

Перспективы развития Apache Iceberg

Владимир Озеров
Кверифай Лабс / CedrusData

План

- Функционал
- Состояние сообщества
- Конкуренты

Что получается

Плановая разработка: новые типы данных

Variant Data Type Support Proposal in Iceberg

Note: The original [document](#) with comments is linked but was created using a corporate account. Sharing was recently changed to be allowed for access for internal users by default, and external access requires individual approval.

Motivation

Background

Variant data types allow for the efficient binary encoding of dynamic semi-structured data such as JSON, Avro, Protobuf, etc. By encoding semi-structured data as a variant column, we retain the flexibility of the source data, while allowing query engines to more efficiently operate on the data. Snowflake¹ has supported the Variant data type on Snowflake tables for many years. As more and more users utilize Iceberg tables in Snowflake, we're hearing an increasing chorus of requests for Variant support. Additionally, other query engines such as Apache Spark have begun adding Variant support² and RedShift has a similar SUPER type³. As such, we believe it would be beneficial to the Iceberg community as a whole to standardize the Variant data type encoding used across Iceberg tables.

Without Variant type, the users store the semi-structured data in String type and are required to parse the data in its format (such as JSON) inefficiently. With the support of Variant type, such data can be encoded in an efficient binary representation internally for better performance. Below, we mainly evaluate the existing encodings (Spark Variant Encoding, Bson) and suggest Spark Variant Encoding as the Variant Encoding in Iceberg.

Добавлено в v3:

- TIMESTAMP[_TZ](9)
- GEOMETRY
- GEOGRAPHY
- UNKNOWN
- **VARIANT** – поддержка

полуструктурированных
данных

<https://docs.google.com/document/d/1sq70XDjWJ2DemWyA5dVB80qKzw0CWoM0LOWM7VJVd8/edit?usp=sharing>

Плановая разработка: deletion vectors

```
== PERFORMANCE COMPARISON ==
v2 delete time: 3.126s
v3 delete time: 1.407s
v3 Delete performance improvement: 55.0%

== READ PERFORMANCE COMPARISON ==
v2 full table read time: 0.769s
v3 full table read time: 0.550s
v2 filtered read time: 0.664s
v3 filtered read time: 0.511s
v3 Read performance improvement: 28.5%
v3 Filtered read performance improvement: 23.0%

== DELETE FILE COMPARISON ==
v2 delete format: PARQUET
v2 delete size: 1801 bytes
v3 delete format: PUFFIN
v3 delete size: 475 bytes
v3 size reduction: 73.6%
```

https://docs.google.com/document/d/1FPl0TUzMrPAFWX_C9NL6m6O1uNSxlpDsR-7xpPL0/edit?usp=sharing
<https://aws.amazon.com/blogs/big-data/unlock-the-power-of-apache-iceberg-v3-deletion-vectors-on-amazon-emr/>
<https://github.com/RoaringBitmap/RoaringBitmap>

Задача: ускорение создания и применения positional deletes:

- **Было:** отдельный файл Parquet
- **Стало:** roaring bitmap в Puffin файле

Статус: реализовано, обязательно к использованию в v3

Проблема: кто и когда это интегрирует в экосистему?

Плановая разработки: row lineage

Row Lineage Proposal

Authors: Russell Spitzer, Nileema Shingte , attila-peter.toth@snowflake.com

Pull Request : <https://github.com/apache/iceberg/pull/11130>

Motivation:

Tracking changes for rows in a table as they are modified is key for a variety of applications like CDC streams and Materialized View maintenance. The current spec makes it very difficult to determine the evolution of a single row over time because we have no way of identifying when the row was added to the table, and we are unable to determine if a row was recently added or if it was just changed. To better support these use cases, we would like to add a marker to every row in an Iceberg table to indicate its origin.

Example Use Cases

- Tracking row modifications over time (CDC)
- Incremental Data Processing
- Audit Logs

<https://docs.google.com/document/d/146YuAnU17prnlhyuvbCtClVSavyd5N7hKryvRaFDTE/edit?usp=sharing>

Задача: CDC и все вытекающие

Статус: интегрировано в V3,
обязательно к использованию

Проблема: кто и когда это
интегрирует в экосистему?

