DuckLake

The SQL-Powered Lakehouse Format for the Rest of Us

Hannes Mühleisen


D
S QL 1337


Distributed?
7:53 / 34:05

Architecture-Independent Distributed Query Processing

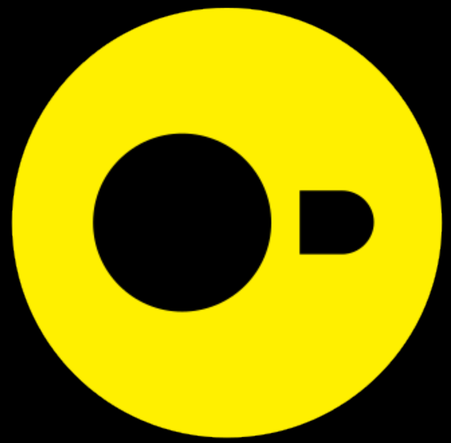
Hannes Fabian Mühleisen

Dissertation submitted to the

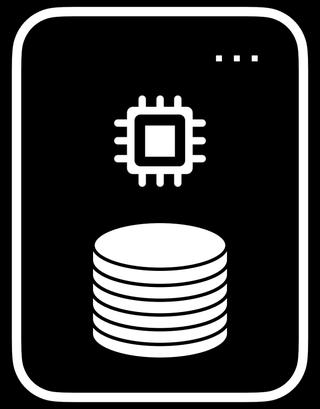
Department of Mathematics and Computer Science

