spreedly

Migrating from chaos into consensus

A tale of modernization in three lessons

Scott Traver, Senior Software Architect

spreedly

Vision

The world is better with a diversified, inclusive, payment ecosystem

Mission

Accelerate commerce with an open, secure and flexible payment platform that welcomes all payment participants

Orchestrate and Optimize

Payments Orchestration

The process of expanding the reach of a payment stack by assembling all the necessary components to intelligently manage payments and ensure the capture of every legitimate transaction.

Payments Optimization

Payment Optimization aims to improve individual transaction outcomes by deconstructing and analyzing the components of individual transactions.

What did we **Think** our data needs are?

Application Data

Basic object CRUD that will grow over time but needs to be flexible in what it accepts because we don't know what we don't know

Vaulted Data

The core of securing payment methods for customers, stored for a 'long time'

Transaction Data

Records of all customer activity

What were our **Actual** data needs?

Application Data

Customer controlled data that configures and enables the payments orchestration platform

Vaulted Data

Batches of up to date and secured payment methods for bulk and individual operations including data localization

Transaction Data

Financial transaction data not mixed with an audit log of application events

Systems are collections

Systems are collections

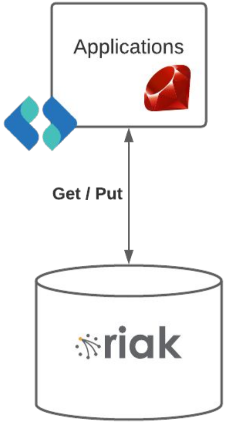
System Level Tech Debt is also a collection over time

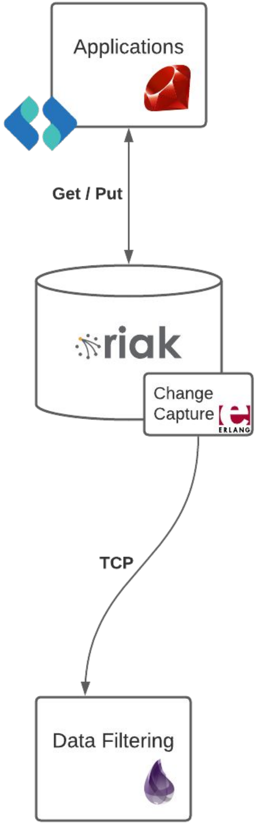
Systems are collections

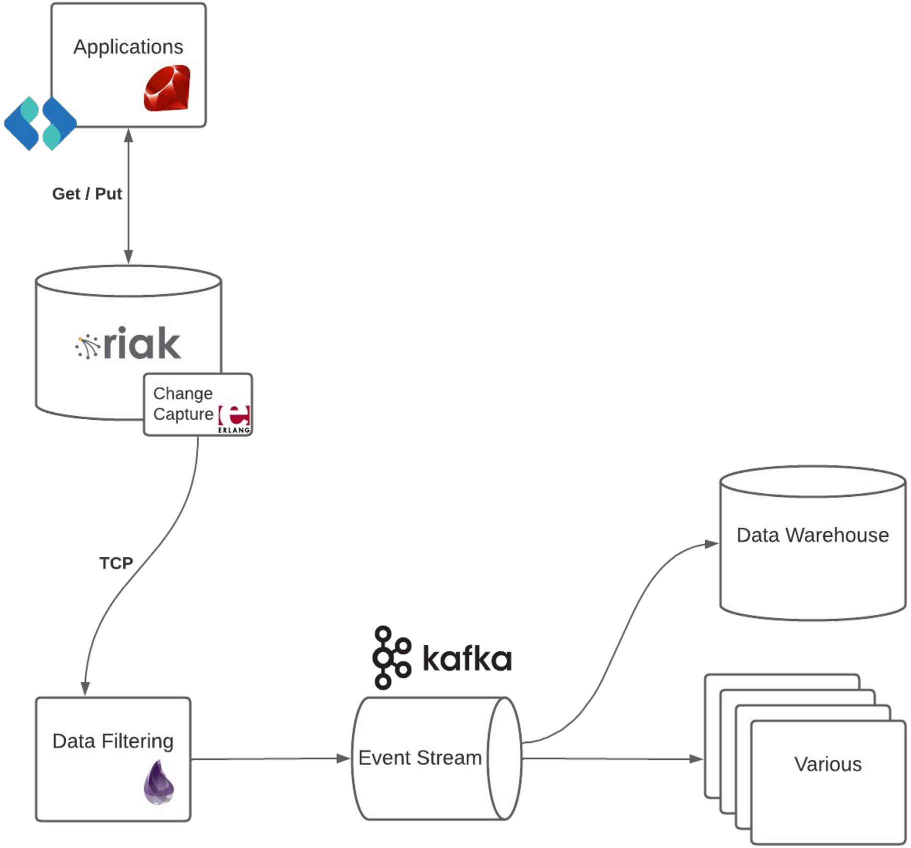
System Level Tech Debt is also a collection over time

