STARGATE

Apache Cassandra et Kubernetes

cassandra + = K8SSANDRA

DataStax Developers

Cédrick Lunven

Director Developer Relations

@clunven

@clun

- Trainer
- Public Speaker
- Developers Support
- Developer Applications
- Developer Tooling

cassandra

SDK

Java

FF4J

- Creator of ff4j (ff4j.org)
- Maintainer for 8 years+

{ REST }

GRPC

- Happy developer for 14 years
- Spring Petclinic Reactive & Starters
- Implementing APIs for 8 years

Capgemini

DataStax Developers

DataStax Developers Advocates Team

Cedrick Lunven

Aleksandr Volochnev

Jack Fryer

Kirsten Hunter

Stefano Lottini

David Gilardi

Ryan Welford

Rags Srinivas

Sonia Siganporia

R

S

Capgemini

DataStax Developers

Agenda (6H)

01 

Apache Cassandra™
Les fondamentaux

02 

Apache Cassandra™
Modélisation

03 

Apache Cassandra™
Développements

04 

Kubernetes
Les fondamentaux

05 

Kubernetes
Stateful & Cassandra

05 

K8ssandra
Tour d'horizon

DataStax

Formation CapGemini 5 octobre

I – Apache Cassandra™ Les fondamentaux

1. Introduction
2. Scalability
3. Distributed Data
4. Consistency
5. Use Cases
6. CQL

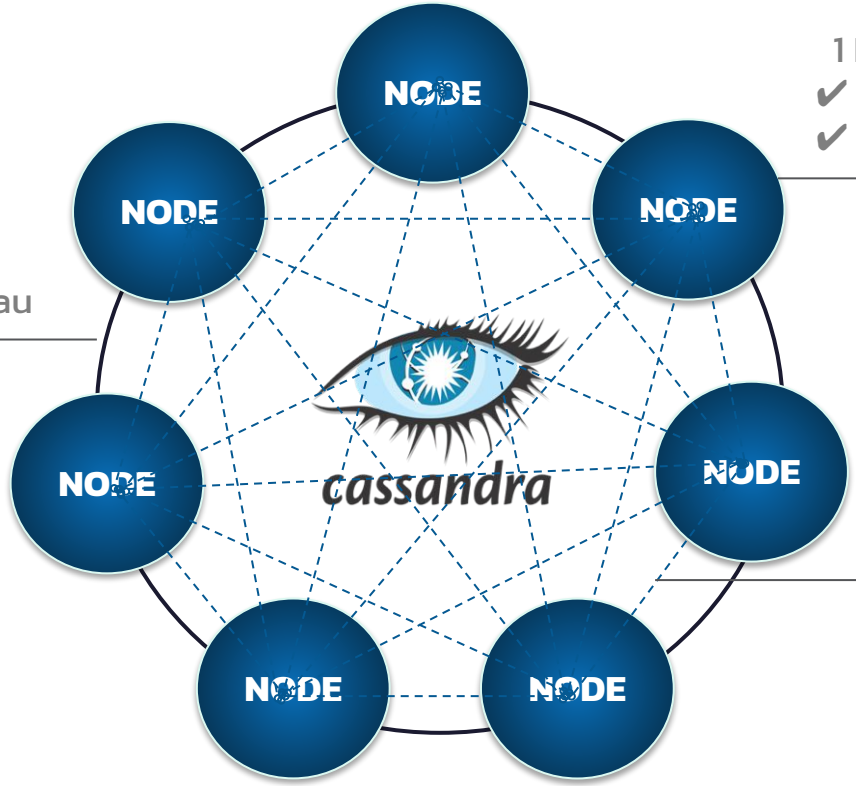
DataStax

Formation CapGemini 5 octobre

I – Apache Cassandra™ Les fondamentaux

1. **Introduction**
2. Scalability
3. Distributed Data
4. Consistency
5. Use Cases
6. CQL

Apache Cassandra™ = Base de données NoSql Distribuée



1 Installation = 1 NOEUD
- ✔ Capacité = ~ 2-4To
- ✔ Performance =4 Tx/sec/core

DataCenter | Anneau

Communication
- ✔ Gossiping
- ✔ Masterless

DataStax

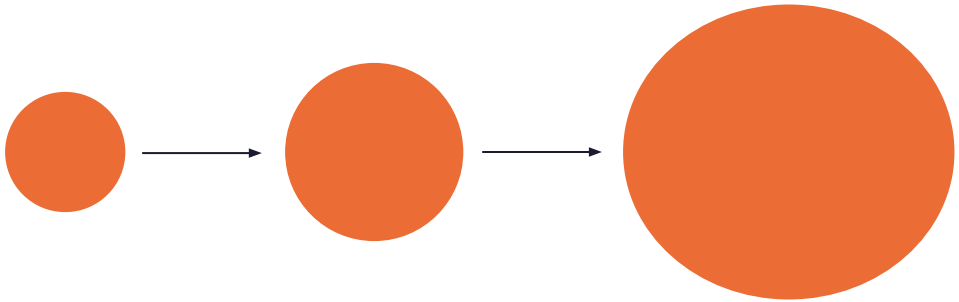
Formation CapGemini 5 octobre

I – Apache Cassandra™ Les fondamentaux

1. Introduction
2. **Scalability**
3. Distributed Data
4. Consistency
5. Use Cases
6. CQL

Vertical Scalability?

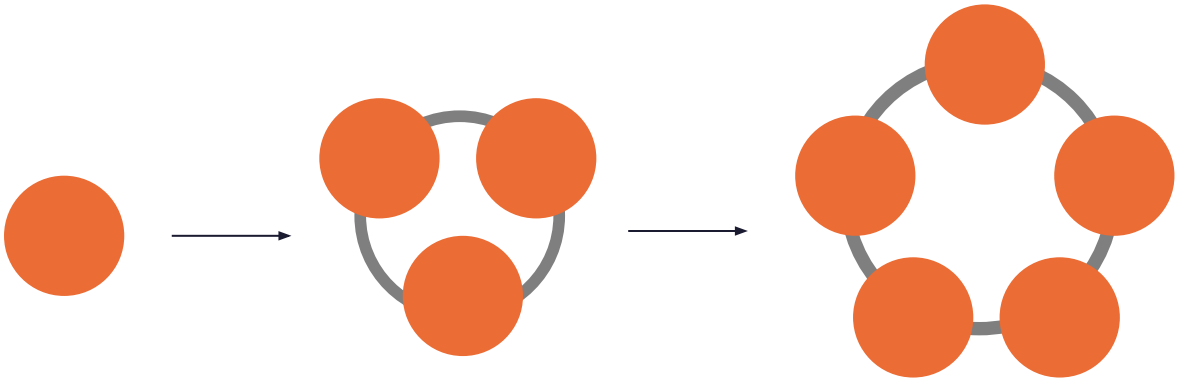
Vertical scaling requires one large expensive machine



+ Easier to get
- Expensive
- Still SPoF

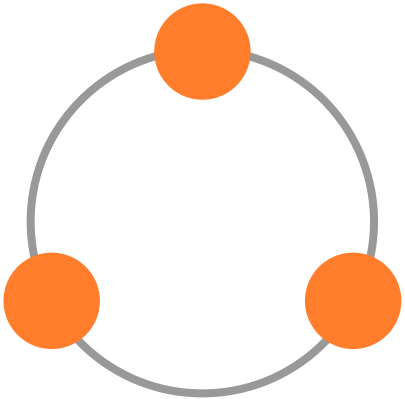
Horizontal Scalability!

Horizontal scaling requires multiple less-expensive commodity hardware

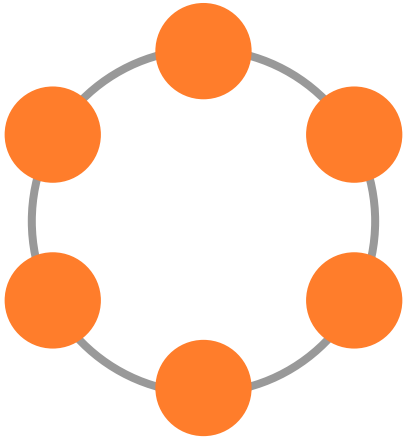


\+ Cheaper
\+ Removes SPoF
\- Logistically harder

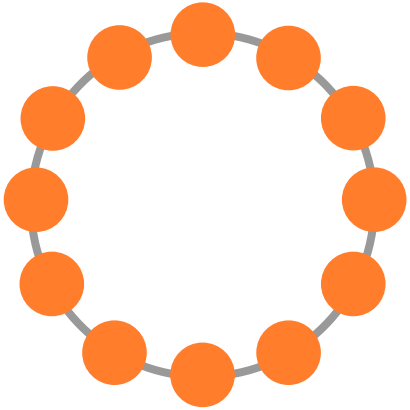
Cassandra Scales Horizontally



100,000
transactions/second

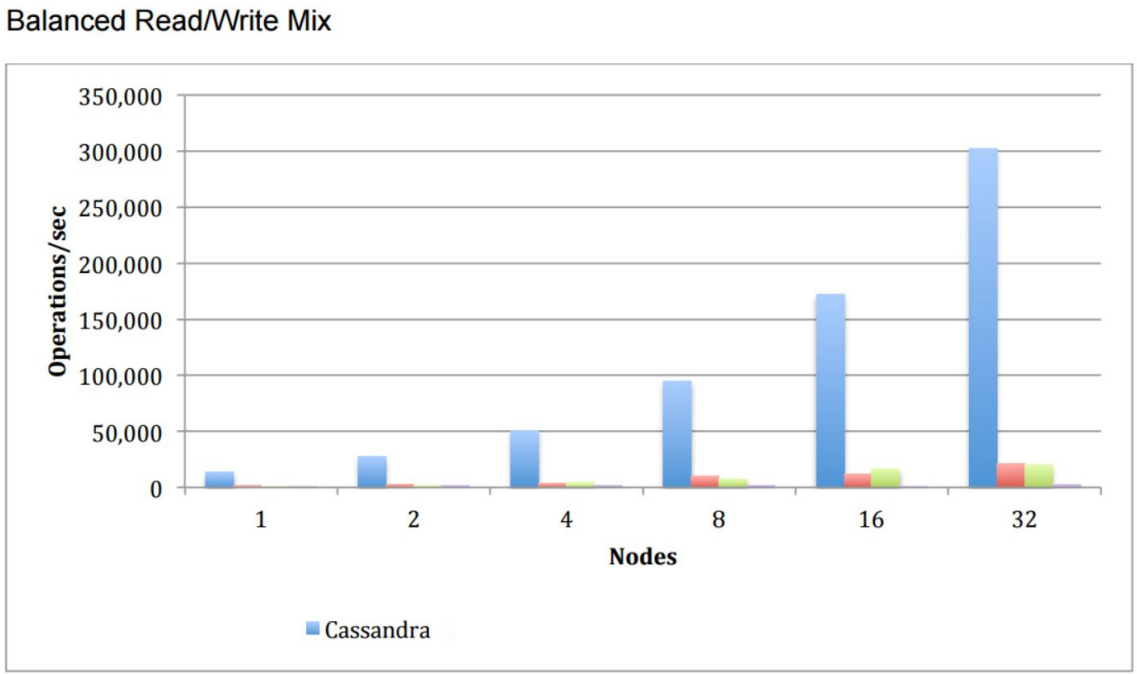


200,000
transactions/second



400,000
transactions/second

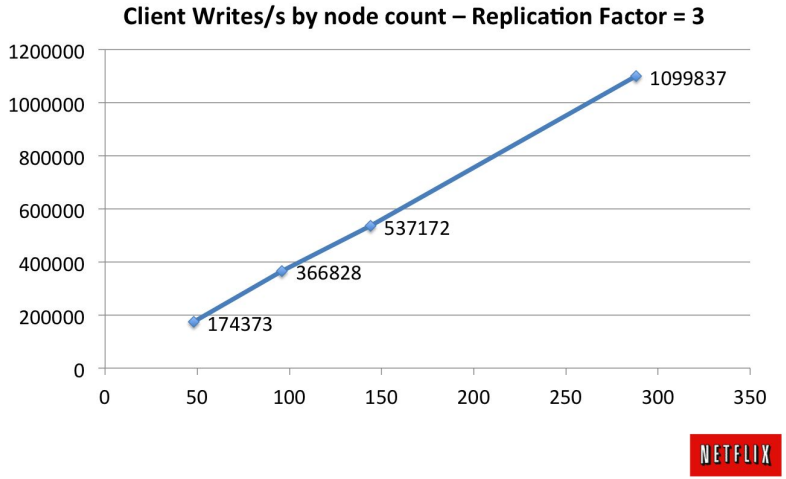
Apache Cassandra™ = Scalabilité Linéaire



Cassandra Scales Linearly

Recommended reading: netflixtechblog.com/benchmarking-cassandra-scalability-on-aws-over-a-million-writes-per-second-39f45f066c9e

Scale-Up Linearity



NETFLIX

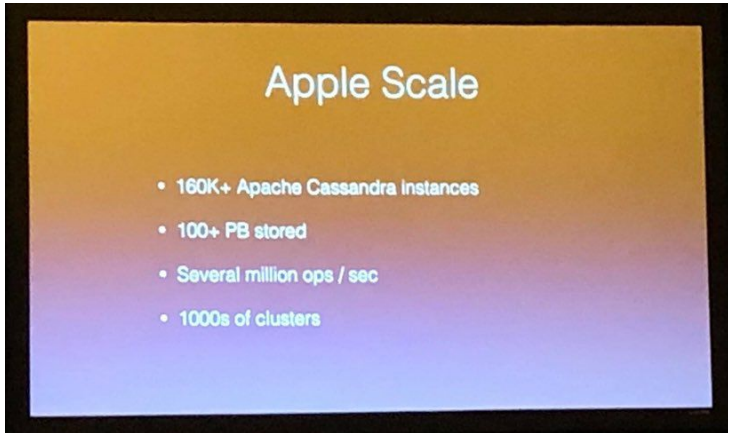
Apache Cassandra™ = Scalabilité inégalée

Apache Cassandra @ Netflix

- 98% of streaming data is stored in Apache Cassandra
- Data ranges from customer details to viewing history to billing and payments
- Foundational datastore for serving millions of operations per second

- 30 million ops/sec on most active single cluster
- 500 TB most dense single cluster
- 9216 CPUs in biggest cluster