Плановая разработка: wide tables

Péter Váry - Monday, May 26, 2025 4:28:47 PM GMT+3

Wide tables in V4

Hi Team,

In machine learning use-cases, it's common to encounter tables with a very high number of columns - sometimes even in the range of several thousand. I've seen cases with up to 15,000 columns. Storing such wide tables in a single Parquet file is often suboptimal, as Parquet can become a bottleneck, even when only a subset of columns is queried.

A common approach to mitigate this is to split the data across multiple Parquet files. With the upcoming File Format API, we could introduce a layer that combines these files into a single iterator, enabling efficient reading of wide and very wide tables.

<https://lists.apache.org/thread/h0941sdq9jwrb6sj0pifijxov8tx7ov9>
<https://github.com/apache/iceberg/pull/13306>

Задача: эффективная работа с очень широкими таблицами, которые могут появляться в ML/AI задачах

Статус: обсуждение, возможный прицел на v4 или v5

Плановая разработка: ускорение работы с S3

Analytics Accelerator Library for Amazon S3 and Iceberg

Authors: michstub@amazon.co.uk/fbbasik@amazon.co.uk/ahmarsu@amazon.co.uk

Iceberg Mail Thread: <https://lists.apache.org/thread/b01bqzxnd22sosb4r2cw513nbrvyywk1>

AAL Github: <https://github.com/awslabs/analytics-accelerator-s3>

Motivation

The purpose of this document is to get community feedback on the Analytics Accelerator Library for S3 being the default stream for S3 in Iceberg going forward. To facilitate this we are providing information on the project's goals and details.

Analytics Accelerator Library (AAL) is an open source library for your client applications that accelerates data access to Amazon S3, lowering processing times and compute costs for data intensive workloads. It does this by providing an implementation of the best practices for accessing data in S3. Currently the integration code is merged in both Hadoop S3A and Iceberg S3FileIO but is behind a feature flag that is defaulted to Off. We would like to propose that AAL is the default Stream for Iceberg.

Our testing shows that it accelerates workloads, and customers such as [Voodoo.io](https://www.voodoo.io) report 10% improvement with S3FileIO, so we want users to benefit from this by default. By implementing best practices for S3 in one centralized location, Iceberg will continue to benefit as we make further improvements. Making this the default data path for Iceberg will enable future optimizations to build upon the higher performance baseline we have established.

More technical details will be part of a follow-up document based on the outcome of this proposal.

AWS Analytics Accelerator Library (AAL) – Java-библиотека с набором оптимизаций для работы с S3:

- Кэш футеров Parquet
- Параллельное чтение
- Coalescing
- И т.д.

Статус: интегрировано в Iceberg, Netflix обещал поделиться результатами бенчмарков

Что не получается

Продвинутые фичи: view interoperability

Ajantha Bhat - Thursday, October 17, 2024 3:17:57 PM GMT+3

[Discuss] Iceberg View Interoperability

Hi everyone,

It's been over six months since Iceberg views were introduced (in version 1.5.0), and while we've seen some benefits—such as versioning and cross-engine recognition—there's still a critical gap in terms of true interoperability. Although views created by one engine are recognized by another, they currently cannot be queried across engines as one engine cannot understand the SQL dialect of another engine.

<https://lists.apache.org/thread/k6szpr5smvrh37sy563xpgjor4g6pr81>

Задача: возможность создавать виртуальные view, которые могут быть использованы различными движками

Статус: обсуждение с октября 2024 ...

Продвинутые фичи: materialized views

Iceberg Materialized View Spec

Background and Motivation

A materialized view is a database entity that stores a query definition as a logical table. The query is precomputed and the resulting data is served when the materialized view is queried. The cost of query execution is pushed to the precomputation step and is amortized over the query executions.

Goal

A common metadata format for materialized views, enabling materialized views to be created, read and updated by different query engines.