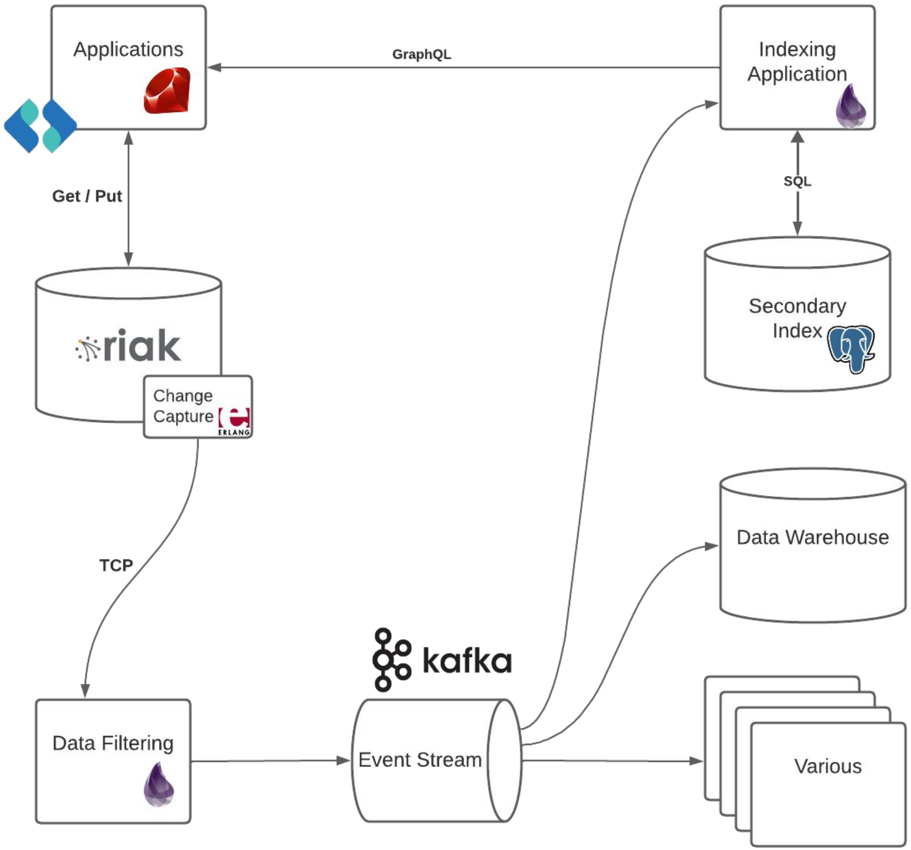
Compounded by the Lindy Effect

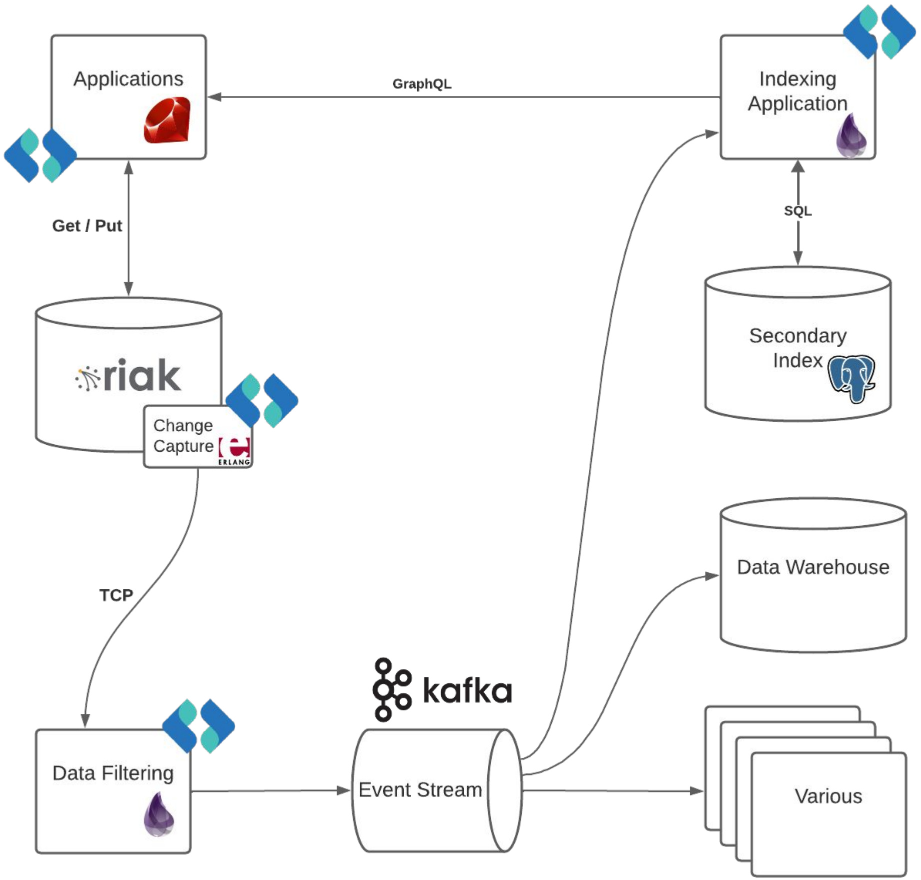
Modernization is a responsibility not a necessity


Applications
Get / Put
riak


Applications
Get / Put
riak
Change Capture
ERLANG
TCP
Data Filtering


Applications
Get / Put
riak
Change Capture
ERLANG
TCP
Data Filtering
kafka
Event Stream
Data Warehouse
Various


Applications
GraphQL
Indexing Application
Get / Put
SQL
riak
Secondary Index
Change Capture
ERLANG
TCP
Data Warehouse
kafka
Data Filtering
Event Stream
Various



