流计算 Oceanus

流湖引擎 Setats

腾讯云

文档目录

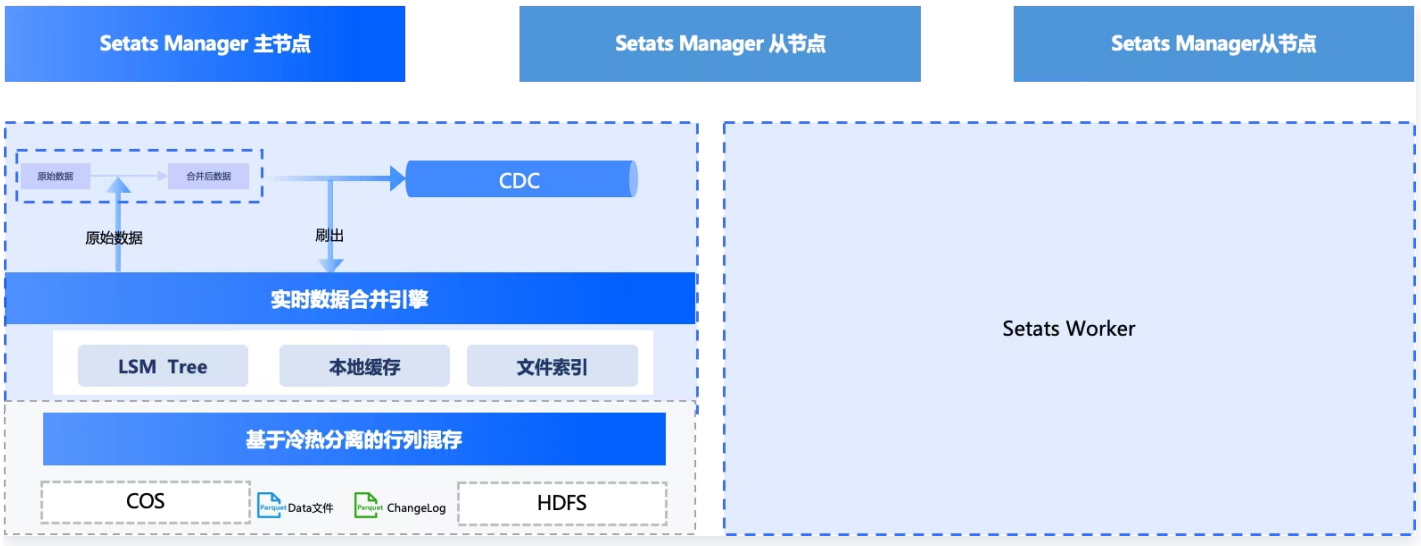
流湖引擎 Setats
Setats 简介
产品概述

最近更新时间：2026-04-16 11:28:30

Setats 是 Oceanus 团队自研的流湖一体统一存储，Setats 的目标是用一套存储同时支撑批写批读、流写流读、状态存储等多种处理模式，并能够实现秒级数据可见性。在传统大数据架构中，数据分析依赖数据湖、消息队列、本地 KV 存储等多套存储系统，这些系统相互割裂，导致数据冗余、运维复杂、成本高昂。Setats 通过统一的存储架构，支持多种不同的数据读写模式，从而消除了多套存储带来的冗余与复杂性。
Setats 目标支持如下使用场景：

能力特点	详细说明
秒级流写流读	数据写入后秒级对下游可见，支持实时变更日志（CDC）订阅，满足低延迟实时数据处理场景需求。
高效批写批读	全量数据以列存格式持久化到远程存储（COS/HDFS），支持高效的离线批量分析与扫描查询，兼容传统数据湖使用方式。
冷热分离行列混存	Setats 对表内置支持数据分层存储，远程以列存格式高效存储全量数据，本地以行存格式缓存热数据加速数据定写入合并与点查等场景。 Setats 自动管理数据在本地和远程的流转与淘汰，无需手动配置 TTL 或定时刷盘策略，从而对用户无感，并兼顾离线分析的高吞吐与在线查询的低延迟。
点查能力	支持高效的 Flink Lookup Join，并通过定制客户端支持高性能主键点查和主键前缀查询能力（后续版本发布）
状态存储	支持 Flink Delta Join 等算子状态存储（后续版本发布）
多索引加速	支持二级索引加速数据查询性能（后续版本发布）
生态兼容	兼容 Iceberg 等常用开源湖格式（后续版本发布）

产品架构



分布式架构

Setats 采用经典的 Manager / Worker 分布式架构。Setats 表数据采用 分区（Partition）+ 分桶（Bucket）两层组织结构：

- **分区**：按照用户指定的分区键对数据进行逻辑划分，支持按时间、业务维度等灵活分区，便于数据生命周期管理与查询裁剪。
- **分桶**：在分区内部，按主键哈希进一步拆分为多个桶，作为数据分布与调度的最小粒度，并可用于主键查询等场景加速。

Manager 节点负责全局的元数据管理、分桶调度与节点协调，支持多副本部署，主节点（Active）与备节点（Standby）之间通过状态同步实现自动故障切换，保障集群管理面的高可用性。Worker 节点以分桶为粒度承担实际的数据读写与存储，每个桶独立管理自己的数据文件与索引，天然支持并行处理与水平扩展，可根据业务负载按需增减 Worker 节点。

即时 Merge-on-Write 数据合并

Setats 支持不同的数据写入语义如部分列更新等，这类写入需要将新数据与历史记录进行合并才能得到完整的最新行；同时 Setats 支持生成完整的 Changelog，即更新操作需要产出包含全字段的 UpdateBefore 与 UpdateAfter，这同样依赖于写入时查询到历史数据。因此，数据合并是 Setats 写入链路中重要环节。

传统数据湖通常采用批量写入后定期合并（Merge-on-Read / 定期 Compaction）的策略，数据写入后需要等待后台合并调度完成才能对下游可见，可见性延迟通常在分钟级甚至更长。而 Setats 需要支持秒级数据可见性，因此采用了一种即时数据合并策略：数据写入后，首先通过 Lookup 操作按主键查找对应的历史数据，并立即完成新旧数据的合并——例如 Partial Update 场景下将新写入的部分列与历史记录的其余列拼接为完整记录。合并完成后，最新的完整数据写入内存缓冲区，同时产生对应的 Changelog 供下游实时订阅。缓冲区中的数据会定期异步写入远程存储（COS/HDFS）进行持久化。查询端通过综合本地缓冲区中的最新数据与远程存储中的历史数据，实现秒级数据可见。

对于相同主键数据多次写入，Setats 整体采用 Merge-on-Write 策略：在 Lookup 历史数据的过程中，同时定位旧版本记录的存储位置并生成删除向量（Deletion Vector），将旧版本标记为失效。这样在查询时，只需根据删除向量过滤已失效的记录即可，无需再执行耗时的合并操作，大幅降低了读取端的开销。

冷热分离行列混存

为了支持快速 Lookup 操作。Setats 在存储架构上采用**冷热分离的行列混合存储架构**来应对这一挑战。

- 全量数据以列存格式（如 Parquet）持久化到远程存储（COS/HDFS），满足离线分析场景下的高吞吐批量扫描需求并降低整体存储成本。
- 热数据以行存格式缓存在本地内存与磁盘中，保障即时合并时历史记录的高效查询，同时支持低延迟的主键点查等场景。当行存缓存未命中时，Setats 支持直接基于远程列存文件进行高效点查，并同时在后台将热点数据加载到缓存中，避免等待文件格式转换造成的阻塞，降低查询的整体延迟。

系统在元数据层通过 LSM Tree、Bloom Filter、Min-max 等索引与批量操作进一步加速 Lookup 的性能。

高性能数据读写引擎

为了支持高性能秒级延迟的数据读写，Setats 实现了一套基于**全异步的高性能数据读写引擎**。单条数据处理过程中的 I/O 操作（如读取本地或远程文件）不会阻塞后续数据的处理，多条数据的合并流程可以并发执行，从而最大化 CPU 利用率，避免 CPU 空转等待 I/O 返回。该设计在大规模即时合并场景下尤为关键，使得 Setats 在保持秒级可见性的同时，仍能维持高写入吞吐。

多引擎对接

Setats 提供了一个统一的数据存储层，支持流式读写、批量读写、主键检索等不同访问模式。Setats 通过一个开放格式支持多种计算引擎的读写与查询，包括 Flink、Spark、Doris、StarRocks 等。通过统一的 Catalog 和表接口，不同的计算引擎可以无缝访问 Setats 中的数据。

运维支持

Setats 服务提供运维管理能力，帮助用户高效管理集群与数据：

- **监控告警**：内置多维度的集群与表级监控指标（如写入吞吐、读取延迟、缓冲区使用率、远程存储读写量等），支持对接主流监控告警平台，实现异常状态的实时感知与及时告警。
- **Setats UI**：提供可视化管理界面，支持查看表的元信息、分区与分桶分布、数据量统计，以及各 Worker 节点的运行状态、资源使用与分桶分配情况，便于运维人员快速定位问题与评估集群健康状况。

产品优势

最近更新时间：2026-04-16 11:28:52

Setats 实现了一套全新的支持**多级存储与异构访问**的统一存储。它能够以一套统一的存储同时满足离线批量分析、完整增量日志、主键查询、状态存储等多种需求，并能够支持端到端的**秒级数据可见性**。
Setats 产品具备如下核心优势：

秒级数据可见性

Setats 具备强大的数据写入与即时可见能力，从数据变更到被下游消费系统感知，延迟通常在**秒级**，极大缩短了数据从采集到分析的链路时延。这种能力对于时效性要求极高的业务场景——如实时风控、监控告警、推荐系统等——尤为关键，使业务决策能够真正基于最新数据驱动。

支持完整 Changelog 增量机制

Setats 在数据更新过程中自动生成完整的 Changelog（变更日志），完整记录每条数据的插入、更新与删除操作，包含全字段的 UpdateBefore 与 UpdateAfter。这为下游 Flink 等流计算引擎提供了强有力的数据基础，使实时增量处理成为可能。基于完整的变更记录，开发者可以构建多层级的数据视图、进行链式增量计算，实现低延迟、高吞吐的实时数仓与复杂事件处理。

支持批处理与 OLAP 查询

Setats 构建了多层次、可增量构建的数据仓库模型，全量数据以列存格式持久化到远程存储，支持通过 Doris、Starrocks、Spark 等计算引擎进行高性能的批处理与多维分析（OLAP）查询。用户既可实现 T+0 的实时分析，也可针对历史数据执行复杂计算，真正实现批流一体化的分析架构。

支持多种 Upsert 语义

Setats 内部构建了灵活的更新逻辑框架，原生支持 Upsert、Partial Update（部分字段更新）、Aggregation（预聚合）等多种更新策略。用户可根据业务需要按字段更新或构建聚合指标，有效降低数据处理复杂度，提升存储与查询效率。

