STREAMING AT SCALE

Distributed Stream Processing with Apache Flink

- Real-Time Analytics
- State & Exactly-Once
- Scalability & Operations
- Kubernetes & Cloud

ALLISON PARKER

Streaming at Scale

Distributed Stream Processing with Apache Flink

Steve Publications

This book is available at https://leanpub.com/streamingatscale

This version was published on 2026-09-10

Contents

Distributed Stream Processing with Apache Flink

This book is a comprehensive, technically rigorous guide to building production-grade real-time data processing systems with Apache Flink. It is written for software engineers, data engineers, platform engineers, and systems architects who need not only to use Flink effectively but to understand what it does underneath the APIs. You will learn how Flink achieves exactly-once processing guarantees through distributed snapshots, how event-time semantics enable deterministic results in the presence of out-of-order data, how stateful operators transform a simple dataflow framework into a platform capable of running complex analytics and fraud detection systems, and how to operate Flink clusters at scale with reliability and observability. The book progresses from foundational distributed systems concepts through Flink's architecture and APIs, then into advanced topics including performance tuning, deployment on Kubernetes, CDC pipelines, security, testing, and real-world architectural patterns. Every chapter includes complete, runnable code examples and practical guidance drawn from production experience.

Introduction: The Real-Time Imperative

In 2011, Nathan Marz published his thesis on the Lambda architecture, which became one of the most influential frameworks for handling large-scale data. The Lambda architecture proposed running data through two separate processing paths: a batch layer for correctness over historical data and a speed layer for low-latency views over recent data. The result was a complicated system with two codebases to maintain, two processing frameworks to operate, and a persistent challenge of keeping both paths producing consistent results.

Three years later, Jay Kreps wrote his famous response, Questioning the Lambda Architecture, which argued that a mature stream processing system backed by an immutable, replayable log could replace the batch layer entirely. The central insight was that if events are logged durably, historical computation becomes a matter of replaying those events through the same streaming logic. This became the Kappa architecture.

Both architectures were responses to the same fundamental shift: the growing value of latency. In many domains, the difference between knowing something in minutes and knowing it in seconds is the difference between opportunity and loss. Fraud detection must act on suspicious transactions before they clear. Recommendation engines must incorporate user behavior instantly to keep users engaged. Monitoring systems must detect anomalies in infrastructure before they cascade into outages. Machine learning features must reflect current reality to make accurate predictions.

Apache Flink emerged as a framework built specifically for this shift [3]. While other stream processing systems treated streaming as an afterthought, adding stateful processing capabilities to frameworks designed for batch, Flink was designed from the start as a stream processing engine. Its architecture, API, and execution model are all shaped by the needs of continuous, stateful computation over unbounded data.

This book is about understanding and mastering that architecture.

What You Will Learn

By the end of this book, you will be able to:

- Design and implement production-grade Flink applications using the DataStream and Table/SQL APIs.
- Configure event-time processing with appropriate watermark strategies to handle out-of-order and late data.
- Manage distributed state efficiently using keyed state, operator state, and state backends including RocksDB.
- Achieve exactly-once processing semantics through checkpointing and two-phase commit sinks.
- Build real-time analytics, fraud detection, and CDC pipelines with Flink and Kafka.
- Deploy Flink clusters on Kubernetes and other orchestration platforms with high availability.
- Monitor, debug, and tune Flink jobs for performance, latency, and resource utilization.
- Apply security, testing, and operational practices that keep Flink jobs reliable in production.

Prerequisites

This book assumes you are already comfortable with:

- Programming in Java or Python at a professional level.
- Basic distributed systems concepts such as partitioning, replication, and fault tolerance.
- Working with command-line tools and basic Linux administration.
- Familiarity with at least one message queue or streaming platform such as Apache Kafka.

You do not need prior experience with Flink or stream processing. The book builds concepts from the ground up.

How This Book Is Structured

The first section establishes the foundations. Chapters 1 through 3 cover the evolution of stream processing, the core concepts of stream processing as a discipline, and Flink's architecture. Chapter 4 takes you through building and running your first Flink application end-to-end.

