Please

Ask questions through the app

📣 *Rate Session*

Thank you!

Modern Stream Processing with Apache Flink®

Till Rohrmann

dataArtisans

GOTO Berlin 2017

dataArtisans

Original creators of
Apache Flink®

dA
PLATFORM

dA Platform 2
Open Source Apache Flink
+ dA Application Manager

What changes faster? Data or Query?

Data changes slowly compared to fast changing queries

ad-hoc queries, data exploration, ML training and (hyper) parameter tuning

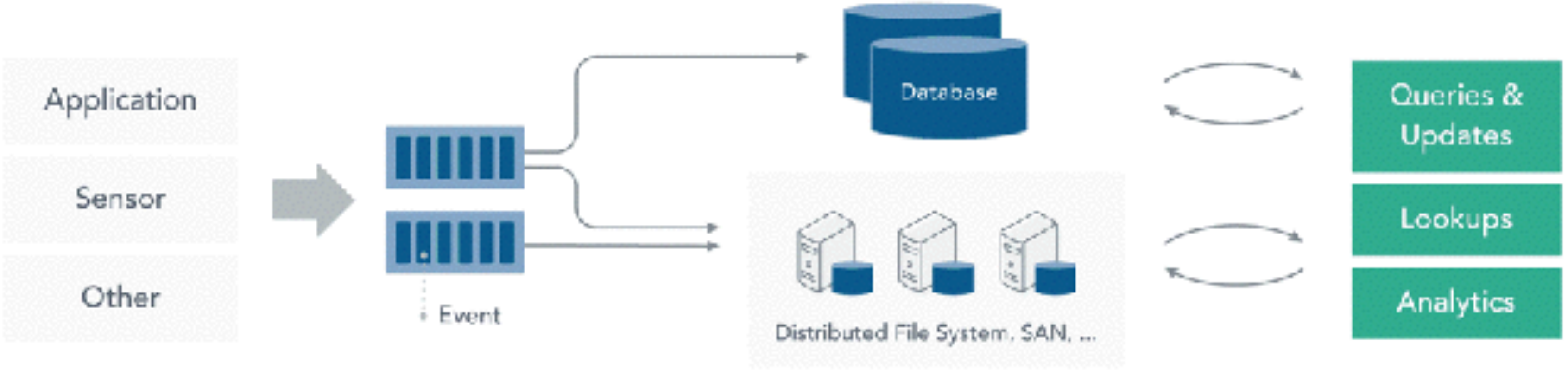
Batch Processing Use Case

Data changes fast application logic is long-lived

continuous applications, data pipelines, standing queries, anomaly detection, ML evaluation, …

Stream Processing Use Case

Batch Processing

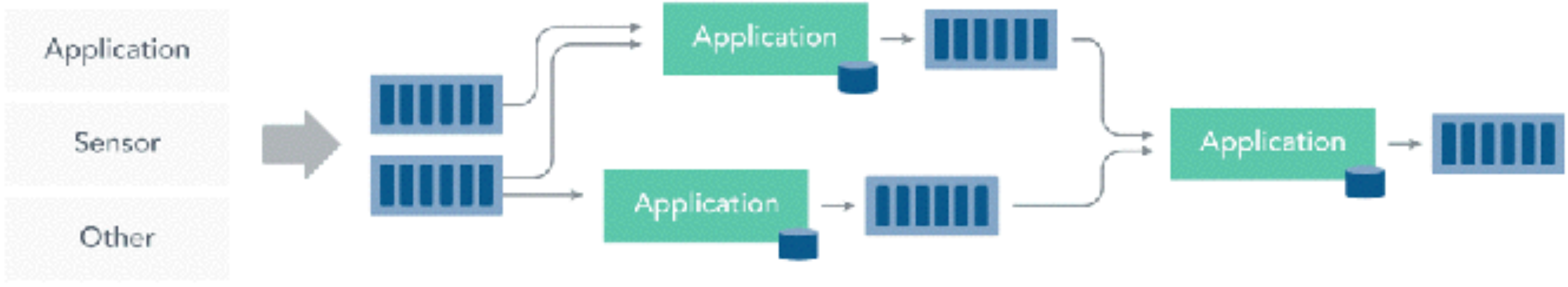


birth of the data
as streams of events

storing data
at rest

applications schedule
computation on the data

Stream Processing



birth of the data
as streams of events

applications computing
over event data streams

Apache Flink in a Nutshell

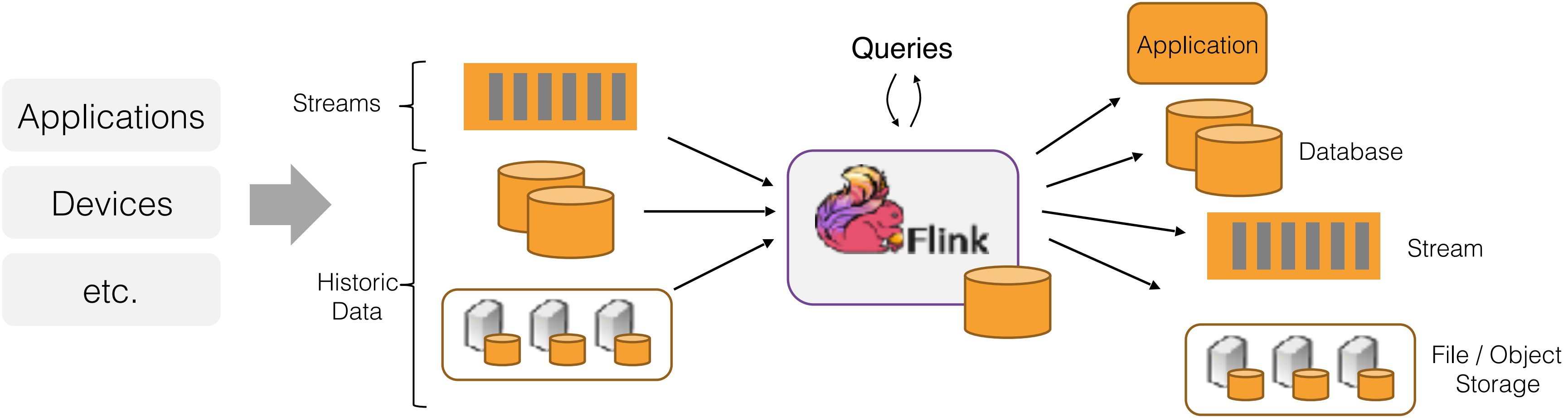
Apache Flink in a Nutshell

Stateful computations over streams

real-time and historic

fast, scalable, fault tolerant, in-memory,

event time, large state, exactly-once

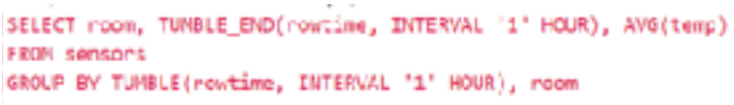


The Core Building Blocks

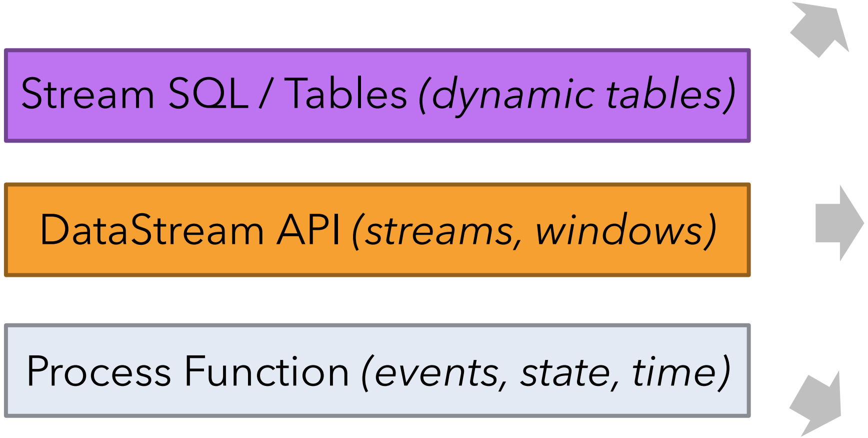
Event Streams	State	(Event) Time	Snapshots
real-time and hindsight	complex business logic	consistency with out-of-order data and late data	forking / versioning / time-travel

