September Project Antalya Roundup: Fresh Features to Run ClickHouse® Faster and Cheaper on Data Lakes

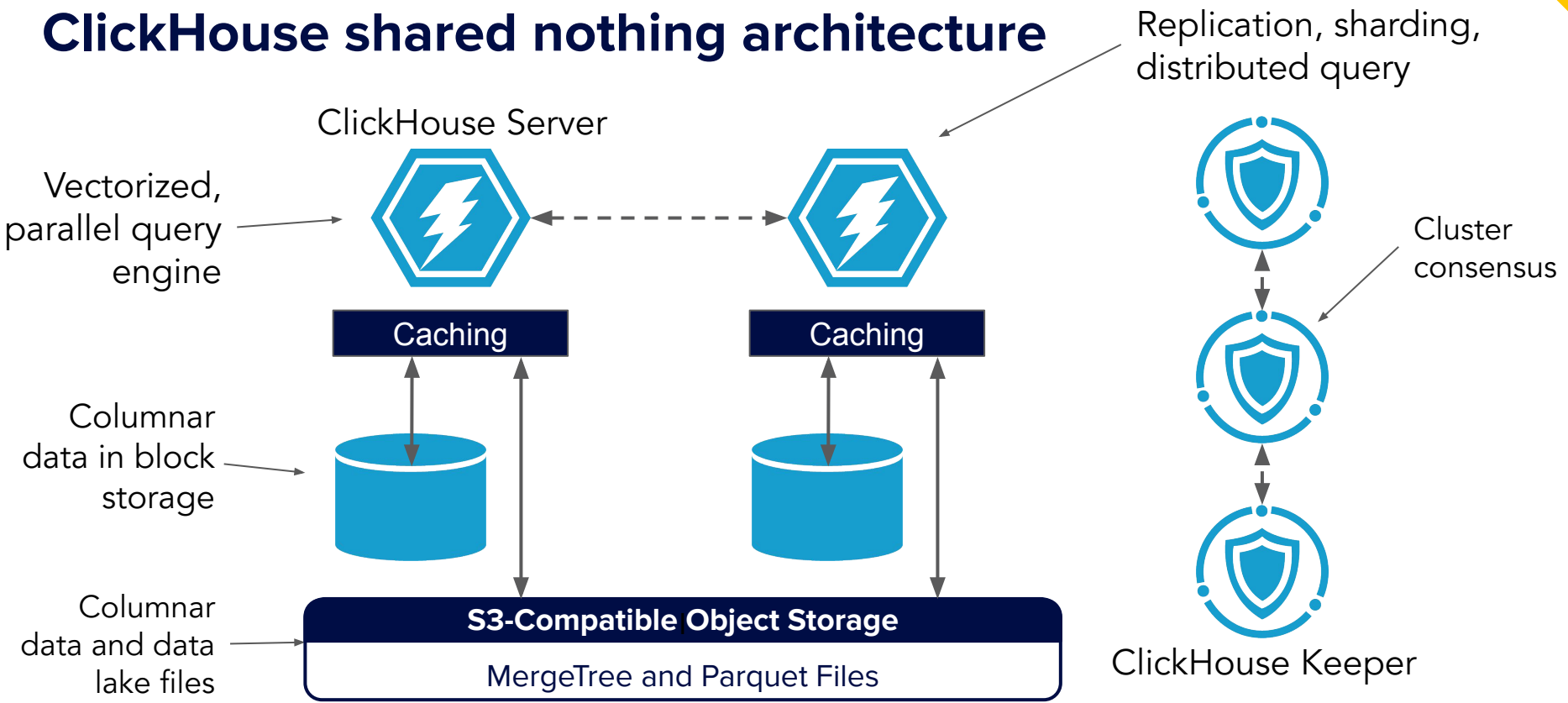
Alexander Zaitsev - Altinity CTO
Robert Hodges - Altinity CEO