The second section covers the core APIs and concepts. Chapters 5 through 9 focus on the DataStream API, event-time processing, windowing, state management, and fault tolerance. These chapters explain not only how each feature works but also why it exists and what happens internally.

The third section covers integration with external systems. Chapters 10 and 11 cover Kafka integration, connectors, CDC, and building custom sources and sinks. Chapter 12 introduces the Table API and SQL, and Chapter 13 covers stream joins and enrichment patterns.

The fourth section dives into advanced operations. Chapters 14 through 18 cover processing patterns, serialization, parallelism and backpressure, memory management, and performance tuning. These chapters provide the depth needed to operate Flink at scale.

The final section covers deployment and production operations. Chapters 19 through 22 cover deployment modes, observability, security, testing, architectural patterns, and anti-patterns. These chapters bring everything together into practical guidance for running Flink in real-world environments.

Version Notes

The code examples and API references in this book are based primarily on Apache Flink 1.18, a stable release that represents the mature Flink APIs as of 2024. Many concepts and APIs are stable across Flink versions, but some details have evolved. Where behavior differs between versions, this book calls those differences out explicitly.

Flink 2.0 was released in early 2025 with several breaking changes, including a new configuration file format, removal of some legacy APIs, and refined memory management. The fundamental architecture and programming model remain the same, so the concepts in this book apply directly. Code examples

use APIs available in Flink 1.18 that are compatible with Flink 2.x in their core form.

Throughout the book, inline citations reference official Flink documentation, research papers, and engineering publications that support the technical claims and guidance presented.

Chapter 1: The Era of Streaming Data

From Batch to Streaming: Why Latency Matters

Data processing has always been about extracting value from information. For most of computing history, that value was realized through batch processing: accumulate data over time, run a computation, and produce a result. Batch processing is simple, reliable, and efficient at scale. The fundamental trade-off is that results are only as fresh as the last batch run.

For many applications, staleness was acceptable. Daily sales reports, monthly financial statements, and weekly analytics dashboards all tolerate delays of hours or days. The business questions they answer are inherently retrospective. But as systems became more interconnected and decisions more time-sensitive, the cost of latency became visible.

Consider a payment processing system. When a credit card transaction arrives, the system must decide within milliseconds whether to approve or decline it. If the fraud detection model only runs hourly in batch mode, it is blind to fraud patterns that emerged in the last hour. By the time the batch analysis completes, fraudulent transactions have already been processed and money lost.

Consider a ride-sharing application. The matching algorithm must pair drivers with riders based on their current locations and recent behavior. If driver positions are updated in 15-minute batches, the application cannot provide accurate arrival estimates or efficient routing.

Consider an IoT sensor network monitoring industrial equipment. A temperature sensor reading that exceeds a threshold must trigger an immediate shutdown to prevent damage. Batch processing cannot provide the sub-second response required.

These are not edge cases. They represent a fundamental class of problems where the value of information decays rapidly with time. The shorter the interval between an event occurring and the system acting on it, the more value that action creates. This is the real-time imperative.

The Lambda Architecture and Its Pain Points

Nathan Marz formalized the challenge in 2011 with the Lambda architecture. The Lambda architecture separates data processing into three layers:

1. The batch layer processes all historical data using a batch processing framework such as Hadoop MapReduce to produce the batch view, which is the authoritative, comprehensive result.
2. The speed layer processes only the most recent data using a low-latency stream processing framework such as Storm to produce the speed view, which fills in the gap between the last batch run and the present moment.
3. The serving layer merges results from both views to answer queries with both freshness and completeness.

The Lambda architecture was elegant in theory. It acknowledged that early stream processors were not mature enough for complex, exactly-once processing, so it relied on the batch layer for correctness and the speed layer for latency. It became widely adopted and influenced how many organizations structured their data pipelines.