Powerful Abstractions

Layered abstractions to
navigate simple to complex use cases



```
SELECT room, TUMBLE_END(rowtime, INTERVAL '1' HOUR), AVG(temp)
FROM sensors
GROUP BY TUMBLE(rowtime, INTERVAL '1' HOUR), room
```



High-level Analytics API

Stream SQL / Tables *(dynamic tables)*

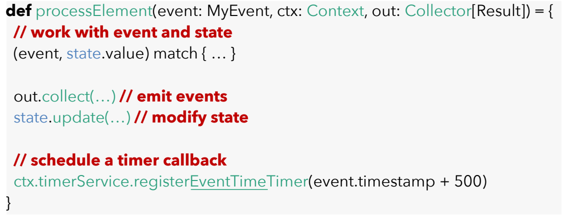
Stream- & Batch Data Processing

DataStream API *(streams, windows)*

Stateful Event-Driven Applications

Process Function *(events, state, time)*

```
val stats = stream
  .keyBy("sensor")
  .timeWindow(Time.seconds(5))
  .sum((a, b) -> a.add(b))
```



```
def processElement(event: MyEvent, ctx: Context, out: Collector[Result]) = {
  // work with event and state
  (event, state.value) match { … }

  out.collect(…) // emit events
  state.update(…) // modify state

  // schedule a timer callback
  ctx.timerService.registerEventTimeTimer(event.timestamp + 500)
}
```

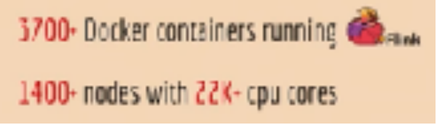
Hardened at scale

UBER

Athena X Streaming SQL
Platform Service

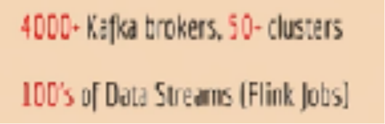
NETFLIX

Streaming Platform as a Service



3700+ Docker containers running

1400+ nodes with 22K+ cpu cores



4000+ Kafka brokers, 50+ clusters

100's of Data Streams (Flink Jobs)

Alibaba Group

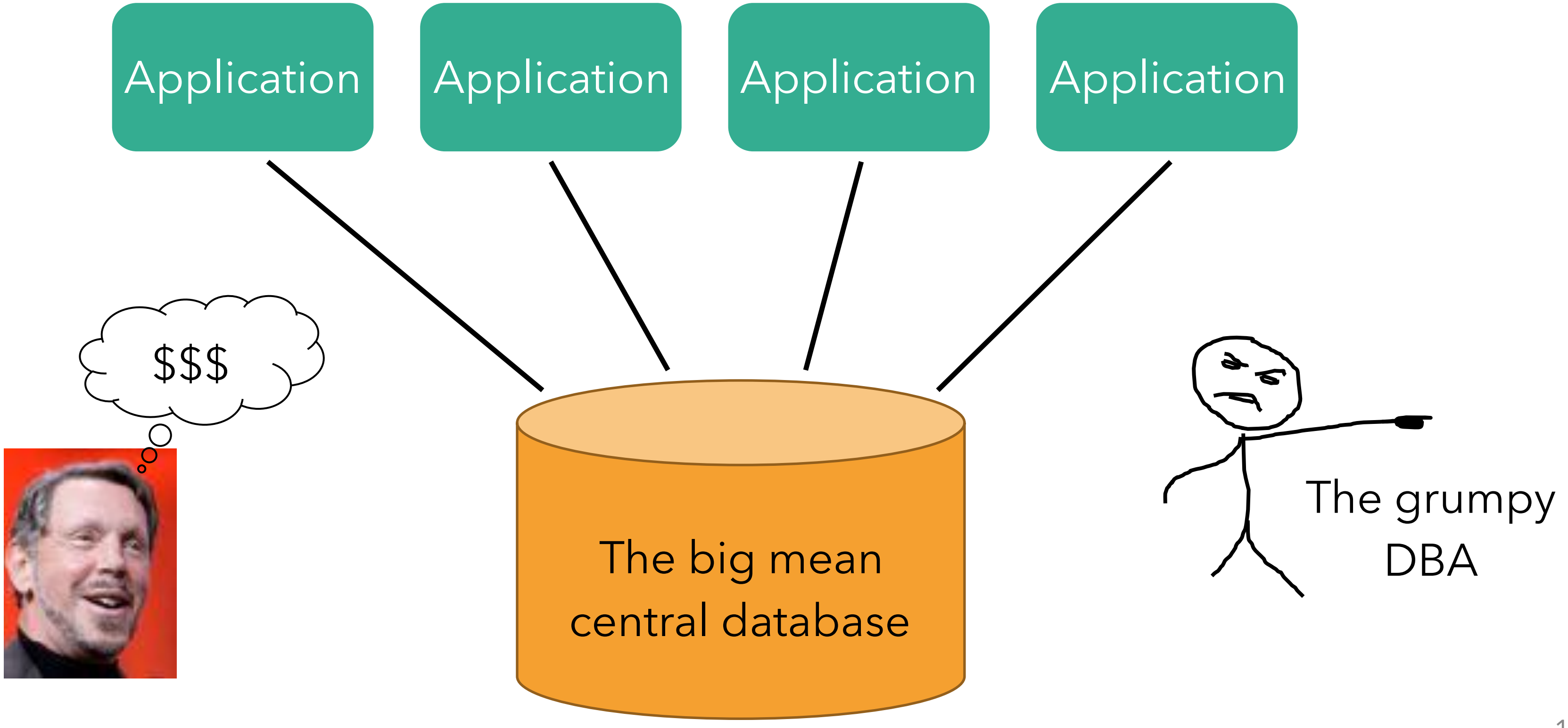
100s jobs, 1000s nodes, TBs state
metrics, analytics, real time ML
Streaming SQL as a platform

ING

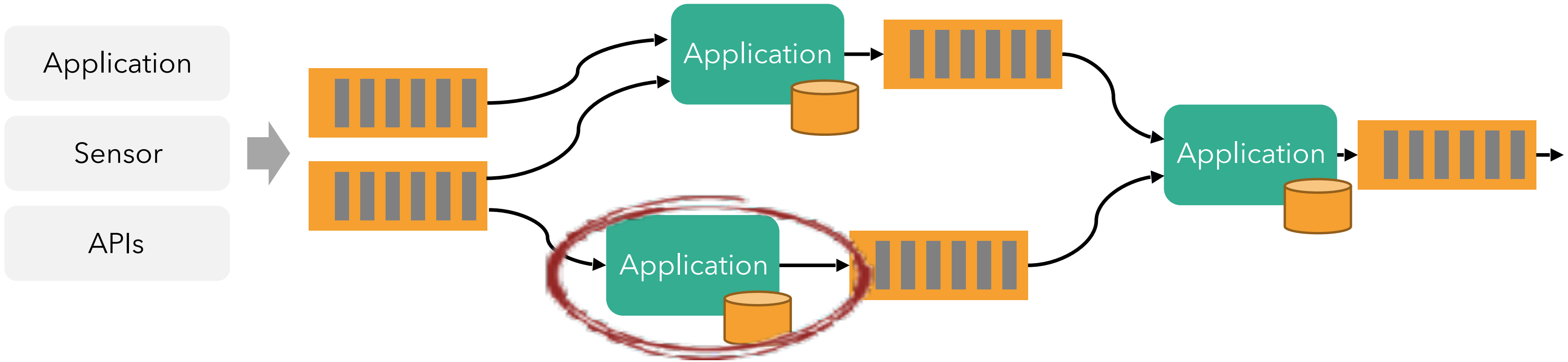
Fraud detection
Streaming Analytics Platform

