QCon+ 案例研习社

如何基于 Flink + Iceberg 构建实时数据湖

阿里巴巴 胡争（子毅）

Apache HBase PMC, Apache Iceberg Committer.
《HBase原理与实践》作者

1 什么是数据湖？

2 Flink: 为何选择Apache Iceberg？

3 Flink + Iceberg 经典场景

4 Apache Flink如何集成Apache Iceberg？

5 社区规划

1 什么是数据湖？

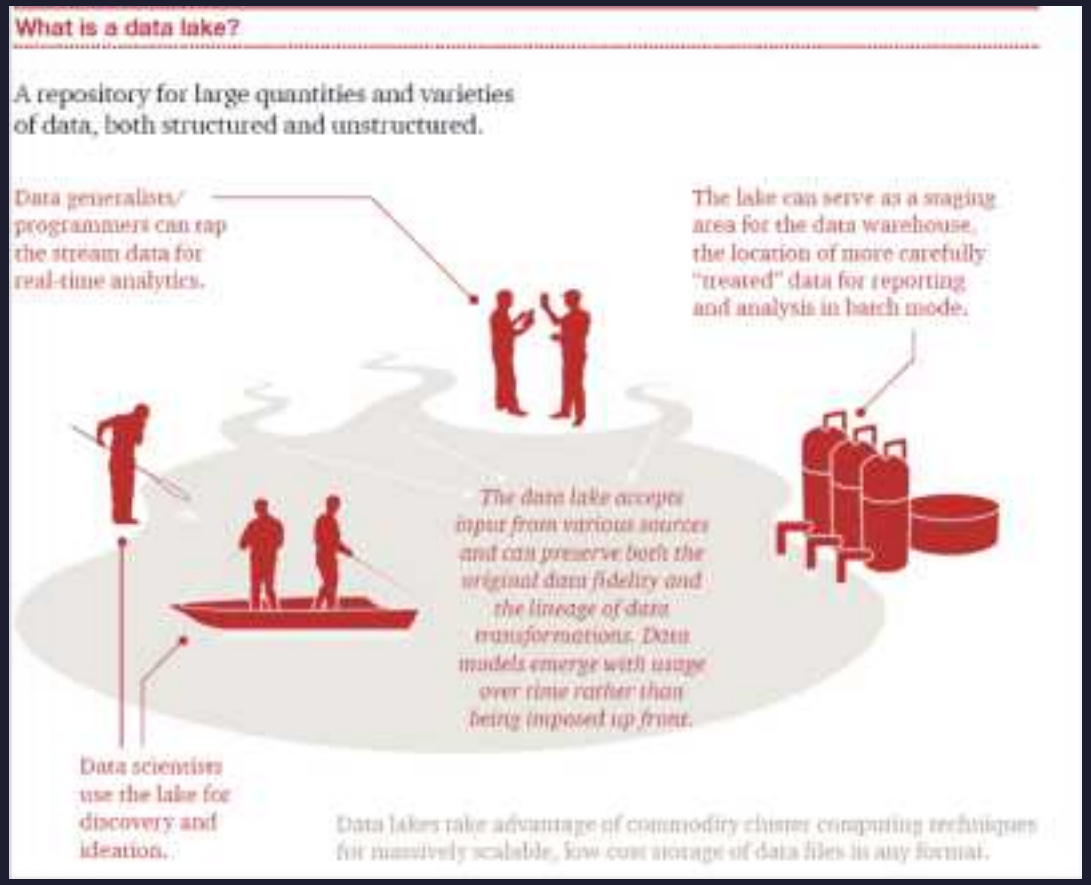
2 Flink: 为何选择Apache Iceberg？

3 Flink + Iceberg 经典场景

4 Apache Flink如何集成Apache Iceberg？

5 社区规划

什么是数据湖？



存储原始数据

结构化数据
半结构化数据
非结构化数据
二进制数据(图片等)

多种计算模型

批处理
流计算
交互式分析
机器学习

完善的数据管理

多种数据源接入
数据连接
Schema管理
权限管理

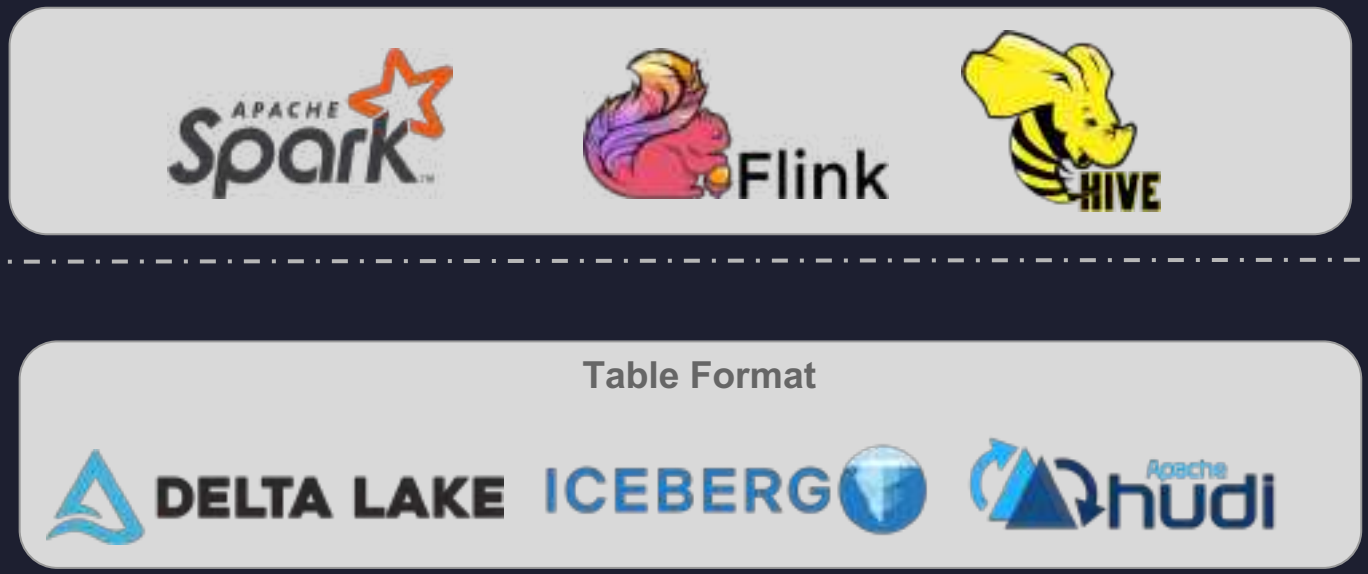
灵活的底层存储

S3/OSS/HDFS
Parquet/Avro/Orc
数据缓存加速
轻量级索引

数据仓库 VS 数据湖

特征	数据仓库	数据湖
数据	来自事务系统、运营数据库和业务线应用程序的关系数据	来自 IoT 设备、网站、移动应用程序、社交媒体和企业应用程序的非关系和关系数据
Schema	Schema设计在数据仓库实施之前（写入型 Schema）	分析时定义Schema（读取型 Schema）
性价比	更快查询结果会带来较高存储成本	更快查询结果只需较低存储成本
数据质量	可作为重要事实依据的高度监管数据	任何可以或无法进行监管的数据（例如原始数据）
用户	业务分析师	数据科学家、数据开发人员和业务分析师（使用监管数据）
分析	批处理报告、BI 和可视化	机器学习、预测分析、数据发现和分析