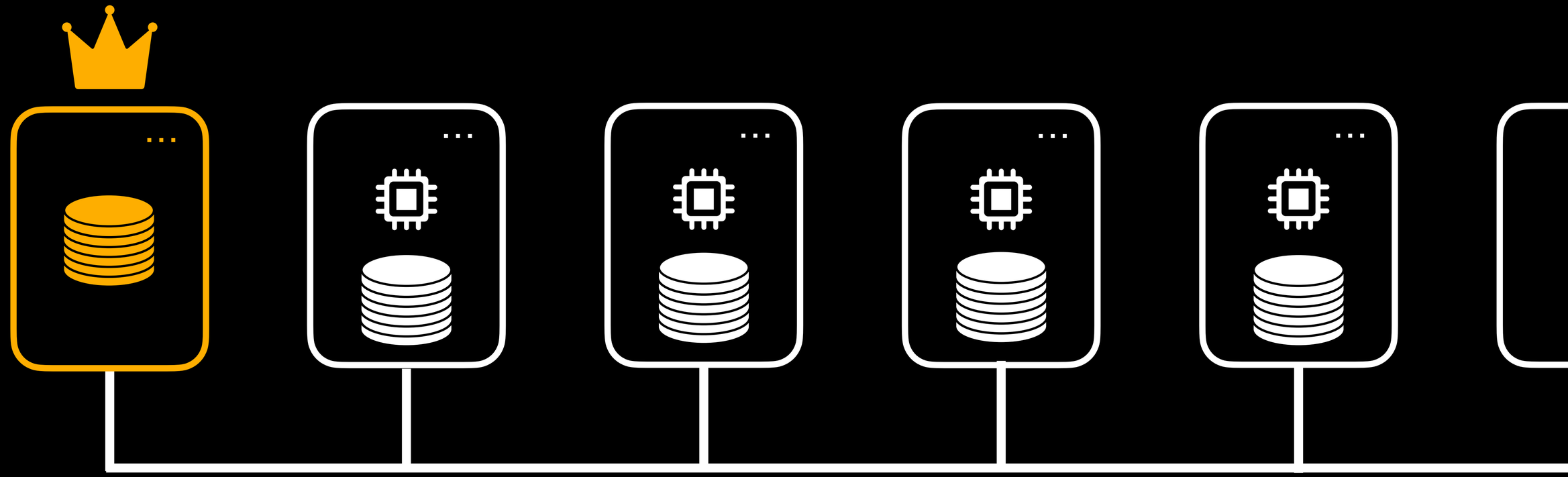
Freie Universität Berlin

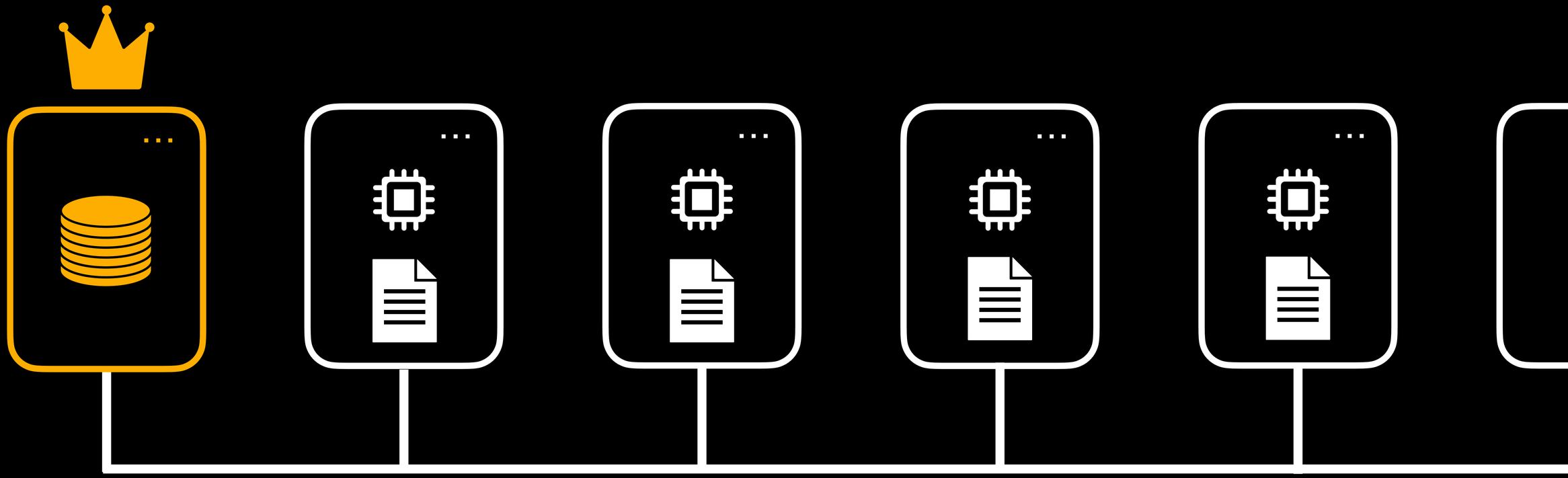
MMXII



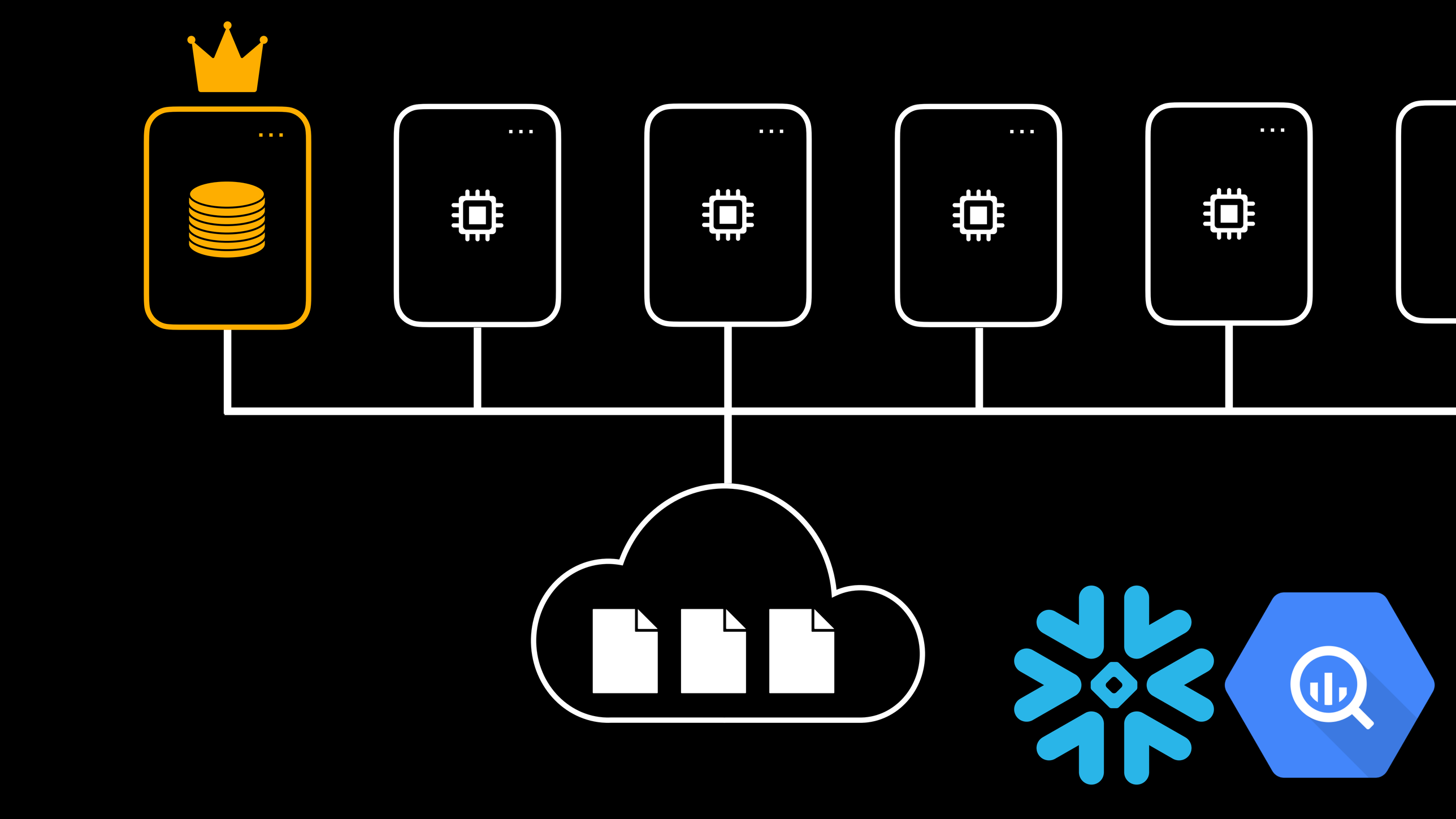
DuckDB

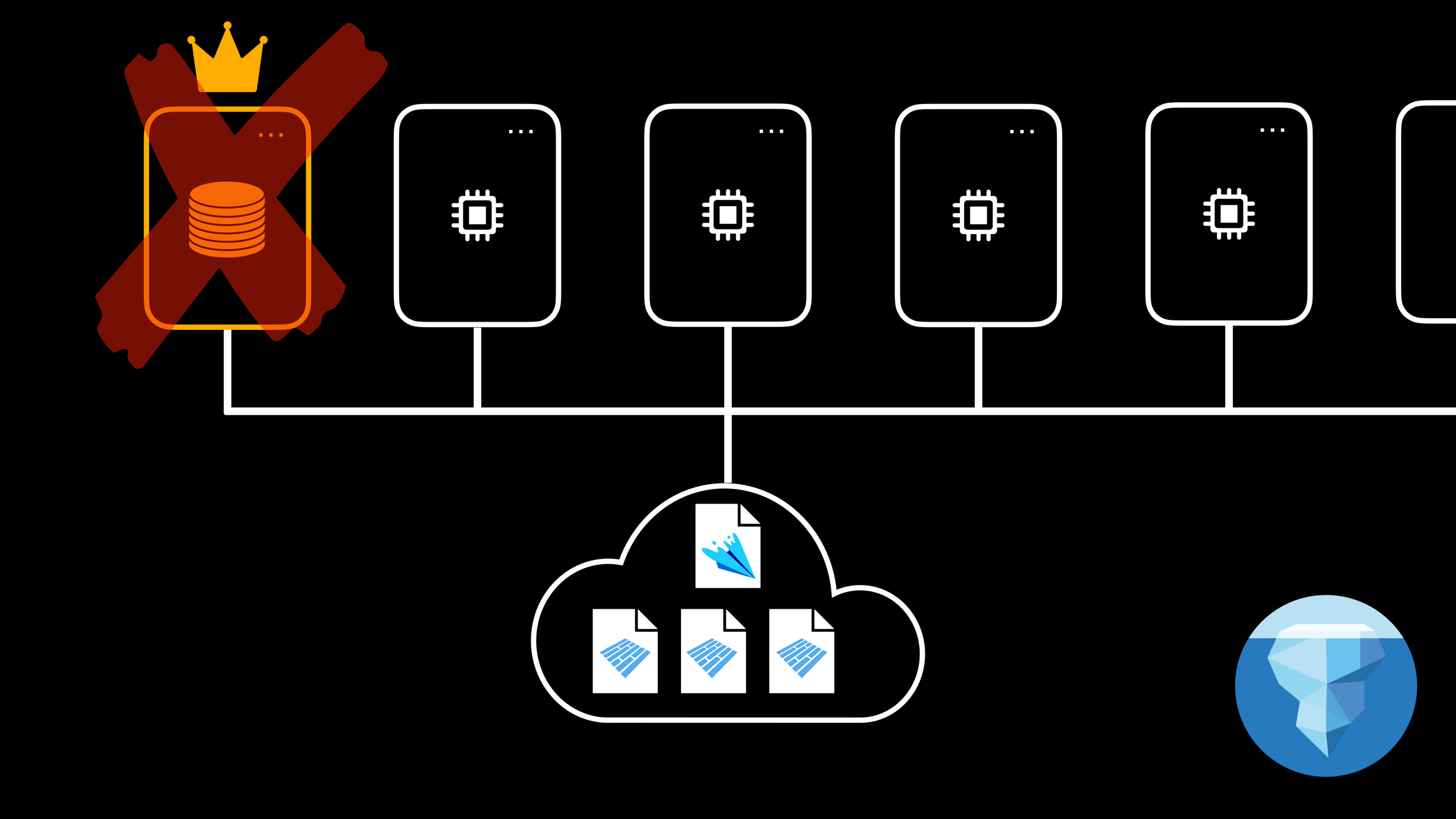


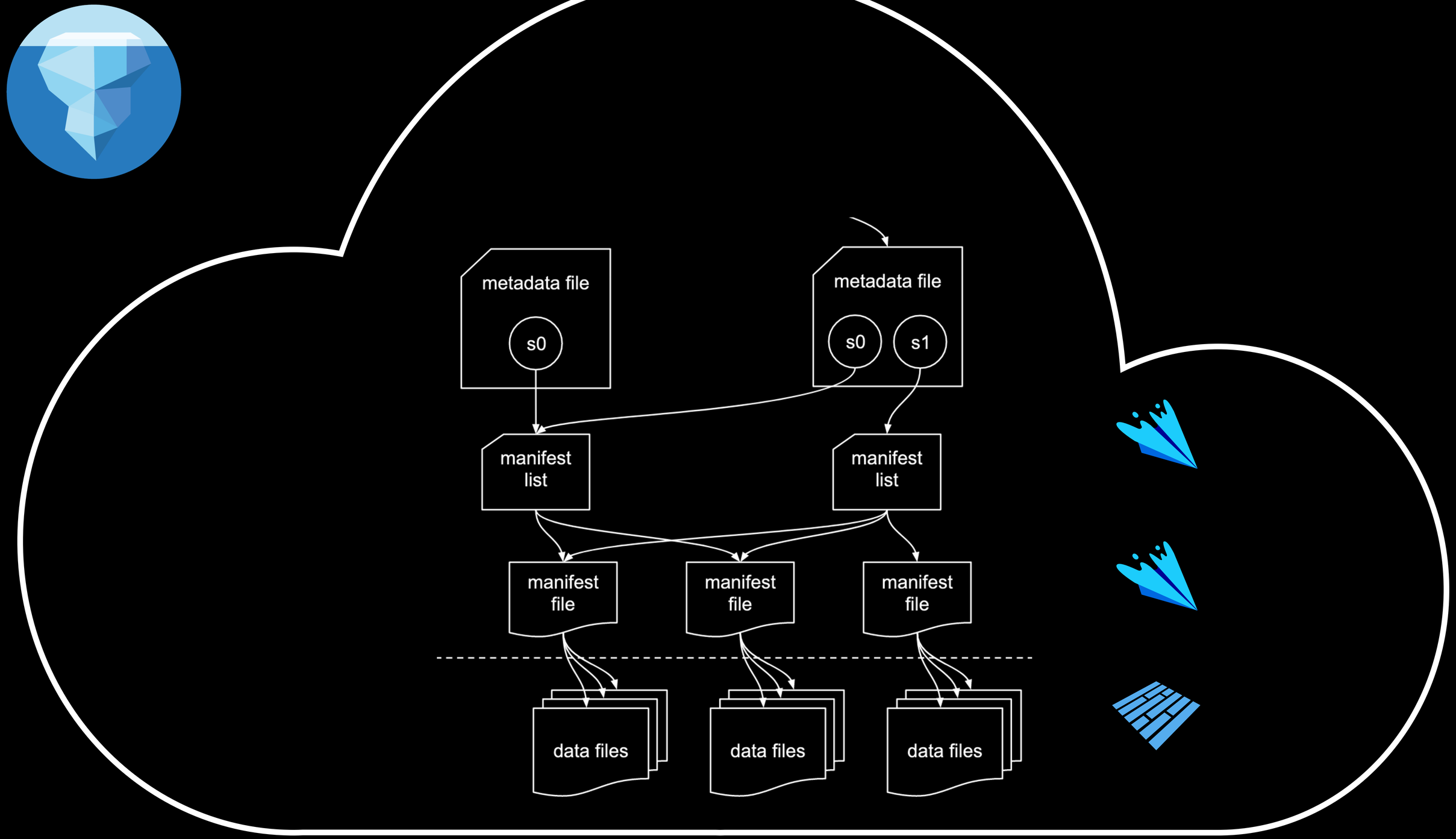





