IT5100B

Industry Readiness

Stream Processing

LECTURE 6

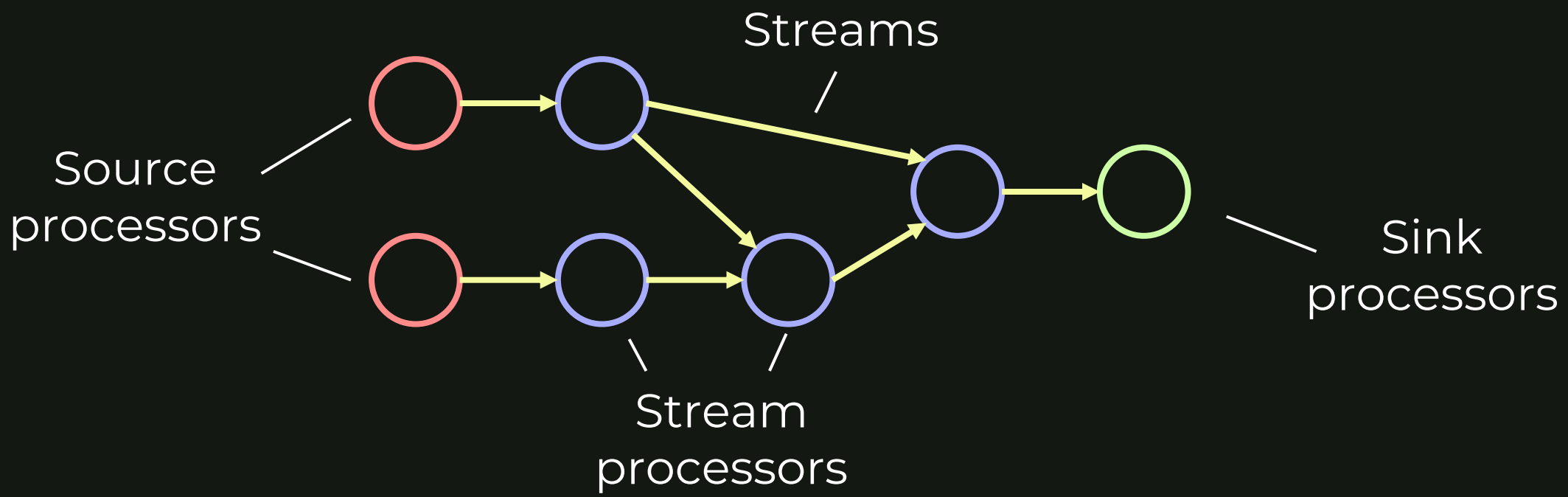
High Throughput Stream Processing

FOO Yong Qi
yongqi@nus.edu.sg

CONTENTS

HIGH THROUGHPUT STREAM PROCESSING

KAFKA STREAMS

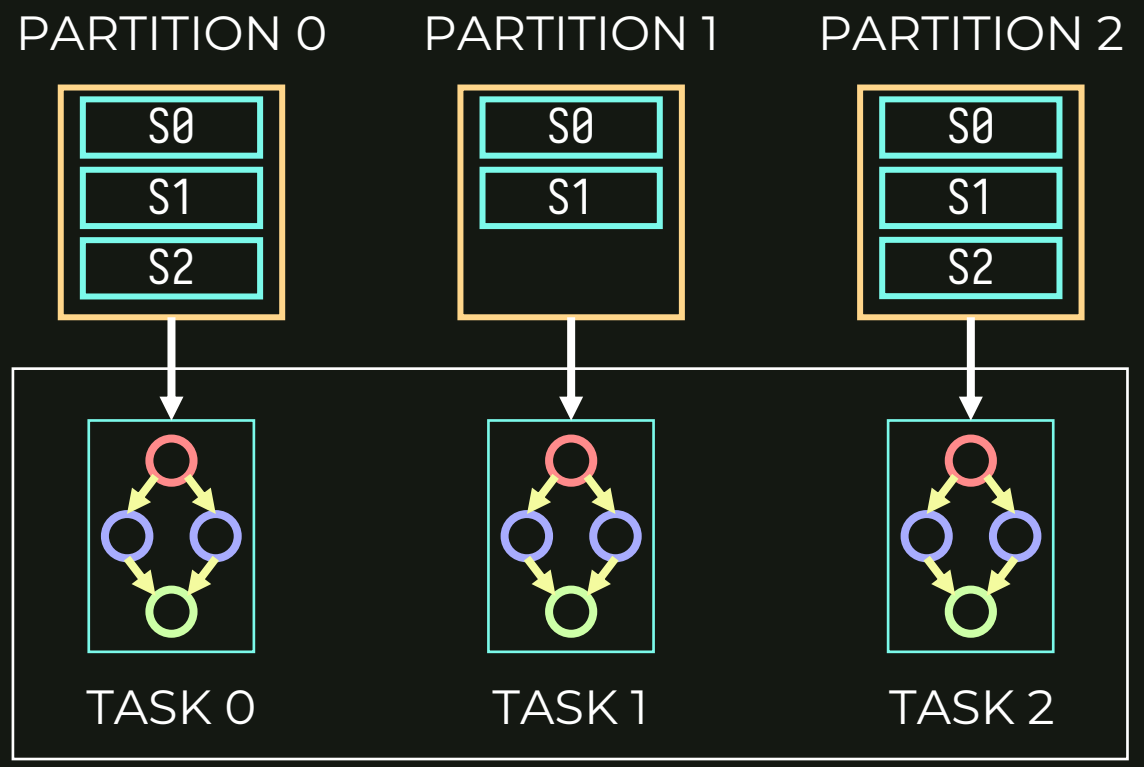


Kafka Streams applications great for deploying microservices that process event streams in Kafka

KAFKA STREAMS

PARALLELISM IN KAFKA STREAMS

KAFKA STREAMS APPLICATION AS CONSUMER GROUP



Kafka Streams leverage partitioning of Kafka topics, achieving parallelism

