAGE (BETA)
explorers.all_explorers
TABLE PROFILE (BETA)
Preview Data
Explore with SQL
TAGS
explorers
Amundsen was last indexed on March 21st 2019 at 6:21:41 am

Stage 4: Automation



Automated Data Management

Data Catalog | RBAC/IAM/ACL | DLP | ...

Automated Operations

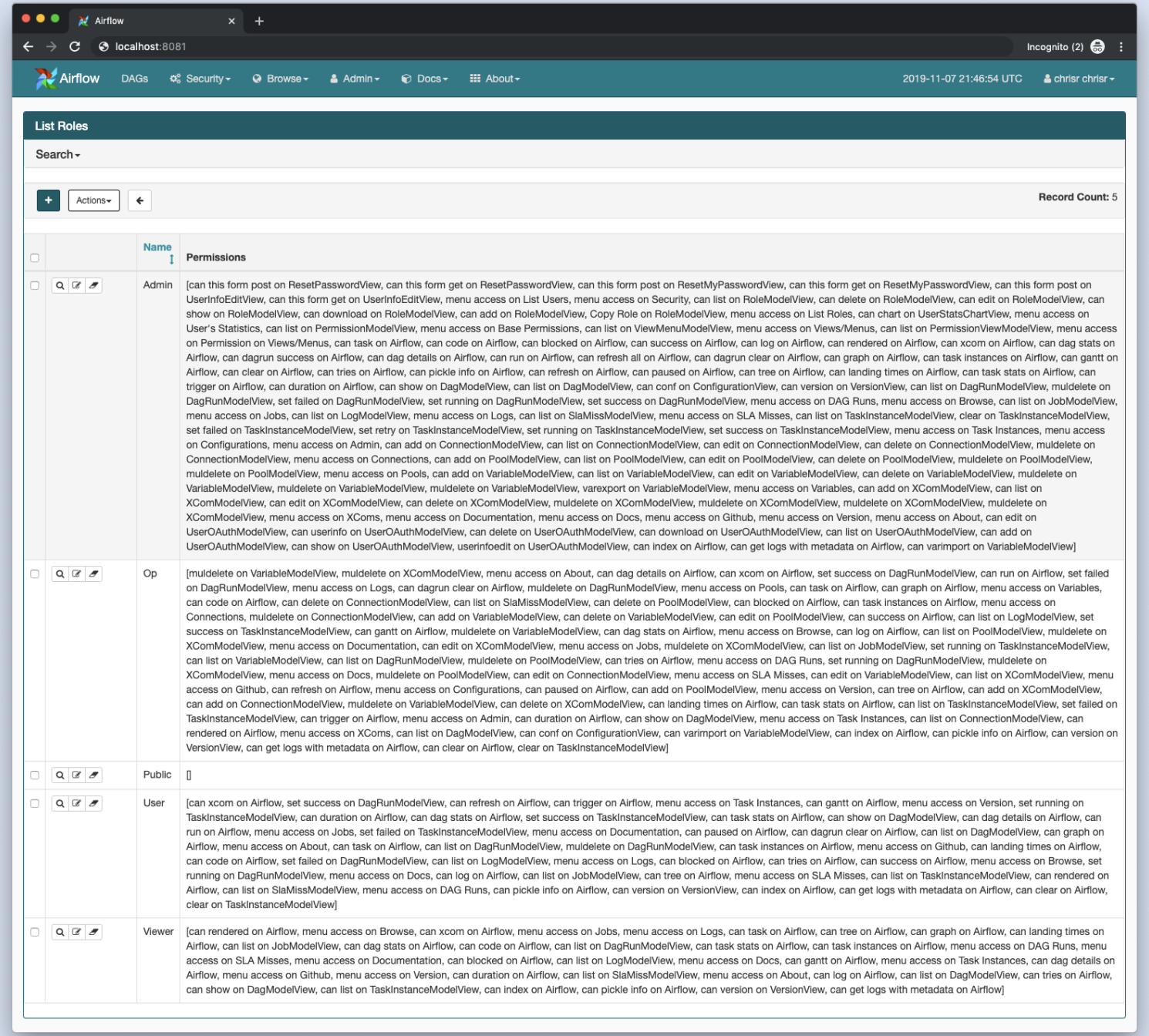
Orchestration | Monitoring | Configuration | ...

