

State of Iceberg REST Catalogs

чего нам не хватает и как собрать Control Plane своими руками





Обо мне



CTO Data Platform
@ Островок



ex. Rosbank,
ex. Aliexpress



20+ лет
в разработке

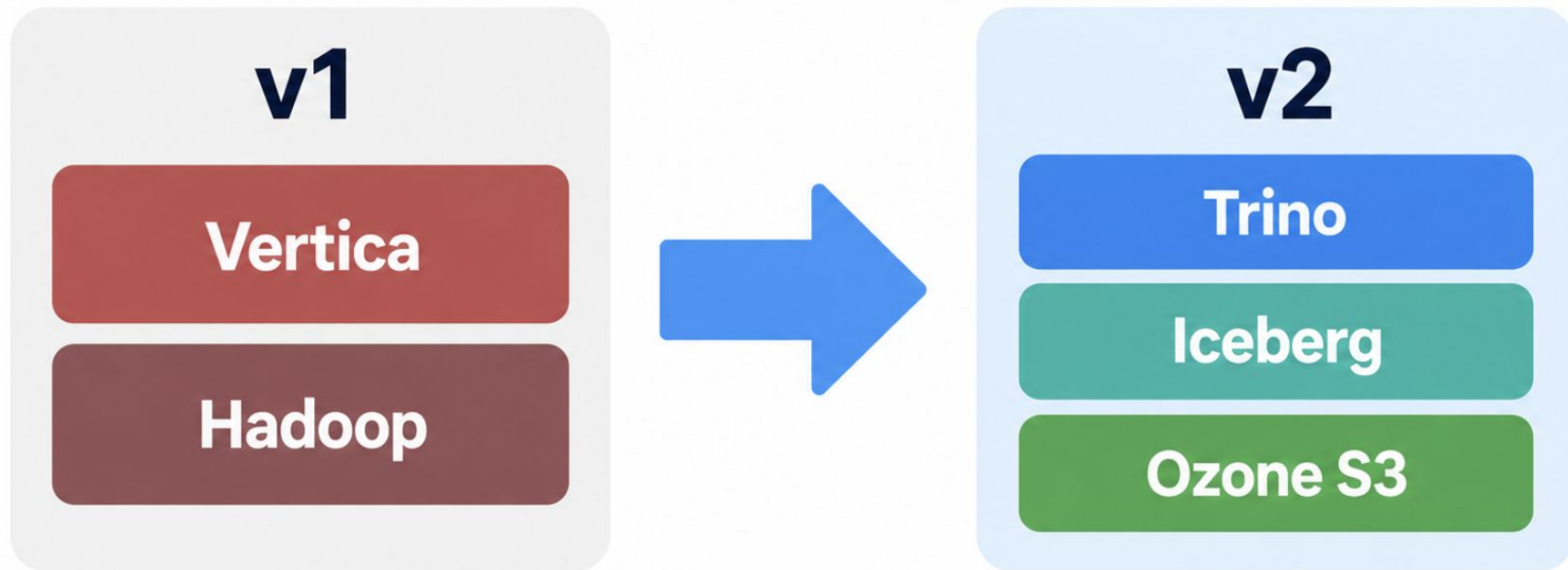


10+ лет
в областях работы
с данными и аналитикой

План доклада

1. Погружение в контекст
2. Борьба с артефактами и методы обслуживания
3. Фидбек луп и Control Plane
4. Процессная модель автоматизации обслуживания
5. Обзор рынка 2025
6. Что мы реализовали в нашем решении
7. Инкремент за год и рынок 2026
8. Какие функции ещё нужны в Control Plane
9. Решения, доступные сейчас
10. Где мы сейчас

миграция платформы на trino и iceberg



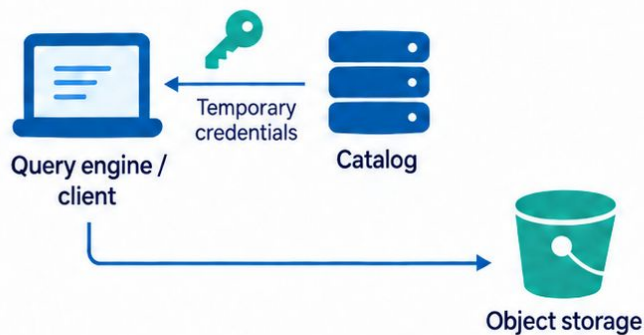
начнем с анатомии iceberg v.2 format



делегирование доступа к s3 хранилищу

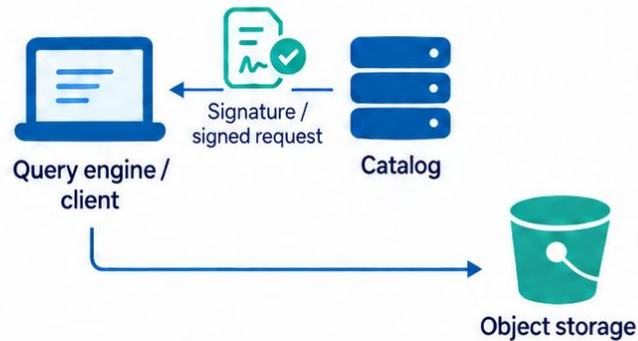
авторизация доступа к storage через REST-каталог

Vended credentials



- Каталог выдает временные ключи
- Движок читает storage напрямую

Remote signing



- Каталог не выдает ключи
- Каталог подписывает запрос на файл

стратегии обработки изменений строк в iceberg

CoW Copy-on-Write	MoR Merge-on-Read
 Только Data files	 +  Data files Delete files

стратегия MoR пишет delete-файлы

data-файл x

	id	name
1	124	bla
2	145	blo
3	176	bli
4	198	blu

Position delete файл

строка 3 в файле x

строка 5 в файле y

Equality delete файл

id = 176

name = blabla

процесс записи в iceberg catalog

Запись data и delete файлов

Движок записывает новые Parquet data-файлы, а также (при обновлениях/удалениях) может создавать delete-файлы (position или equality, v2).



1

2

Создание manifest entries

Фиксируются пути файлов (data и delete), число строк, партиции и статистики.



3

Создание / обновление manifest files

Entries собираются в Avro manifest-файлы (отдельно для data и delete файлов).



4

Создание manifest list

Новый Avro-файл указывает на все manifest-файлы этого snapshot.



5

Создание нового metadata.json

Создается новый metadata-файл с новым snapshot (указатель на manifest list, текущий snapshot id и другая метainформация).



6

Атомарный commit

Каталог переключает указатель со старого metadata.json на новый.

Здесь фиксируется транзакция

процесс чтения из iceberg catalog

1. Получение metadata.json

Движок запрашивает актуальный metadata.json для нужного снимка (обычно последний).



1

2

2. Чтение manifest list

Из metadata.json берутся ссылки на manifest list этого снимка.



3

3. Чтение manifest файлов

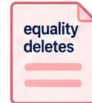
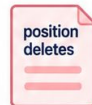
Движок читает manifest-файлы и отбирает только те data-файлы, которые нужны для запроса (по партициям и статистикам).



4

4. Учет delete файлов

Если есть удаления, движок читает соответствующие delete файлы (position deletes или equality deletes в Iceberg v2).



5

5. Чтение data файлов

Движок читает только нужные Parquet-файлы (колоночный формат), применяя column/partition pruning и статистики.



6

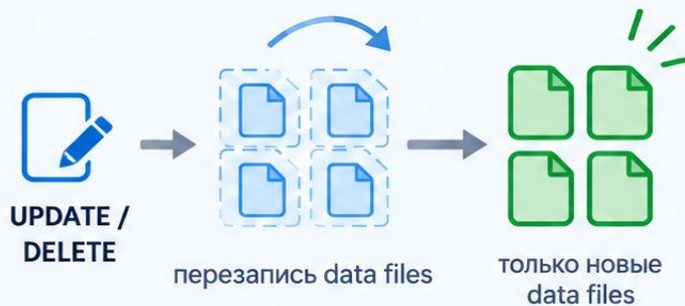
6. Возврат результата

Движок объединяет данные, применяет удаления и возвращает актуальные строки пользователю.



разница между CoW и MoR

Copy-on-Write

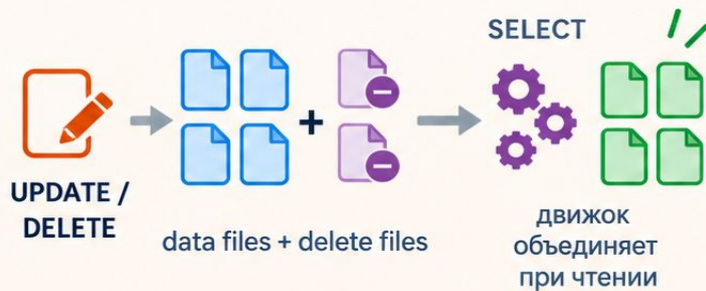


Запись:
дороже



Чтение:
дешевле

Merge-on-Read



Запись:
дешевле



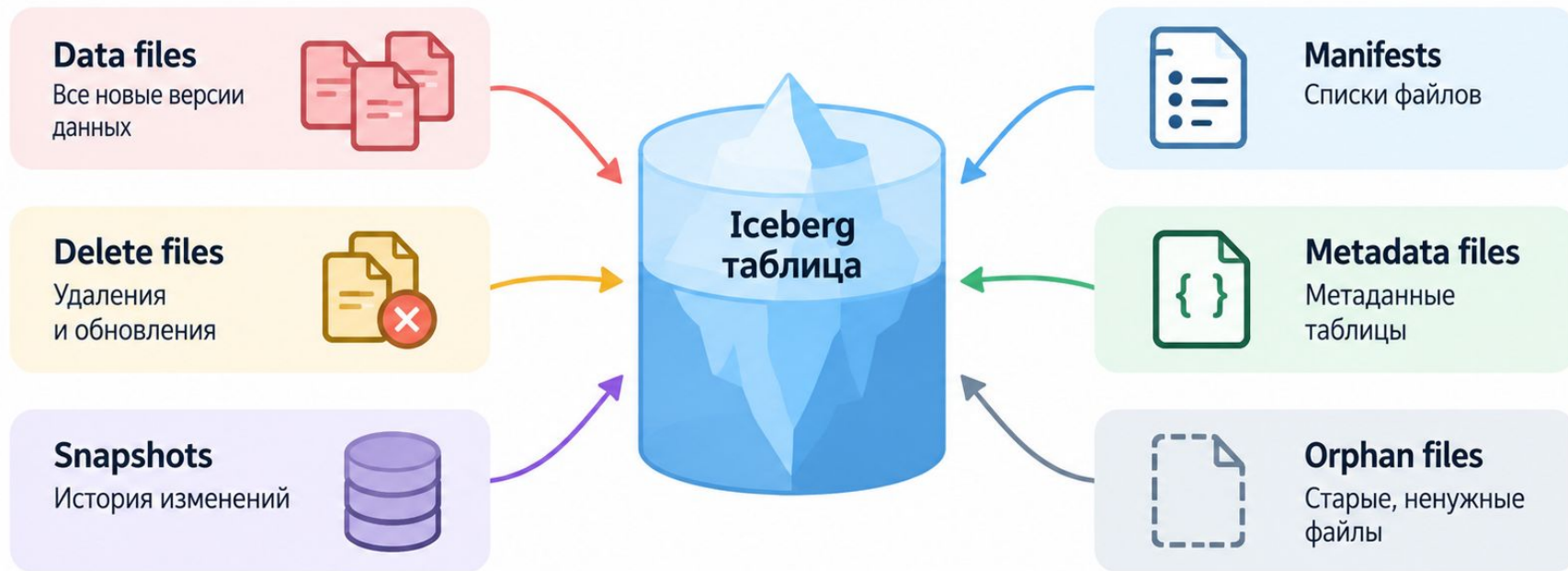
Чтение:
дороже

REST Metrics Reporting в Iceberg REST Catalog

Отчеты отправляет compute engine (Trino, Spark и др.)



