sumup®

How SumUp Built a Scalable, Performant Payment System on CockroachDB

Anton Antonov, Engineering Manager

RoachFest25

sumup®

A world where everyone can build a thriving business.

2024 Highlights

- ***1B+** transactions per year*
- ***2k** payments/minute average*
- ***10k+** payments/minute peak*
- ***4M+** merchants in **36** markets*

Payments tribe: Payments processing

The Payments platform for card payments & SumUp’s product ecosystem

Sofia (Bulgaria) office-based

Tech Radar Highlights

Go

CockroachDB

AWS RDS (PostgreSQL)

Kafka

AWS EKS

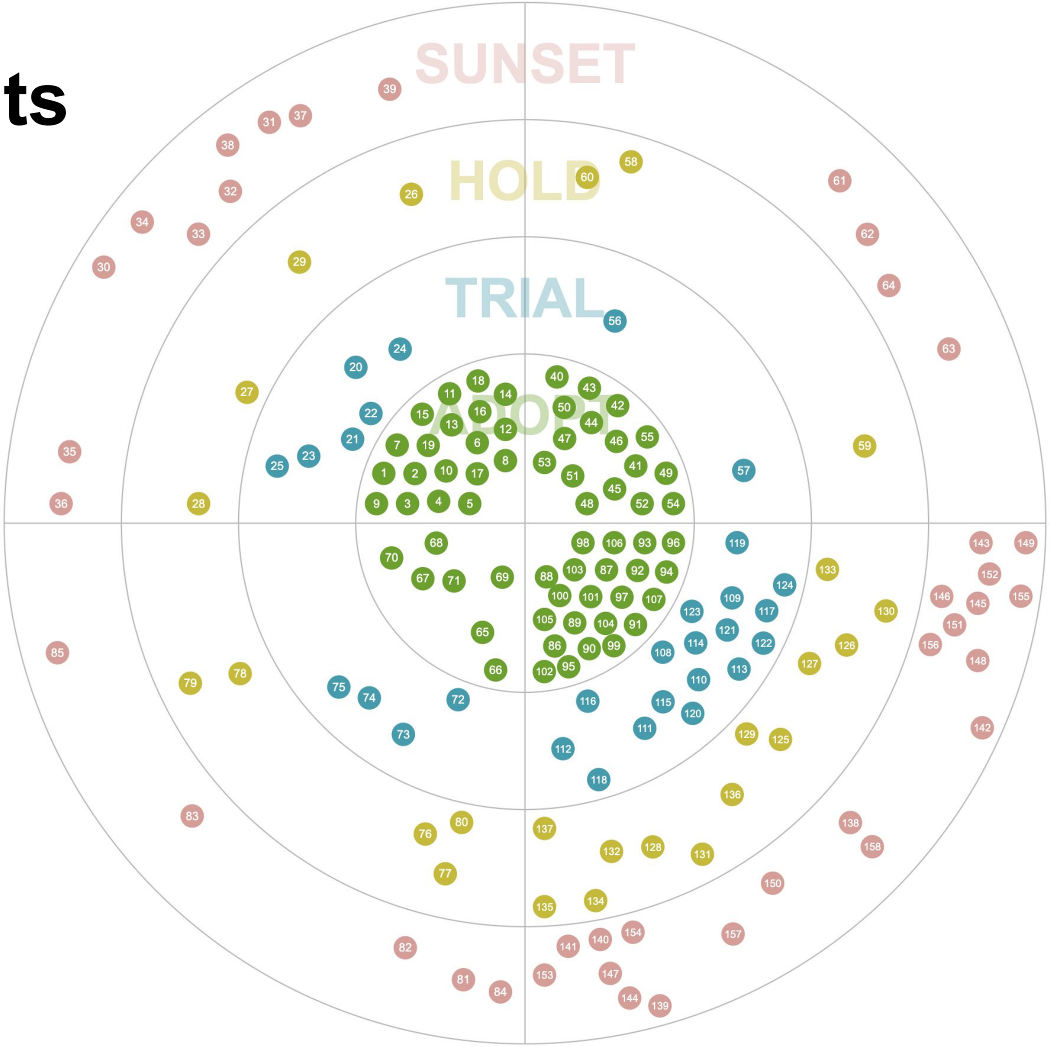
ArgoCD

Terraform

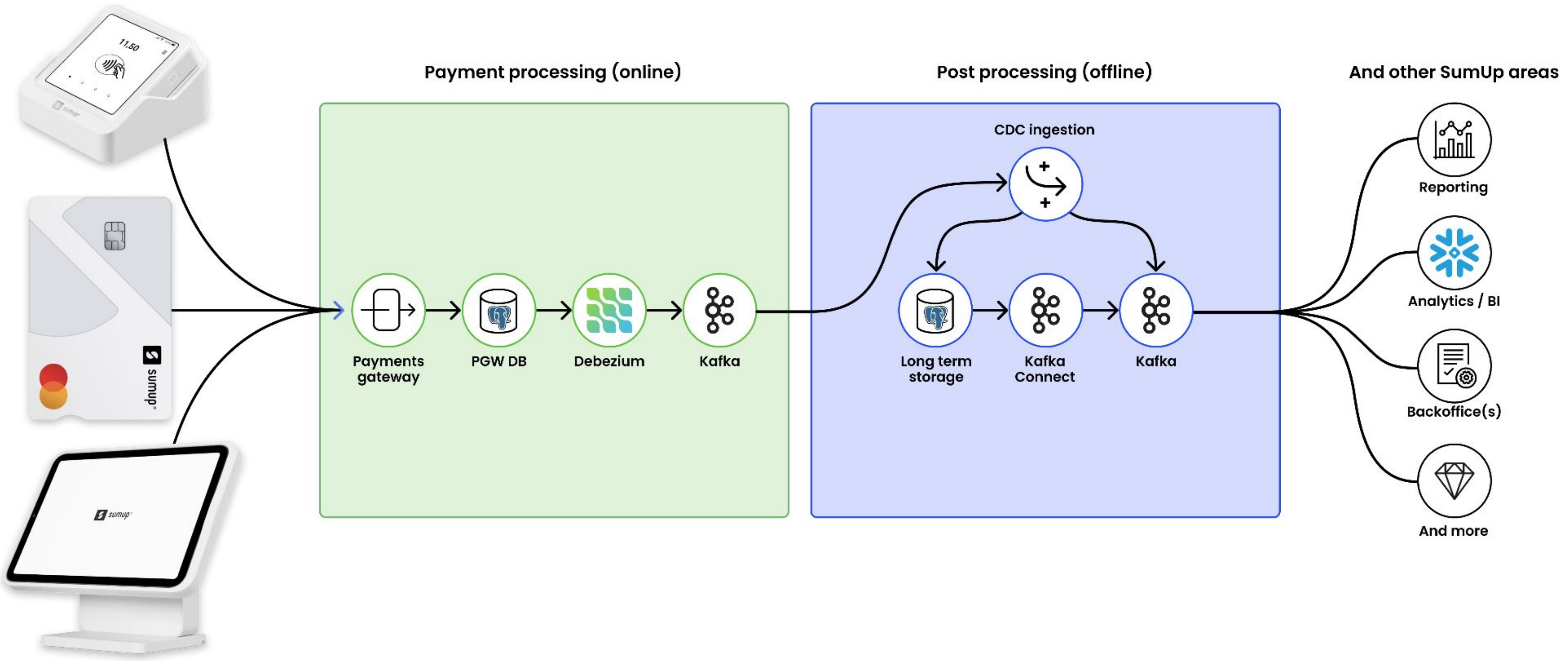
Snowflake

Elixir

And many more….



