CRATEDB: A SEARCH ENGINE OR A DATABASE? BOTH!

HOW WE BUILT A SQL DATABASE ON TOP OF ELASTICSEARCH AND LUCENE

CRATE.IO

Maximilian Michels
@stadtlegende
max@crate.io
mxm@apache.org

WHY ARE WE TALKING ABOUT THIS?

- Traditional databases are well-researched and there are plenty of them (Postgres, MySQL, Oracle…)
  - Scalable search using these can be tricky
- Search engines are databases optimized for search and scale (Lucene, Solr, Elasticsearch)
  - You can’t typically use SQL with Search Engines
- Why not stick with a mature query language standard which everybody knows?

HISTORY OF SQL

- First draft of SQL is from 1974
- Latest draft is from 2016
- SQL is 44 (!) years old
- It is a mature standard
- It is a great query specification language
- Why break with it?

1974?

HOW SQL BECAME NOSQL

- Distributed databases focused on an entirely new problem
  - How to distribute data?
  - How to ensure we can find the data again?
  - Consistency vs Availability vs Partition Tolerance
- Implementing (distributed) SQL is complex
  - *We can build an API which is much better and simpler than SQL* !?
  - *Put/Get should be enough, the rest can be handled by the client* !?
  - *Finally we can leave all the legacy behind…*

THE RETURN OF SQL

- Ecosystem
  - Millions of developers/data scientists know SQL
  - There are endless tools compatible with SQL
- Query Expressiveness
  - Non-trivial queries are difficult to model with NoSQL
  - Simplicity of NoSQL means more complexity on the applications layer
- SQL actually makes sense!!

“A scalable SQL database optimized for search
without the NoSQL bullshit.”

CrateDB

- Since 2014: https://github.com/crate/crate
- Apache 2.0 licensed (community edition)
- Built using Elasticsearch, Lucene, Netty, Antlr, …
- SQL-99 compatible
- REST / Postgres Wire Protocol / JDBC / Python …

WHAT TO EXPECT

- What is great about CrateDB
  - Easy to setup
  - No funny APIs, just SQL
  - Excellent search performance
  - Great scale out - Massive reads / writes
  - Great documentation
  - Container aware
- Not so great
  - Transactions

USING CRATEDB

CrateDB

CRATEDB IS JUST LIKE A SQL DB

- SQL is the only query API

```
CREATE TABLE buzzwords.speakers (id INT PRIMARY KEY, name STRING)
CREATE TABLE buzzwords.talks (id INT PRIMARY KEY, title STRING,
                              abstract STRING, speaker INT);

INSERT INTO buzzwords.speakers (id, name) VALUES (1, 'max')
INSERT INTO buzzwords.talks (id, title, abstract, speaker) VALUES
                            (1, 'Talk about CrateDB', 'bla', 1)

SELECT * FROM buzzwords.talks t1 LEFT JOIN buzzwords.speakers t2 ON t1.id = t2.id
```

CrateDB

BUT THERE IS MORE

- denormalized (no joins necessary)

```
CREATE TABLE buzzwords.speakers
      (name STRING, talk OBJECT AS (title STRING, abstract STRING))
```

```
INSERT INTO buzzwords.speakers (name, talk) VALUES
            ('max', {title = 'CrateDB', abstract = 'Lorem ipsum'})
```

```
SELECT talk['title'] as title FROM buzzwords.speakers
ORDER BY title
```

CLUSTERING

- `CREATE TABLE buzzwords.speakers (name STRING, talk OBJECT AS (title STRING, abstract STRING))`
- `CLUSTERED BY name into 4 shards`

SHARD

CrateDB

CLUSTERING / REPLICATION

- `CREATE TABLE buzzwords.speakers (name STRING, talk OBJECT AS (title STRING, abstract STRING))`
- `CLUSTERED BY name into 4 shards`
- `WITH (number_of_replicas = 1)`

PRIMARY

REPLICA

CrateDB

CLUSTERING / REPLICATION / PARTITIONED TABLES

- `CREATE TABLE buzzwords.speakers (name STRING, talk OBJECT as (title = STRING, abstract = STRING), year INT)`
- `CLUSTERED BY name into 4 shards`
- `PARTITIONED BY (year, …)`
- `WITH (number_of_replicas = 1)`

PRIMARY

REPLICA

CrateDB

MORE FEATURES

- Postgres protocol compatible
- Geo search
- Text Analyzers
- UDFs
- Snapshots
- User management
- Authentication
- Table/Schema/View Privileges
- SSL encryption
- MQTT Ingestion
- Generated Columns
- Views
- Hash Joins
- Web Interface