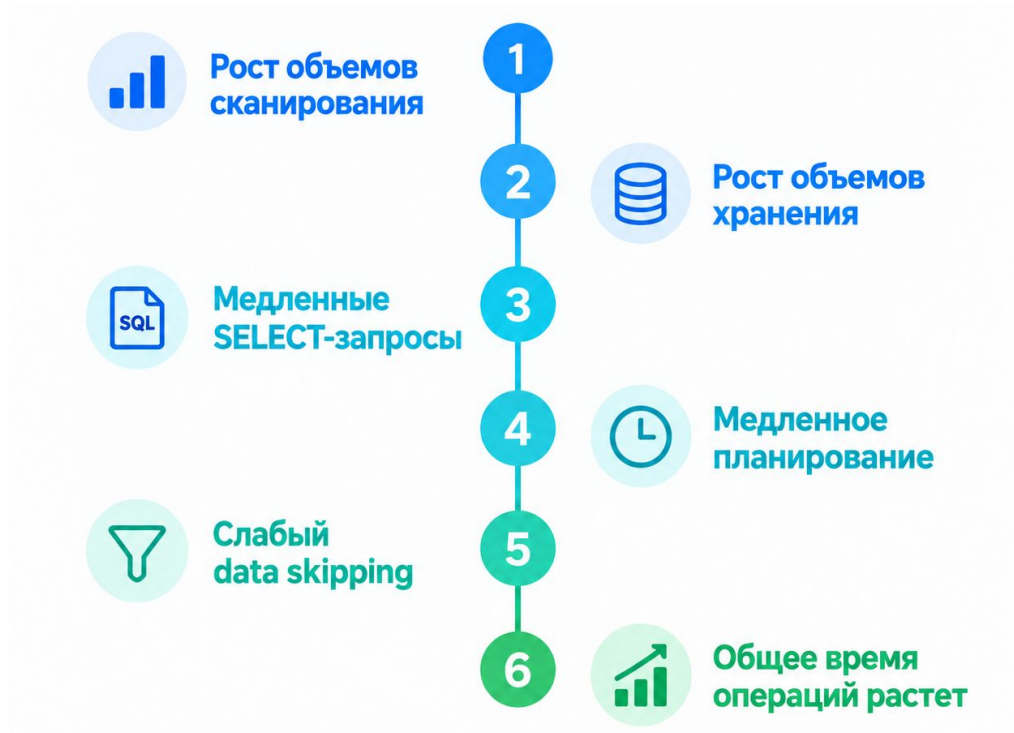
таблицы iceberg постоянно накапливают артефакты



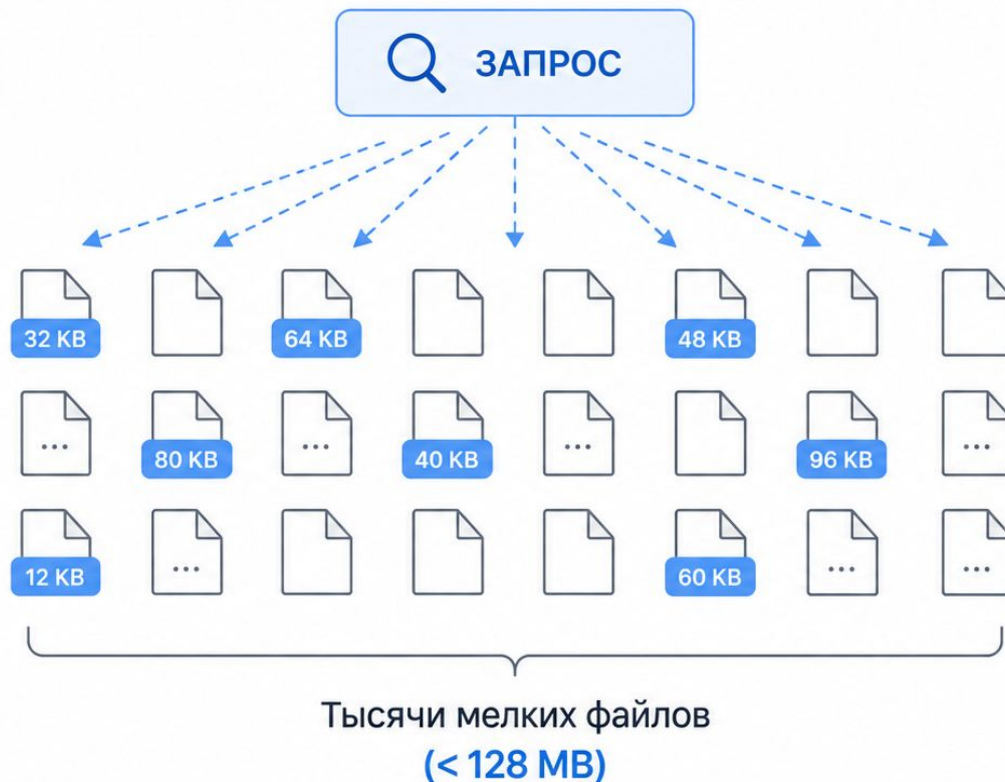
Со временем всего становится больше

если не выполнять обслуживание

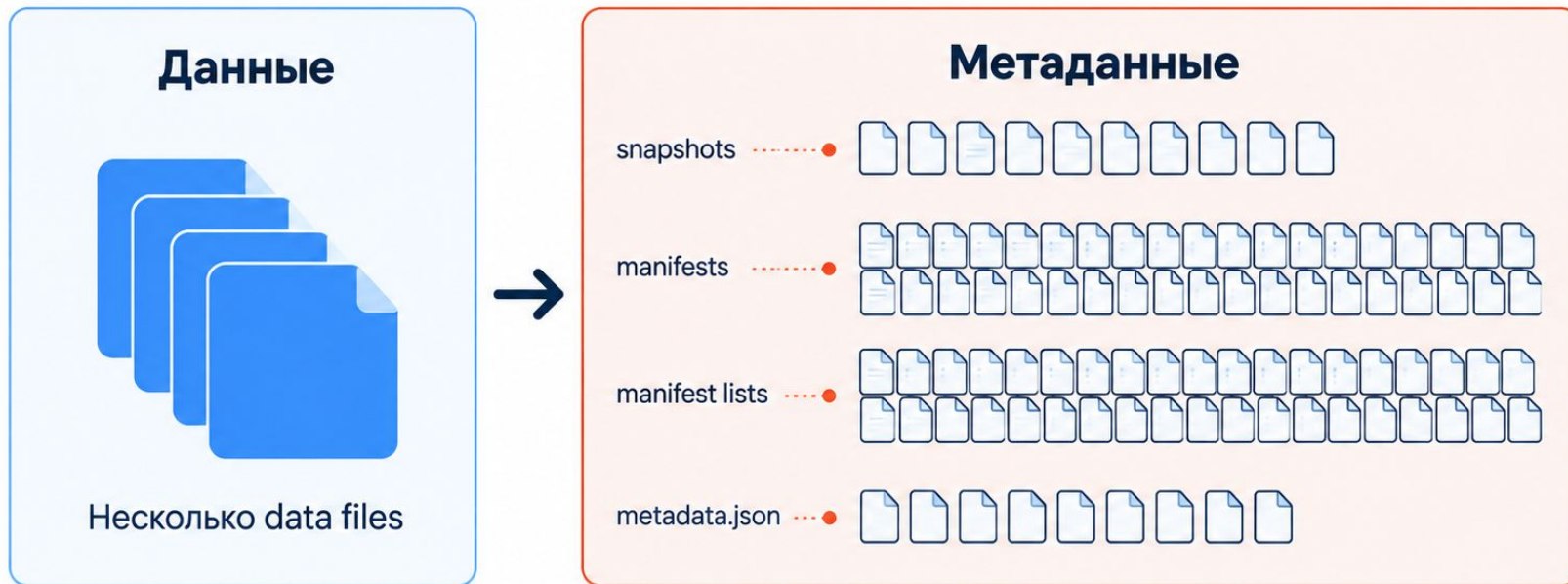
топ-6 последствий большого скопления артефактов