Where we started



The AWS RDS problems - Scale & Performance

Vertical scaling

No multi-region writes

Hotspots with write-heavy workloads

Single primary write node architecture

Single point of failure

Row contention

Index contention

We need
Horizontal Scalability

Multi-node writes

The AWS RDS problems - Availability & Maintenance

No native online schema changes

No native CDC

Again, single primary write node architecture

(Complex & Third-party) Change-Data-Capture via Debezium

Operational overhead

Maintenance downtimes

Reliance on third-party tools

We need
Distributed & native CDC ;
Highly-available cluster

And then we architected for the foreseeable future

- **Deep** observability
- Zero-downtime upgrades
- ACID
- SQL
- Online backups
- Provisionable via IaaC
- Automatic Rebalancing on Failure

pci DSS COMPLIANT

(RAFT)

ACID properties provide such effective abstraction that we typically don't have to recall their specific meaning.

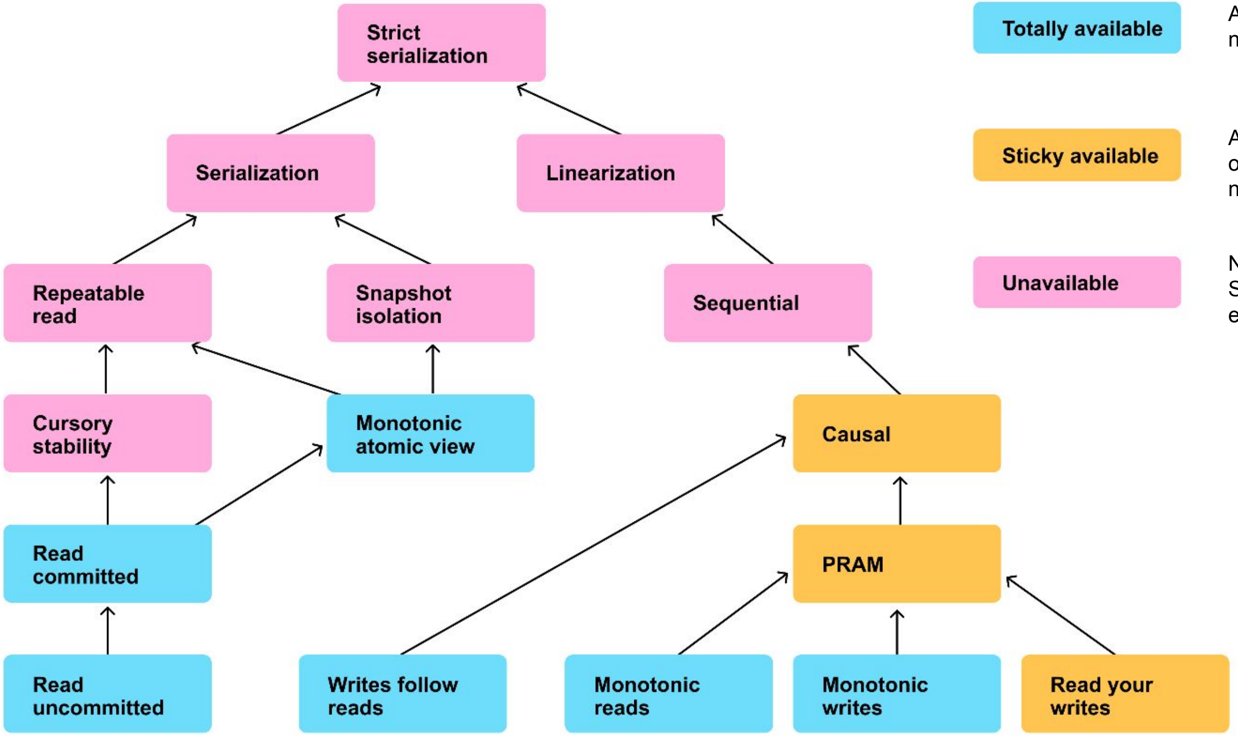
ACID properties provide such effective abstraction that we typically don't have to recall their specific meaning.

Rule of the extra 9

A service cannot be more **available** than the intersection of all its critical **dependencies.** If your service aims to offer 99.99 percent availability, then all of your critical dependencies must be **significantly** more than 99.99 percent available.

Ref: The Calculus of Service Availability (https://queue.acm.org/detail.cfm?id=3096459)

Concurrent systems. Consistency models.



🟦 **Totally available** Available on every non-faulty node, even when the network is completely down

🟨 **Sticky available** Available on every non-faulty node, so long as clients only talk to the same servers, instead of switching to new ones.

🟪 **Unavailable** Not available during some types of network failures. Some or all nodes must pause operations in order to ensure safety.

For our use-case,

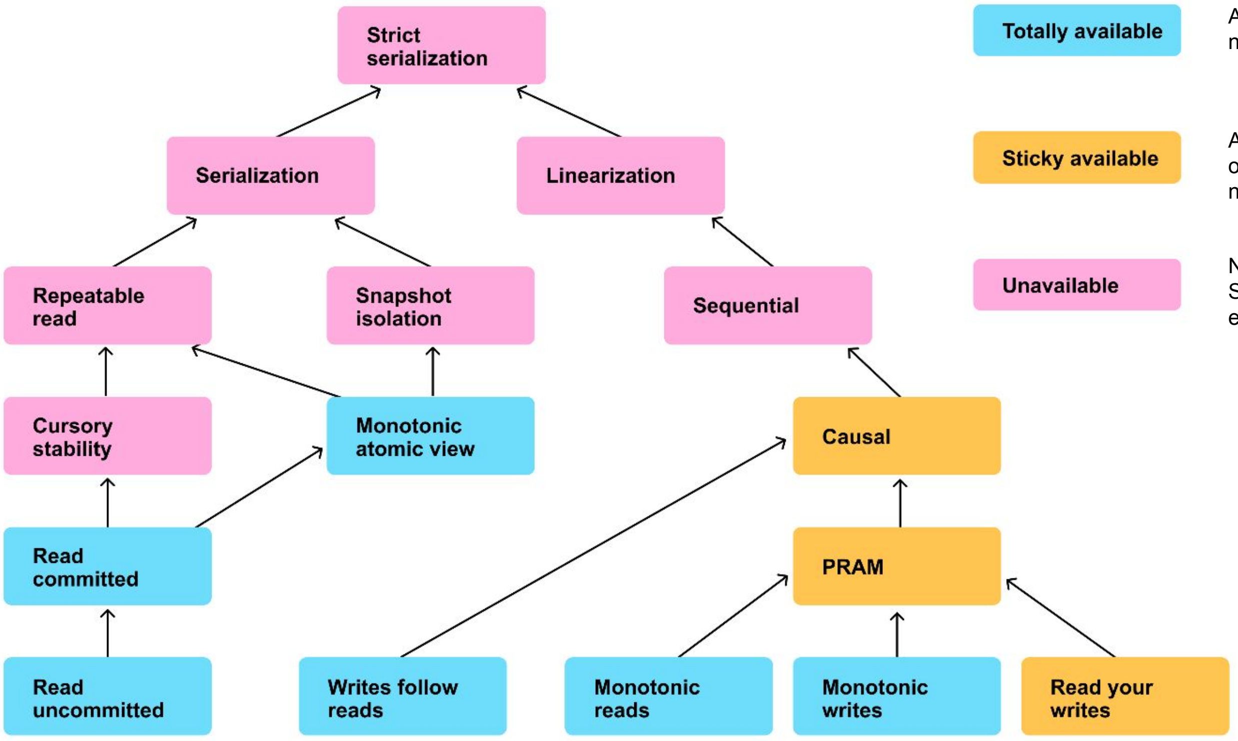
Strict Serializable consistency (preferred)