Distributed application infrastructure

Good old centralized architecture



Modern distributed app. architecture

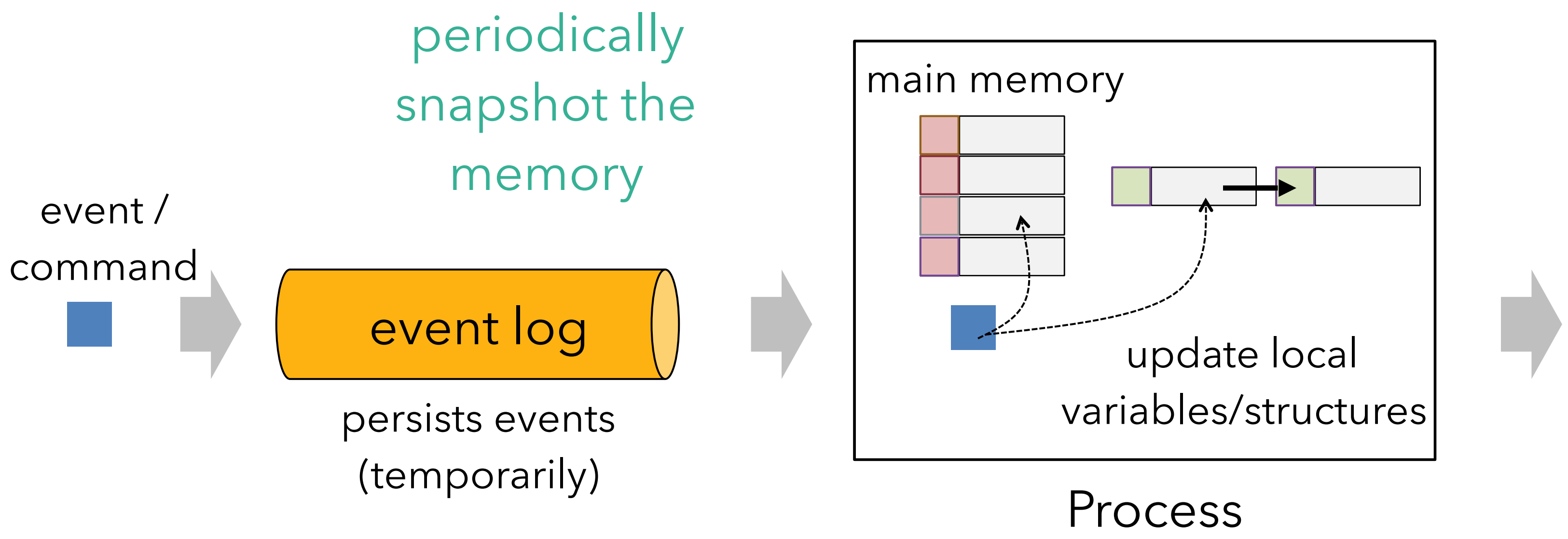


The limit to what you can do is how sophisticated can you compute over the stream!

Boils down to: How well do you handle **state** and **time**

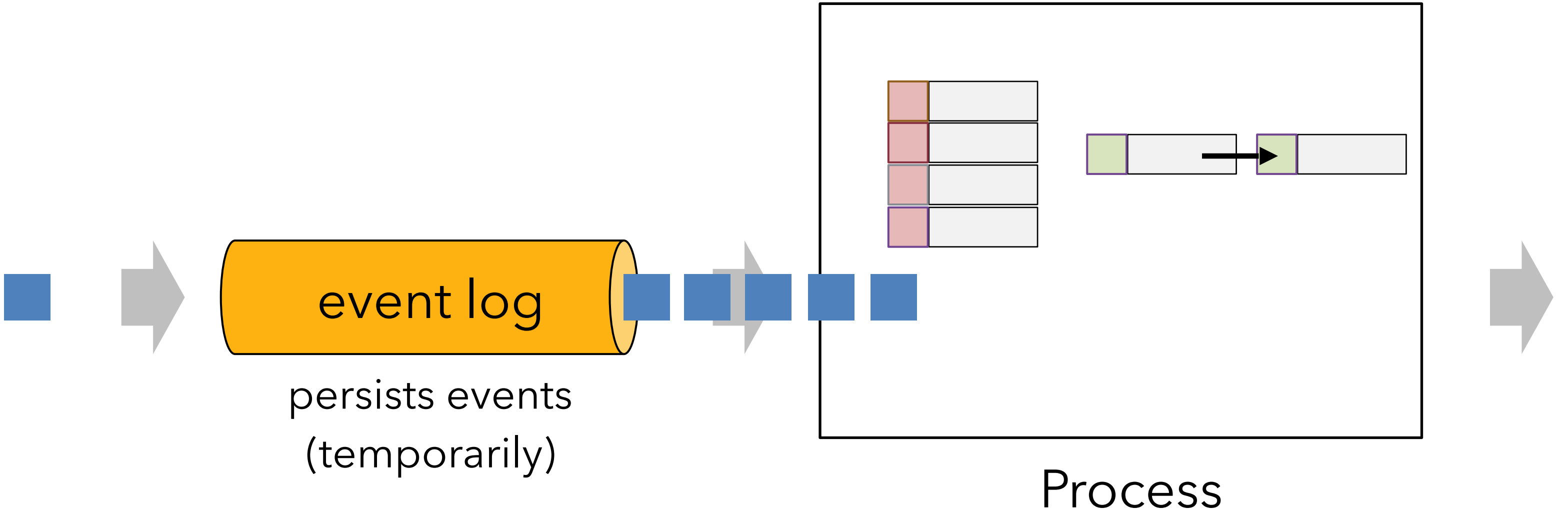
A Flink-favored approach

Event Sourcing + Memory Image


periodically snapshot the memory
event / command
event log
persists events (temporarily)
main memory
update local variables/structures
Process

Event Sourcing + Memory Image

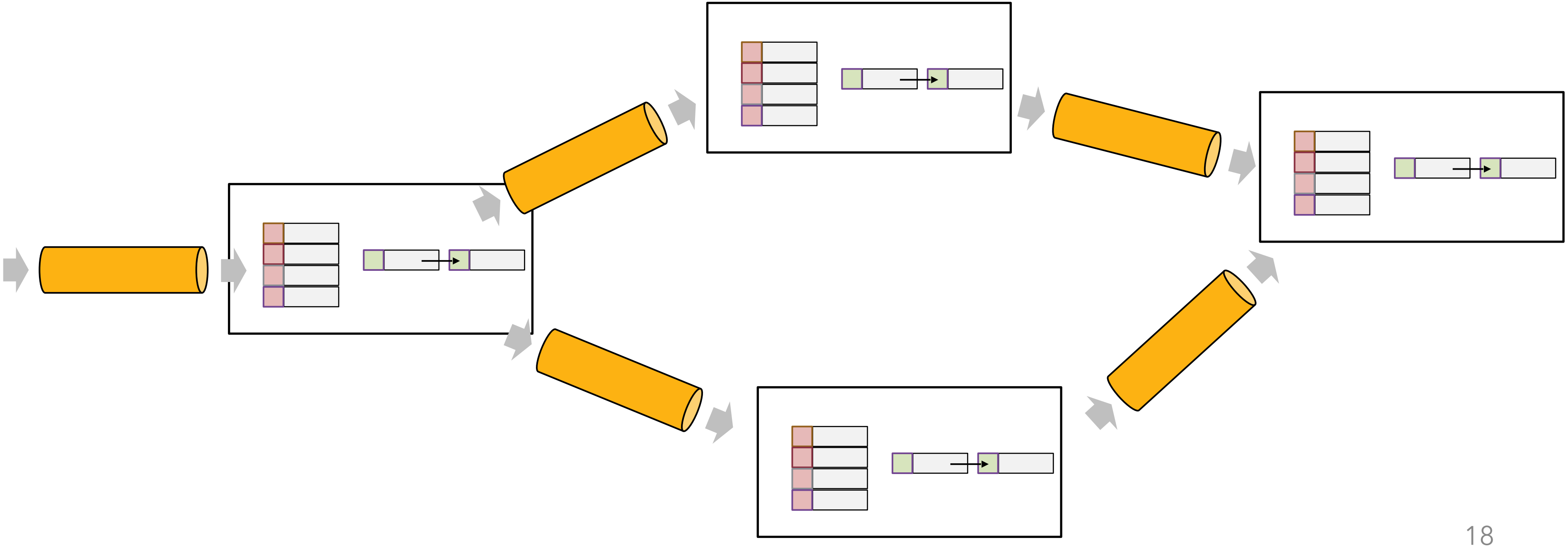
Recovery: Restore snapshot and replay events since snapshot