O(100) Clusters
O(10000) Instances
O(10,000,000) Replications per second
O(100,000,000) Operations per second
O(1,000,000,000,000,000) Petabytes of data



https://www.youtube.com/watch?v=BODvXsQYyaY

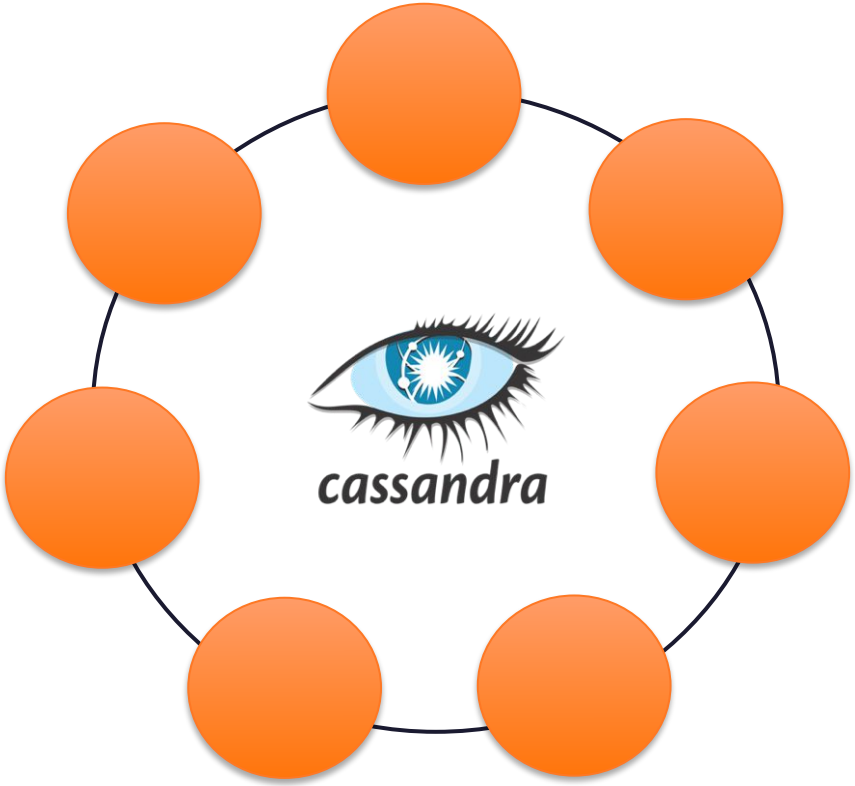
DataStax

Formation CapGemini 5 octobre

I – Apache Cassandra™ Les fondamentaux

1. Introduction
2. Scalability
3. **Distributed Data**
4. Consistency
5. Use Cases
6. CQL

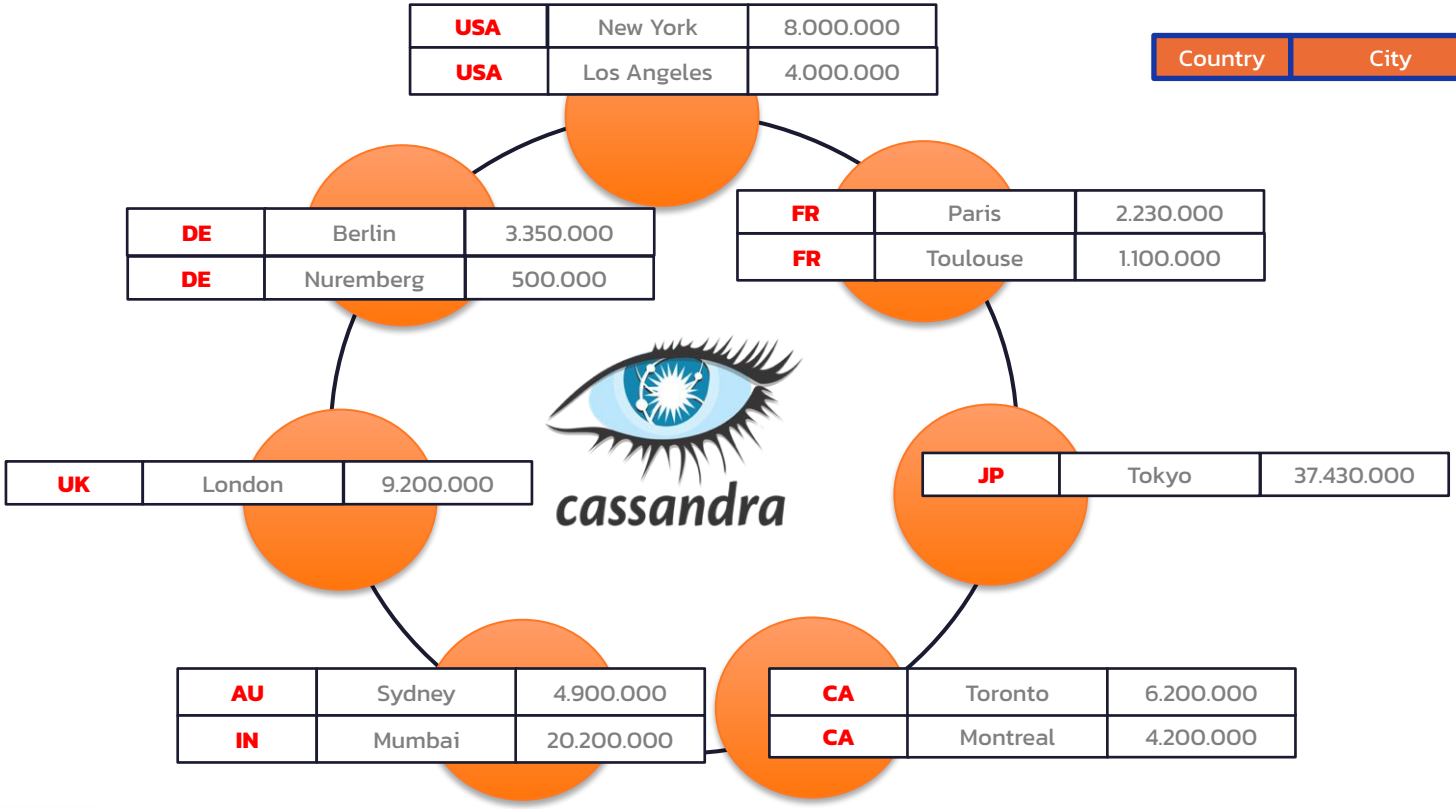
Data is Distributed



Country	City	Population
USA	New York	8.000.000
USA	Los Angeles	4.000.000
FR	Paris	2.230.000
DE	Berlin	3.350.000
UK	London	9.200.000
AU	Sydney	4.900.000
DE	Nuremberg	500.000
CA	Toronto	6.200.000
CA	Montreal	4.200.000
FR	Toulouse	1.100.000
JP	Tokyo	37.430.000
IN	Mumbai	20.200.000

Partition Key

Data is Distributed



Country	City	Population

Partitioning

CO	City	Population
AU	Sydney	4.900.000
CA	Toronto	6.200.000
CA	Montreal	4.200.000
DE	Berlin	3.350.000
DE	Nuremberg	500.000

Partition Key

Partitioner

Murmur3 Hashing

CO	City	Population
59	Sydney	4.900.000
12	Toronto	6.200.000
12	Montreal	4.200.000
45	Berlin	3.350.000
45	Nuremberg	500.000