or

Linearizable without transactions.

https://jepsen.io/consistency

Concurrent systems. Consistency models.



Strict serialization

Serialization

Linearization

Repeatable read

Snapshot isolation

Sequential

Cursory stability

Monotonic atomic view

Causal

Read committed

PRAM

Read uncommitted

Writes follow reads

Monotonic reads

Monotonic writes

Read your writes

🟦 Totally available	Available on every non-faulty node, even when the network is completely down
🟨 Sticky available	Available on every non-faulty node, so long as clients only talk to the same servers, instead of switching to new ones.
🟪 Unavailable	Not available during some types of network failures. Some or all nodes must pause operations in order to ensure safety.

For our use-case,

Strict Serializable consistency (preferred)

or

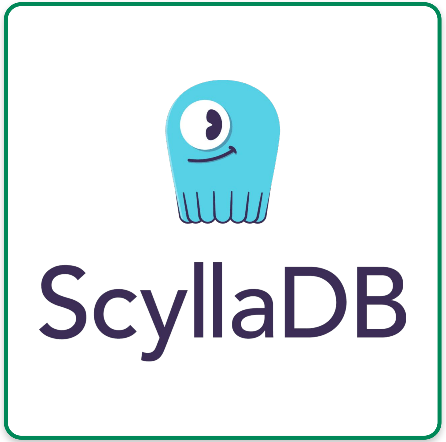
Linearizable without transactions.

https://jepsen.io/consistency

The candidates

CockroachDB

yugabyteDB



ScyllaDB

MongoDB®

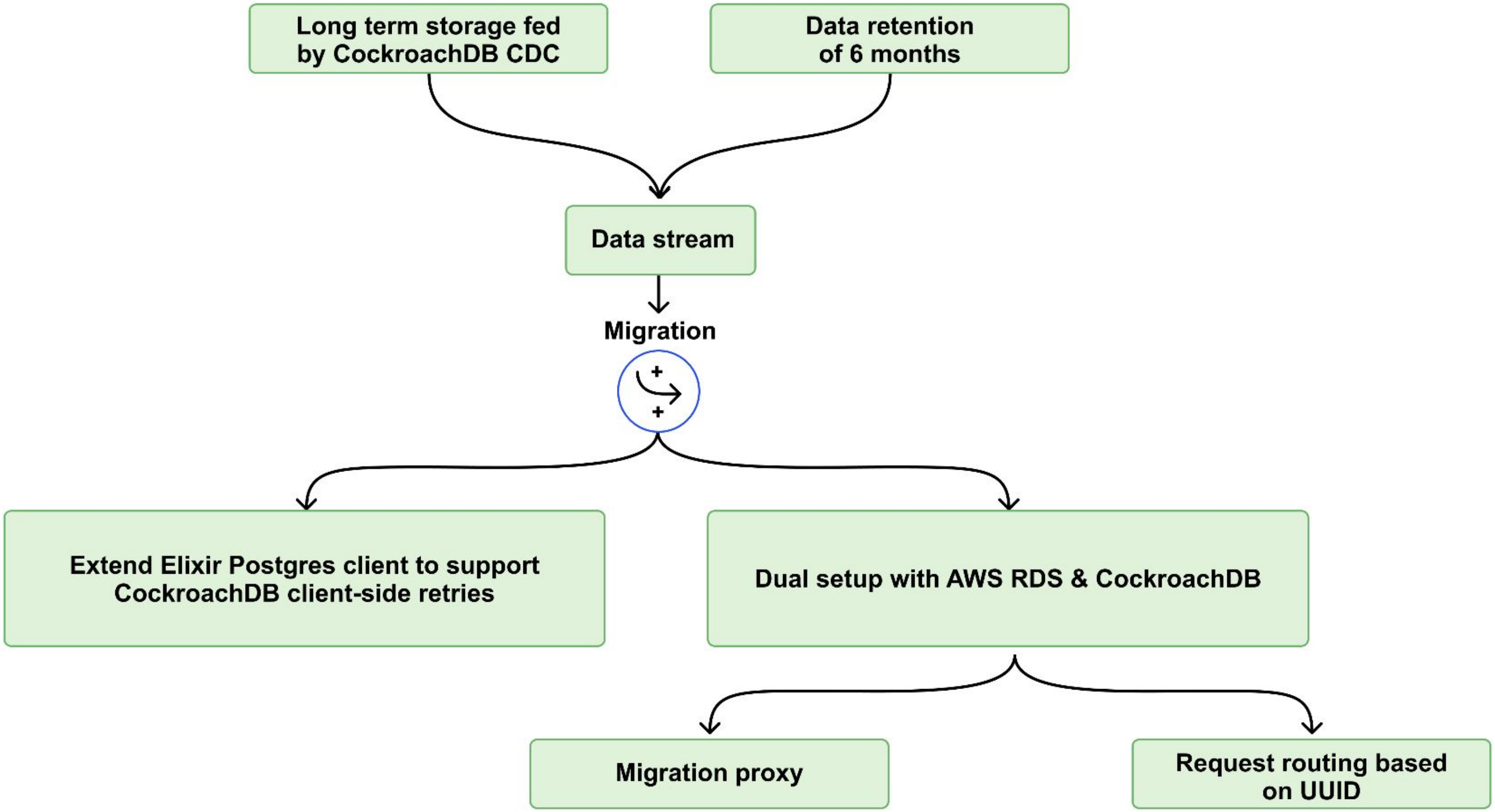
Candidates

	CockroachDB Cloud (Dedicated)	YugaByte Managed	ScyllaDB Cloud	MongoDB Atlas
PCI-DSS Compliant	Yes	Yes	Yes	Yes
Consistency model	Strict Serializable	Serializable	Eventual	Linearizable (single-document ops)
SQL / Schema	Yes and full online schema changes support	Yes, partial online schema changes support	Yes, kind of	No, document model
Native CDC?	Yes, distributed	No, Debezium needed.	Yes, but local-per-node	Yes
Multi-region Writes	Yes, truly global	Limited	Yes	Yes, with caveats*