маленькие файлы заставляют движок много читать из s3

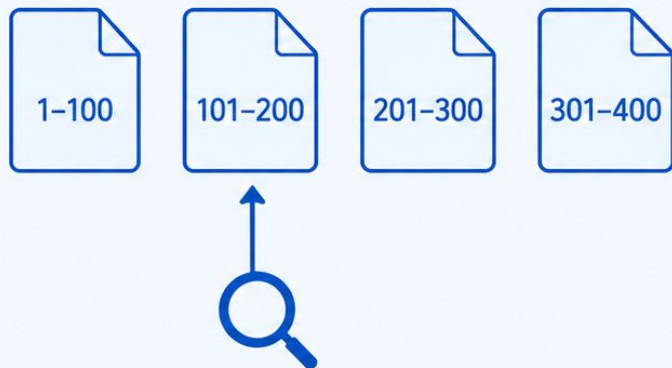


распухание метаданных удлиняет планирование



проблемы с сортировкой мешают data-skipping

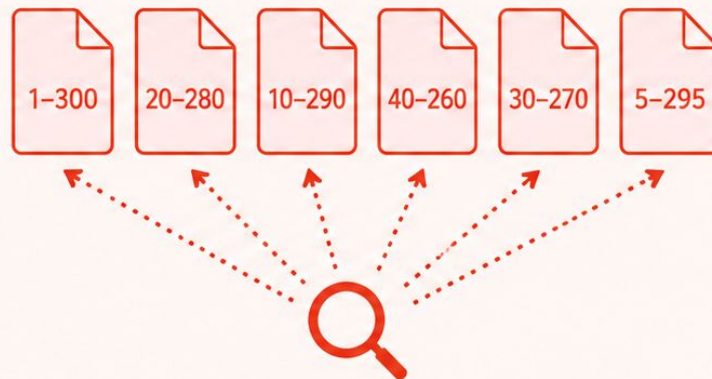
Как должно быть



Диапазоны почти не пересекаются
можно пропустить лишние файлы



Что не так



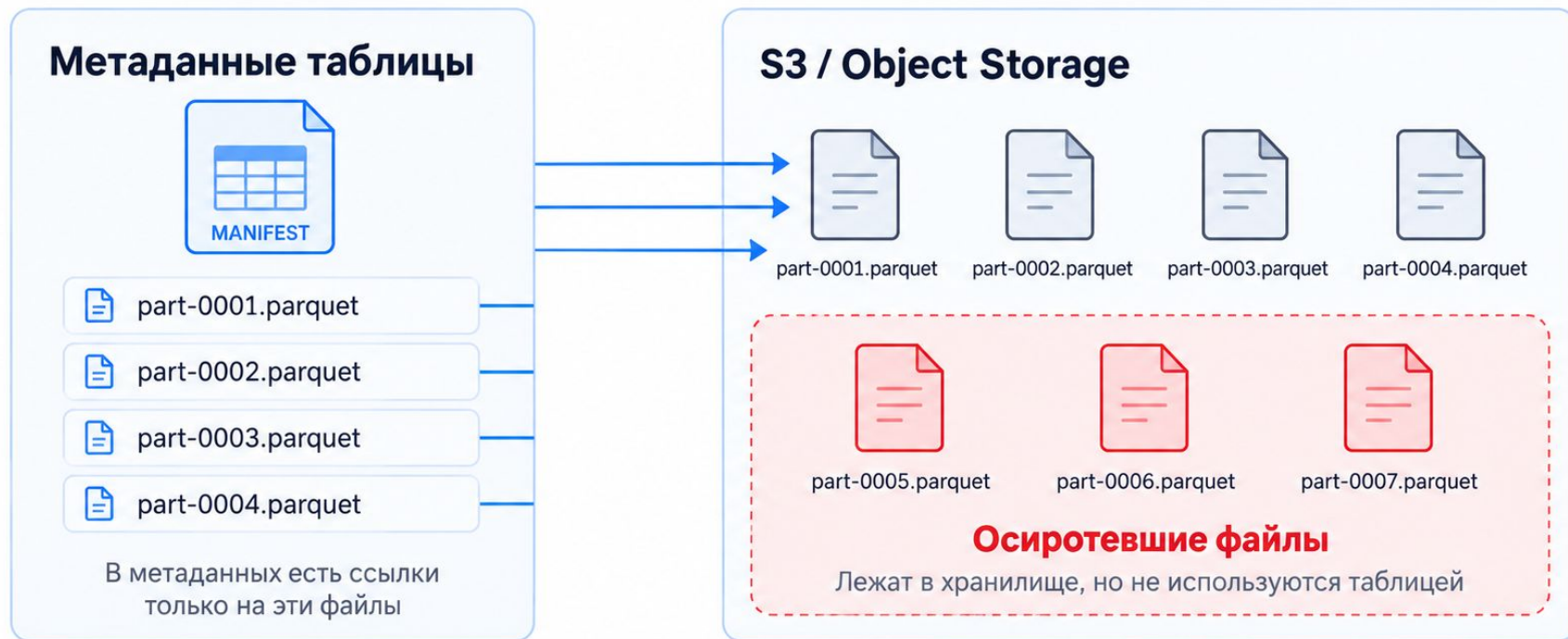
Сильное перекрытие min/max
приходится читать много файлов

множество delete-файлов замедляет select

- Data-файлы читаются вместе с delete-файлами
- Движок применяет удаления во время чтения
- Растут накладные расходы на планирование
- SELECT работает медленнее
- Чем больше delete-файлов, тем хуже чтение



осиротевшие файлы заполняют хранилище



Итог: место в хранилище растёт, хотя таблица эти файлы уже не использует.

5 методов борьбы с ростом артефактов

- Rewrite data files — уплотнение data-файлов
- Rewrite delete files — перепись delete-файлов
- Rewrite manifests — уплотнение manifest-файлов
- Expire snapshots — удаление старых snapshot'ов
- Delete orphan files — очистка файлов-сирот

компакция уменьшает кол-во мелких файлов

rewrite_data_files

- Объединяет мелкие data-файлы
- Уменьшает число файлов
- Выравнивает размер файлов
- Можно запускать по таблице или по партициям

уплотнение манифестов оптимизируем метаданные



Метаданные
компактнее



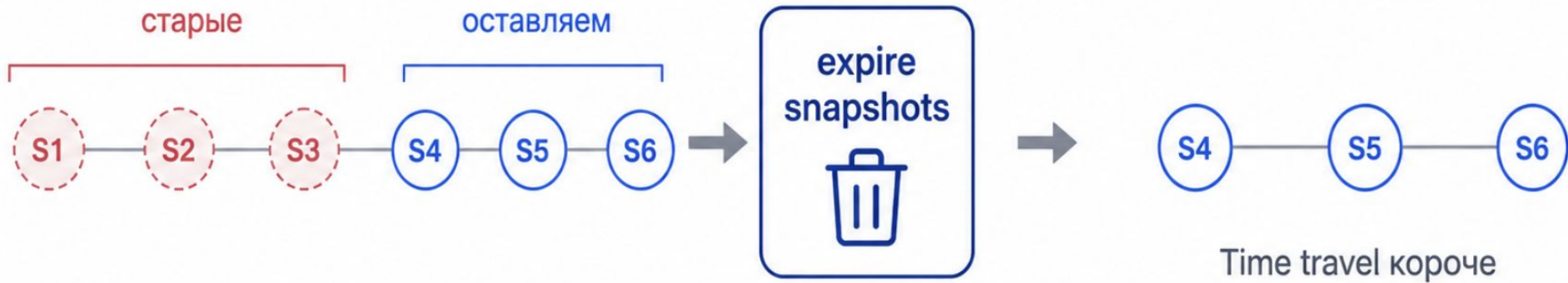
Быстрее
planning



Data files
не меняются

Работает по метаданным

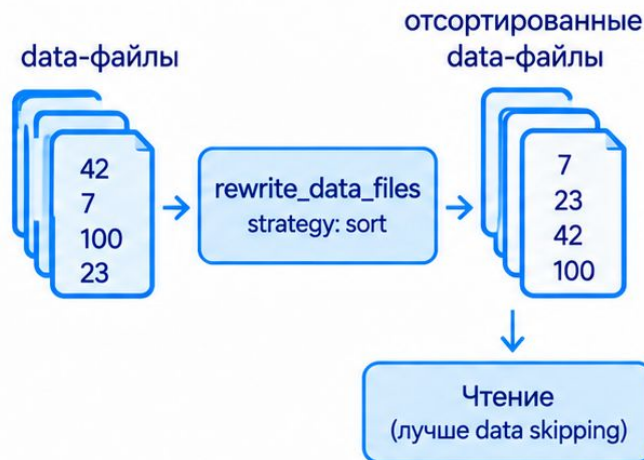
экспирация снапшотов удаляет старые снпшоты



стратегия sort помогает улучшать data-skipping

rewrite_data_files: стратегия sort

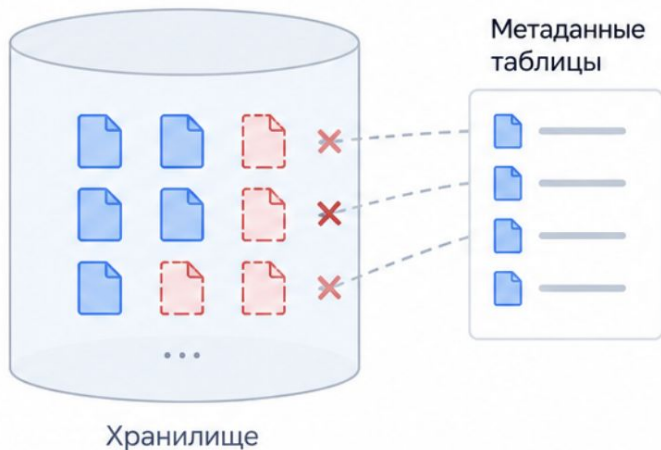
- Data-файлы переписываются и сортируются по ключу
- Близкие значения кладутся рядом
- Min/Max диапазоны становятся точнее
- Data skipping работает лучше
- SELECT с фильтрами работает быстрее



удаление файлов-сирот очищает мусор на s3

До

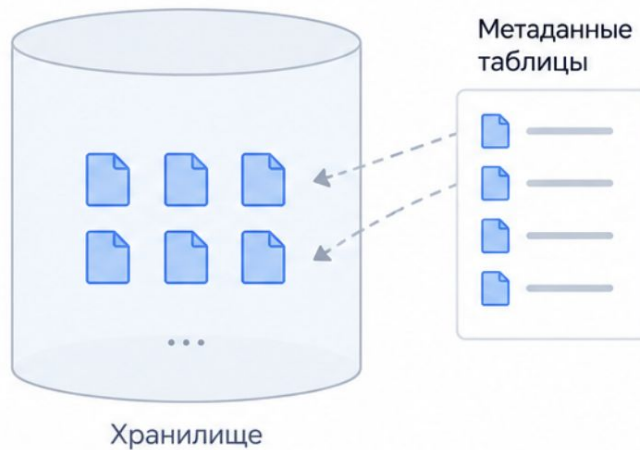
В хранилище есть лишние файлы
(не используются таблицей)



-  Используется таблицей
-  Файл-сирота
(не используется)