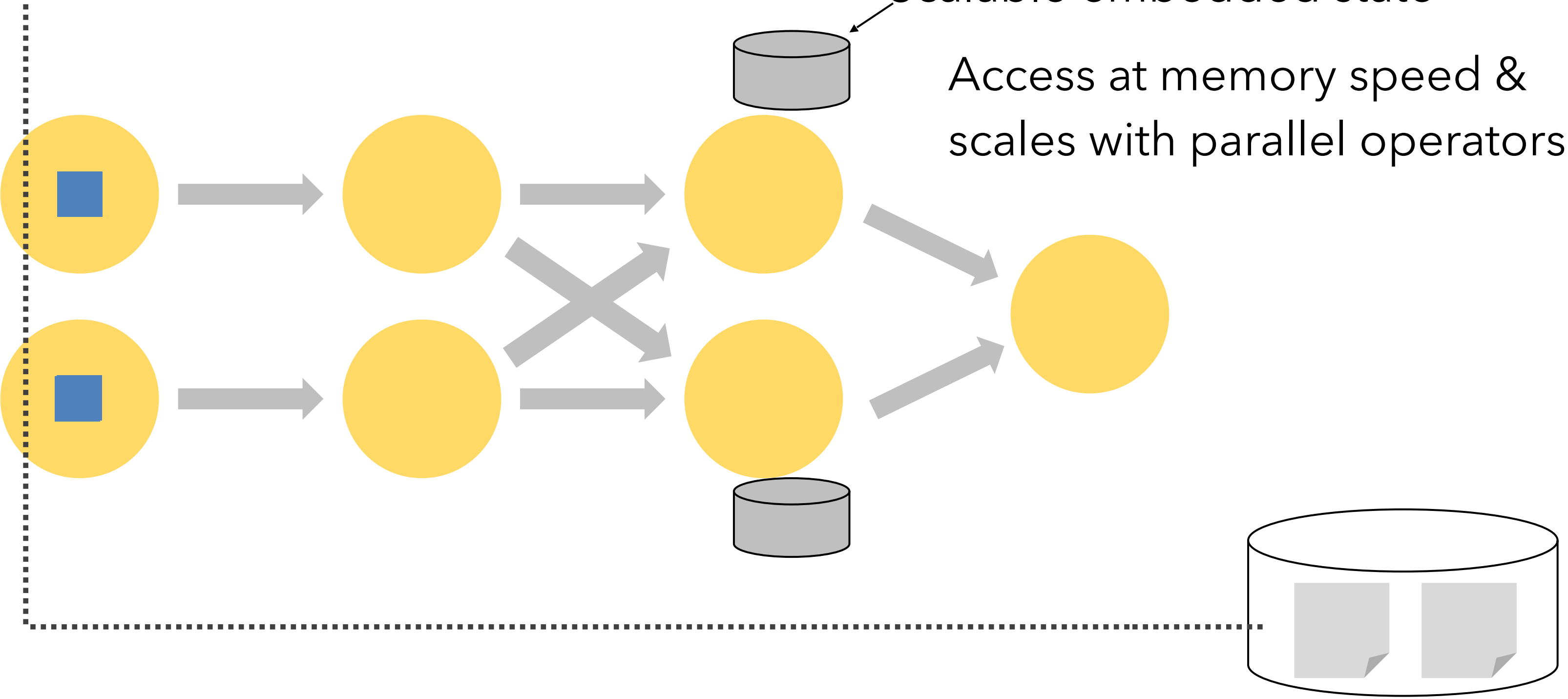
Distributed Memory Image

Distributed application, many memory images.

Snapshots are all consistent together.