Candidates

	CockroachDB Cloud (Dedicated)	YugaByte Managed	ScyllaDB Cloud	MongoDB Atlas
PCI-DSS Compliant	Yes	Yes	Yes	Yes
Consistency model	Strict Serializable	Serializable	Eventual	Linearizable (single-document ops)
SQL / Schema	Yes and full online schema changes support	Yes, partial online schema changes support	Yes, kind of	No, document model
Native CDC?	Yes, distributed	No, Debezium needed.	Yes, but local-per-node	Yes
Multi-region Writes	Yes, truly global	Limited	Yes	Yes, with caveats*

Migration plan



Long term storage fed by CockroachDB CDC

Data retention of 6 months

Data stream

Migration

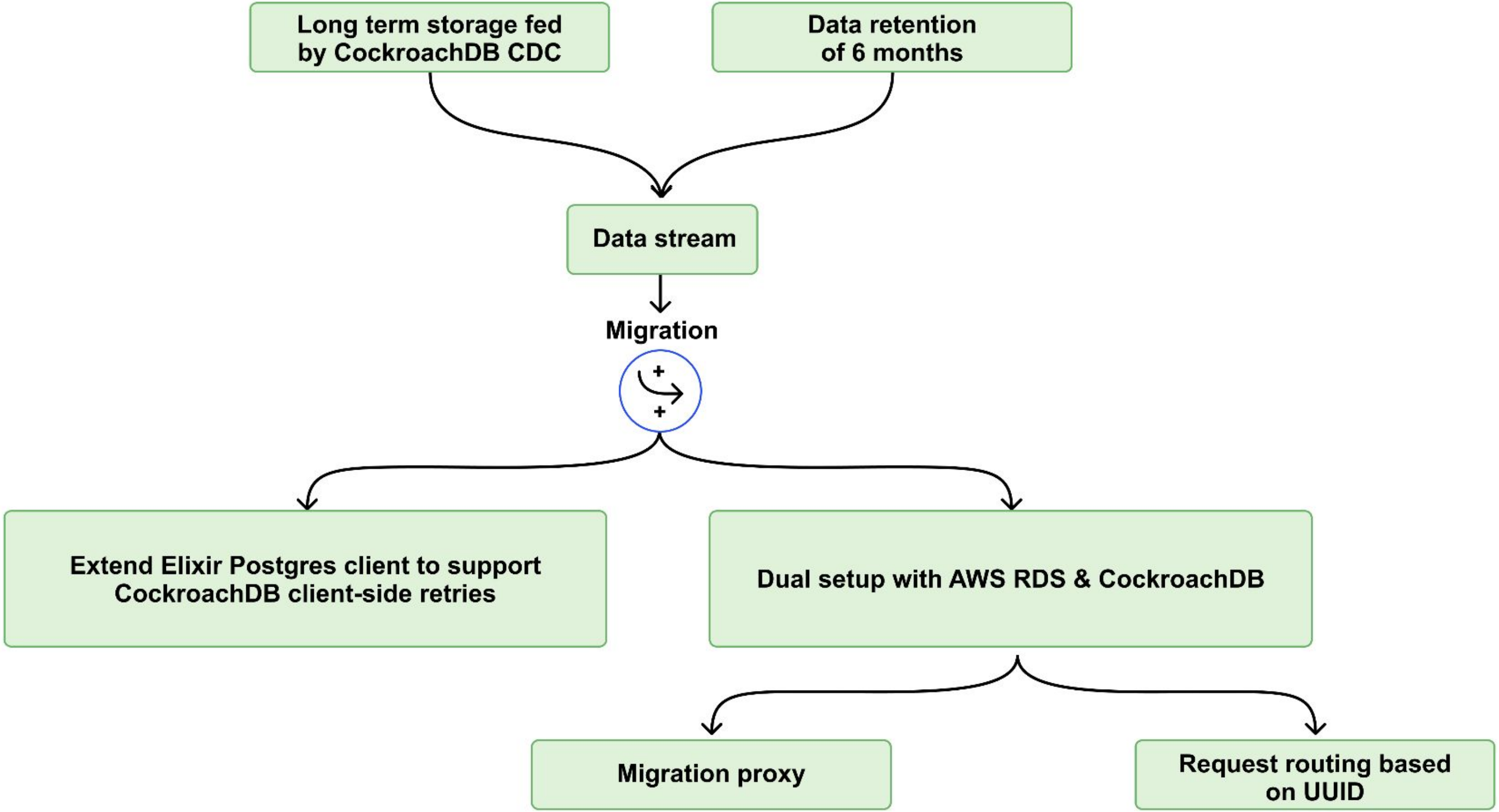
Extend Elixir Postgres client to support CockroachDB client-side retries