ARCHITECTURE

CrateDB

CRATEDB TECH STACK

WEB INTERFACE
CLIENTS

SQL QUERY PROCESSING
LUCENE QUERY GENERATION
DISTRIBUTED QUERY EXECUTION
POSTGRES / REST PROTOCOL

TRANSPORT / ROUTING / REPLICATION
SNAPSHOTS
LUCENE REQUESTS

DOCUMENT STORE
INDEXING FIELDS / COLUMN STORAGE
QUERYING DOCUMENT STORES

INTRODUCTION TO Lucene

- Lucene stores documents which are CrateDB's rows
- Documents have fields
-
```
{
  _id   : '123',
  name  : 'Bob',
  title : 'How I Learned to Stop Worrying
           and Love the Bomb',
  price : 23.42
}
```

- Fields are indexed for efficient lookup
- Fields have column store for efficient aggregation

Inverted Index

```
...
BOB → 123, ...
HOW → 123, ...
STOP → 123, ...
...
```

Column Storage

```
NAME → BOB, ALICE, MAX, ...
PRICE → 23.42, 47.21, 38.00, ...
```

INTRODUCTION TO ELASTICSEARCH

- Elasticsearch core concepts revolve around **indices**, **shards**, and **replicas**
- An index is a document store composed of n parts, called shards
- Each shard has 0 or more replicas which hold copies of the shard data
- Replicas are not only useful for fault tolerance but also increase the search performance

HOW TABLES RELATE TO INDICES AND SHARDS

- Each table in CrateDB is represented by an ES index with a mapping

Index Mapping

```
"PROPERTIES":{
  “NAME":{"TYPE":"KEYWORD"},
  "TALKS":{"DYNAMIC":"TRUE",
    "PROPERTIES":{
      "ABSTRACT":{"TYPE":"KEYWORD"},
      "TITLE":{"TYPE":"KEYWORD"}
    }
  }
}
```

- Each partition in a partitioned table is represented by an ES index
  - Partition indices are created by encoding the partition value in the index name

Table mapping

TABLE	T1	T2			T3	…
INDEX	t1	t2.day1	t2.day2	…	t3	…
SHARD1	X	X	X	…	X	…
SHARD2	X	X	X	…	X	…
SHARD3	X				X	…
SHARD4	X					…
…						…

FROM QUERY TO EXECUTION

SELECT name, count(*) as talks FROM buzzwords.speakers
WHERE room = ‘kessel’ AND year = 2018 GROUP BY name ORDER BY name

PARSER	ANALYZER	PLANNER	EXECUTOR
WHAT DIFFERENT PARTS IS THE QUERY COMPOSED OF?	WHAT DO THESE PARTS REFER TO AND WHAT DO THEY MEAN?	HOW CAN WE RETRIEVE THE DESIRED INFORMATION? HOW CAN WE DO THAT EFFICIENTLY?	HOW TO EXECUTE A PLAN AND RECEIVE RESULTS?

FROM QUERY TO EXECUTION

SELECT name, count(*) as talks FROM buzzwords.speakers
WHERE room = ‘kessel’ AND year = 2018 GROUP BY name ORDER BY name

PARSER

WHAT DIFFERENT PARTS IS THE QUERY COMPOSED OF?

ANALYZER

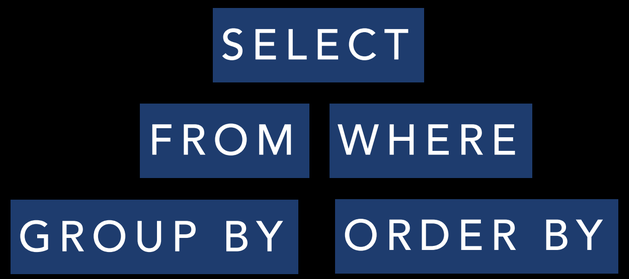
WHAT DO THESE PARTS REFER TO AND WHAT DO THEY MEAN?

PLANNER

HOW CAN WE RETRIEVE THE DESIRED INFORMATION?
HOW CAN WE DO THAT EFFICIENTLY?

EXECUTOR

HOW TO EXECUTE A PLAN AND RECEIVE RESULTS?



FROM QUERY TO EXECUTION

SELECT name, count(*) as talks FROM buzzwords.speakers
WHERE room = ‘kessel’ AND year = 2018 GROUP BY name ORDER BY name

PARSER	ANALYZER	PLANNER	EXECUTOR
WHAT DIFFERENT PARTS IS THE QUERY COMPOSED OF?	WHAT DO THESE PARTS REFER TO AND WHAT DO THEY MEAN?	HOW CAN WE RETRIEVE THE DESIRED INFORMATION? HOW CAN WE DO THAT EFFICIENTLY?	HOW TO EXECUTE A PLAN AND RECEIVE RESULTS?