После

Остаются только нужные файлы

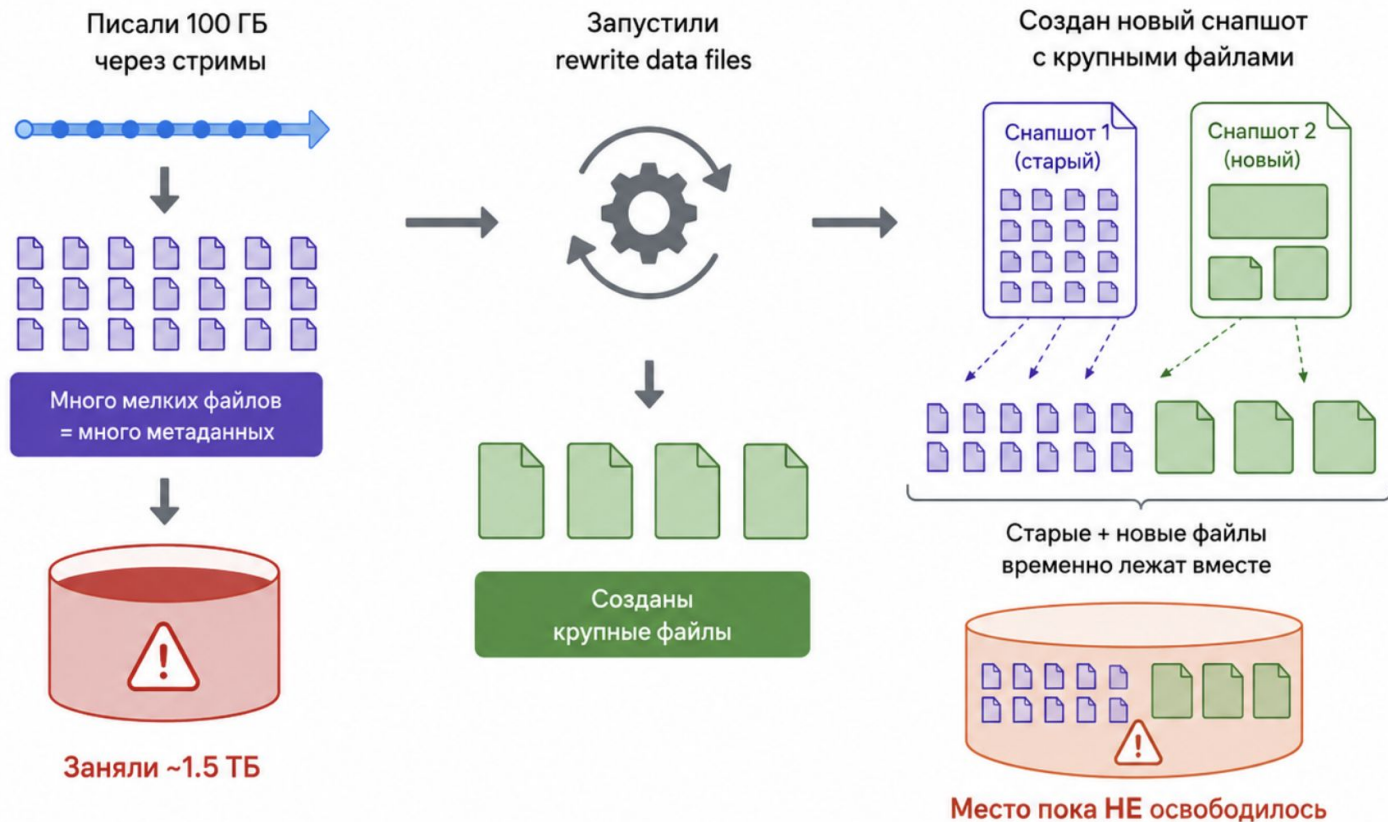


Освобождено место
в хранилище

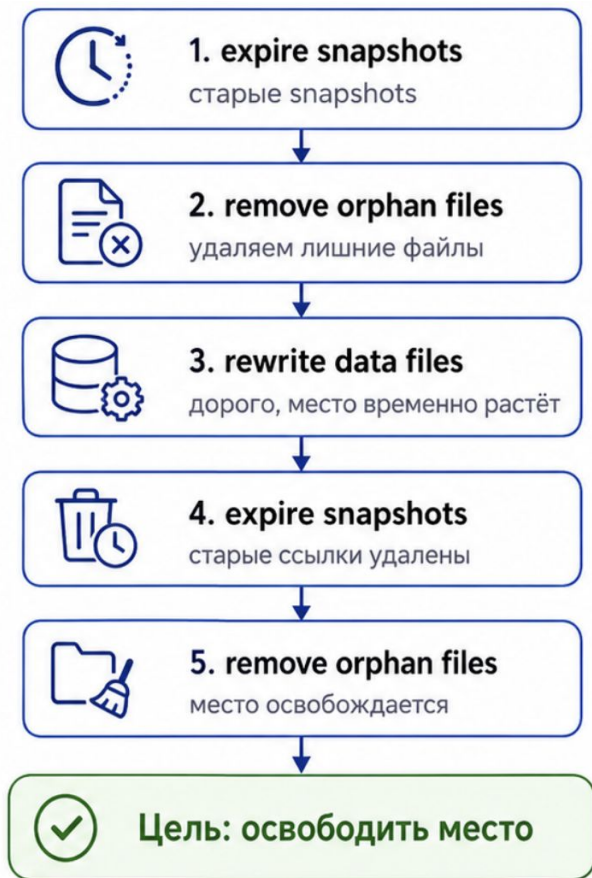


`delete_orphan_files`

другая последовательность меняет результат



разная последовательность - разный результат



почему у maintenance engine - Spark , а не Trino

TRINO

- optimize
- expire snapshots
- remove orphan files
- удобно запускать рядом с query workload

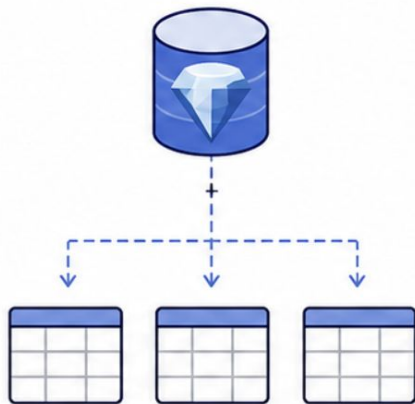
SPARK

- rewrite_data_files с богатой конфигурацией
- partial-progress commits
- rewrite manifests
- специализированные actions / procedures

у команд появились свои iceberg rest-каталоги

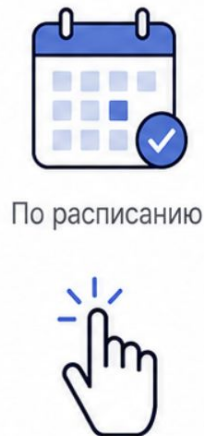
На старте таблиц было немного, затем сделали отдельные каталоги для команд

1. Начало миграции



Немного таблиц

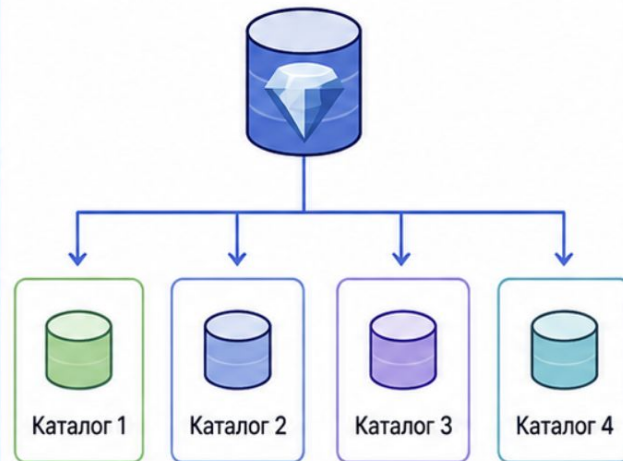
2. Обслуживание



По расписанию

Иногда вручную

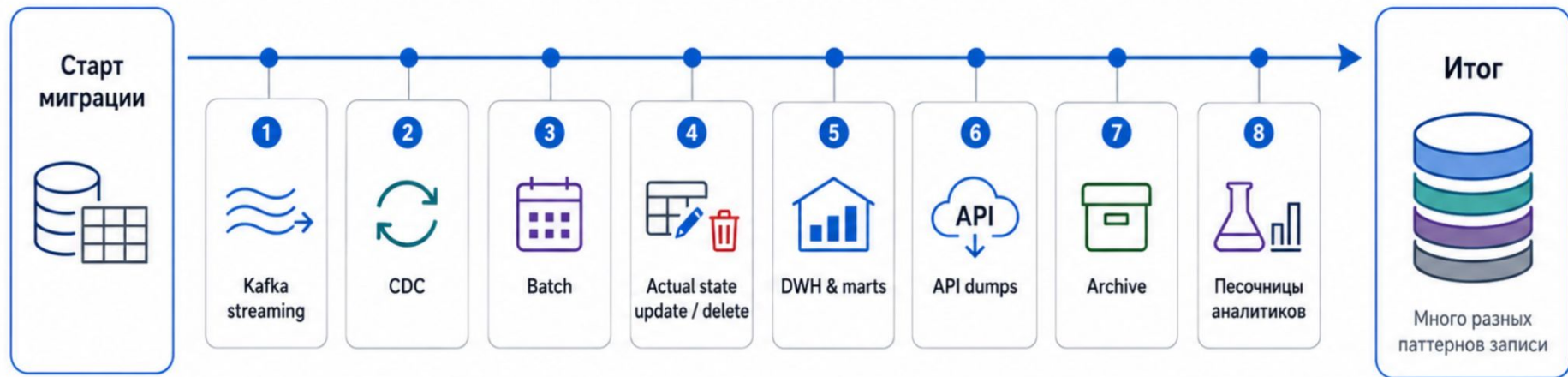
3. Несколько Iceberg Catalogs



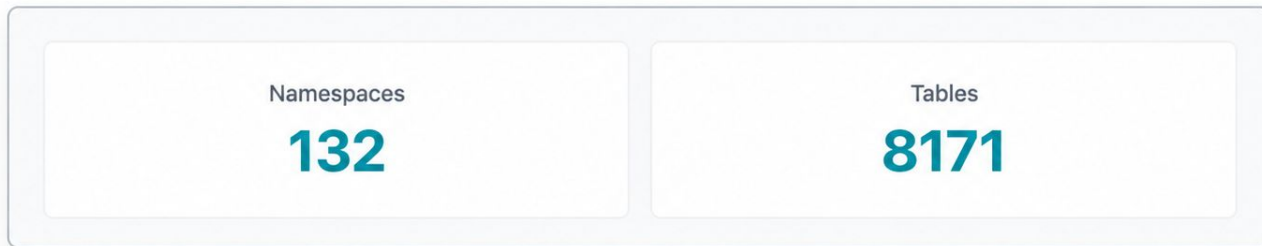
По каталогу на команду / business unit

Во время миграции добавлялись новые виды ETL

Платформа становилась разнообразнее и сложнее



в один момент у нас в Lakehouse стало 8000 таблиц



масштаб, который требует автоматизации обслуживания

деградация растёт быстрее чем мы успеваем с ней разбираться

Total Tables

8675



Tables Needing Optimization

3262



скорость деградации зависит от типа нагрузки

Классический daily batching

Обычный паттерн записи: ежедневный batch



Разные workload — разная скорость деградации

Тип нагрузки	Скорость деградации
Стримы / append-only / логи	● высокая
Частые update / delete	● высокая
DWH факты / справочники	● средняя
Витрины чтения	● низкая

разным таблицам iceberg нужно разное обслуживание

Тип нагрузки определяет, какие операции нужны и как часто



Streaming

append / events

Часто



CDC / updates

update / delete

Часто



DWH / marts

batch / analytics

Периодически



Archive

cold data

Редко

Cron знает время, но не знает состояние таблицы

Для обслуживания важны метрики и сигналы



CRON

- 03:00 каждое воскресенье
- ↓
- Запустить optimize

Проблемы

- слишком рано
- слишком поздно
- одинаково для всех



STATE-DRIVEN



metrics + signals



policy + priority



нужная операция



Запуск только когда
действительно нужно

Нужен feedback-loop

Не просто запускать job, а замыкать цикл



Так родилась идея Iceberg Control Plane

Iceberg Control Plane

Две большие зоны ответственности



Governance



Доступ



Политики



Делегация



Мульти-арендность



Operations



Метрики



Решения



Обслуживание



Наблюдение

Control Plane = Governance + Operations

control plane - управляет обслуживанием , каталог нет



Iceberg REST Catalog

Хранит состояние таблицы



метаданные / текущая версия



координация commit



аутентификация



авторизация



ключи / подпись

API каталога



Iceberg Control Plane

Управляет жизненным циклом



метрики здоровья



рекомендации обслуживания



планирование



исполнение обслуживания



мониторинг результатов

fleet management

собираем метаданные

- Метаданные Iceberg
- Статистика файлов
- История снимков и коммитов
- Метрики использования



Источники: iceberg rest catalog,
системные таблицы trino, логи запросов,
metric reports

откуда собирать метаданные?

trino system view

- table\$snapshots
- table\$files
- table\$partitions
- table\$manifests
- table\$metadata_log_entries

Pyiceberg

```
from pyiceberg.catalog import load_catalog

catalog = load_catalog("rest", uri="https://catalog/...", **creds)
t = catalog.load_table("ns.table")

t.inspect.snapshots()           # $snapshots
t.inspect.files()               # $files (data + delete)
t.inspect.data_files()          # $files where content = 0
t.inspect.delete_files()        # $files where content in (1, 2)
t.inspect.partitions()          # $partitions
t.inspect.manifests()           # $manifests
t.inspect.all_manifests()       # $all_manifests
t.inspect.metadata_log_entries() # $metadata_log_entries
t.inspect.entries()             # $entries (status added/existing/deleted)
t.inspect.history()             # $history
t.inspect.refs()                # $refs
```

считаем базовые метрики по iceberg таблицам

- Гистограмма размеров файлов
- Число файлов
- Число snapshots
- Число manifest files
- Число delete files
- Число orphan files
- Перекрытие Min/Max



Источники: metadata tables, manifests, snapshots

Принимаем решение на основе порогов

- 1 **Счётчики** = число data files, delete files, manifest files
- 2 **Доли (ratio)** = доля delete files, доля мелких файлов
- 3 **Размеры (bytes)** = размеры файлов, delete files, manifest files
- 4 **Время (age)** = возраст snapshot, cooldown партии
- 5 **Распределения** = гистограмма размеров, перекрытие Min/Max

автоматически запускаем код на Spark



- Триггерим Spark-пайплайн задач по Iceberg maintenance
- Запускаем задачи в нужном порядке
- Выполняем их на Spark-кластере

наблюдение за результатами

- Статус выполнения задач
- Изменение метрик и состояния
- События и алерты
- Аудит и отчётность
- Улучшение правил и политик



Хранилище метрик
и событий (PostgreSQL)

что мы нашли в OSS осенью 2025 ?