开源数据湖架构



1 什么是数据湖？

2 Flink: 为何选择Apache Iceberg？

3 Flink + Iceberg 经典场景

4 Apache Flink如何集成Apache Iceberg？

5 社区规划

Delta、Hudi、Iceberg对比

Iteams	Open Souce Delta	Apache Iceberg	Apache Hudi
Open Source Time	2019/04/12	2018/11/06(incubation)	2019/01/17(incubation)
Github Star	2800+	692	1400+
Releases	5	5	48
ACID	Yes	Yes	Yes
Isolation Level	Write/Snapshot serialization	Write Serialzation	Snapshot serialization
Time Travel	Yes	Yes	Yes
Row-level DELETE (batch)	Yes	Ongoing	No
Row-level DELETE (streaming)	No	Ongoing	Yes
Abstracted Schema	No	Yes	No
Engine Pluggable	No	Yes	No
Open File Format	Yes	Yes	Yes(Data) + No(Log)
Filter push down	No	Yes	No
Auto-Compaction	No	Ongoing	Yes
Python Support	Yes	Yes	No
File Encryption	No	Yes	No

Delta、Hudi、Iceberg对比

Databricks Delta

Open Source Delta

Apache Hudi

Apache Iceberg

Apache Iceberg 核心优势

通用化标准设计

完美解耦计算引擎
Schema标准化
开放的数据格式
支持Java和Python

完善的Table语义

Schema定义与变更
灵活的Partition策略
ACID语义
Snapshot语义

丰富的数据管理

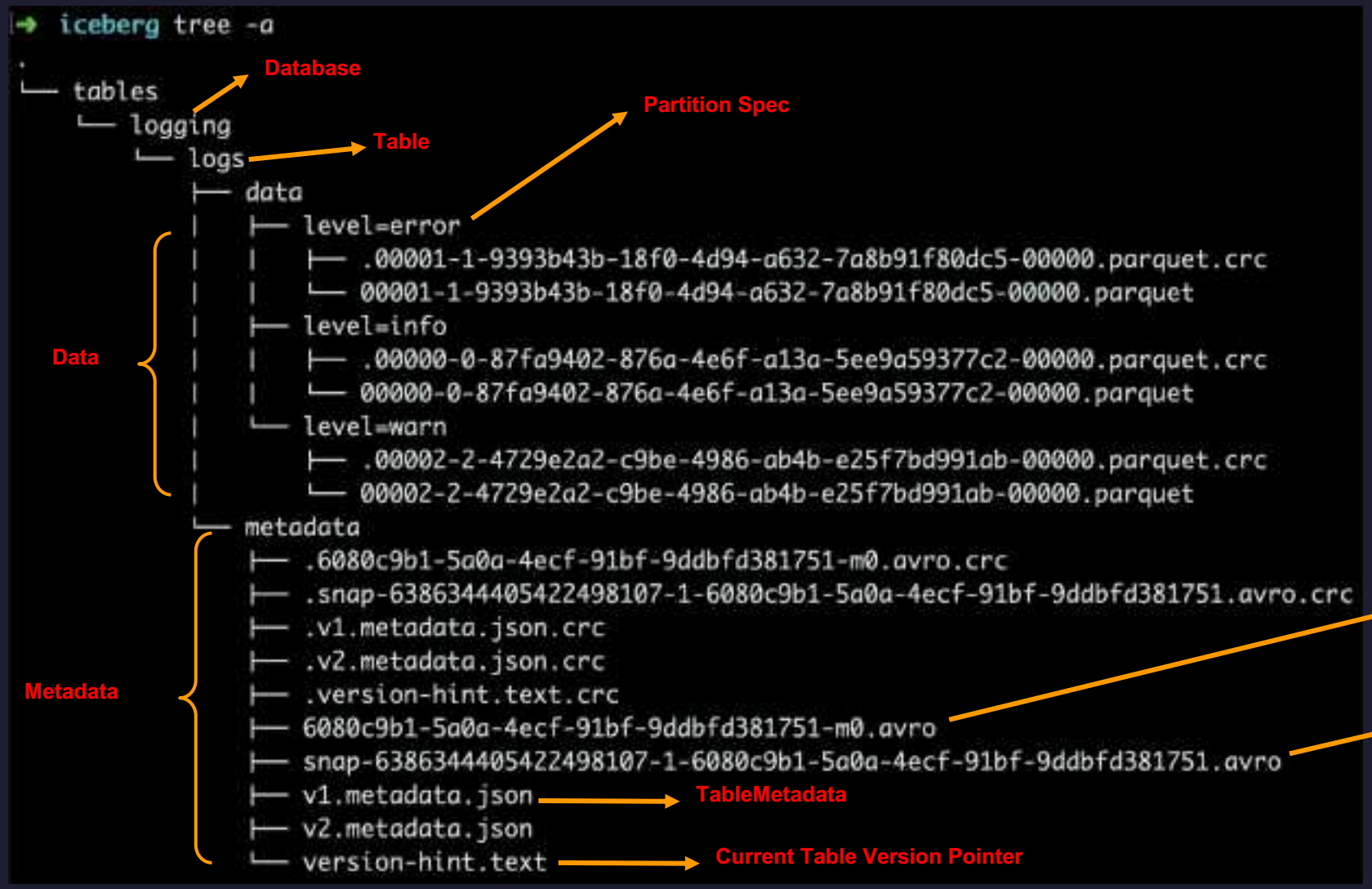
存储的流批统一
可扩展的META设计
支持批更新和CDC
支持文件加密

性价比

计算下推设计
低成本的元数据管理
向量化计算
轻量级索引

选择Apache Iceberg的公司

Apache Iceberg 文件分布



QCon+

1 什么是数据湖？

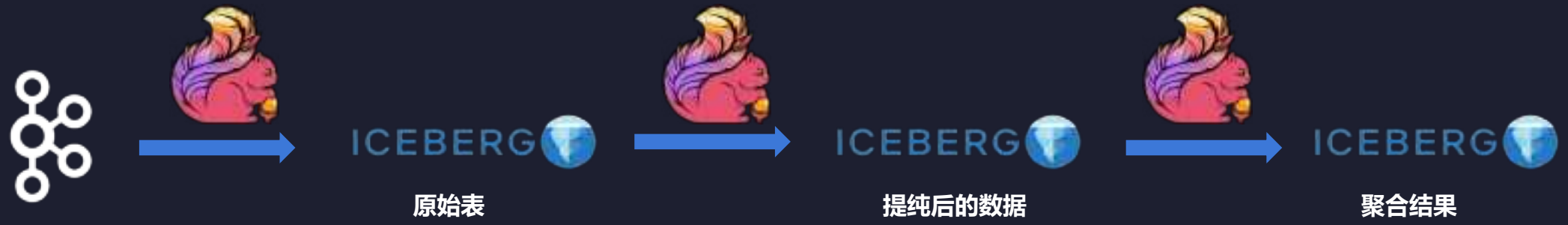
2 Flink: 为何选择Apache Iceberg？

3 Flink + Iceberg 经典场景

4 Apache Flink如何集成Apache Iceberg？

5 社区规划

场景一: 构建近实时Data Pipeline

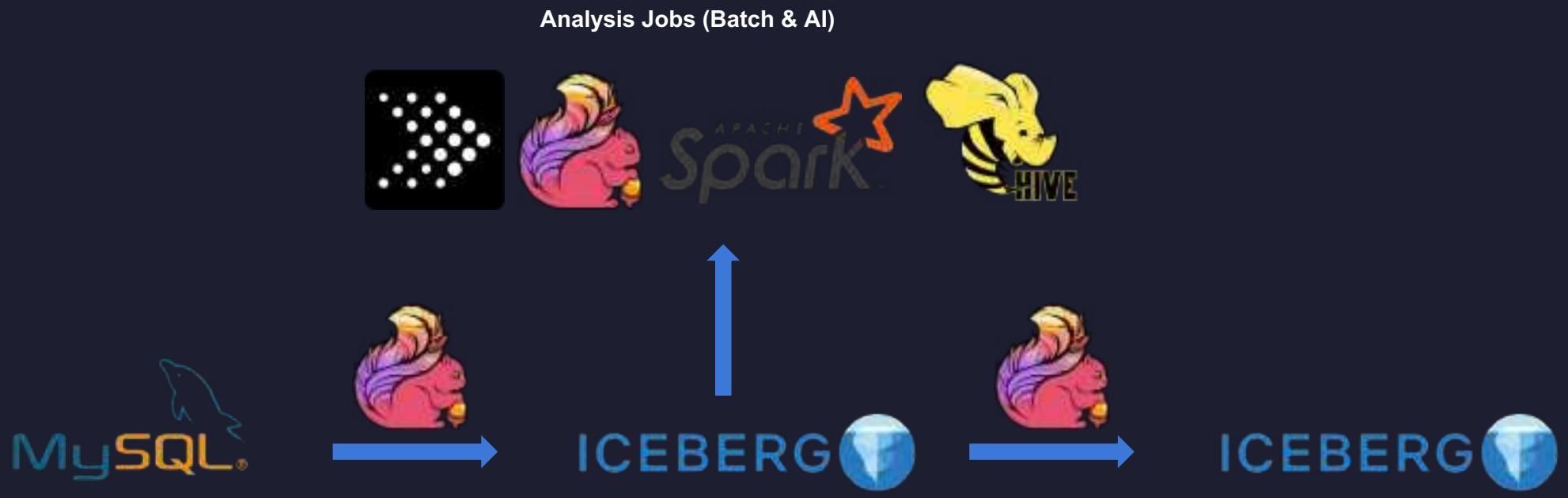


原始表

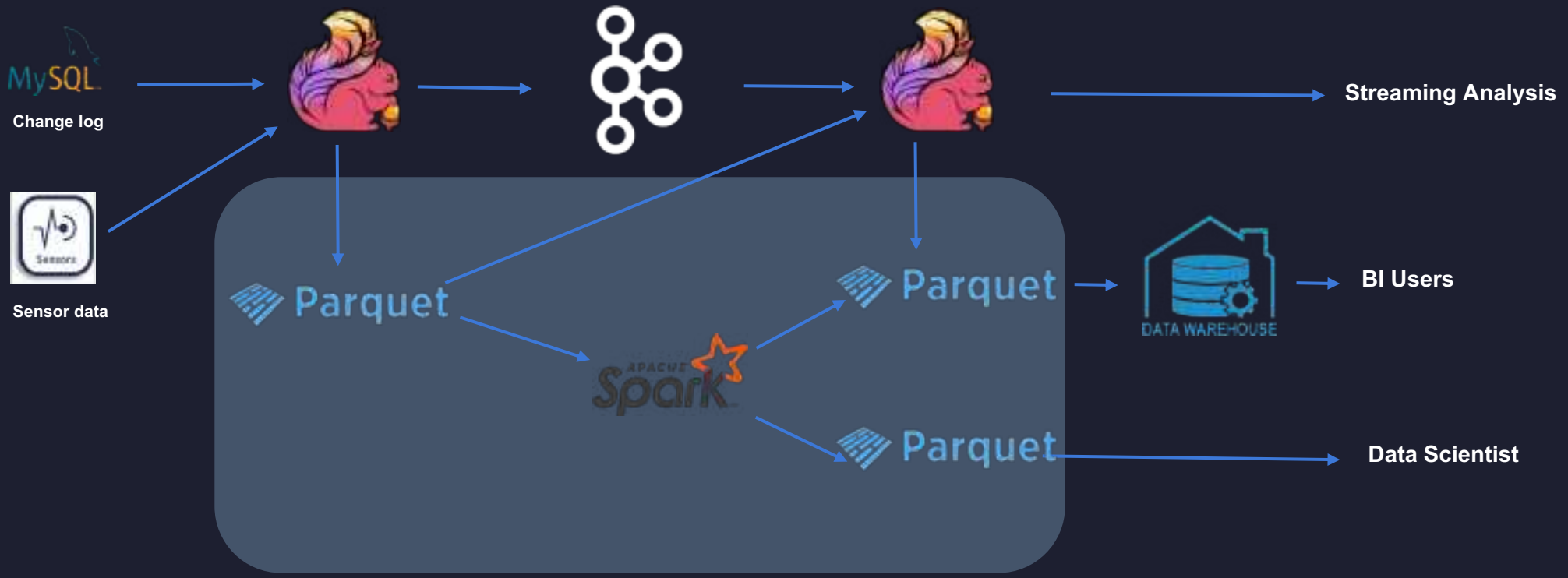
提纯后的数据

聚合结果

场景二: CDC数据实时摄入摄出

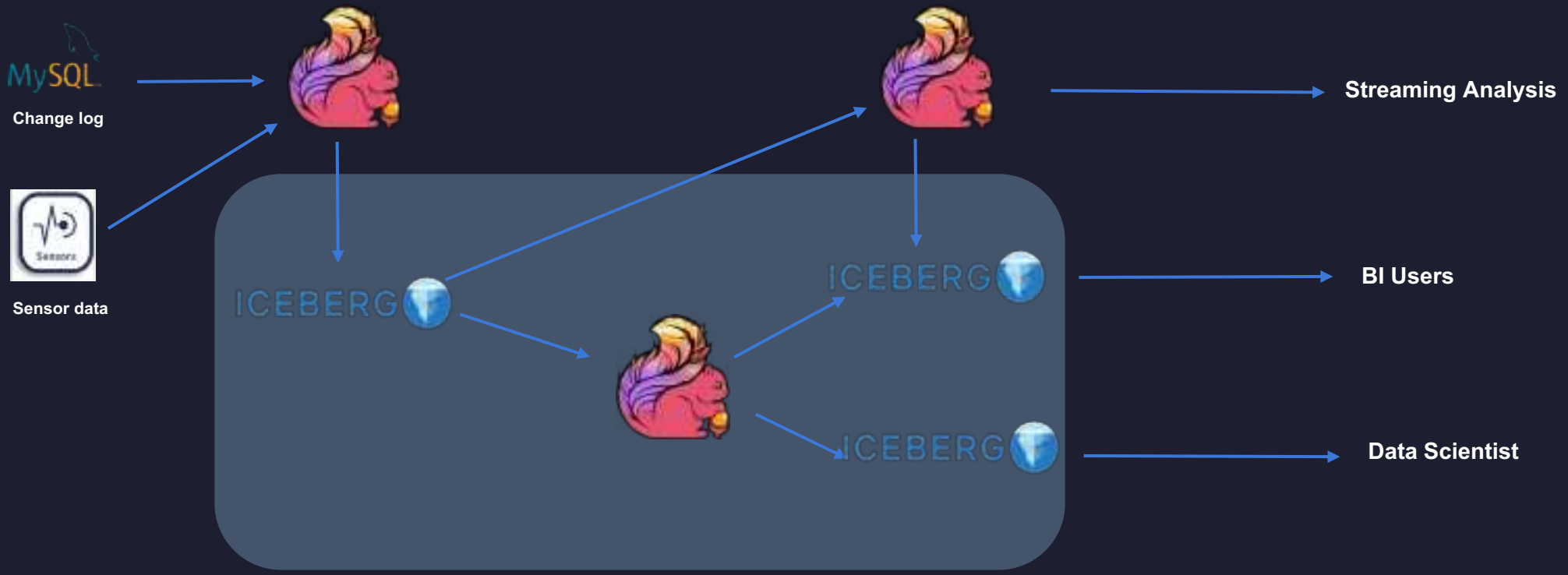


场景三: 近实时场景的流批统一 (1)