Configure your access

- RBAC
- IAM
- ACL

Configure your policies

- Role based access controls
- Identity access management
- Access control lists


Airflow
localhost:8081
Incognito (2)
Airflow
DAGs
Security
Browse
Admin
Docs
About
2019-11-07 21:46:54 UTC
chrisr chrisr
List Roles
Search
Actions
Record Count: 5
Name
Permissions
Admin
[can this form post on ResetPasswordView, can this form get on ResetPasswordView, can this form post on ResetMyPasswordView, can this form get on ResetMyPasswordView, can this form post on UserInfoEditView, can this form get on UserInfoEditView, menu access on List Users, menu access on Security, can list on RoleModelView, can delete on RoleModelView, can edit on RoleModelView, can show on RoleModelView, can download on RoleModelView, can add on RoleModelView, Copy Role on RoleModelView, menu access on List Roles, can chart on UserStatsChartView, menu access on User's Statistics, can list on PermissionModelView, menu access on Base Permissions, can list on ViewMenuModelView, menu access on Views/Menus, can list on PermissionViewModelView, menu access on Permission on Views/Menus, can task on Airflow, can code on Airflow, can blocked on Airflow, can success on Airflow, can log on Airflow, can rendered on Airflow, can xcom on Airflow, can dag stats on Airflow, can dagrun success on Airflow, can dag details on Airflow, can run on Airflow, can refresh all on Airflow, can dagrun clear on Airflow, can graph on Airflow, can task instances on Airflow, can gantt on Airflow, can clear on Airflow, can tries on Airflow, can pickle info on Airflow, can refresh on Airflow, can paused on Airflow, can tree on Airflow, can landing times on Airflow, can task stats on Airflow, can trigger on Airflow, can duration on Airflow, can show on DagModelView, can list on DagModelView, can conf on ConfigurationView, can version on VersionView, can list on DagRunModelView, muldelete on DagRunModelView, set failed on DagRunModelView, set running on DagRunModelView, set success on DagRunModelView, menu access on DAG Runs, menu access on Browse, can list on JobModelView, menu access on Jobs, can list on LogModelView, menu access on Logs, can list on SlaMissModelView, menu access on SLA Misses, can list on TaskInstanceModelView, clear on TaskInstanceModelView, set failed on TaskInstanceModelView, set retry on TaskInstanceModelView, set running on TaskInstanceModelView, set success on TaskInstanceModelView, menu access on Task Instances, menu access on Configurations, menu access on Admin, can add on ConnectionModelView, can list on ConnectionModelView, can edit on ConnectionModelView, can delete on ConnectionModelView, muldelete on ConnectionModelView, menu access on Connections, can add on PoolModelView, can list on PoolModelView, can edit on PoolModelView, can delete on PoolModelView, muldelete on PoolModelView, muldelete on PoolModelView, menu access on Pools, can add on VariableModelView, can list on VariableModelView, can edit on VariableModelView, can delete on VariableModelView, muldelete on VariableModelView, muldelete on VariableModelView, muldelete on VariableModelView, varexport on VariableModelView, menu access on Variables, can add on XComModelView, can list on XComModelView, can edit on XComModelView, can delete on XComModelView, muldelete on XComModelView, muldelete on XComModelView, muldelete on XComModelView, muldelete on XComModelView, menu access on XComs, menu access on Documentation, menu access on Docs, menu access on Github, menu access on Version, menu access on About, can edit on UserOAuthModelView, can userinfo on UserOAuthModelView, can delete on UserOAuthModelView, can download on UserOAuthModelView, can list on UserOAuthModelView, can add on UserOAuthModelView, can show on UserOAuthModelView, userinfoedit on UserOAuthModelView, can index on Airflow, can get logs with metadata on Airflow, can varimport on VariableModelView]
Op