Iceberg REST Каталоги

**Apache
Polaris**

**Apache
Gravitino**

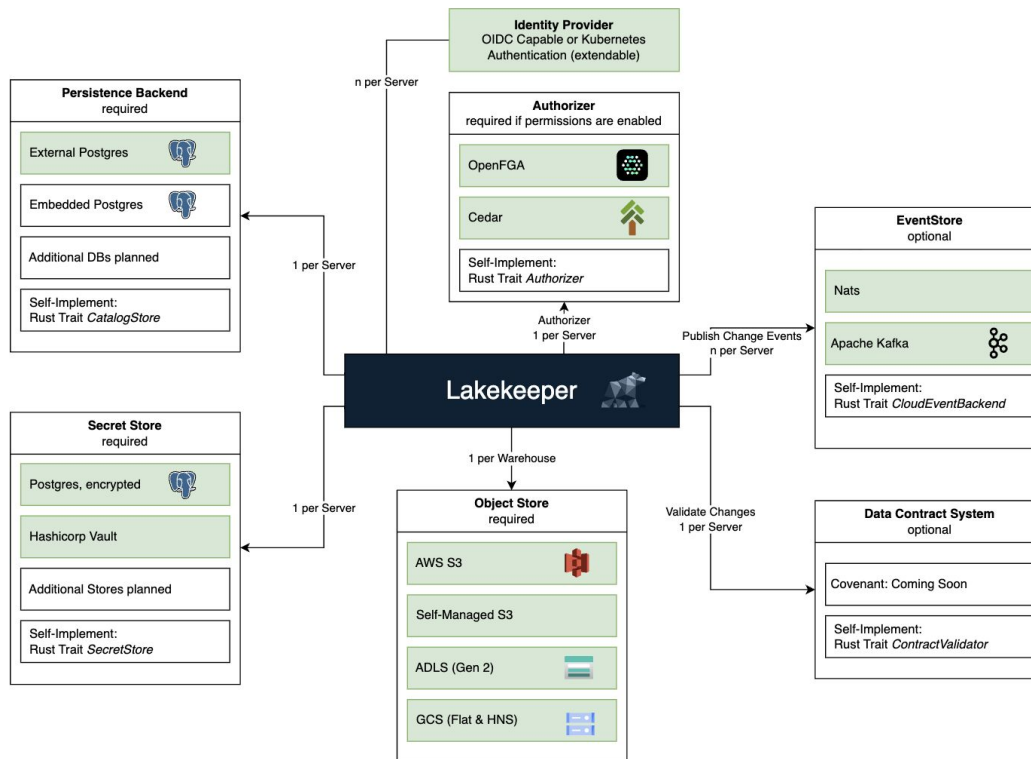
Lakekeeper

Инструменты обслуживания

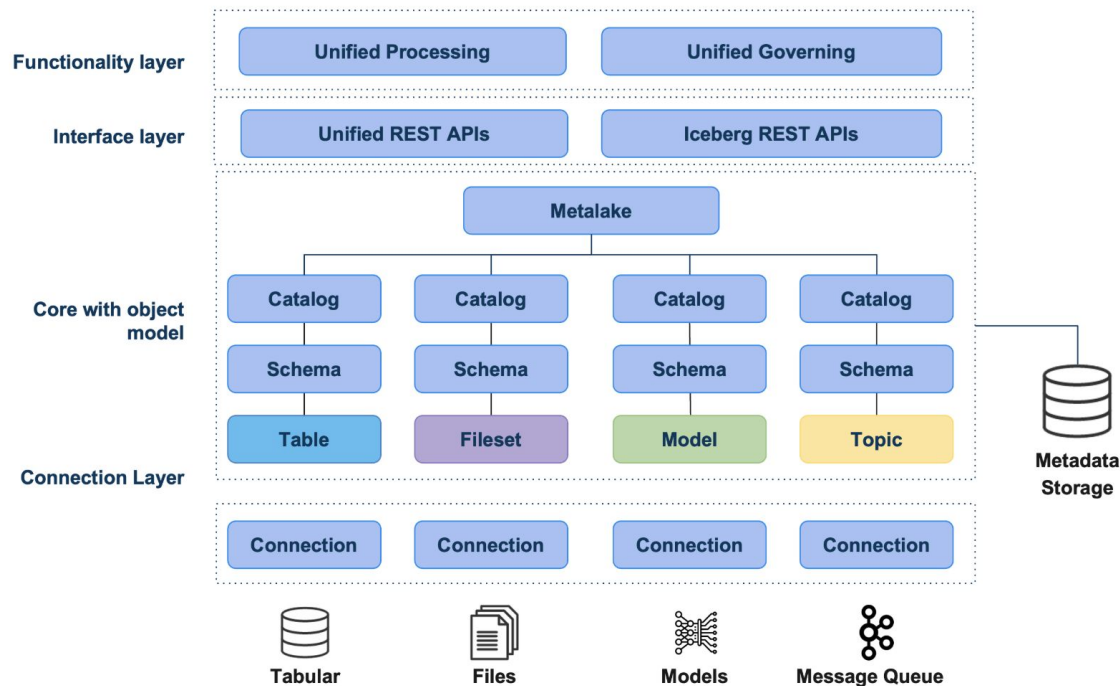
Amoro

Nimtable

Мультиотенантный каталог - Lakekeeper

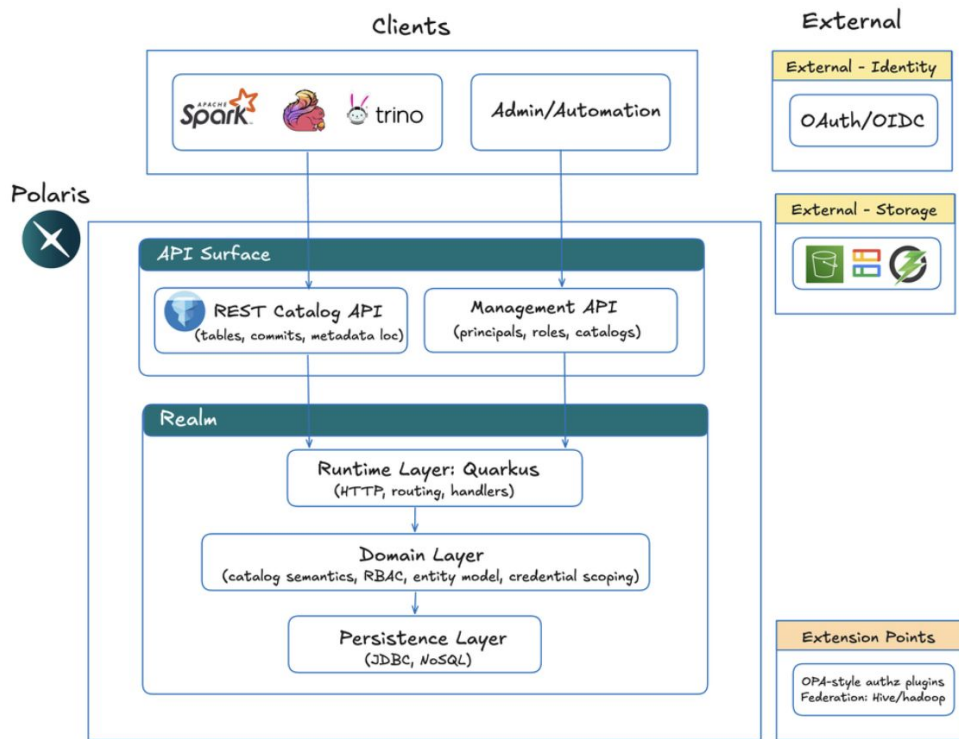


федеративный каталог - Apache Gravitino



<https://gravitino.apache.org/docs/1.0.0/overview>

Apache Polaris



Apache Amoro



Version: 0.8.1-incubating(bc906c0) Commit Time: 2025-08-28T00:55:33+0000 ▾

local.db.user

Optimizing Status: IDLE Records: 7 Table Format: Iceberg(V2) Health Score: 100

Details Files Snapshots Optimizing Operations

Table record count

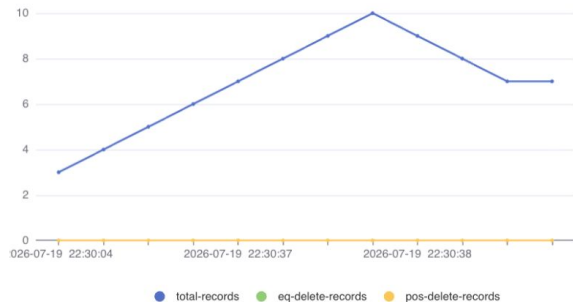


Table file count



main ▾ 1 branches 0 tags Operation: all ▾

	Snapshot ID	Operation	Records	File Count	Commit Time
+	3263132090008493388	replace(optimizing)	7	1	2026-07-19 22:31:14
+	9170911591919795114	delete	7	6	2026-07-19 22:30:38
+	3733300619620080382	delete	8	7	2026-07-19 22:30:38
+	2428054694799386244	delete	9	8	2026-07-19 22:30:38

что умеет Apache Amoro



Возможности

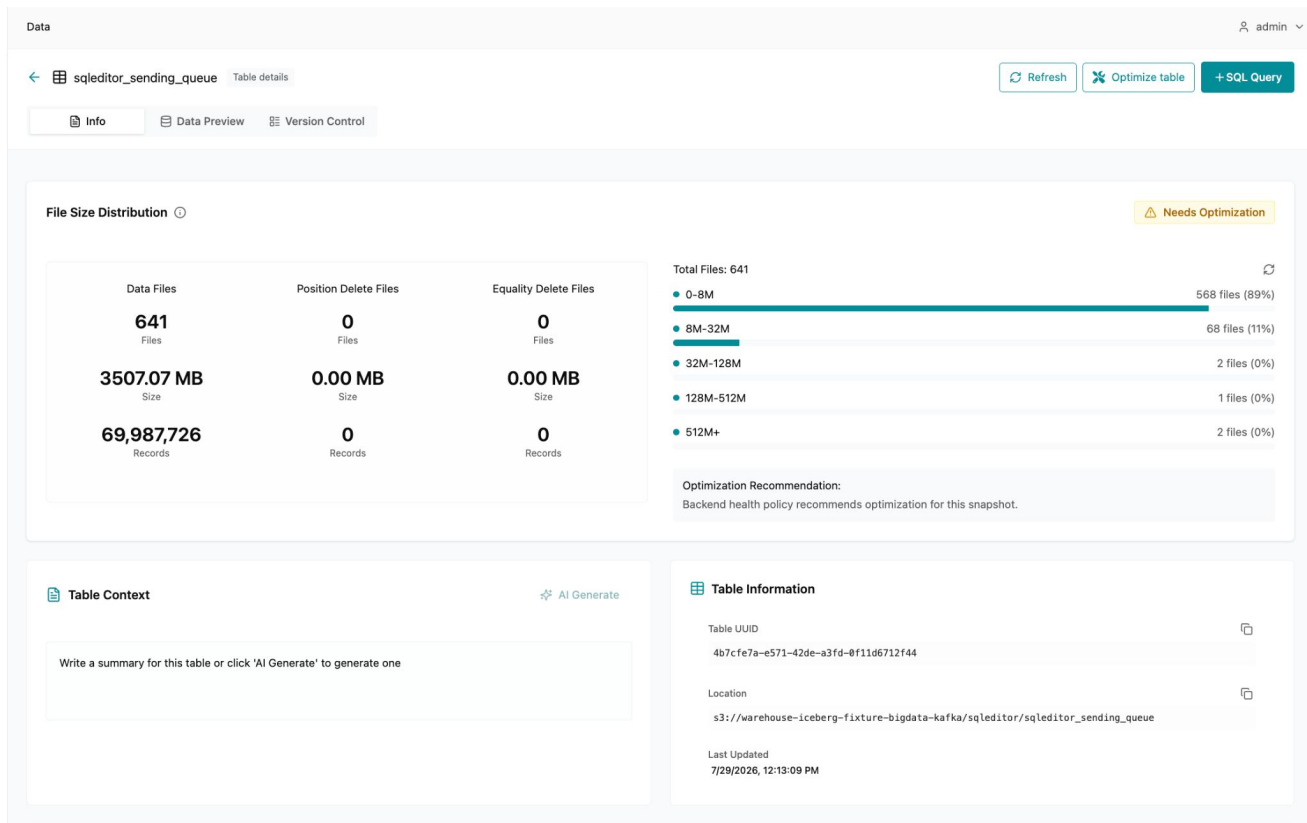
- major / minor / full compaction
- fragment / segment файлы
- per table — optimizer
- настройки на уровне каталога