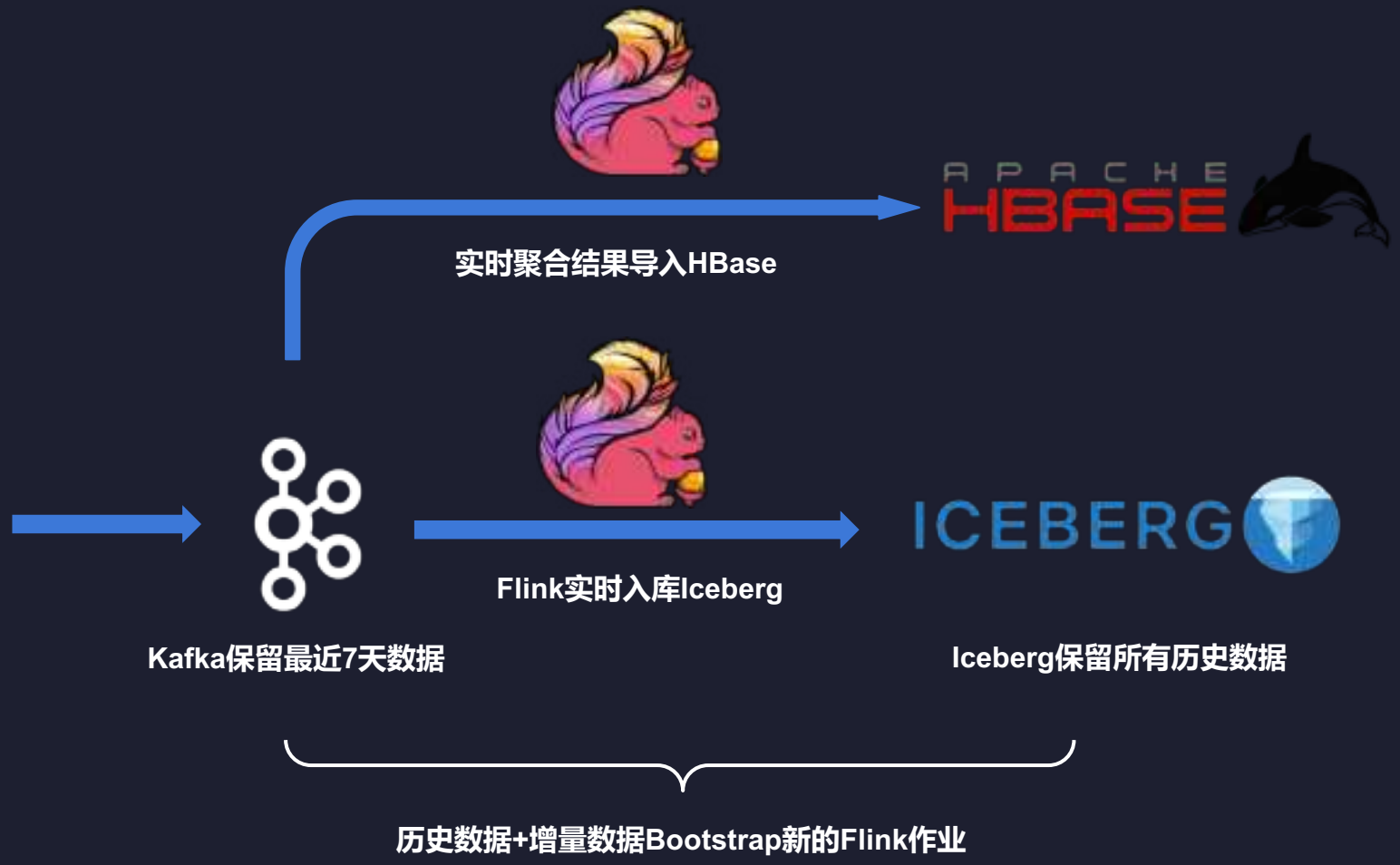
原有架构

场景三: 近实时场景的流批统一 (2)

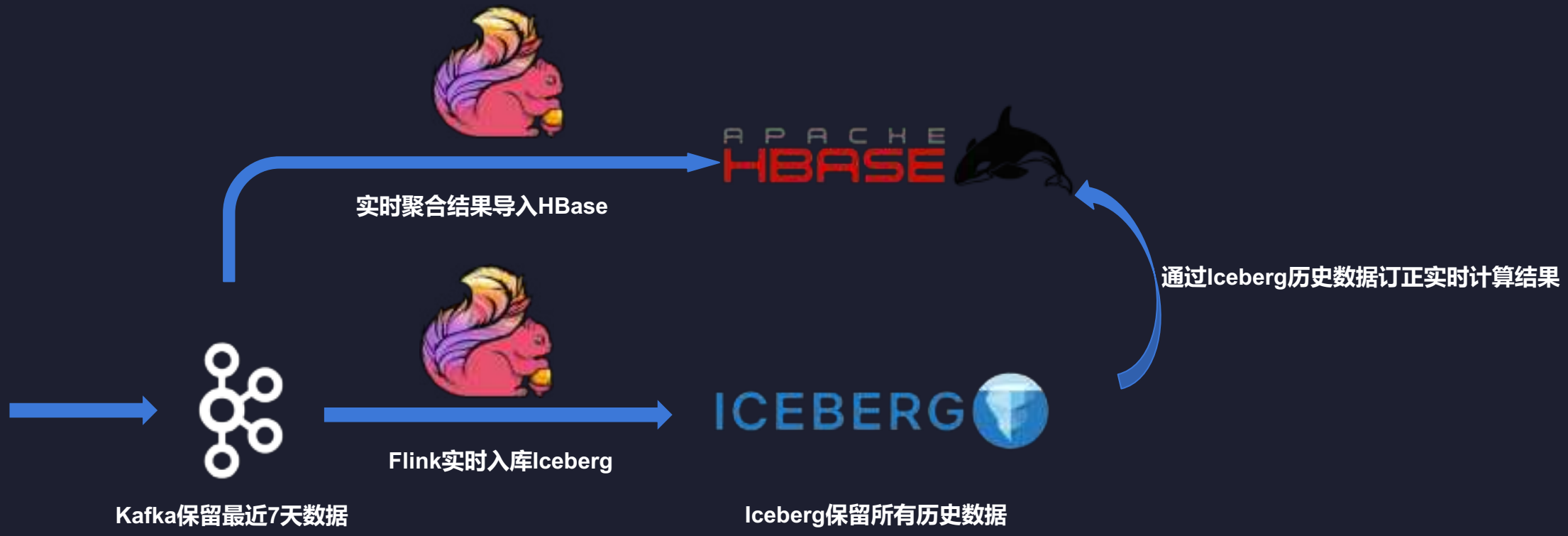


现有架构

场景四: 从Iceberg历史数据启动Flink任务



场景五: 通过Iceberg数据来订正实时聚合结果



1 什么是数据湖？

2 Flink: 为何选择Apache Iceberg？

3 Flink + Iceberg 经典场景

4 Apache Flink如何集成Apache Iceberg？

5 社区规划

如何对齐 Flink 和 Iceberg 的 Schema？（1）

Flink SQL

```
CREATE TABLE employee (
  id          INT,
  name        STRING,
  locations   MAP<STRING, ROW<posX BIGINT, posY BIGINT>>
);
```

Flink Table API

```
TableSchema employee = TableSchema.builder()
   .field("id", DataTypes.INT())
   .field("name", DataTypes.STRING())
   .field("locations",
       DataTypes.MAP(
           DataTypes.STRING(),
           DataTypes.ROW(
               DataTypes.FIELD("posX", DataTypes.BIGINT()),
               DataTypes.FIELD("posY", DataTypes.BIGINT())
           )
       )
   ).build();
```