In practice, the Lambda architecture introduced significant operational complexity. Organizations had to maintain two complete processing pipelines running different code on different frameworks. The batch code and streaming code had to produce consistent results, which required duplicating business logic and continuously verifying that both paths agreed. When requirements changed, engineers had to update both codebases in lockstep.

Disney documented these pain points explicitly when they evaluated their streaming architecture. The batch and streaming sides required different codebases maintained in sync, so the same data was processed at least twice, increasing cost and operational burden across storage, network, and compute resources [1]. The complexity of keeping the two paths aligned often led to inconsistencies that were difficult to diagnose.

From Lambda to Kappa: Unified Processing Models

In 2014, Jay Kreps published Questioning the Lambda Architecture, arguing that advances in stream processing made the dual-layer approach unnecessary

[2]. The Kappa architecture eliminates the batch layer entirely. All data is treated as a stream. The system writes every event to an immutable, durable log, typically Apache Kafka. Stream processing logic runs over this log continuously to produce real-time views. When historical recomputation is needed, such as when fixing a bug or changing business logic, the same streaming application replays from the beginning of the log.

The Kappa architecture offers several advantages:

- Single codebase: the same streaming logic handles both real-time processing and historical recomputation.
- Simpler operations: only one processing framework to maintain and scale.
- Consistency: results are inherently consistent because there is only one computation path.
- Flexibility: data is retained in the log, enabling retrospective analysis and reprocessing.

The critical assumption behind Kappa is that the stream processor must be capable of handling complex stateful computations with strong consistency guarantees. Early stream processors could not. Flink was built to fill this gap.

Core Challenges of Distributed Stream Processing

Building a production-grade stream processor that can replace batch requires solving several hard distributed systems problems:

Time and ordering. Events arrive out of order due to network delays, parallel processing, and system failures. A correct stream processor must handle this disorder deterministically, ensuring that results are based on when events actually occurred, not when they arrived. This requires a rigorous model of time and mechanisms for tracking progress through that time.

State. Real computations require remembering past events. Counting transactions per user requires maintaining a counter for each user. Detecting fraud patterns requires tracking recent behavior. Detecting anomalies requires knowing what is normal. Maintaining this state across a distributed cluster while ensuring consistency and fault tolerance is non-trivial.

Fault tolerance. Stream processors must run continuously for weeks or months without data loss or duplication. When a node fails, the system

must recover seamlessly, restoring state and continuing from where it left off without manual intervention. Achieving this requires distributed snapshot protocols that capture consistent state across all nodes without stopping processing.

Exactly-once semantics. Many applications require that each event affects the final result exactly once, not zero times and not twice. This guarantee must extend end-to-end, from source through processing to sink, including external systems such as Kafka topics and databases.

Scalability. Throughput must scale with data volume, and the system must handle load spikes without degradation. Scaling must not require losing state or restarting processing.

Latency. The system must process events with minimal delay. End-to-end latency, from event arrival to result availability, should be measured in milliseconds, not seconds or minutes.

Where Flink Fits in the Streaming Landscape

Apache Flink is one of several stream processing frameworks in use today. The most common alternatives are Apache Spark Structured Streaming, Apache Storm, and Kafka Streams. Each takes a different approach to the challenges above.

Apache Storm was the first widely adopted distributed stream processor, introduced in 2011 by Twitter. Storm excels at raw throughput and low latency but provides limited support for event-time processing and stateful computation. Its fault tolerance model is at-most-once, which means events can be lost during failures, making it unsuitable for applications requiring strong guarantees.

Apache Spark Structured Streaming is an extension of Spark's batch processing engine to handle streaming data. Spark uses microbatching: it divides the stream into small batches and processes each batch with Spark's optimized execution engine. This approach leverages Spark's maturity and performance but introduces latency proportional to the microbatch interval. While Spark 2.3 introduced continuous processing mode for sub-millisecond latency, it remains limited in feature support compared to microbatching.