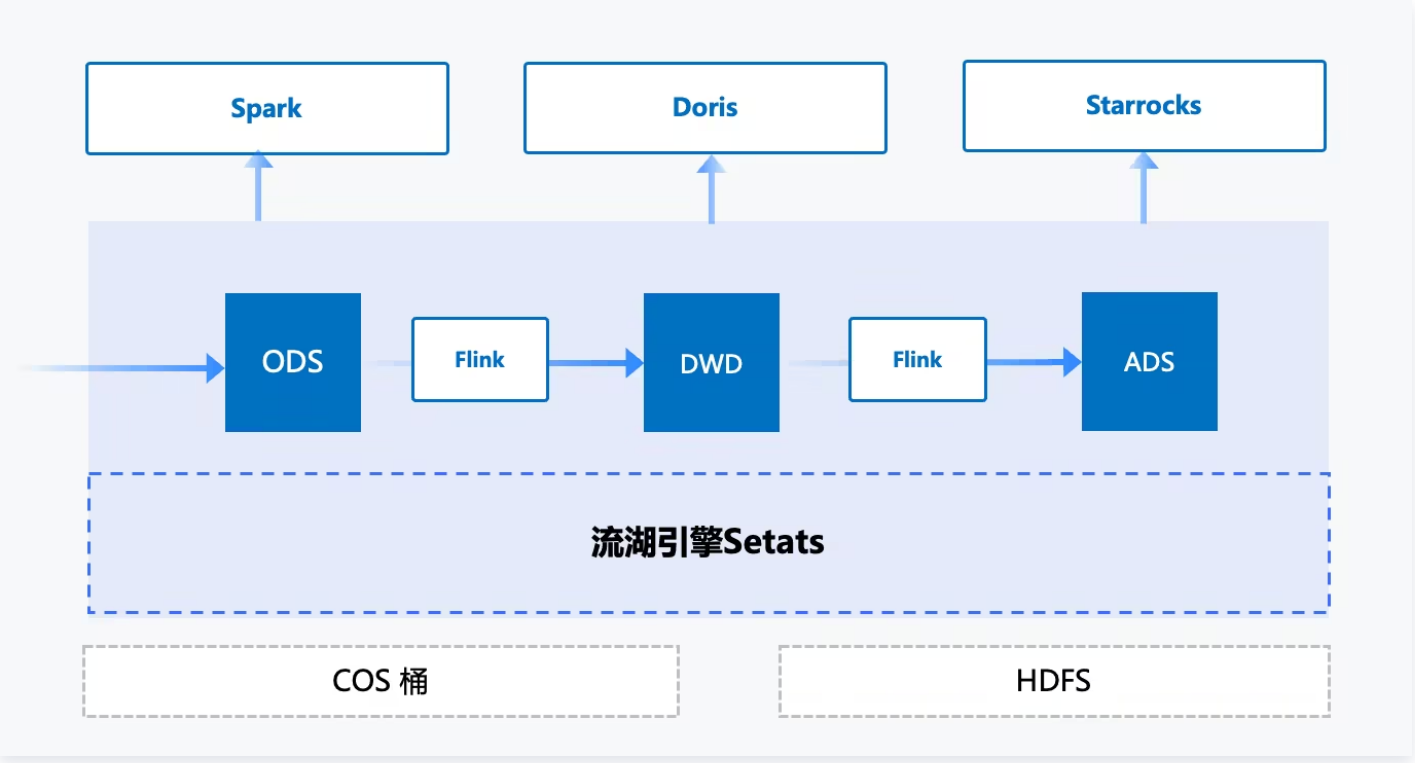
高性能数据读写

Setats 的核心是一套自研的高性能数据读写引擎。它通过优化底层存储格式——行列混存实现冷热数据分层、多级文件索引加速历史记录定位——并结合全异步的执行模型，使得数据到达时能够快速查找前值并即时完成合并。这一设计使 Setats 能够在写入时同步生成完整的增量日志与合并后数据，从而实现数据与 Changelog 的秒级可见，是支撑上述所有能力的底层技术基石。

应用场景

最近更新时间：2026-04-16 11:28:52

在传统离线数仓中，数据从 ODS 到 DWD、DWS 再到 ADS 的流转通常以 T+1 批处理方式完成，难以满足业务对实时性的要求。Setats 天然支持实时分层数仓架构：每一层的 Setats 表在数据更新时自动生成完整的 Changelog，下游层级通过订阅上游的 Changelog 进行增量计算，层层串联，形成从数据源到最终聚合结果的实时增量数据流水线。
例如，ODS 层的原始数据写入 Setats 后，Flink 作业订阅其 Changelog 进行清洗与关联，将结果写入 DWD 层的 Setats 表；DWD 层的变更再次触发下游 Flink 作业完成指标聚合，写入 DWS 层。整个过程端到端延迟在秒级，每一层的数据变更都实时向下传递，无需等待批处理调度。
同时，每一层 Setats 表都支持批量计算与 OLAP 查询。用户可以直接通过 Doris、StarRocks、Spark 等引擎对任意层级的数据进行交互式查询与多维分析，极大提升了数据探索的灵活性和问题排查的效率。
通过这一架构，用户可以用一套 Setats 存储同时承载实时增量流转与数据分析两大需求，无需在消息队列、数据湖与 OLAP 引擎之间搬迁数据，大幅简化数仓架构并降低运维成本。



腾讯云流湖引擎 Setats 方案可以广泛应用于多个行业和场景，包括但不限于出行、游戏、教育和电商等领域。

- **出行：**车辆通过车机终端持续上报位置、车速、油量/电量、胎压等大量信号数据，具有写入频率高、字段多、更新频繁的特点。Setats 可以直接承接车机信号的实时写入，以车辆 ID 为主键利用 Partial Update 实现各信号字段的独立更新，无需每次上报全量字段。Setats 支持几千列大宽表的高效写入，并支持数据秒级可见，调度系统可实时查询任意车辆的最新状态，进行动态路线规划与车辆调配；历史信号数据持久化到远程列存中，供轨迹回溯、故障分析与运营报表等离线场景使用。
- **游戏：**买量（用户获取）是核心增长手段，需要实时追踪广告投放效果与用户转化链路。Setats 支持将各渠道的买量事件数据实时写入，通过 Changelog 驱动 Flink 作业进行实时归因计算与指标聚合，运营人员可通过

Doris 或 StarRocks 秒级获取各渠道、各素材、各地域的投放效果，快速发现高 ROI 渠道并及时关停低效投放，显著提升买量效率与预算利用率。

- **教育：**Setats 可以帮助教育机构实时跟踪学生的学习进度和行为模式。课程平台将学生的学习事件（如课程观看进度、答题记录、互动行为等）实时写入 Setats，教师和教务人员可以基于秒级可见的数据及时了解每位学生的学习状况，提供个性化的教学建议，快速识别学习困难的学生并给予针对性支持。
- **电商：**Setats 可以帮助商家实现精准的用户画像分析。通过将用户的浏览、搜索、加购、下单等行为数据实时写入 Setats，利用 Partial Update 能力持续更新用户画像的各维度特征，营销系统可以实时获取最新的用户偏好，及时调整推荐算法和营销策略，快速响应市场变化，优化促销活动，从而提升转化率和客户满意度。此外，Setats 支持对大量商品与订单数据的高效写入与查询，确保商家能够实时获取销售数据，做出及时的库存和营销调整。

基本概念

最近更新时间：2026-04-16 11:28:52

Setats 集群

一个独立部署和管理的 Setats 服务实例，由 Manager 节点和 Worker 节点组成。用户的所有数据读写与管理操作均在集群内完成。

Manager

Setats 集群中的管理节点，负责全局的元数据管理、分桶调度与节点协调。Manager 支持多副本高可用部署，主节点（Active）与备节点（Standby）之间通过状态同步实现自动故障切换，确保集群管理服务不中断。

Worker

Setats 集群中的工作节点，以分桶（Bucket）为粒度承担实际的数据读写与存储。每个 Worker 管理若干个桶，负责数据的写入、合并、缓存与持久化。

数据库（DB）

集群下按照数据结构来组织、存储和管理数据的逻辑单元。用户可以在数据库中创建、更新或删除表。数据库用于对表进行分组管理，遵循层次结构：集群 > 数据库 > 表。

表（Table）

Setats 中用户数据存储的基本单位，以行和列的形式组织。表存储在特定的数据库中。Setats 支持基于主键的 INSERT、UPDATE、DELETE 操作，以及 Partial Update（部分列更新）、Aggregation（预聚合）等多种 Upsert 语义。

分区（Partition）

根据用户指定的分区键，将表的数据逻辑上划分为更小、更易管理的子集。支持按时间、业务维度等灵活分区，便于数据生命周期管理与查询裁剪。分区列中的每个唯一值（或值的组合）定义一个独立的分区。当未定义分区列时，表的全部数据属于同一个默认分区。

分桶（Bucket）

在分区内部，按主键哈希进一步将数据水平拆分为多个桶。桶是 Setats 中**数据分布、调度与迁移的最小粒度**。桶的数量可以按表进行配置。每个桶独立管理自己的数据文件与索引，由一个 Worker 节点负责处理。

缓冲区（Buffer）

每个桶在 Worker 本地维护的数据缓冲区，采用**行存格式**存储最近写入的热数据。缓冲区承担即时合并时历史记录的高效查询，同时支持低延迟的主键点查与实时流读。缓冲区中的数据会定期异步写入远程存储进行持久化。

远程存储文件（Remote Data File）

缓冲区中的数据定期刷写到远程存储（COS/HDFS），以**列存格式（Parquet）**持久化为数据文件。这些文件存储表的全量历史数据，适用于离线批量分析与扫描查询场景。

Changelog

Setats 在数据更新过程中自动生成的变更日志。每次数据变更都会产生包含全字段的 UpdateBefore（更新前快照）和 UpdateAfter（更新后快照）记录，完整记录每条数据的插入、更新和删除操作。下游流计算引擎（如 Flink）可以实时订阅 Changelog 进行增量处理。

删除向量（Delete Vector）

Setats 采用 Merge-on-Write 策略，在写入时通过 Lookup 操作定位旧版本记录的存储位置，并生成删除向量将其标记为失效。查询时只需根据删除向量过滤已失效的记录，无需再执行耗时的合并操作，大幅降低读取端开销。

Setats 集群管理
创建流湖引擎 Setats 服务

最近更新时间：2026-04-16 11:28:52

概述

Setats 集群是运行 Setats 服务的基本单元，是数据存储和处理的核心基础设施。创建集群是使用 Setats 的第一步，也是构建流式数据湖仓平台的基础。
Setats 集群由 Manager 和 Worker 两种类型的节点组成：

- Manager 节点：负责集群的数据管理和调度协调
- Worker 节点：负责实际的数据读写和处理

集群创建完成后，还需要配置 Warehouse（数据仓库）才能在实际作业中使用。Warehouse 定义了数据存储的位置（HDFS/COS）、认证方式等关键配置。