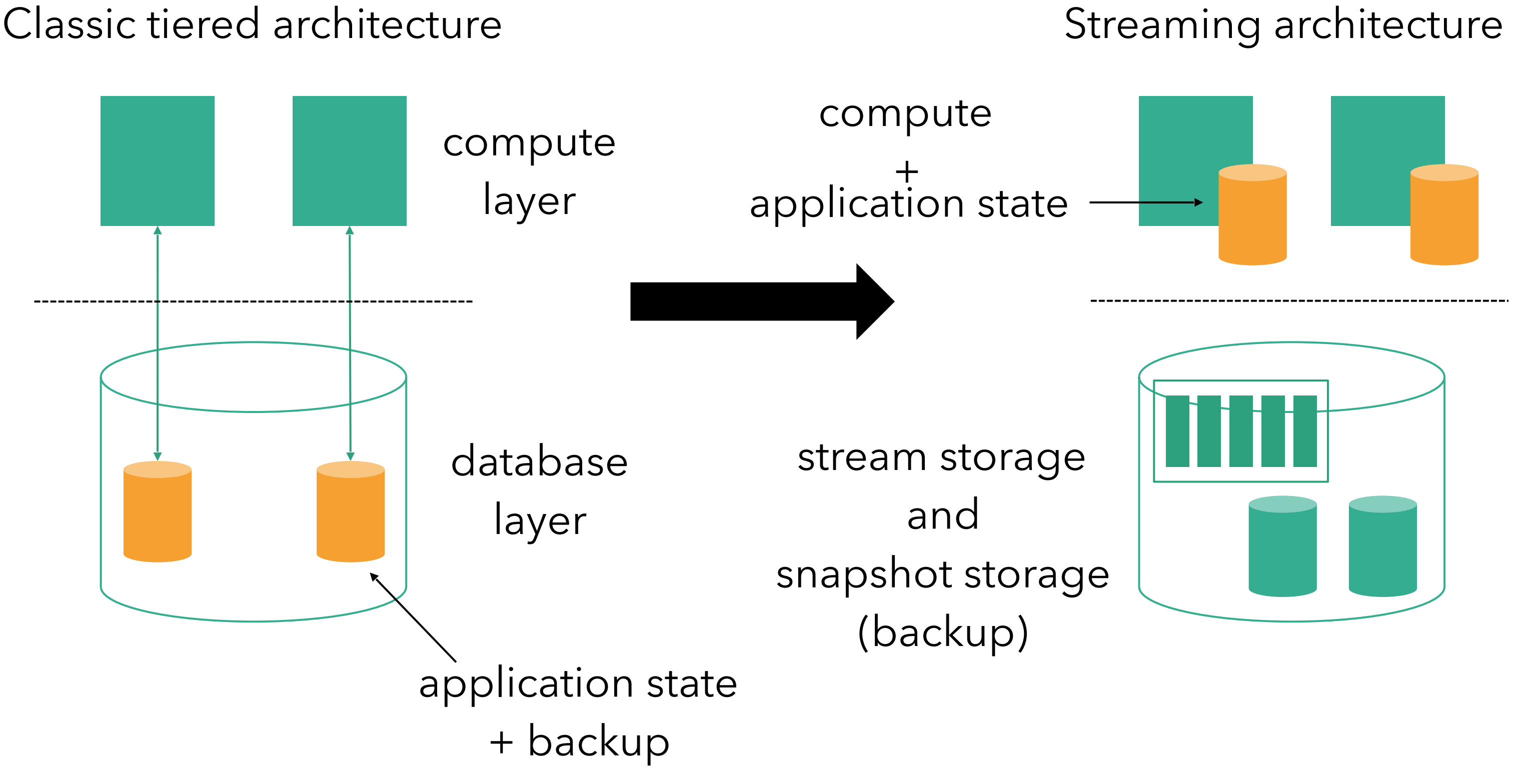
Stateful Event & Stream Processing



Scalable embedded state

Access at memory speed & scales with parallel operators

Compute, State, and Storage



Classic tiered architecture

compute layer

database layer

application state + backup

Streaming architecture

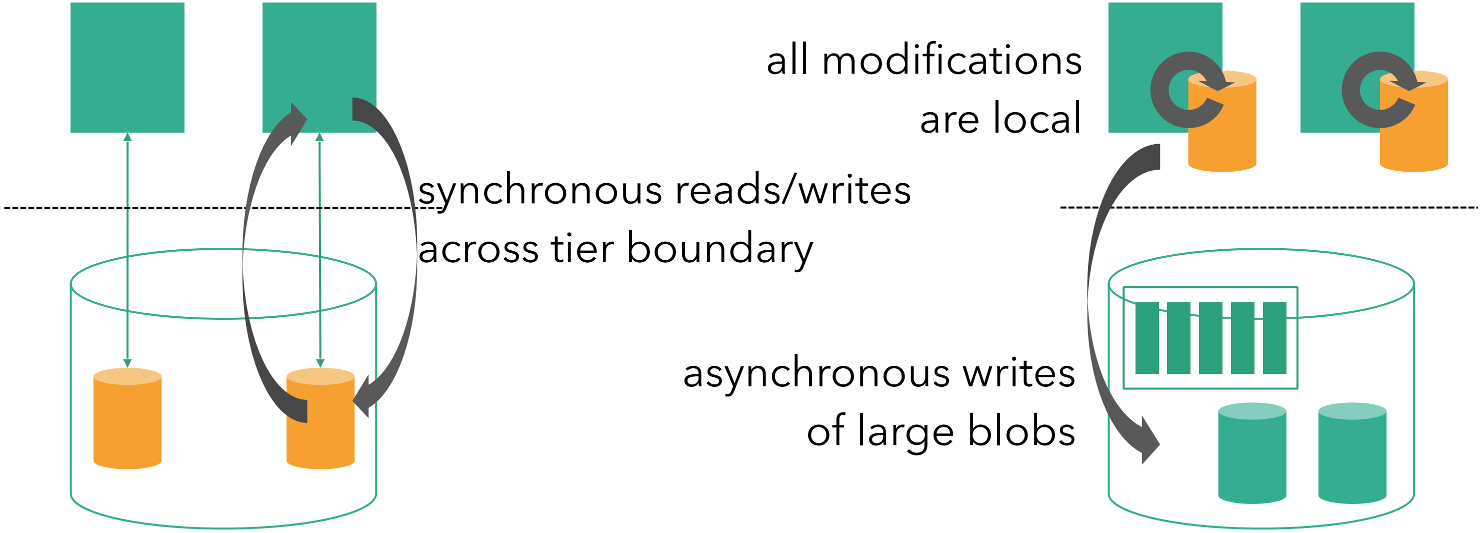
compute + application state

stream storage and snapshot storage (backup)

Performance

Classic tiered architecture

Streaming architecture



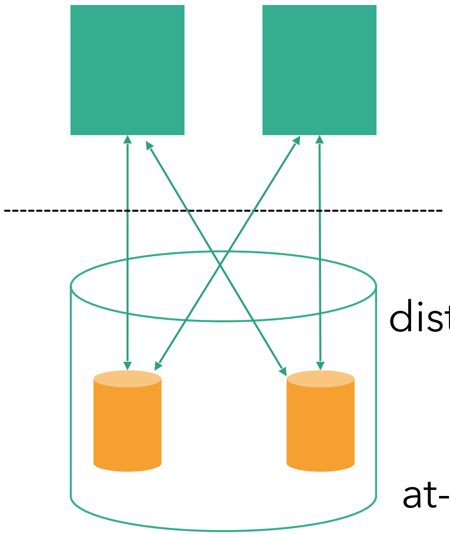
all modifications are local

synchronous reads/writes across tier boundary

asynchronous writes of large blobs

Consistency

Classic tiered architecture

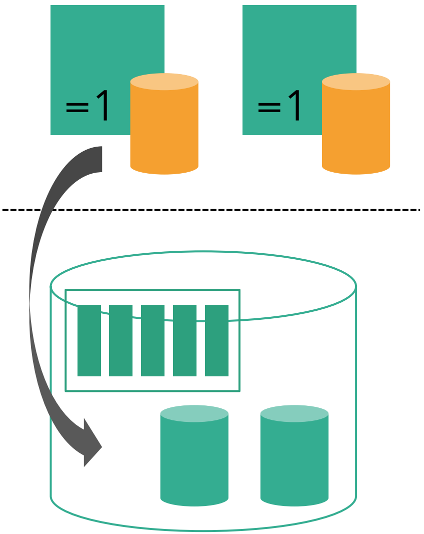


distributed transactions

at scale typically
at-most / at-least once

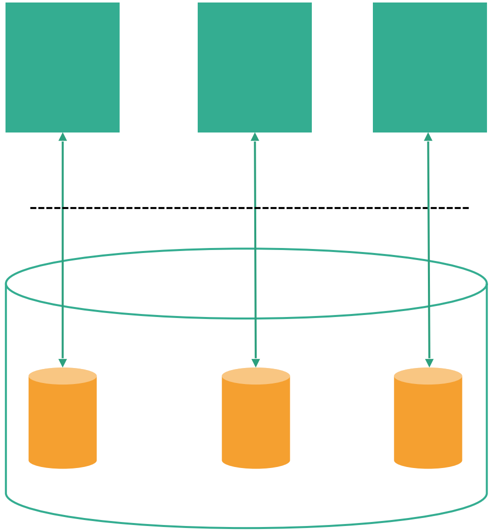
Streaming architecture

exactly once
per state



Scaling a Service

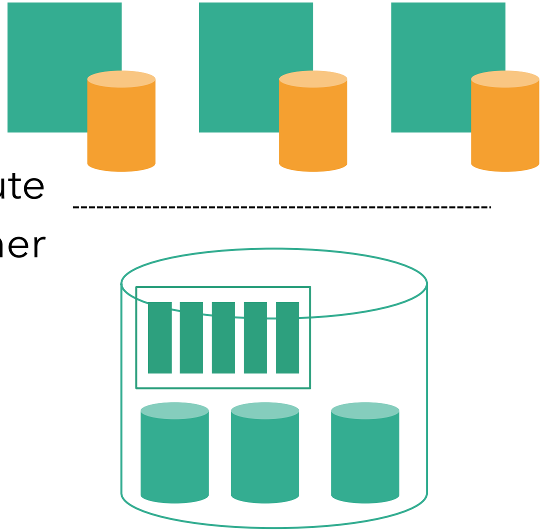
Classic tiered architecture



provision compute

separately provision additional database capacity

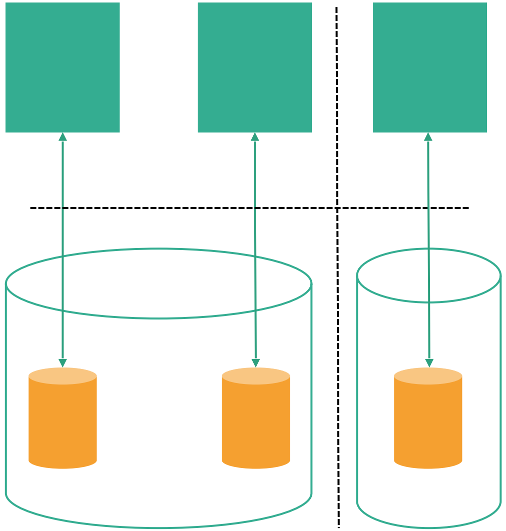
Streaming architecture



provision compute and state together

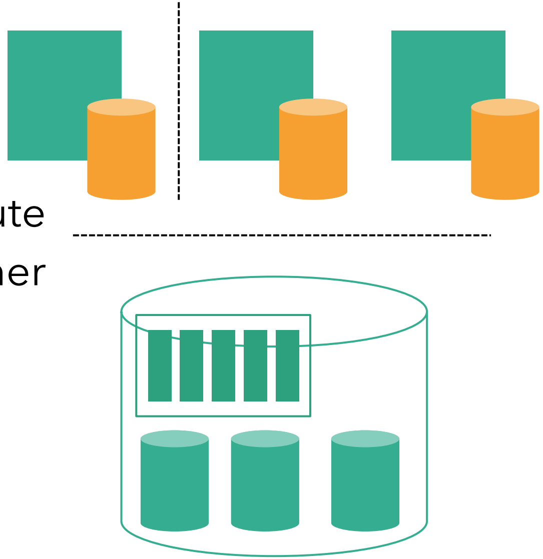
Rolling out a new Service

Classic tiered architecture



provision a new database
(or add capacity to an existing one)

Streaming architecture



provision compute
and state together

simply occupies some
additional backup space

What users built on checkpoints…

- Upgrades and Rollbacks
- Cross Datacenter Failover
- State Archiving
- Application Migration
- Spot Instance Region Chasing
- A/B testing
- …

What is the next wave of stream processing applications?