Applications

GraphQL

Indexing Application

Get / Put

SQL

riak

Secondary Index

Change Capture

TCP

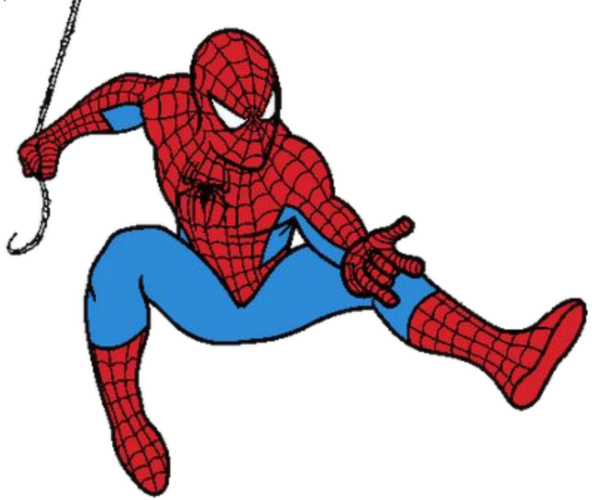
Data Warehouse

kafka

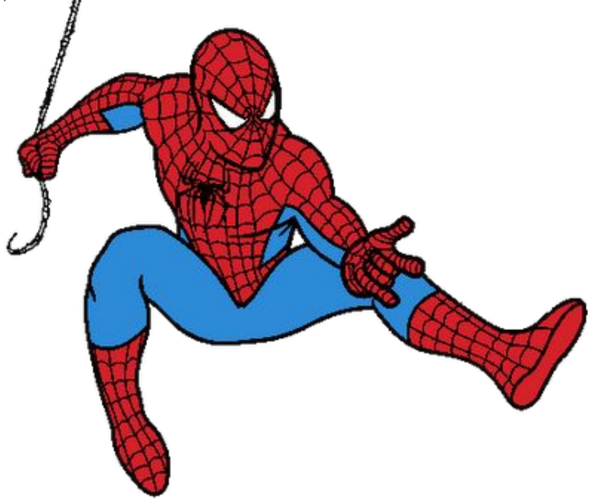
Data Filtering

Event Stream

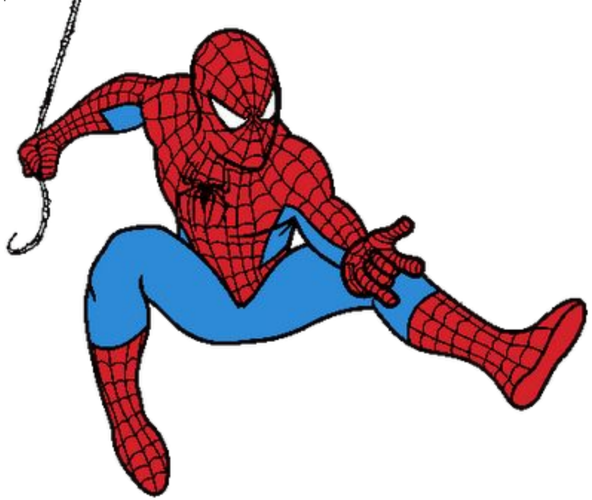
Various



With great power comes...

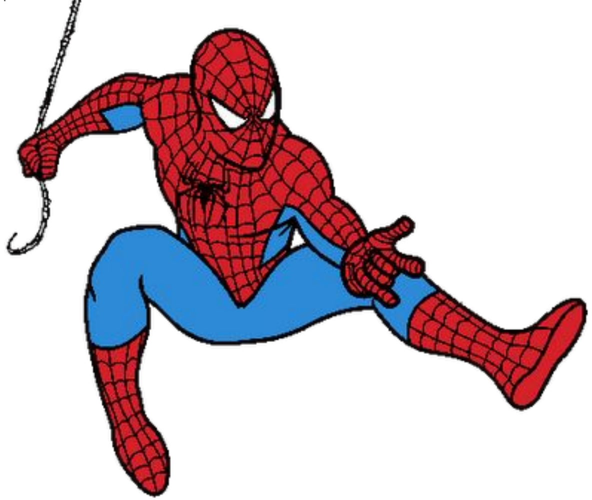


With great power comes...



Great on call rotations, operational costs, failovers, late nights, burnout...

