

Stream Processing with PyFlink and Iceberg

Python pipelines, event time and analytic-table integration

FA-0680 · 2 days

Data & Analytics · DevOps, Cloud & Infrastructure

Introduction

A two-day practical introduction to stream processing with PyFlink and analytic tables with Apache Iceberg. Develop one small Python-based pipeline in a prepared compatible environment and examine state, time semantics, recovery and table writes.

Cluster-deployment choices and advanced optimisation are comparative demonstrations. End-to-end latency, delivery guarantees and performance depend on configuration, sources, sinks and workload; the course does not establish a production-ready real-time platform.

Learning outcomes

- Explain Flink runtime roles and build a small PyFlink DataStream application.
- Use introductory keyed state, event-time and window concepts.
- Describe checkpoint/recovery behaviour and the conditions behind delivery guarantees.
- Configure a prepared Iceberg catalog/table integration and inspect supported writes/queries.
- Identify connector compatibility, late-data and operational limitations.

Prerequisites

- Python programming proficiency and environment/package-management skills.
- Distributed-systems/parallel-processing foundations and basic data-framework familiarity.
- SQL and relational-data concepts.

- Use the arranged mutually compatible PyFlink/Python, Flink/JDK, Iceberg connector and catalog environment; Python code still depends on the Flink runtime and connector JARs.

Training outline

1. Day 1 — PyFlink Foundations, Runtime and a Stateful Pipeline

Stream-processing Concepts

- The Evolution of Data Processing: From Batch to Stream
- Key Concepts in Stream Processing

Overview of Apache Flink

- Flink's Position in the Big Data Ecosystem
- Core Features and Capabilities
- Use Cases and Industry Applications

Architecture and Recovery

Flink's Distributed Architecture

- JobManager and TaskManager Roles
- Execution Model and Dataflow

Fault Tolerance Mechanisms

- Checkpointing and State Snapshots
- Exactly-once state processing and the conditions for end-to-end guarantees

Deployment Modes

- Standalone Cluster
- Kubernetes and other supported deployment options; check the selected release for YARN compatibility

Distinguish internal state consistency from end-to-end guarantees; checkpointing and connector/sink behaviour must be configured appropriately.

DataStream Practice

- Setting Up the Python Development Environment for PyFlink

Building a Simple PyFlink Application

- Defining Data Sources and Sinks in Python

- Applying Transformations with PyFlink

Understanding Streams and Transformations

- Stateless vs. Stateful Transformations
- Keyed Streams and Partitioning

State and Timers

- The Importance of State in Stream Processing
- Selected keyed-state examples and an overview of operator state; verify API support in the arranged PyFlink release

Implementing Process Functions

- Timers and Event Time Processing

State Backends and Their Configurations

- HashMapStateBackend and heap-state considerations
- EmbeddedRocksDBStateBackend and selected alternatives supported by the lab release

Use selected keyed-state/timer examples and compare supported state backends against the lab version; distinguish Python-supported operations from Java-only APIs.

2. Day 2 – Event Time, Windows and Iceberg Integration

Time and Windowing

- Understanding Event Time vs. Processing Time
- Generating and Assigning Timestamps in Python
- Watermarks and Their Role in Event Time Processing

Window Operators and Functions in PyFlink

- Tumbling Windows
- Sliding Windows
- Session Windows
- Late Data Handling Strategies

Iceberg Tables and SQL

Introduction to Apache Iceberg

- Motivation and Key Features
- Comparison with Traditional Table Formats

Setting Up Iceberg with PyFlink

- Configuring Iceberg Catalogs in Python
- Creating and Managing Iceberg Tables

Performing Data Operations

- Supported append, overwrite and upsert/delete semantics; connector and table-format constraints
- Schema evolution and partition design, with engine-specific support checks

Querying Iceberg Tables with Flink SQL using PyFlink

- Writing and Executing SQL Queries
- Inspect query plans and selected performance considerations; no universal optimisation result

Use one bounded stream-to-table exercise. Compare append, overwrite and upsert/delete semantics only where supported by the connector/table format; streaming INSERTOVERWRITE is not a supported substitute for batch overwrite.

Review and Operational Limits

Inspect late events, checkpoint recovery and a selected failure case. Identify further monitoring, retention, table maintenance and deployment validation needed before production use.