FROM QUERY TO EXECUTION

SELECT name, count(*) as talks FROM buzzwords.speakers
WHERE room = ‘kessel’ AND year = 2018 GROUP BY name ORDER BY name

PARSER	ANALYZER	PLANNER	EXECUTOR
WHAT DIFFERENT PARTS IS THE QUERY COMPOSED OF?	WHAT DO THESE PARTS REFER TO AND WHAT DO THEY MEAN?	HOW CAN WE RETRIEVE THE DESIRED INFORMATION? HOW CAN WE DO THAT EFFICIENTLY?	HOW TO EXECUTE A PLAN AND RECEIVE RESULTS?

SELECT

Field(name), CountAgg

FROM

table buzzwords.speakers

WHERE

Field(name) = ‘kessel’,
Field(yeaer) = 2018

GROUP BY

Field(name)

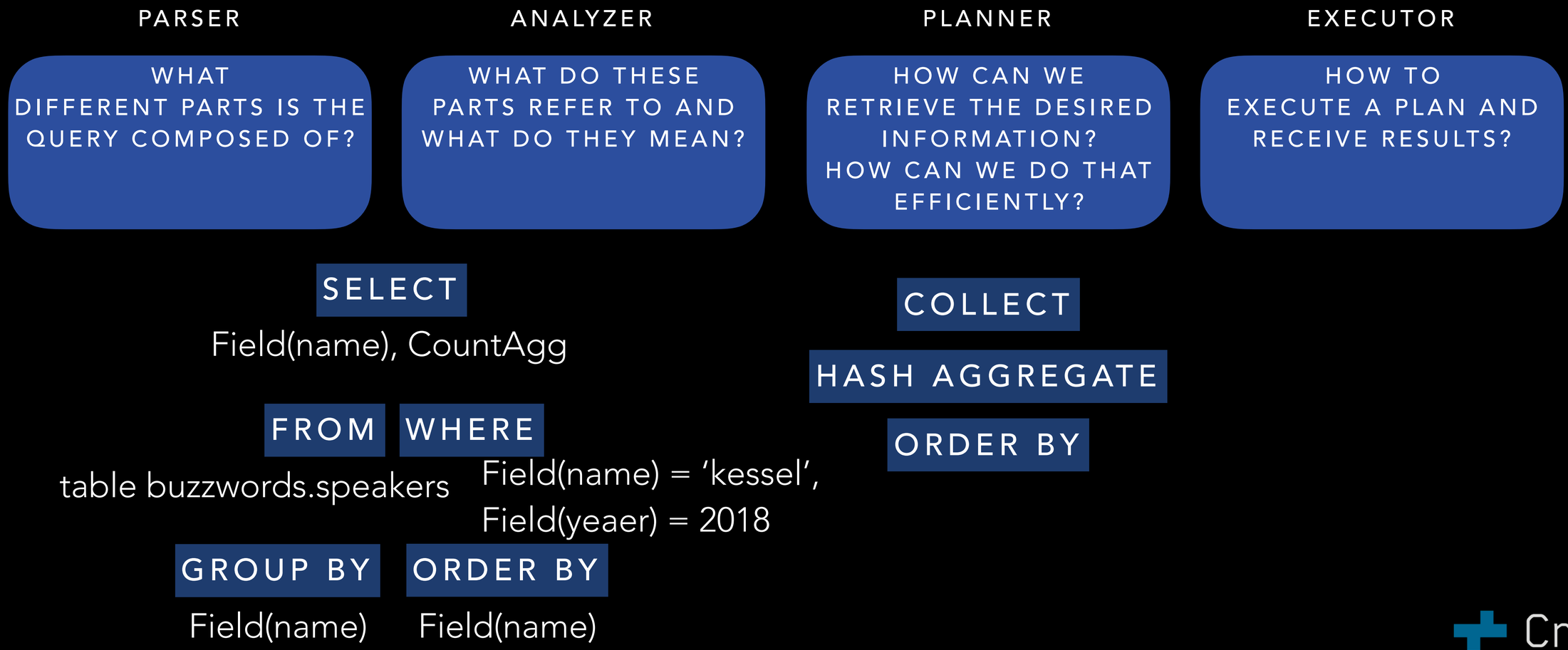
ORDER BY

Field(name)

CrateDB

FROM QUERY TO EXECUTION

SELECT name, count(*) as talks FROM buzzwords.speakers
WHERE room = ‘kessel’ AND year = 2018 GROUP BY name ORDER BY name



CrateDB

FROM QUERY TO EXECUTION

SELECT name, count(*) as talks FROM buzzwords.speakers
WHERE room = ‘kessel’ AND year = 2018 GROUP BY name ORDER BY name

PARSER	ANALYZER	PLANNER	EXECUTOR
WHAT DIFFERENT PARTS IS THE QUERY COMPOSED OF?	WHAT DO THESE PARTS REFER TO AND WHAT DO THEY MEAN?	HOW CAN WE RETRIEVE THE DESIRED INFORMATION? HOW CAN WE DO THAT EFFICIENTLY?	HOW TO EXECUTE A PLAN AND RECEIVE RESULTS?