> **说明：**
> 目前流湖引擎的 Setats 服务功能通过白名单开放，如您需要使用 Setats 服务，请提交工单申请开通。

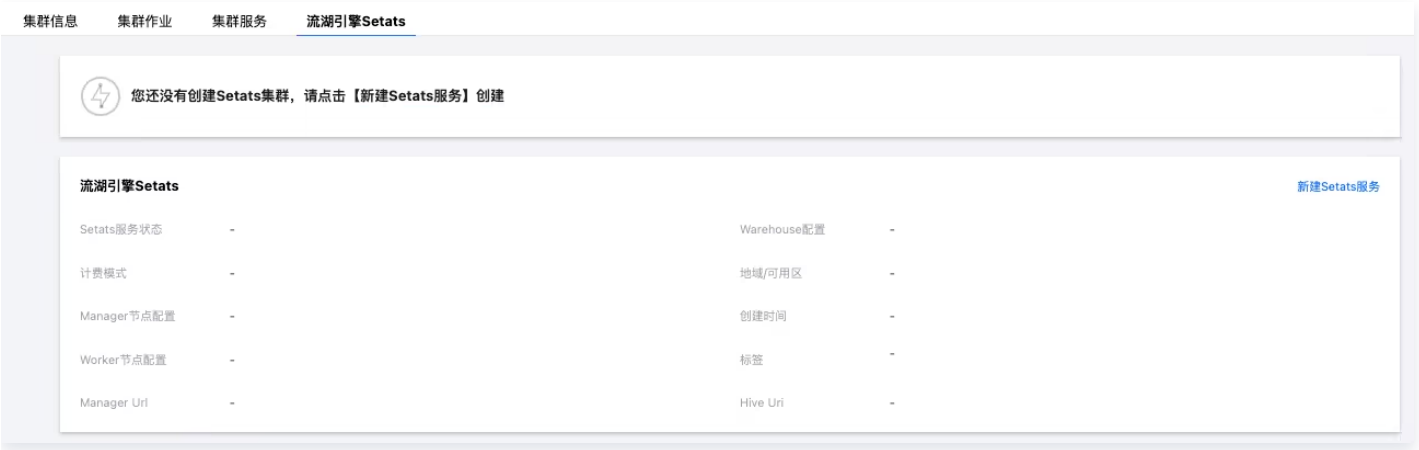
前提条件

1. 已创建腾讯云账号，创建账号可参考 注册腾讯云 。
2. 若使用子账号登录，需要获得用户账户的访问授权，详情请参见 CAM 访问管理 和 流计算服务委托授权 。

操作步骤

新建 Setats 服务

创建 Setats 服务 ，可登录 流计算 Oceanus 控制台 ，在**计算资源 > 计算集群 > 流湖引擎 Setats** 中 单击**新建Setats服务**，进行集群的购买。



流湖引擎 Setats

根据预估数据量进行资源配置。

配置资源

可用区：系统自动配置当前集群的可用区。

计费模式：包年包月。

Manager 节点配置：

配置项	可选值	推荐值	备注
节点规格	8CU / 16CU / 32CU	8CU	
资源规格	1CU-200GB 存储 / 1CU-400GB 存储 / 自定义	自定义	1CU-200GB 存储：每 CU 配置 200GB 存储 1CU-400GB 存储：每 CU 配置 400GB 存储 自定义：需要设置存储规格，为节点配置云硬盘。
存储规格	类型：SSD 云硬盘 / 通用型 SSD 云硬盘 / 增强型 SSD 云硬盘 大小：200GB ~ 32000GB	基于业务需要配置	资源规格为自定义模式时，需要配置云硬盘。
数量	1，3，5	3	

Worker 节点配置：

配置项	可选值	推荐值	备注
节点规格	8CU / 16CU / 32CU	16CU	
资源规格	1CU-200GB 存储 / 1CU-400GB 存储 / 自定义	1CU-200GB 存储	1CU-200GB 存储：每 CU 配置200GB 存储 1CU-400GB 存储：每 CU 配置400GB 存储 自定义：需要设置存储规格，为节点配置云硬盘。
存储规格	类型：SSD 云硬盘 / 通用型 SSD 云硬盘 / 增强型 SSD 云硬盘 大小：200GB ~ 32000GB	-	资源规格为自定义模式时，需要配置云硬盘。
数量	1 ~ 100	3	-

- 标签：配置集群标签。
- 购买时长：选择购买的时长。
- 自动续费：默认不勾选，为了避免业务受到影响，建议勾选自动续费，如果勾选自动续费模式，账户余额足够时，设备到期后按月自动续费。

配置完成后，请确认配置信息，点击 确认 进行付款，并完成集群购买，单击 **关闭** 将放弃本次创建集群的操作。

> **说明：**
> - 资源规格选择固定比例存储（1CU=200GB 存储 / 1CU=400GB 存储）时，存储磁盘为本地磁盘。
> - 资源规格选择自定义模式时，存储磁盘为云硬盘。

> **注意：**
> 云硬盘存储只能扩容，不支持缩容，配置时请谨慎根据实际需要进行选择和配置。

购买完成，需要配置 Warehouse。

常见问题

集群创建后能否直接使用?

不能。必须先进行 Warehouse 配置 才能在作业中使用。

Manager 和 Worker 建议配置

节点类型	开发测试	生产环境
Manager	1 个	3 个（HA）
Worker	1-2 个	根据业务量配置，建议策略至少 3 个以上

Warehouse 配置

最近更新时间：2026-04-16 11:28:52

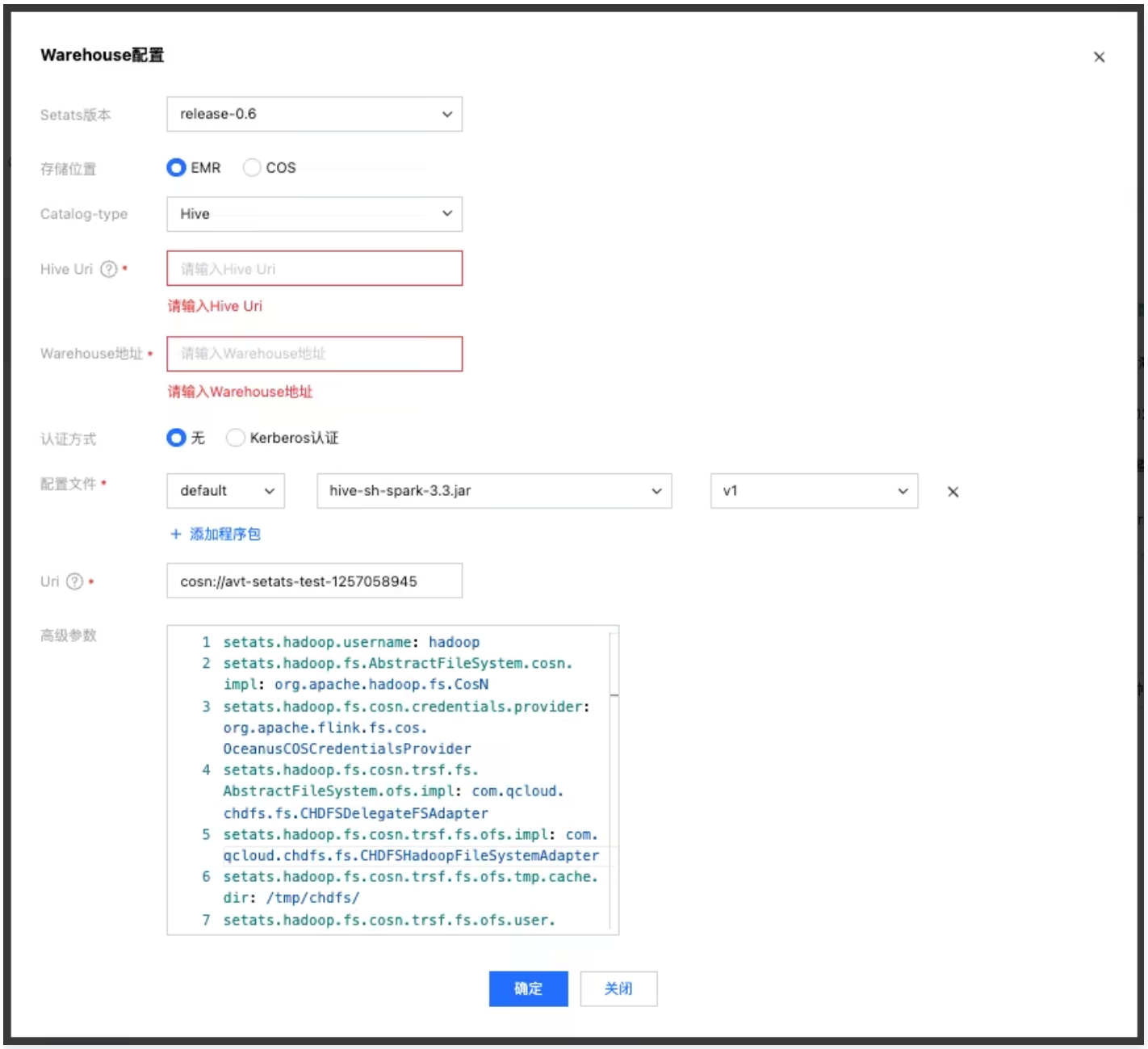
概述

Warehouse 是 Setats 集群的数据存储配置核心组件，负责管理集群中所有作业的数据存储、元数据访问和文件系统接口。在 Setats 中创建集群后，必须正确配置 Warehouse 才能在作业中进行数据读写、表管理、CheckPoint 持久化等操作。

操作步骤

配置 WareHouse

登录 流计算 Oceanus 控制台，创建流湖引擎 Setats 服务。
创建完成之后在**计算资源**中计算集群的流湖引擎 Setats 标签页，单击 Warehouse 配置的图标。



Warehouse 配置

配置项概览表

配置项	说明	是否必填	取值范围	默认值
Setats 版本	Setats 版本	是	0.6 或以上	最新稳定版
存储位置	数据存储后端类型	是	EMR/COS	-
Catalog-Type	元数据目录类型	是	Hive/Hadoop	-
Hive Uri	Hive Metastore 服务地址	条件必填*	thrift://IP:PORT	-

Warehouse 地址	数据仓库根目录	是	HDFS/COS URL	-
认证方式	安全认证机制	条件必填	Kerberos/None	None
Uri	存储系统定位地址	是	具体存储地址	-
配置文件	存储系统配置文件	条件必填	hdfs-site.xml 等	-
高级参数	额外的配置参数	否	Key-Value 格式	-

注意：
当 Catalog-Type 为 Hive 时，Hive Uri 为必填项。

详细配置说明

存储位置

指定 Setats 集群使用的数据存储后端类型。
可选值：

- EMR：使用腾讯云 EMR 集群的 HDFS 存储
- COS：使用腾讯云对象存储（COS）

选择建议：

- 如果已有 EMR 集群或有低延迟访问需求，选择 EMR
- 如果需要大规模、低成本存储，选择 COS

Catalog-Type

指定使用的元数据目录类型。
可选值：

- Hive：使用 Hive Metastore 作为元数据存储
- Hadoop：使用 Hadoop 文件系统，仅支持文件级别的元数据

Hive Uri

Hive Metastore 服务的 Thrift 访问地址。
格式：thrift://<IP1>:<PORT1>,thrift://<IP2>:<PORT2>,...