<https://docs.google.com/document/d/1UnhldHhe3Grz8JBnqwXPA6ZZord1xMedY5ukEhZYF-A/edit?usp=sharing>

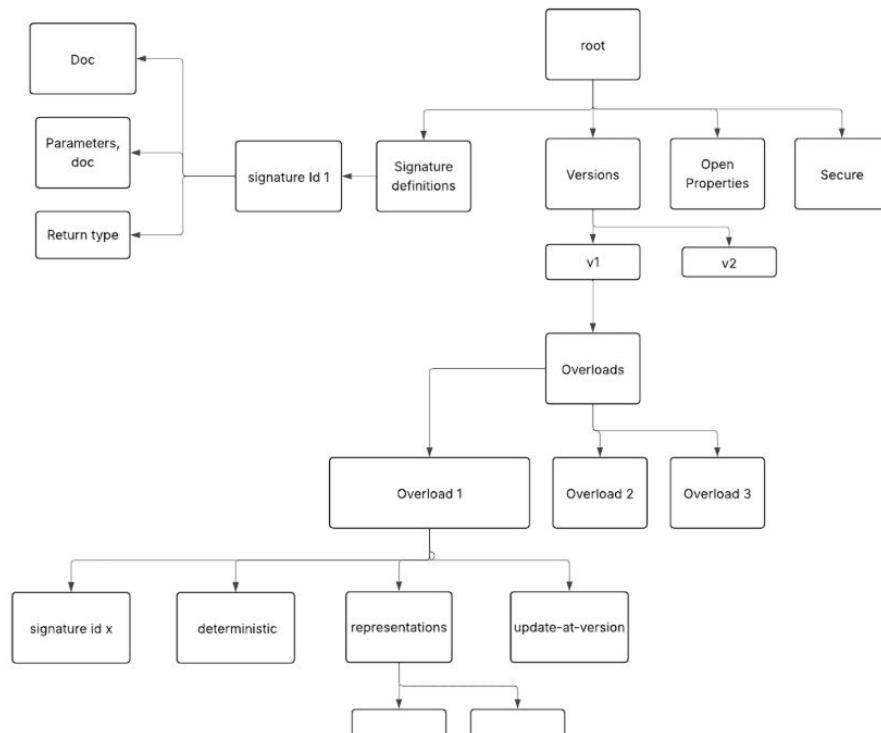
Задача: возможность создавать материализованные представления

Статус: обсуждение с августа 2024 ...

Продвинутые фичи: UDF

UDF metadata structure

Update on Sep 8, 2025



Задача: поддержка user-defined functions, которые могут быть выполнены различными движками

Статус: ???

<https://docs.google.com/document/d/1BDvOfrhH0ZQiQv9elBqeAu8k8VifmeqI9VzliW1F0vc/edit?usp=sharing>

Продвинутые фичи: транзакции

Multi-Table Transactions in Iceberg

Author: Eduard Tudenhoefner (etudenhoefner@gmail.com)

WIP PR: <https://github.com/apache/iceberg/pull/6948>

Motivation

Iceberg currently supports transactions at the table level via the [Transaction API](#), which allows one or more updates to a single table in an atomic manner. However, there is user demand in performing an all-or-nothing operation across multiple tables. This would enable more complex workflows that rely on updating multiple tables within a single transaction.

Задача: транзакции, multi-statement и/или multi-table

Статус: обсуждение с начала времен ...

https://docs.google.com/document/d/1UxXifU8iqP_byaW4E2RuKZx1nobxmAvc5urVcWas1B8/edit?usp=sharing
https://docs.google.com/document/d/1ir4Ah8ceQmo6fwxG_0Ii4vKDlJHUKSch/edit?usp=sharing
https://docs.google.com/document/d/1KVqUJc1WgftHfLz118vMbEE7HV8_pUDk4s-GJFDyAOE/edit?usp=sharing

Признаки взросления

Безопасность: RBAC

Support Securable Objects in Iceberg REST Catalog

Jack Ye (yzhaogin@amazon.com)

Aditya K (krisnad@amazon.com)

Peter Gvozdjak (petergv@amazon.com)

This proposal introduces securable objects with related concepts and APIs to the Iceberg REST catalog (IRC) specification.

Background

There are 2 main reasons for us to look into this area and draft this proposal:

IRC lacks privilege related concepts to express access decisions:

In [a proposal we published previously regarding access decision](#), we would like to express the idea like, for example, a LoadTableResponse tells engine the list of privileges (e.g. read only, read and write, insert but no update, etc.) the caller has on the table, such that **compute engines** can enforce it accordingly.

However, as we explored deeper into this topic, we found that there is no standard in Iceberg to express such privileges on table objects. And when we started to come up with keywords like SELECT, INSERT, DELETE, etc. to express such privileges, we realized that we were basically defining a securable object framework that is well-known in database systems (see [Reference Works](#) section for more details). This is the primary reason that led to us publishing this proposal before we push more progress on the work on access decisions.