APACHE
hadoop

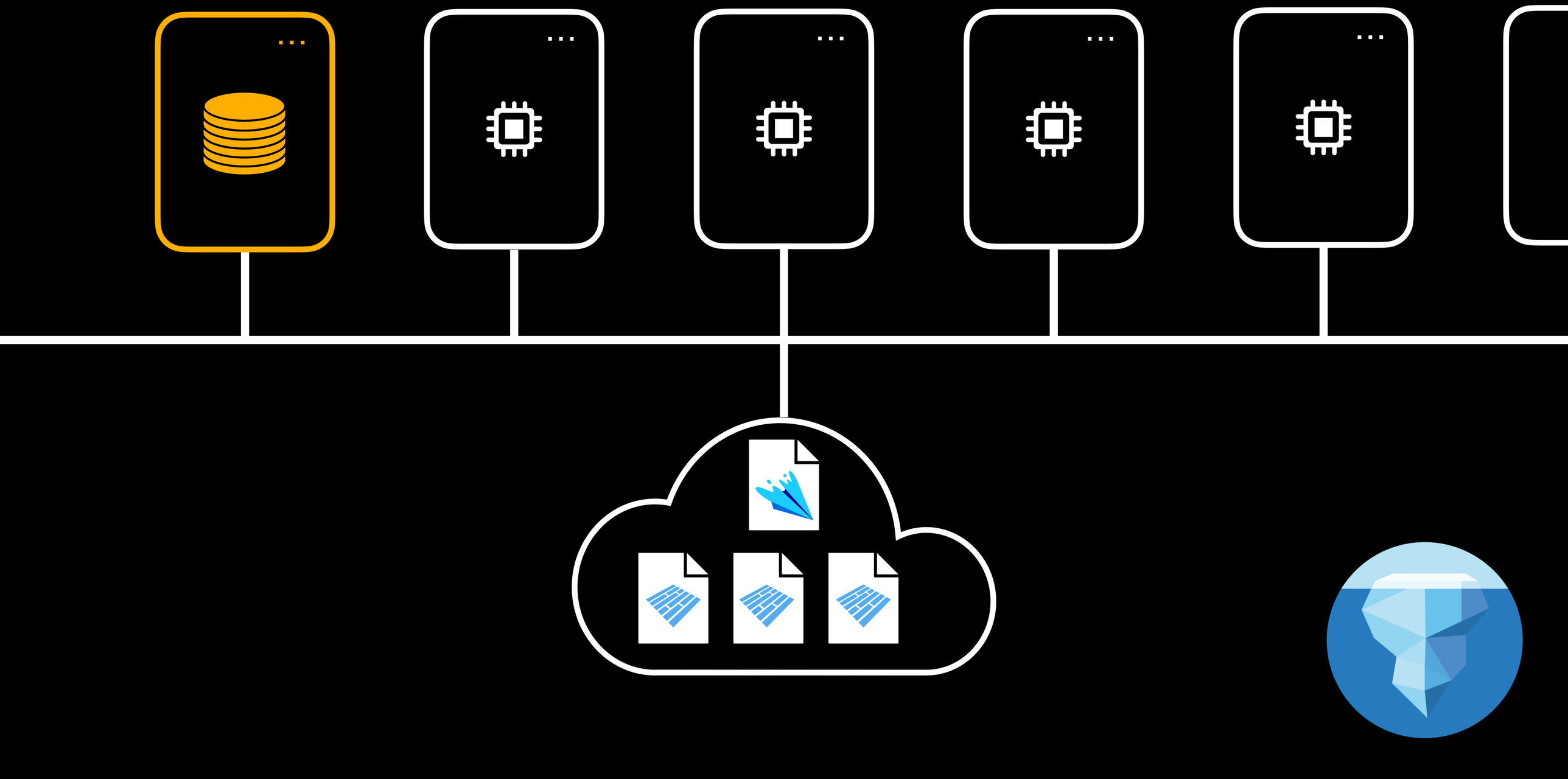


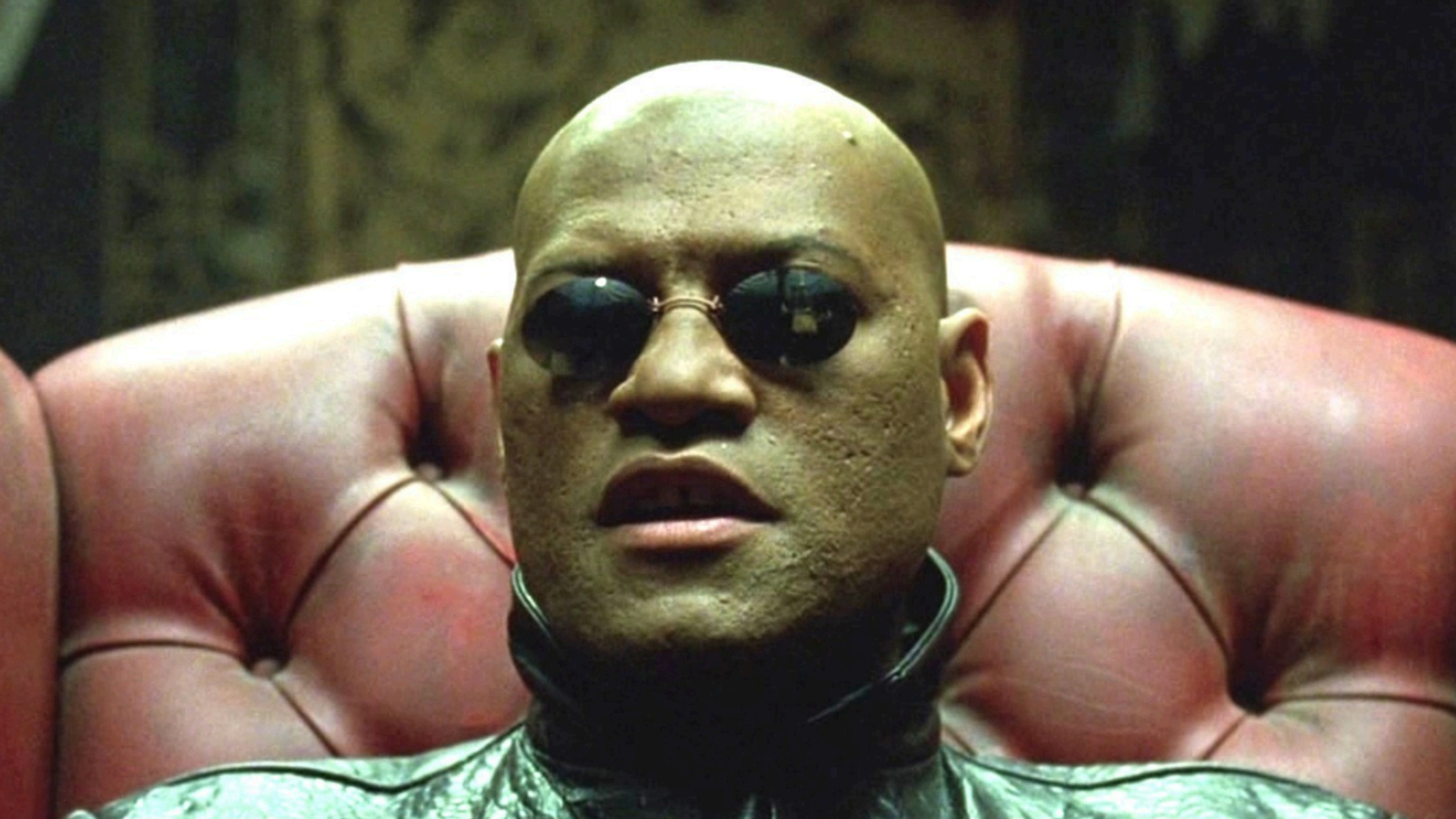



metadata file
s0
metadata file
s0
s1
manifest list
manifest list
manifest file
manifest file
manifest file
data files
data files
data files