Warehouse 地址

Setats 仓库的根目录，所有作业数据和元数据将存储在此目录下。
格式：

- HDFS：hdfs://<namenode>/<path>

- COS：cosn://<bucket-name>/<path>

命名规范：

- 目录名建议使用小写字母和下划线
- 建议包含环境标识（dev/test/prod）
- 建议包含项目标识

认证方式

指定存储系统的安全认证机制。
可选值：

- Kerberos：使用 Kerberos 协议进行安全认证（适用于 EMR Kerberos 场景）
- None：不使用额外认证（适用于 EMR 非 Kerberos 或 COS 场景）

选择建议：

- EMR 集群启用了 Kerberos，选择 Kerberos
- EMR 集群未启用 Kerberos，选择 None
- COS/CHDFS 存储，选择 None

Uri

存储系统的定位地址，用于连接和访问存储系统。
格式说明：

存储类型	Uri 格式	示例
EMR HDFS	HDFS 服务名称或 IP	HDFS11000127 或 10.0.0.1:9000
COS	COS 桶标识	cosn://test-123

配置文件

存储系统所需的配置文件。

高级参数

额外的配置参数，用于更细粒度的控制存储行为。

配置完成

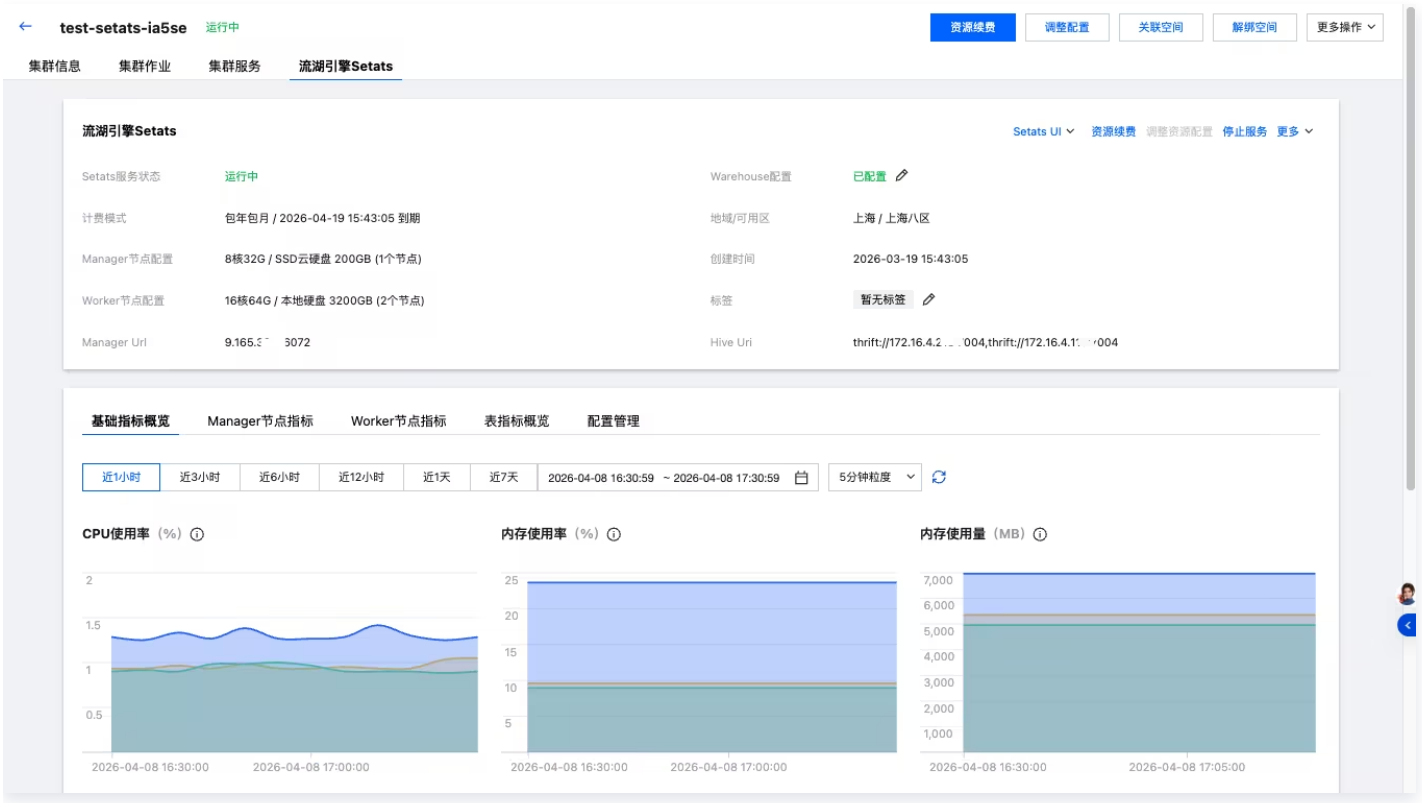
提供 Manager Url 给作业使用。


集群信息
集群作业
集群服务
流湖引擎Setats
流湖引擎Setats
云监控
Setats UI
资源续费
调整资源配置
停止服务
更多
Setats服务状态
运行中
Warehouse配置
已配置
计费模式
包年包月 / 自动续费 / 2026-04-20 15:04:25 到期
地域/可用区
上海 / 上海八区
Manager节点配置
8核32G / SSD云硬盘 200GB (1个节点)
创建时间
2026-03-20 15:04:25
Worker节点配置
16核64G / 本地硬盘 3200GB (3个节点)
标签
暂无标签
Manager Url
Hive Uri
thrift://172.16.4.1
2.16.4.86:7004

查看集群信息

最近更新时间：2026-04-16 11:28:52

查看**流湖引擎 Setats** 信息，可登录 流计算 Oceanus 控制台，在**计算资源**中，选择计算集群，在集群的**流湖引擎Setats**标签页中 **新建Setats服务**，具体可参考 创建流湖引擎 Setats 服务 。创建 Setats 集群后可在**计算资源**中计算集群的流湖引擎 Setats 标签页中查看集群信息。



配置管理

最近更新时间：2026-04-16 15:59:53

概述

Setats 提供了灵活的系统配置管理能力，用户可以根据业务需求调整集群的运行参数。通过合理的配置优化，可以显著提升集群的性能、稳定性和可靠性。
Setats 服务调整配置以后，需要重启才能生效，重启会影响作业。

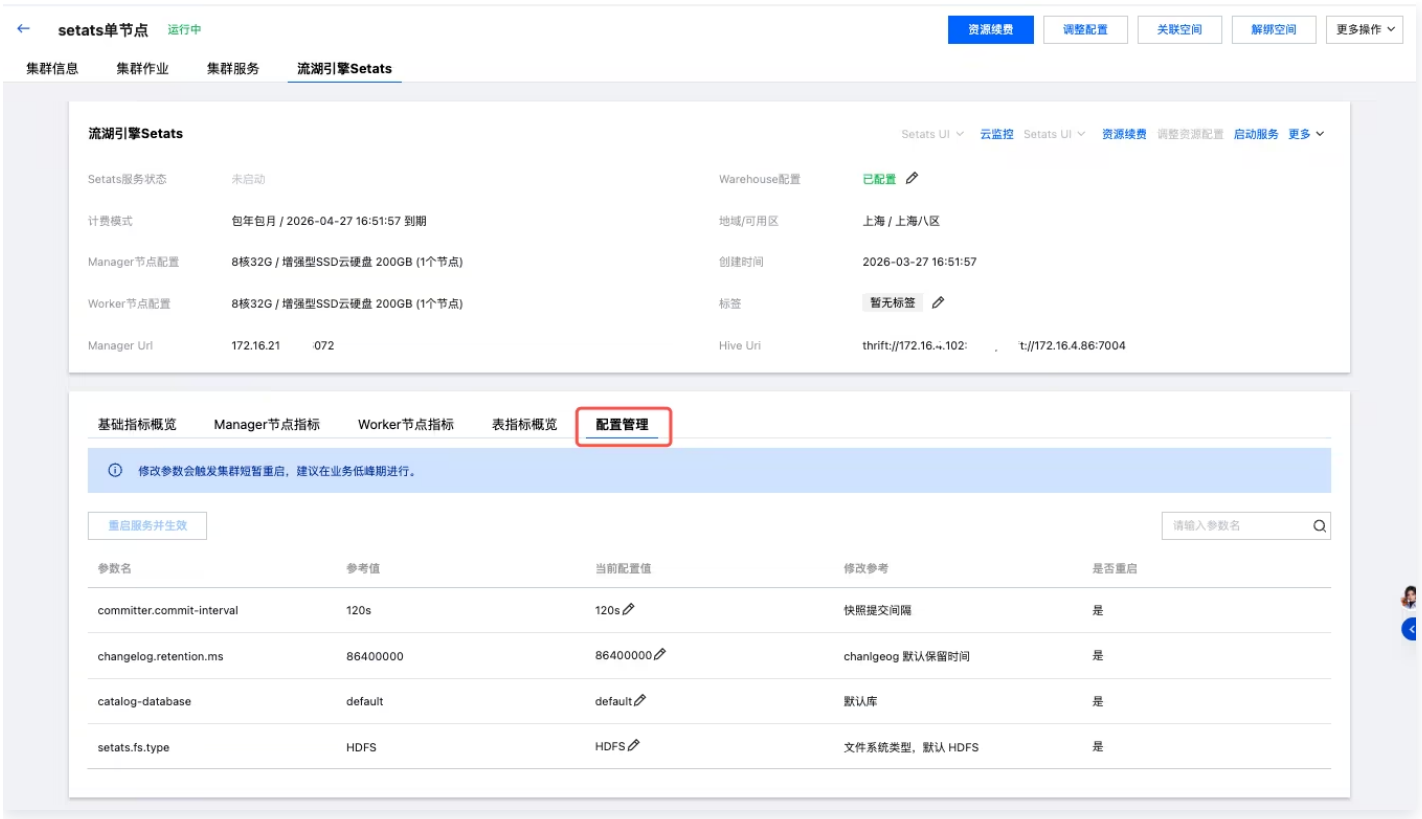
操作步骤

登录 流计算 Oceanus 控制台，在**计算资源**中，选择计算集群，在集群的**流湖引擎 Setats** 标签页中 **新建 Setats 服务**，具体可参考 创建流湖引擎 Setats 服务 。
创建 Setats 集群后可在**计算资源 >** 计算集群的流湖引擎 Setats 标签页中选择**配置管理**。

- 选择指标，并修改配置值。
- **重启服务并生效，确定** 执行操作。

配置信息参考

参数名	参考值	当前配置值	修改参考	是否重启
committer.commit-interval	120s	120s	快照提交间隔	是
changelog.retention.ms	86400000	86400000	changelog 默认保留时间	是
catalog-database	default	default	默认库	是
setats.fs.type	HDFS	HDFS	文件系统类型，默认 HDFS	是


setats单节点
运行中
资源续费
调整配置
关联空间
解绑空间
更多操作
集群信息
集群作业
集群服务
流湖引擎Setats
流湖引擎Setats
Setats UI
云监控
Setats UI
资源续费
调整资源配置
启动服务
更多
Setats服务状态
未启动
Warehouse配置
已配置
计费模式
包年包月 / 2026-04-27 16:51:57 到期
地域/可用区
上海 / 上海八区
Manager节点配置
8核32G / 增强型SSD云硬盘 200GB (1个节点)
创建时间
2026-03-27 16:51:57
Worker节点配置
8核32G / 增强型SSD云硬盘 200GB (1个节点)
标签
暂无标签
Manager Url
172.16.21 072
Hive Uri
thrift://172.16.4.102: t://172.16.4.86:7004
基础指标概览
Manager节点指标
Worker节点指标
表指标概览
配置管理
修改参数会触发集群短暂重启，建议在业务低峰期进行。
重启服务并生效
请输入参数名
参数名
参考值
当前配置值
修改参考
是否重启
committer.commit-interval
120s
120s
快照提交间隔
是
changelog.retention.ms
86400000
86400000
chanlgeog 默认保留时间
是
catalog-database
default
default
默认库
是
setats.fs.type
HDFS
HDFS
文件系统类型，默认 HDFS
是