Dual setup with AWS RDS & CockroachDB

Migration proxy

Request routing based on UUID

Migration plan



Long term storage fed by CockroachDB CDC

Data retention of 6 months

Data stream

Migration

Extend Elixir Postgres client to support CockroachDB client-side retries

Dual setup with AWS RDS & CockroachDB

Migration proxy

Request routing based on UUID

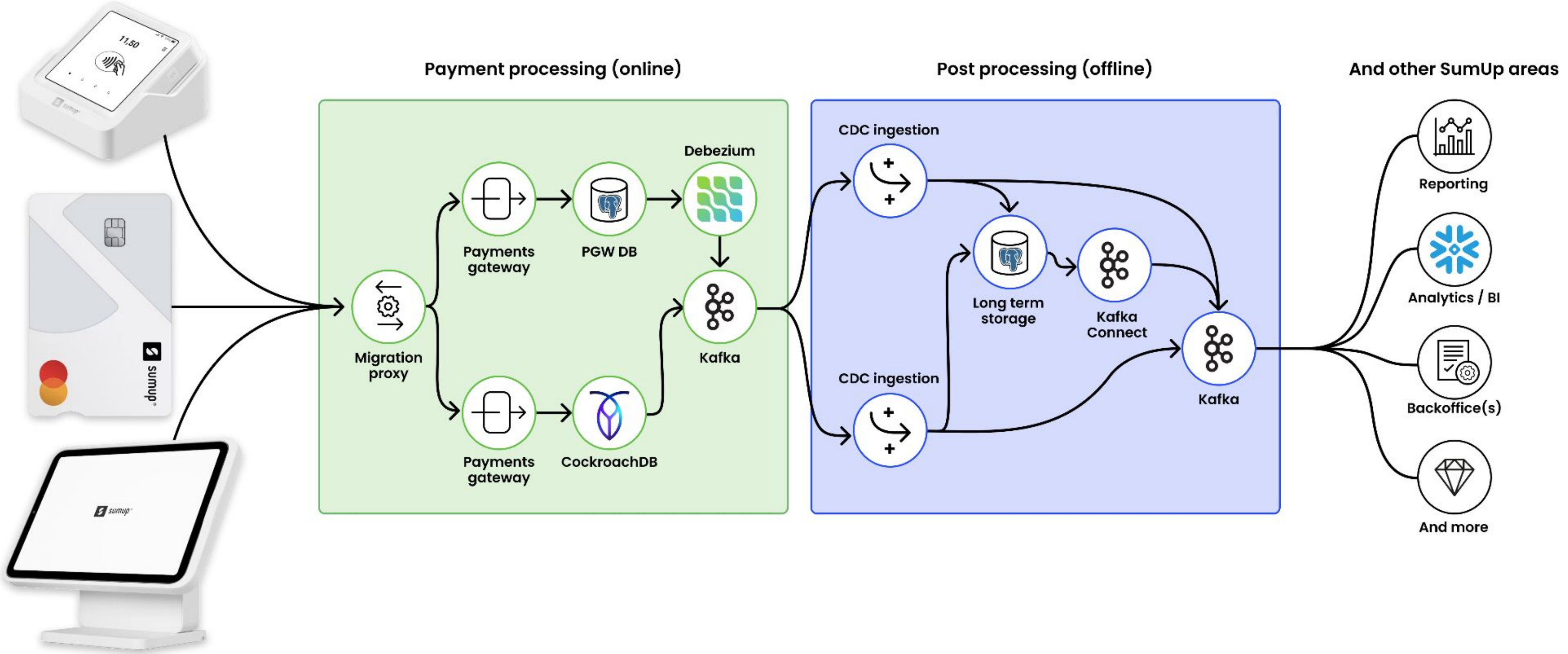
Migration plan

	UUIDv6 (Postgres)	UUIDv4 (CRDB)
Monotonic	Yes	No (Random)
Deterministic	No	No
Index-Friendly	Yes (B-Trees)	No, different implementation in CRDB
CRDB Performance	Hotspot risk	Best practice
Use Case fit	Great for OTLP in Postgres	Great for distributed SQL

Migration plan

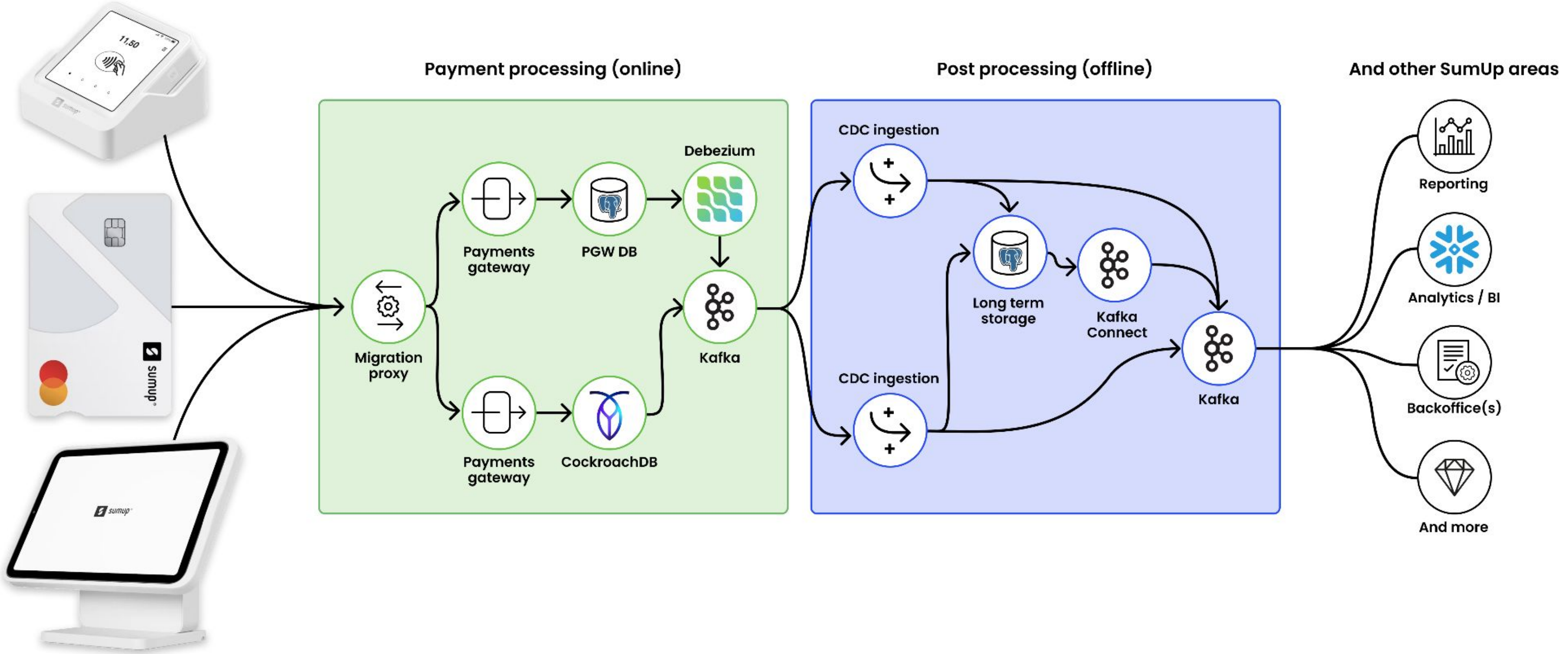
	UUIDv6 (Postgres)	UUIDv4 (CRDB)
Monotonic	Yes	No (Random)
Deterministic	No	No
Index-Friendly	Yes (B-Trees)	No, different implementation in CRDB
CRDB Performance	Hotspot risk	Best practice
Use Case fit	Great for OTLP in Postgres	Great for distributed SQL