Iceberg API

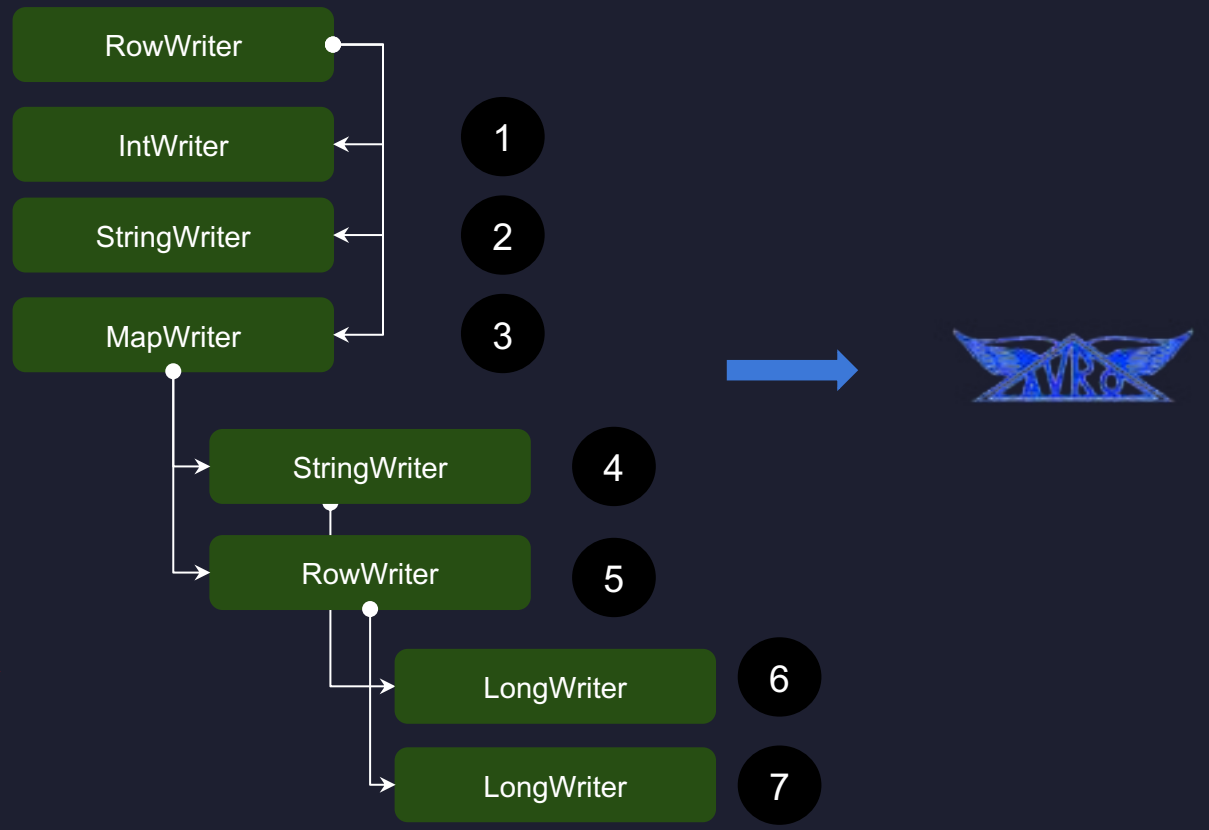
```
Schema employee = new Schema(
   NestedField.required(1, "id", IntegerType.get()),
   NestedField.optional(2, "name", StringType.get()),
   NestedField.optional(3, "locations", MapType.ofOptional(4, 5,
       StringType.get(),
       StructType.of(
           NestedField.required(6, "posX", LongType.get(), "X field"),
           NestedField.required(7, "posY", LongType.get(), "Y field"))
       )
   )
);
```

如何对齐 Flink 和 Iceberg 的 Schema？(2)

Flink Table Type	Iceberg Table DataType	说明
BooleanType	BOOLEAN	
IntType	INTEGER	Flink的TinyInt和SmallInt将转换成Iceberg的INTEGER，因为Iceberg没有更细粒度的INT
BigIntType	LONG	
FloatType	FLOAT	
DoubleType	DOUBLE	
DateType	DATE	
TimeType	TIME	Flink支持millseond，Iceberg支持microseond，Flink读Iceberg数据将截断
LocalZonedTimestampType	TIMESTAMP	Flink支持millseond，Iceberg支持microseond，Flink读Iceberg数据将截断
TimestampType	TIMESTAMP	Flink支持millseond，Iceberg支持microseond，Flink读Iceberg数据将截断
VarCharType	STRING	
BinaryType	FIXED	
VarBinarType	BINARY	
DecimalType	DECIMAL	
ListType	LIST	
MapType	MAP	

Flink 记录如何写入 Iceberg 表的 AVRO 文件？

```
TableSchema employee = TableSchema.builder()
    .field("id", DataTypes.INT())
    .field("name", DataTypes.STRING())
    .field("locations",
        DataTypes.MAP(
            DataTypes.STRING(),
            DataTypes.ROW(
                DataTypes.FIELD("posX", DataTypes.BIGINT()),
                DataTypes.FIELD("posY", DataTypes.BIGINT())
            )
        )
    ).build();
```



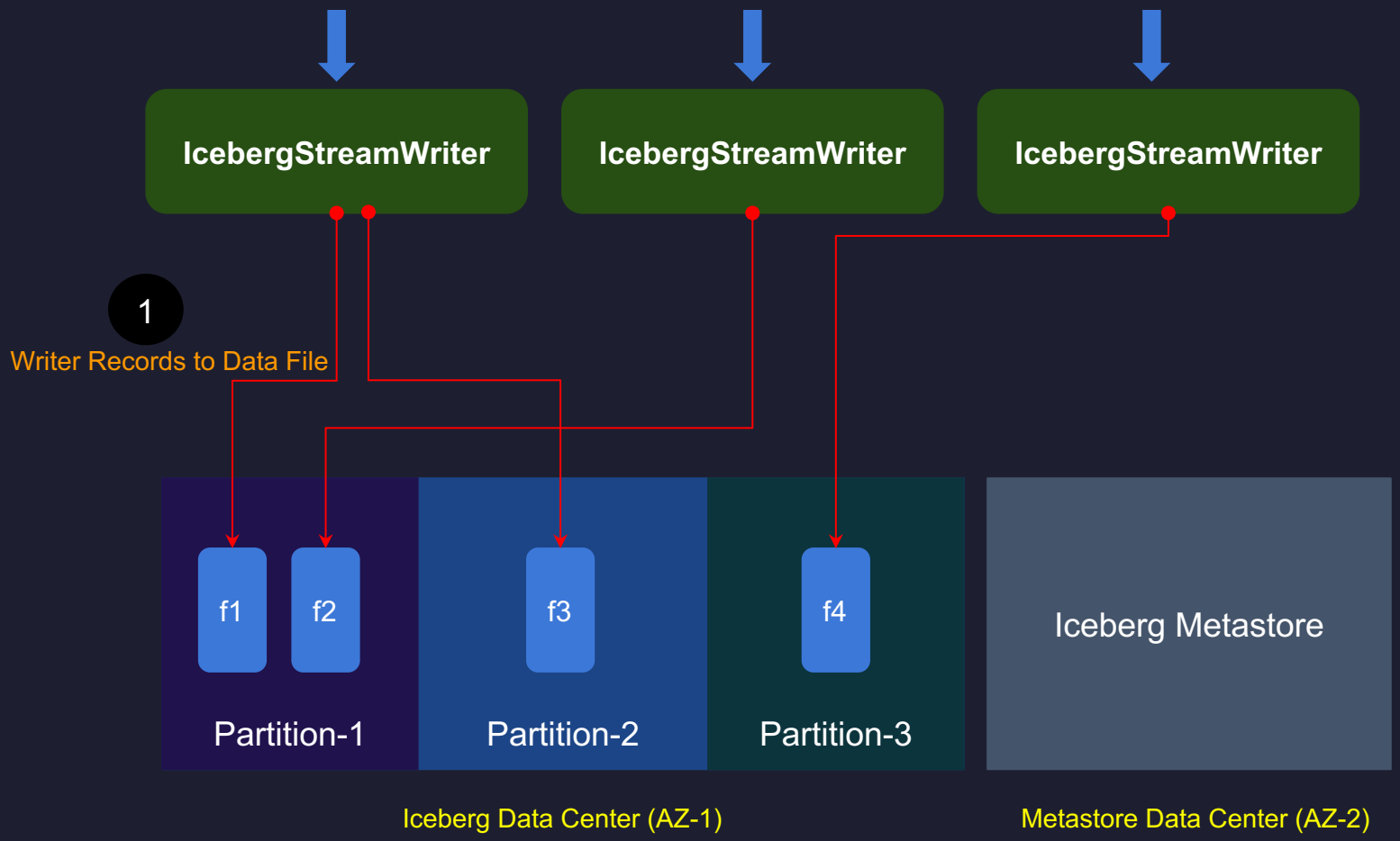
Flink Input DataStream

Iceberg FlinkAvroWriter (Flink RowData => Avro Binary)