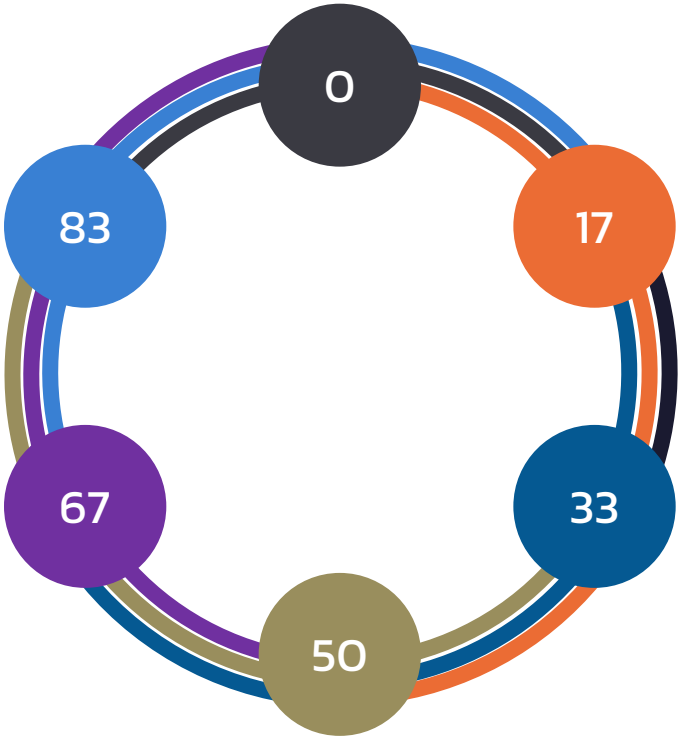
Tokens

Cassandra Nodes

Data is Replicated

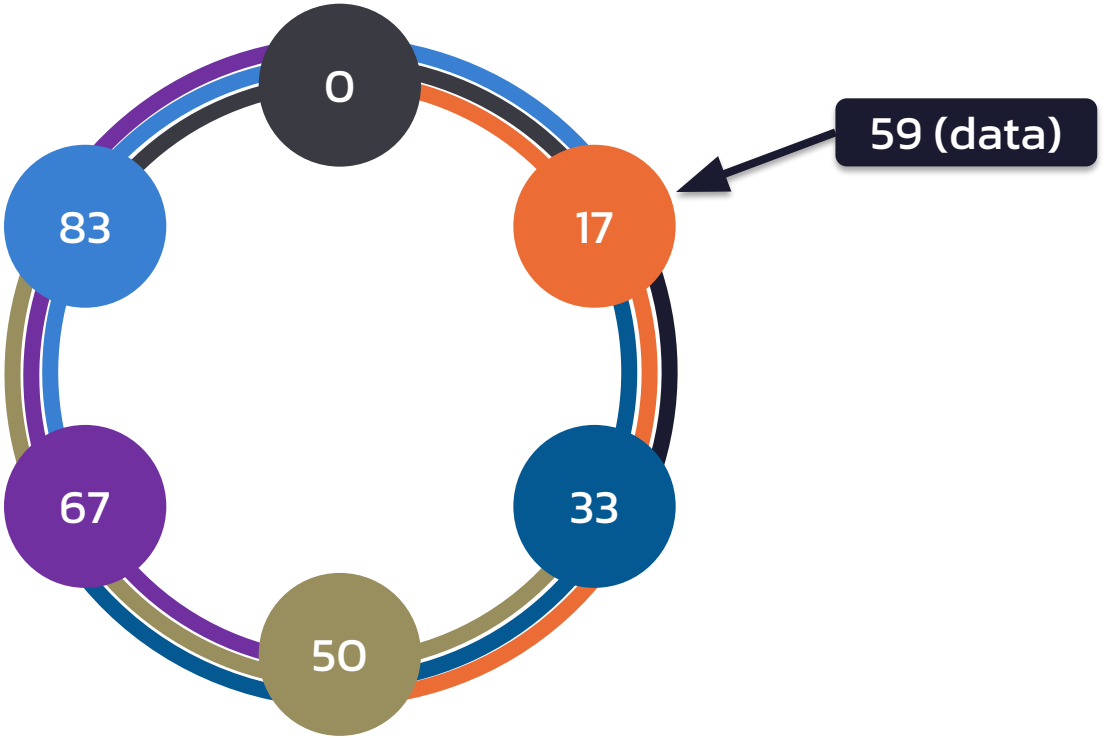
RF = 3

Replication Factor 3 means that every row is stored on 3 different nodes



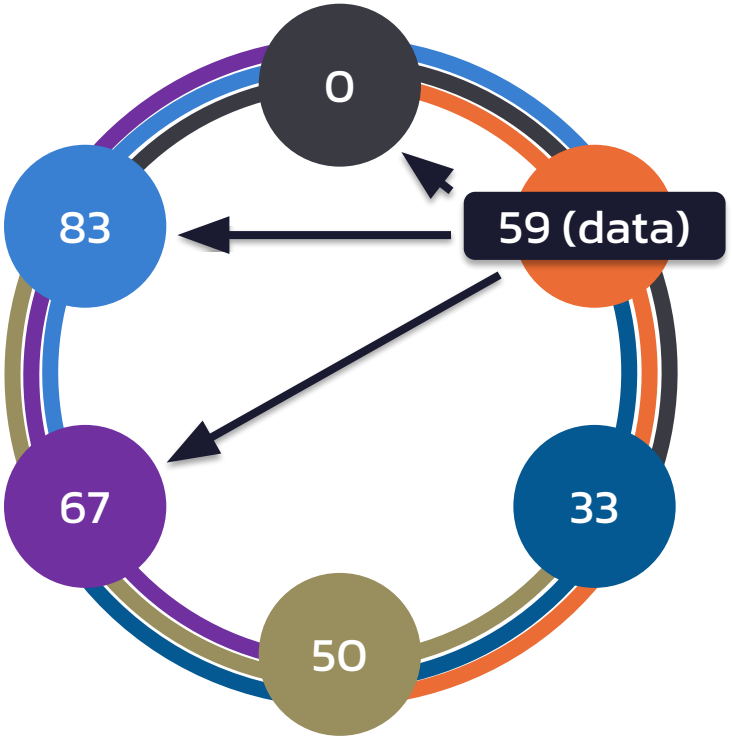
Replication within the Ring

RF = 3



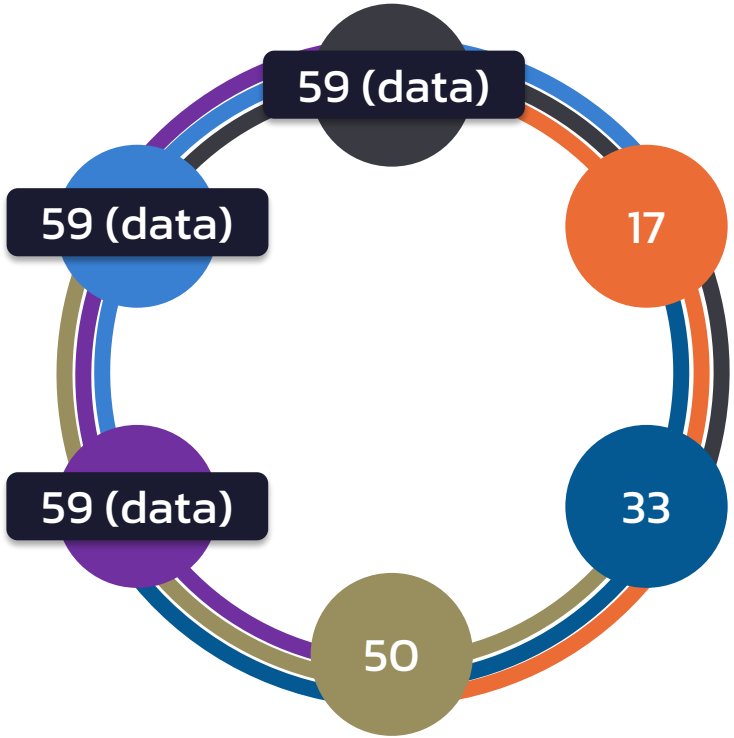
Replication within the Ring

RF = 3



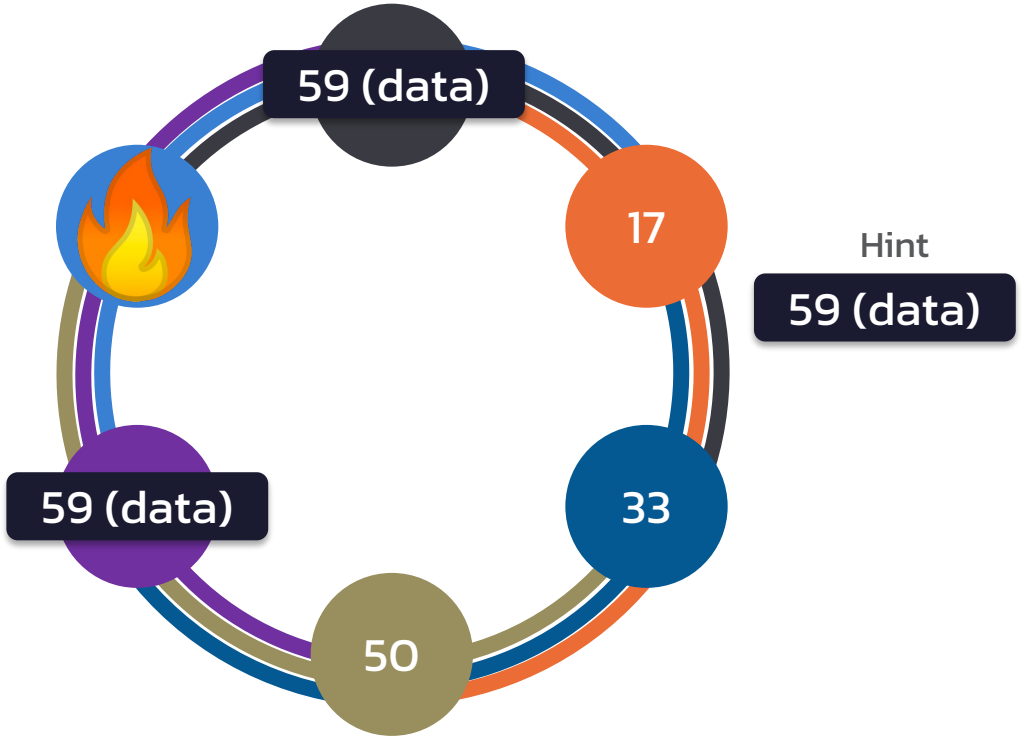
Replication within the Ring

RF = 3



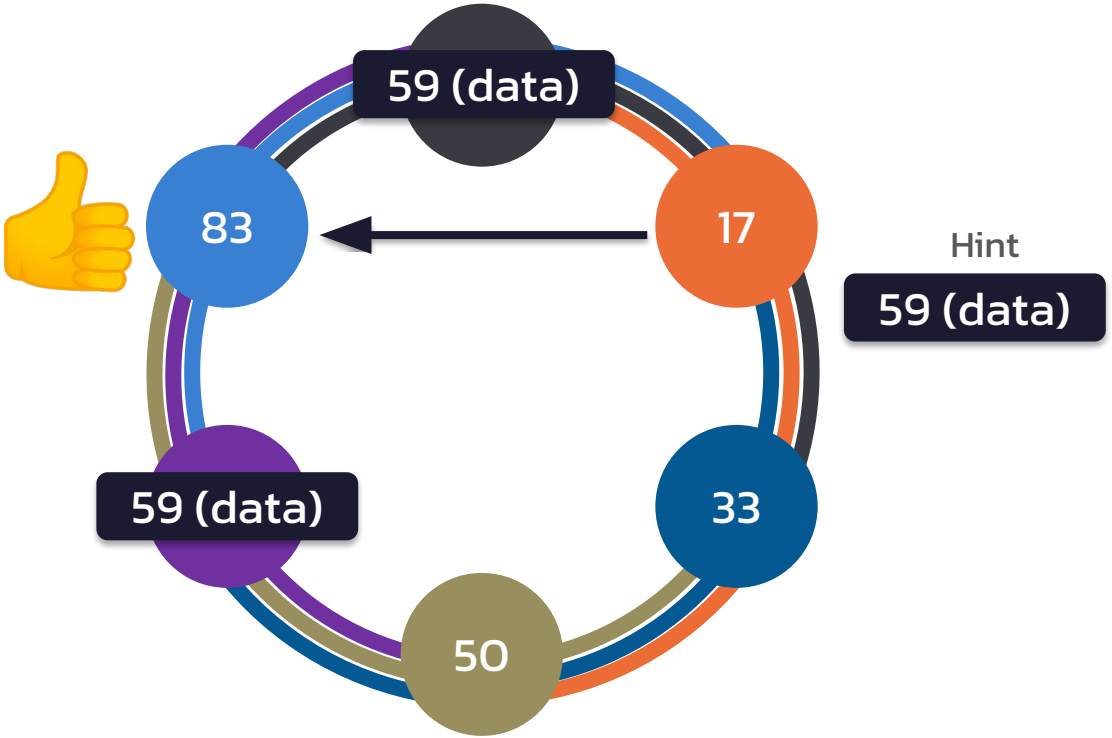
Node Failure

RF = 3



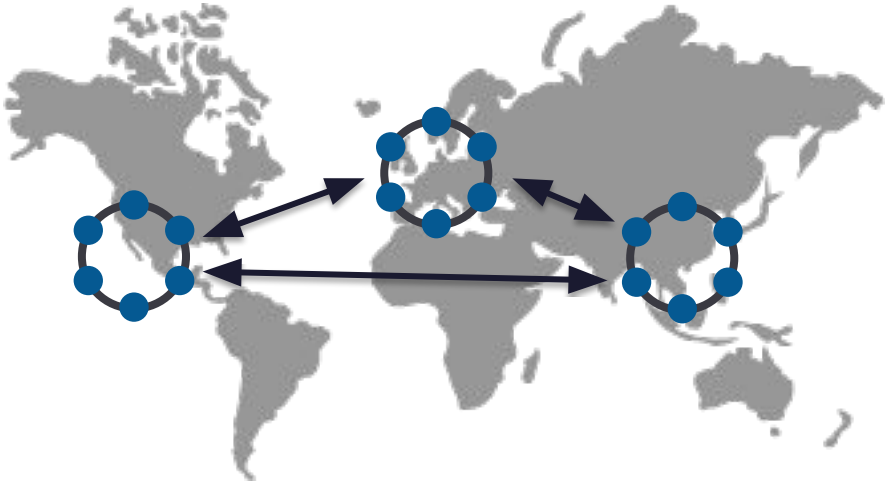
Node Failure Recovered

RF = 3

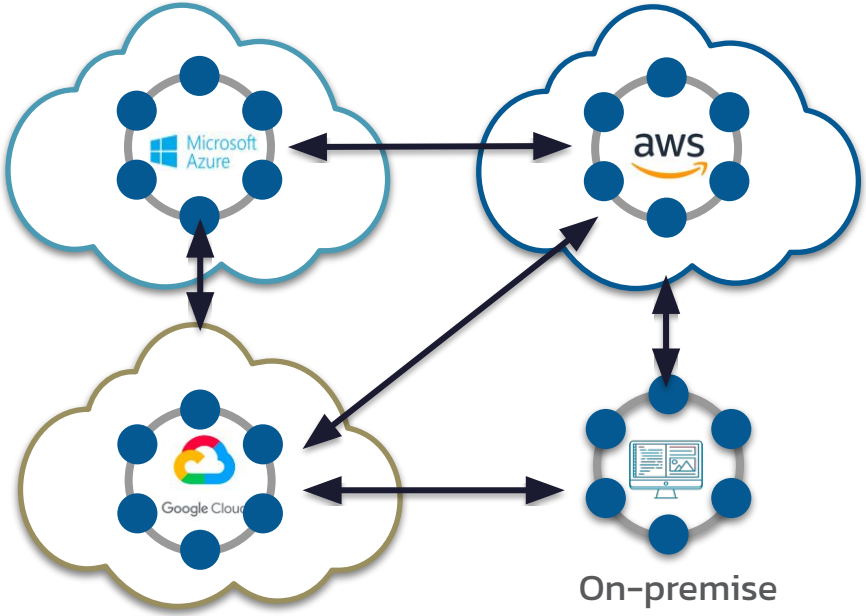


Data Distributed Everywhere

- Geographic Distribution

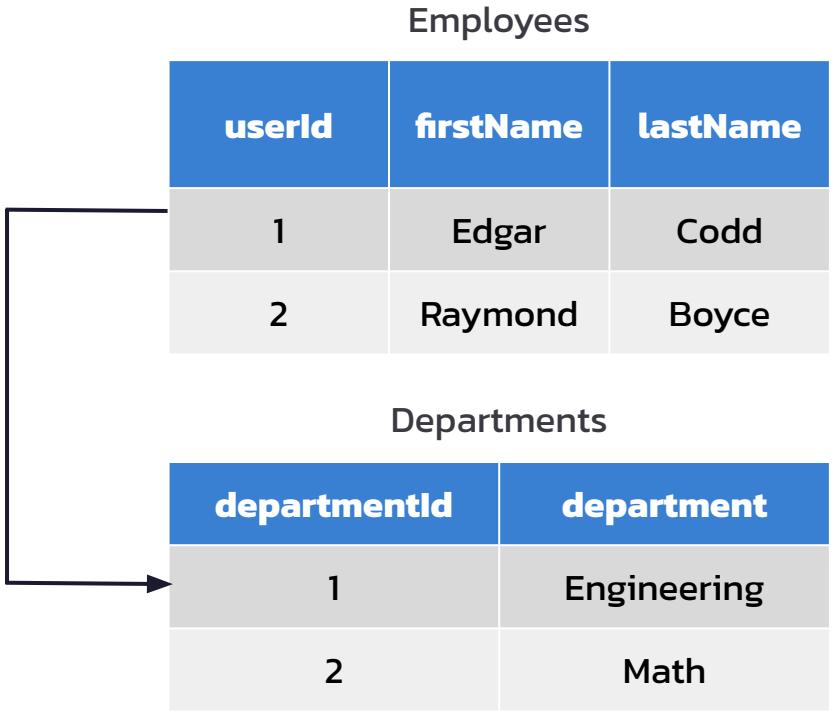


- Hybrid-Cloud and Multi-Cloud



Normalization

"Database normalization is the process of structuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity. It was first proposed by Edgar F. Codd as part of his relational model."



PROS: Simple write, Data Integrity
CONS: Slow read, Complex Queries

Denormalization

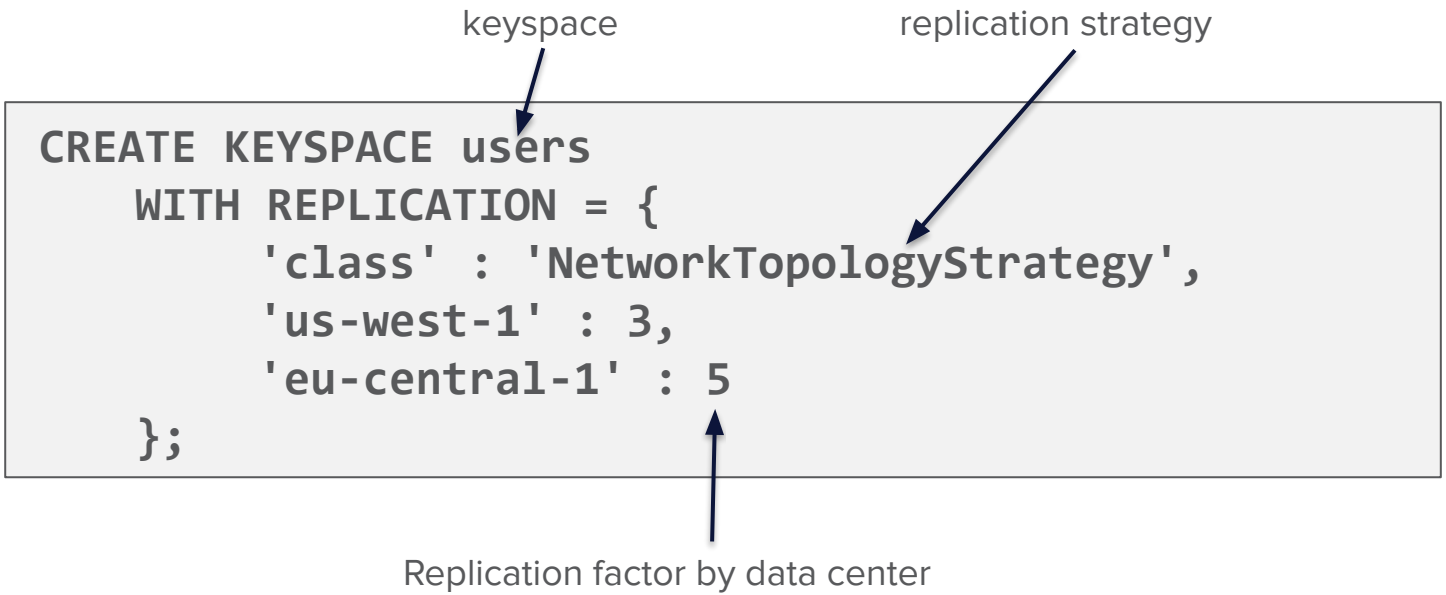
"Denormalization is a strategy used on a database to increase performance. In computing, denormalization is the process of trying to improve the read performance of a database, at the expense of losing some write performance, by adding redundant copies of data"