STREAM PROCESSING IN THE WILD

What if we have the following requirements:

- High throughput distributed stream processing (heavy processing on large number of events per second)
- Different data sources and sinks than just Kafka (Hadoop, ElasticSearch, Databases etc.)
- Fault-tolerant stateful stream processing

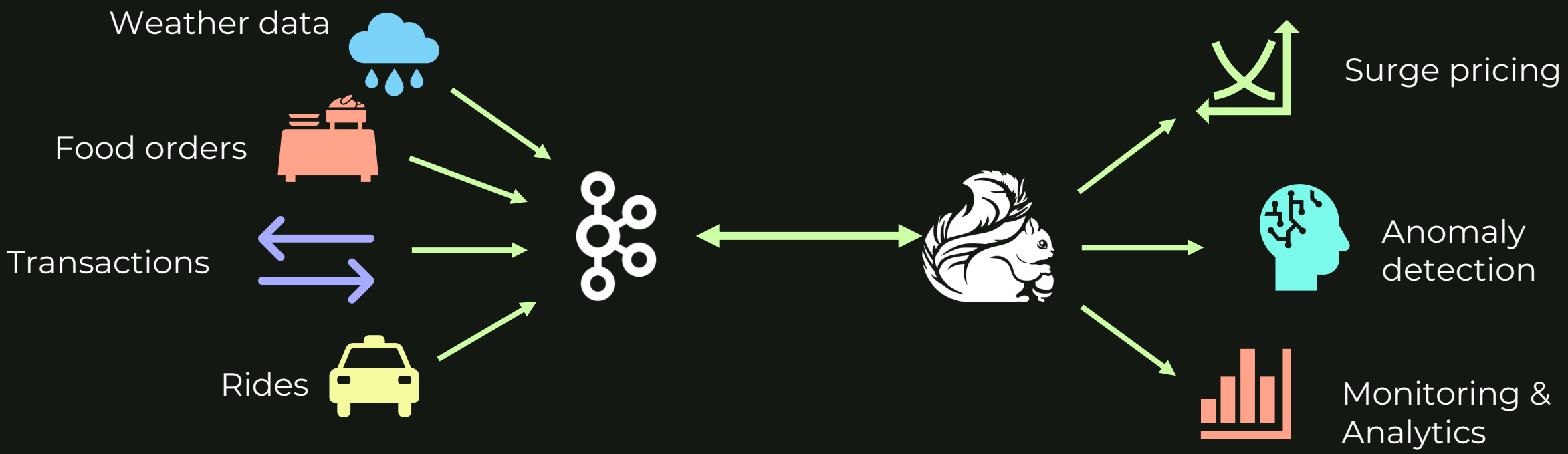
APACHE FLINK®

STREAM PROCESSING ENGINE

Framework and distributed processing engine for stateful computations over unbounded and bounded (batches) data streams