[muldelete on VariableModelView, muldelete on XComModelView, menu access on About, can dag details on Airflow, can xcom on Airflow, set success on DagRunModelView, can run on Airflow, set failed on DagRunModelView, menu access on Logs, can dagrun clear on Airflow, muldelete on DagRunModelView, menu access on Pools, can task on Airflow, can graph on Airflow, menu access on Variables, can code on Airflow, can delete on ConnectionModelView, can list on SlaMissModelView, can delete on PoolModelView, can blocked on Airflow, can task instances on Airflow, menu access on Connections, muldelete on ConnectionModelView, can add on VariableModelView, can delete on VariableModelView, can edit on PoolModelView, can success on Airflow, can list on LogModelView, set success on TaskInstanceModelView, can gantt on Airflow, muldelete on VariableModelView, can dag stats on Airflow, menu access on Browse, can log on Airflow, can list on PoolModelView, muldelete on XComModelView, menu access on Documentation, can edit on XComModelView, menu access on Jobs, muldelete on XComModelView, can list on JobModelView, set running on TaskInstanceModelView, can list on VariableModelView, can list on DagRunModelView, muldelete on PoolModelView, can tries on Airflow, menu access on DAG Runs, set running on DagRunModelView, muldelete on XComModelView, menu access on Docs, muldelete on PoolModelView, can edit on ConnectionModelView, menu access on SLA Misses, can edit on VariableModelView, can list on XComModelView, menu access on Github, can refresh on Airflow, menu access on Configurations, can paused on Airflow, can add on PoolModelView, menu access on Version, can tree on Airflow, can add on XComModelView, can add on ConnectionModelView, muldelete on VariableModelView, can delete on XComModelView, can landing times on Airflow, can task stats on Airflow, can list on TaskInstanceModelView, set failed on TaskInstanceModelView, can trigger on Airflow, menu access on Admin, can duration on Airflow, can show on DagModelView, menu access on Task Instances, can list on ConnectionModelView, can rendered on Airflow, menu access on XComs, can list on DagModelView, can conf on ConfigurationView, can varimport on VariableModelView, can index on Airflow, can pickle info on Airflow, can version on VersionView, can get logs with metadata on Airflow, can clear on Airflow, clear on TaskInstanceModelView]
Public
[]
User
[can xcom on Airflow, set success on DagRunModelView, can refresh on Airflow, can trigger on Airflow, menu access on Task Instances, can gantt on Airflow, menu access on Version, set running on TaskInstanceModelView, can duration on Airflow, can dag stats on Airflow, set success on TaskInstanceModelView, can task stats on Airflow, can show on DagModelView, can dag details on Airflow, can run on Airflow, menu access on Jobs, set failed on TaskInstanceModelView, menu access on Documentation, can paused on Airflow, can dagrun clear on Airflow, can list on DagModelView, can graph on Airflow, menu access on About, can task on Airflow, can list on DagRunModelView, muldelete on DagRunModelView, can task instances on Airflow, menu access on Github, can landing times on Airflow, can code on Airflow, set failed on DagRunModelView, can list on LogModelView, menu access on Logs, can blocked on Airflow, can tries on Airflow, can success on Airflow, menu access on Browse, set running on DagRunModelView, menu access on Docs, can log on Airflow, can list on JobModelView, can tree on Airflow, menu access on SLA Misses, can list on TaskInstanceModelView, can rendered on Airflow, can list on SlaMissModelView, menu access on DAG Runs, can pickle info on Airflow, can version on VersionView, can index on Airflow, can get logs with metadata on Airflow, can clear on Airflow, clear on TaskInstanceModelView]
Viewer
[can rendered on Airflow, menu access on Browse, can xcom on Airflow, menu access on Jobs, menu access on Logs, can task on Airflow, can tree on Airflow, can graph on Airflow, can landing times on Airflow, can list on JobModelView, can dag stats on Airflow, can code on Airflow, can list on DagRunModelView, can task stats on Airflow, can task instances on Airflow, menu access on DAG Runs, menu access on SLA Misses, menu access on Documentation, can blocked on Airflow, can list on LogModelView, menu access on Docs, can gantt on Airflow, menu access on Task Instances, can dag details on Airflow, menu access on Github, menu access on Version, can duration on Airflow, can list on SlaMissModelView, menu access on About, can log on Airflow, can list on DagModelView, can tries on Airflow, can show on DagModelView, can list on TaskInstanceModelView, can index on Airflow, can pickle info on Airflow, can version on VersionView, can get logs with metadata on Airflow]