Kafka Streams is a client library that runs stream processing logic directly within Kafka consumer applications. It is lightweight and easy to deploy since

it requires no separate cluster. However, it is tightly coupled with Kafka, and scaling it requires managing application deployments across infrastructure. It does not provide the same level of state management and fault tolerance sophistication as dedicated stream processors.

Flink distinguishes itself in several ways. First, it is a native stream processor, not a batch system extended to handle streaming. Its execution model is true streaming with event-by-event processing, not microbatching. Second, it provides mature support for event-time semantics, sophisticated windowing, and stateful computation with exactly-once guarantees. Third, its checkpointing mechanism based on the Chandy-Lamport algorithm enables fault tolerance without sacrificing performance. Fourth, its architecture separates the execution graph from the physical deployment, enabling flexible resource management and deployment modes.

Flink was created by researchers at University of Berlin in 2010 as a successor to Apache Storm with improved fault tolerance and state management. It joined the Apache Software Foundation in 2014 as Apache Flink and has since become one of the most widely adopted stream processing frameworks, used by companies including Alibaba, Amazon, Tencent, and Shopify for critical production workloads.

The rest of this book will explain how Flink solves the challenges of distributed stream processing in detail, and how to use it to build production-grade real-time applications.

Summary

This chapter established the context for stream processing as a discipline. Batch processing dominated for decades because it was simple and effective for its era. The Lambda architecture attempted to add real-time capabilities on top of batch but introduced complexity that many organizations found unsustainable. The Kappa architecture argued that a sufficiently capable stream processor could replace batch entirely. Apache Flink was designed to be that stream processor.

The next chapter builds on this foundation by introducing the core concepts of stream processing: data streams, event time, ordering, partitioning, and fault tolerance models. Understanding these concepts is essential before diving into Flink's specific implementations.

Chapter 2: Stream Processing Fundamentals

Data Streams, Events, and Event Streams

In stream processing, the fundamental unit of data is the event. An event is a single occurrence in the real world represented as data: a user clicking a button, a sensor recording a temperature, a credit card transaction being processed, a row being inserted into a database. Each event carries a timestamp indicating when it occurred and a set of attributes describing what happened.

A data stream is an ordered sequence of events. In distributed systems, this stream is typically partitioned and stored in a durable log, such as an Apache Kafka topic. Each partition is an independent, ordered sequence, and together the partitions form the complete stream. Events are append-only: once written, they are never modified or deleted, though they may eventually expire based on retention policies.

This model contrasts sharply with traditional batch processing, where data is stored in tables or files that can be updated, and where computation produces a new table or file from the old one. In stream processing, data flows continuously through a pipeline, and computation transforms events as they arrive. The result is also a stream.

Stream vs Batch: Execution Models and Mindset

The difference between batch and stream processing is not merely about latency. It is a fundamental difference in execution model and mindset.

In batch processing, the system starts with a finite dataset, runs a computation over it, and produces a finite result. The computation can be retried from scratch without loss of correctness because the input is immutable and the output is replaced entirely each time. Failures are handled by restarting

the job. The system knows when it is done: when it has processed all input data.

In stream processing, the input is unbounded. The system never finishes processing because events continue to arrive indefinitely. Failures cannot be handled by restarting from the beginning because that would mean reprocessing months or years of data. Instead, the system must maintain state, track progress, and recover from failures by restoring to a consistent point and continuing forward.

This difference has implications for how you design applications. In batch processing, you can assume all data is available and sort or repartition freely. In stream processing, you can only work with events as they arrive in order within each partition. Global ordering across partitions is impossible without serialization, which defeats the purpose of distributed processing.

Consider counting events per key in both models. In batch, you can partition by key, count within each partition, then merge the counts. In streaming, you maintain a running count for each key that updates with every new event. The streaming approach requires state, fault tolerance for that state, and mechanisms to produce intermediate results continuously rather than waiting for a final answer.

Ordering, Latency, and Throughput in Distributed Systems

Ordering is one of the most subtle aspects of stream processing. Within a single partition, events are strictly ordered. If event A is written before event B, all readers will see A before B. Across partitions, no global ordering is guaranteed. Events with different keys may be processed in any order relative to each other.