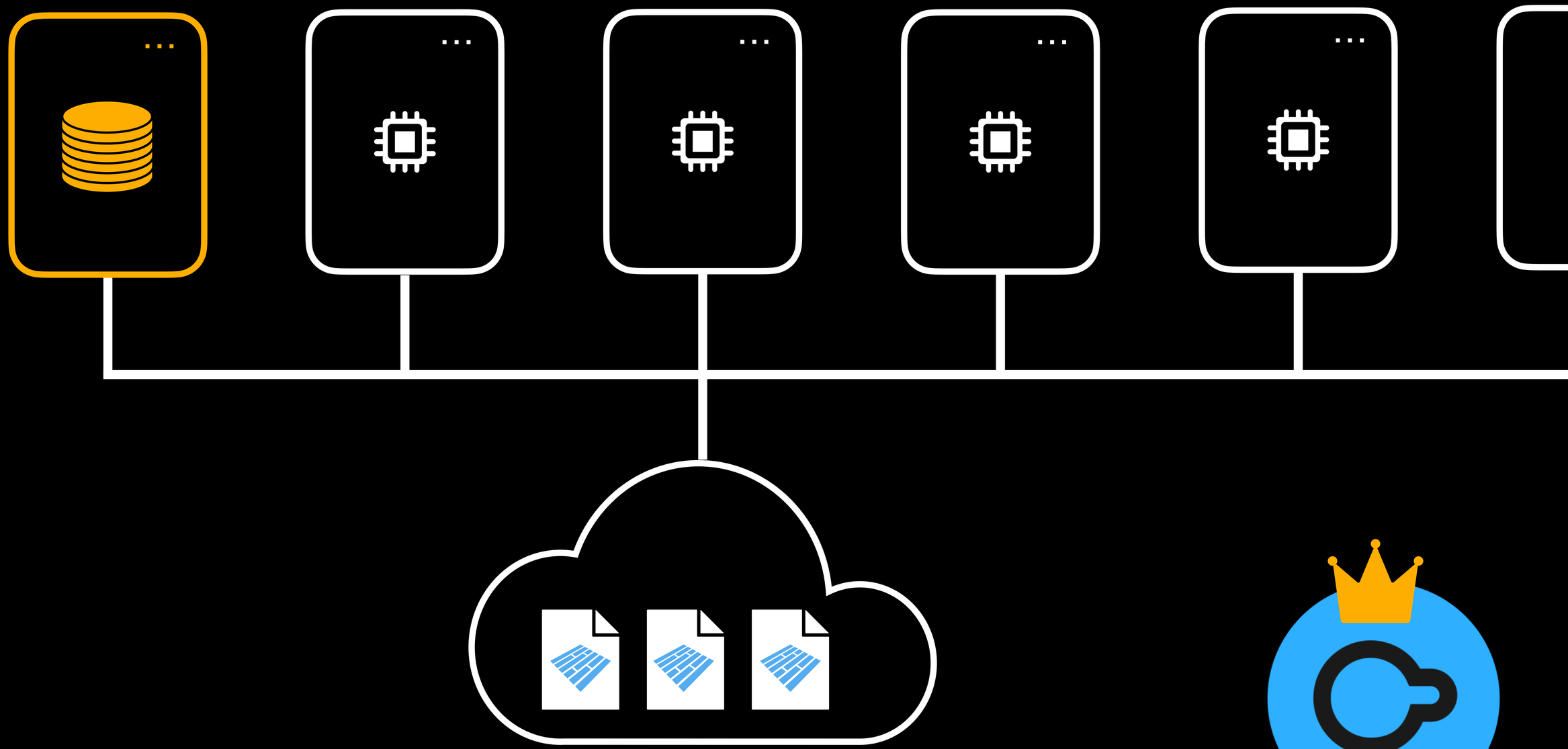
Java
REST
Postgres

Java
REST
Postgres





DuckLake



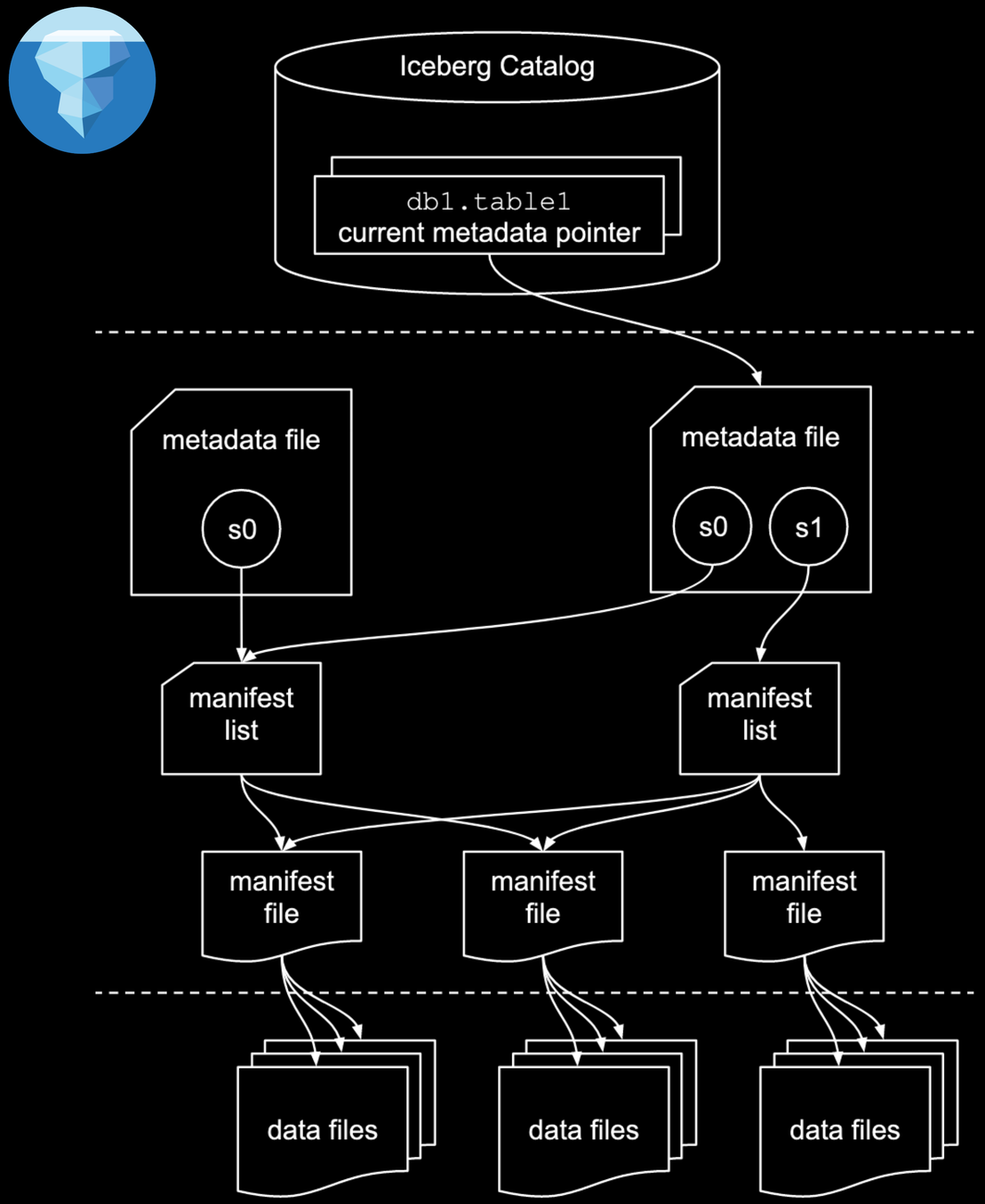
Simplicity
Scalability
Speed

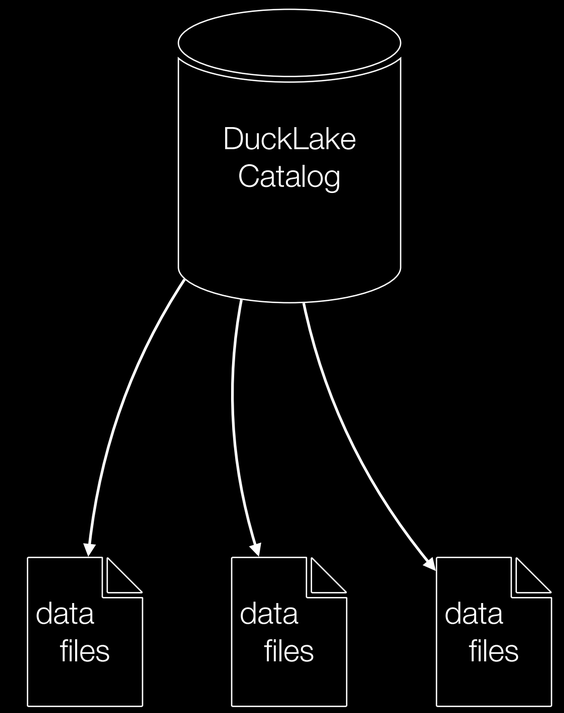
Simplicity

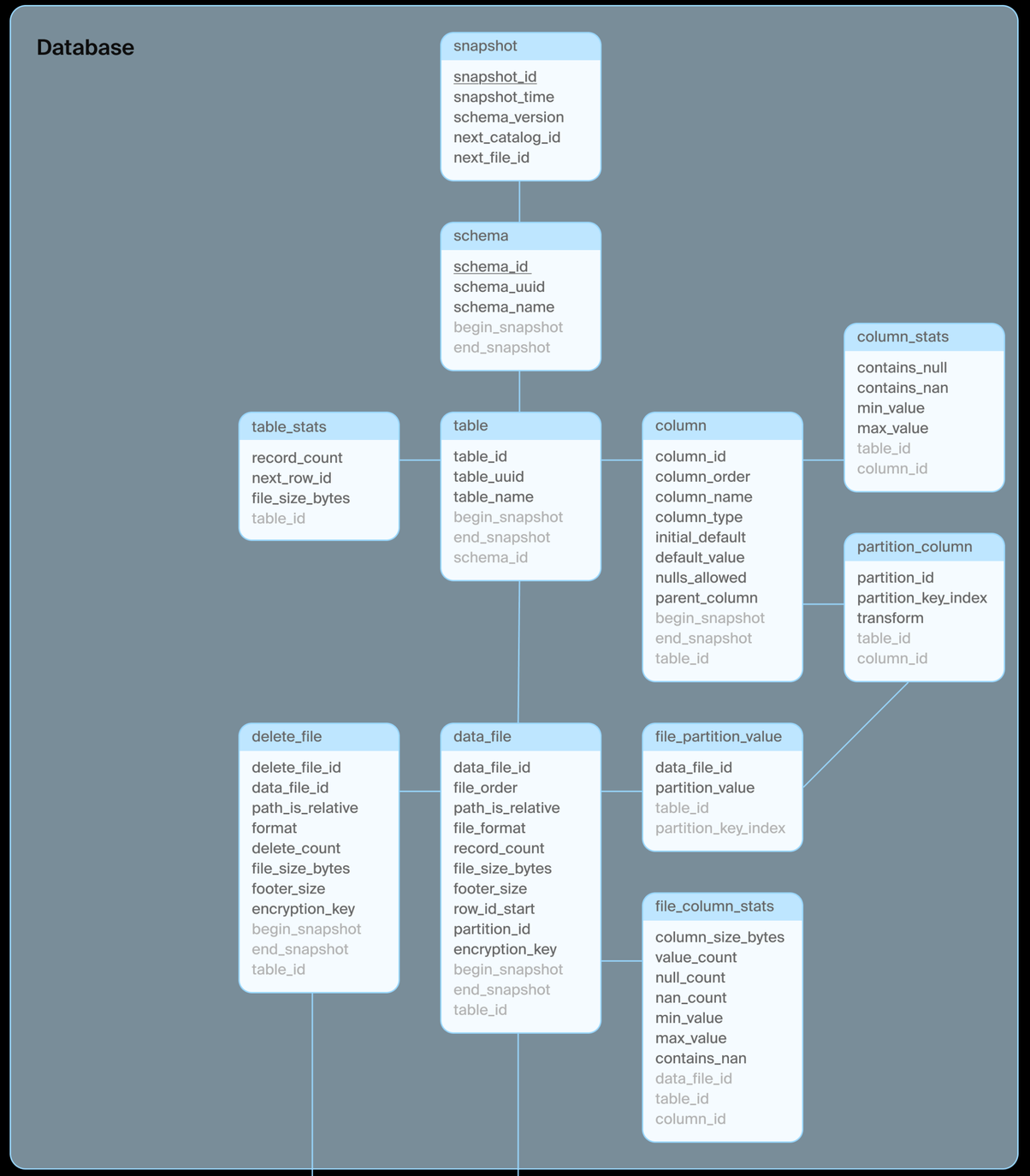
```
$ curl install.duckdb.org | bash
```

```
$ curl install.duckdb.org | bash
$ ~/.duckdb/cli/latest/duckdb
```

```
$ curl install.duckdb.org | bash
$ ~/.duckdb/cli/latest/duckdb
◗ ATTACH 'ducklake:metadata.ducklake' AS dl1;
```


Iceberg Catalog
db1.table1
current metadata pointer
metadata file
s0
metadata file
s0
s1
manifest list
manifest list
manifest file
manifest file
manifest file
data files
data files
data files