作业开发指南
Flink 作业开发
Flink 流式写入

最近更新时间：2026-04-16 11:28:52

概述

Setats 支持通过 Flink 持续写入实时数据，使用 setats connector 将 Upsert 数据写入 Setats 主键表。数据写入后会触发 Eager 合并并生成 Changelog，因此能够同时满足秒级可见、后续 CDC 消费和 OLAP 查询等场景。

> ⚠ **注意：**
> 1. 以下作业使用 Hive catalog 和 COS 加速桶演示。
> 2. 如果使用 Hadoop catalog ，需要修改 ‘catalog-type’='hadoop' 。
> 3. Oceanus 可以不提前建表，Flink Setats Sink 会自动建表。

前提条件

- 已创建 Setats 集群并完成 Warehouse 配置。
- 已获取 manager-url。
- 已准备 Flink 运行环境，例如流计算 Oceanus。
- 已确认 Warehouse 对应的底层存储、Hive Metastore、网络访问和权限配置均已打通。
- 如果使用 COS 加速桶 作为 Warehouse，建议额外确认：
  - 加速桶已提前创建目标目录，例如 cosn://<bucket>/warehouse。
  - 已为加速桶配置 Oceanus 所在子网的访问权限。
  - Flink 集群具备访问 Hive Metastore、COS / CHDFS 的网络能力。

Oceanus 高级参数示例

如果作业运行在 Oceanus，且 Warehouse 使用 COS 加速桶，可参考以下高级参数模板：

```
pipeline.max-parallelism: 2048
containerized.taskmanager.env.HADOOP_USER_NAME: hadoop
containerized.master.env.HADOOP_USER_NAME: hadoop
taskmanager.memory.managed.fraction: 0.1
```

```
execution.checkpointing.externalized-checkpoint-retention:
RETAIN_ON_SUCCESS
execution.checkpointing.tolerable-failed-checkpoints: 5
flink.hadoop.fs.cosn.trsf.fs.AbstractFileSystem.ofs.impl:
com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter
flink.hadoop.fs.cosn.trsf.fs.ofs.impl:
com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter
flink.hadoop.fs.cosn.trsf.fs.ofs.tmp.cache.dir: /tmp/chdfs/
flink.hadoop.fs.cosn.trsf.fs.ofs.user.appid: <YourAppId>
flink.hadoop.fs.cosn.trsf.fs.ofs.bucket.region: <YourRegion>
flink.hadoop.fs.cosn.trsf.fs.ofs.upload.flush.flag: true
flink.hadoop.fs.AbstractFileSystem.cosn.impl: org.apache.hadoop.fs.CosN
flink.hadoop.fs.cosn.impl: org.apache.hadoop.fs.CosFileSystem
flink.hadoop.fs.cosn.bucket.region: <YourRegion>
flink.hadoop.fs.cosn.userinfo.appid: <YourAppId>
flink.hadoop.fs.ofs.impl:
com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter
flink.hadoop.fs.ofs.tmp.cache.dir: /tmp/chdfs/
flink.hadoop.fs.ofs.upload.flush.flag: true
flink.hadoop.fs.ofs.user.appid: <YourAppId>
flink.hadoop.fs.ofs.bucket.region: <YourRegion>
```

说明：
请根据实际环境替换 `AppId`、`Region`、`HADOOP_USER_NAME` 等参数。不同集群环境中可用的参数项可能略有差异，请以实际运行环境校验结果为准。

建表语法

创建 Setats 目标表

以下示例以 Hive Catalog 为例：

```
CREATE TABLE `setats_sink` (
  `id` STRING,
  `dt` STRING,
  `name` STRING,
  `address` STRING,
  PRIMARY KEY (`id`, `dt`) NOT ENFORCED
) PARTITIONED BY (`dt`) WITH (
```

```
  'connector' = 'setats',
  'manager-url' = '<Manager Url>',
  'catalog-type' = 'hive',
  'catalog-name' = 'setats',
  'catalog-database' = 'testdb',
  'catalog-table' = 'setats_sink',
  'warehouse' = 'cosn://<bucket>/warehouse',
  'location' = 'cosn://<bucket>/warehouse/testdb/setats_sink',
  'uri' = 'thrift://<metastore-host-1>:7004,thrift://<metastore-host-
2>:7004',
  'bucket' = '10',
  'bucket-key' = 'id',
  'merge-engine' = 'PARTIAL_UPDATE',
  'setats.sink.flush-interval' = '5s'
);
```

关键参数说明

参数	必填	说明	默认值
connector	是	固定为 setats	（无，必须填写）
manager-url	是	Setats Manager 地址	（无，必须填写）
warehouse	是	Setats Warehouse 路径	（无，必须填写）
catalog-type	是	Catalog 类型，常用 Hive 或 Hadoop	（无，必须填写）
catalog-name	是	Catalog 名称	（无，必须填写）
catalog-database	是	数据库名称	default
catalog-table	建议填写	目标表名，建议与实际表名保持一致	（无）
location	建议填写	表的物理存储路径，使用 COS / HDFS 时建议显式指定	（无）
uri	Hive Catalog 必填	Hive Metastore 地址	（无，必须填写）

bucket	否	目标表的桶数量，建议结合数据规模和并发写入能力配置	-1
bucket-key	否	分桶键，用于决定数据写入时落到哪个 Bucket。建议选择分布较均匀、相对稳定的字段，通常可使用主键字段或主键子集	(无)
merge-engine	否	合并策略： DEDUPLICATE：合并取最后一条 PARTIAL_UPDATE：部分列更新，取每个字段最后一个不为空的值	DEDUPLICATE
setats.sink.flush-interval	否	Sink 刷新间隔，例如 5s	5s

写入示例

```
-- 创建测试数据源

CREATE TABLE `test_source_datagen` (
  `id` STRING,
  `name` STRING,
  `address` STRING
) WITH (
  'connector' = 'datagen',
  'rows-per-second' = '1',
  'fields.id.start' = '0',
  'fields.id.end' = '100000',
  'fields.id.kind' = 'sequence',
  'fields.name.length' = '5',
  'fields.address.length' = '10'
);
-- 创建 Setats Sink 表

CREATE TABLE `demo_setats_table1` (
  `id` STRING,
  `dt` STRING,
  `name` STRING,
  `address` STRING,
  PRIMARY KEY (`id`, `dt`) NOT ENFORCED
```

```
) PARTITIONED BY (`dt`) WITH (
  'connector' = 'setats',
  'manager-url' = '<Manager Url>',
  'catalog-type' = 'hive',
  'catalog-name' = 'setats',
  'catalog-database' = 'testdb',
  'catalog-table' = 'demo_setats_table1',
  'warehouse' = 'cosn://<bucket>/warehouse',
  'location' = 'cosn://<bucket>/warehouse/testdb/demo_setats_table1',
  'uri' = 'thrift://<metastore-host-1>:7004,thrift://<metastore-host-
2>:7004',
  'bucket' = '10',
  'merge-engine' = 'PARTIAL_UPDATE',
  'setats.sink.flush-interval' = '5s'
);
-- 执行写入

INSERT INTO `demo_setats_table1`
SELECT
  `id`,
  DATE_FORMAT(NOW(), 'yyyyMMdd') AS `dt`,
  `name`,
  `address`
FROM `test_source_datagen`;
```

部分列更新

Setats 主键表支持部分列更新，Flink 写入任务中可以只声明需要更新的列，未出现在写入记录中的其他列会保留原值。对于订单、画像、用户标签等“按主键增量补列”的业务场景，建议结合 merge-engine = 'PARTIAL_UPDATE' 使用。
示例：

```
CREATE TABLE `setats_partial_sink` (
  `id` STRING,
  `dt` STRING,
  `name` STRING,
  PRIMARY KEY (`id`, `dt`) NOT ENFORCED
) PARTITIONED BY (`dt`) WITH (
  'connector' = 'setats',
  'manager-url' = '<Manager Url>',
```

```
  'catalog-type' = 'hive',
  'catalog-name' = 'setats',
  'catalog-database' = 'testdb',
  'catalog-table' = 'demo_setats_table1',
  'warehouse' = 'cosn://<bucket>/warehouse',
  'uri' = 'thrift://<metastore-host-1>:7004,thrift://<metastore-host-
2>:7004',
  'merge-engine' = 'PARTIAL_UPDATE'
);
```

发布与运行

在 Oceanus 等托管环境中发布 SQL 作业时，建议按以下顺序检查：

1. 依赖 JAR 是否已上传并关联到当前空间。
2. Hive 配置 JAR 是否已随作业一并引用。
3. Warehouse、location、uri、manager-url 是否已替换为真实值。
4. 加速桶场景下，相关高级参数是否已正确配置。

注意事项

- dt 等分区字段建议结合业务查询模式设计，避免过细分区带来额外管理成本。
- bucket 数量建议结合写入并发、数据规模和 Worker 数量综合评估。
- manager-url、warehouse、location 等参数必须与集群配置保持一致。

Flink CDC流式读取

最近更新时间：2026-04-16 11:28:52

概述

Setats 支持通过 Flink 以 CDC 方式流式读取表的 Changelog 数据。数据写入 Setats 后，系统会基于主键合并结果生成变更日志，下游可通过 setats-cdc connector 持续消费新增、更新和删除事件。
该能力适用于以下场景：

- 实时订阅 Setats 中的主键变更数据
- 将 Setats 作为统一增量数据源，分发给下游 Flink 作业
- 基于 Changelog 构建维表缓存、实时聚合或消息分发链路

前提条件

- 已创建 Setats 集群并完成 Warehouse 配置。
- 已获取 manager-url。
- 已有上游任务持续向目标 Setats 表写入数据。

建表语法

```
CREATE TABLE `setats_cdc_source` (
  `id` STRING,
  `dt` STRING,
  `name` STRING,
  `address` STRING,
  PRIMARY KEY (`id`, `dt`) NOT ENFORCED
) WITH (
  'connector' = 'setats-cdc',
  'warehouse' = 'cosn://<bucket>/warehouse',
  'catalog-type' = 'hive',
  'catalog-name' = 'setats',
  'catalog-database' = 'testdb',
  'catalog-table' = 'demo_setats_table1',
  'uri' = 'thrift://<metastore-host-1>:7004,thrift://<metastore-host-
2>:7004',
  'manager-url' = '<Manager Url>',
  'bucket.discovery-interval' = '30s',
  'startup.mode' = 'earliest'
);
```