APACHE FLINK®

USE CASES



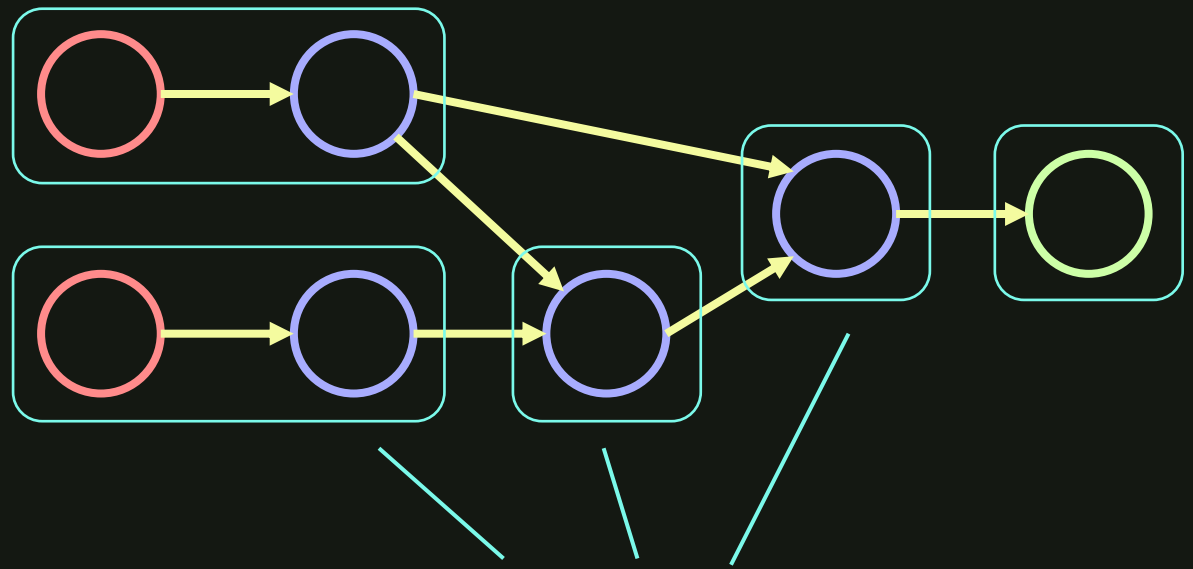
Companies like Uber, Gojek and Lyft use Apache Kafka + Apache Flink (and forks) for many real-time services: traffic monitoring, primetime + heatmaps, fraud detection, ride receipts, driver incentives, anomaly detection etc.

APACHE FLINK®

USE CASES

Alibaba uses Flink for machine learning, monitoring and others; Flink offers more powerful stream processing features and different sources and sinks!