DuckLake Catalog
data files
data files
data files

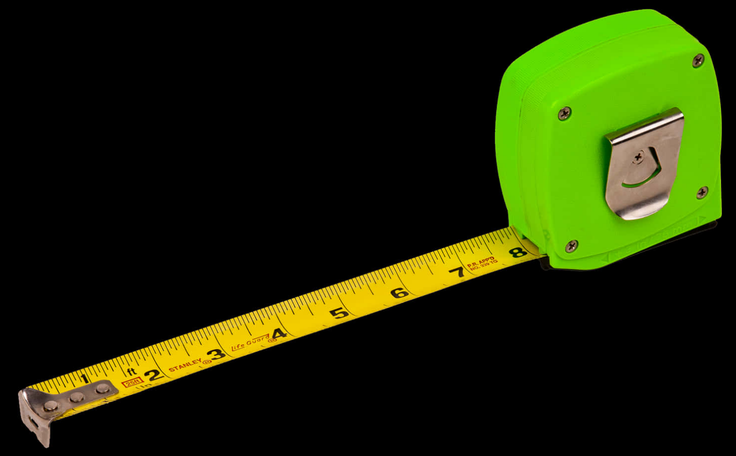

Database
snapshot
snapshot_id
snapshot_time
schema_version
next_catalog_id
next_file_id
schema
schema_id
schema_uuid
schema_name
begin_snapshot
end_snapshot
column_stats
contains_null
contains_nan
min_value
max_value
table_id
column_id
table_stats
record_count
next_row_id
file_size_bytes
table_id
table
table_id
table_uuid
table_name
begin_snapshot
end_snapshot
schema_id
column
column_id
column_order
column_name
column_type
initial_default
default_value
nulls_allowed
parent_column
begin_snapshot
end_snapshot
table_id
partition_column
partition_id
partition_key_index
transform
table_id
column_id
delete_file
delete_file_id
data_file_id
path_is_relative
format
delete_count
file_size_bytes
footer_size
encryption_key
begin_snapshot
end_snapshot
table_id
data_file
data_file_id
file_order
path_is_relative
file_format
record_count
file_size_bytes
footer_size
row_id_start
partition_id
encryption_key
begin_snapshot
end_snapshot
table_id
file_partition_value
data_file_id
partition_value
table_id
partition_key_index
file_column_stats
column_size_bytes
value_count
null_count
nan_count
min_value
max_value
contains_nan
data_file_id
table_id
column_id

Standard <> Implementation

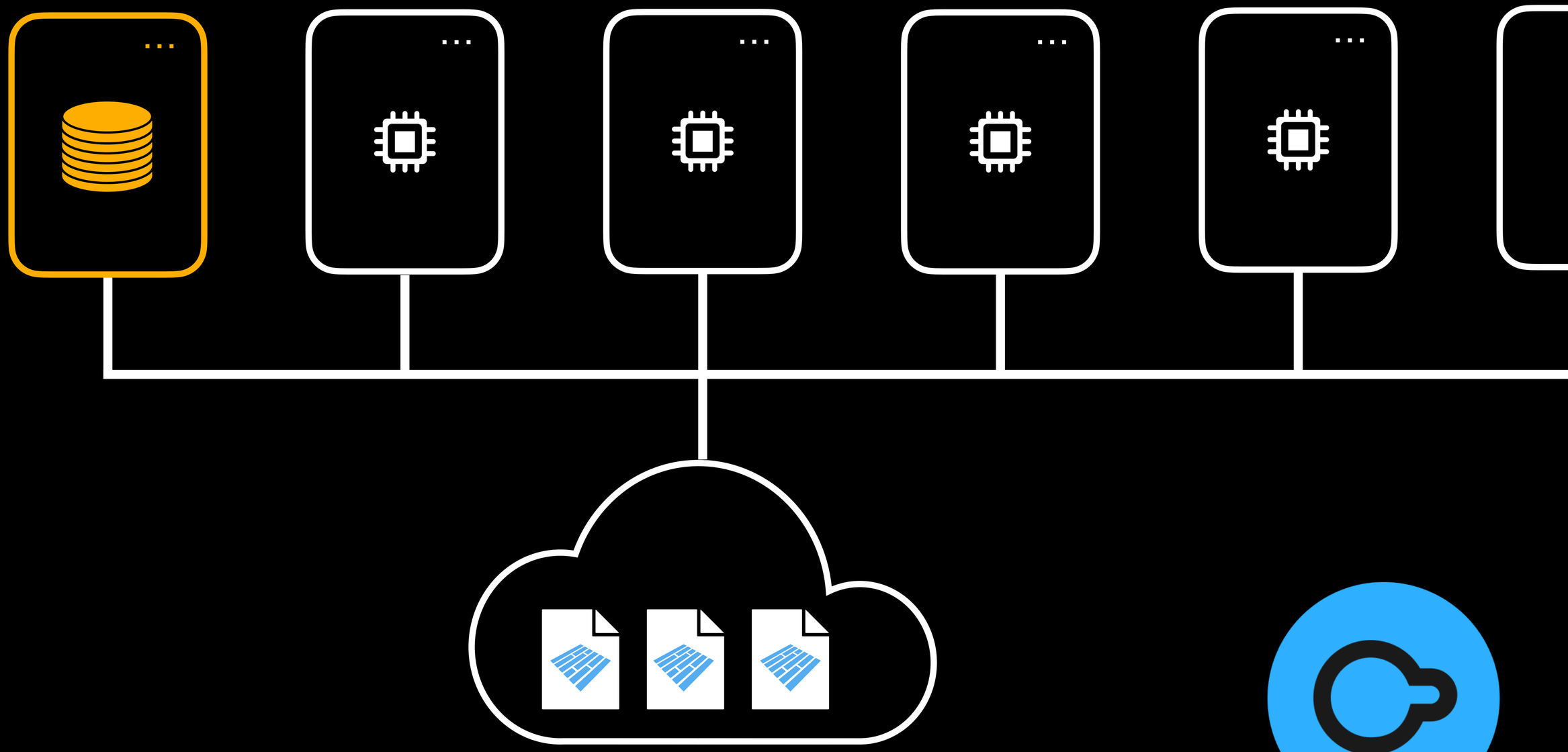

- Postgres
- Docker
- Polaris
- Java™
- REST
- JSON
- AVRO
- Parquet
- S3
- Spark

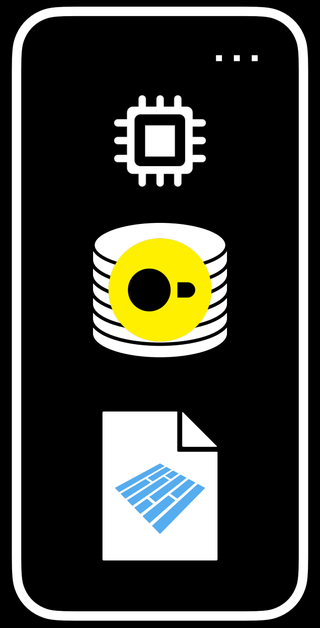
- DuckDB
- (Postgres)
- (S3)