PROS: Quick Read, Simple Queries
CONS: Multiple Writes, Manual Integrity

Employees

userId	firstName	lastName	department
1	Edgar	Codd	Engineering
2	Raymond	Boyce	Math
3	Sage	Lahja	Math
4	Juniper	Jones	Botany

Keyspaces

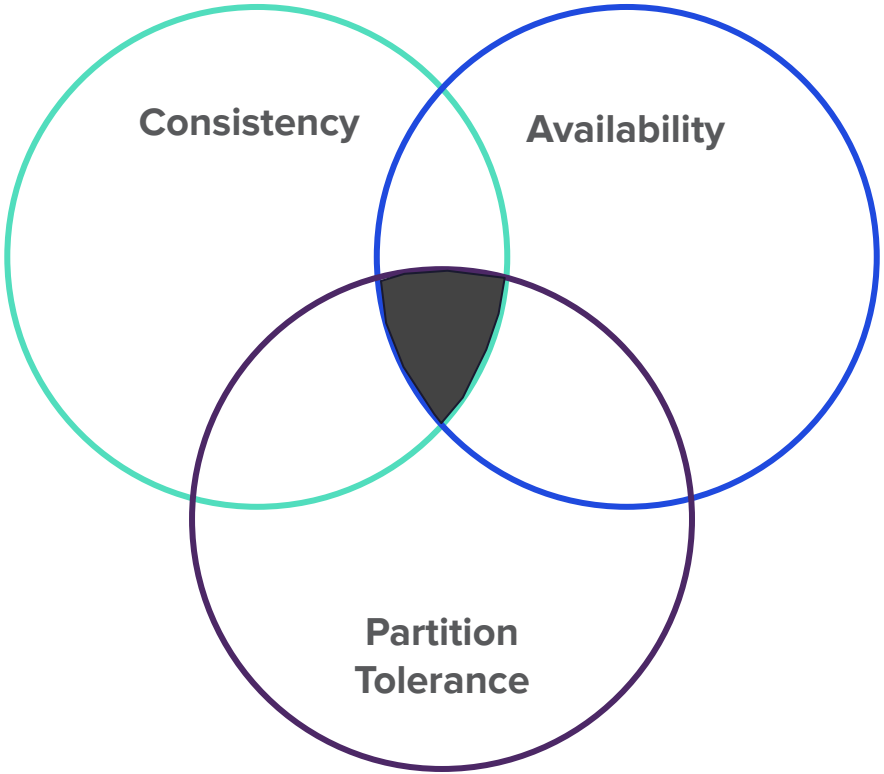

keyspace
replication strategy
CREATE KEYSPACE users
WITH REPLICATION = {
'class' : 'NetworkTopologyStrategy',
'us-west-1' : 3,
'eu-central-1' : 5
};
Replication factor by data center

I – Apache Cassandra™ Les fondamentaux

1. Introduction
2. Scalability
3. Distributed Data
4. **Consistency**
5. Use Cases
6. CQL

CAP Theorem

In the distributed environment, you can have only two guaranteed qualities out of three :(

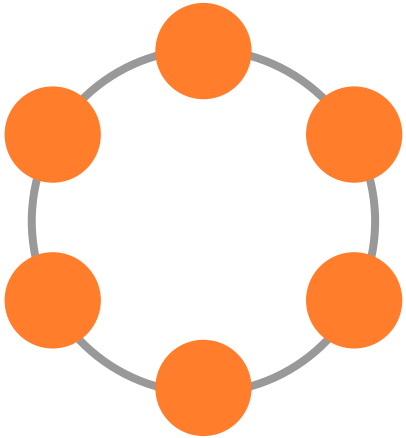


Is Cassandra AP or CP?

Cassandra is configurably consistent. In any moment, of the time, for any particular query you can set your the Consistency Level you require to have.

Cassandra Consistency Levels:

- ANY
- ONE
- TWO, THREE
- QUORUM
- ALL

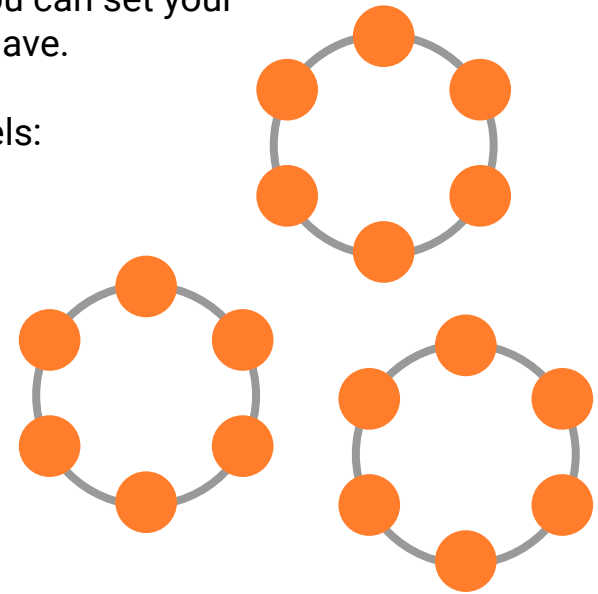


Is Cassandra AP or CP?

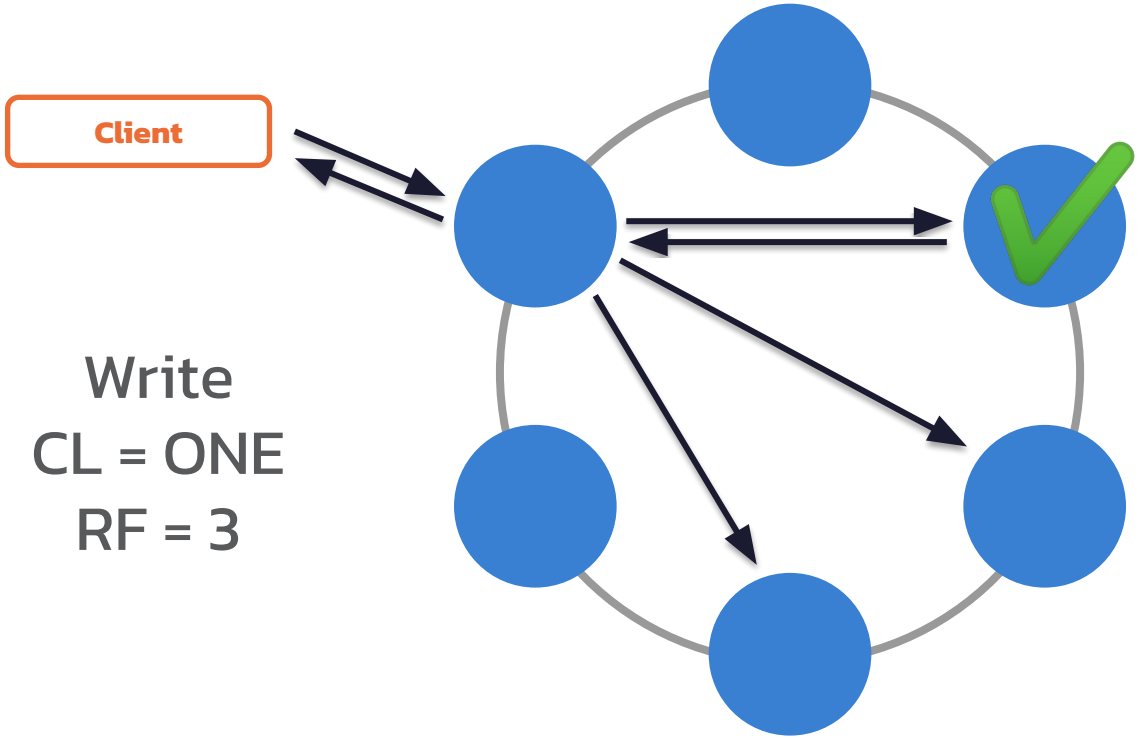
Cassandra is configurably consistent. In any moment, of the time, for any particular query you can set your the Consistency Level you require to have.

Cassandra Multi-DC Consistency Levels:

- LOCAL_ONE
- LOCAL_QUORUM
- EACH_QUORUM

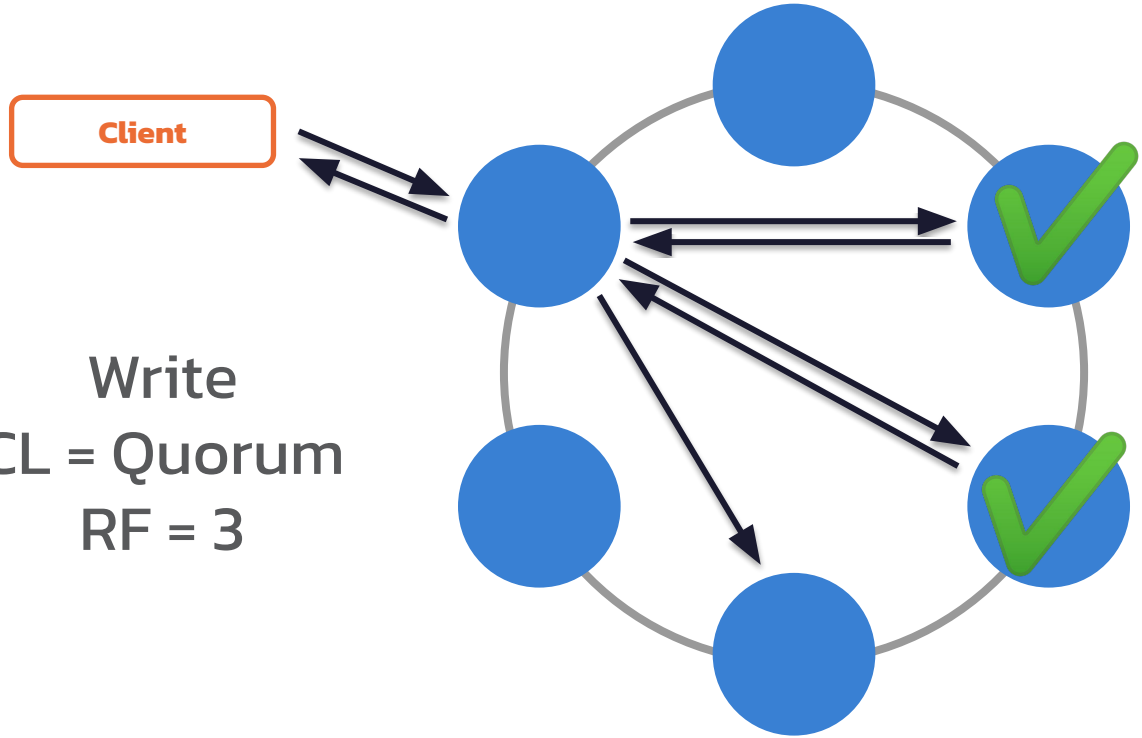


Consistency Level One



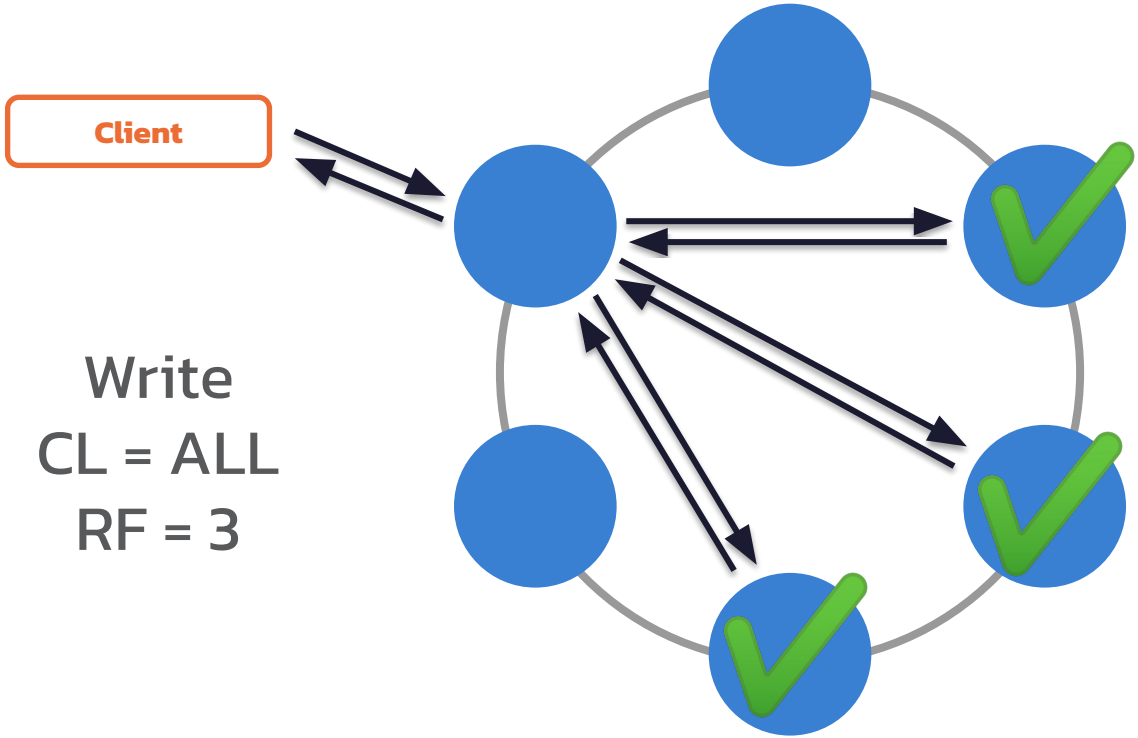
Write
CL = ONE
RF = 3

Consistency Level Quorum



Write
CL = Quorum
RF = 3

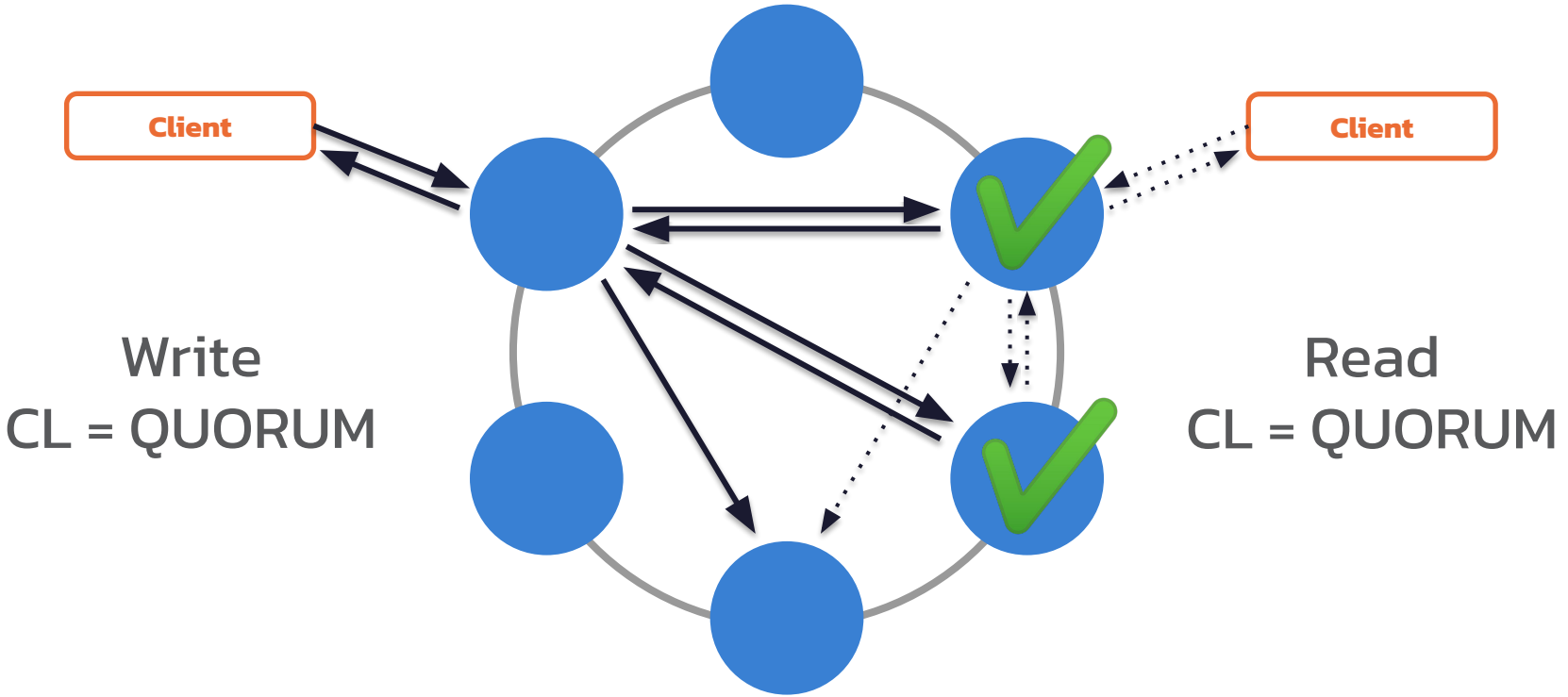
Consistency Level ALL



Write
CL = ALL
RF = 3

Consistency Level ALL

Immediate Consistency – The Right Way



Immediate Consistency – The Right Way

CL Write + CL Read > RF ← Immediate Consistency

I – Apache Cassandra™ Les fondamentaux

1. Introduction
2. Scalability
3. Distributed Data
4. Consistency
5. **Use Cases**
6. CQL

Apache Cassandra™

Open Source
Apache Licence

Active Everywhere
Zero Downtime

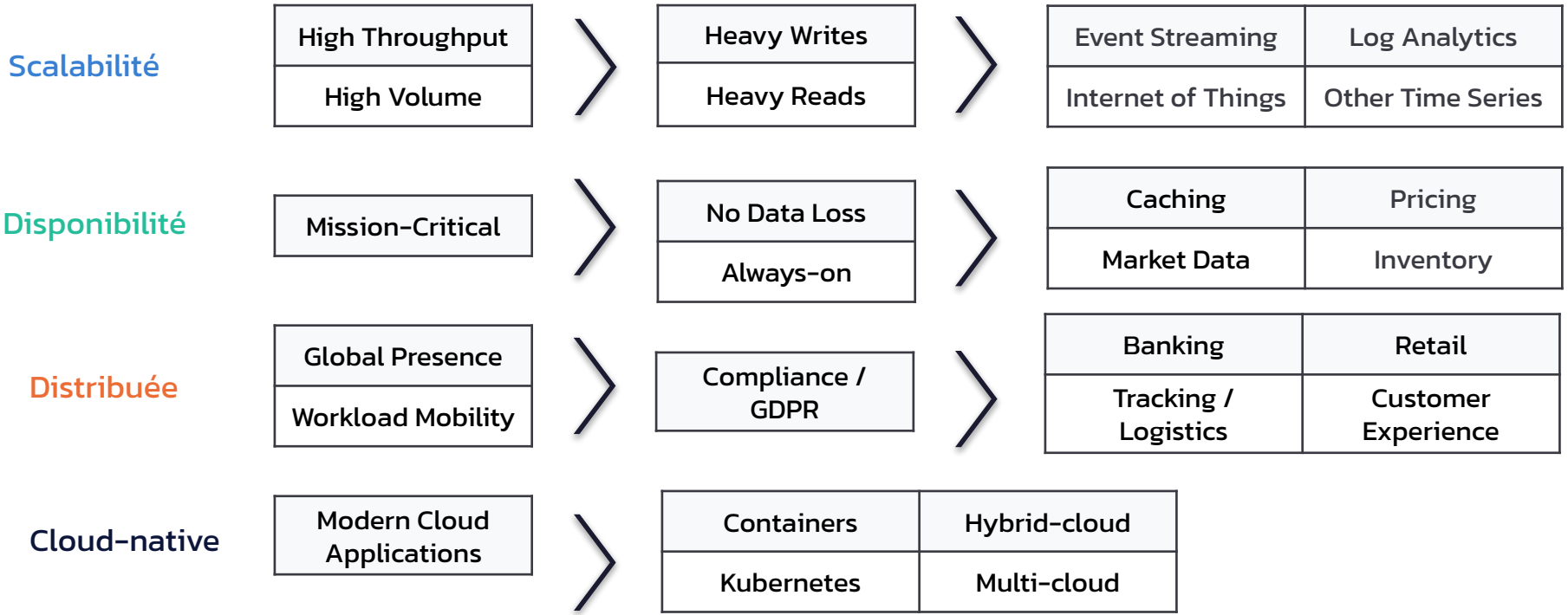
Flexible, Familiar interface
CQL

Scalability
Horizontal

High Performance
Masterless, Replication

Seamless Replication
Asynchronous

Cas d'usages Apache Cassandra™ :



Hands-on

https://www.datastax.com/learn/cassandra-fundamentals#coursework

Introduction to Apache Cassandra™

Learn about Cassandra benefits and install your first Cassandra NoSQL database instance

START CODING

Cassandra Query Language

Learn about the most essential data definition and data manipulation statements in Cassandra Query Language (CQL)

START CODING

Apache Cassandra™ Keyspaces and Data Replication Strategies

Learn how to create a keyspace and select a data replication strategy for a Cassandra NoSQL database

START CODING

I – Apache Cassandra™ Les fondamentaux

1. Introduction
2. Scalability
3. Distributed Data
4. Consistency
5. Use Cases
6. **CQL**

Data Structure: a Cell

An intersection of a row and a column, stores data.

Data Structure: a Row

A single, structured data item in a table.

1	John	Doe	Wizardry

Data Structure: a Partition

ID	First Name	Last Name	Department
1	John	Doe	Wizardry
399	Marisha	Chapez	Wizardry
415	Maximus	Flavius	Wizardry
999	Anna	Chase	Dark Magic
42	Aleks	Volochnev	Dark Magic

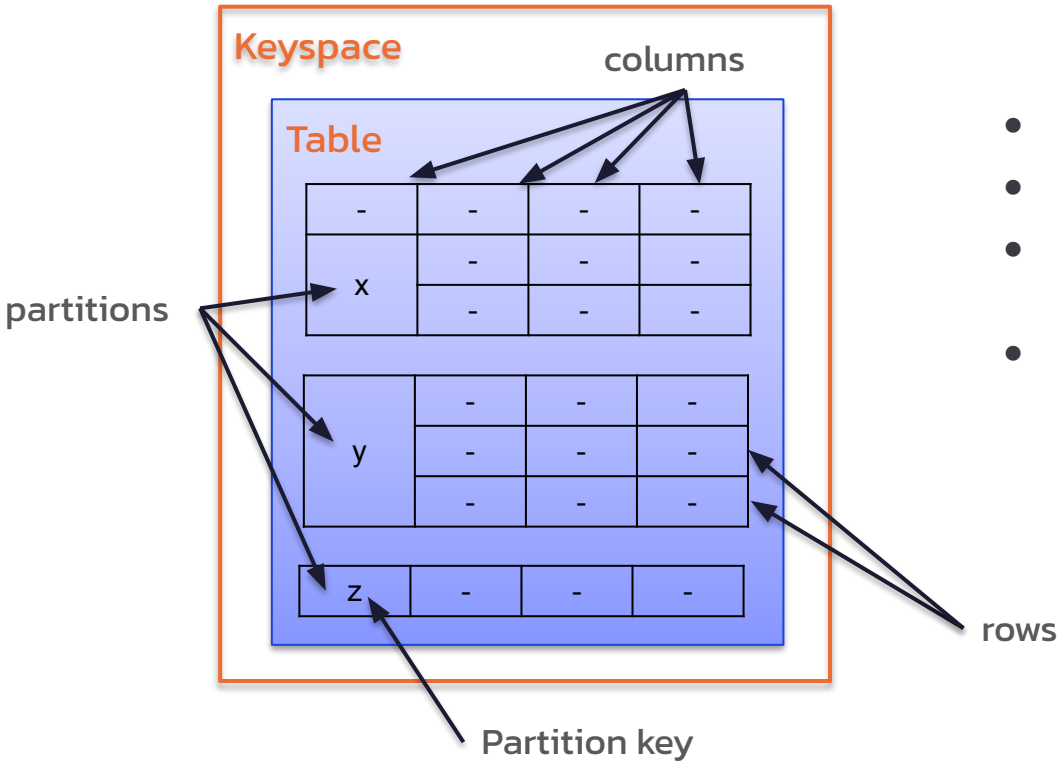
A group of rows having the same partition token, a base unit of access in Cassandra.

Data Structure: a Table

A group of columns and rows storing partitions.

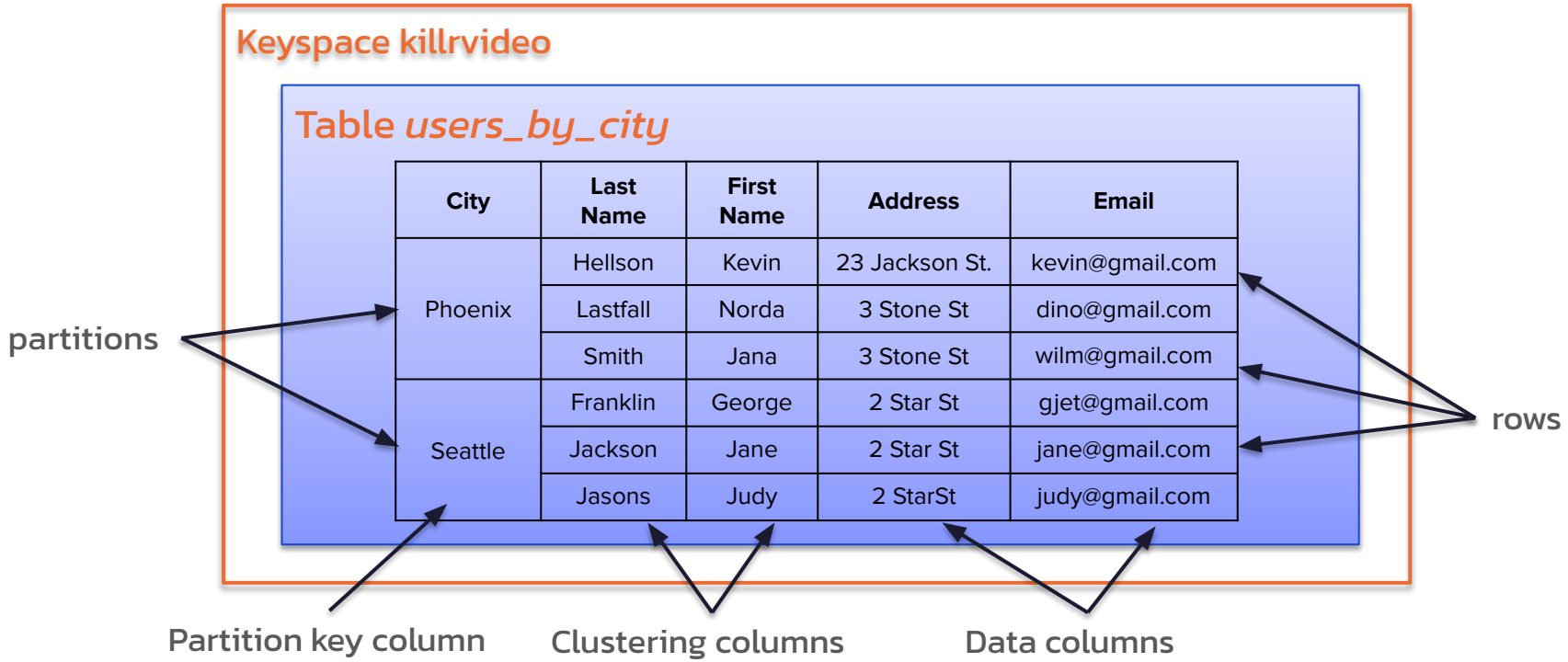
ID	First Name	Last Name	Department
1	John	Doe	Wizardry
2	Mary	Smith	Dark Magic
3	Patrick	McFadin	DevRel