Kafka ACLs with Terraform

```
provider "kafka" {
  bootstrap_servers = ["localhost:9092"]
  ca_cert      = file("../secrets/snakeoil-ca-1.crt")
  client_cert  = file("../secrets/kafkacat-ca1-signed.pem")
  client_key   = file("../secrets/kafkacat-raw-private-key.pem")
  skip_tls_verify   = true
}

resource "kafka_acl" "test" {
  resource_name       = "syslog"
  resource_type       = "Topic"
  acl_principal       = "User:Alice"
  acl_host            = "*"
  acl_operation       = "Write"
  acl_permission_type = "Deny"
}
```

Automate management

- New user access
- New data access
- Service account access
- Temporary access
- Unused access

Detect violations

- Auditing
- Data loss prevention

Cloud Data Loss Prevention

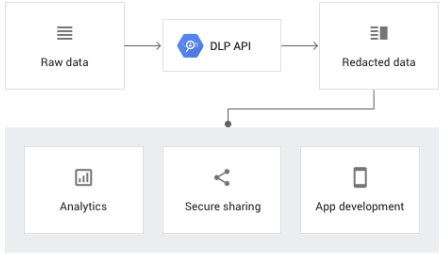
Automatically discover and redact sensitive data everywhere.

Try it free
Contact sales

View documentation for this product.

Classify and redact sensitive data

Cloud DLP helps you better understand and manage sensitive data. It provides fast, scalable classification and redaction for sensitive data elements like credit card numbers, names, social security numbers, US and selected international identifier numbers, phone numbers, and GCP credentials. Cloud DLP classifies this data using more than 90 predefined detectors to identify patterns, formats, and checksums, and even understands contextual clues. You can optionally redact data as well, using techniques like masking, secure hashing, tokenization, bucketing, and format-preserving encryption. Try Cloud DLP in this demo application.



Discover and classify sensitive data

One of the first steps to properly managing your sensitive data is knowing where it exists. Cloud DLP gives you the power to scan, discover, classify, and report on data from virtually anywhere. Cloud DLP has native support for scanning and classifying sensitive data in Cloud Storage, BigQuery, and Cloud Datastore and a streaming content API to enable support for additional data sources, custom workloads and applications.

Automatically mask your data to safely unlock more of the cloud

Today your data is your most critical asset. Cloud DLP provides tools to classify,

Detecting sensitive data

```
{
  "item":{
    "value":"My phone number is (415) 555-0890"
  },
  "inspectConfig":{
    "includeQuote":true,
    "minLikelihood":"POSSIBLE",
    "infoTypes":{
      "name":"PHONE_NUMBER"
    }
  }
}
```

```
{
  "result":{
    "findings":[
      {
        "quote":"(415) 555-0890",
        "infoType":{
          "name":"PHONE_NUMBER"
        },
        "likelihood":"VERY_LIKELY",
        "location":{
          "byteRange":{
            "start":"19",
            "end":"33"
          },
        },
      }
    ]
  }
}
```

Progress

- Users can find the data that they need
- Automated data management and operations