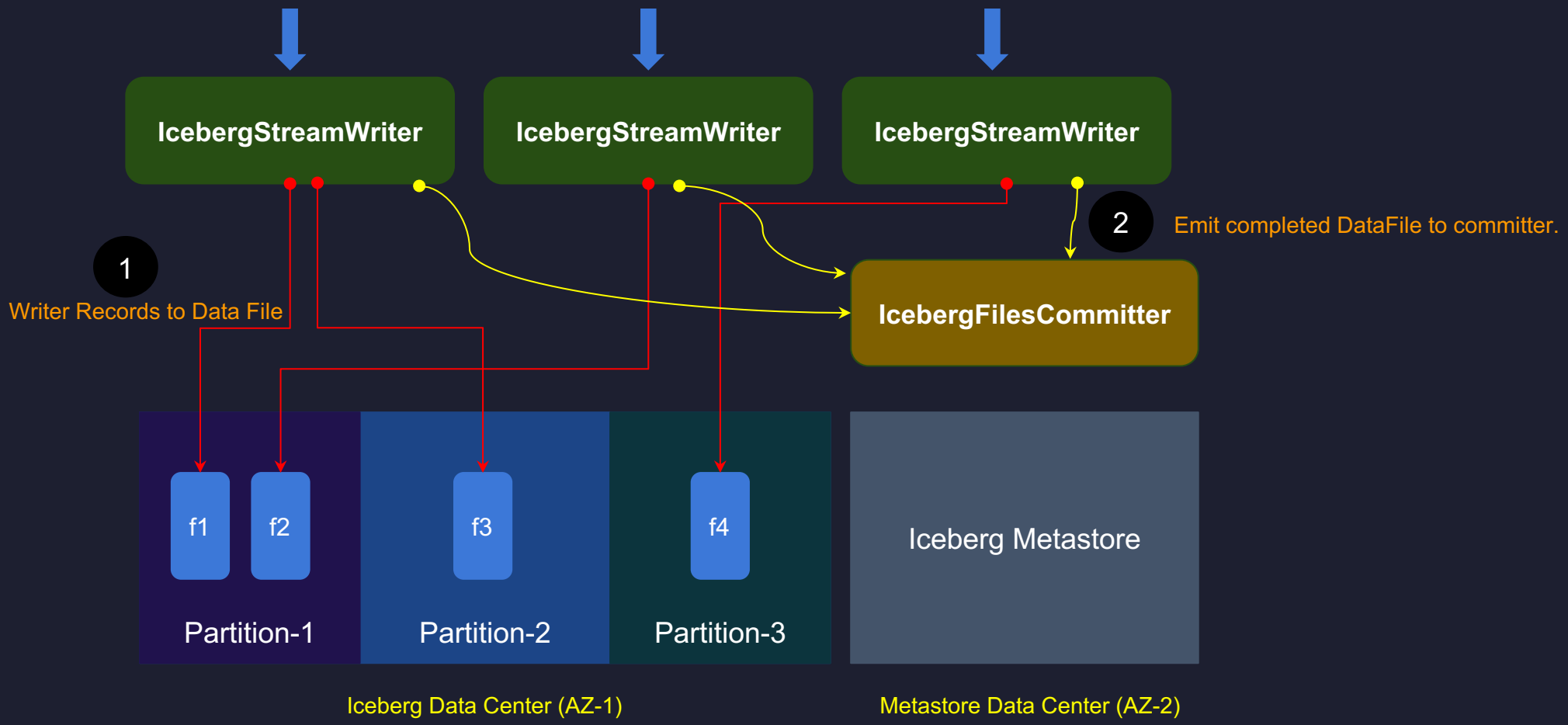
Avro File

QCon+

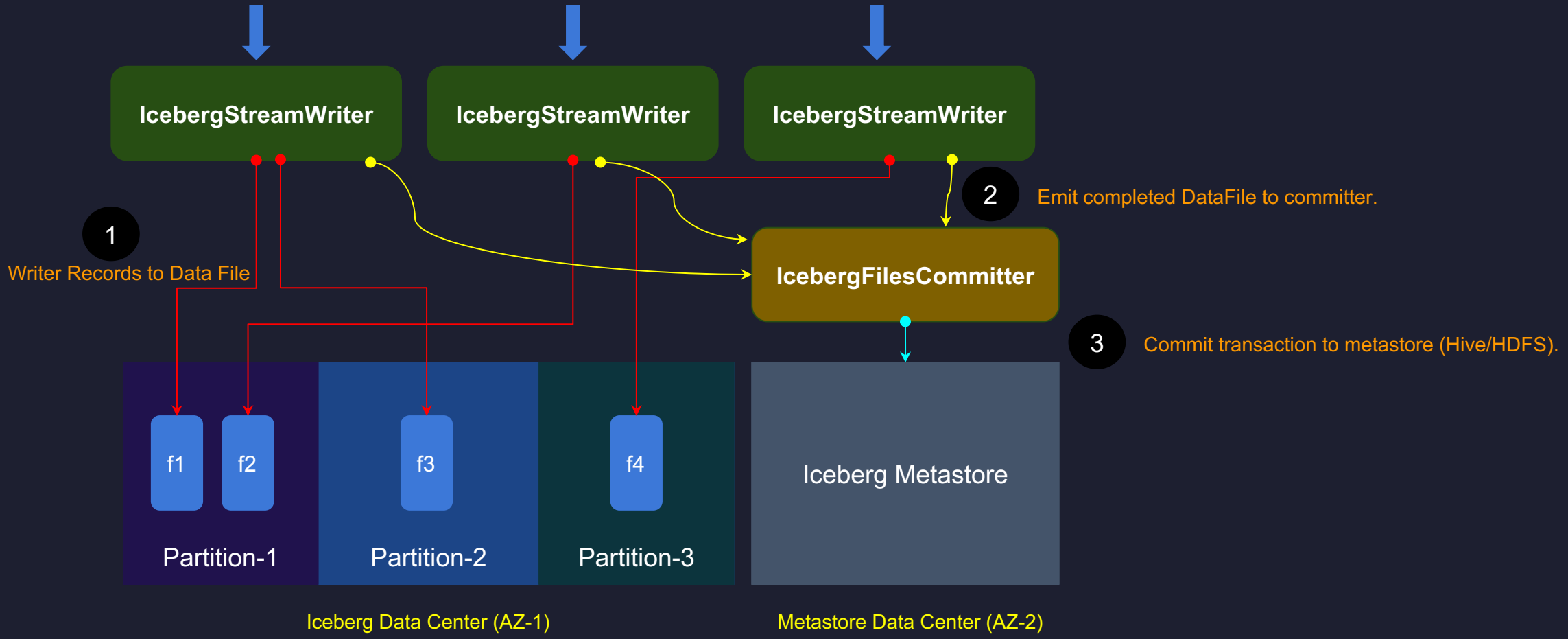
如何设计 Iceberg Sink 的 Operator ？



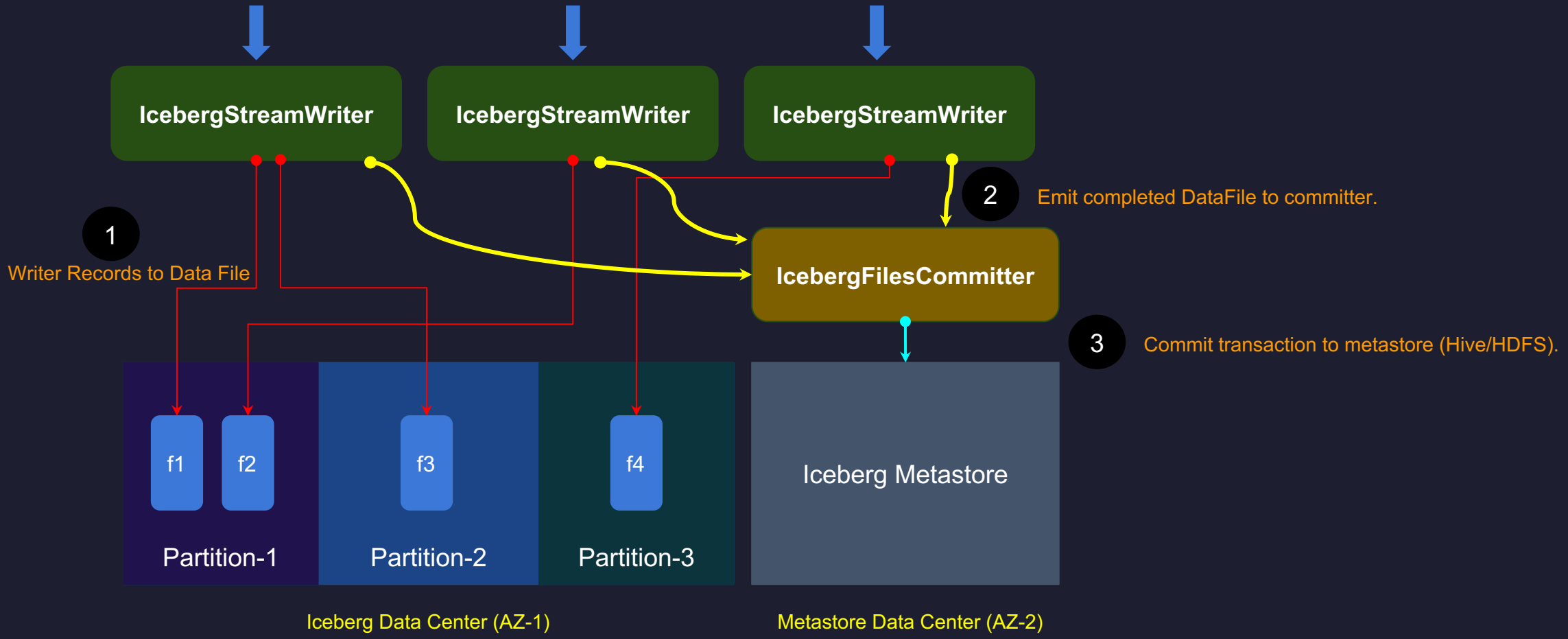