Migration

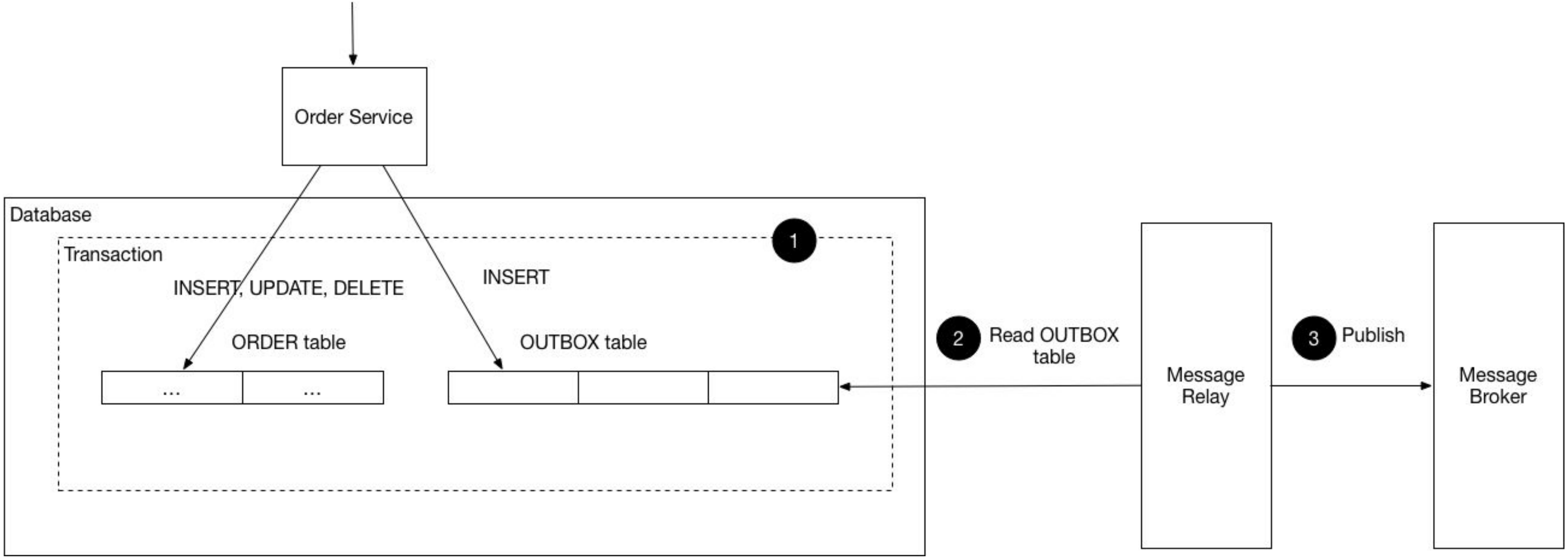


11,50
sumup
Payment processing (online)
Migration proxy
Payments gateway
PGW DB
Debezium
Kafka
Payments gateway
CockroachDB
Post processing (offline)
CDC ingestion
Long term storage
Kafka Connect
Kafka
CDC ingestion
And other SumUp areas
Reporting
Analytics / BI
Backoffice(s)
And more

Migration

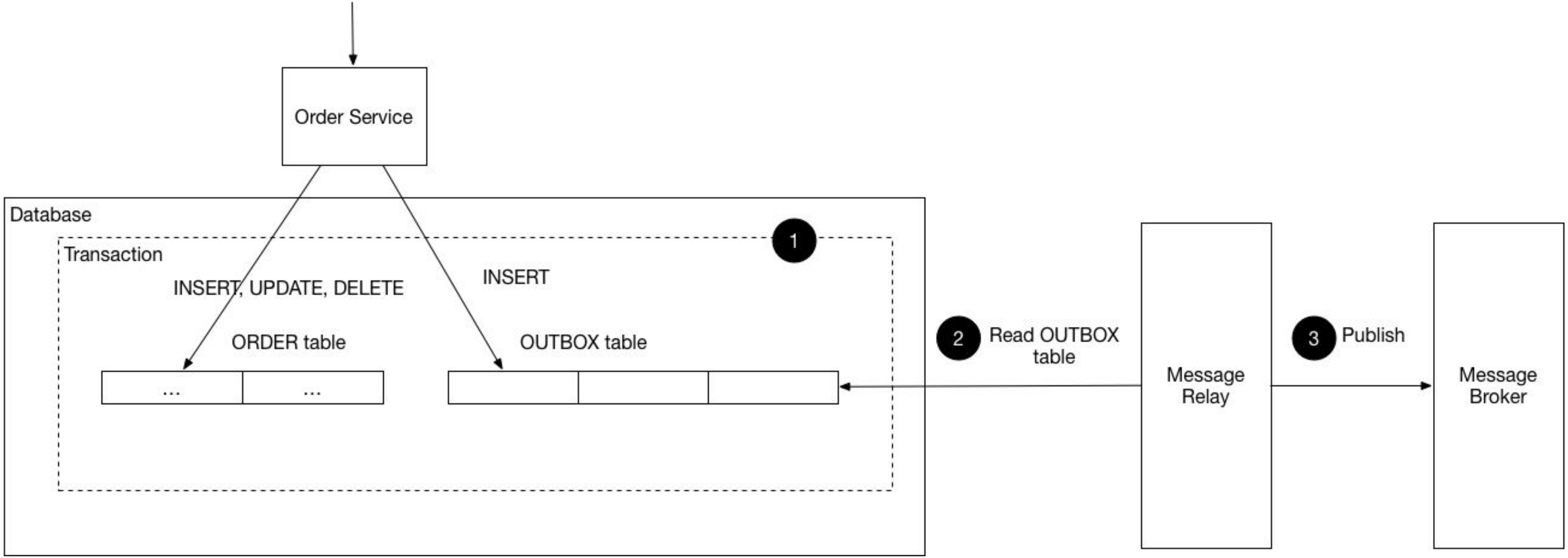

Payment processing (online)
Post processing (offline)
And other SumUp areas
11,50
sumup
Migration proxy
Payments gateway
PGW DB
Debezium
Kafka
Payments gateway
CockroachDB
CDC ingestion
Long term storage
Kafka Connect
CDC ingestion
Kafka
Reporting
Analytics / BI
Backoffice(s)
And more

Migration - Outbox table pattern



Ref: Chris Richardson’s https://microservices.io/patterns/data/transactional-outbox.html

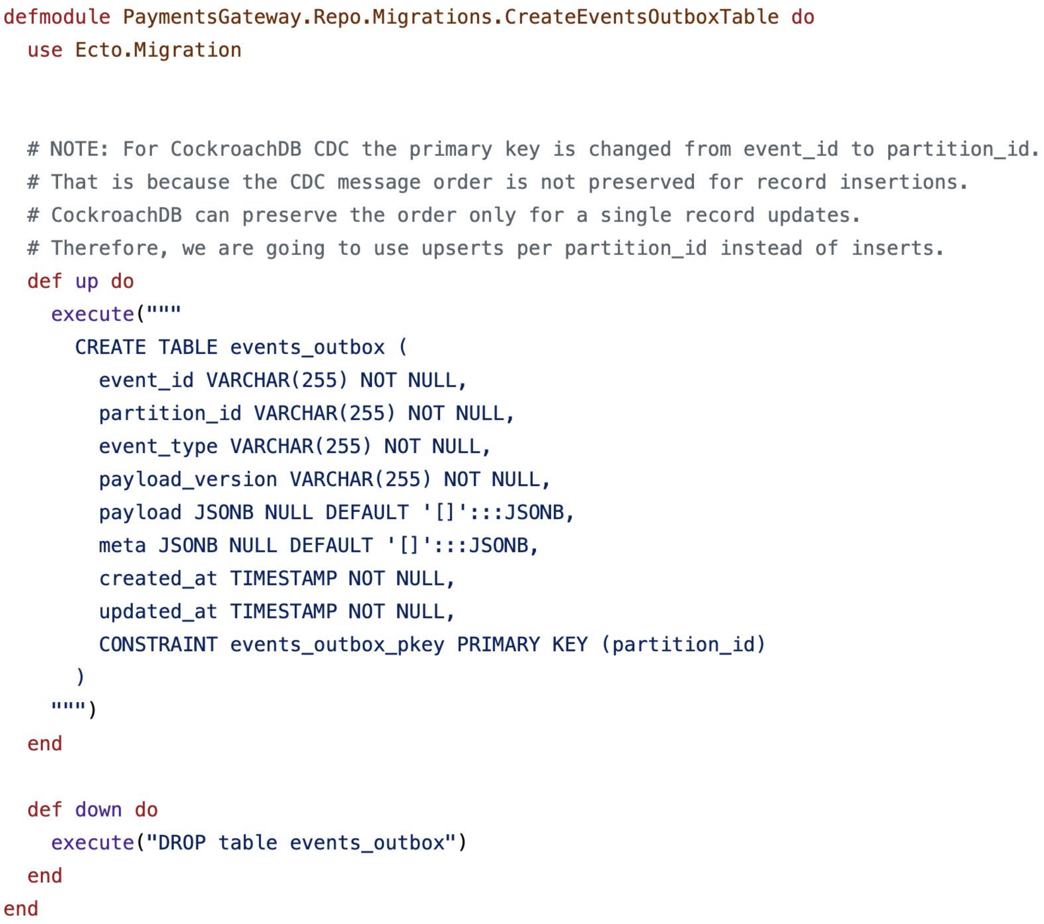
Migration - Outbox table pattern



Ref: Chris Richardson’s https://microservices.io/patterns/data/transactional-outbox.html