Scalability

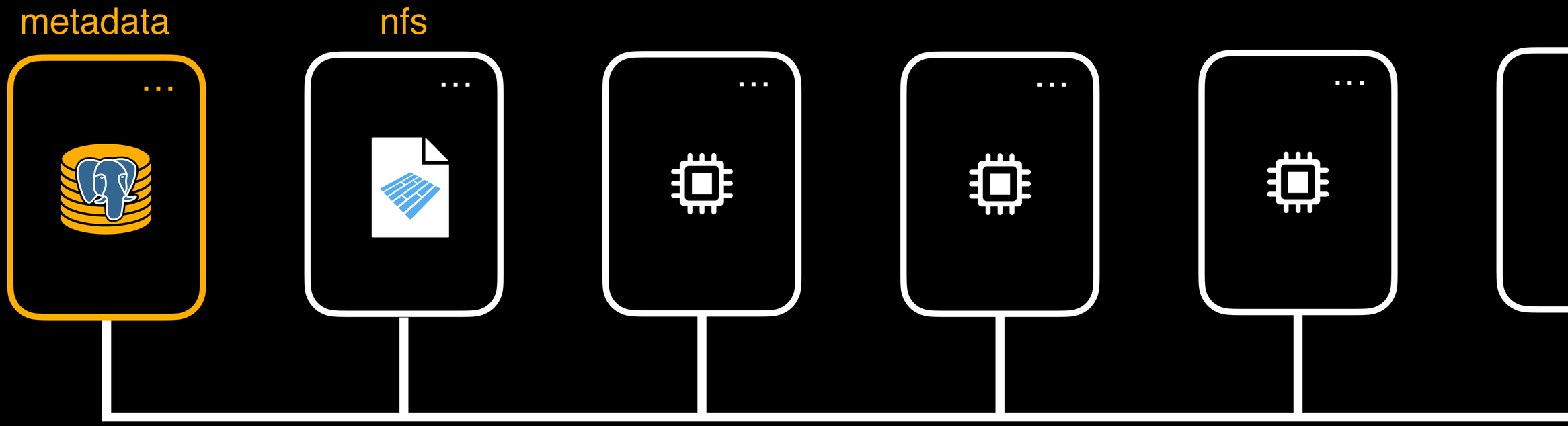



7
7
7

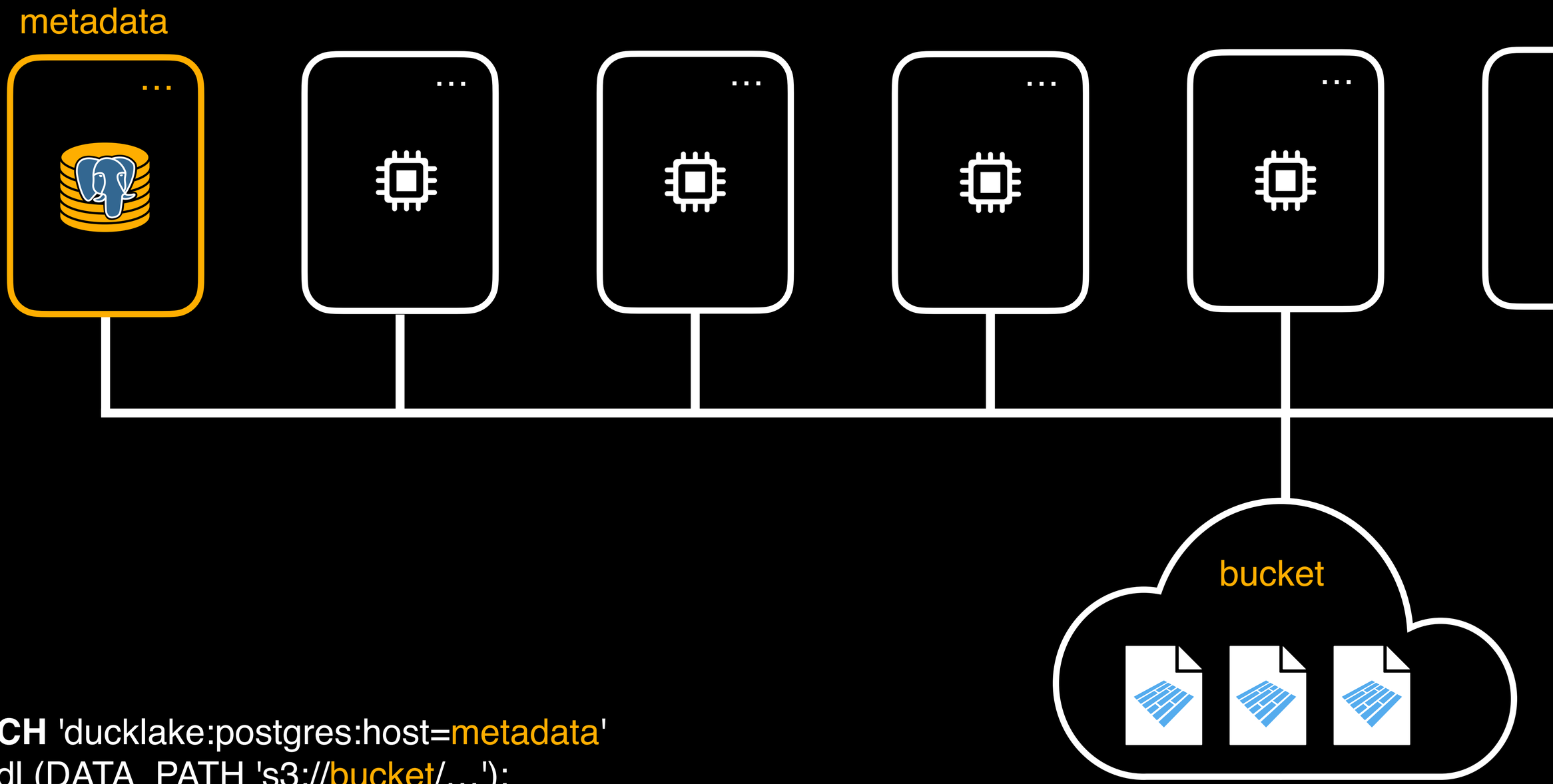




ATTACH 'ducklake:metadata1.ducklake' AS dl1;




```
ATTACH 'ducklake:postgres:host=metadata' AS dl
   (DATA_PATH '/nfs/data_files/');
```

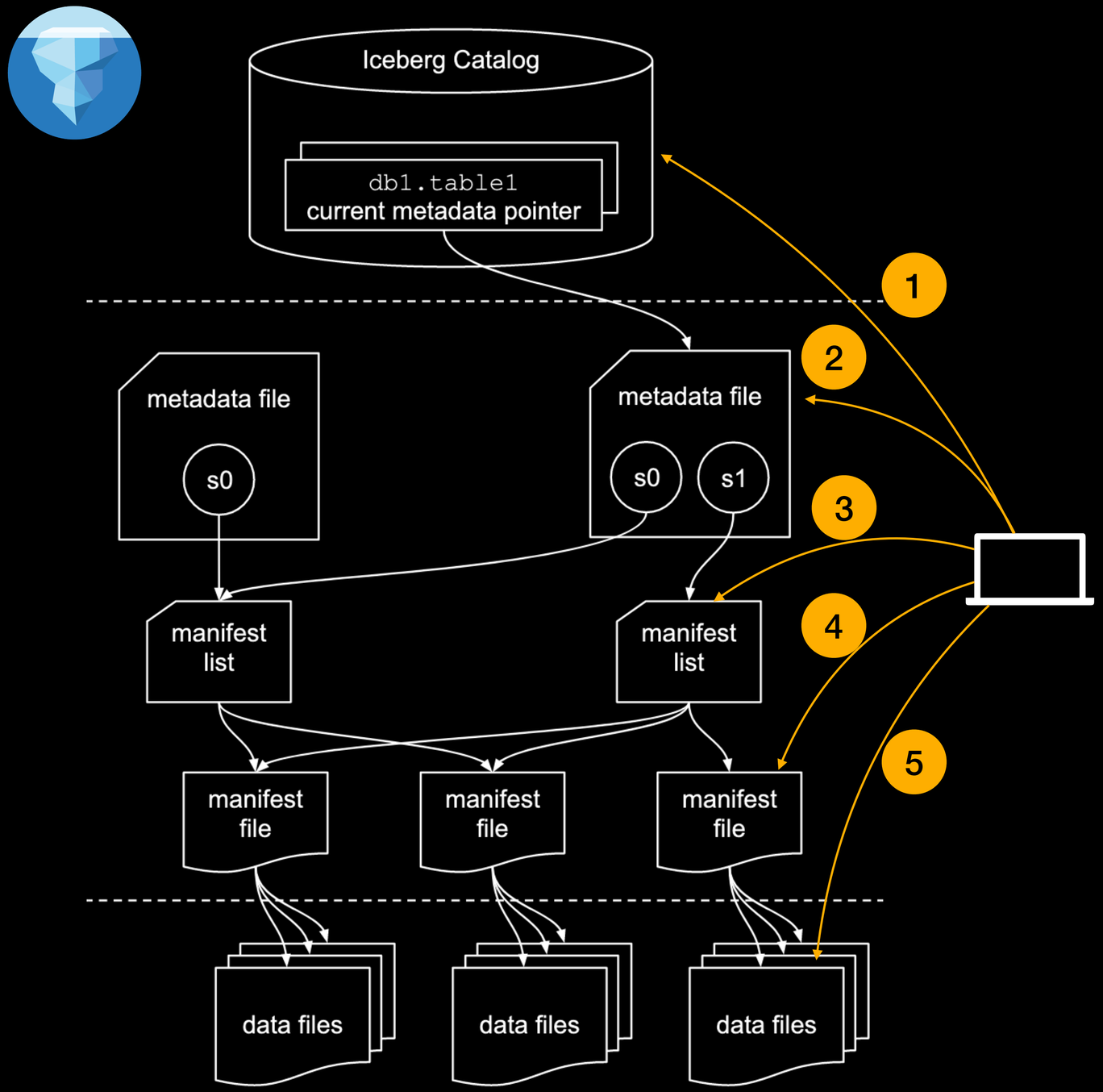


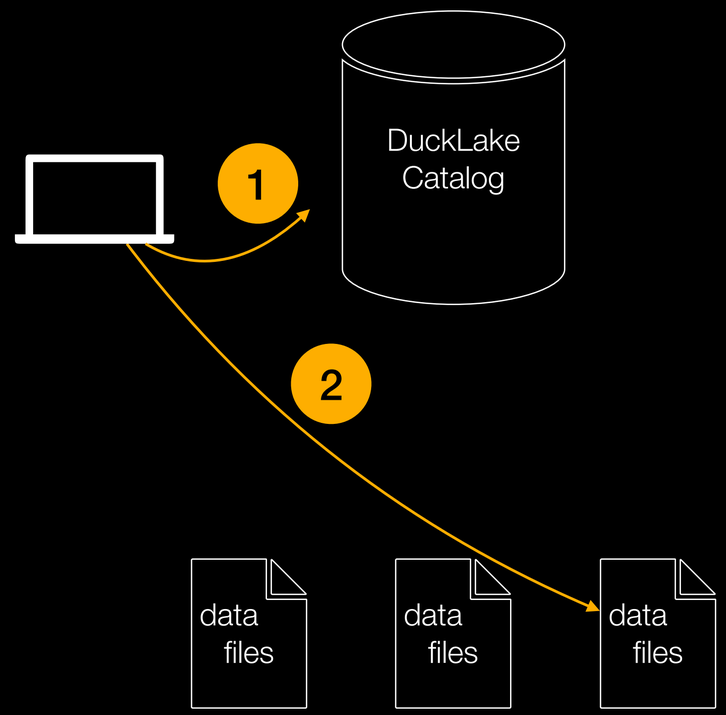

```
ATTACH 'ducklake:postgres:host=metadata'
  AS dl (DATA_PATH 's3://bucket/…');
```

- 1 Petabyte of data
- 10 Columns
- 100 M snapshots, 10 years
  - 270 GB/day

Planning Time	DuckDB	Postgres
Insert	~5ms	~3ms
Select All Files	~0.8s	~200s
Select 1 Month	~0.25s	~0.25s
Select 1 Day	~0.25s	~0.25s

Speed


Iceberg Catalog
db1.table1
current metadata pointer
1
2
3
4
5
metadata file
s0
metadata file
s0
s1
manifest list
manifest list
manifest file
manifest file
manifest file
data files
data files
data files


1
DuckLake Catalog
2
data files
data files
data files

Fair Benchmarking Considered Difficult: Common Pitfalls In Database Performance Testing

Mark Raasveldt, Pedro Holanda, Tim Gubner & Hannes Mühleisen
Centrum Wiskunde & Informatica (CWI)
Amsterdam, The Netherlands
[raasveld,holanda,tgubner,hannes]@cwi.nl

ABSTRACT

Performance benchmarking is one of the most commonly used methods for comparing different systems or algorithms, both in scientific literature and in industrial publications. While performance measurements might seem objective on the surface, there are many different ways to influence benchmark results to favor one system over the other, either by accident or on purpose. In this paper, we perform a study of the common pitfalls in DBMS performance comparisons, and give advice on how they can be spotted and avoided so a fair performance comparison between systems can be made. We illustrate the common pitfalls with a series of mock benchmarks, which show large differences in performance where none should be present.