Problems

- Data engineering still manages configuration and deployment

Six stages of data pipeline maturity

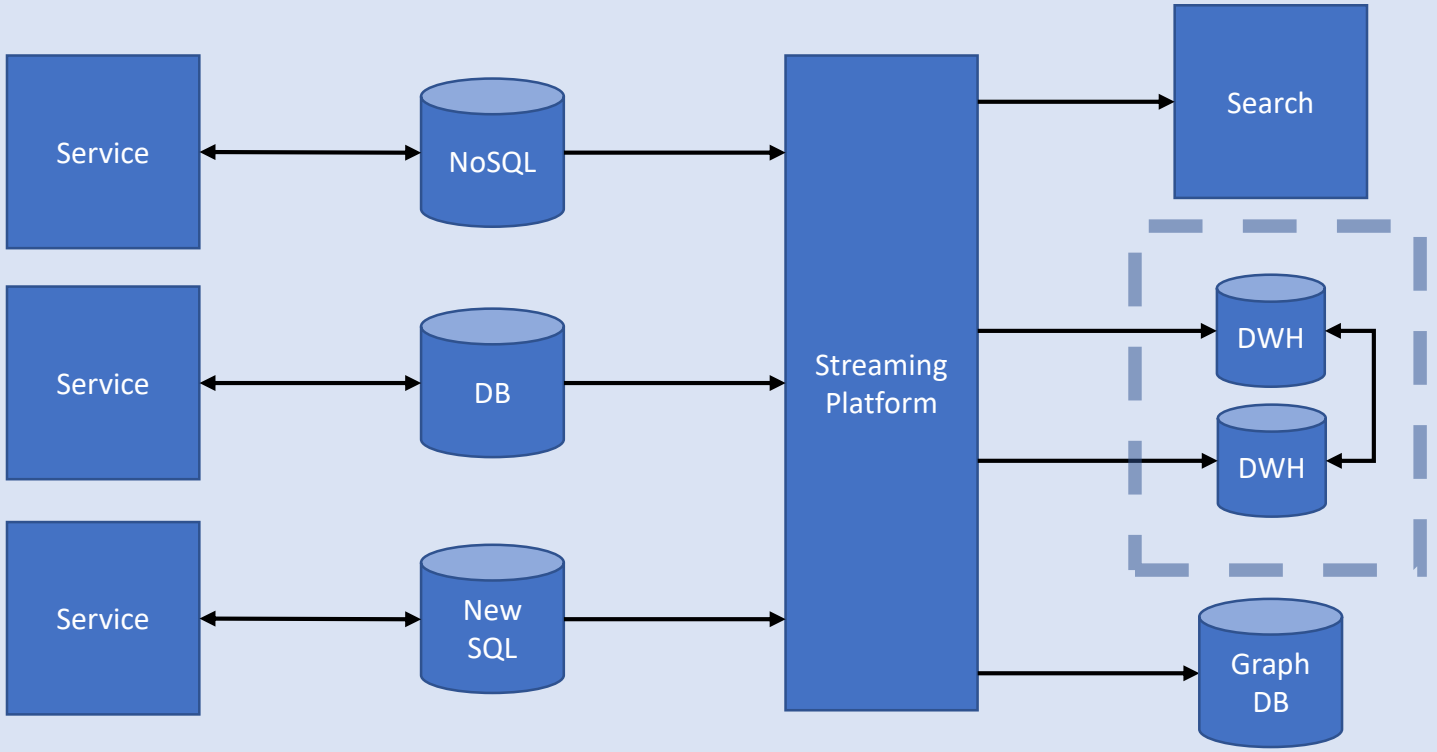
- Stage 0: None
- Stage 1: Batch
- Stage 2: Realtime
- Stage 3: Integration
- Stage 4: Automation
- **Stage 5: Decentralization**

You might be ready for decentralization if…

- You have a fully automated realtime data pipeline
- People still come to you to get data loaded

If we have an automated data pipeline and data warehouse,
do we need a **single team** to manage this?

Stage 5: Decentralization



Automated Data Management

Data Catalog | RBAC/IAM/ACL | DLP | ...

Automated Operations

Orchestration | Monitoring | Configuration | ...

From monolith to ~~microservices~~ microwarehouses

How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh

Many enterprises are investing in their next generation data lake, with the hope of democratizing data at scale to provide business insights and ultimately make automated intelligent decisions. Data platforms based on the data lake architecture have common failure modes that lead to unfulfilled promises at scale. To address these failure modes we need to shift from the centralized paradigm of a lake, or its predecessor data warehouse. We need to shift to a paradigm that draws from modern distributed architecture: considering domains as the first class concern, applying platform thinking to create self-serve data infrastructure, and treating data as a product.

20 May 2019

Zhamak Dehghani

Zhamak is a principal technology consultant at ThoughtWorks with a focus on distributed systems architecture and digital platform strategy at Enterprise. She is a member of ThoughtWorks Technology Advisory Board and contributes to the creation of ThoughtWorks Technology Radar.

ENTERPRISE ARCHITECTURE

BIG DATA

CONTENTS

Becoming a data-driven organization remains one of the top strategic goals of many companies I work with. My clients are well aware of the benefits of becoming intelligently empowered: providing the best customer experience based on data and hyper-personalization; reducing operational costs and time through data-driven optimizations; and giving employees super powers with trend analysis and business intelligence. They have been investing heavily in building enablers such as data and intelligence platforms. Despite increasing effort and investment in building such