With great power comes…



Great on call rotations, operational costs, failovers, late nights, burnout…

But it doesn’t have to be that way!

Lesson 1

De-risk the migration by taking it slow

Source of Truth Migration Strategy

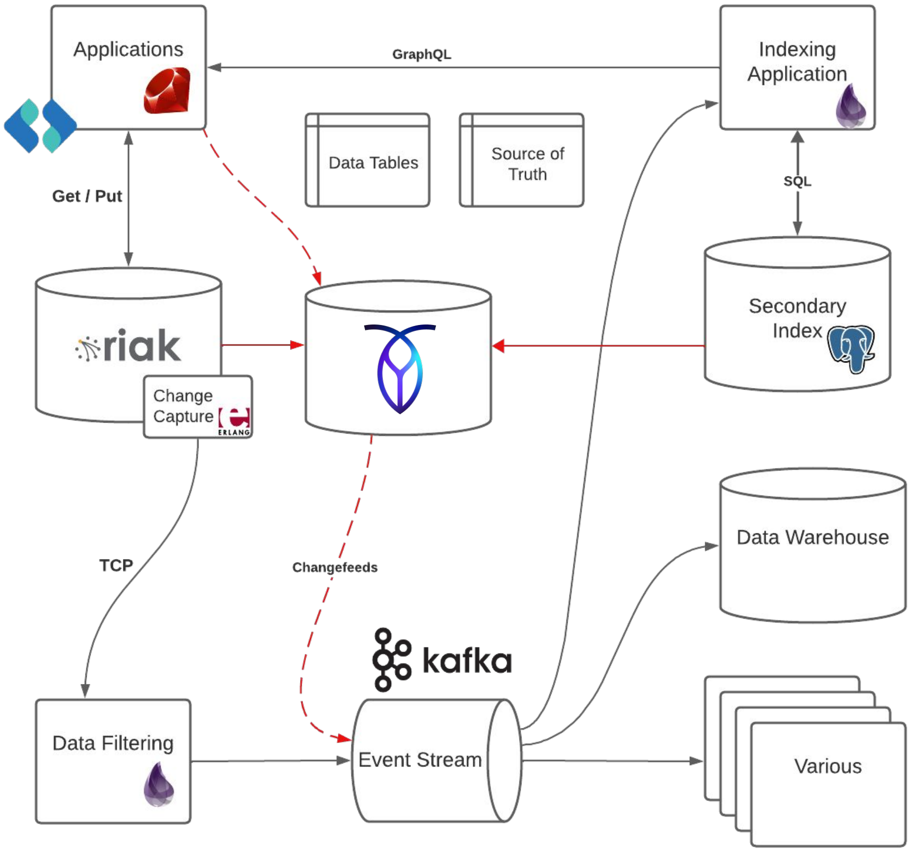
Use a migration strategy that allows forwards and backwards data compatibility for de-risking the migration from downtime