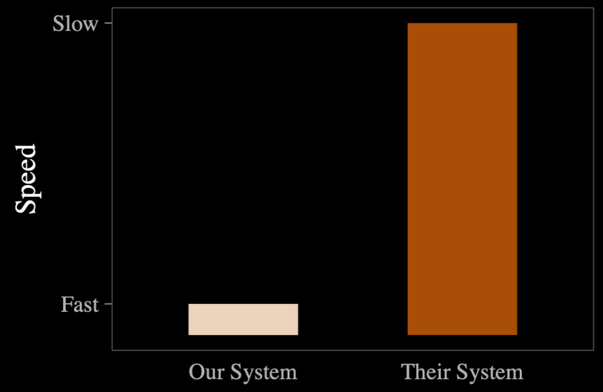
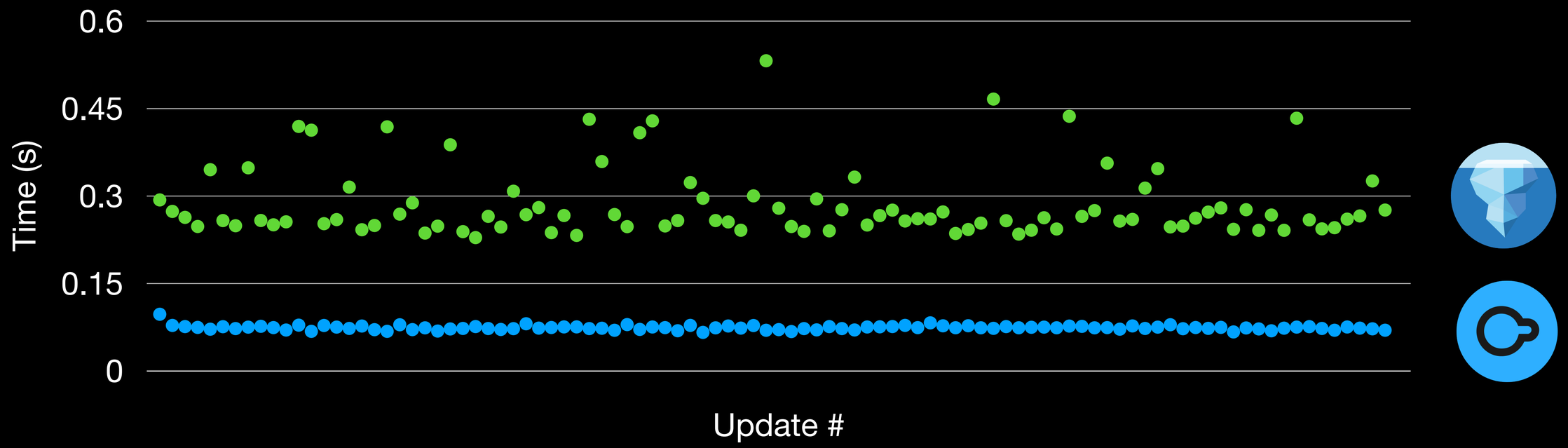


Figure 1: Generic benchmark results.


0.6
0.45
0.3
0.15
0
Time (s)
Update #

```
        ------------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent-1223a723-f206-4458-857d-d14611b6a7a9
        confs: [default]
        0 artifacts copied, 2 already retrieved (0kB/2ms)
25/06/18 13:51:11 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java
sses where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
[Stage 1:>                                                          25/06/18 13:51:24 WARN GarbageCollectionMetrics: To enable
n-built-in garbage collector(s) List(G1 Concurrent GC), users should configure it(them) to spark.eventLog.gcMetrics.yo
GenerationGarbageCollectors or spark.eventLog.gcMetrics.oldGenerationGarbageCollectors
[Stage 1:===>                                                       [Stage 1:======>
Stage 1:================>                                          [Stage 1:=================================>
0.2873690128326416

💣 Program crashed: Bad pointer dereference at 0x0000db8a0000db95

Thread 0:

0                0x0000000180764c34 mach_msg2_trap + 8 in libsystem_kernel.dylib
1 [ra]           0x000000018076d764 mach_msg_overwrite + 484 in libsystem_kernel.dylib
2 [ra]           0x0000000180764fa8 mach_msg + 24 in libsystem_kernel.dylib
3 [ra]           0x0000000180891e7c __CFRunLoopServiceMachPort + 160 in CoreFoundation
4 [ra]           0x0000000180890798 __CFRunLoopRun + 1208 in CoreFoundation
5 [ra]           0x000000018088fc58 CFRunLoopRunSpecific + 572 in CoreFoundation
6 [ra]           0x0000000104ac326c CreateExecutionEnvironment + 400 in libjli.dylib
7 [ra]           0x0000000104abf344 JLI_Launch + 1128 in libjli.dylib
8 [ra]           0x0000000104a83bac main + 392 in java
9 [ra] [system]  0x0000000180406b98 start + 6076 in dyld
```

Multi-Table

```
ATTACH 'ducklake:metadata.ducklake' AS dl;

CREATE SCHEMA dl.s1;
CREATE TABLE  dl.s1.t1 …;
CREATE TABLE  dl.s1.t2 …;
…
```

Holistic Transactions

```
BEGIN
  CREATE TABLE …
  ALTER TABLE …
  INSERT … INTO t1
  UPDATE t2 …
  DELETE FROM t3
COMMIT
```

Time Travel

```
FROM tbl AT (VERSION => 3);
FROM tbl AT (TIMESTAMP => …);
```

Change Feed

```
ATTACH 'ducklake:metadata.ducklake' AS dl;
FROM dl.table_changes('tbl', 3, 4);
```

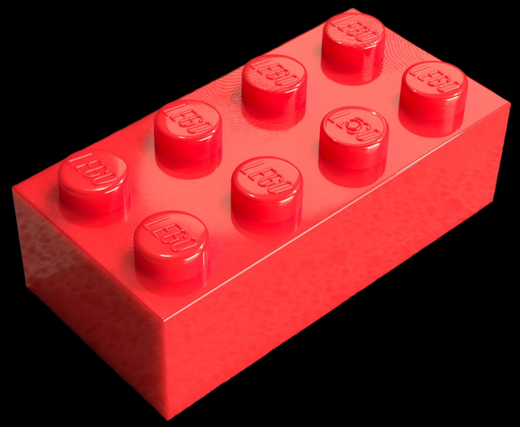
Inlining

```
ATTACH 'ducklake:metadata.ducklake' AS dl
  (DATA_INLINING_ROW_LIMIT 100);
```

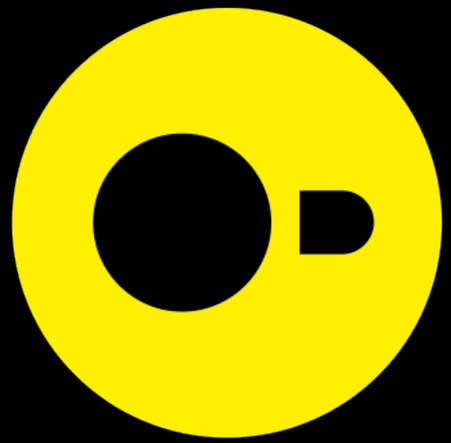
Encryption

```
ATTACH 'ducklake:metadata.ducklake' AS dl
   (DATA_PATH '/some/untrusted/location' ENCRYPTED);
```