关键参数说明

参数	必填	说明	默认值
connector	是	固定为 setats	(无，必须填写)
manager-url	是	Setats Manager 地址	(无，必须填写)
warehouse	是	Setats Warehouse 路径	(无，必须填写)
catalog-type	是	Catalog 类型，常用 Hive 或 Hadoop	(无，必须填写)
catalog-name	是	Catalog 名称	(无，必须填写)
catalog-database	是	数据库名称	default
catalog-table	是	目标表名，与实际表名保持一致	无
uri	Hive catalog 必填	Hive metastore 地址	(无，Hive catalog 下必须填写)
bucket	是	目标表的桶数量，建议结合数据规模和并发写入能力配置	(无，必须填写)
bucket-key	是	分桶键，用于决定数据写入时落到哪个 Bucket。建议选择分布较均匀、相对稳定的字段，通常可使用主键字段或主键子集	默认主键，建议显示设置合理业务字段作为 bucket-key
bucket.discovery-interval	否	动态监测新增的 BucketShadingGroup	300s
startup.mode	否	latest: 从最新读 earliest：从最早读 from-timestamp: 指定时间戳读 Scan.timestamp-millis: 在 "timestamp" 启动模式下，启动时间的毫秒时间戳	latest

scan.timestamp-millis	否	scan.timestamp-millis 模式下，设置启动时间戳。示例：1767196800000	无

使用示例

```
-- 创建 Setats CDC 源表
CREATE TABLE `demo_setats_table1` (
  `id` STRING,
  `dt` STRING,
  `name` STRING,
  `address` STRING,
  PRIMARY KEY (`id`, `dt`) NOT ENFORCED
) WITH (
  'connector' = 'setats-cdc',
  'warehouse' = 'cosn://<bucket>/warehouse',
  'catalog-name' = 'setats',
  'catalog-type' = 'hive',
  'uri' = 'thrift://<metastore-host-1>:7004,thrift://<metastore-host-
2>:7004',
  'catalog-database' = 'testdb',
  'catalog-table' = 'demo_setats_table1',
  'manager-url' = '<Manager Url>',
  'bucket.discovery-interval' = '30s',
  'startup.mode' = 'earliest'
);
-- 创建调试 Sink 表
CREATE TABLE `logger_sink_table` (
  `id` STRING,
  `dt` STRING,
  `name` STRING,
  `address` STRING
) WITH (
  'connector' = 'logger',
  'all-changelog-mode' = 'true',
  'print-identifier' = 'DebugData'
);
-- 执行 CDC 读取
INSERT INTO `logger_sink_table`
```

```
SELECT `id`, `dt`, `name`, `address`
FROM `demo_setats_table1`;
```

查看消费结果

logger connector 会将 CDC 结果打印到 TaskManager 日志中。发布作业后，可在 Flink UI / Oceanus 控制台查看对应日志，确认是否已持续收到变更事件。

注意事项

setats-cdc 读取的是 Changelog ，而不是静态快照。
Changelog 的保留时间由集群参数 changelog.retention.ms 控制，默认 24 小时；需要确保下游消费延迟不超过该时长。

Flink Lookup Join

最近更新时间：2026-04-16 11:28:52

概述

Setats 支持高效的 Lookup Join，可以将 Setats 表作为维表，通过持续消费 changelog 并在本地构建缓存，实现流式事实表与维表的实时关联查询。

前提条件

- 已创建 Setats 集群并完成 Warehouse 配置。
- 已获取 manager-url。
- 维表数据已写入 Setats，且具备可持续消费的 Changelog。
- Flink 运行环境已完成相关依赖准备。

维表建表语法

```
CREATE TABLE `user_dim` (
  `id` BIGINT,
  `dt` STRING,
  `name` STRING,
  `age` INT,
  `address` STRING,
  PRIMARY KEY (`id`, `dt`) NOT ENFORCED
) WITH (
  'connector' = 'setats-cdc',
  'warehouse' = 'cosn://<bucket>/warehouse',
  'catalog-name' = 'setats',
  'catalog-type' = 'hive',
  'uri' = 'thrift://<metastore-host-1>:7004,thrift://<metastore-host-
2>:7004',
  'catalog-database' = 'testdb',
  'catalog-table' = 'user_dim',
  'manager-url' = '<Manager Url>',
  'lookup.force-local' = 'true',
  'lookup.bucket.discover.interval' = '10s',
  'lookup.refresh.async' = 'false',
  'lookup.table.store-type' = 'ROCKSDB',
  'lookup.table.store-cache-size' = '1024'
```

```
);
```

Lookup Join 参数说明

参数	说明	默认值
lookup.force-local	是否强制本地 Lookup。当前请固定设置为 `true`	true
lookup.continuous.discovery-interval	维表 Changelog 刷新间隔	10s
lookup.bucket.discover.interval	新 Bucket / 分区发现间隔	10s
lookup.refresh.async	是否异步刷新本地维表缓存	false
lookup.table.store-type	本地存储类型，可选 ROCKSDB 或 HEAP	ROCKSDB
lookup.table.store-cache-size	ROCKSDB 模式下额外 Heap Cache 可保存的 Join Key 数量	1024

使用示例：

```
-- 创建流表
CREATE TABLE `order_stream` (
  `order_id` BIGINT,
  `user_id` BIGINT,
  `dt` STRING,
  `amount` DECIMAL(10, 2),
  `proctime` AS PROCTIME()
) WITH (
  'connector' = 'kafka'
  -- 省略其他 Kafka 参数
);
-- 创建 Setats 维表
CREATE TABLE `setats_user_info_lookup` (
  `user_id` BIGINT,
  `user_name` STRING,
  `user_level` INT,
  `city` STRING,
```

```
  PRIMARY KEY (`user_id`) NOT ENFORCED
) WITH (
  'connector' = 'setats-cdc',
  'warehouse' = 'cosn://<bucket>/warehouse',
  'catalog-name' = 'setats',
  'catalog-type' = 'hive',
  'uri' = 'thrift://<metastore-host-1>:7004,thrift://<metastore-host-
2>:7004',
  'catalog-database' = 'testdb',
  'catalog-table' = 'user_info',
  'manager-url' = '<Manager Url>',
  'lookup.force-local' = 'true',
  'lookup.bucket.discover.interval' = '10s',
  'lookup.refresh.async' = 'false',
  'lookup.table.store-type' = 'ROCKSDB',
  'lookup.table.store-cache-size' = '1024'
);
-- 执行 Lookup Join
SELECT
  o.order_id,
  o.amount,
  u.user_name,
  u.user_level,
  u.city
FROM `order_stream` AS o
JOIN `setats_user_info_lookup` FOR SYSTEM_TIME AS OF o.proctime AS u
ON o.user_id = u.user_id;
```

运行建议

存储类型选择：

- **ROCKSDB`：适合维表较大、Key 较多的场景。**
- **`HEAP`：适合维表较小、延迟更敏感的场景。**

Spark 作业开发
Spark 批式查询

最近更新时间：2026-04-16 11:28:52

概述

Setats 支持通过 Spark 进行批式数据查询。用户可以使用 Spark SQL 直接访问 Setats 表中的最新快照数据，用于离线分析、明细抽样、聚合统计和结果回查等场景。

前提条件

- 已创建 Setats 集群并完成 Warehouse 配置。
- 已获取 Setats Spark Connector Jar 包。
- 已确认 Spark 集群可访问 Warehouse 底层存储，以及 Hive Metastore（若使用 Hive Catalog）。
- 已将所需 Jar 包部署到 Spark 节点本地目录或 HDFS 可访问路径。
- 如果使用 EMR Spark 3.3 + Hive Catalog + COS 加速桶，建议提前准备：
  - setats-spark-bundle-<spark-version>_<scala-version>-<connector-version>.jar
  - Hive Metastore 可用地址
  - Warehouse 路径，例如 cosn://<bucket>/warehouse/

环境准备

登录 Spark 节点

以 EMR Spark 3.3 为例，建议先登录 Spark 所在节点，并确认 Connector Jar 已上传到本地或 HDFS。如果 Jar 存放在本地目录，可直接通过 --jars 引用；如果存放在 HDFS，也可以在启动参数中引用对应路径。

启动 Spark SQL

以下示例使用 Hive Catalog：

```
sh /usr/local/service/spark/bin/spark-sql \
  --jars /usr/local/service/spark/setats-spark-bundle-3.3_2.12-
<connector-version>.jar \
  --conf spark.sql.setats.force.read.from.service=true \
  --conf spark.sql.setats.force.read.service.data=true \
  --conf spark.sql.session.timeZone='UTC' \
  --conf spark.driver.cores=4 \
  --conf spark.driver.memory=4g \
```

```
  --conf spark.executor.memory=4g \
  --conf spark.executor.cores=4 \
  --conf
spark.sql.extensions=com.tencent.oceanus.spark.extensions.SetatsSparkSes
sionExtensions \
  --conf spark.sql.catalog.setats=com.tencent.oceanus.spark.SparkCatalog
\
  --conf spark.sql.catalog.setats.type=hive \
  --conf spark.sql.catalog.setats.warehouse=cosn://<bucket>/warehouse/ \
  --conf spark.sql.catalog.setats.uri=thrift://<metastore-host-
1>:7004,thrift://<metastore-host-2>:7004
```

如果使用 Hadoop Catalog，可将 spark.sql.catalog.setats.type 改为 hadoop，并根据实际情况省略 uri。

Spark Catalog 参数说明

参数	说明
spark.sql.setats.force.read.from.service	开启元数据读 Setats 服务
spark.sql.setats.force.read.service.data	开启内存数据读 Setats 服务
spark.sql.session.timeZone	Spark SQL 会话时区。进行时间旅行查询时建议设置为 UTC，避免按时间点读取历史快照时出现时区偏移
spark.sql.extensions	Setats Spark Session 扩展类：com.tencent.oceanus.spark.extensions.SetatsSparkSessionExtensions
spark.sql.catalog.setats	Setats Spark Catalog 实现类：com.tencent.oceanus.spark.SparkCatalog
spark.sql.catalog.setats.type	Catalog 类型，如 hive 或 hadoop
spark.sql.catalog.setats.warehouse	Setats Warehouse 地址
spark.sql.catalog.setats.uri	Hive Metastore 地址，仅 Hive Catalog 需要

查询示例

基本查询：

```
SELECT *
FROM `setats`.`testdb`.`demo_setats_table1`;
```

指定分区查询：

```
SELECT *
FROM `setats`.`testdb`.`demo_setats_table1`
WHERE dt = '20260319';
```

聚合查询：

```
SELECT
  dt,
  COUNT(*) AS cnt
FROM `setats`.`testdb`.`demo_setats_table1`
GROUP BY dt;
```

查询同步结果表示例：

```
SELECT *
FROM `setats`.`testdb`.`mysql_user_behavior_sink`;
```

系统表

Spark 批式查询支持访问 Setats 系统表，用于查看表的快照、Manifest、数据文件和索引文件等元数据信息。当前支持的系统表如下：

系统表	说明
snapshots	查询表的快照历史
manifests	查询 Manifest 文件信息
manifests_detail	查询 Manifest 明细信息
files	查询当前快照对应的数据文件信息
files_with_dv	查询带删除向量的数据文件信息
index_files	查询索引文件信息
bucket_manifests	查询 Bucket 与 Manifest 的关联信息