10 September 2025

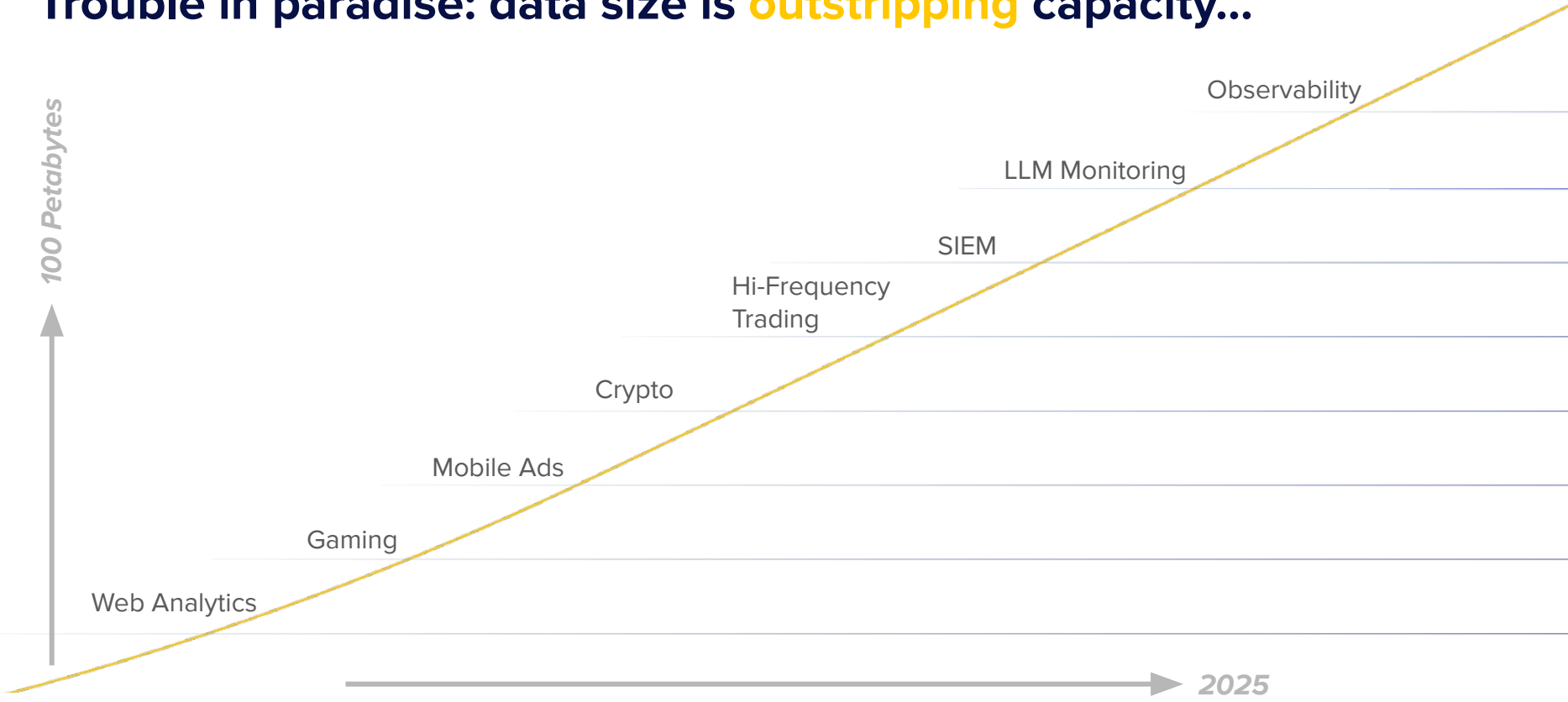
Run Open Source ClickHouse® Better

Altinity.Cloud Enterprise Support

ClickHouse shared nothing architecture

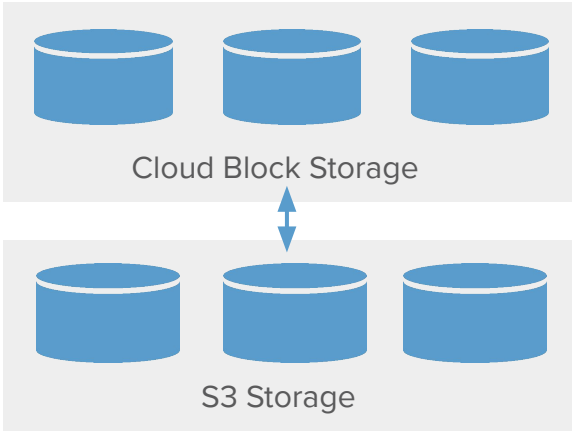


Trouble in paradise: data size is outstripping capacity...