Migration

Migrations via Elixir Phoenix framework migrations 😥



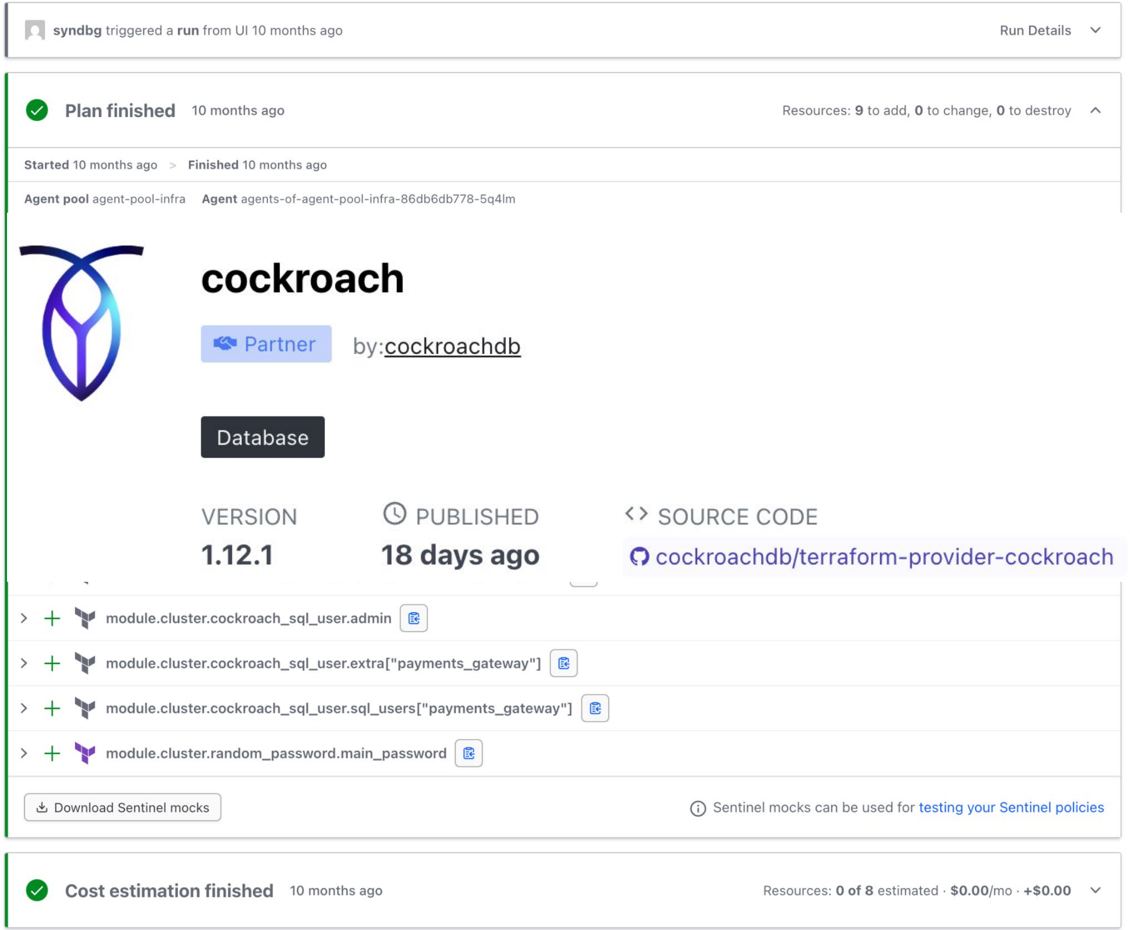
```
defmodule PaymentsGateway.Repo.Migrations.CreateEventsOutboxTable do
  use Ecto.Migration


  # NOTE: For CockroachDB CDC the primary key is changed from event_id to partition_id.
  # That is because the CDC message order is not preserved for record insertions.
  # CockroachDB can preserve the order only for a single record updates.
  # Therefore, we are going to use upserts per partition_id instead of inserts.
  def up do
    execute("""
      CREATE TABLE events_outbox (
        event_id VARCHAR(255) NOT NULL,
        partition_id VARCHAR(255) NOT NULL,
        event_type VARCHAR(255) NOT NULL,
        payload_version VARCHAR(255) NOT NULL,
        payload JSONB NULL DEFAULT '[]':::JSONB,
        meta JSONB NULL DEFAULT '[]':::JSONB,
        created_at TIMESTAMP NOT NULL,
        updated_at TIMESTAMP NOT NULL,
        CONSTRAINT events_outbox_pkey PRIMARY KEY (partition_id)
      )
    """)
  end

  def down do
    execute("DROP table events_outbox")
  end
end
```

Migration

IaaC via Cockroach Labs
Cloud Terraform Provider
& HashiCorp Cloud Platform



syndbg triggered a **run** from UI 10 months ago

Plan finished 10 months ago

Resources: **9** to add, **0** to change, **0** to destroy

Started 10 months ago > **Finished** 10 months ago

Agent pool agent-pool-infra **Agent** agents-of-agent-pool-infra-86db6db778-5q4lm