This allowed us to migrate even as granular as data by data which gave us great monitoring and rollback capabilities

Let the database manage the IDs

Not all UUIDs are created equal

Letting cockroach manage the PKs provides several technical advantages


Applications
GraphQL
Indexing Application
Data Tables
Source of Truth
Get / Put
SQL
Secondary Index
riak
Change Capture
ERLANG
Data Warehouse
TCP
Changefeeds
kafka
Data Filtering
Event Stream
Various

Lesson 2

Don’t migrate data which you don’t need to

Know your data

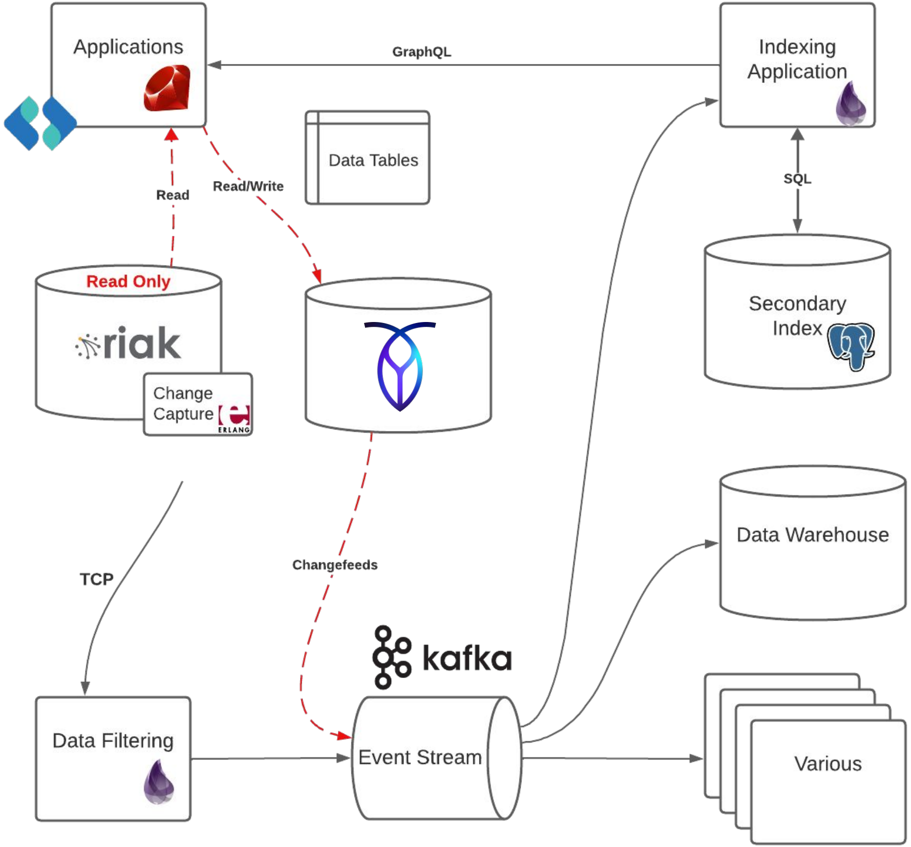
Inefficient data management lead to favoring redaction over deletion which stored lots of unnecessary and unusable records