Ограничения и особенности

- 40/40/20 штрафы
- AMS + собственный Java crawler
- масштабируется потоками, а не воркерами
- только Iceberg v2
- нет поддержки rewrite manifests
- нет настройки политик — система сама принимает решения

Nimtable



<https://github.com/nimtable/nimtable>

что умеет NimTable

- Собирает через REST-каталог
- Не использует собственный клиент
- Есть Table Health и recommendations
- Есть ручной запуск и scheduler jobs
- Широкая статистика
- Выполнение сразу внутри Spark



Выбираем построить свой Control plane!



Практический вывод

- Не ждать монолит
- Выбрать strong catalog foundation
- Строить lifecycle plane

Добавили кеш результатов

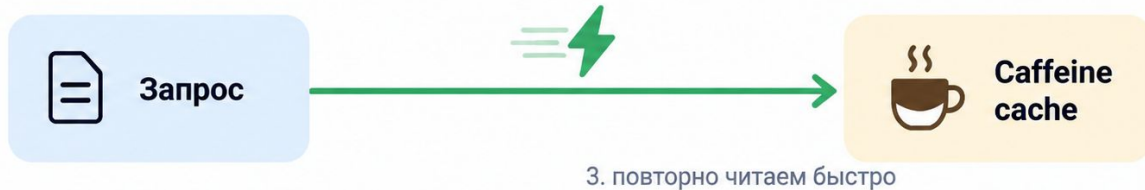
Первый запрос

Результат
вычисляется
и сохраняется



Повторный запрос

Берём результат
из кеша



Повторные запросы работают быстрее

Добавили пакетный сбор метаданных в БД



Перешли на инкрементальный сбор метаданных



не полный перескан

Добавили функцию “авто-кандидат” в селекторы партиций

Scope ⓘ

Auto candidates ▾

Partition candidates are read from Iceberg manifests, filtered by thresholds, freshness safety, and latest-partition safety, then rendered into the worker-specific WHERE clause.

Eligibility thresholds

☒ Use operation thresholds ⓘ

Min data files ⓘ

10

Max average file size MB ⓘ

128

Freshness safety

☐ Exclude recently updated partitions ⓘ

Cooldown minutes ⓘ

5

Latest partition safety

☐ Exclude latest partition ⓘ

Latest partition field ⓘ

Choose field ▾

Partition field discovery requires a table or pipeline context.

Ordering type ⓘ

Auto by partition spec ▾

Run limit

Max partitions per run ⓘ

50

Selector preview
Preview resolves against current table metadata.

 [Preview selector](#)

Run now

опцию запуска - позиционных дилитов отдельно

▶ Run Optimization Job

Cancel

Job type

rewrite-position-delete-files

Trigger

Immediate run

Target file size (MB)

256

Min input files

10

Rewrite all

☐ Rewrite all matching position delete files, not only groups that meet the default rewrite criteria

Scope ?

Whole table

Selector preview

Preview resolves against current table metadata.



Preview selector

Run now

Добавили пайплайны (группы задач)

Single job

Pipelines

New pipeline

bigdata-kafka.abuser.experiment

Cancel

Create pipeline

Pipeline name

daily-maintenance

Trigger

Immediate run

Retry policy

Max attempts

3

Initial delay minutes

5

Backoff multiplier

2

Max delay minutes

360

Steps

+ Add step

Step 1

order 1

^ Up

v Down

Remove

Operation

compact

Target file size MB

128

Strategy

binpack

Sort order

Where clause fallback

Selector mode ⓘ

None

Selector preview

Preview resolves against current table metadata.

Preview selector

добавили пороги для iceberg таблиц

Min data file count

50

Max average data file size (MB)

128

Min total delete file count

disabled

Leave blank to disable this threshold.

запуск пайплайна на основе порогов по метаданным

New pipeline

bigdata-kafka.diadmin.antifraud_fraudulentorder

Cancel

Create pipeline

Pipeline name

daily-maintenance

Trigger

Smart

Min data file count

50

Max average data file size (MB)

128

Min total delete file count

disabled

Leave blank to disable this threshold.

Retry policy

Max attempts

3

Initial delay minutes

5

Backoff multiplier

2

Max delay minutes

360

Steps

+ Add step

Step 1

order 1

Up

Down

Remove

Operation

compact

Target file size MB

128

Strategy

binpack

Sort order

Where clause fallback

Selector mode ⓘ

None

Selector preview

Preview resolves against current table metadata.

Preview selector

расширили набор метаданных

Snapshot Details

Snapshot ID:
786814396408727947

Sequence Number:
154

Date:
Jun 30, 2026, 10:05 PM

Parent ID:
5765543391182508296

Operation:
append

Manifest List Location:
s3://warehouse-iceberg-fixture-bigdata-kafka/diadmin/antifraud_fraudulentorder/metadata/snap-786814396408727947-1-7e5a28c0-d47c-45b3-8056-4ce6bb553739.avro

Summary

added-data-files	1
added-files-size	6924
added-records	2
changed-partition-count	1
iceberg-version	Apache Iceberg 1.1.4 (commit 1435d9acb515d242537c72a7bf43fd8d791da01)
kafka.connect.commit-id	2e0bf73b-1927-471a-99a7-e968d6099a86
kafka.connect.offsets.kafka-connect-diadmin-iceberg-control.connect-diadmin-iceberg-sink	{\"0\":2552032}
kafka.connect.task-id	diadmin-iceberg-sink-0
total-data-files	162
total-delete-files	0
total-equality-deletes	0
total-files-size	1395669
total-position-deletes	0
total-records	21868

🕒 Time Travel to this Snapshot

▼ Manifests

Manifest List

Location: s3://warehouse-iceberg-fixture-bigdata-kafka/diadmin/antifraud_fraudulentorder/metadata/snap-786814396408727947-1-7e5a28c0-d47c-45b3-8056-4ce6bb553739.avro

s3://warehouse-iceberg-fixture-bigdata-kafka/diadmin/antifraud_fraudulentorder/metadata/7e5a28c0-d47c-45b3-8056-4ce6bb553739-#0.avro

s3://warehouse-iceberg-fixture-bigdata-kafka/diadmin/antifraud_fraudulentorder/metadata/246a6b9e-0755-40e9-9cee-0942ffd09761-#0.avro

s3://warehouse-iceberg-fixture-bigdata-kafka/diadmin/antifraud_fraudulentorder/metadata/24e19590-7beb-493d-8345-c4db6abaf423-#0.avro

s3://warehouse-iceberg-fixture-bigdata-kafka/diadmin/antifraud_fraudulentorder/metadata/900a0d00-ff28-498d-a26a-49048819d6c1-#0.avro

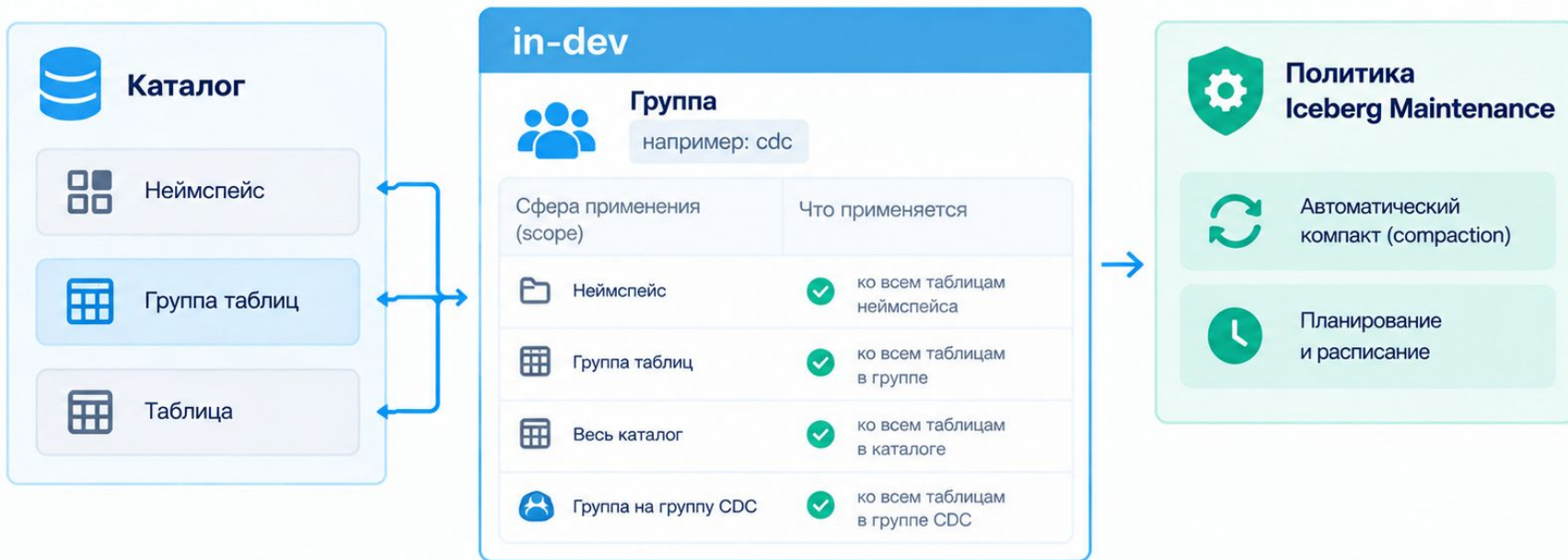
s3://warehouse-iceberg-fixture-bigdata-kafka/diadmin/antifraud_fraudulentorder/metadata/9f7888e0-8e2c-48da-8f08-1e6b4849f722-#0.avro

s3://warehouse-iceberg-fixture-bigdata-kafka/diadmin/antifraud_fraudulentorder/metadata/ac08daec-3327-4913-b11c-b67d767e6556-#0.avro

s3://warehouse-iceberg-fixture-bigdata-kafka/diadmin/antifraud_fraudulentorder/metadata/c78b3e7e-5ff7-421b-b0a6-d5f779e271b8-#0.avro

групповые пайплайны - в разработке

В Iceberg Control Plane можно задать параметр score для политики Iceberg Maintenance на группу таблиц / неймспейс / весь каталог.