如何设计 Iceberg Sink 的 Operator ？



如何设计 Iceberg Sink 的 Operator ？



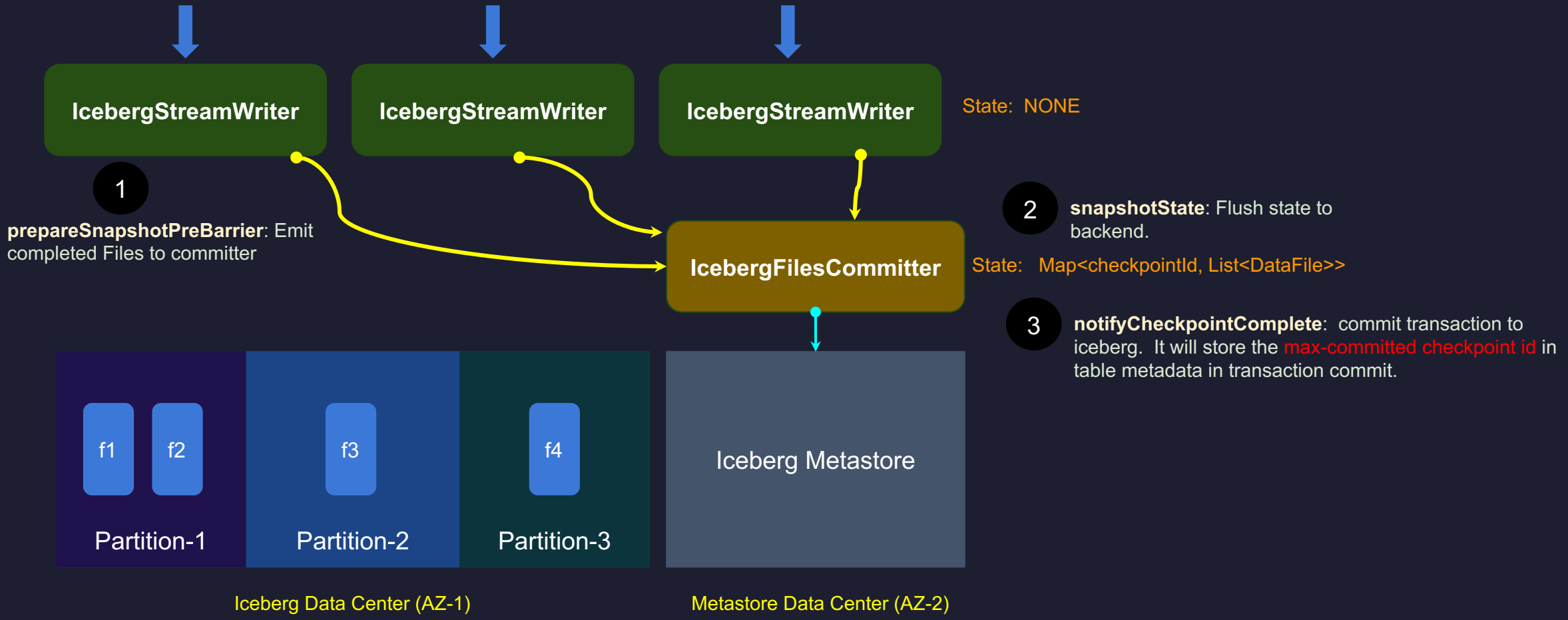
如何设计 Operator 的 State ？(1)



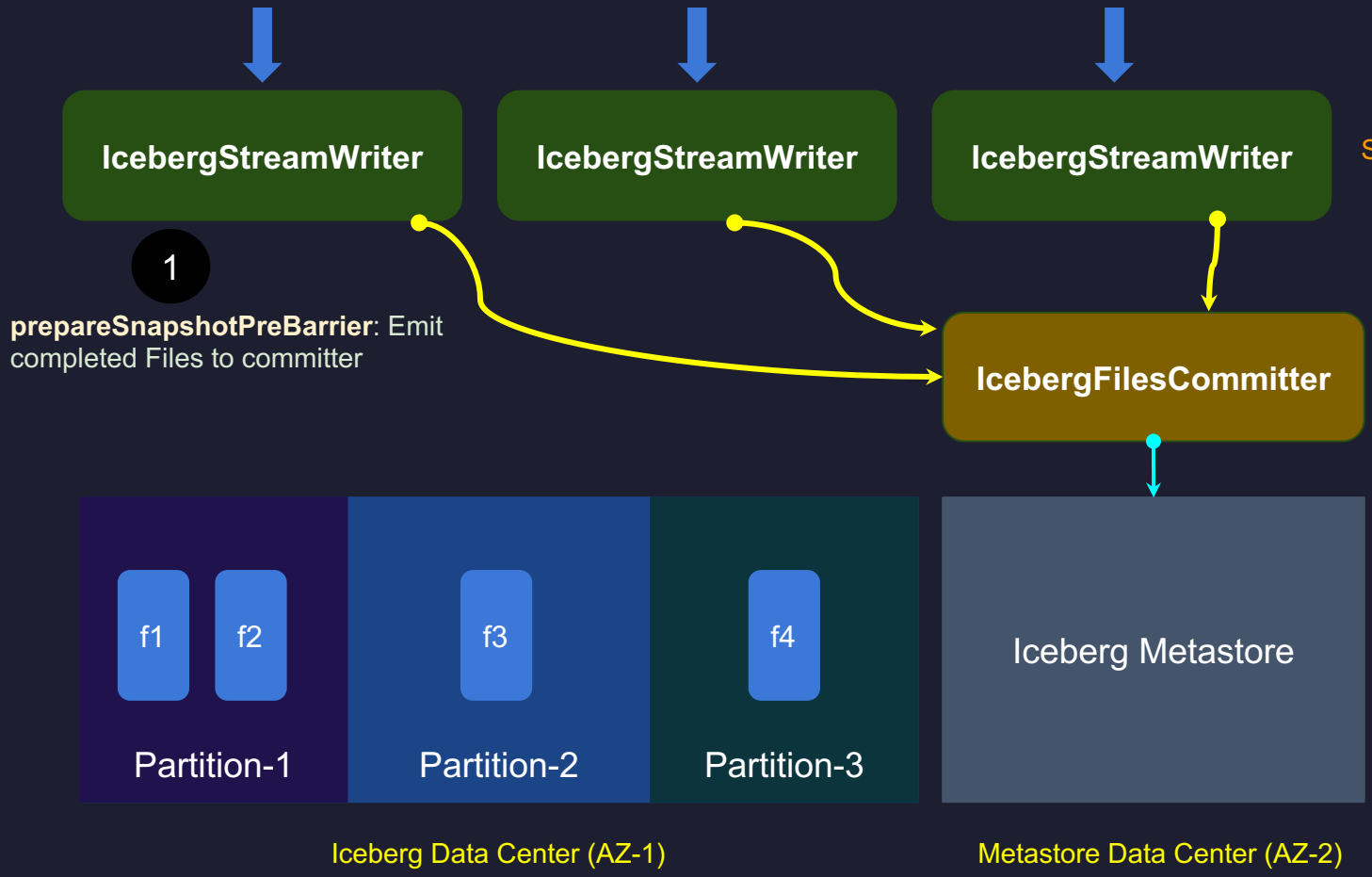
如何设计 Operator 的 State？ (2)