<https://docs.google.com/document/d/1KmIDbPuN6lYF0nWs9ostXIB9F4b8iH3zZO0hjqs1Im4/edit?usp=sharing>

Задача: универсальный механизм применения политик доступа для всех потребителей

Статус: обсуждение ...

Безопасность: FGAC (политики)

Iceberg REST Fine Grained Access Control proposal

June 2025

Authors:

Robert Stupp (snazy@apache.org)

Laurent Goujon (laurent@apache.org)

Dmitri Bourlatchkov (dimas@apache.org)

Alex Dutra (adutra@apache.org)

We hereby propose an interoperable interaction between Iceberg REST catalogs and query engines for the Iceberg ecosystem providing row and column level access control and data transformations.

The [interoperability aspect](#) means that different query engines and catalogs shall be able to work seamlessly together.

<https://docs.google.com/document/d/1A5EHXZoluvW7GtEth3GzQz6n5N-fErYltUb6B93Pmw/edit?usp=sharing>

Задача: универсальный механизм применения fine-grained политик доступа:

- column-level security
- column masking
- row filters

Предложение: отправляем политики trusted движку, который их применяет

Статус: обсуждение, прототипирование

Безопасность: FGAC (secure views)

[OSS] Secure Views for dynamic policy enforcement

Authors: Prashant Singh Russell Spitzer Vishwa Lakkundi

Last updated: Jun 24, 2025

Reviewers & Contributors: laurent@dremio.com jb.onofre@dremio.com royhasson@microsoft.com
kevinliu5@microsoft.com

Prev OSS work : [Iceberg Spec Extensions for Data Access Decision Exchange](#)

Motivation

For data warehouse and lakehouse systems, robust access control is vital to maintaining security, privacy, and compliance. In multi-cloud, multi-engine environments, it serves as a key component of comprehensive data governance.

It is already possible to store access management policies in several catalogs and systems such as Glue and Ranger and implement enforcement of such policies in the several engines like Spark, Trino and others. The OSS proposal [here](#) discusses an approach to extend Iceberg spec to enable exchanging Access Decision' rather than merely the access control policies.

<https://docs.google.com/document/d/1Ajicez7xPhzwKXenGZ19h0hnqxrwAq3rSaiDV1v0x-s/edit?usp=sharing>

<https://docs.google.com/document/d/1A5EHXZoluvW7GtEth3GzQz6n5N-fErYLTUbf6B93Pmw/edit?usp=sharing>

Задача: универсальный механизм применения fine-grained политик доступа:

- column-level security
- column masking
- row filters

Предложение: подменяем table на view, в определение которого “вшины” правила маскирования и фильтрации

Статус: обсуждение, прототипирование

Disaster Recovery: относительные пути

Support for Relative Paths

<https://s.apache.org/iceberg-spec-relative-path>

talat@apache.org, dweeks@apache.org

Background

Currently, location fields and paths to metadata files and data files in Iceberg are absolute URIs. This means that when a table is moved or copied to a new storage location (e.g., from HDFS to GCS, to a different S3 bucket, or change in prefix), all of these embedded absolute paths need to be rewritten in the metadata and delete files to reflect the new location.

Key use-cases that require metadata and data relocation include:

- **Replication:** Copying table state and history to another data center or availability zone for high availability or read scaling.
- **Backup:** Archiving table state and history for disaster recovery and compliance purposes.
- **Data Migration:** Moving tables between different storage systems (e.g., HDFS to GCS, on-prem to cloud).

<https://docs.google.com/document/d/1a6tXvbWVbvOxiRexCialsA6NOqcbxPKjeoJGagYtg/edit?usp=sharing>

Задача: возможность безопасного перемещения объектов между директориями и файловыми системами

Статус: активное обсуждение

Disaster Recovery: логическая репликация

Incremental Iceberg Replication Feature Proposal

Xinli Shang, Yufei Gu

Abstract

This proposal introduces native replication capabilities for Apache Iceberg tables as a new library. The feature enables automatic synchronization of both data and metadata across multiple storage locations through configurable table properties. A key use case is automatic cross-data center backup. Replication integrates seamlessly with Iceberg's commit workflow and supports both synchronous and asynchronous modes, all while preserving ACID guarantees.