What changes faster? Data or Query?

Data changes slowly compared to fast changing queries

ad-hoc queries, data exploration, ML training and tuning

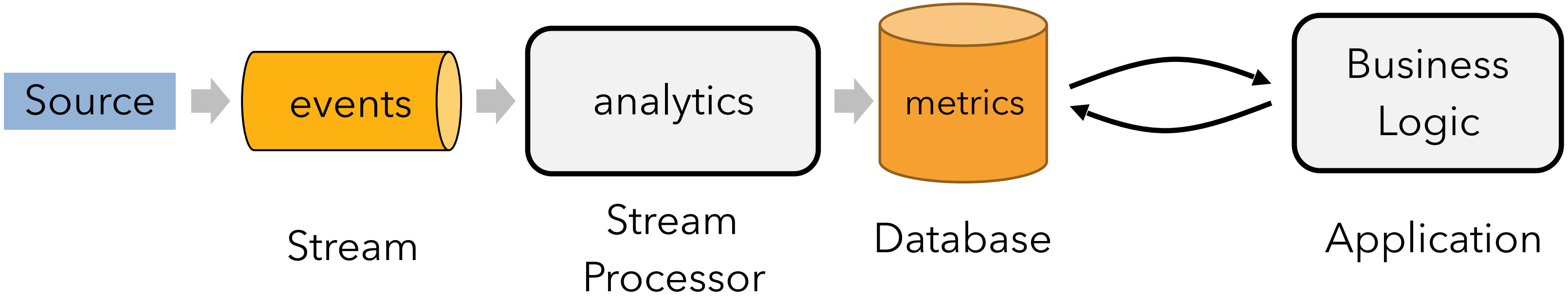
Batch Processing Use Case

Data changes fast application logic is long-lived

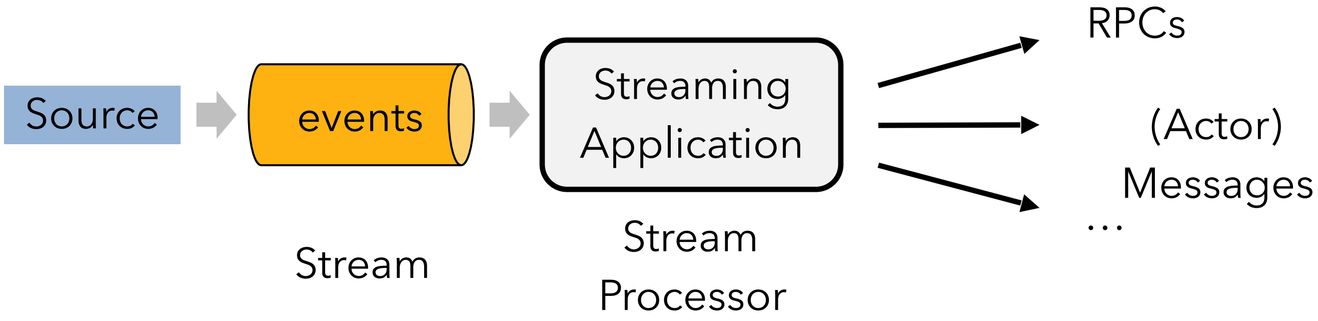
continuous applications, data pipelines, standing queries, anomaly detection, ML evaluation, …

Stream Processing Use Case

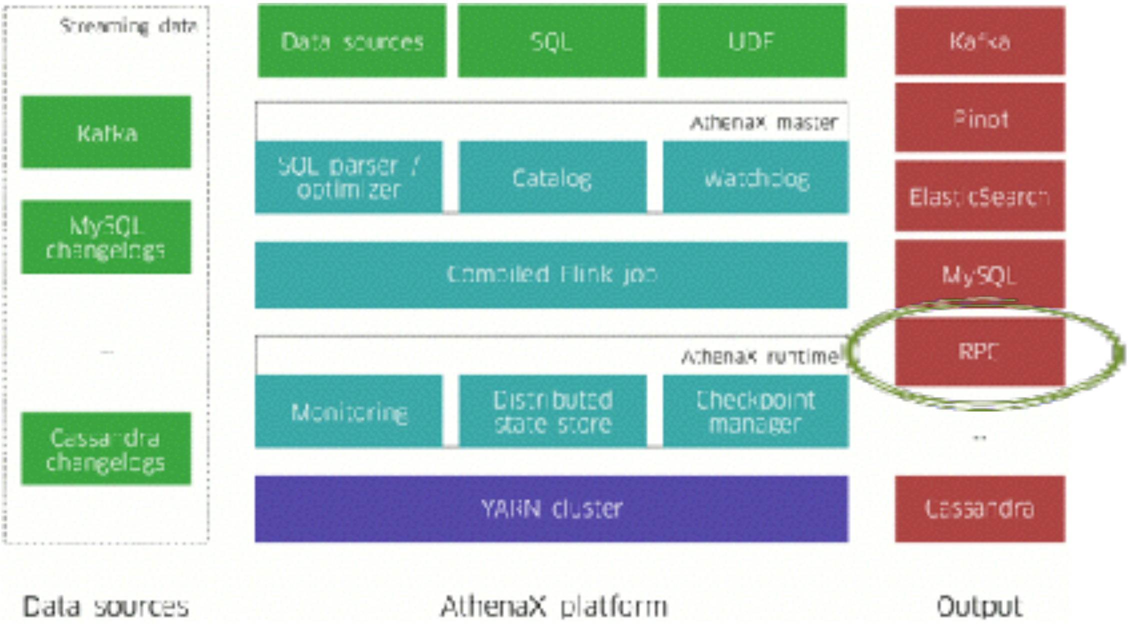
Analytics & Business Logic



Blending Analytics & Business Logic



AthenaX by Uber

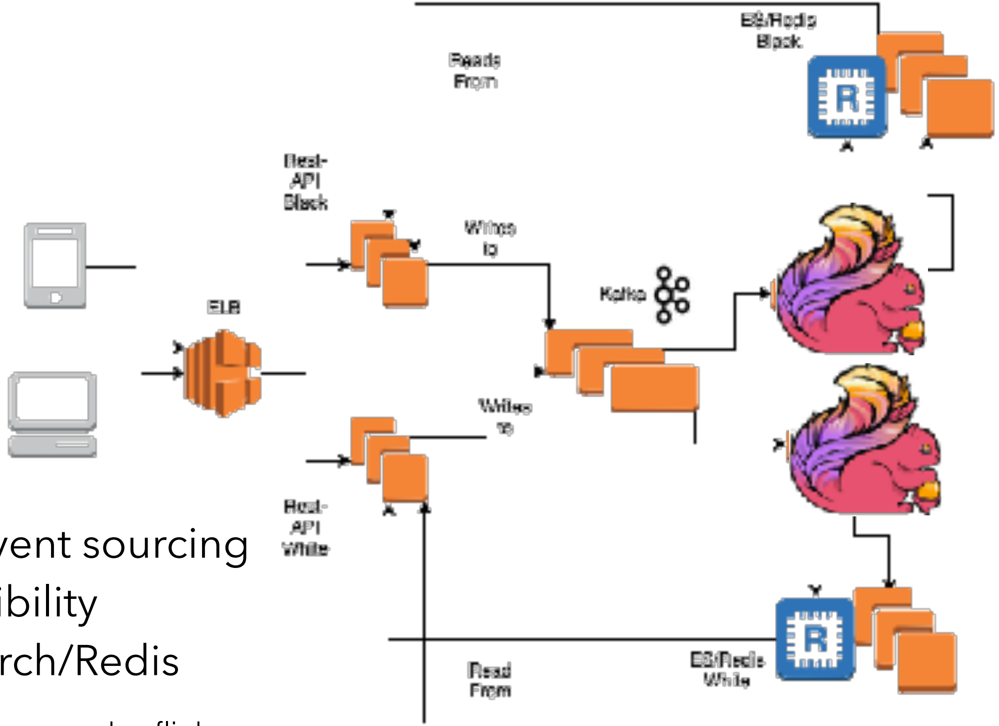


Can one build an entire sophisticated web application (say a social network) on a stream processor?