cockroach

by:cockroachdb

VERSION 1.12.1

PUBLISHED 18 days ago

SOURCE CODE cockroachdb/terraform-provider-cockroach

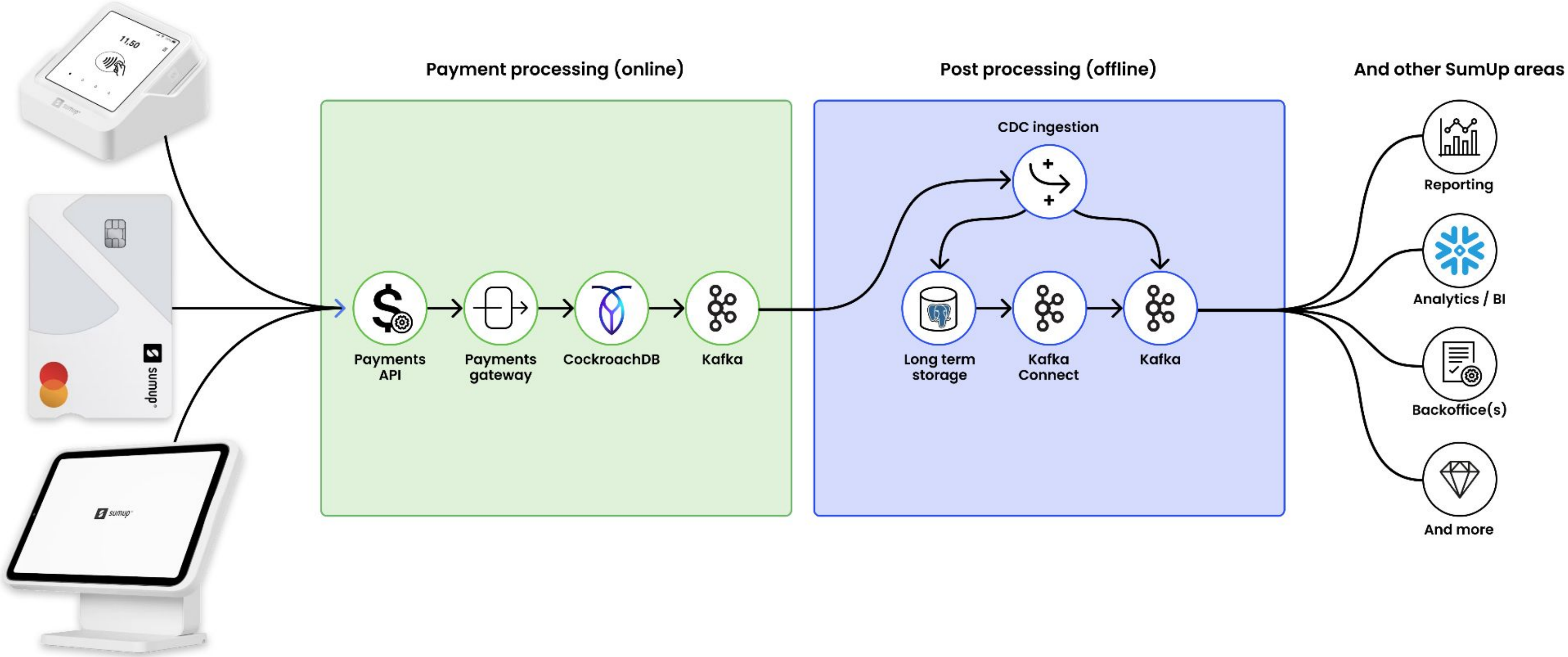
- + module.cluster.cockroach_sql_user.admin
- + module.cluster.cockroach_sql_user.extra["payments_gateway"]
- + module.cluster.cockroach_sql_user.sql_users["payments_gateway"]
- + module.cluster.random_password.main_password

Sentinel mocks can be used for testing your Sentinel policies

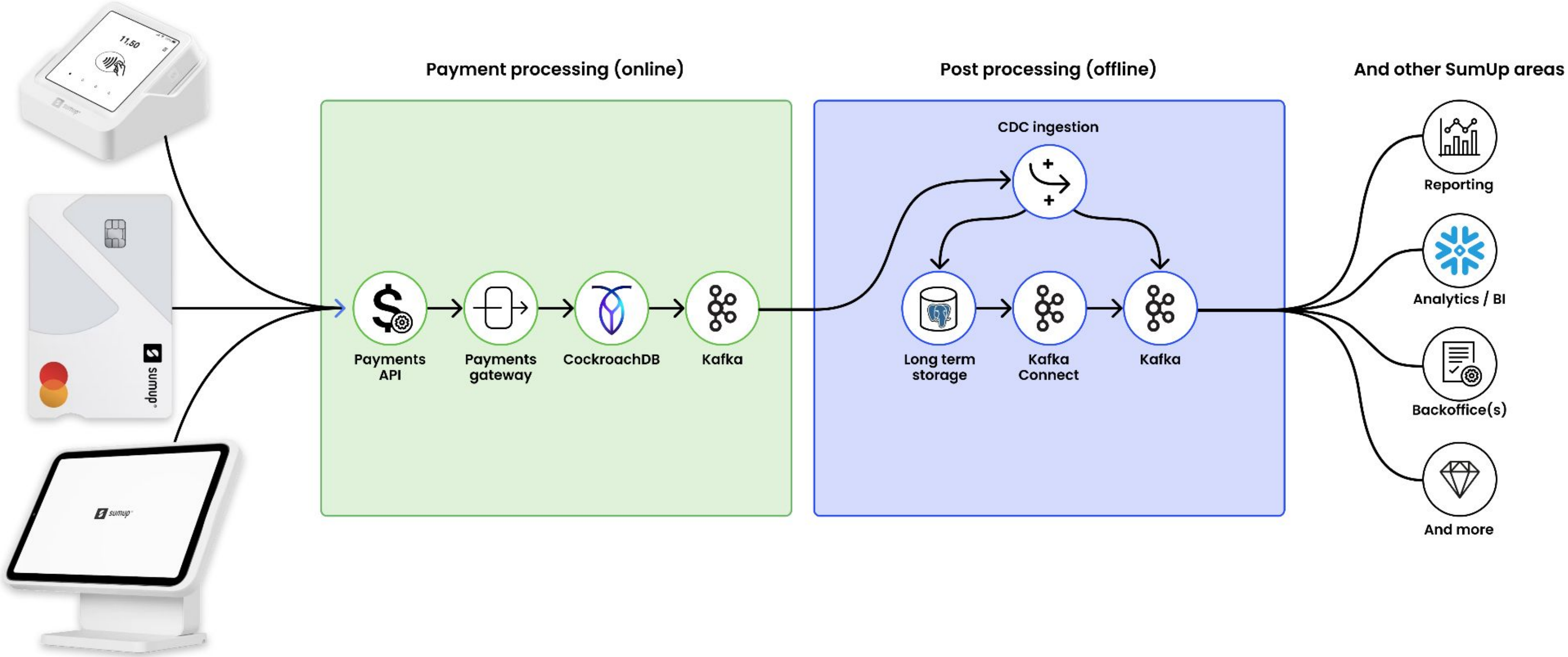
Cost estimation finished 10 months ago

Resources: **0 of 8** estimated · **$0.00**/mo · **+$0.00**

Now


Payment processing (online)
Post processing (offline)
And other SumUp areas
Payments API
Payments gateway
CockroachDB
Kafka
CDC ingestion
Long term storage
Kafka Connect
Kafka
Reporting
Analytics / BI
Backoffice(s)
And more
11,50
sumup

Now


Payment processing (online)
Post processing (offline)
And other SumUp areas
Payments API
Payments gateway
CockroachDB
Kafka
CDC ingestion
Long term storage
Kafka Connect
Kafka
Reporting
Analytics / BI
Backoffice(s)
And more

The wins, recap

More SumUp areas utilizing CRDB

Identity

Payments Reporting

Payments Ledger

POS

And a few more

Multi-cloud

Multi-region global clusters

First-party Golang ecosystem

No data loss

Zero-downtime upgrades

Self-healing

Auto-rebalancing

Thanks!

Cheers from our London office.
Scan the QR code for our open roles!

SumUp. It's possible