The implementation extends the commit process with replication logic, automatically copying data files and metadata to configured target locations whenever tables are modified. This provides built-in disaster recovery, data locality optimization, and cross-region analytics capabilities without the need for external tools or custom solutions.

This proposal is complementary to the [\[Public\] Relative Path Spec Definition](#) and the [Table Replication procedure](#), and does not conflict with either.

https://docs.google.com/document/d/1yrVLs0CQylHs9WbBVx_EK6ad419Adsl9xHozpmQEMrs/edit?usp=sharing

Задача: возможность логического копирования содержимого объектов в разные файловые системы

Статус: активное обсуждение, частично реализовано

Качество

Add REST Compatibility Kit #10908

Merged

danielcweeks merged 2 commits into apache:main from danielcweeks:rest-compatibility-

Conversation 62

Commits 2

Checks 48

Files changed 11



danielcweeks commented on Aug 10, 2024

Member

...

This PR adds the ability to run the Catalog Table and View tests against a REST Catalog implementation. By default, it will run the tests against a local server using the catalog adaptor to proxy to a JDBC backend catalog.

Catalog configurations can be overridden to point to a different catalog implementation or service running externally.



<https://github.com/apache/iceberg/pull/10908>

Задача: набор тестов для проверки корректности реализации REST каталогов

Статус: реализовано

Роль REST каталога

REST: Read planning

Iceberg REST Scan API Support

Authors: Rahil Chertara (rchertar@amazon.com), Jack Ye (yzhaoqin@amazon.com)

WIP PR: <https://github.com/apache/iceberg/pull/9252>

In Iceberg, when a user submits a query to an engine, the engine will invoke one of Iceberg's catalog implementations (such as AWS Glue, HiveMetaStore, REST, etc.) to load the table's metadata. From the table metadata, iceberg can obtain the table schema, and from the engine Iceberg can obtain the selected columns and filters, to initialize an iceberg table scan.

Once an Iceberg scan has been created, it will initiate a planning phase. During the planning, an iceberg scan will read the table's snapshot to find the relevant data at client side. A table's snapshot contains a list of manifest files with each manifest file containing pointers to the underlying data files in local or cloud storage. Once Iceberg has filtered through the data and delete files that are relevant for this query it will start converting those files into file scan tasks. The file scan tasks are then returned back to the engine in order for the engine to execute the tasks distributedly.

In this doc, we propose moving this planning logic to server side, by introducing scan APIs to Iceberg REST catalog spec.

<https://docs.google.com/document/d/1FdjCnFZM1fNtgyb9-v9fU4FwOX4An-poEwSaJe8RqUg/edit?usp=sharing>

Задача: делегировать планирование скана таблицы REST-каталогу

Мотивация:

- Повышение производительности
- Облегчение интеграции с языками программирования
- Безопасность
- Catalog federation

Статус: реализовано ... но какой движок это делает?

REST: Write planning

Iceberg UpdateTable Fine-Grained Metadata Commit Support

Author: Drew Gallardo (dru@amazon.com)

WIP PR: <https://github.com/apache/iceberg/pull/11287>

Motivation

Currently, the process of committing data to an Iceberg table involves six main operations: [AppendFiles](#), [OverwriteFiles](#), [DeleteFiles](#), [RewriteFiles](#), [RowDelta](#), and [ReplacePartitions](#). These operations manage changes by reflecting new file additions or removals in the table metadata through snapshots, which capture the table's latest state.

Clients ingest data by writing files (e.g. Parquet), which are wrapped into `DataFile` or `DeleteFile` objects before invoking the data operations. These operations are responsible for making the manifest, manifest list and snapshot, before the catalog trying to commit the new table metadata as the latest one.

In this proposal, we aim to enhance the [UpdateTable REST API](#) by introducing new table update actions enabling fine-grained metadata commits. This change allows clients to submit requests for file-level changes, while shifting the responsibility for generating the manifest, manifest list and snapshot, and committing these metadata from the client to the catalog service. By transferring these operations to the service, we simplify the client-side process and give the catalog finer control over data modifications.