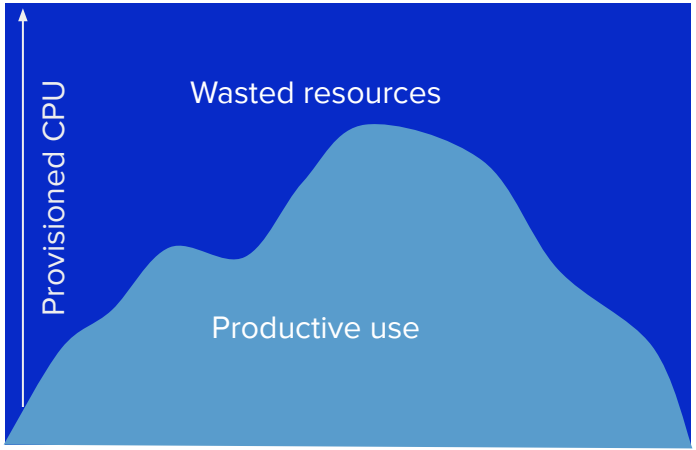


...Leading to pressure on storage and compute cost

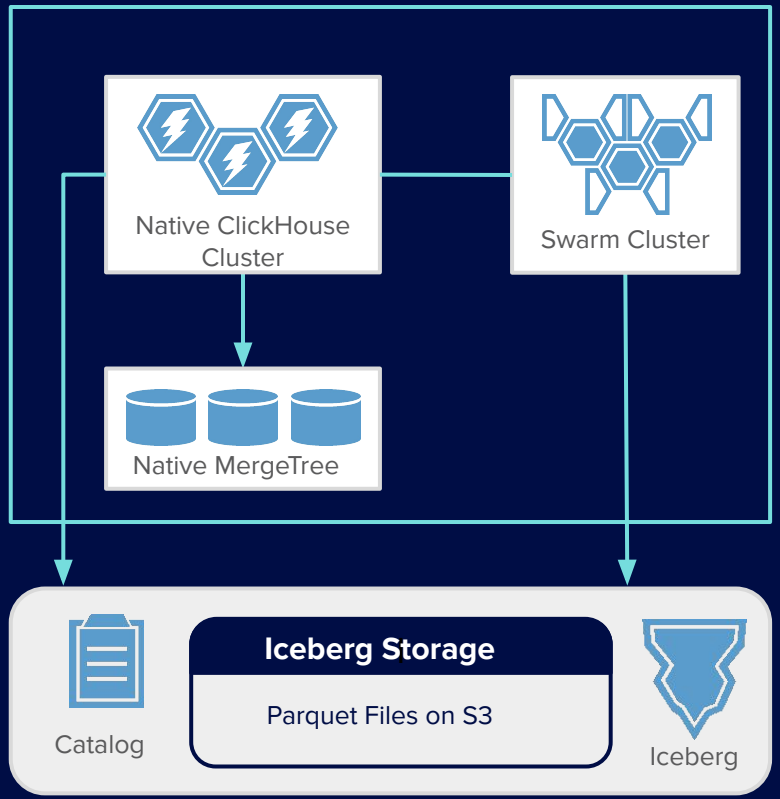
Block storage with replication is 10x more expensive



Overprovisioning wastes compute



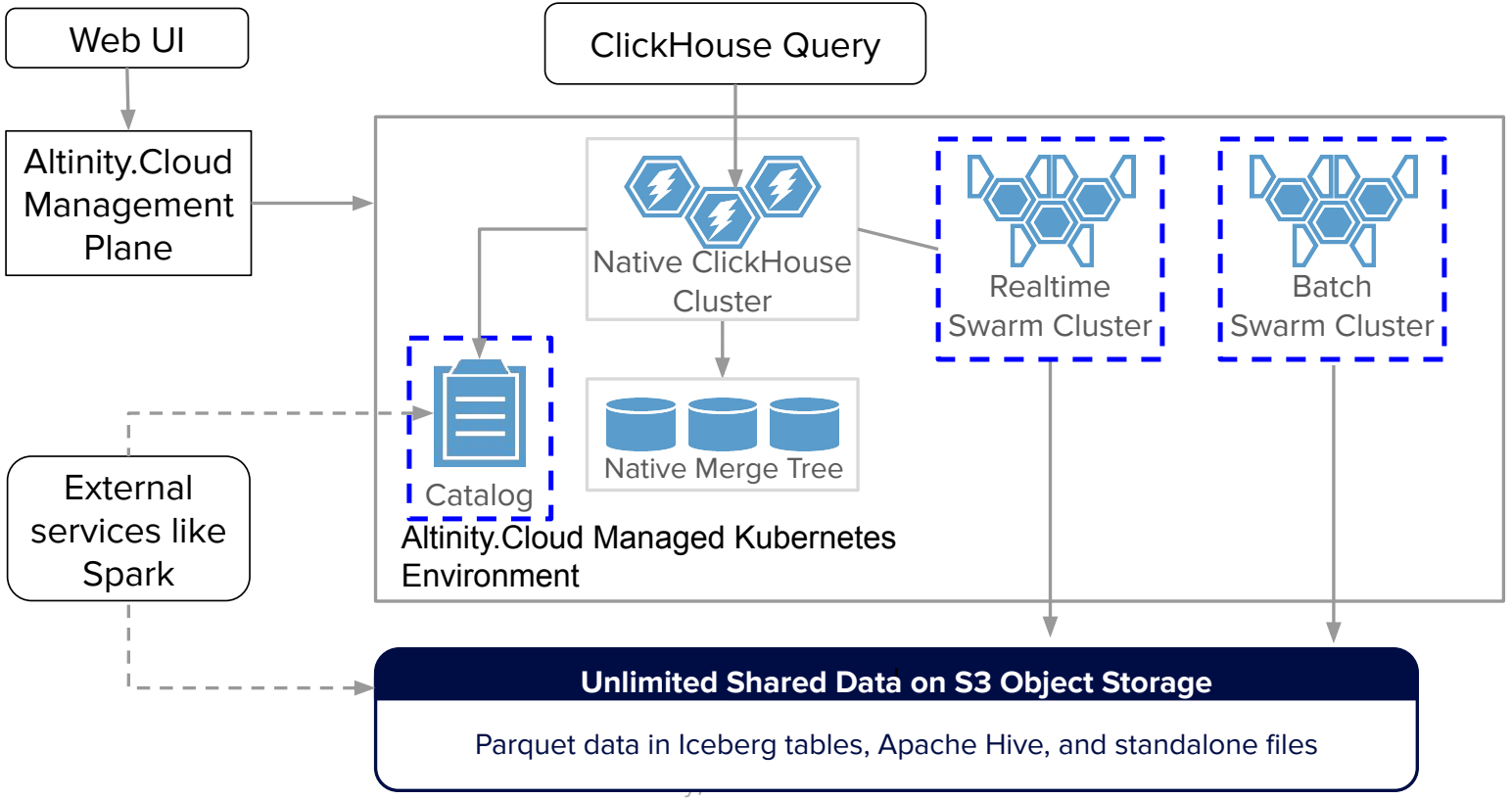
Not to mention: manageability and stability



