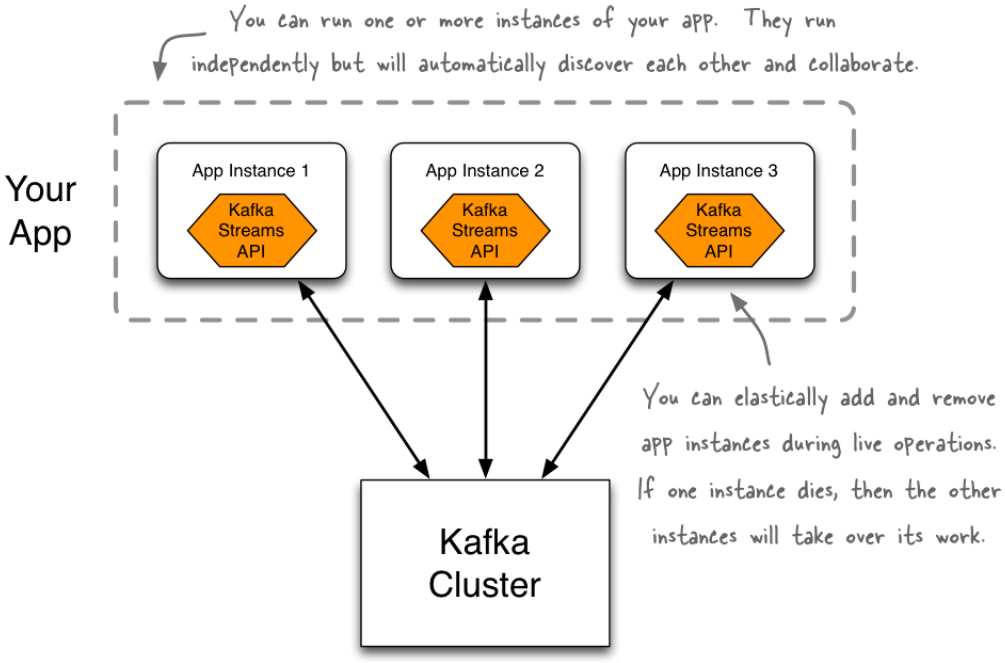
Introduction

This section provides a quick introduction to the Streams API of Apache Kafka®.

The Kafka Streams API in a Nutshell

The Streams API of Apache Kafka®, available through a Java library, can be used to **build highly scalable, elastic, fault-tolerant, distributed applications and microservices**. First and foremost, the Kafka Streams API allows you **to create real-time applications that power your core business**. It is the easiest yet the most powerful technology to process data stored in Kafka. It builds upon important concepts for stream processing such as efficient management of application state, fast and efficient aggregations and joins, properly distinguishing between event-time and processing-time, and seamless handling of late-arriving and out-of-order data.

A unique feature of the Kafka Streams API is that the applications you build with it are **normal Java applications**. These applications can be packaged, deployed, and monitored like any other Java application -- there is **no need to install separate processing clusters or similar special-purpose and expensive infrastructure**!



An application that uses the Kafka Streams API is a normal Java application. Package, deploy, and monitor it like you would do for any other Java application. Even so, your application will be highly scalable, elastic, and fault-tolerant.

Use Case Examples

The Kafka Streams API is applicable to a wide range of use cases and industries.

- Travel companies can build applications with the Kafka Streams API to make real-time decisions to find best suitable pricing for individual customers, to cross-sell additional services, and to process bookings and reservations.
- The finance industry can build applications to aggregate data sources for real-time views of potential exposures and for detecting and minimizing fraudulent transactions.
- Logistics companies can build applications to track their shipments fast, reliably, and in real-time.
- Retailers can build applications to decide in real-time on next best offers, personalized promotions, pricing, and inventory manage
- Automotive and manufacturing companies can build applications to ensure their production lines perform optimally, to gain real-time insights

into their supply chains, and to monitor telemetry data from connected cars to decide if an inspection is needed.
- And many more.

A Closer Look

Before we dive into the Concepts and Architecture or get our feet wet by walking through the Quick Start, let us take a first, closer look.