<https://docs.google.com/document/d/1n-cEE4-vFreTLnUTPgo7U8ih44MFo1ZHjyk4NCxxalc/edit?usp=sharing>

Задача: делегировать создание метаданных REST-каталогу

Мотивация:

- Более эффективное разрешение конфликтов
- Облегчение интеграции с языками программирования
- Безопасность

Статус: обсуждение, прототипирование

Конкурентная среда

Iceberg vs Delta

ICEBERG 

 DELTA LAKE

Iceberg vs Delta

ICEBERG 

 DELTA LAKE

Iceberg 1.10.0 (11 Сен 2025)

+66 НОВЫХ КОНТРИБЮТОРОВ

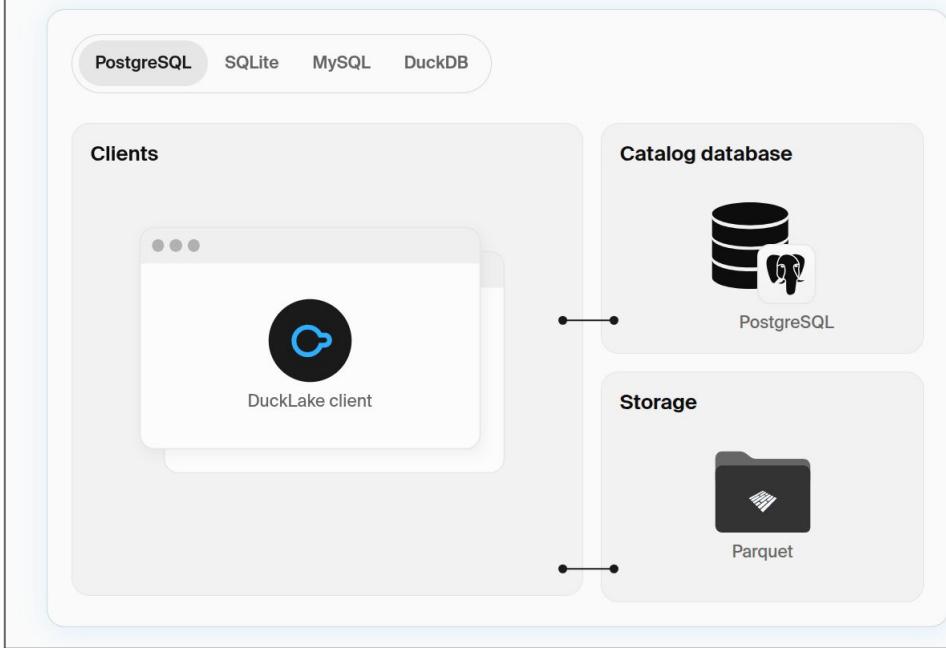
New Contributors

- [@ccmao1130](#) made their first contribution in [#12860](#)
- [@szarnyasyg](#) made their first contribution in [#12932](#)
- [@mrsubhash](#) made their first contribution in [#12894](#)
- [@akhilputhiry](#) made their first contribution in [#12406](#)
- [@gyfora](#) made their first contribution in [#12991](#)
- [@sundy-li](#) made their first contribution in [#13002](#)
- [@huaxiangsun](#) made their first contribution in [#12771](#)
- [@coderfender](#) made their first contribution in [#12901](#)
- [@hariuserx](#) made their first contribution in [#12886](#)
- [@juldrixx](#) made their first contribution in [#13041](#)
- [@hsingh574](#) made their first contribution in [#12855](#)
- [@JeonDaehong](#) made their first contribution in [#13021](#)
- [@futurepastori](#) made their first contribution in [#13128](#)
- [@sfc-gh-bchannel](#) made their first contribution in [#12975](#)
- [@Bhargavkonidena](#) made their first contribution in [#13113](#)
- [@plusplusjajia](#) made their first contribution in [#13118](#)
- [@nika-qubit](#) made their first contribution in [#13192](#)

Iceberg vs Ducklake

DuckLake's architecture

With DuckLake, all you need to run your own data warehouse is a catalog database and storage for Parquet files.



Iceberg vs Ducklake

DuckLake in Operation

Is DuckLake production-ready?

While we tested DuckLake extensively, it is not yet production-ready as demonstrated by its version number 0.3. We expect DuckLake to mature over the course of 2025.

How is authentication implemented in DuckLake?

DuckLake piggybacks on the authentication of the metadata catalog database. For example, if your catalog database is PostgreSQL, you can use PostgreSQL's [authentication ↗](#) and [authorization ↗](#) [methods ↗](#) to protect your DuckLake. This is particularly effective when enabling encryption of DuckLake files.