SELECT
Field(name), CountAgg

FROM table buzzwords.speakers

WHERE
Field(name) = ‘kessel’,
Field(yeaer) = 2018

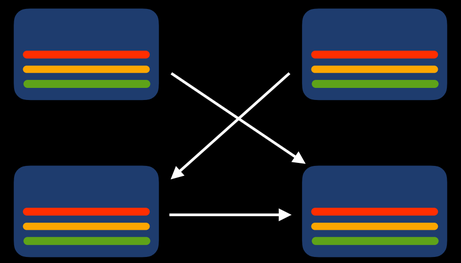
GROUP BY
Field(name)

ORDER BY
Field(name)

COLLECT

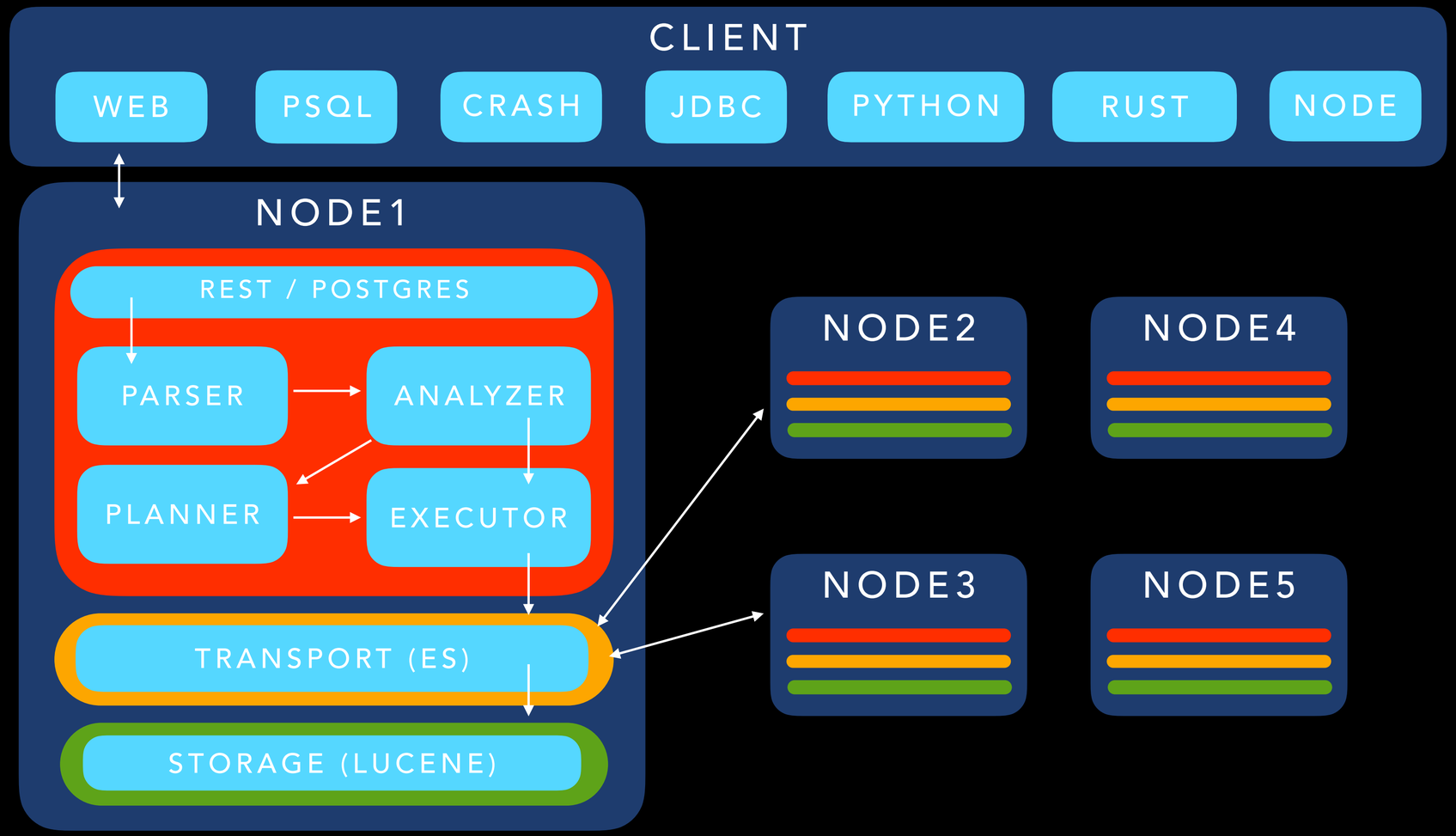
HASH AGGREGATE

ORDER BY



CrateDB

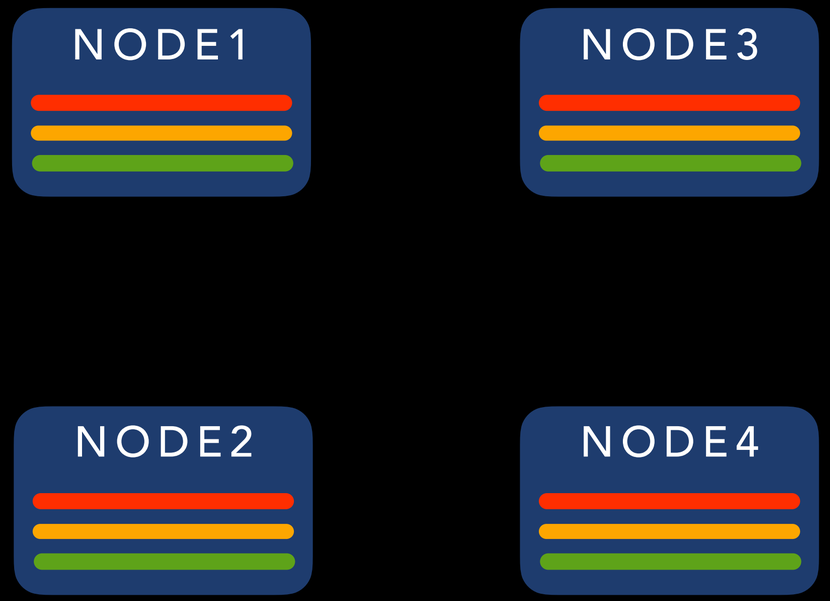
FROM QUERY TO EXECUTION