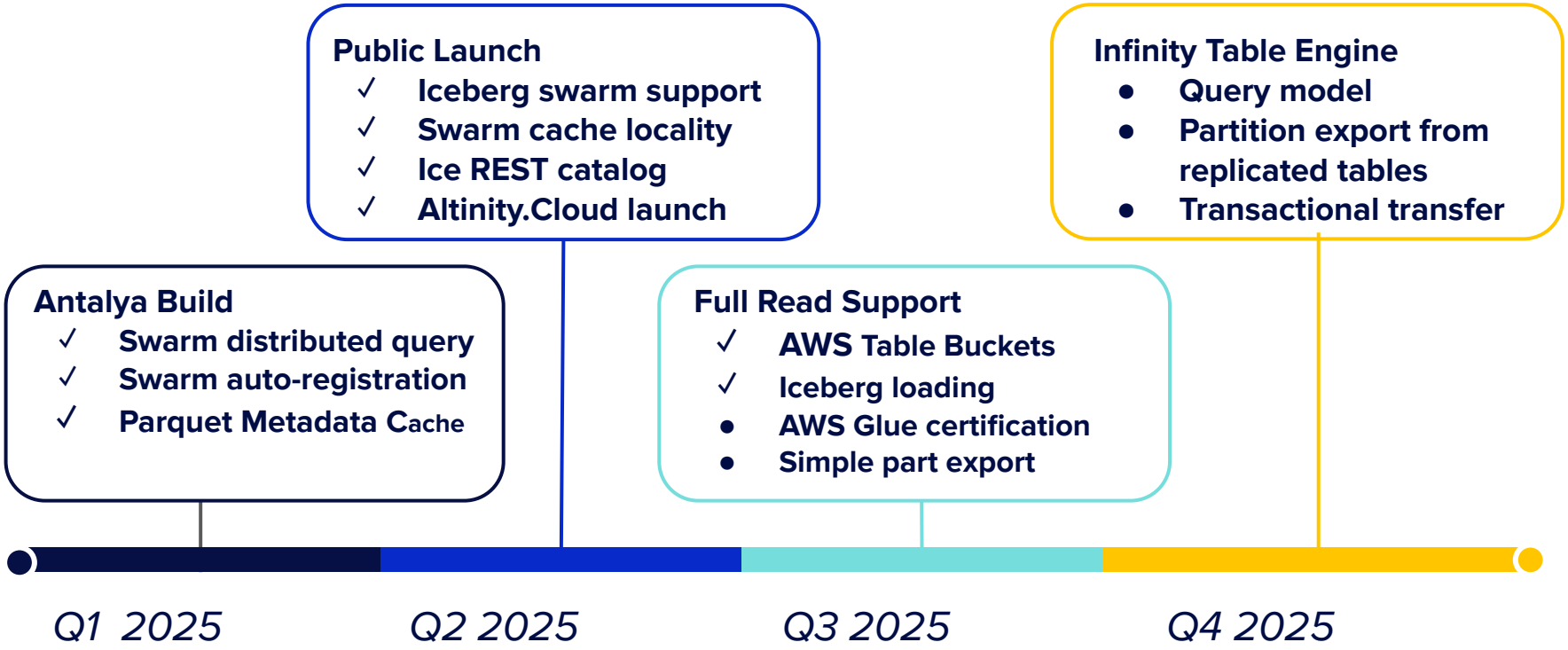
Project Antalya makes ClickHouse run fast and cheap on shared data

- Extends native ClickHouse capabilities
- Adds Iceberg for shared storage
- Adds swarm clusters for scalable compute
- 100% open source