Read through migrations

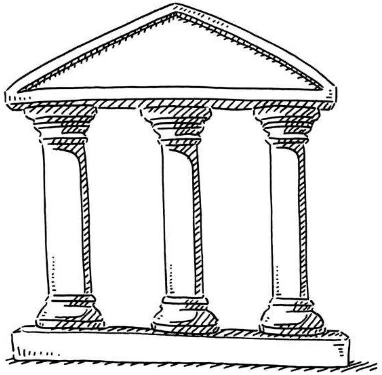
With a read through migration strategy we were able to migrate on demand and lower the risk of the old datastore sooner

Use a datastore managed data retention policy

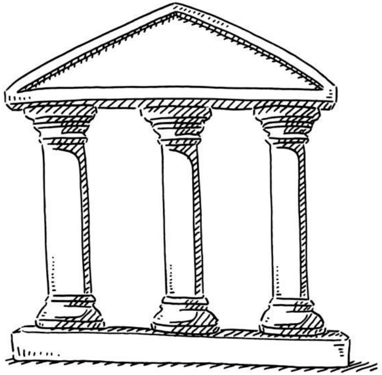
Cockroach gives the power to expire data and flow record of it to secondary systems for appropriate long term storage


Applications
GraphQL
Indexing Application
Data Tables
Read
Read/Write
SQL
Read Only
riak
Change Capture
ERLANG
Secondary Index
Data Warehouse
TCP
Changefeeds
kafka
Data Filtering
Event Stream
Various

The main thing...



The main thing...



Is to keep the main thing the main thing

Lesson 3

Data is an Ecosystem not a database

Single Responsibility = Simplicity

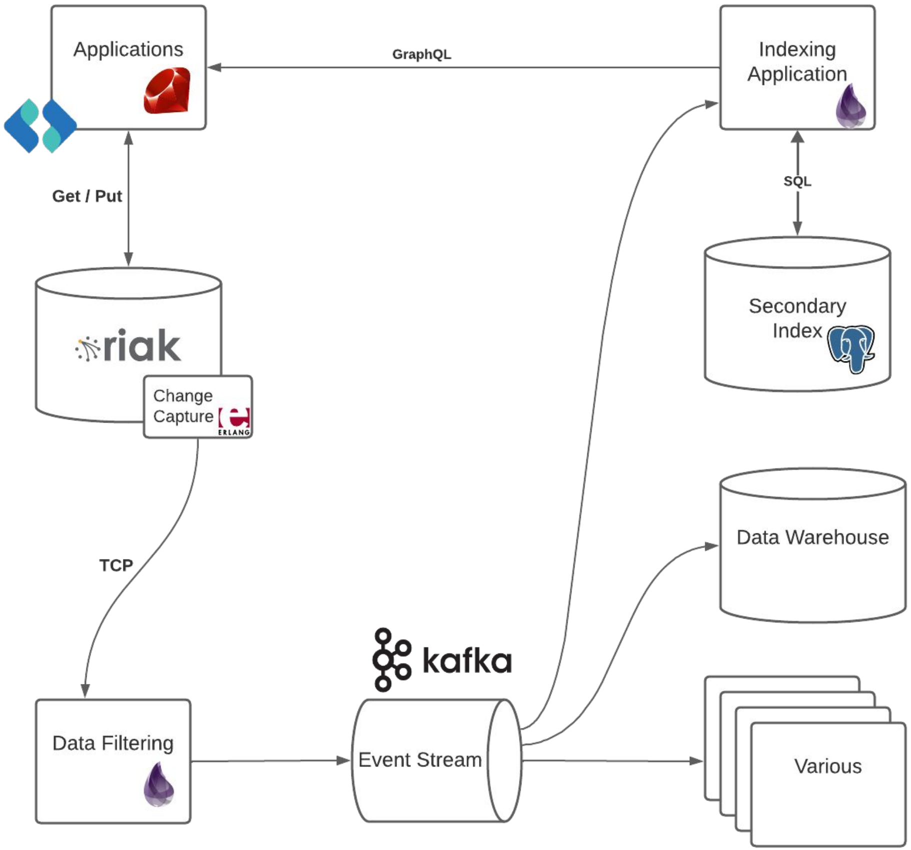
We had self inflicted split brain caused by conflating various business data needs and data types