Data Structure: Overall



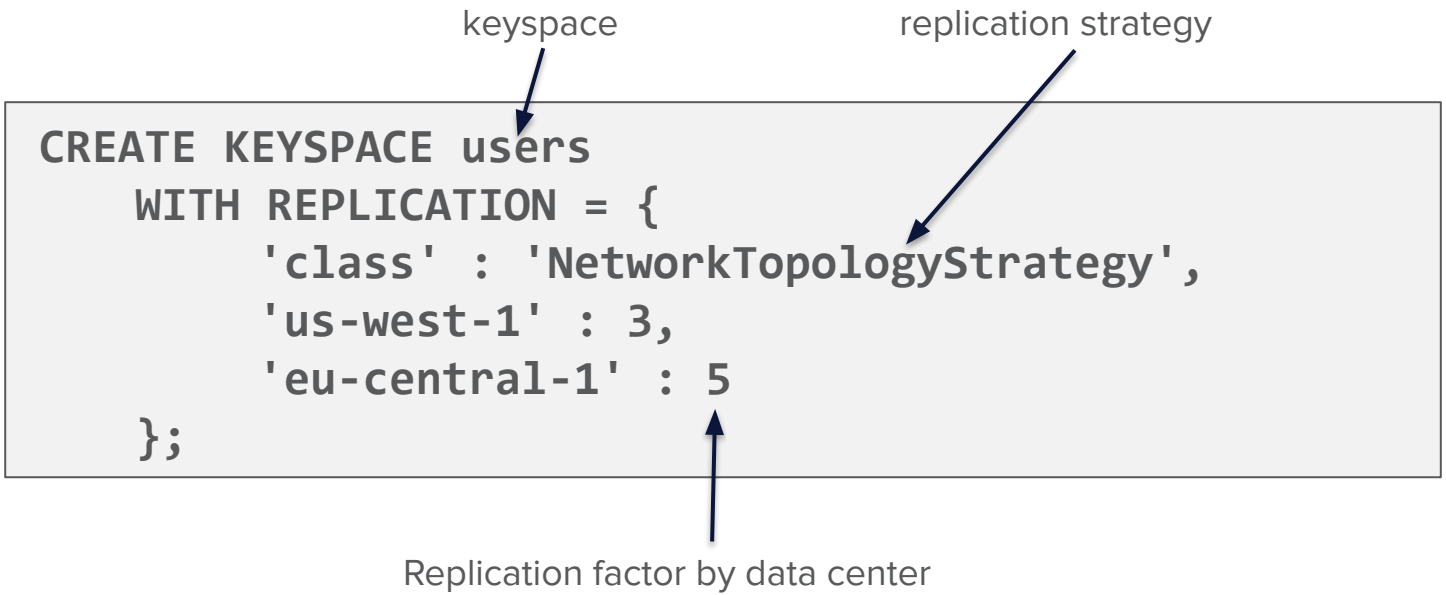
- Tabular data model, with one twist
- *Tables* are organized in *rows* and *columns*
- Groups of related rows called *partitions* are stored together on the same node (or nodes)
- Each row contains a *partition key*
  - One or more columns that are hashed to determine which node(s) store that data

Example Data: Users organized by city

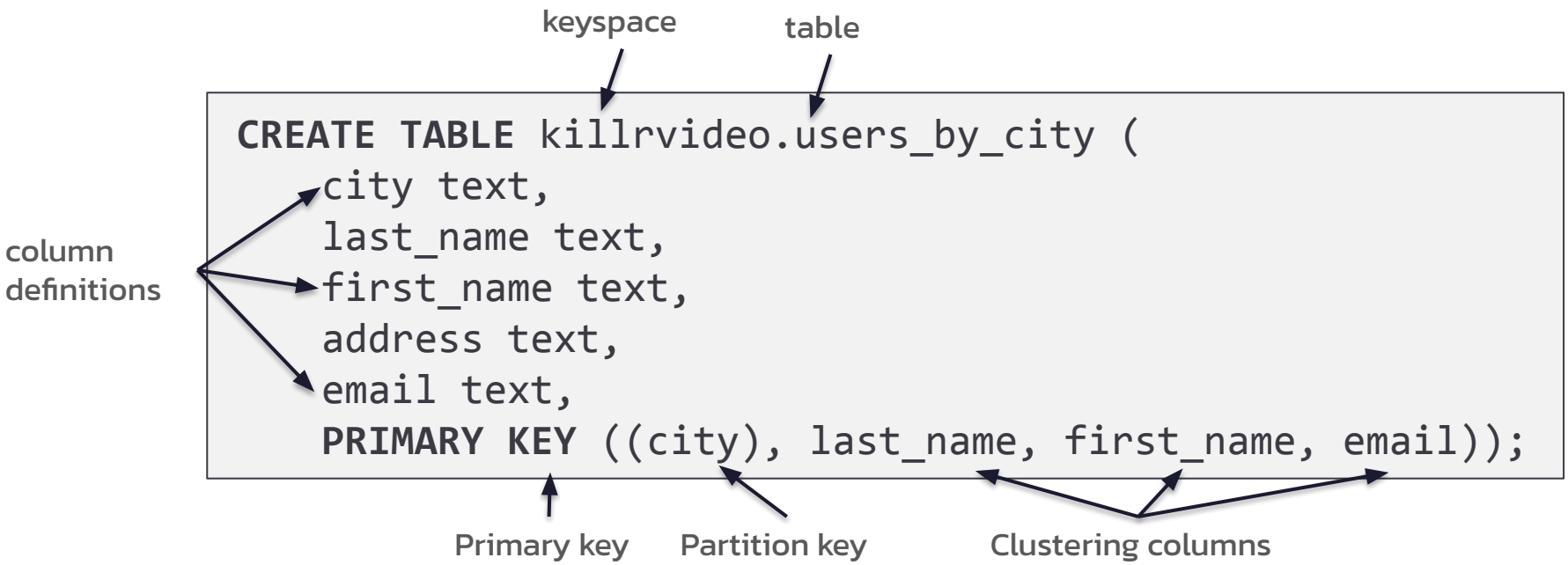


City	Last Name	First Name	Address	Email
Phoenix	Hellson	Kevin	23 Jackson St.	kevin@gmail.com
	Lastfall	Norda	3 Stone St	dino@gmail.com
	Smith	Jana	3 Stone St	wilm@gmail.com
Seattle	Franklin	George	2 Star St	gjet@gmail.com
	Jackson	Jane	2 Star St	jane@gmail.com
	Jasons	Judy	2 StarSt	judy@gmail.com

Keyspaces


keyspace
replication strategy
CREATE KEYSPACE users
WITH REPLICATION = {
'class' : 'NetworkTopologyStrategy',
'us-west-1' : 3,
'eu-central-1' : 5
};
Replication factor by data center

Creating a Table in CQL



Primary Key

An identifier for a row. Consists of at least one Partition Key and zero or more Clustering Columns.

MUST ENSURE UNIQUENESS.
MAY DEFINE SORTING.

```
CREATE TABLE killrvideo.users_by_city (
      city text,
      last_name text,
      first_name text,
      address text,
      email text,
      PRIMARY KEY ((city), last_name, first_name, email));
```

Good Examples:

```
PRIMARY KEY ((city), last_name, first_name, email);
```

```
PRIMARY KEY (user_id);
```

Bad Example:

```
PRIMARY KEY ((city), last_name, first_name);
```

Partition Key

An identifier for a partition.
Consists of at least one column, may have more if needed

PARTITIONS ROWS.

```
CREATE TABLE killrvideo.users_by_city (
      city text,
      last_name text,
      first_name text,
      address text,
      email text,
      PRIMARY KEY ((city), last_name, first_name, email));
```

Good Examples:

```
PRIMARY KEY (user_id);
```

```
PRIMARY KEY ((video_id), comment_id);
```

Bad Example:

```
PRIMARY KEY ((sensor_id), logged_at);
```

Clustering Column(s)

Used to ensure uniqueness and sorting order. Optional.

```
CREATE TABLE killrvideo.users_by_city (
      city text,
      last_name text,
      first_name text,
      address text,
      email text,
      PRIMARY KEY ((city), last_name, first_name, email));
```

Partition key

Clustering columns

```
PRIMARY KEY ((city), last_name, first_name);
```

Not Unique

```
PRIMARY KEY ((city), last_name, first_name, email);
```

```
PRIMARY KEY ((video_id), comment_id);
```

Not Sorted

```
PRIMARY KEY ((video_id), created_at, comment_id);
```

The Slide of the Year Award!

Rules of a Good Partition

- **Store together what you retrieve together**
- Avoid big partitions
- Avoid hot partitions

Example: open a video? Get the comments in a single query!

```
PRIMARY KEY ((video_id), created_at, comment_id);
```

```
PRIMARY KEY ((comment_id), created_at);
```

The Slide of the Year Award!

Rules of a Good Partition

- Store together what you retrieve together
- **Avoid big partitions**
- Avoid hot partitions

```
PRIMARY KEY ((video_id), created_at, comment_id);
```

```
PRIMARY KEY ((country), user_id);
```

- Up to 2 billion cells per partition
- Up to ~100k rows in a partition
- Up to ~100MB in a Partition

The Slide of the Year Award!

Rules of a Good Partition

- Store together what you retrieve together
- **Avoid big and constantly growing partitions!**
- Avoid hot partitions

Example: a huge IoT infrastructure, hardware all over the world, different sensors reporting their state every 10 seconds. Every sensor reports its UUID, timestamp of the report, sensor's value.

- Sensor ID: UUID
- Timestamp: Timestamp
- Value: float

```
PRIMARY KEY ((sensor_id), reported_at);
```

The Slide of the Year Award!

Rules of a Good Partition

- Store together what you retrieve together
- **Avoid big and constantly growing partitions!**
- Avoid hot partitions

Example: a huge IoT infrastructure, hardware all over the world, different sensors reporting their state every 10 seconds. Every sensor reports its UUID, timestamp of the report, sensor's value.

```
PRIMARY KEY ((sensor_id), reported_at);
```

```
PRIMARY KEY ((sensor_id, month_year), reported_at);
```

BUCKETING

- Sensor ID: UUID
- **MonthYear**: Integer or String
- Timestamp: Timestamp
- Value: float

The Slide of the Year Award!

Rules of a Good Partition

- Store together what you retrieve together
- Avoid big partitions
- **Avoid hot partitions**

```
PRIMARY KEY (user_id);
```

```
PRIMARY KEY ((video_id), created_at, comment_id);
```

```
PRIMARY KEY ((country), user_id);
```

Hands-on

https://www.datastax.com/learn/cassandra-fundamentals#coursework

Inserts, Updates, Deletes and Upserts in Apache Cassandra™

Learn how to insert, update, delete, and upsert data into tables in a Cassandra NoSQL database

START CODING

Queries in Apache Cassandra™

Learn how to retrieve data from Cassandra NoSQL database tables

START CODING

Using Advanced Data Types in Apache Cassandra™

Learn about universally unique identifiers (UUIDs), collections, tuples, user-defined types (UDTs) and counters

START CODING

Next Step: Academy.datastax.com

Designed for professionals that use Apache Cassandra clusters to manage data

application developers
data architects
database designers
database administrators

Agenda (6H)

01

Apache Cassandra™
Les fondamentaux

02

Apache Cassandra™
Modélisation

03

Apache Cassandra™
Développements

04

Kubernetes
Les fondamentaux

05

cassandra

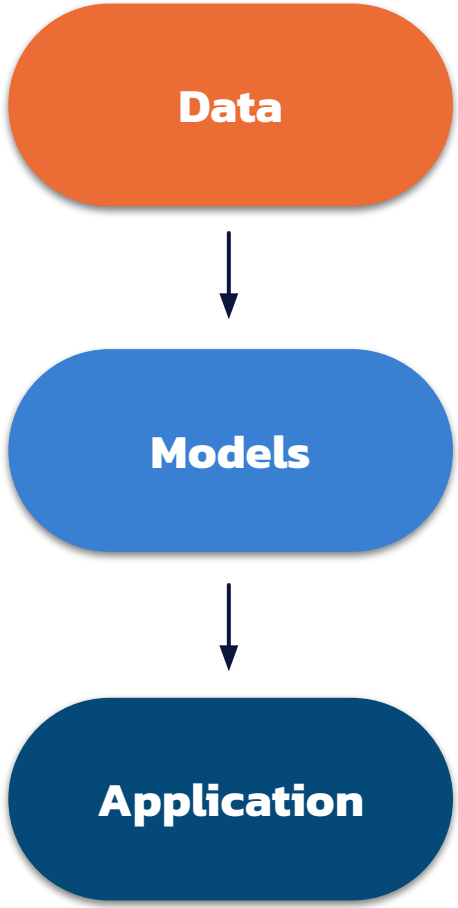
Kubernetes
Stateful & Cassandra

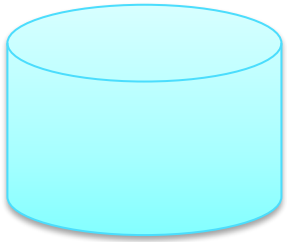
05

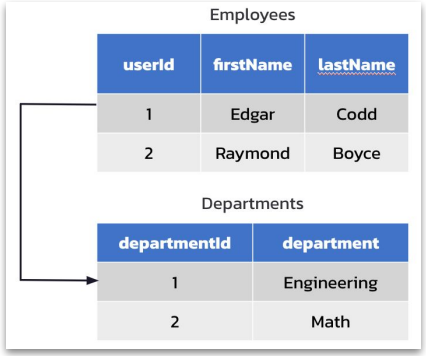
K8ssandra
Tour d'horizon

Relational Data Modelling

1. Analyze raw data
2. Identify entities, their properties and relations
3. Design tables, using **normalization** and foreign keys.
4. Use JOIN when doing queries to join normalized data from multiple tables