CrateDB

ARCHITECTURE HIGHLIGHTS

- Distributed storage / Distributed query execution
- Masterless
- Replication
- Only ephemeral storage needed (Container aware)
- Optimized for search: Indexing of all fields with Lucene (tuneable)



HANDS-ON

CrateDB

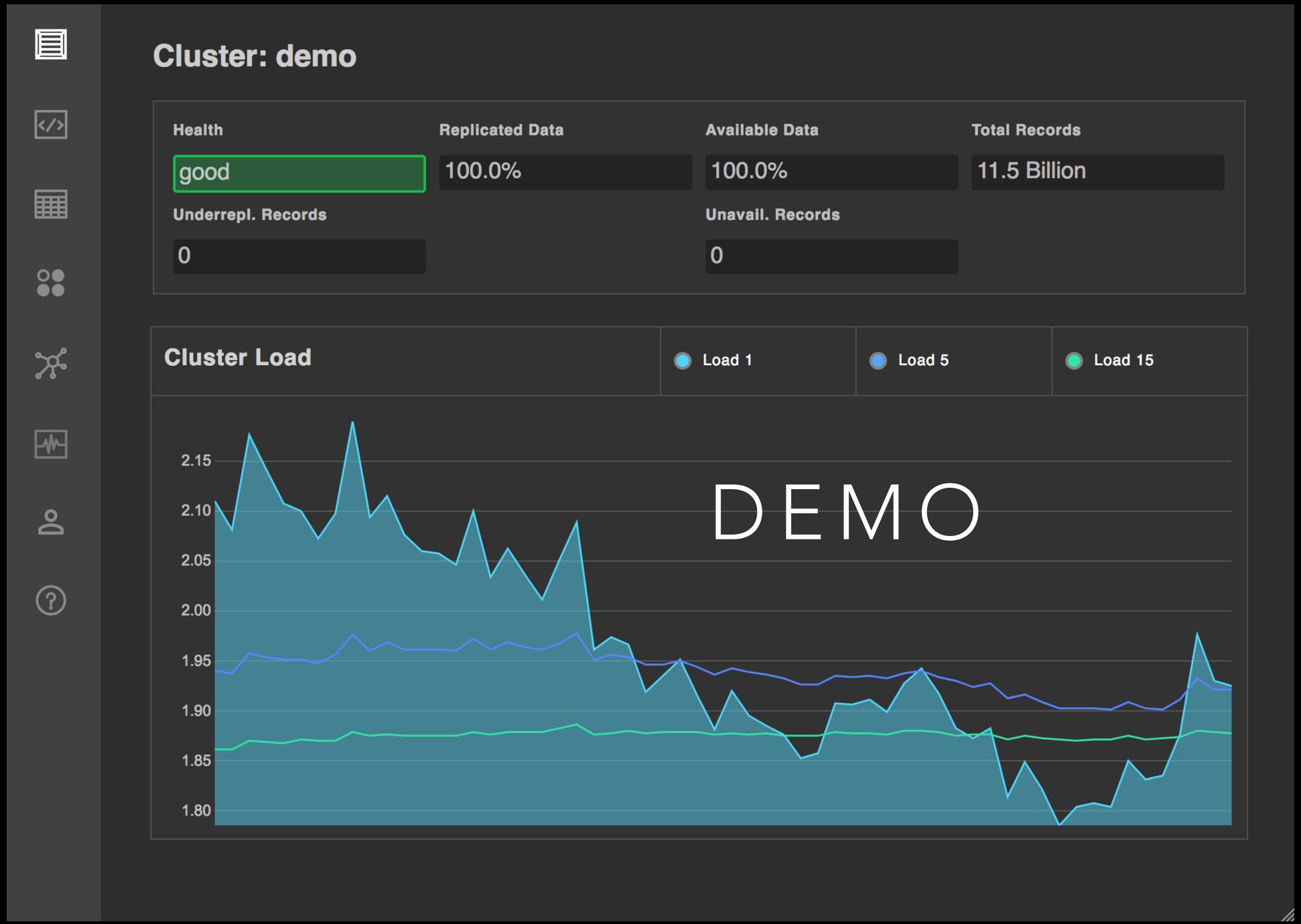
WHAT CAN YOU DO WITH CRATEDB?