例如，可通过系统表查看目标表的快照历史：

```
SELECT *
FROM `setats`.`testdb`.`demo_setats_table1`.`snapshots`;
```

分区信息查询

当前在 Spark SQL 中查询分区信息时，需要使用 SHOW PARTITIONS 语句：

```
SHOW PARTITIONS `setats`.`testdb`.`demo_setats_table1`;
```

时间旅行

Setats 支持在 Spark 中基于历史快照执行时间旅行查询，适用于历史数据回查、结果核对和问题定位等场景。
如果需要按快照版本查询历史数据，建议先通过 snapshots 系统表确认目标快照，再执行时间旅行查询。
例如：

```
SELECT *
FROM `setats`.`testdb`.`demo_setats_table1`
VERSION AS OF <snapshot_id>;
```

如果需要按时间点查询历史数据，可使用时间点方式访问历史快照：

```
SELECT *
FROM `setats`.`testdb`.`demo_setats_table1`
TIMESTAMP AS OF '<yyyy-MM-dd HH:mm:ss>';
```

表名格式

Spark 中访问 Setats 表的格式为：

```
`<catalog_name>`.`<database>`.`<table_name>`
```

例如：setats.testdb.demo_setats_table1

数仓外表查询
Doris外表查询

最近更新时间：2026-04-16 11:28:52

概述

Setats 支持通过 Doris 外表方式进行 OLAP 查询。用户可以在 Doris 中创建 Setats Catalog，直接访问 Setats 表的列存数据，用于即席分析、报表查询和离线聚合等场景。

前提条件

- 已创建 Setats 集群并完成 Warehouse 配置。
- 已部署 Doris 集群。
- 若使用 Hive Catalog 或元数据加速桶，Doris 还需要访问 Hive Metastore。
- 若底层存储启用了 Kerberos / CHDFS / COS 加速桶，需要提前准备对应配置文件和认证信息。

场景一：HDFS / Hadoop Catalog

如果 Setats 使用 Hadoop Catalog 或基于 HDFS 的元数据管理，可参考以下方式创建 Catalog：

```
CREATE CATALOG setats COMMENT 'setats catalog' PROPERTIES (
  'type' = 'setats',
  'setats.catalog.type' = 'hadoop',
  'warehouse' = 'hdfs://<nameservice>/usr/setats',
  'dfs.nameservices' = '<nameservice>',
  'dfs.ha.namenodes.<nameservice>' = 'nn1,nn2',
  'dfs.namenode.rpc-address.<nameservice>.nn1' = '<nn1-host>:4007',
  'dfs.namenode.rpc-address.<nameservice>.nn2' = '<nn2-host>:4007',
  'dfs.client.failover.proxy.provider.<nameservice>' =
'org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvid
er'
);
```

如果 EMR 开启了 Kerberos，可继续追加以下属性：

```
'hadoop.security.authentication' = 'kerberos',
'hadoop.kerberos.keytab' = '/usr/local/service/doris/kerberos.keytab',
'hadoop.kerberos.principal' = '<your-principal>'
```

场景二：Hive Metastore + COS 加速桶

如果 Setats 使用 Hive Metastore，并且底层数据或元数据依赖 COS 加速桶，可参考以下配置：

```
CREATE CATALOG setats COMMENT 'setats catalog' PROPERTIES (
  'type' = 'setats',
  'setats.catalog.type' = 'hms',
  'hive.metastore.uris' = 'thrift://<metastore-host-
1>:7004,thrift://<metastore-host-2>:7004',
  'cos.endpoint' = 'cos.<region>.myqcloud.com',
  'cos.access_key' = '<SecretId>',
  'cos.secret_key' = '<SecretKey>',
  'fs.ofs.bucket.region' = '<region>',
  'fs.ofs.user.appid' = '<AppId>',
  'fs.ofs.tmp.cache.dir' = '/tmp/chdfs/',
  'fs.ofs.upload.flush.flag' = 'true',
  'broker.name' = '<BrokerName>'
);
```

Catalog 参数说明

参数	说明
type	Catalog 类型，固定为 setats
setats.catalog.type	Setats Catalog 类型，如 hadoop 或 hms
warehouse	Setats Warehouse 地址，Hadoop Catalog 常用
hive.metastore.uris	Hive Metastore 地址，Hive / HMS 场景需要
dfs.nameservices	HDFS NameService 名称
dfs.ha.namenodes.<nameservice>	HA NameNode 列表
dfs.namenode.rpc-address.<nameservice>.nn*	NameNode RPC 地址
dfs.client.failover.proxy.provider.<nameservice>	HDFS HA Failover Provider
cos.endpoint	COS 访问域名

cos.access_key / cos.secret_key	COS 访问凭据；需要有 QcloudCOSListOnly 和 QcloudCOSDataReadOnly 权限
fs.ofs.bucket.region	加速桶地域
fs.ofs.user.appid	腾讯云 AppID
broker.name	Doris Broker 名称

查询数据

切换 Catalog 查询

```
SWITCH setats;
SHOW DATABASES;
USE `testdb`;
SELECT * FROM `demo_setats_table1`;
```

跨 Catalog 查询

```
SELECT *
FROM `setats`.`testdb`.`demo_setats_table1`;
```

StarRocks 外表查询

最近更新时间：2026-04-16 11:28:52

概述

Setats 支持通过 StarRocks 外表方式进行 OLAP 查询。用户可以在 StarRocks 中创建 Setats External Catalog，直接查询 Setats 表的列存快照数据，满足实时报表、交互式分析和明细回查等需求。

> **说明：**
> 如果需要使用 Setats 服务，那么需要升级 StarRocks 集群版本，请提交工单申请。

前提条件

- 已创建 Setats 集群并完成 Warehouse 配置。
- 已部署 StarRocks 集群，并确认 FE / BE 节点能够访问 HDFS / COS 等底层存储。
- 如果 Warehouse 使用 Hive Catalog，需要额外确认 StarRocks 到 Hive Metastore 的连通性。
- 如果涉及 Kerberos、CHDFS 或 COS 加速桶，需要提前准备相关网络、凭据和 Hadoop 配置。

场景一：HDFS / Hadoop Catalog

以下示例适用于 Hadoop Catalog / HDFS 场景：

```
CREATE EXTERNAL CATALOG setats
PROPERTIES (
  'type' = 'setats',
  'setats.catalog.type' = 'hadoop',
  'warehouse' = 'hdfs://<nameservice>/usr/setats',
  'dfs.nameservices' = '<nameservice>',
  'dfs.ha.namenodes.<nameservice>' = 'nn1,nn2',
  'dfs.namenode.rpc-address.<nameservice>.nn1' = '<nn1-host>:4007',
  'dfs.namenode.rpc-address.<nameservice>.nn2' = '<nn2-host>:4007',
  'dfs.client.failover.proxy.provider.<nameservice>' =
'org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvid
er'
);
```

如果 EMR 启用了 Kerberos，请根据实际环境补充 Kerberos 相关参数。

场景二：Hive Metastore + COS 加速桶

如果 Setats 使用 Hive Metastore，并且底层数据或元数据依赖 COS 加速桶，可参考以下配置：

```
CREATE EXTERNAL CATALOG setats
PROPERTIES
(
    "type" = "setats",
    "setats.catalog.type" = "hive",
    "hive.metastore.uris" =
"thrift://172.28.39.128:7004,thrift://172.28.39.118:7004",
    'tencent.cos.access_key' = '',
    'tencent.cos.secret_key' = '',
    'tencent.cos.endpoint' = ''
);
```

Catalog 参数说明

参数	说明
type	Catalog 类型，固定为 setats
setats.catalog.type	Setats Catalog 类型，如 hadoop
warehouse	Setats Warehouse 地址
dfs.nameservices	HDFS NameService 名称
dfs.ha.namenodes.<nameservice>	HA NameNode 列表
dfs.namenode.rpc-address.<nameservice>.nn*	NameNode RPC 地址
dfs.client.failover.proxy.provider.<nameservice>	HDFS HA Failover Provider
tencent.cos.access_key/tencent.cos.secret_key	COS 访问凭据；需要有 QcloudCOSListOnly 和 QcloudCOSDataReadOnly 权限
tencent.cos.endpoint	COS 访问域名，比如 cos.ap-guangzhou.myqcloud.com

查询数据

切换 Catalog 查询

```
SET CATALOG setats;
SHOW DATABASES;
USE `testdb`;
SELECT * FROM demo_setats_table1;
```

全限定名查询

```
SELECT *
FROM `setats`.`testdb`.`demo_setats_table1`;
```

监控指标
查看监控指标

最近更新时间：2026-04-16 11:28:52

概述

Setats 提供了全面的监控指标系统，帮助用户实时了解集群的运行状态、资源使用情况和业务性能表现。通过监控指标，用户可以：

- 实时监控: 实时查看集群和节点的运行状态
- 资源分析: 分析 CPU、内存、磁盘、网络等资源使用情况
- 性能优化: 根据监控数据优化集群配置和作业参数
- 故障诊断: 快速定位和诊断集群故障
- 容量规划: 基于历史数据预测资源需求，提前规划扩容

Setats 的监控系统采用分层架构，从底层的基础设施监控到上层的业务指标监控，提供了多维度的监控能力。监控数据支持实时查看和历史回溯，用户可以根据时间范围选择查看特定时间段的数据。