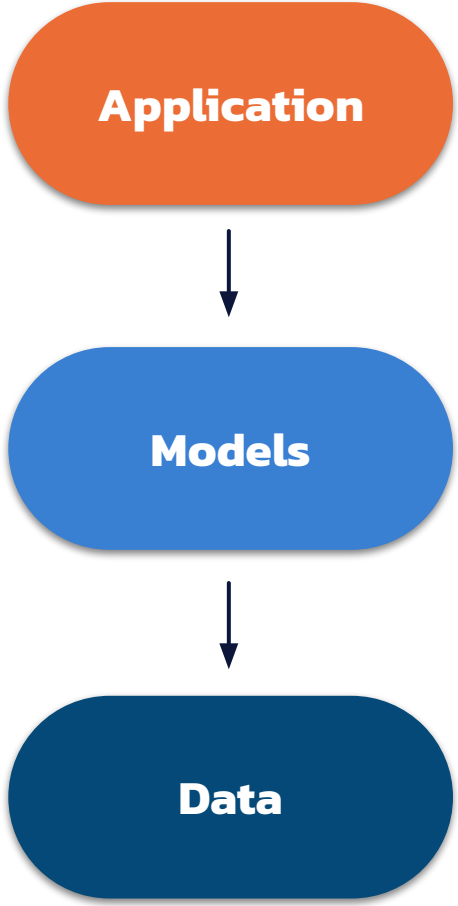






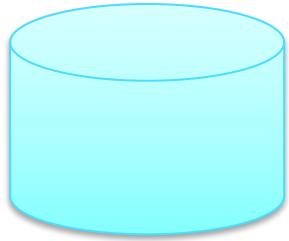
NoSQL Data Modelling

1. Analyze user behaviour (customer first!)
2. Identify workflows, their dependencies and needs
3. Define Queries to fulfill these workflows
4. Knowing the queries, design tables, using **denormalization**.
5. Use BATCH when inserting or updating denormalized data of multiple tables



Employees

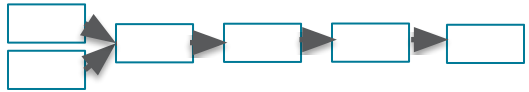
userId	firstName	lastName	department
1	Edgar	Codd	Engineering
2	Raymond	Boyce	Math
3	Sage	Lahja	Math
4	Juniper	Jones	Botany

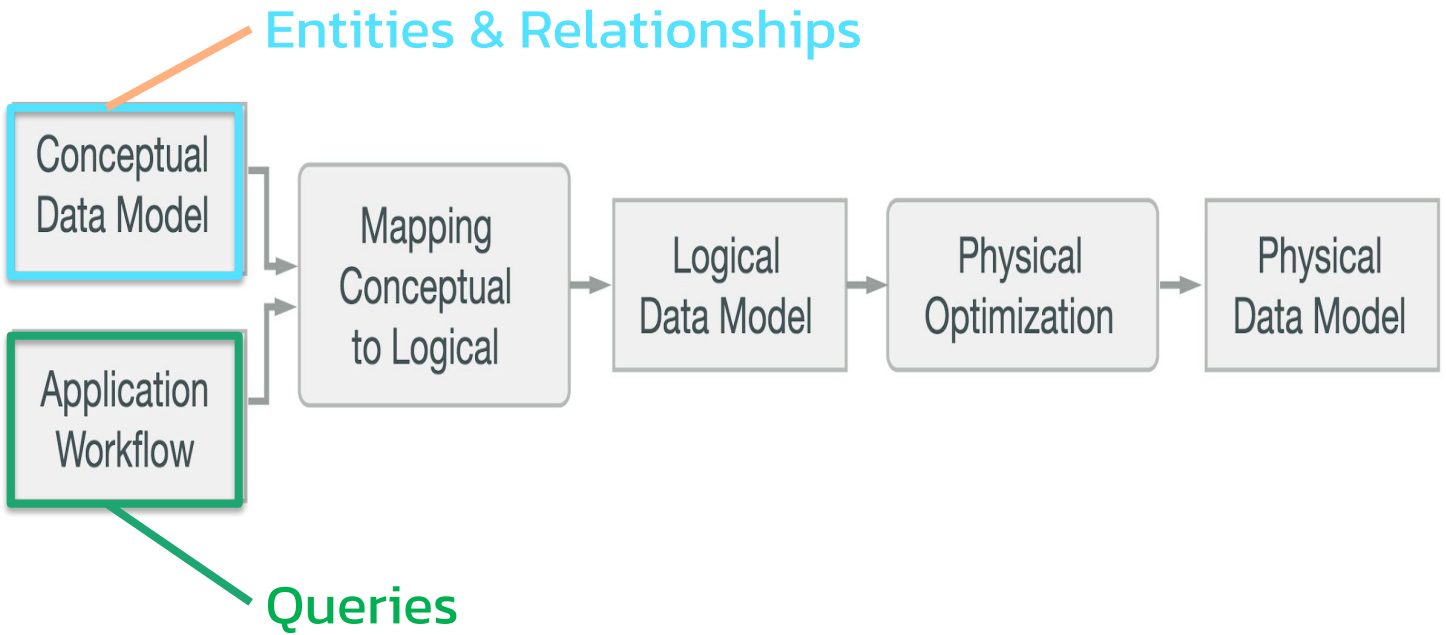


K8SSANDRA

DataStax Developers

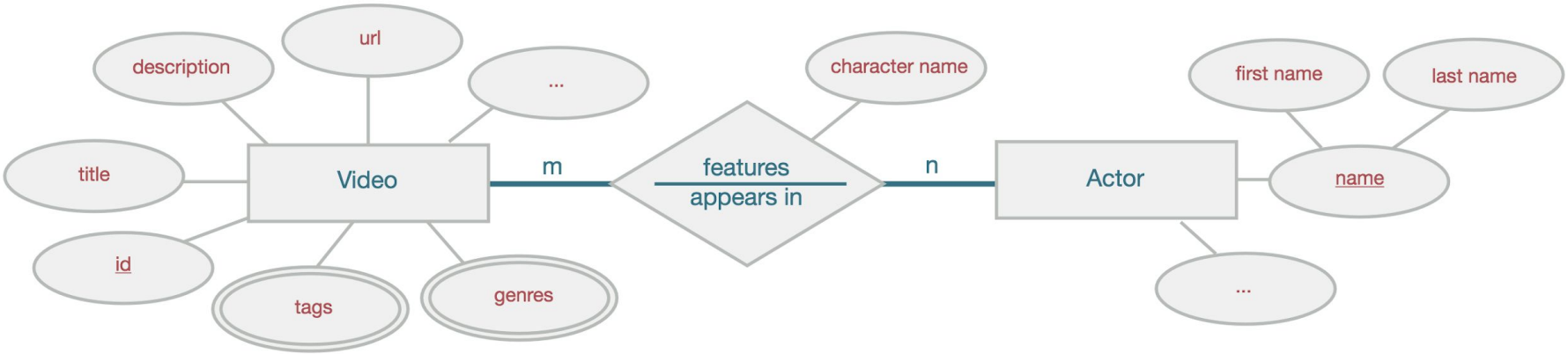
Designing Process: Step by Step



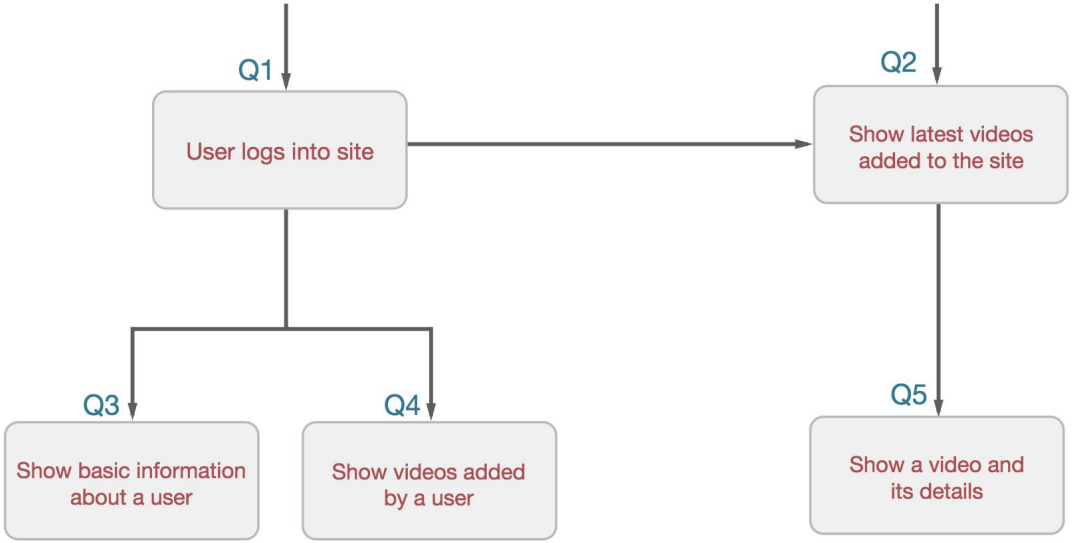


Conceptual Data Model

- Abstract view of the domain
- Technology independent
- Not specific to any database system



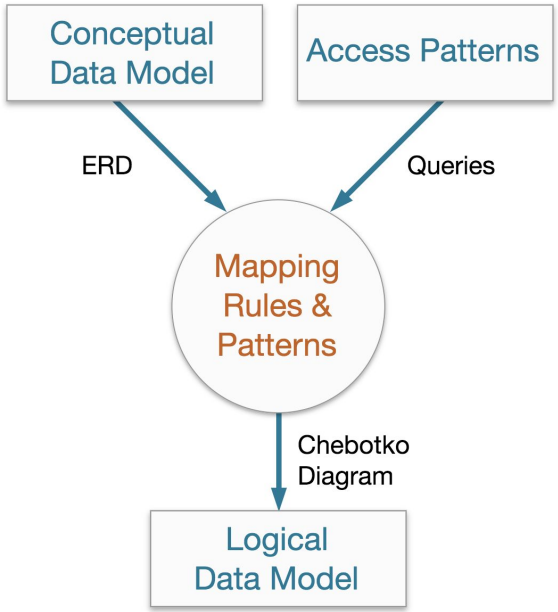
Application Workflow



ACCESS PATTERNS

Q1: Find a user with a **specified email**
Q2: Find most recently uploaded **videos**
Q3: Find a **user with a specified id**
Q4: Find videos uploaded by a **user with a known id**(show most recently uploaded videos first)
Q5: Find a video with a **specified video id**

Mapping Conceptual to logical

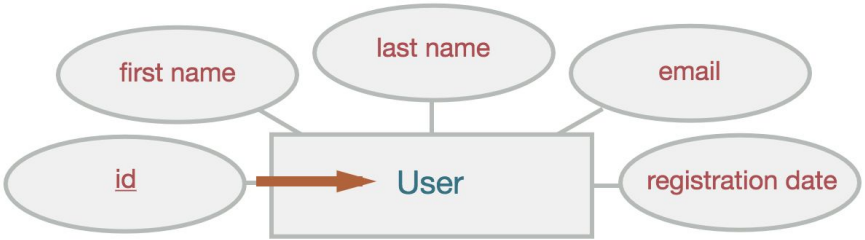


- Mapping Rule 1: Entities and relationships
- Mapping Rule 2: Equality search attributes
- Mapping Rule 3: Inequality search attributes
- Mapping Rule 4: Ordering attributes
- Mapping Rule 5: Key attributes

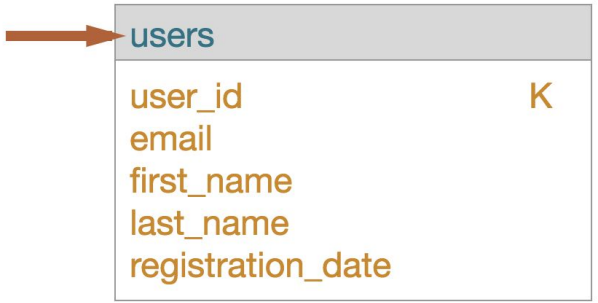
MR1 : Entities and Relationships

- Entity and relationship types map to tables
- Entities and relationships map to partitions or rows
- Partition may have data about one or more entities and relationships
- Attributes are represented by columns

Conceptual Model



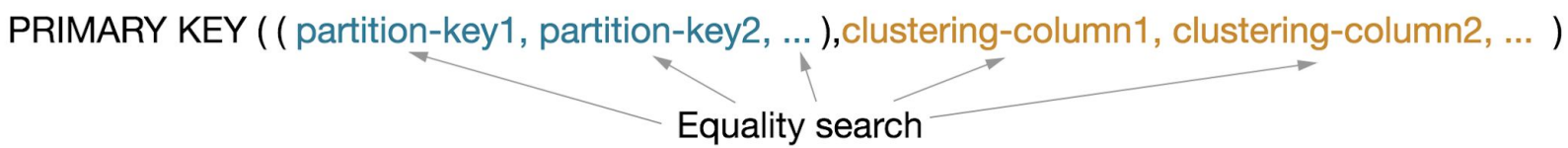
Logical Model



MR2: Equality Search Attributes

Attributes to query on must be in the front of the primary key

- Primary key is an ordered set of columns, made up of partition key and clustering columns
- A partition key is formed by one or more of the leading primary key columns
- Supported queries must include all partition key columns in the query

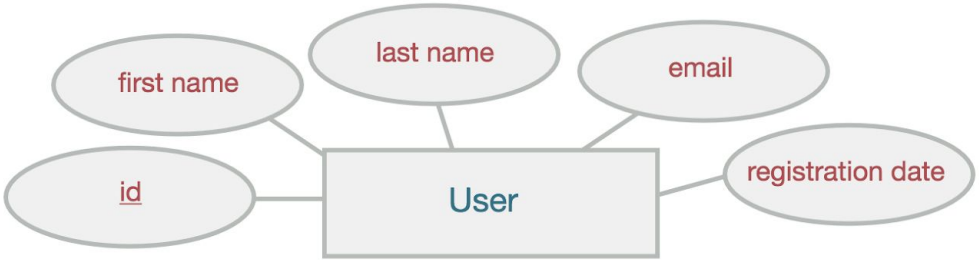


Violating this rule means that you may not be able to query on a specific column

MR3: Inequality Search Attributes

Entity type example

- Inequality search attributes become clustering columns
- Querying on: ***last_name*** = *?* *and* ***registration_date*** > *?*

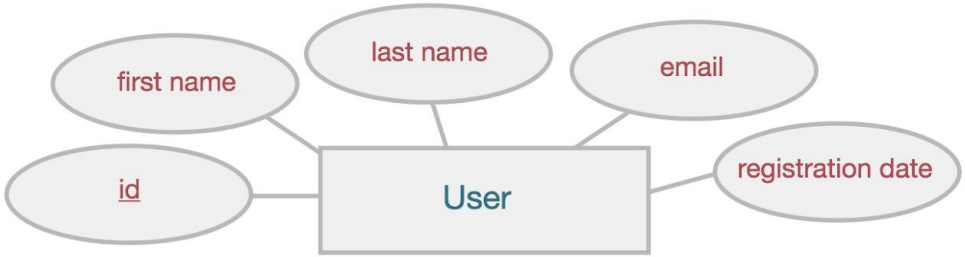


users	
last_name	K
registration_date	C↑
user_id	C↑
first_name	
email	

MR4: Ordering Attributes

Entity type example

- Ordering attributes become clustering columns
- Querying on: `last_name` = ? *and* `registration_date` > ?
- Ordering attributes: `registration_date` *(ASC)*