приоритеты обслуживания таблиц

На основе порогов и расписаний

избегаем дублей и конфликтов через очередь и приоритеты запуска задач на кластере Spark



как оценить результаты обслуживания iceberg ?

- сохранить scan report от движков
- сохранить логи запросов от движков
- найти признаки для соединения
- соединить данные в таблице
- построить отчет

будущие планы по фичам

- сбор метаданных из событий каталога
- переход на DataFusion как движок для compaction
- аналитика по обслуживанию

а что изменилось за год в OSS в 2026



Iceberg REST Каталоги

Apache
Polaris

Apache
Gravitino

Lakekeeper



Инструменты обслуживания

Amoro

Floe

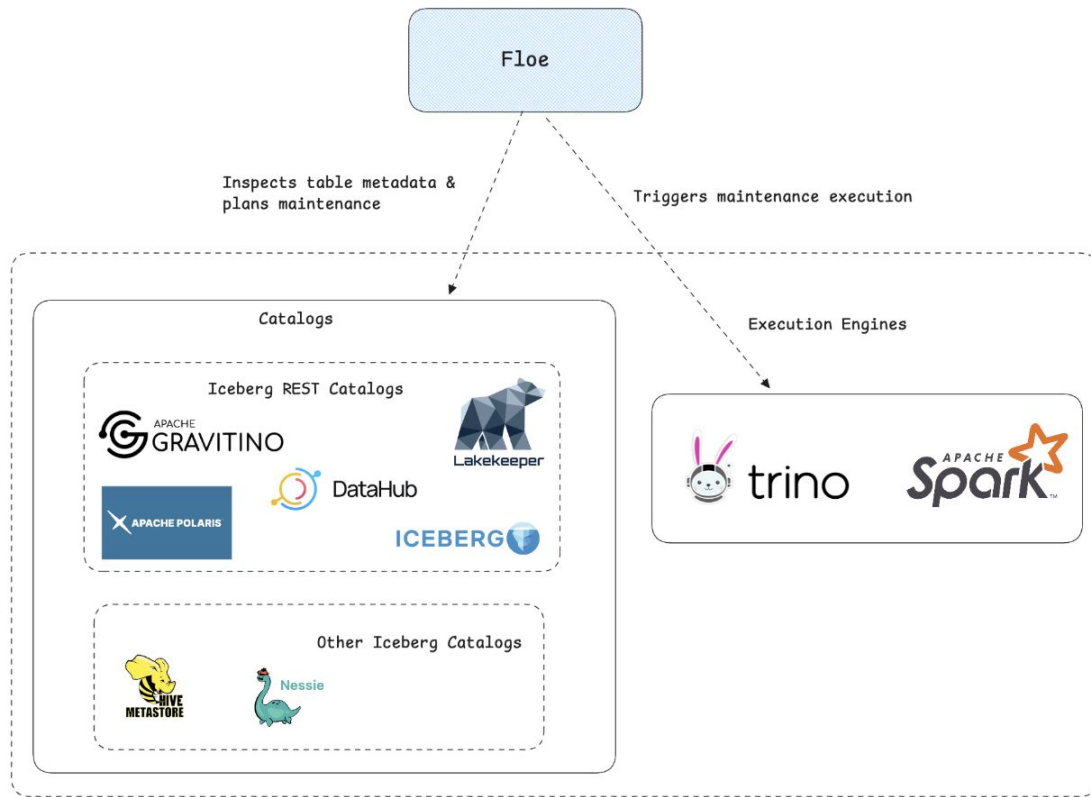
Icekeeper

Nimtable

Apache Amoro

- **новых релизов нет (только 0.81 от 2025 года)**

floe - policy-based обслуживание таблиц iceberg



Что нового в Lakekeeper ?



Безопасность и доступ

- Secure View: DEFINER и INVOKER
- Маскировочные view без доступа к базовой таблице
- ACL: OpenFGA и OPA через Trino bridge
- OIDC
- Trusted Engines



Делегация и маршрутизация

- Vended credentials
- Remote signing
- Чтение с реплик PostgreSQL
- Запись в master PostgreSQL

Gravitino добавили Jobs и TMS

The screenshot displays the Gravitino 1.2.0-SNAPSHOT web interface. The top navigation bar includes links for Catalogs, Jobs, Data Compliance, and Access. The main content area shows a 'Jobs' section with a table of job IDs. A 'Run Job' dialog box is open, allowing users to create a new job by template. The dialog features a search bar for the template name, a list of available templates (currently showing 'builtin-sparkpi'), and buttons to 'Register Job Template' and 'Go to Job Templates'. The selected template details are shown on the left, including its type, comment, executable, class name, arguments, environment variables, and custom fields. The right side of the dialog contains a table for defining custom fields with their names and values.

Run Job

Create a new job by the template.

* Template Name

builtin-sparkpi

builtin-sparkpi

+ Register Job Template

→ Go to Job Templates

Job Type: spark
Comment: Built-in SparkPi job template
Executable: /root/gravitino/auxlib/gravitino-jobs-1.2.0-SNAPSHOT.jar
Class Name: org.apache.gravitino.maintenance.jobs.spark.SparkPiJob

Arguments

{{slices}}

Environment Variables

No environment variables

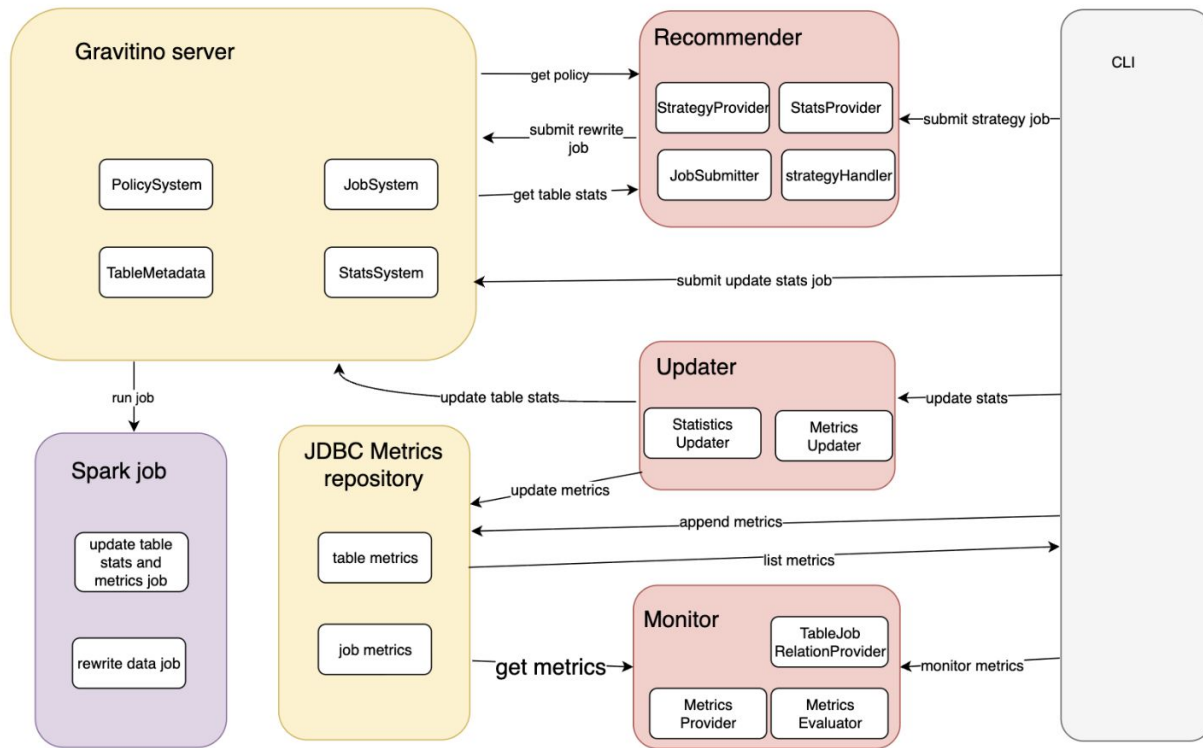
Custom Fields

version: v1

spark_executor_memory	{{spark_executor_memory}}
spark_master	{{spark_master}}
spark_driver_memory	{{spark_driver_memory}}
spark_executor_cores	{{spark_executor_cores}}
spark_executor_instances	{{spark_executor_instance...

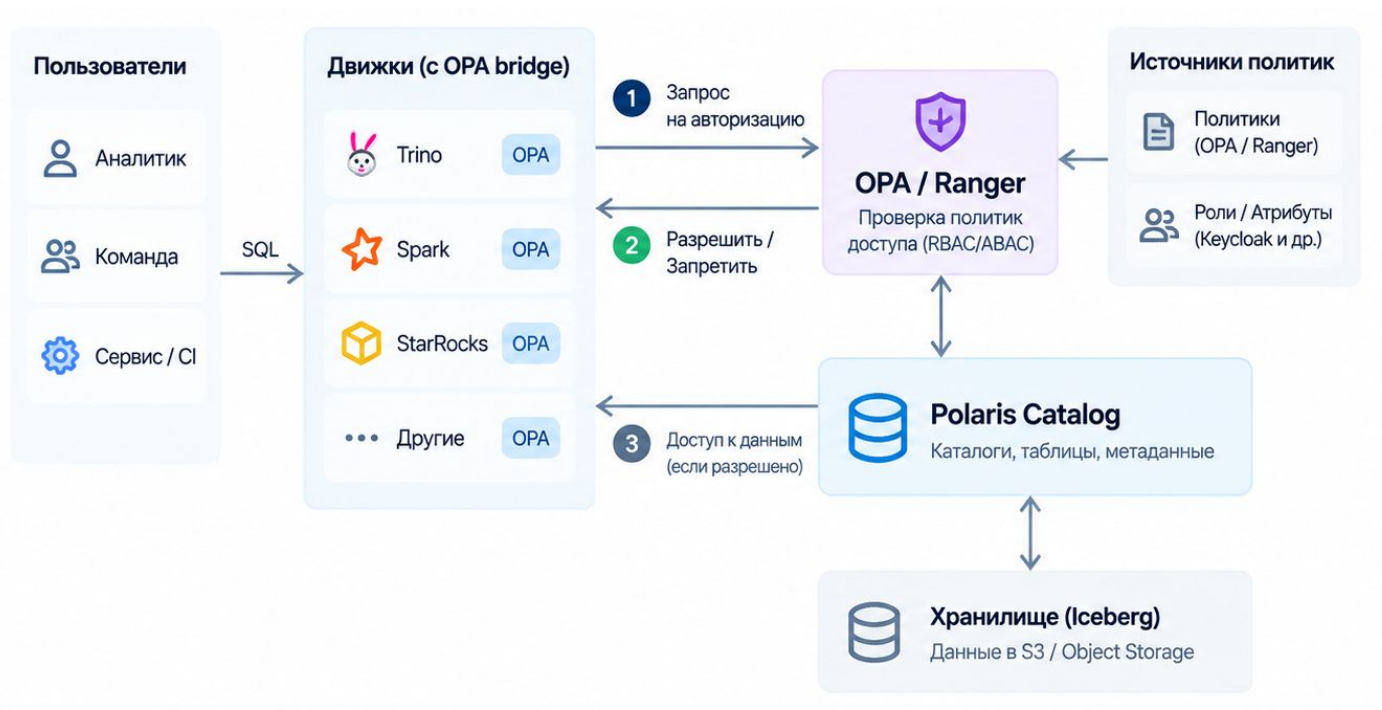
Cancel Submit

Gravitino TMS Архитектура



<https://gravitino.apache.org/docs/1.3.0/table-maintenance-service/extension-guide/>

Что нового в Polaris?