(Yes, we can!™)

@ D_TRB

THE SOCIAL NETWORK
FOR PETROLHEADS



Social network implemented using event sourcing and CQRS (Command Query Responsibility Segregation) on Kafka/Flink/Elasticsearch/Redis

More: https://data-artisans.com/blog/drivetribe-cqrs-apache-flink

The next wave of stream processing applications…

… is all types of stateful applications that react to data and time!

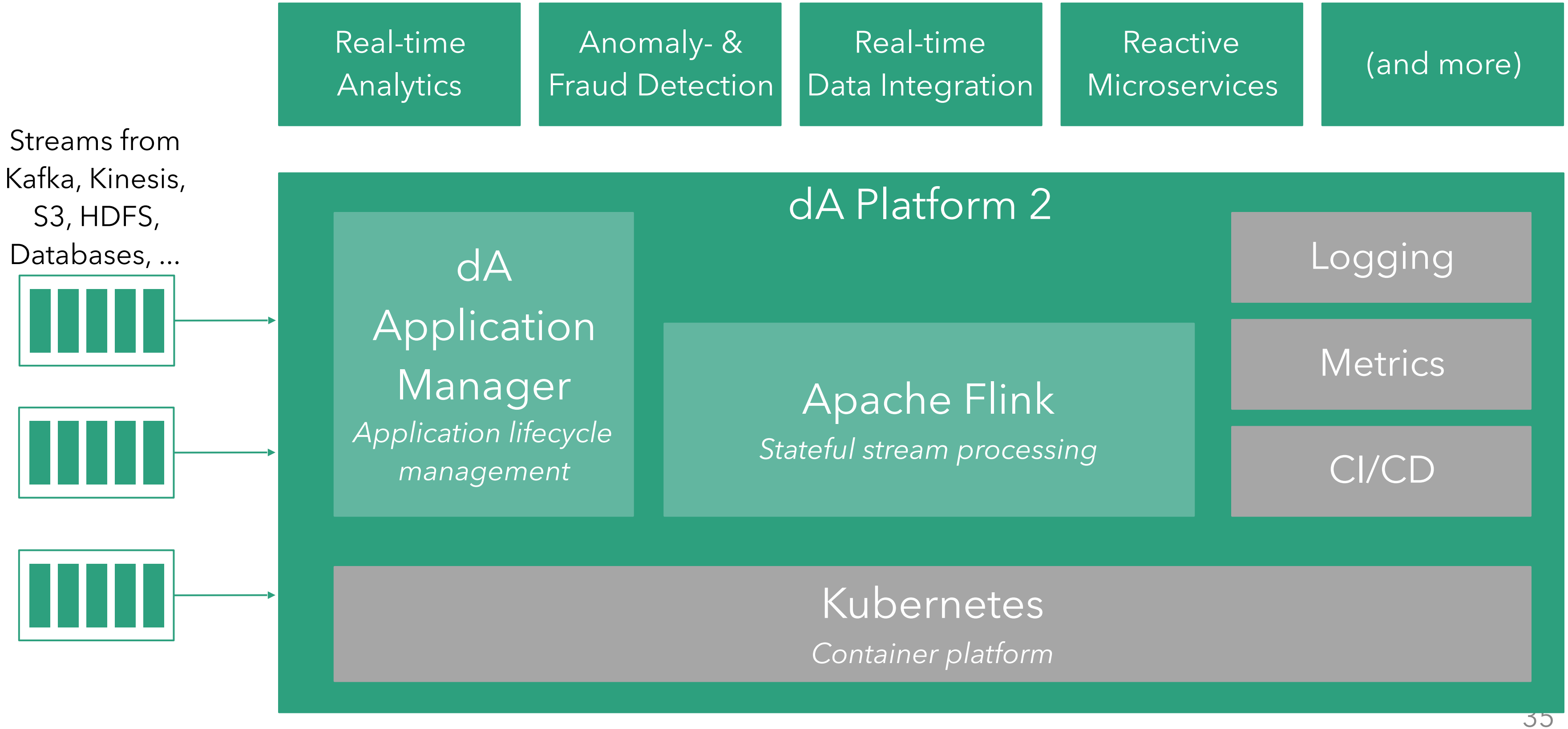
+

Stateful stream applications

Continuous applications

versioning, upgrading, rollback, duplicating, migrating, …

The dA Platform Architecture



Streams from Kafka, Kinesis, S3, HDFS, Databases, ...

Real-time Analytics

Anomaly- & Fraud Detection

Real-time Data Integration

Reactive Microservices

(and more)