- Monitoring with realtime analysis (IoT, Industry 4.0, Cyber Security)
- Data Science
- Stream Analysis
- Text Analysis
- Time Series Analysis
- Geospatial Queries

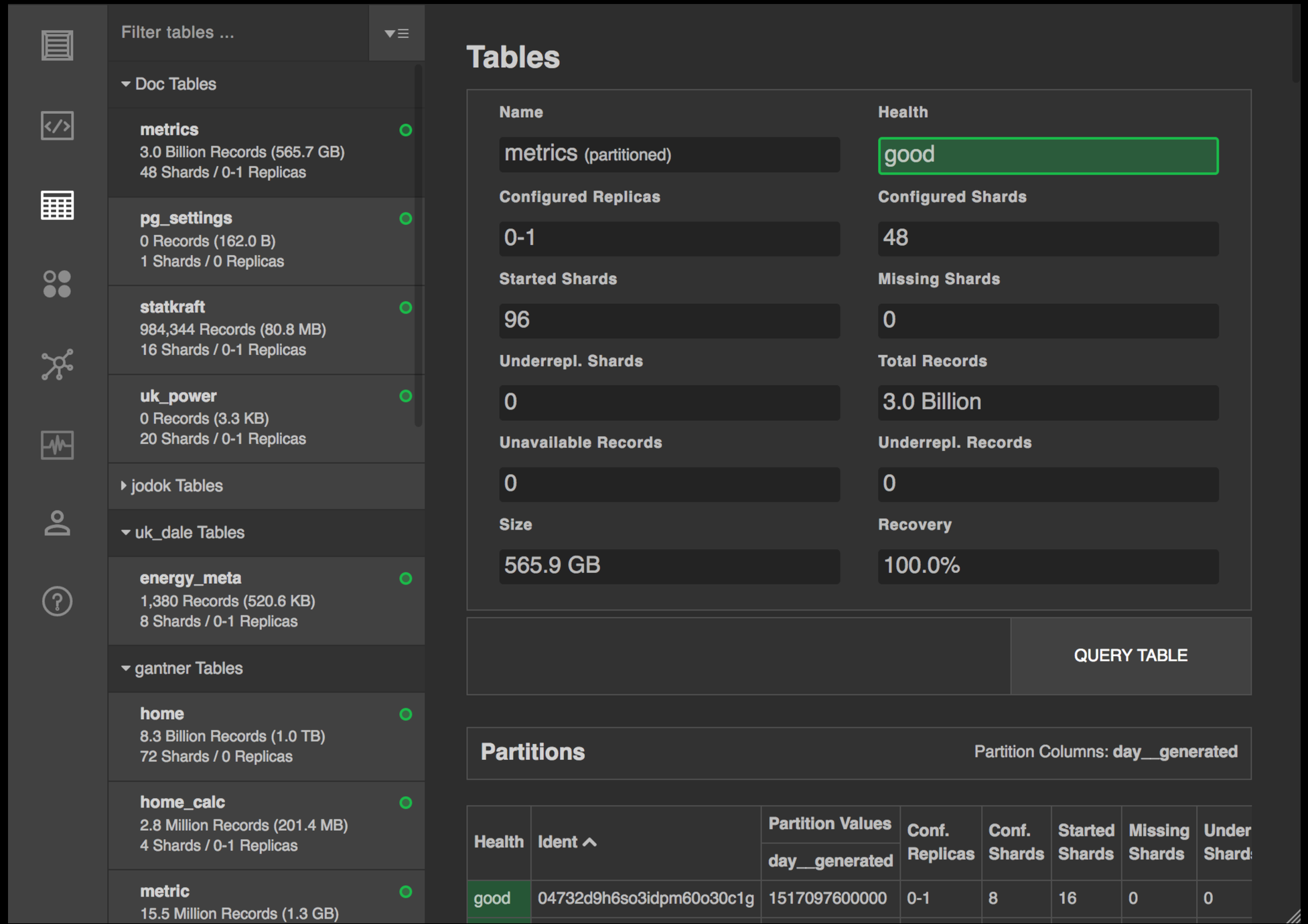
ALPLA

- Plastic bottle manufacturer for food, drinks, cosmetics, cleaning products
- Employs 18,300 employees at 172 locations across 45 countries.
- Real-time insights into the manufacturing process
  - Throughput, failure rates, machine maintenance
  - Lower operational costs