Altinity.Cloud can deploy the Antalya stack anywhere



Antalya Project Plan

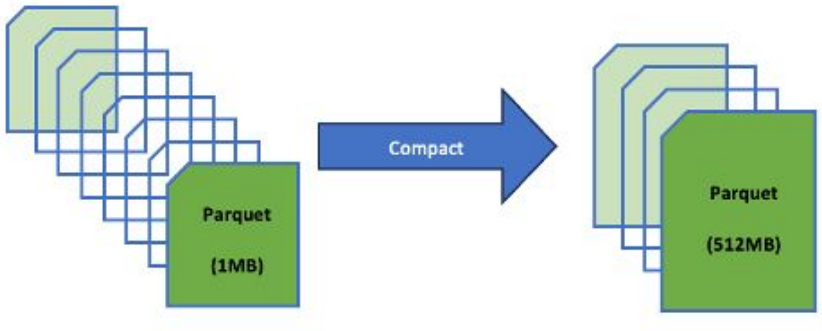


Highlighted New Features

- AWS S3 Table buckets
- Ice-rest-catalog management
- Writing to Iceberg from ClickHouse

S3 Table Buckets

- Introduced at the end of 2024
- Uses S3 as a storage
- Parquet is a primary format
- Accessible via Iceberg rest and Glue APIs
- Build-in catalog management features (e.g. compaction)



Connecting from ClickHouse – Problem

ClickHouse connection to Iceberg – does not work for S3 Tables:

```
CREATE DATABASE iceberg
ENGINE =
DataLakeCatalog('https://s3tables.<Region>.amazonaws.com/iceberg')
SETTINGS
catalog_type = 'rest',
auth_header = '[HIDDEN]',
warehouse =
'arn:aws:s3tables:<Region>:<accountID>:bucket/<bucketname>'
```

See https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-tables-integrating-open-source.html

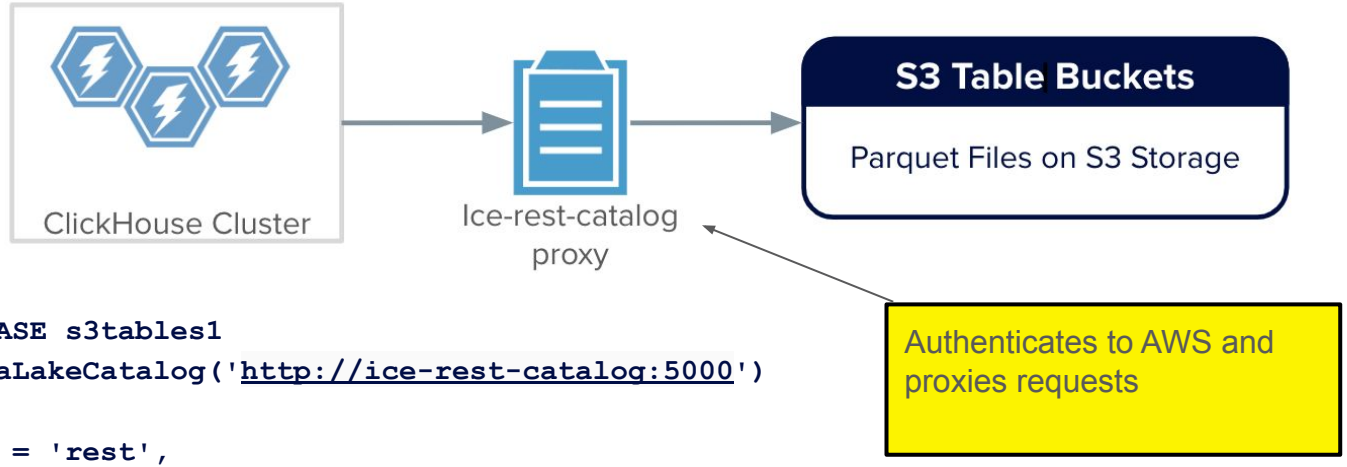
Pylceberg connection to S3 buckets:

```
rest_catalog = load_catalog(
  Catalog_name,
  **{
    "type": "rest",

"warehouse":"arn:aws:s3tables:<Region>:<accountID>:bucket/<bucketname>",
    "uri":
"https://s3tables.<Region>.amazonaws.com/iceberg",
    "rest.sigv4-enabled": "true",
    "rest.signing-name": "s3tables",
    "rest.signing-region": "<Region>"
  }
)
```

Currently not supported in ClickHouse

Connecting from ClickHouse – ice-rest-catalog




```
CREATE DATABASE s3tables1
ENGINE = DataLakeCatalog('http://ice-rest-catalog:5000')
SETTINGS
catalog_type = 'rest',
auth_header = '[HIDDEN]',
warehouse = 'arn:aws:s3tables:<Region>:<accountID>:bucket/<bucketname>'
```