This partition-level ordering is a deliberate design choice that enables parallelism. If we required global ordering, all events would have to flow through a single processing thread, limiting throughput to the capacity of one node. By relaxing ordering to partition level, we can process each partition independently on different nodes and scale throughput linearly with the number of partitions.

Latency and throughput are often in tension. Throughput is the volume of events processed per unit of time, typically measured in events per second or

bytes per second. Latency is the time between an event arriving at the system and the result of processing it becoming available. Systems can optimize for one at the expense of the other.

High throughput typically requires batching: accumulating many events and processing them together to amortize overhead. This increases latency because events wait in a buffer before being processed. Low latency requires processing events as soon as they arrive, which increases overhead per event and limits throughput.

Stream processors tune this trade-off through configuration. Buffer sizes determine how many events accumulate before being processed. Batch intervals determine how long the system waits before flushing a batch. Memory allocations determine how much state can be held in fast memory versus slower disk. These knobs are not academic settings: they directly affect whether a system meets its service level objectives.

Partitioning and Shuffling in Stream Processing

Partitioning is how distributed systems distribute data across nodes for parallel processing. When data enters the stream processor, it is partitioned by some strategy. The most common strategy is key-based partitioning, where a hash function maps each event's key to a specific partition. All events with the same key go to the same partition, ensuring that events for a particular entity are processed in order.

Shuffling is the process of redistributing data between operators in a computation graph. When one operator produces data that must be consumed by another operator in a different partitioning, the system must move data across the network to match the required distribution. This is called a shuffle, and it is one of the most expensive operations in distributed processing because it consumes network bandwidth and serialization resources.

Common shuffle strategies include:

- Key-by shuffle: events are redistributed so that all events with the same key go to the same downstream partition. This is used for key-based operations such as aggregations and joins.
- Random shuffle (rebalance): events are distributed randomly across all downstream partitions, typically for load balancing.

- Broadcast shuffle: all events are sent to all downstream partitions, typically for joining a small lookup table with a large stream.
- Forwarding: events go directly to the corresponding downstream partition without redistribution, used when operators are chained and share the same partitioning.

The choice of shuffle strategy affects both correctness and performance. Using the wrong strategy can produce incorrect results, such as events for the same key being processed by different parallel instances that do not share state. Using an inefficient strategy, such as broadcasting a large dataset, can overwhelm the cluster.

Fault Tolerance Models: At-Most-Once, At-Least-Once, Exactly-Once

Fault tolerance models describe what happens to data processing when failures occur. There are three standard models:

At-most-once: each event is processed zero or one time. On failure, the system restarts without restoring state, so some events may be lost. At-most-once is simple to implement but unacceptable for most production applications because it can drop events permanently.

At-least-once: each event is processed one or more times. On failure, the system restores state and continues from the last known position, which may mean reprocessing some events. At-least-once is easier to implement than exactly-once and acceptable when idempotent operations or downstream deduplication handle duplicates.

Exactly-once: each event is processed exactly one time. This is the strongest guarantee and the most difficult to achieve, particularly end-to-end across external systems. Exactly-once does not mean events are never duplicated during recovery: it means that the final observable result is the same as if every event was processed exactly once without any failures.

Exactly-once semantics are often misunderstood. They do not require that each event is read once from the source, transformed once, and written once to the sink. They require that the final result, after all processing including recovery from failures, is identical to what it would be if no failures

ever occurred. This is achieved through combinations of idempotent writes, transactional commits, and stateful deduplication.

Flink supports all three models through its checkpointing mechanism. The choice of model affects both the correctness guarantees and the operational overhead. Exactly-once requires more complex configuration and slightly higher checkpoint costs, but it eliminates the need for downstream deduplication logic and provides strong guarantees that are essential for financial systems and analytics pipelines.

The Role of Time in Stream Processing

Time is the most subtle and most important concept in stream processing. In batch processing, time is straightforward: the system knows when it starts, when it finishes, and all data is available at once. In stream processing, events arrive continuously, and their timestamps are determined by when they occurred in the real world, not when the system processed them.