Iceberg vs Ducklake

Tables

DuckLake 0.3 uses 22 tables to store metadata and to stage data fragments for data inlining. Below we describe all those tables and their semantics.

The following figure shows the most important 11 tables defined by the DuckLake schema:



- **Как развивать?**
- **Как поддерживать?**
- **Где сообщество?**

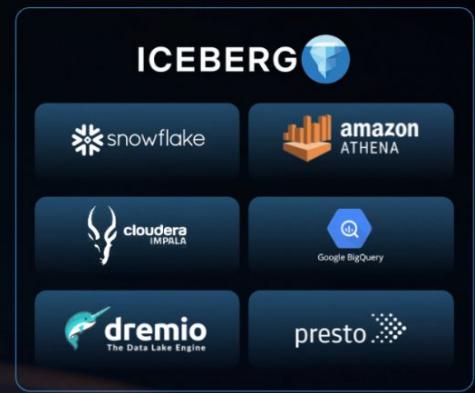
Iceberg vs Paimon



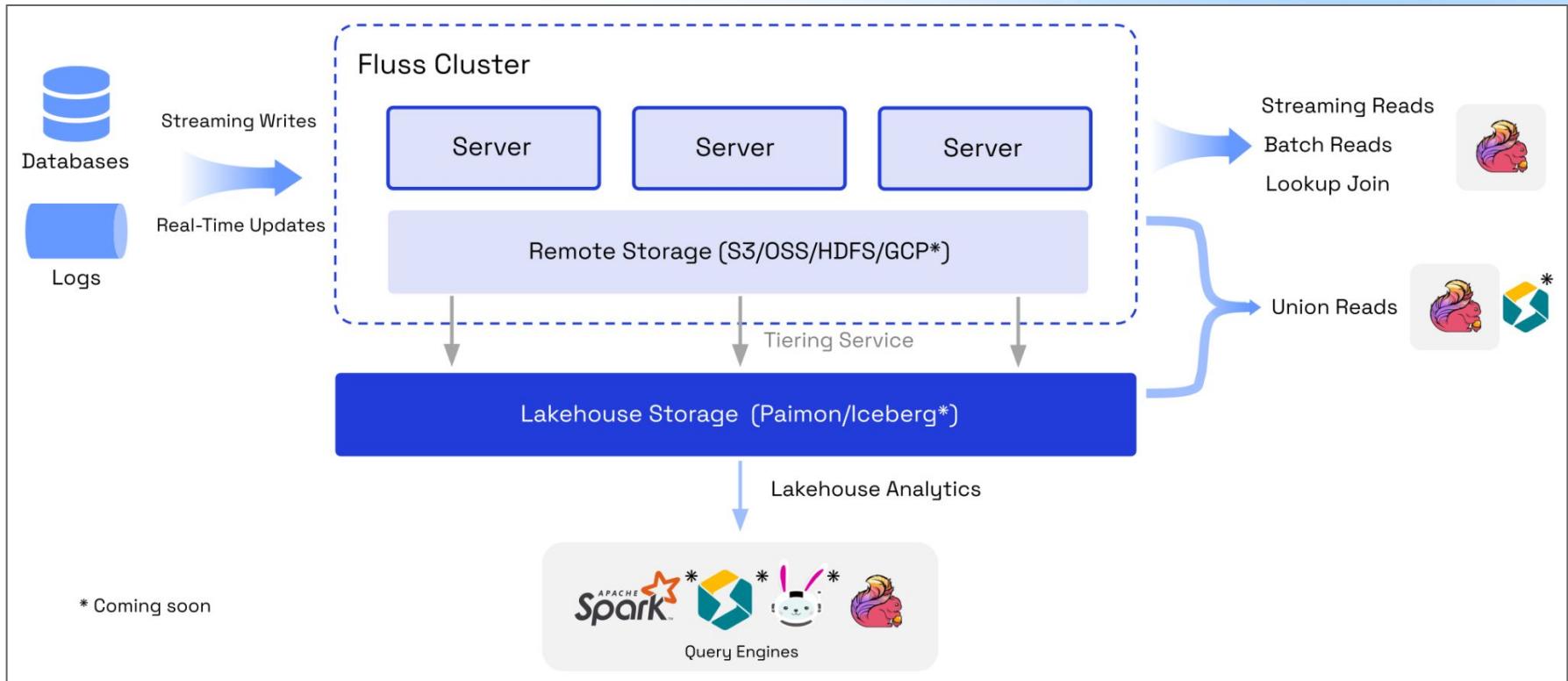
Apache Paimon Ecosystem with Iceberg Snapshots