See https://github.com/Altinity/ice/tree/master/examples/s3tables

Connecting from ClickHouse – upstream vs Antalya

Upstream ClickHouse does not work:

```
2025.06.19 05:01:38.947571 [ 799 ] {85d354ee-ba11-4ac1-89e7-3d8684a7e449} <Error>
RestCatalog(s3://ice-rest-catalog:5000): Code: 48. DB::Exception: Unexpected location format:
s3://3b8aac22-bb5f-4548-y5ta9hipodcohbq9o6pwbbiht7dcgusw2b--table-s3. (NOT_IMPLEMENTED)
```

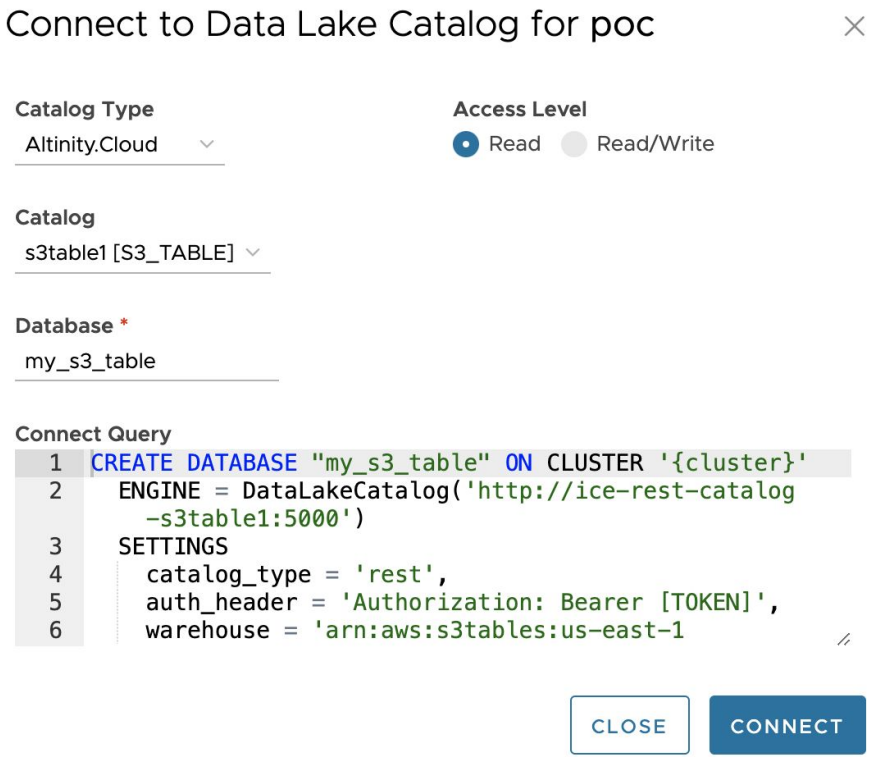
Antalya builds allow file locations outside of the warehouse location:

```
SELECT _path, _file FROM s3tables1."btc.transactions" LIMIT 1 FORMAT Vertical

Row 1:
──────
_path:
5397fe31-0492-4c62-1tggx1u3jtyuca9y8tg45ym7qunkquse1b--table-s3/data/1755084192112-a5d77700bb4a6b0a0f6c571c25
9c1a0086b938b04f82a8a823b772467e344a60.parquet
_file: 1755084192112-a5d77700bb4a6b0a0f6c571c259c1a0086b938b04f82a8a823b772467e344a60.parquet
```

Location is pretty much random for S3 table buckets

Connecting from ClickHouse – Altinity.Cloud



Catalog Management in ice-rest-catalog

- MANIFEST_COMPACTION – merge multiple manifests
- DATA_COMPACTION – merge multiple Parquet files
- SNAPSHOT_CLEANUP – delete old snapshots
- ORPHAN_CLEANUP – delete non-referenced files in data and metadata paths

Automatically, configured in .ice-rest-catalog.yaml – part of the catalog service.

On demand with `ice-rest-catalog perform-maintenance [-dry-run] [type]`

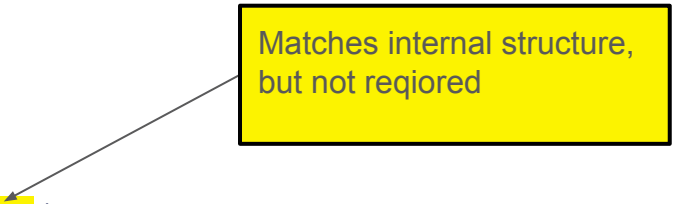
Writing to Iceberg from ClickHouse

Writing to local Iceberg tables (25.7+):
https://github.com/ClickHouse/ClickHouse/pull/82692