dA Platform 2

dA Application Manager

Application lifecycle management

Apache Flink

Stateful stream processing

Logging

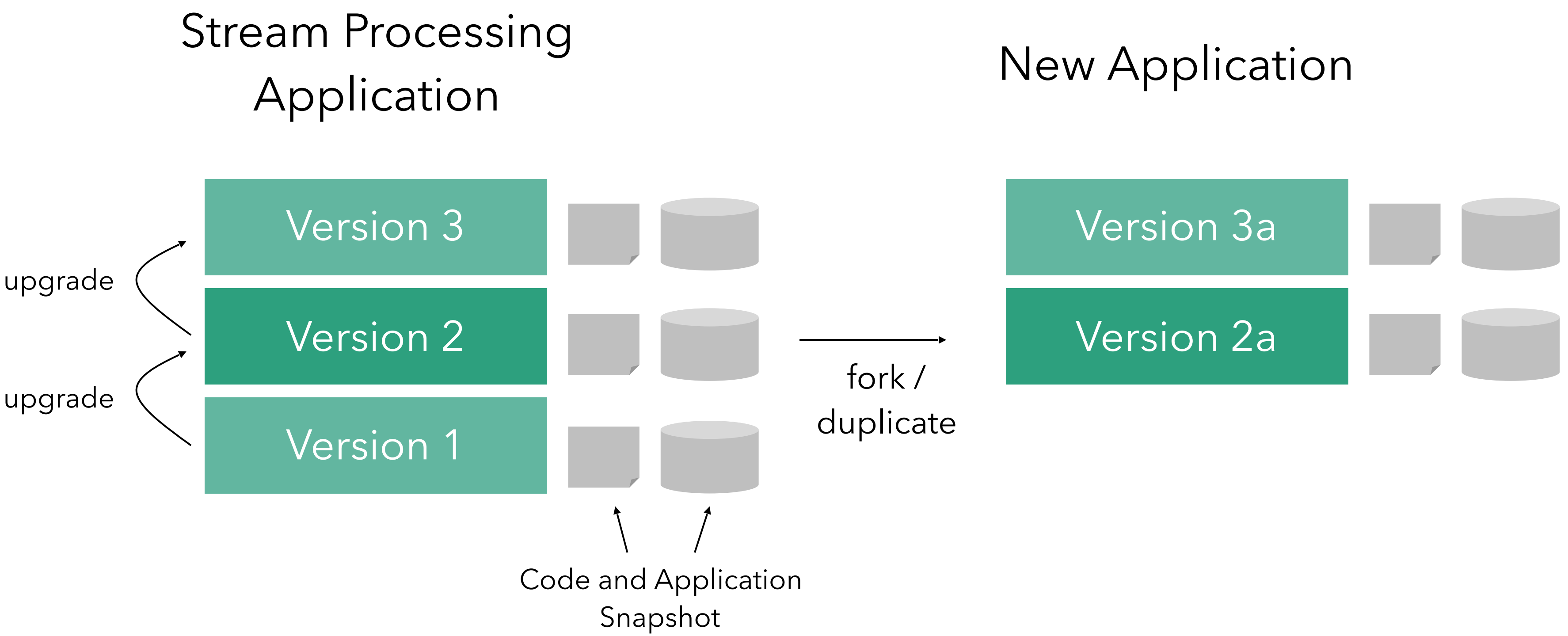
Metrics

CI/CD

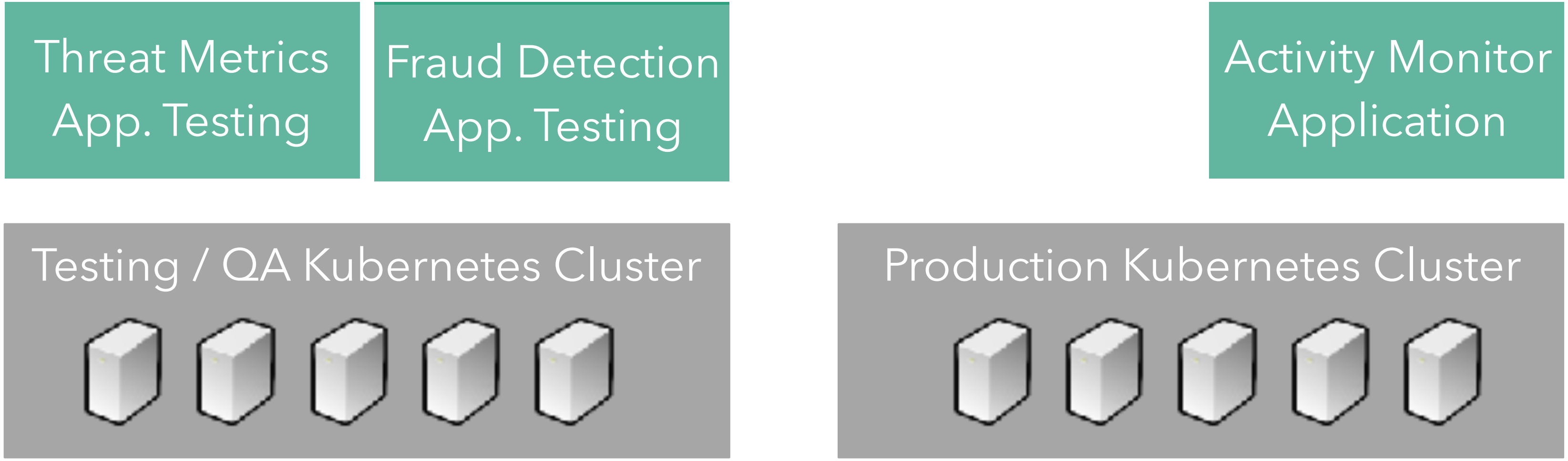
Kubernetes

Container platform

Versioned Applications, not Jobs/Jars



Deployments, not Flink Clusters



Threat Metrics App. Testing

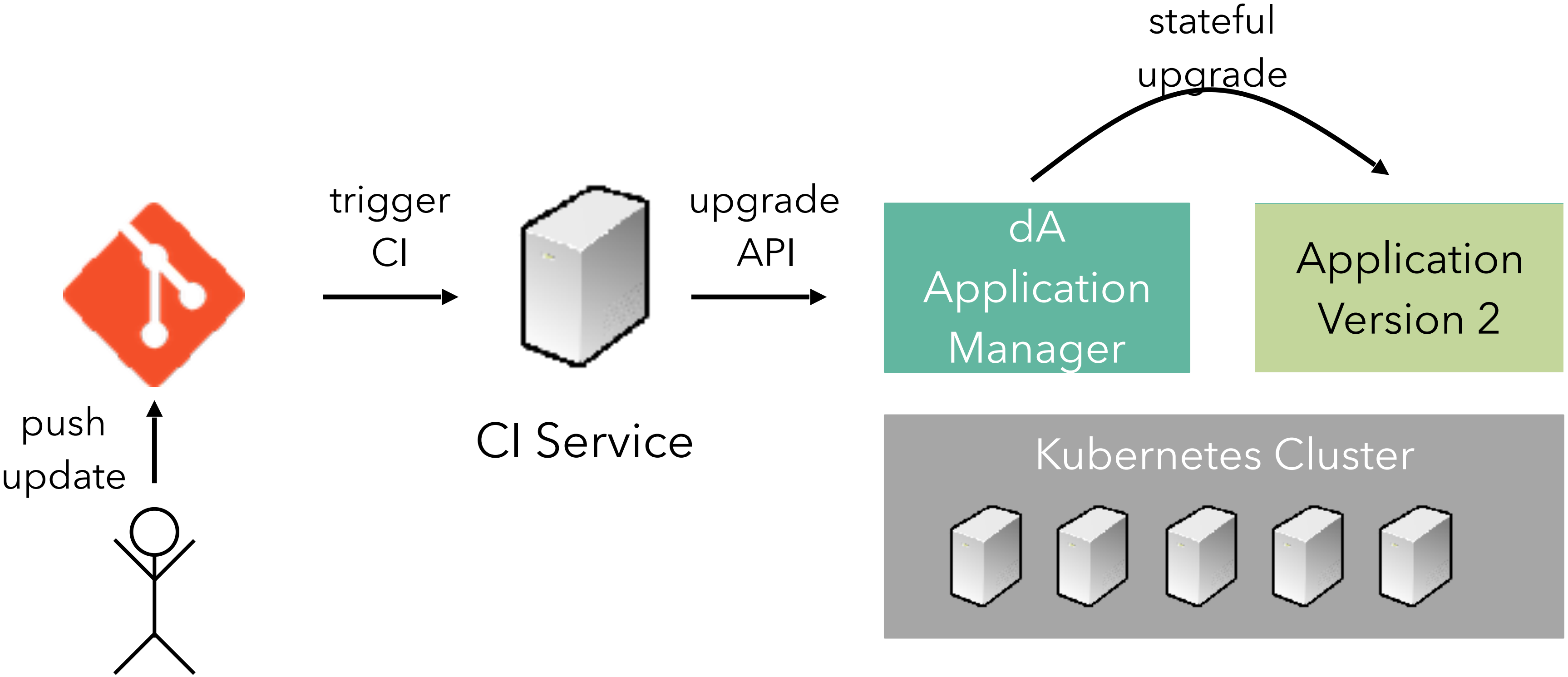
Fraud Detection App. Testing

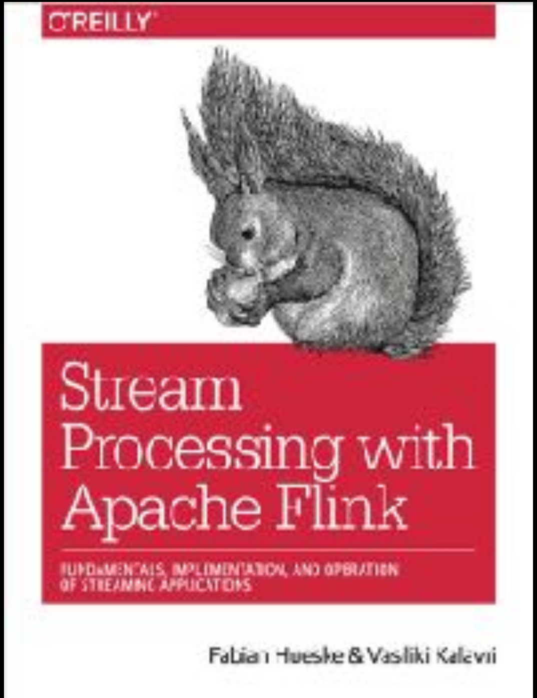
Activity Monitor Application

Testing / QA Kubernetes Cluster

Production Kubernetes Cluster

Hooks for CI/CD pipelines





O'REILLY

Stream Processing with Apache Flink

FUNDAMENTALS, IMPLEMENTATION, AND OPERATION OF STREAMING APPLICATIONS

Fabian Hueske & Vasiliki Kalavri

Thank you!

@stsffap

@ApacheFlink

@dataArtisans

dataArtisans

We are hiring!

data-artisans.com/careers

Please

Remember to rate this session

Thank you!

@GOTOber

gotober.com