Iceberg Snapshots



Iceberg vs Paimon



<https://fluss.apache.org/>

Amazon S3 Tables

Store tabular data at scale in S3

Amazon S3 Tables deliver the first cloud object store with built-in Apache Iceberg support and streamline storing tabular data at scale. Continual table optimization automatically scans and rewrites table data in the background, achieving up to 3x faster query performance compared to unmanaged Iceberg tables. These performance optimizations will continue to improve over time. Additionally, S3 Tables include optimizations specific to Iceberg workloads that deliver up to 10x higher transactions per second compared to Iceberg tables stored in general purpose S3 buckets. For more details on S3 Tables' query performance improvements, refer to the [blog post](#).

With S3 Tables support for the Apache Iceberg standard, your tabular data can be easily queried with popular AWS and third-party query engines. Use S3 Tables to store tabular data such as daily purchase transactions, streaming sensor data, or ad impressions as an Iceberg table in S3, and optimize performance and cost as your data evolves using automatic table maintenance. Read the [blog post](#) to learn more.



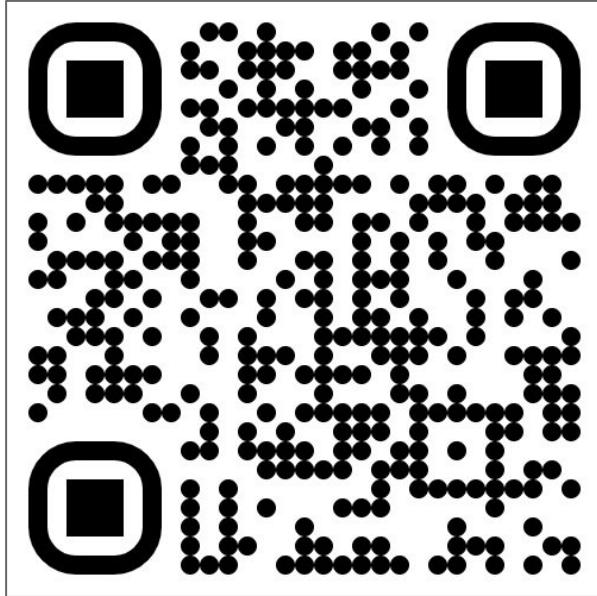
Итого

Позитив:

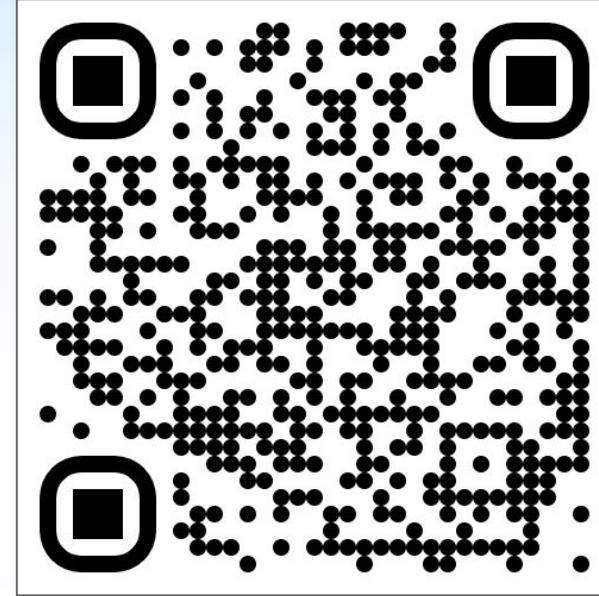
- Уверенный лидер среди табличных форматов
- Активное независимое сообщество
- Быстрое наращивание функционала
- Централизация management-сервисов внутри REST catalog

Негатив:

- (Очень) долгая проработка важных фич
- Распределение ролей между компонентами не до конца понятно
- Фрагментация аналитического ландшафта
- Не адресует некоторые важные сценарии (например, real-time аналитика)



Trino Community
t.me/cedrusdatachat



Iceberg Community
t.me/iceberg_lakehouse_chat