Symptoms of this included the continual addition of custom tooling further growing complexity and maintenance

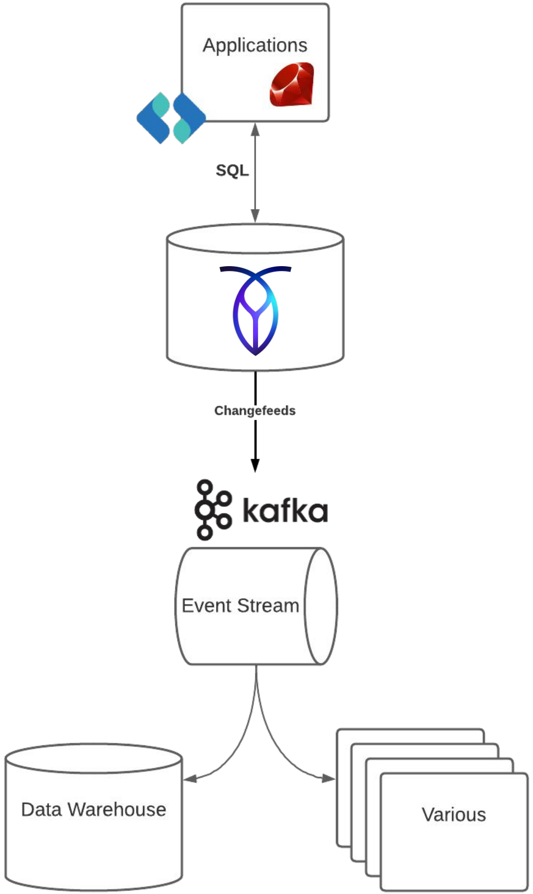
Use integration tools to unify the ecosystem

Changefeeds admit that there can exist a larger data ecosystem outside of your primary business application

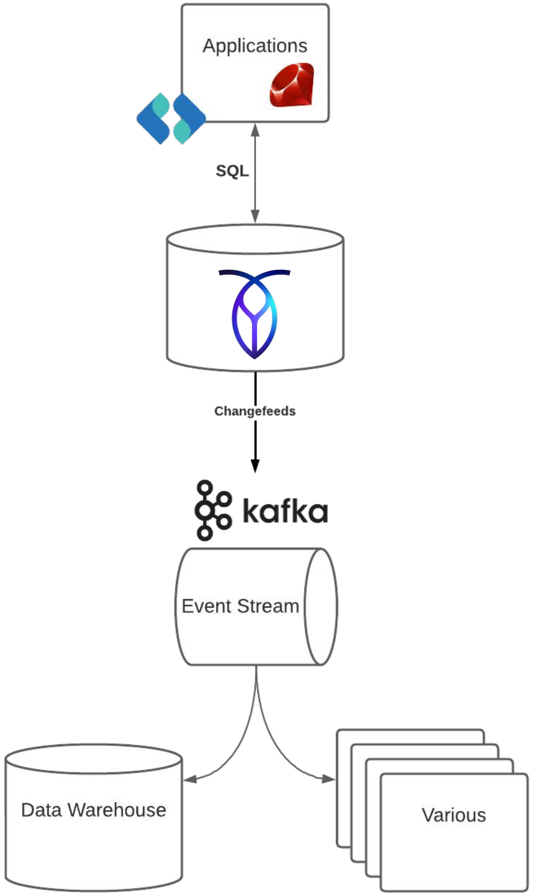
Before


Applications
GraphQL
Indexing Application
Get / Put
SQL
riak
Secondary Index
Change Capture
ERLANG
TCP
Data Warehouse
kafka
Data Filtering
Event Stream
Various

After


Applications
SQL
Changefeeds
kafka
Event Stream
Data Warehouse
Various

Migration Project



✗

- Replace outdated datastore
- Remove data cycle
- Remove 3 custom codebases, 2 databases, 3 protocols
- Reduce unusable data bloat

✓

- Cloud first data storage approach
- Common data interface and standard modern record adapter for applications
- Enable multi-region localization with HA
- Enable data retention policy
- Enable batch and bulk workloads

Modernization is not a spectator sport

Thank You

Scott Traver

scott@spreedly.com

https://www.spreedly.com/