Interoperability

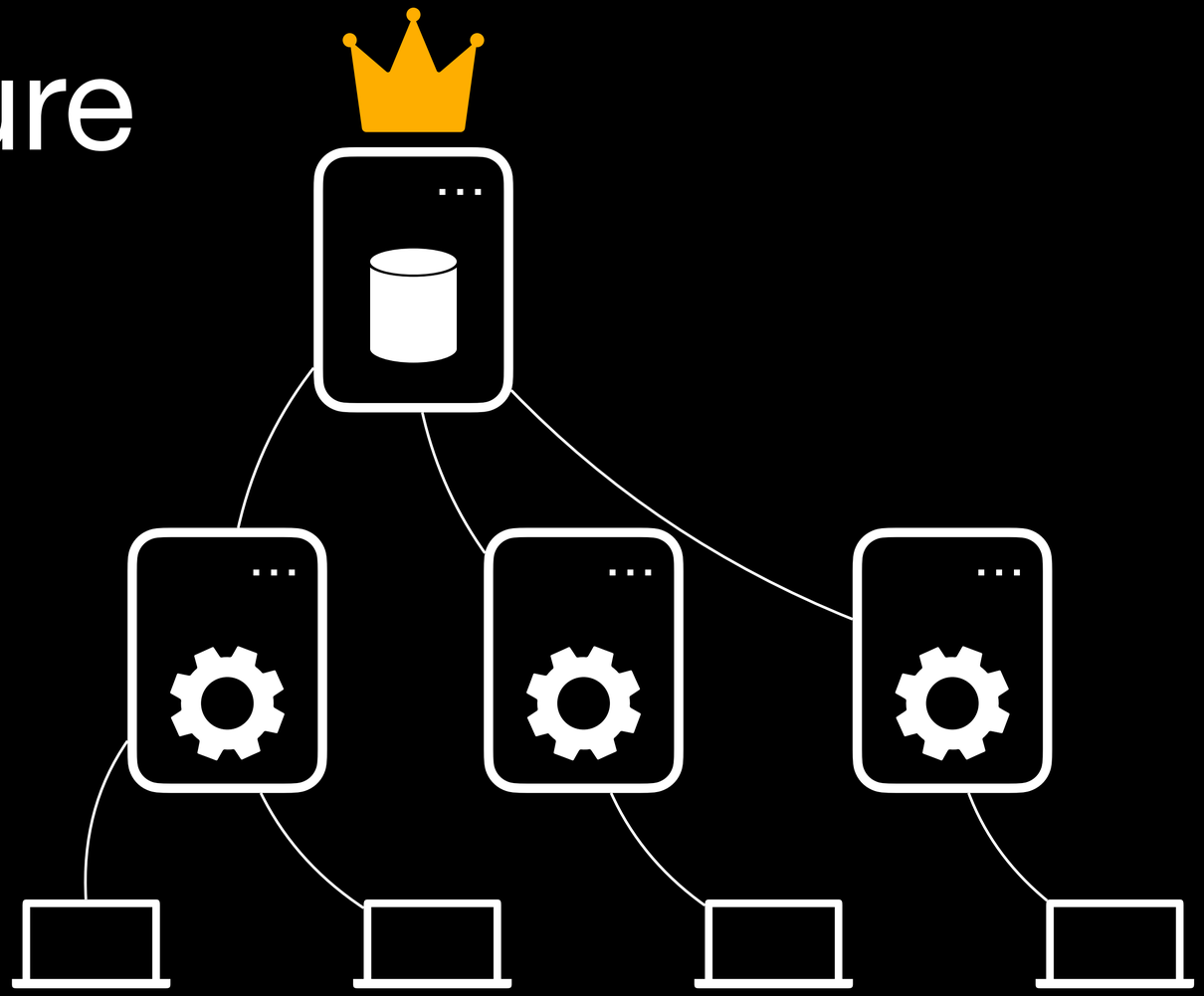


Free



DuckDB

1985 Data Architecture

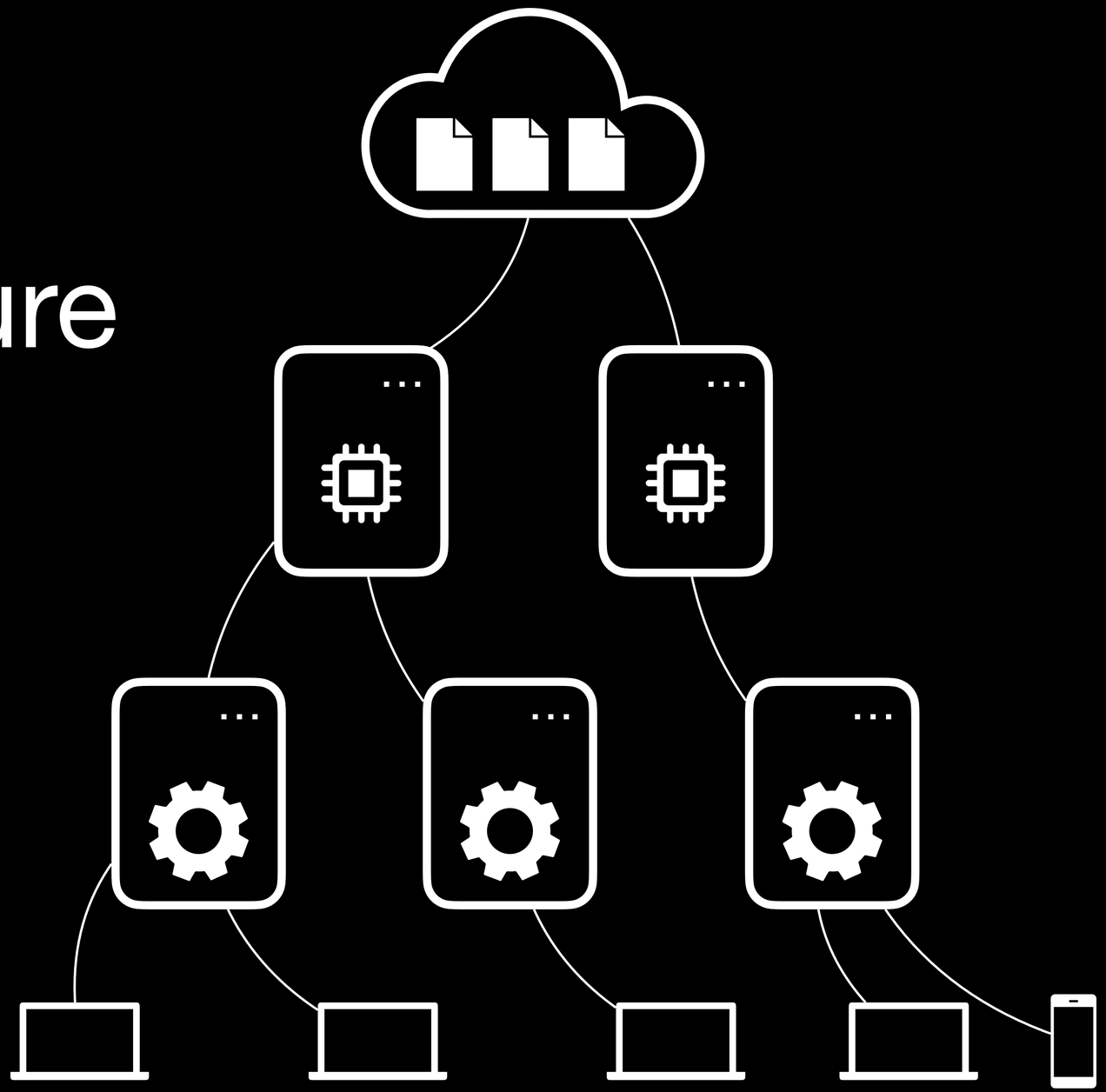


Data Warehouse

Application Servers

Clients

2015
Data Architecture

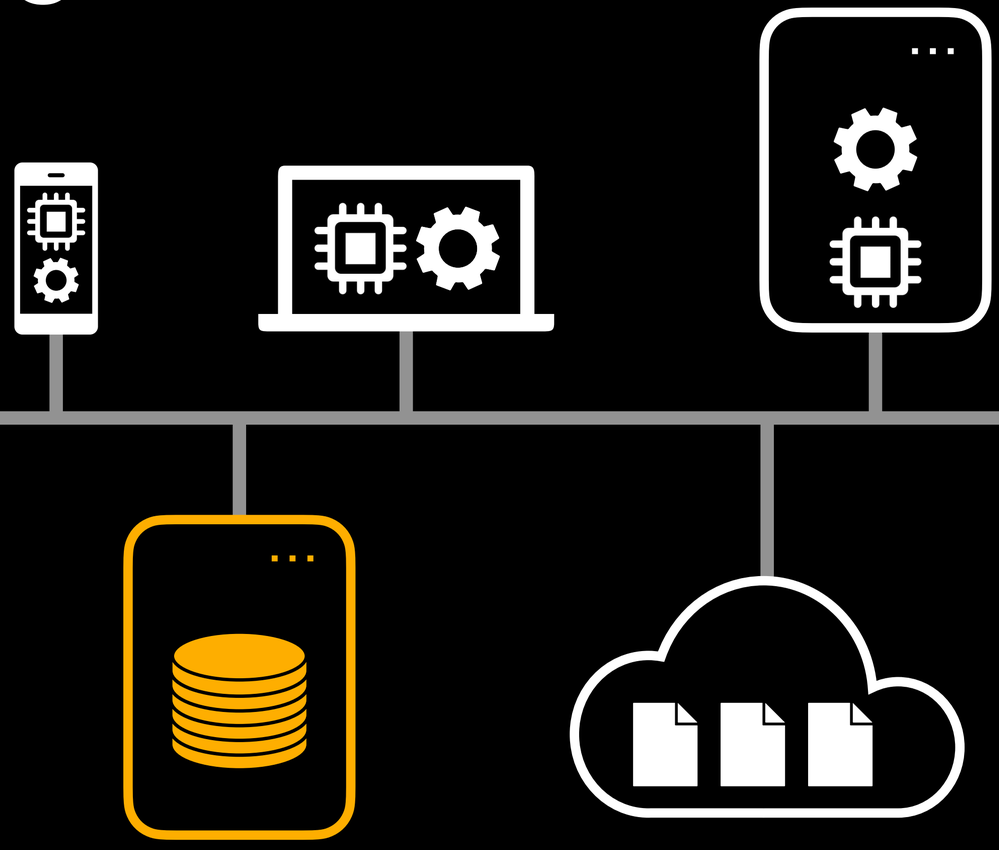


Cloud
Data Warehouse

Application Servers

Clients

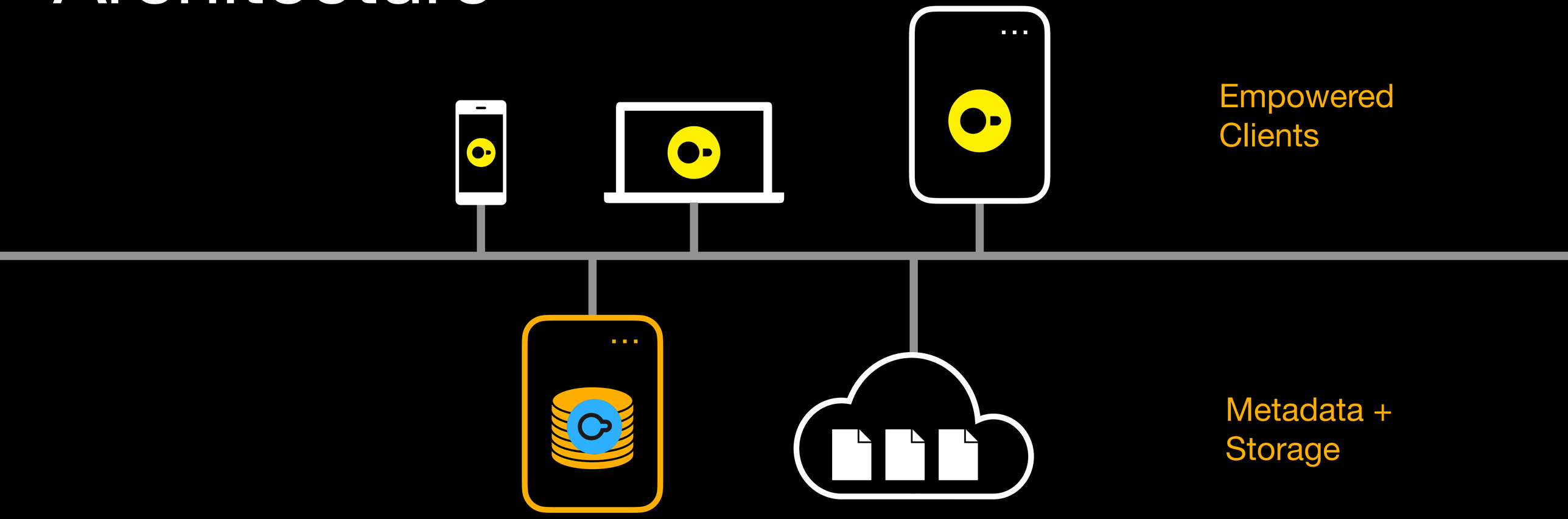
2025 Data Architecture

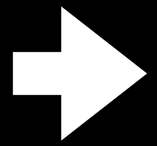


Empowered Clients

Metadata + Storage

2025 Data Architecture



Data Fear  Data Confidence

DuckLake

ducklake.select
hannes.muehleisen.org