Чего не хватает в Iceberg REST Catalogs?

Сегодня ответ уже не такой, как в 2025



Нам не хватает
не «идеального каталога»,
а **законченного feedback loop**
вокруг fleet таблиц



2025

- Catalog = metadata + security
- Maintenance — отдельно



2026

- Catalogs умеют больше
- policy • events • federation
- Но loop ещё неполный

8

Чего еще не хватает

Как перенести данные на вторую площадку?

Во время миграции мы рассматривали несколько вариантов



Вариант 1: через вычислительный движок

- Spark / Trino / другой движок
- Читать данные на исходной площадке
- Записать данные на второй площадке

Подходит, когда перенос делает вычислительный движок



Вариант 2: нативные функции Iceberg

- register table
- add files
- rewrite table path

Подходит, когда используем встроенные возможности Iceberg

Что сработало для миграции?

Разовый перенос: S3 copy + register table



Как доставлять новые данные во вторую площадку?

Рассматривали несколько вариантов

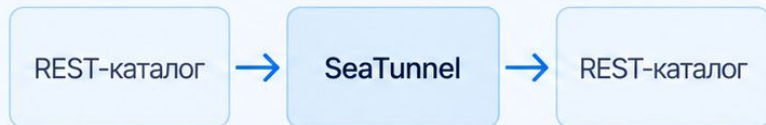
- **Кросс-региональные запросы через движки**
- **Внешние инструменты: SeaTunnel, Data Prepper**
- **Нативный Iceberg**
- **Своя реализация CDC**

Варианты доставки новых данных во вторую площадку

SeaTunnel и OpenSearch Data Prepper

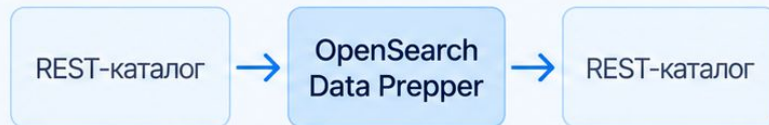
Внешние инструменты для переноса данных

SeaTunnel



Может переносить данные между каталогами

OpenSearch Data Prepper



Отдельный инструмент для доставки данных

Ограничения

- Работают с CoW-таблицами
- Не подходят для MoR-таблиц с delete files
- Delete files мешают корректной репликации изменений

свой CDC - медленно и больно

Проблема CDC в Iceberg: IncrementalChangelogScan

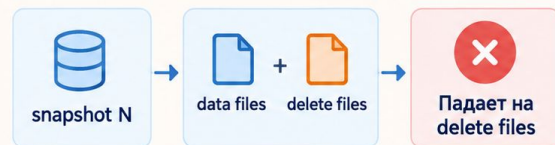
Для CoW работает, для MoR с delete files — нет

CoW



Только новые data files

MoR



Для CDC не подходит

Рабочие решения для постоянного переноса

Два подхода в зависимости от сценария

1

rewrite table path + register table overwrite

Полная перерегистрация таблицы с новым расположением



- ✓ Подходит для миграции или полного переноса
- ✓ Сохраняет историю таблицы
- ✓ Простое и надёжное решение

2

fast append

Только добавление новых данных (append-only)

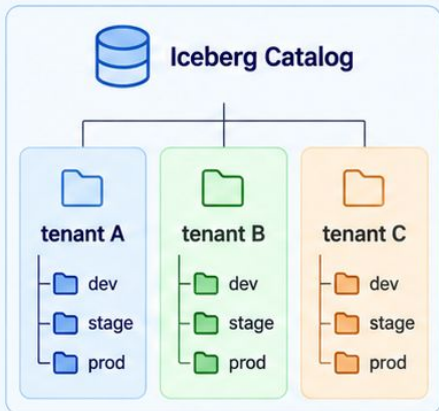


- ✓ Подходит для постоянной репликации и инкрементальной загрузки
- ✓ Быстро и эффективно
- ✓ Не требует перерегистрации таблицы

Как организовать мульти-арендность Iceberg REST Catalog?

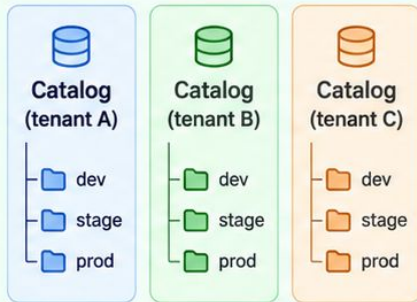
Простые варианты организации tenants, catalogs и environments

1 Namespace per tenant



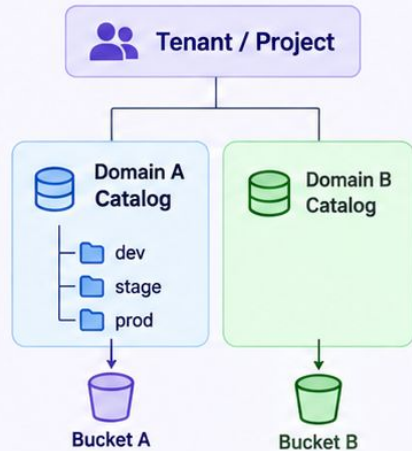
- 1 каталог
- tenant = верхний namespace
- env внутри namespace

2 Catalog per tenant



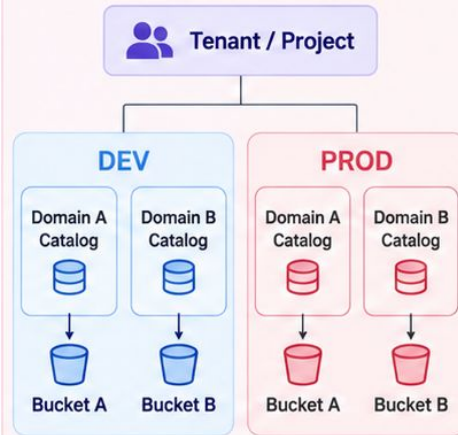
- tenant = свой каталог
- простая изоляция
- удобно для команд

3 Project + domain catalogs



- tenant = проект
- домен = свой каталог + bucket
- env внутри каталога

4 Separate dev / prod



- tenant = проект
- отдельно для dev и prod
- максимум изоляции

Федерация каталогов Iceberg

Один REST-интерфейс для доступа к нескольким каталогам



Что нужно для федерации:

- подключать сторонние каталоги
- давать доступ к ним через один общий интерфейс



Gravitino

Более богатая федерация

- Поддерживает федерацию каталогов
- Metalake — логическое пространство
- Можно подключать разные метасторы: REST, Paimon, Lance, StarRocks, PostgreSQL, Kafka
- Есть pushdown авторизации на подключенный метастор



Polaris

Тоже умеет федерацию

- Поддерживает подключение каталогов
- Дает единый REST-интерфейс
- Подходит для общего доступа к нескольким каталогам



Для федерации нам подходят Gravitino и Polaris, но у Gravitino возможности шире за счет Metalake.

Что мы имеем на финале?

79

Если нужен сильный security

Lakekeeper — лучший выбор. Есть базовый ACL и маскировка через secure view. Но metrics reports нужно собирать отдельно.

80

Если нужна федерация

Gravitino выглядит сильнее остальных. Больше возможностей для federation, чем мы нашли в других каталогах.

81

Если нужна policy-based compaction

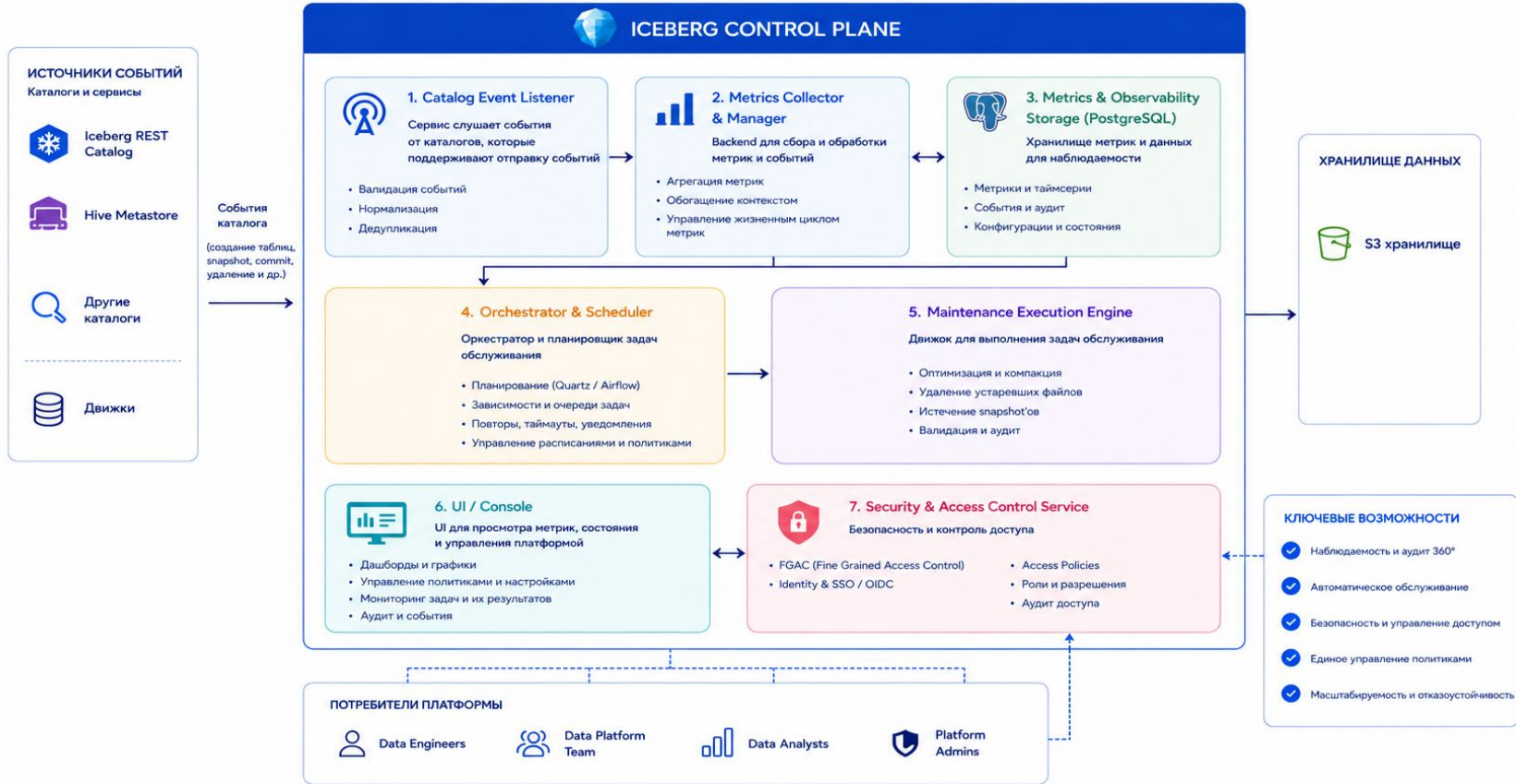
Подойдут Flobе или собственная реализация.

Итог: разные задачи требуют разных компонентов control plane.

Франкенштейн получается следующий:



Iceberg Control Plane Architecture



that all folks



Вопросы

Спасибо!