Consider a clickstream application that counts page views per minute. Events arrive with timestamps indicating when the user clicked. However, events may arrive late due to network delays, client clock skew, or processing bottlenecks. A click that occurred at 10:00:30 may not arrive at the processor until 10:00:45 or even 10:01:00. If the system uses the arrival time to assign the click to a minute, the count will be wrong.

This is the distinction between event time and processing time:

- Event time is when the event actually occurred, as recorded by the event's timestamp.
- Processing time is when the system processes the event, as recorded by the system clock.

Processing time is simple but produces incorrect results in the presence of any delay. Event time produces correct results but requires the system to handle out-of-order events and late arrivals, because events do not necessarily arrive in timestamp order.

The challenge of event-time processing is determining when it is safe to produce results for a given time window. If the window is 10:00 to 10:01, and events have been arriving with timestamps in that range, at what point does

the system know it has seen all events for that window? In a distributed system with unpredictable delays, there is no absolute answer.

Stream processors solve this with watermarks, a mechanism for tracking progress through event time. Watermarks are signals that indicate the system has seen all events up to a certain timestamp. When the watermark advances past the end of a window, the window is closed and results are produced. Events that arrive after the watermark has passed are considered late and may be dropped, processed in a special way, or included with allowed lateness configurations.

Event-time processing is fundamental to deterministic, correct results. Without it, stream processing applications produce answers that depend on the whims of network timing rather than on the actual data. Every stream processor worth using supports event-time processing, and Flink's event-time model is one of its core strengths.

Summary

This chapter introduced the fundamental concepts of stream processing: events and streams as continuous flows of data, the difference between stream and batch execution models, partitioning and shuffling as mechanisms for distributing computation, fault tolerance models describing what happens during failures, and time semantics as the foundation for correct results.

The next chapter introduces Flink's architecture, showing how these abstract concepts map to concrete components: the JobManager, TaskManagers, job graphs, execution graphs, and the distributed execution model that makes Flink what it is.

Chapter 3: Flink Architecture Overview

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

The Flink Runtime: Components and Responsibilities

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Job Submission: From Application Code to Running Job

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

The Job Graph: Nodes, Edges, and Stream Transformations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

From Job Graph to Execution Graph

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Task Slots, Parallelism, and Resource Allocation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

The Life Cycle of a Flink Job

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Chapter 4: Building Your First Flink Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Development Environment and Dependencies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Project Structure for a Flink Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Writing a WordCount Streaming Job

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Running Locally and Inspecting Results

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Understanding the Generated Job Graph

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Packaging and Submitting to a Cluster

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Chapter 5: The DataStream API

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Sources and Sinks: The Boundaries of Your Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Core Transformations: Map, FlatMap, Filter, KeyBy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Reduction and Aggregation Operators

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Process Functions and Fine-Grained Control

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Operator Chaining and Pipelining

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Side Outputs and Branching Streams

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Chapter 6: Time, Timestamps, and Watermarks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Processing Time, Event Time, and Ingestion Time

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Why Event Time Matters for Correct Results

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Assigning Timestamps to Streams

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Watermarks: The Heart of Event-Time Processing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Watermark Strategies and Generators

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Handling Out-of-Order Events

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Chapter 7: Windowing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

The Concept of Windows

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Time Windows: Tumbling, Sliding, and Session

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Count and Global Windows

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Triggers: Controlling When Windows Fire

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Evictors and Window Functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Late Data and Allowed Lateness

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Chapter 8: State Management Fundamentals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Why State Changes Everything

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Keyed State vs Operator State

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

State Types: Value, List, Map, Reducing, Aggregating

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Accessing and Updating State

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

State Backends: Memory, FileSystem, and RocksDB

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Choosing the Right State Strategy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Chapter 9: Fault Tolerance: Checkpoints and Savepoints

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

The Distributed Snapshot Problem

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Checkpointing in Flink: The Barrier Flow Algorithm