7:00AM	**WOMEN & ALLIES IN TECH COMMUNITY BREAKFAST CO-SPONSORED BY NETFLIX - REGISTRATION REQUIRED**							
9:00AM	**It Really is a Series of Tubes** Molly Wright Steenson - AI, Ethics & Design: Author, Designer, Professor, Research Leader							
10:10AM	**BREAK**							
	Microservices Patterns & Practices Host: Colin Breck	**Modern Data Architectures** Host: Gwen Shapira	**Modern CS in the Real World** Host: Werner Schuster	**Practices of DevOps & Lean Thinking** Host: John Willis	**Pushing the Web Forward: JavaScript, Frameworks, Transpilers, and WebAssembly** Host: Dylan Schiemann Host: Phil Haack	**Socially Conscious Software** Host: Alex Qin	**SPONSORED SOLUTIONS TRACK III** Host: Nitin Bharti	**SPONSORED SOLUTIONS TRACK IV** Host: Nitin Bharti
10:35AM	**Managing Failure Modes in Microservice Architectures** Adrian Cockcroft - Amazon Web Services Ballroom A	**Future of Data Engineering** Chris Riccomini - WePay Ballroom BC	**Anti-Entropy Using CRDTs on HA Datastores @Netflix** Sailesh Mukil - Netflix Pacific DEKJ	**DevOps & Lean Thinking Panel** Bayview AB	**MakeCode: Types, Games and Machine Code** Michal Moskal - Microsoft Pacific LMNO	**How Do We Heal?** Alex Qin - the code cooperative Seacliff ABC	**Real-time Log Analytics at Scale with Kafka** Kresten Thorup - Humio Pacific BC	**Java on Kubernetes 101 Hands-On Lab!** Pratik Patel - IBM & William Korando - IBM Marina
11:25AM	**BREAK**							
11:50AM	**User & Device Identity for Microservices @ Netflix Scale** Satyajit Thadeshwar - Netflix Ballroom A	**Data Mesh Paradigm Shift in Data Platform Architecture** Zhamak Dehghani - ThoughtWorks Ballroom BC	**The Talk You've Been Await-Ing For** Steve Klabnik - Rust Core Team Pacific DEKJ	**Mapping the Evolution of Socio-Technical Systems** Cat Swetel - Ticketmaster Bayview AB	**Javascript Open Space** Facilitated Peer Sharing Seacliff D	**Holistic EdTech & Diversity** Antoine Patton - Unlock.Academy Seacliff ABC	**The Next Era in Engineering Velocity** Bryan Helmkamp - Code Climate Pacific BC	**Session Continues**
12:40PM	**LUNCH**							
1:40PM	**Stateful Programming Models in Serverless Functions** Chris Gillum - Microsoft Ballroom A	**Taming Large State: Lessons From Building Stream Processing** Sonali Sharma - Netflix & Shriya Arora - Netflix Ballroom BC	**Swift 4 TF** Paige Bailey - Google Pacific DEKJ	**DevOps & Lean Thinking Open Space** Facilitated Peer Sharing Seacliff D	**Goodbye View Source: Hello JavaScript in the Age of Compilers** Amal Hussein - npm Pacific LMNO	**Socially Conscious Software** Jabari Bell - giya.us Seacliff ABC	**The Case for Human-Centric Observability** Austin Parker - LightStep Pacific BC	**Java on Kubernetes 101 Hands-On Lab!** Pratik Patel - IBM & William Korando - IBM Marina
2:30PM	**BREAK**							
2:55PM	**Controlled Chaos: Taming Organic, Federated Growth of Microservices** Tobias Kunze - Glasnostic Ballroom A	**Modern Data Architectures Open Space** Facilitated Peer Sharing Seacliff D	**ML/AI Panel** Amy Unruh - Google & Chris Albon - Devoted Health & June Andrews - Stitch Fix & Paige Bailey - Google Pacific DEKJ	**The System of Profound Knowledge** Ben Rockwood - Packet Bayview AB	**Build Your Own WebAssembly Compiler** Colin Eberhardt - Scott Logic Pacific LMNO	**Socially Conscious Software** Tiffani Ashley Bell Seacliff ABC	**My Code Is Broken but So Is the Debugging Experience** Matt Robenolt - Sentry Pacific BC	**Session Continues**
3:45PM	**BREAK**							
4:10PM	**Beyond Microservices: Streams, State and Scalability** Gwen Shapira - Confluent Ballroom A	**Kafka Needs No Keeper** Colin McCabe - Confluent Ballroom BC	**Sorbet: Why and How We Built a Typechecker for Ruby** Dmitry Petrashko - Stripe Pacific DEKJ	**Incident Management in the Age of DevOps & SRE** Damon Edwards - Rundeck Bayview AB	**Make Your Electron App Feel at Home Everywhere** Kilian Valkhof - Polypane Pacific LMNO	**Socially Conscious Software Open Space** Facilitated Peer Sharing Seacliff D	**Azul** Gil Tene - Azul Pacific BC	**Java on Kubernetes 101 Hands-On Lab!** Pratik Patel - IBM & William Korando - IBM Marina
5:00PM	**BREAK**							
5:25PM	**Microservices Open Space** Facilitated Peer Sharing Seacliff D	**Practical Change Data Streaming Use Cases With Apache Kafka & Debezium** Gunnar Morling - Red Hat Ballroom BC	**Probabilistic programming for software engineers** Michael Tingley - Facebook Pacific DEKJ	**Balancing Priorities: Revenue Generation vs. Revenue Protection** Dominica DeGrandis - Tasktop Bayview AB	**Javascript** Pacific LMNO	**Impact Starts With You** Julia Nguyen - Mailchimp Seacliff ABC	**Checkr** Pacific BC	**Session Continues**
6:15PM	**EXHIBITOR RECEPTION IN EXHIBITOR AREAS ON MARKET STREET LEVEL AND PACIFIC LEVEL**							