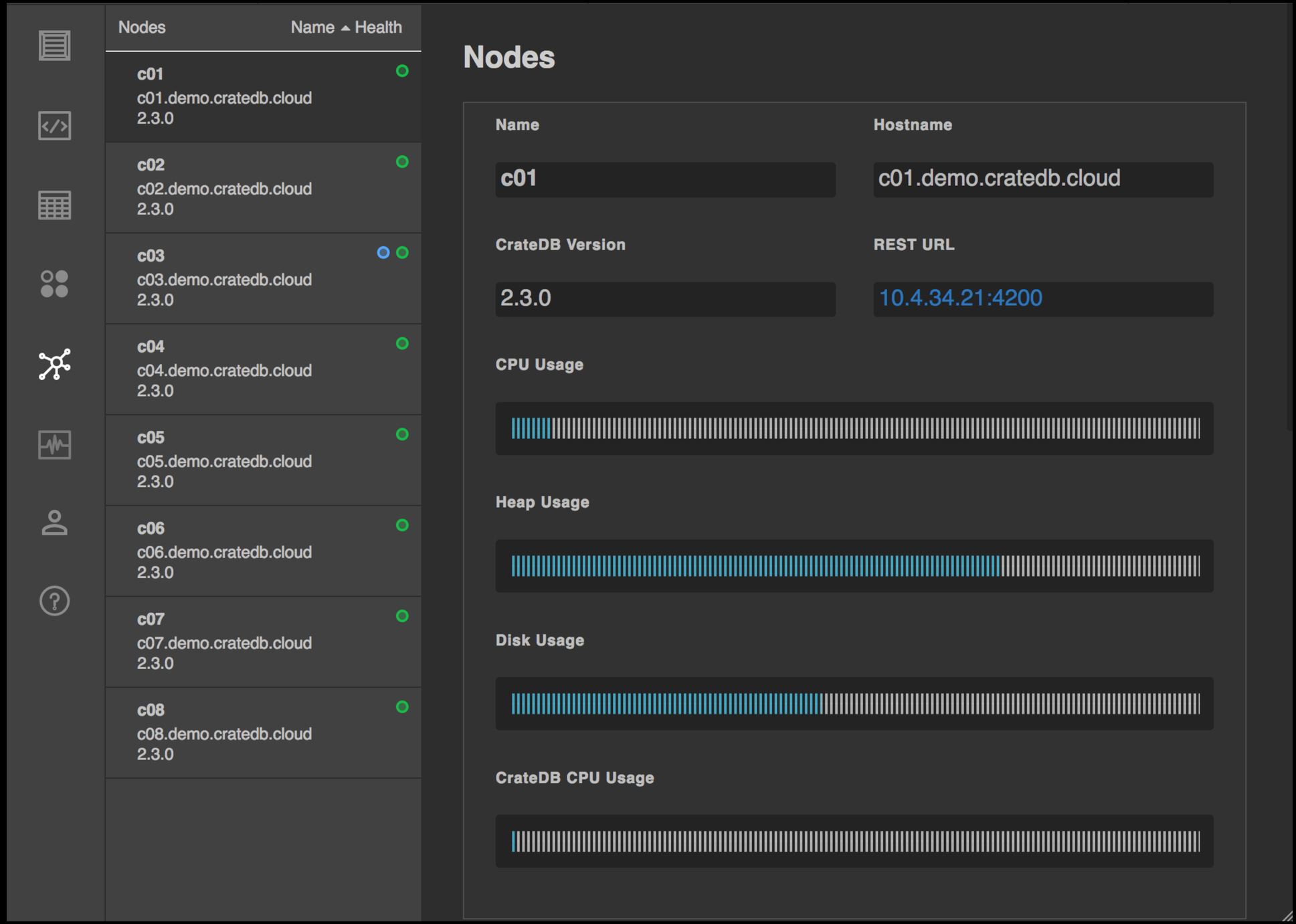
CrateDB Web Interface



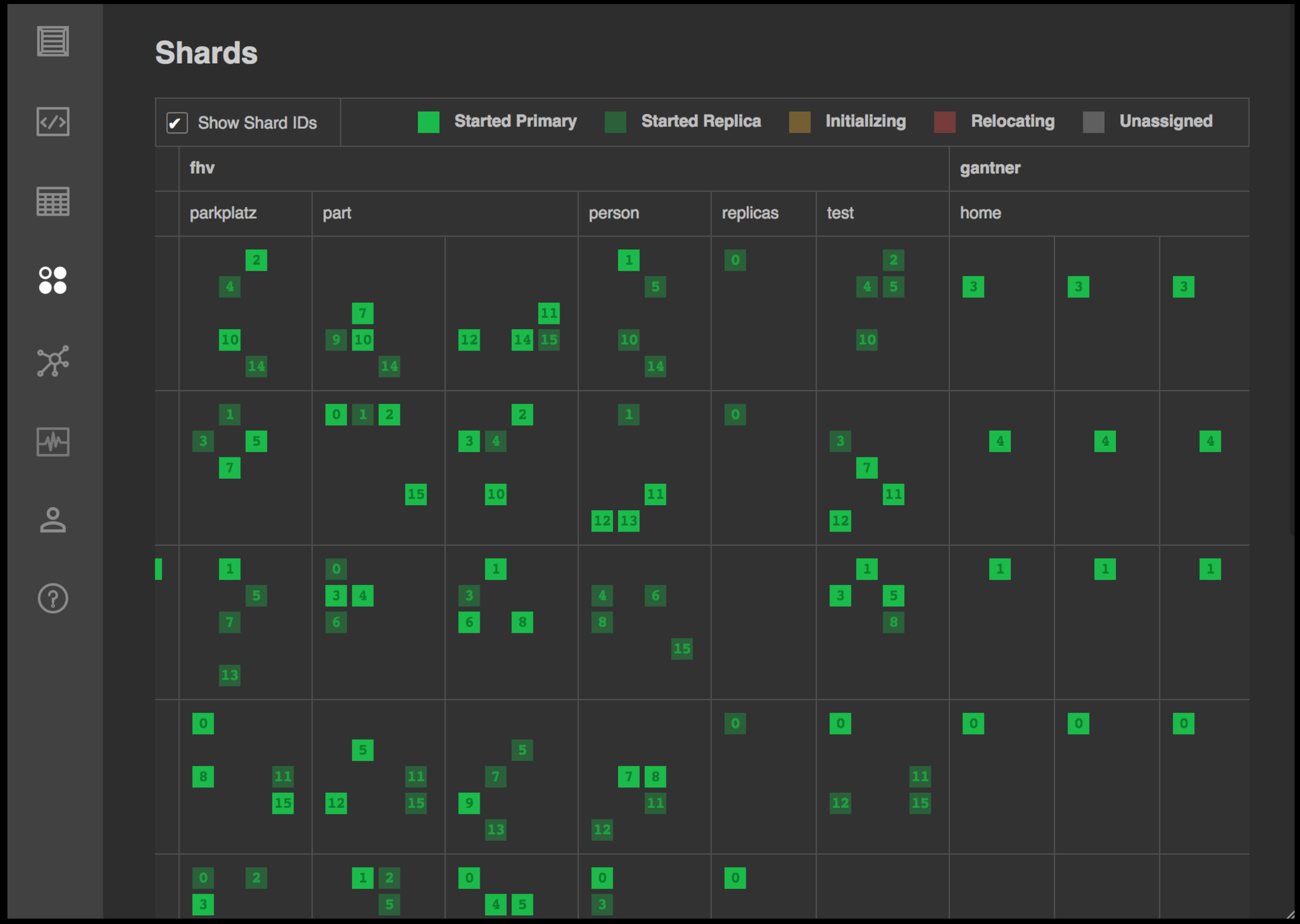
CrateDB Web Interface



CrateDB Web Interface



CrateDB Web Interface



CONCLUSION

CrateDB

WHAT WE HAVE LEARNED

- Elasticsearch is a distributed search engine built on top of Lucene
- CrateDB is a distributed SQL database on top of Elasticsearch/Lucene
- CrateDB is perfect when you
  - want or have to use realtime SQL
  - store large amounts of structured or unstructured data
  - have many thousands of queries per second

SEE FOR YOURSELF!

- Try out CrateDB
  - Download at crate.io/download
  - or `$ curl try.crate.io | bash`
  - or `$ docker run crate`
  - or build from source github.com/crate/crate

- Check out https://crate.io/docs
- Contributions welcome
  - Check out the developer documentation
  - Check out GitHub issues
  - Stackoverflow
  - Join our Slack channel

Developer documentation: https://github.com/crate/crate/blob/master/devs/docs/index.rst
Slack channel: https://crate.io/docs/support/slackin/

CrateDB

THANK YOU!

Maximilian Michels
@stadtlegende
max@crate.io
mxm@apache.org