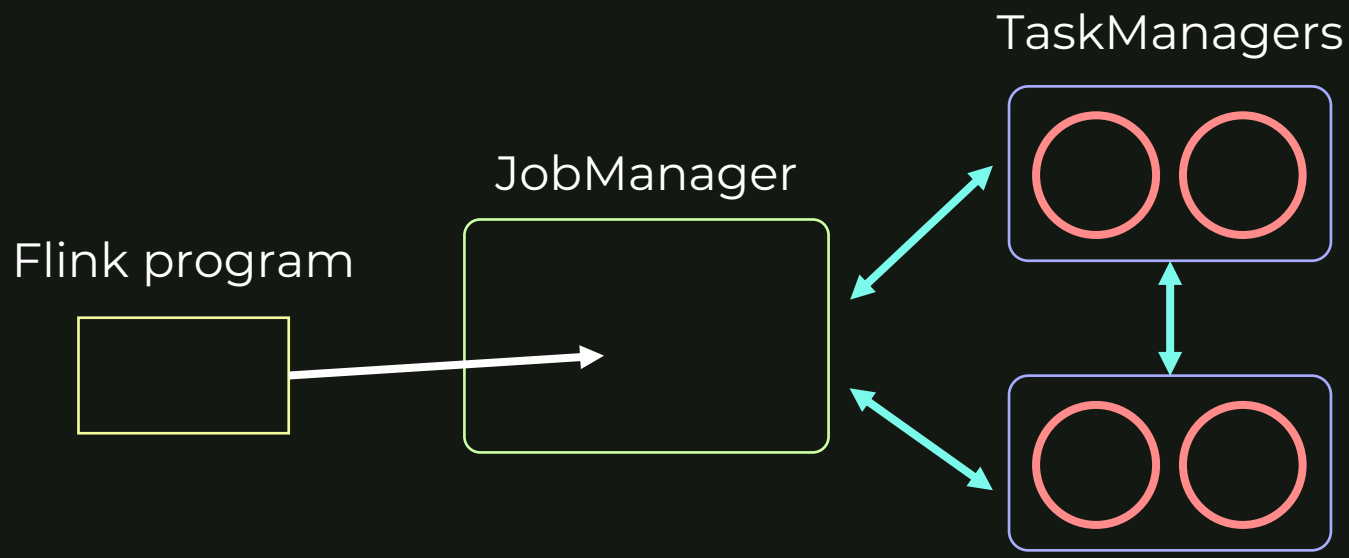
FLINK TASKS



Unlike Kafka Streams, Flink topology is broken into tasks that can be run on separate machines

FLINK CLUSTER

SESSION CLUSTERS



Flink cluster consists of one or more JobManagers (coordination) and one or more TaskManagers (running tasks); task slots in TaskManagers run tasks

In Session Cluster, Flink cluster is always running; submit jobs to JobManager to let Flink run the job

FLINK CLUSTER

WITH DOCKER

Get new `docker-compose.yaml` file on Canvas and use `docker-compose` to run the cluster (comes with Kafka and Flink)

```
docker-compose up -d
```

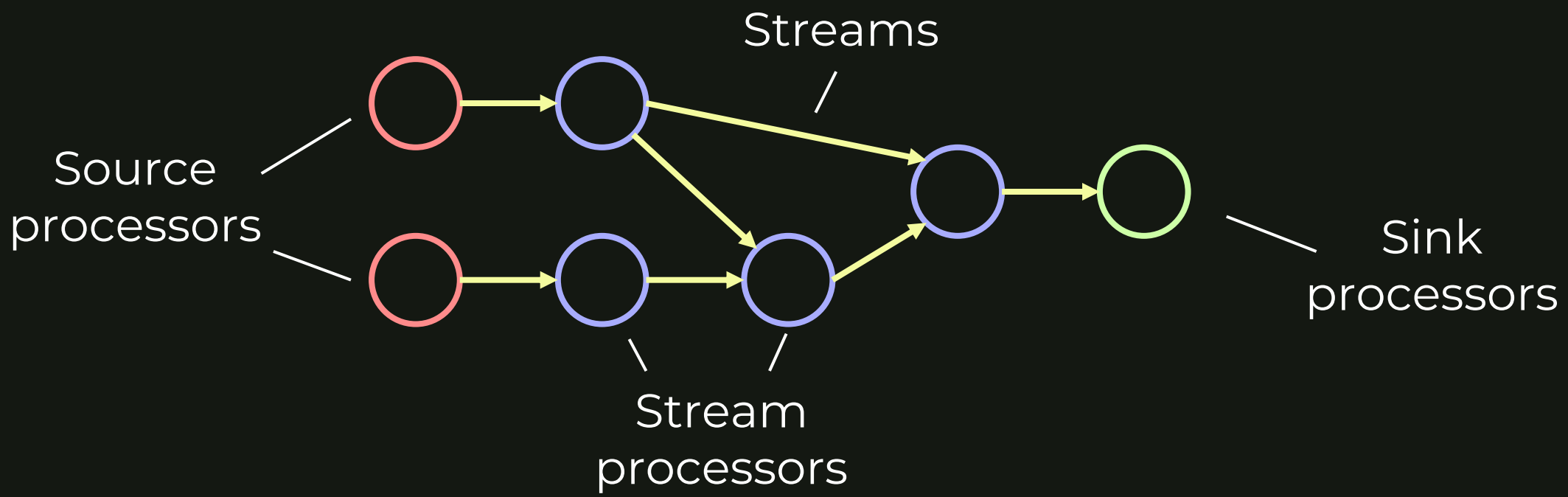
You may choose to scale the number of task managers:

```
docker-compose scale taskmanager=<N>
```

WRITING FLINK JOBS

- Flink has poor backwards compatibility; you must use the right versions for everything
- Use the `pom.xml` files in Canvas and create the Java project using Maven
- Change the `mainClass` in the `pom.xml` file to your main class
- Use `mvn package` to package your code into a JAR file for submission to your Flink JobManager

FLINK JOB GRAPH



Just like Kafka Streams, a Flink Job creates a job graph / topology that processes streams