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Configuration and Tuning Checkpoints

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Fault Recovery and State Restoration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Savepoints: Managed Checkpoints for Operations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Exactly-Once, At-Least-Once, and At-Most-Once Guarantees

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Chapter 10: Kafka Integration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Why Kafka and Flink Are a Natural Pair

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

The Flink Kafka Consumer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

The Flink Kafka Producer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Offset Management and Checkpointing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Partitioning Semantics and Ordering

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Kafka Integration Patterns and Best Practices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Chapter 11: Connectors and External Systems

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Connector Architecture and Interfaces

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Database Connectors: JDBC, Elasticsearch, Redis

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

File System Connectors: HDFS, S3, Local

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Change Data Capture: Debezium and Flink CDC

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Building Custom Sources and Sinks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Choosing Connectors for Your Use Case

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Chapter 12: The Flink Table API and SQL

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

The Table API: Declarative Stream Processing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Table Environments and Execution Mode

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Defining Tables and Schemas

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

SQL for Streaming: Queries and Semantics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Bridging DataStream and Table APIs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

When to Use SQL vs DataStream

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Chapter 13: Stream Joins and Enrichment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Joining Event Streams: The Challenge

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Stream-Stream Joins: Interval and Window Joins

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Stream-Table Joins: Enriching with Reference Data

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Temporal Table Joins and Schema Evolution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Broadcast State for Lookup and Enrichment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Handling Asynchrony and Skewed Data

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Chapter 14: Advanced Data Processing Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Deduplication and Idempotency

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Anomaly Detection on Streams

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Sessionization and User Activity Tracking

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Multi-Dimensional Aggregations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Multi-Stage Pipelines and Topology Design

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Real-Time Feature Computation for ML

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Chapter 15: Serialization and Data Formats

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Flink's Type System

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Serialization: Kryo, Java, Generic Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Optimizing Serialization Performance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Avro, Protobuf, and Schema Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Schema Evolution in Streaming

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Binary Formats and Network Efficiency

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Chapter 16: Parallelism, Backpressure, and Scaling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Parallelism Model and Task Allocation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Understanding and Detecting Backpressure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Rescaling Jobs Without Losing State

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Data Skew and Its Impact

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Slot Sharing and Co-Localization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Elastic Scaling Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Chapter 17: Memory Management and Resource Tuning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Flink's Memory Layout

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Task, Network, and Managed Memory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Tuning TaskManager and JobManager Resources

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

JVM Garbage Collection and Flink

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Monitoring Memory Pressure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Memory-Related Failure Modes

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Chapter 18: Performance Tuning and Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Latency vs Throughput Trade-Offs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Optimizing State Access and Size

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Checkpoint Optimization Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Network Shuffle Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Operator-Level Optimization Techniques

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Profiling and Diagnosing Performance Problems

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Chapter 19: Deployment and Operations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Deployment Modes: Standalone, YARN, Kubernetes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Cluster Modes: Session, Application, Per-Job

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

High Availability Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Resource Management and Cluster Sizing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Configuration Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Production Deployment Checklist

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Chapter 20: Observability, Monitoring, and Debugging

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

The Flink Web Dashboard

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Built-In Metrics and Custom Metrics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Integrating with Prometheus and Grafana

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Logging Strategies for Distributed Jobs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Debugging Techniques and Tools

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Incident Response and Troubleshooting

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Chapter 21: Security, Testing, and Reliability

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Authentication and Authorization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Encryption in Transit and at Rest

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Network Security and Kerberos

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Testing Flink Applications

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

CI/CD Pipelines for Flink Jobs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Production Reliability Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Chapter 22: Production Architecture Patterns and Anti-Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

End-to-End Real-Time Analytics Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Fraud Detection Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

CDC-Based Event-Driven Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Multi-Tenant Flink Platforms

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Common Anti-Patterns and How to Avoid Them

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Version Upgrades and Long-Term Maintenance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

Conclusion: Principles for Streaming at Scale

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.

References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/streamingatscale.