A key motivation of the Kafka Streams API is to bring stream processing out of the Big Data niche into the world of mainstream application development, and to radically improve the developer and operations experience by making stream processing simple and easy. Using the Kafka Streams API you can implement standard Java applications to solve your stream processing needs -- whether at small or at large scale -- and then run these applications on client machines at the perimeter of your Kafka cluster. Your applications are fully elastic: you can run one or more instances of your application, and they will automatically discover each other and collaboratively process the data. Your applications are also fault-tolerant: if one of the instances dies, then the remaining instances will automatically take over its work -- without any data loss! Deployment-wise you are free to chose from any technology that can deploy Java applications, including but not limited to Puppet, Chef, Ansible, Docker, Mesos, YARN, Kubernetes, and so on. This lightweight and integrative approach of the Kafka Streams API -- "Build applications, not infrastructure!" -- is in stark contrast to other stream processing tools that require you to install and operate separate processing clusters and similar heavy-weight infrastructure that come with their own special set of rules on how to use and interact with them.

The following list highlights several key capabilities and aspects of the Kafka Streams API that make it a compelling choice for use cases such as microservices, event-driven systems, reactive applications, and continuous queries and transformations.

Powerful

- Makes your applications highly scalable, elastic, distributed, fault-tolerant
- Supports exactly-once processing semantics
- Stateful and stateless processing
- Event-time processing with windowing, joins, aggregations
- Supports Interactive Queries to unify the worlds of streams and databases
- Choose between a declarative, functional API and a lower-level imperative API for maximum control and flexibility

Lightweight

- Low barrier to entry
- Equally viable for small, medium, large, and very large use cases
- Smooth path from local development to large-scale production
- No processing cluster required
- No external dependencies other than Kafka

Fully integrated

- 100% compatible with Kafka 0.11.0 and 1.0.0
- Easy to integrate into existing applications and microservices
- No artificial rules for packaging, deploying, and monitoring your applications
- Runs everywhere: on-premises, public clouds, private clouds, containers, etc.
- Integrates with databases through continous change data capture (CDC) performed by Kafka Connect

Real-time

- Millisecond processing latency
- Record-at-a-time processing (no micro-batching)
- Seamlessly handles late-arriving and out-of-order data
- High througput

Secure

- Supports encryption of data-in-transit
- Supports authentication and authorization

In summary, the Kafka Streams API is a compelling choice for building mission-critical stream processing applications and microservices. Give it a try and run your first Hello World application! The next sections Quick Start, Concepts, Architecture, and the Developer Guide will get you started.

Tip

Reading tip: If you are interested in learning about our original motivation to create the Kafka Streams API, you may want to read the Confluent blog post Introducing Kafka Streams: Stream Processing Made Simple.

Requirements

Kafka

The following versions are supported:

	Kafka Broker (columns)		
Streams API (rows)	3.0.x / 0.10.0.x	3.1.x / 0.10.1.x and 3.2.x / 0.10.2.x	3.3.x / 0.11.0.x and 4.0.x / 1.0.x and 4.1.x / 1.1.x and 5.0.x / 2.0.x and 5.1.x / 2.1.x and 5.2.x / 2.2.x and 5.3.x / 2.3.x
3.0.x / 0.10.0.x	compatible	compatible	compatible
3.1.x / 0.10.1.x and 3.2.x / 0.10.2.x		compatible	compatible
3.3.x / 0.11.0.x		compatible with exactly-once turned off (requires broker version Confluent Platform 3.3.x or higher)	compatible
4.0.x / 1.0.x and 4.1.x / 1.1.x and 5.0.x / 2.0.x and 5.1.x / 2.1.x and 5.2.x / 2.2.x and 5.3.x / 2.3.x		compatible with exactly-once turned off (requires broker version Confluent Platform 3.3.x or higher); requires message format 0.10 or higher	compatible; requires message format 0.10 or higher

The Streams API is not compatible with Kafka clusters running older Kafka versions (0.7, 0.8, 0.9).

Confluent

- When you need Avro schema support:Confluent Schema Registry

2 Votes

Last updated on Sep 10, 2019.