OUR FIRST FLINK JOB

ALMOST IDENTICAL (EXCEPT NAMES) TO KAFKA STREAMS

```
KafkaSource<KafkaEvent> source = KafkaSource.<KafkaEvent>builder()
        .setBootstrapServers(BOOTSTRAP_SERVERS)
        .setTopics(SOURCE_TOPIC)
        .setGroupId("flink-processor")
        .setStartingOffsets(OffsetsInitializer.committedOffsets(
                OffsetResetStrategy.EARLIEST))
        .setDeserializer(new KafkaEventDeserializer(SOURCE_TOPIC))
        .setProperty("partition.discovery.interval.ms", "60000")
        .build();
```

First step is to configure the source and sink connectors

OUR FIRST FLINK JOB

ALMOST IDENTICAL (EXCEPT NAMES) TO KAFKA STREAMS

```
KafkaSink<KafkaEvent> sink = KafkaSink.<KafkaEvent>builder()
        .setBootstrapServers(BOOTSTRAP_SERVERS)
        .setRecordSerializer(new KafkaEventSerializer(SINK_TOPIC))
        .setTransactionalIdPrefix("flink")
        .build();
```

First step is to configure the source and sink connectors

OUR FIRST FLINK JOB

ALMOST IDENTICAL (EXCEPT NAMES) TO KAFKA STREAMS

```
StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
env.fromSource(source, WatermarkStrategy.noWatermarks(), "sensor-data-raw")
        .filter(x -> x.value >= 0)
        .name("filter negative")
        .map(x -> new KafkaEvent(x.key, x.value - 273.15, x.timestamp))
        .name("K to C")
        .sinkTo(sink).name("sensor-data-avg");
env.execute("Temperature Adjustment");
```

Building the job looks almost the same as doing processing with Stream, Flux and KStream!

KEY POINT #1

Writing Flink jobs is very similar to writing Kafka Streams Applications!

FLINK VS KAFKA STREAMS

FAULT-TOLERANCE & PARALLELISM

- Flink supports checkpoints—globally consistent snapshots of state over stateful computations
- Flink operators can live on different machines, streams elements are serialized

For simplicity, let your data classes be POJOs (Plain-Old Java Objects)

- Default constructor
- Public fields

DEPLOYMENT MODEL & CONNECTORS

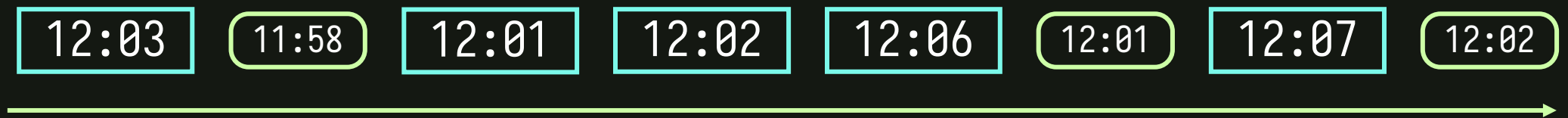
Kafka Streams

- Relies on Kafka for fault tolerance
- Only connects to Kafka
- Stream processing microservices

Flink

- Has its own checkpointing system
- Has a variety of connectors
- Stream processing cluster

WATERMARKS & TIME



Flink allows you to customize the strategy for determining stream time by customizing watermark emission

If working with time-sensitive operations, define watermarking strategy

EXAMPLE APPLICATION

You work in the safety department of a real estate corporation monitoring IoT temperature sensor devices

Alert personnel of fires in the building: sensors that read above 40 degrees average in a one minute window is potentially a fire

(for solutions, see Canvas)

STATEFUL STREAM PROCESSING

STATEFUL STREAM PROCESSING

A lot of stream processing is stateful (output depends on not just one event, but previous event values)

Flink exposes powerful fault-tolerant state API to manage stream processing state

STATEFUL STREAM PROCESSING

EXAMPLE APPLICATION

You are working in the financial services industry; detect potentially fraudulent transactions using the following heuristic:

If a big transaction (>$500) is made right after a small one (<$0.10), transaction is fraudulent

(for solutions, see Canvas)

KEY POINT #2

Flink exposes rich APIs for general purpose stream processing, such as working with state, ML etc.

CONTENTS

- High Throughput Stream Processing
- Flink vs Kafka Streams
- Stateful Stream Processing

KEY POINTS

- Writing Flink jobs is very similar to writing Kafka Streams Applications!
- Flink exposes rich APIs for general purpose stream processing, such as working with state, ML etc.