- Only metadata files are updated
- No catalog support

Writing to Iceberg from ClickHouse using ice

1. Insert into catalog warehouse location




```
INSERT INTO FUNCTION s3(
's3://$CATALOG_S3_BUCKET_NAME/aws-public-blockchain/btc/data/2025-05-13.parquet')
SELECT *
FROM s3('s3://aws-public-blockchain/v1.0/btc/transactions/date=2025-05-13/*.parquet', NOSIGN)
```

2. Run ice with `–no-copy` flag to create a snapshot

```
ice insert aws-public-blockchain.btc -p --no-copy --skip-duplicates \
s3://$CATALOG_S3_BUCKET_NAME/aws-public-blockchain/btc/data/*.parquet
```

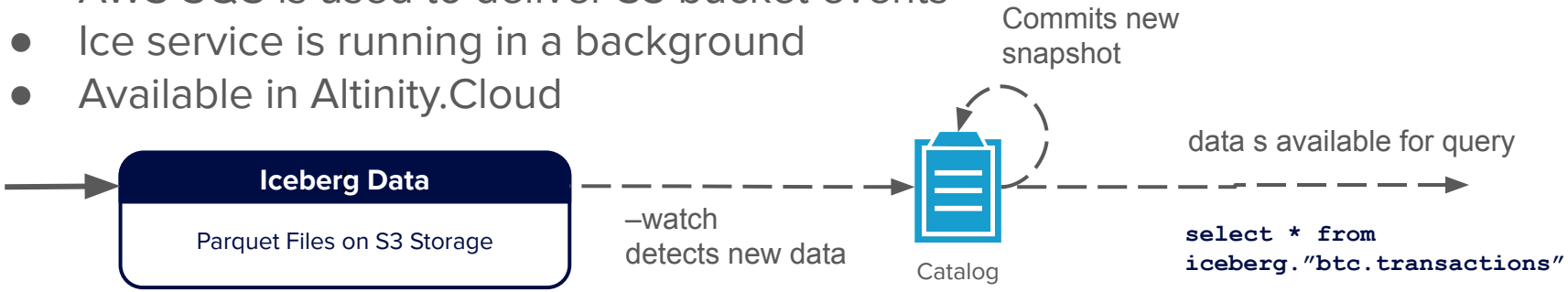
See https://github.com/Altinity/ice/

Writing to Iceberg from ClickHouse using ice –watch

```
INSERT INTO FUNCTION s3(
's3://$CATALOG_S3_BUCKET_NAME/aws-public-blockchain/btc/external-data/2025-05-13.parquet')
SELECT *
FROM s3('s3://aws-public-blockchain/v1.0/btc/transactions/date=2025-05-13/*.parquet', NOSIGN)

ice insert aws-public-blockchain.btc -p --no-copy --skip-duplicates \
s3://$CATALOG_S3_BUCKET_NAME/aws-public-blockchain/btc/external-data/*.parquet \
--watch="$CATALOG_SQS_QUEUE_URL"
```

- AWS SQS is used to deliver S3 bucket events
- Ice service is running in a background
- Available in Altinity.Cloud



See https://github.com/Altinity/ice/tree/master/examples/s3watch

Writing to Iceberg from ClickHouse using ice –watch

```
INSERT INTO FUNCTION s3(
's3://altialya-2fv4arm7-iceberg/aws-public-blockchain/btc/external-data/{_partition_id}.parquet')
PARTITION BY date
SELECT date, * REPLACE (
  toDateTime64(block_timestamp, 6) as block_timestamp,
  toDateTime64(last_modified, 6) as last_modified
)
FROM s3('s3://aws-public-blockchain/v1.0/btc/transactions/date=*/*.parquet', NOSIGN)
WHERE date between '2025-01-01' and '2025-01-31'
SETTINGS output_format_parquet_use_custom_encoder=0
```

Nanoseconds are not supported by Iceberg

Custom encoder may produce incorrect data types

Requires a lot of RAM!

Writing to Iceberg from ClickHouse – WIP

```
ALTER TABLE my_table EXPORT PARTITION|PART to s3(...)
```

- Export from from MergeTree to Parquet
- Preserves ORDER BY (no re-sorting)
- Atomic
- Retriable
- Memory efficient