1. 如何在failover的时候保证 exactly-once 语义？
2. 如何保证Restore的时候不会提交重复的DataFile到Iceberg？
3. 如果中途某次checkpoint失败，其他checkpoint都正常。
   如何设计来保证数据不丢且语义正确？

如何设计 Operator 的 State ? (3)

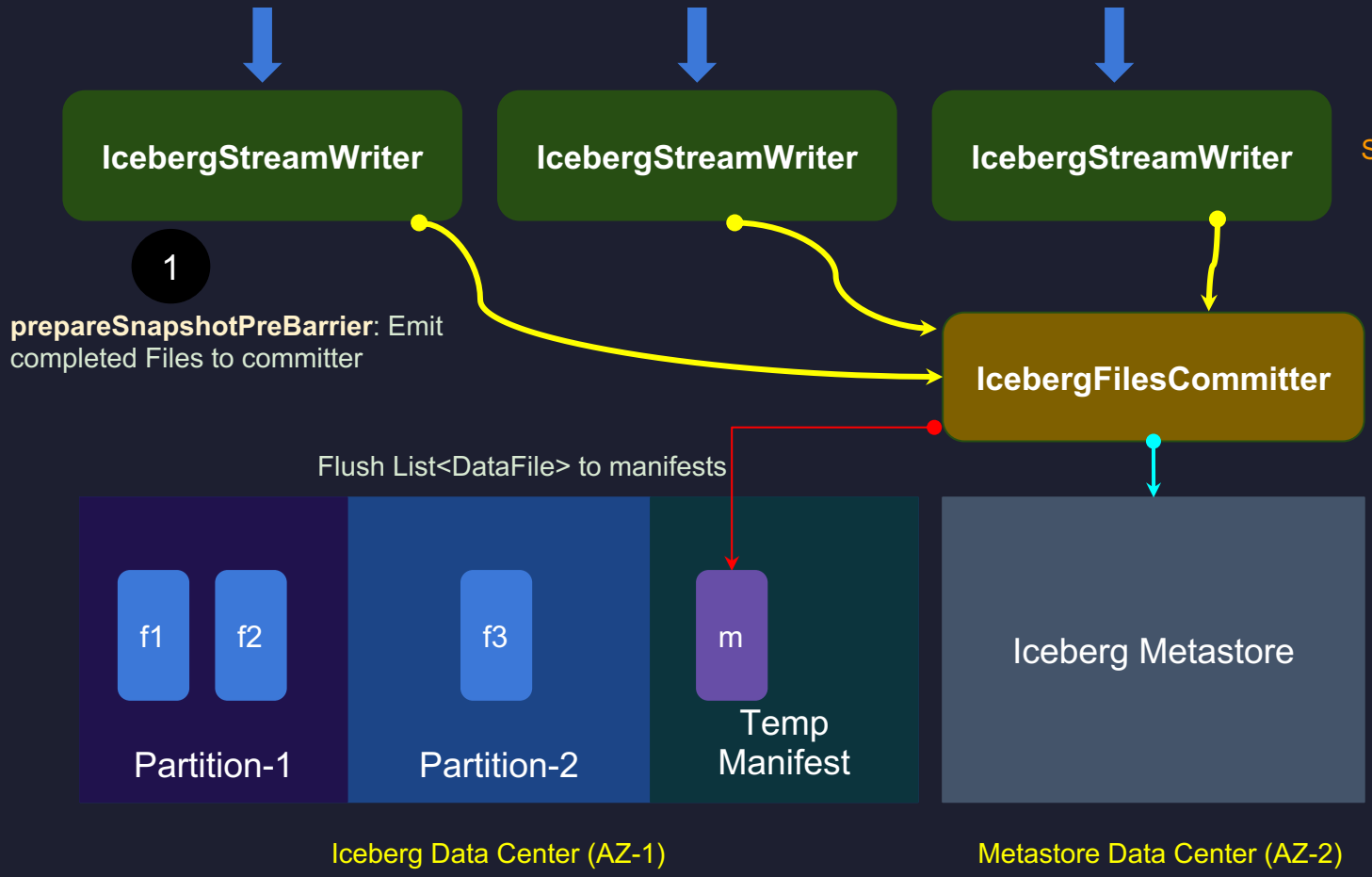


如何设计 Operator 的 State ? (4)



在云端Data和Mestore分别落在两个数据中心，由于网络延迟或者机房问题，容易导致commit iceberg transaction经常性失败。长期性的失败，将导致state膨胀过大。

如何设计 Operator 的 State ? (5)



本质上是将State backend中的List<DataFile>改成了Manifest路径，大大减少了State的数据量。

示例：Flink SQL流式导入Iceberg数据湖

创建HIVE CATALOG

```
CREATE CATALOG hive_catalog WITH (
  'type'='iceberg',
  'catalog-type'='hive',
  'uri'='thrift://localhost:9083',
  'clients'='5',
  'property-version'='1',
  'warehouse'='hdfs://nn:8020/warehouse/path'
);
USE CATALOG hive_catalog;
```

创建Database

```
CREATE DATABASE iceberg_db;
USE iceberg_db;
```

创建Sample表

```
CREATE TABLE sample (
    id BIGINT COMMENT 'unique id',
    data STRING
)PARTITIONED BY (data);
```

导入数据到Sample表

```
INSERT INTO sample
    SELECT id, data from kafka_table;
```

QCon+

1 什么是数据湖？

2 Flink: 为何选择Apache Iceberg？

3 Flink + Iceberg 经典场景

4 Apache Flink如何集成Apache Iceberg？

5 社区规划

Apache Iceberg Release 0.10.0 重点功能

1. Flink Streaming作业通过SQL或者Table API流式导入数据到Apache Iceberg数据湖
2. Flink Batch 作业批量写入数据到Iceberg
3. Flink Batch 作业读取Apache Iceberg表数据

社区后续规划

1. 优化小文件问题
2. Flink对接Row-Level Delete实现
3. 更完善的SQL支持
   - 实现增删改Column的DDL
   - Flink支持hidden partition功能
   - SQL操作compaction以及实现日常运维（如查看history、snapshot、manifest等）

THANKS

QCon+ 案例研习社