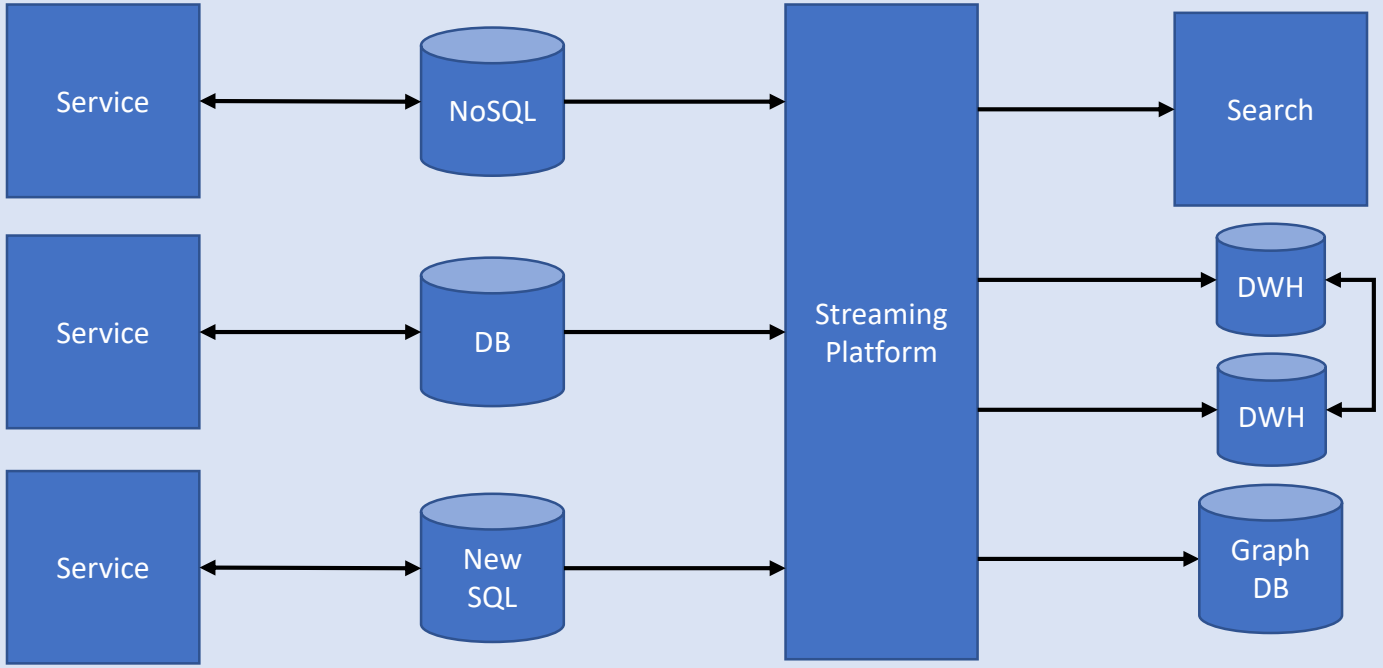
Partial decentralization

- Raw tools are exposed to other engineering teams
- Requires Git, YAML, JSON, pull requests, terraform commands, etc.

Full decentralization

- Polished tools are exposed to everyone
- Security and compliance manage access and policy
- Data engineering manages data tooling and infrastructure
- Everyone manages data pipelines and data warehouses

Modern Data Pipeline



Automated Data Management

Data Catalog | RBAC/IAM/ACL | DLP | ...

Automated Operations

Orchestration | Monitoring | Configuration | ...

Thanks!

(..and we're hiring)