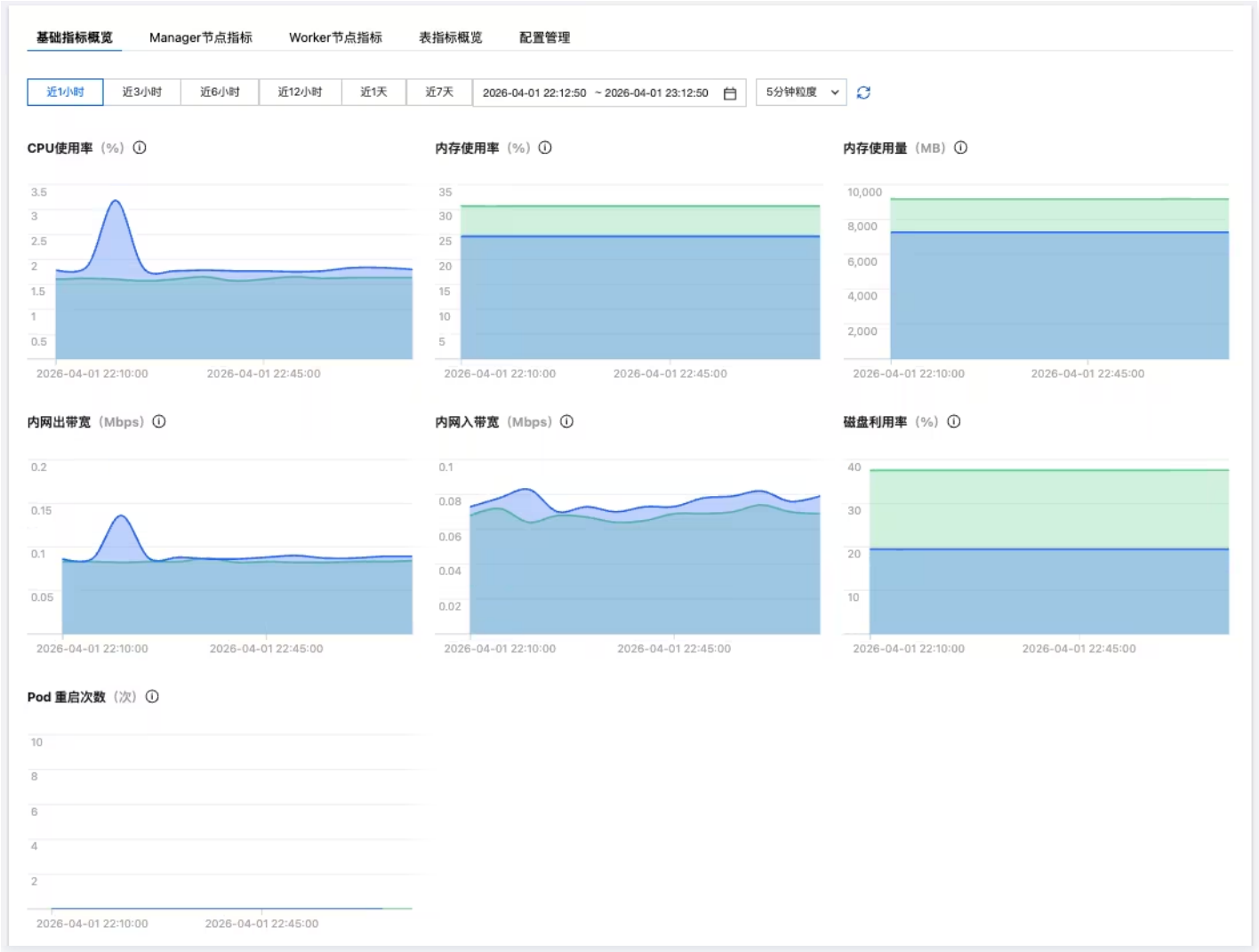
监控指标分类

1. 节点资源监控

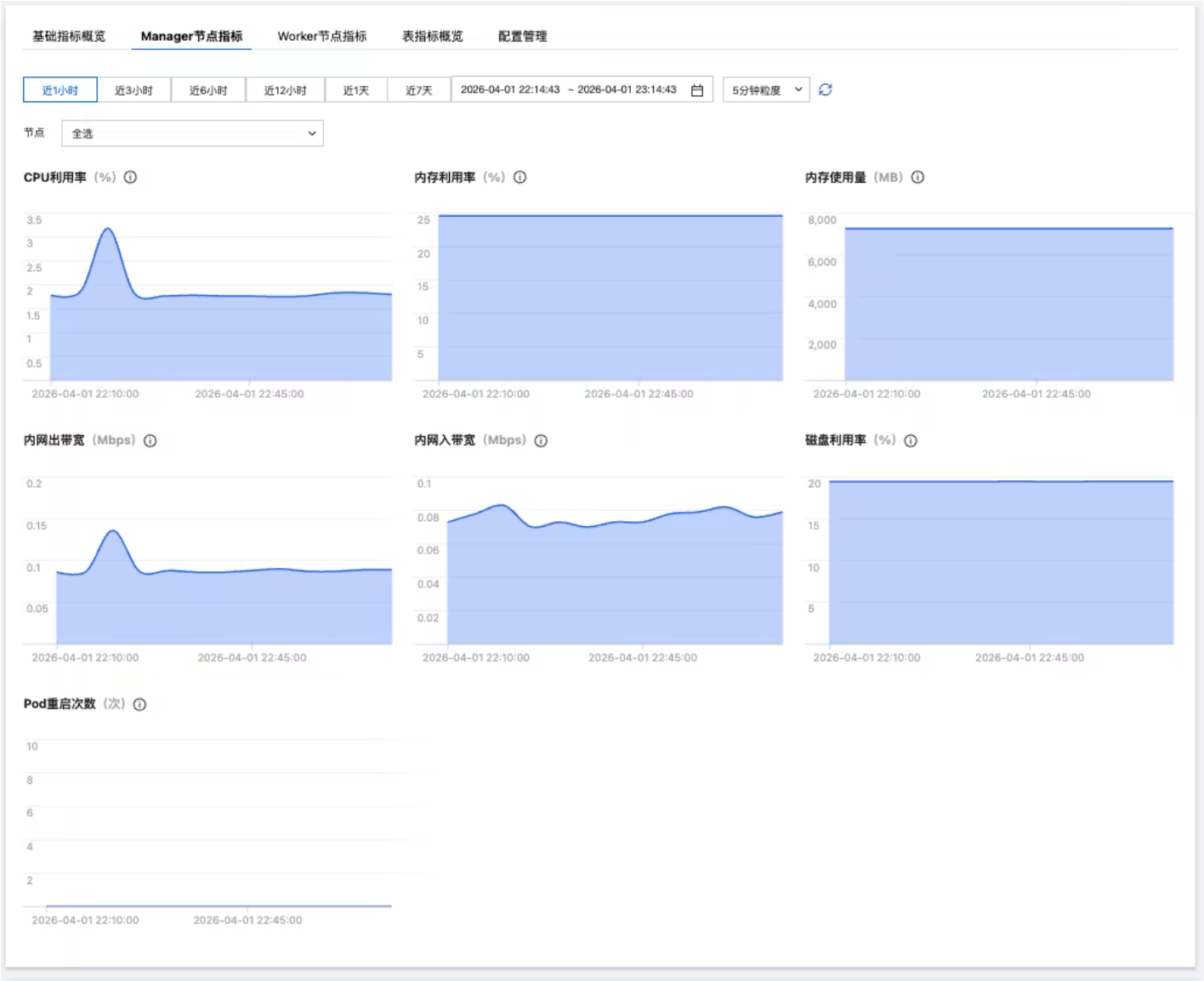
   监控节点的硬件资源使用情况，包括 CPU、内存、磁盘、网络等基础资源。
2. 业务监控

   监控 Setats 的业务指标。

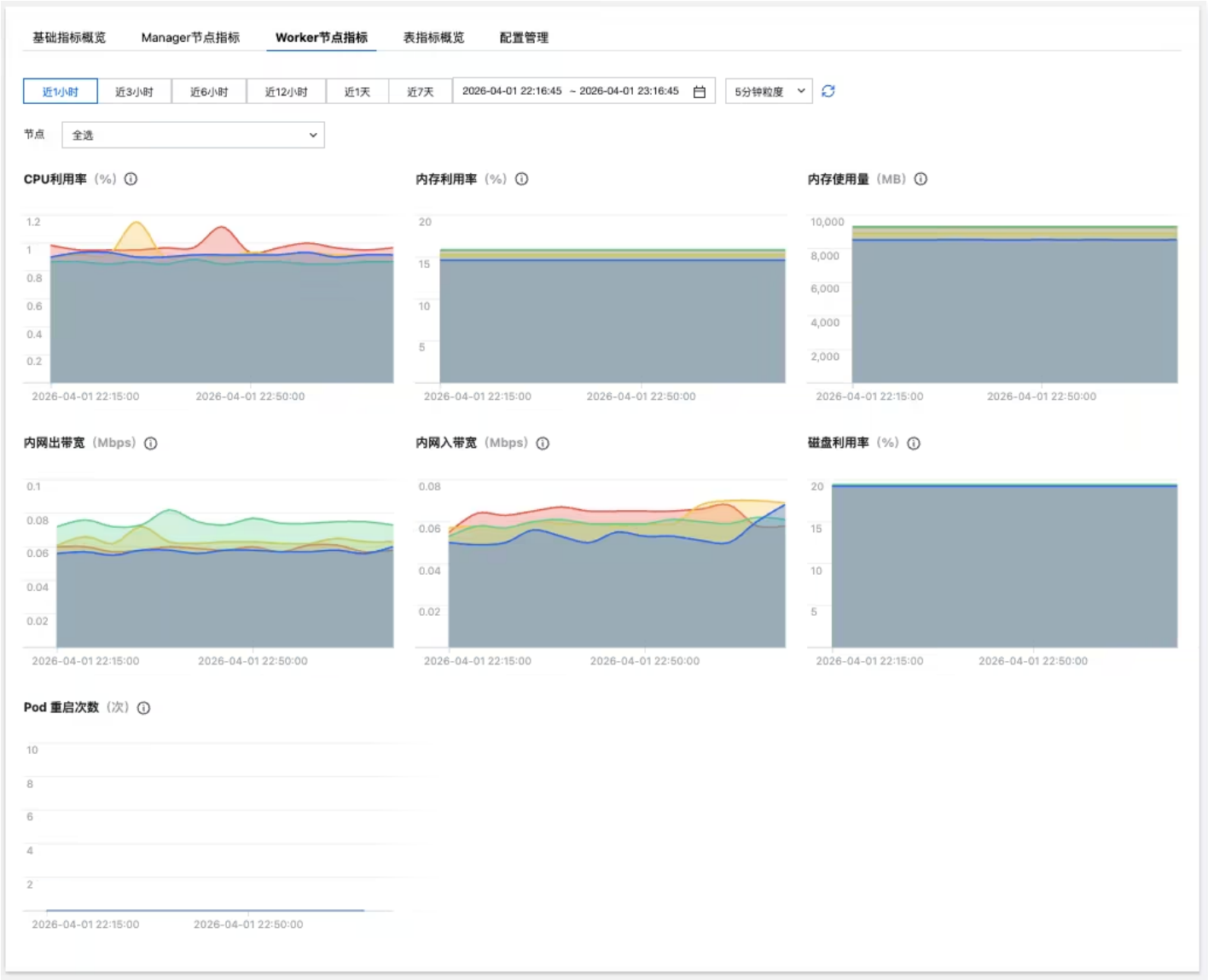
基础指标概览



Manager 节点指标


基础指标概览
Manager节点指标
Worker节点指标
表指标概览
配置管理
近1小时
近3小时
近6小时
近12小时
近1天
近7天
2026-04-01 22:14:43 ~ 2026-04-01 23:14:43
5分钟粒度
节点
全选
CPU利用率（%）
内存利用率（%）
内存使用量（MB）
内网出带宽（Mbps）
内网入带宽（Mbps）
磁盘利用率（%）
Pod重启次数（次）
2026-04-01 22:10:00
2026-04-01 22:45:00

Worker 节点指标


基础指标概览
Manager节点指标
Worker节点指标
表指标概览
配置管理
近1小时
近3小时
近6小时
近12小时
近1天
近7天
2026-04-01 22:16:45 ~ 2026-04-01 23:16:45
5分钟粒度
节点
全选
CPU利用率（%）
内存利用率（%）
内存使用量（MB）
内网出带宽（Mbps）
内网入带宽（Mbps）
磁盘利用率（%）
Pod 重启次数（次）
2026-04-01 22:15:00
2026-04-01 22:50:00

表指标概览


基础指标概览
Manager节点指标
Worker节点指标
表指标概览
配置管理
近1小时
近3小时
近6小时
近12小时
近1天
近7天
2026-04-01 22:17:52 ~ 2026-04-01 23:17:52
5分钟粒度
Commit总次数（次）
Commit成功总次数（次）
Commit失败总次数（次）
2026-04-01 22:40:00
cluster-5fv1k54o 0
2026-04-01 22:15:00
2026-04-01 22:50:00