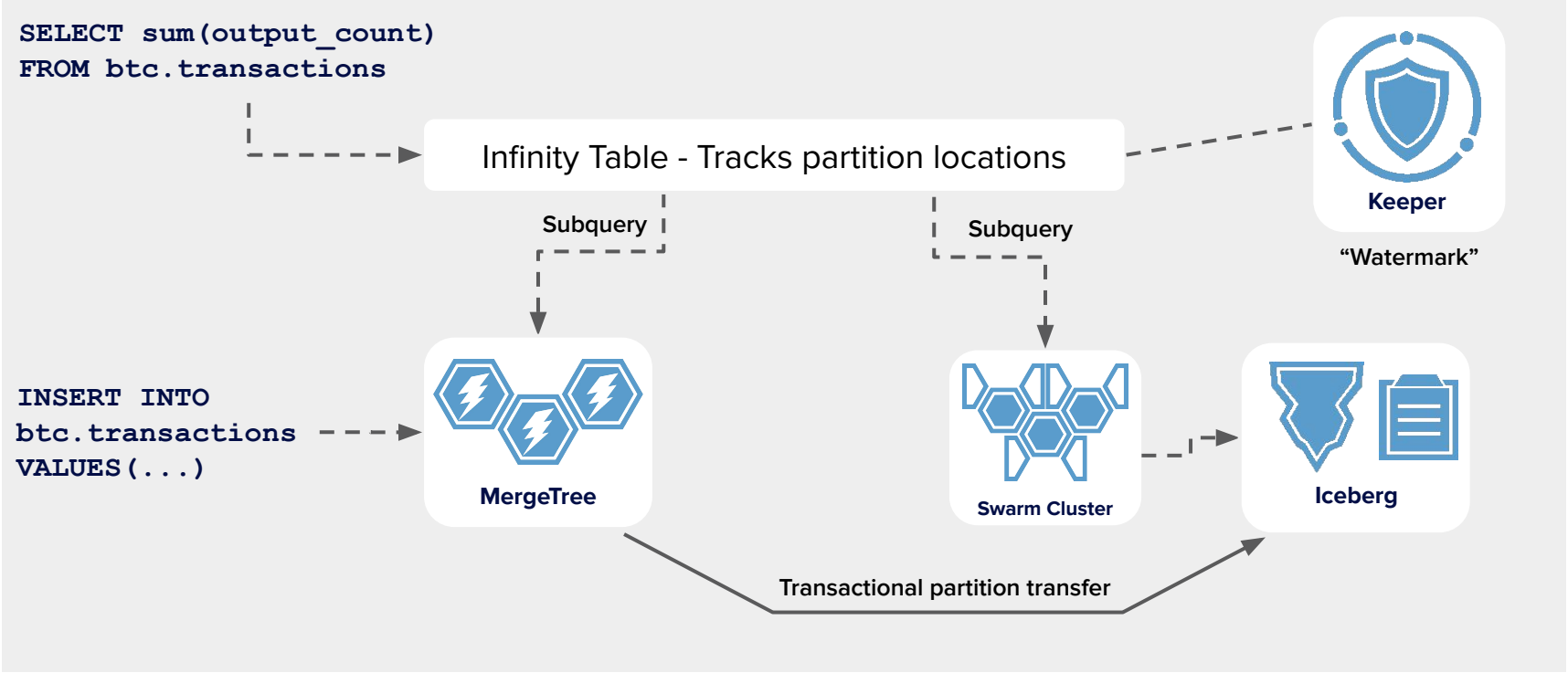
Upstream ClickHouse vs Project Antalya

	Upstream	Antalya
Swarm discovery	25.3+	yes
Catalog databases	yes	yes
Hosted catalogs	n/a	yes
Swarm queries	Partially, using table functions	yes
Parquet metadata cache	not supported	yes
Iceberg metadata cache	25.4+	yes
S3 table buckets	not supported	yes
High performance Iceberg writes	Direct inserts (25.7+)	yes, via ice

Project Antalya Roadmap

1. Open source release – 04/25
2. ice – open source tools for loading and running Iceberg REST catalogs – 05/25
3. Performance report – 06/25
4. Altinity.Cloud release with catalog and swarms support – 06/25
5. AWS S3 tables support - 07/25
6. Altinity Antalya 25.6 release – 09/25
7. Writing to Iceberg tables – Q3/25
8. Infinity (MergeTree+Iceberg) tables – Q4/25
9. Ingest swarms – Q1/26

Infinity Tables – Seamless extension of MergeTree to Iceberg



Project Antalya resources

- Check out Altinity documentation for overview and quickstart

  https://docs.altinity.com/altinityantalya/

- Project Antalya code is in the Altinity ClickHouse repo (log issues there)

  https://github.com/Altinity/ClickHouse

- Read “Getting Started with Altinity’s Project Antalya” to find out more

  https://altinity.com/blog/getting-started-with-altinitys-project-antalya

- Join the Altinity Public Slack to find out more: **https://altinity.com/slack**

Summary

- Project Antalya is extending ClickHouse to use Iceberg as table storage
- Swarm clusters enable compute/storage separation on reads
- Caches are vital to ensure performance
- Additional tricks include spot instances, fast operator reconciliation, and diagnostic queries
- Ice and ice-rest-catalog provide infrastructure
- Altinity.Cloud provides fully managed solution
- Upcoming attractions:
  - Production ready writes to Iceberg
  - Infinity tables
  - Ingest swarms

Thank you! Questions?

Learn more and join our community

https://altinity.com/slack

Q4 meetups: NYC, London, Atlanta, SFO

https://altinity.com

Project Antalya

ALTINITY