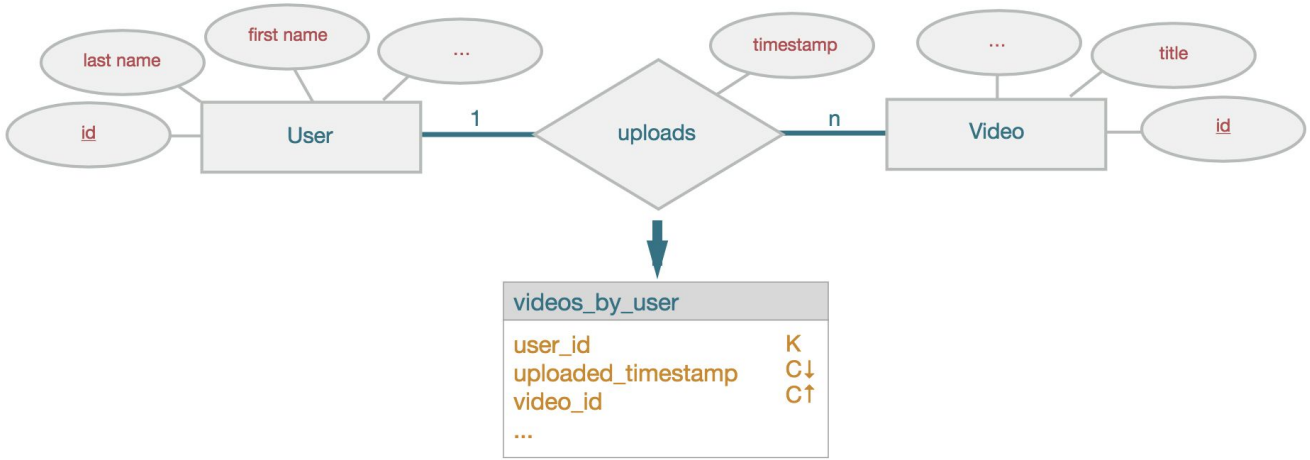


users_registration_date	
last_name	K
registration_date	C↑
user_id	C↑
first_name	
email	

MR4: Ordering Attributes

Relationship type example

- Ordering attributes become clustering columns
- Querying on: ***`user_id`*** = *?* *and* ***`uploaded_timestamp`*** > *?*
- Ordering attributes: ***`uploaded_timestamp`*** *(DESC)*



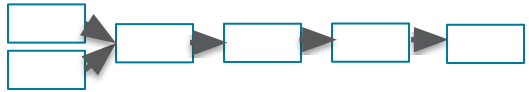
MR5: Key Attributes

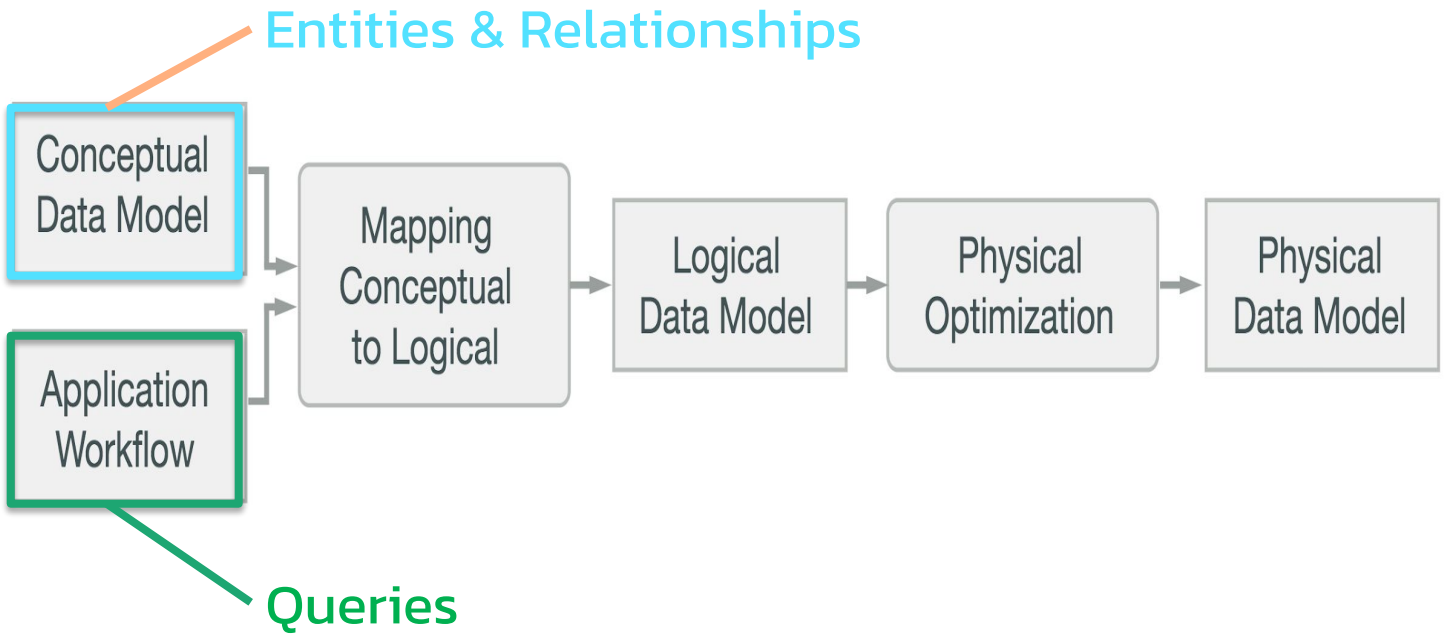
Key attributes map to primary key columns

- Primary key must include the columns that represent key attributes
- Position and order of such columns may vary
- Primary key may have additional columns to support specific queries

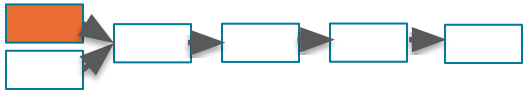
Violating this rule may result in upsert operations—loss of data

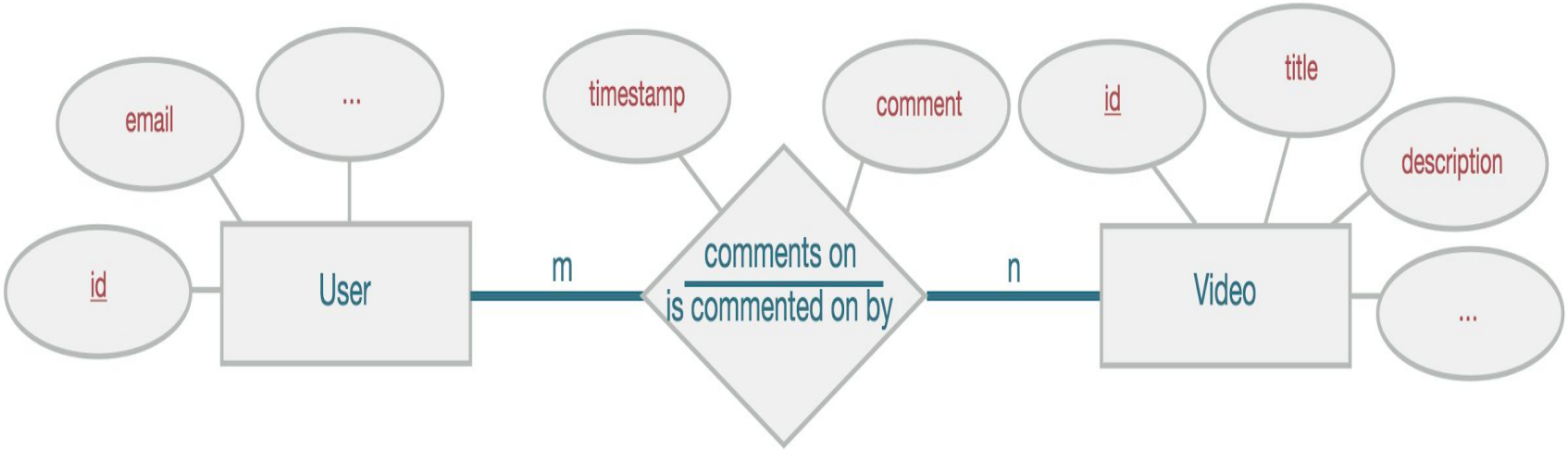
Designing Process: Step by Step



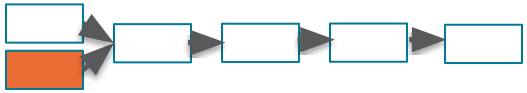


Designing Process: Conceptual Data Model





Designing Process: Application Workflow



Use-Case I:

- A User opens a Profile

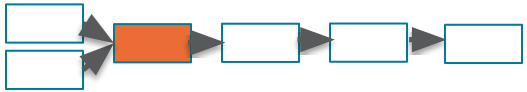
WF2: Find comments related to target user using its identifier, get most recent first

Use-Case II:

- A User opens a Video Page

WF1: Find comments related to target video using its identifier, most recent first

Designing Process: Mapping



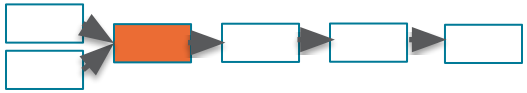
Query I: Find comments posted for a user with a known id (show most recent first)

→ comments_by_user

Query II: Find comments for a video with a known id (show most recent first)

→ comments_by_video

Designing Process: Mapping



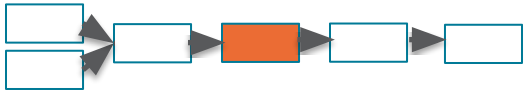
```
SELECT * FROM comments_by_user
WHERE userid = <some UUID>
```

→ comments_by_user

```
SELECT * FROM comments_by_video
WHERE videoid = <some UUID>
```

→ comments_by_video

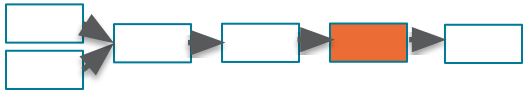
Designing Process: Logical Data Model



comments_by_user	
userid	K
creationdate	C ↓
commentid	C ↑
videoid	
comment	

comments_by_video	
videoid	K
creationdate	C ↓
commentid	C ↑
userid	
comment	

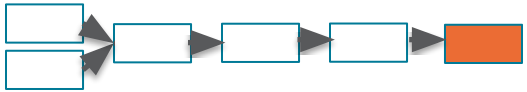
Designing Process: Physical Data Model



comments_by_user		
userid	UUID	K
commentid	TIMEUUID	C ↓
videoid	UUID	
comment	TEXT	

comments_by_video		
videoid	UUID	K
commentid	TIMEUUID	C ↓
userid	UUID	
comment	TEXT	

Designing Process: Schema DDL

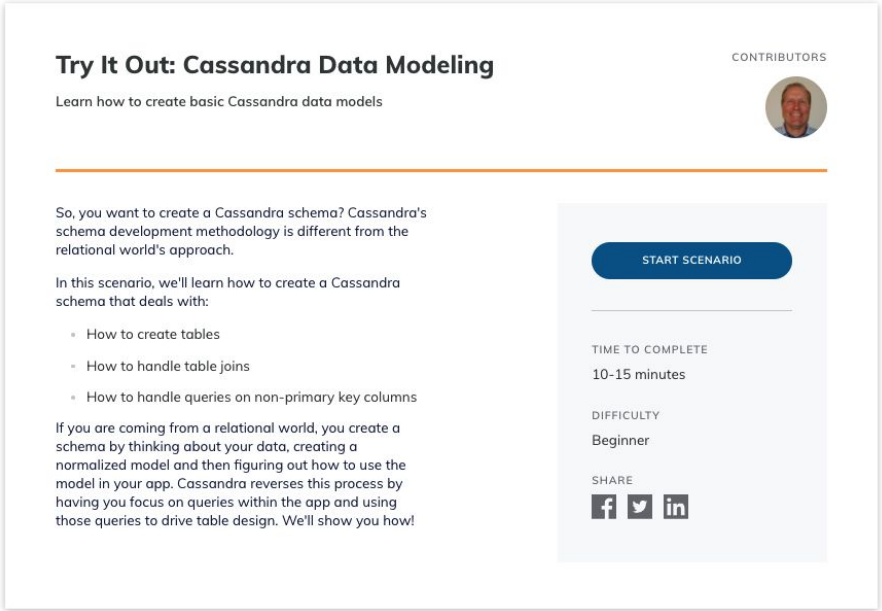


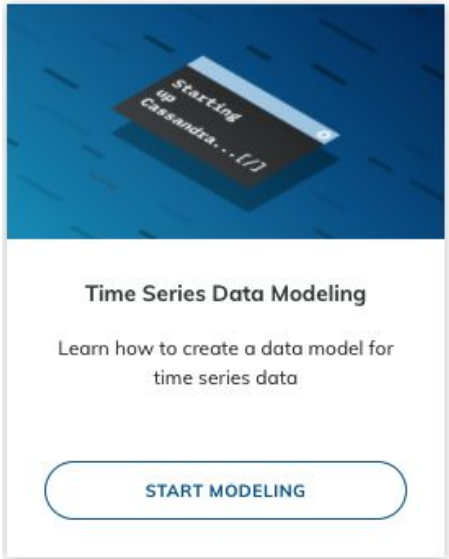
```
CREATE TABLE IF NOT EXISTS comments_by_user (
    userid uuid,
    commentid timeuuid,
    videoid uuid,
    comment text,
    PRIMARY KEY ((userid), commentid)
) WITH CLUSTERING ORDER BY (commentid DESC);

CREATE TABLE IF NOT EXISTS comments_by_video (
    videoid   uuid,
    commentid timeuuid,
    userid    uuid,
    comment   text,
    PRIMARY KEY ((videoid), commentid)
) WITH CLUSTERING ORDER BY (commentid DESC);
```

Hands-on

https://www.datastax.com/dev/modeling

https://www.datastax.com/learn/data-modeling-by-example





Next Step: Academy.datastax.com

Designed for professionals that use Apache Cassandra clusters to manage data

application developers
data architects
database designers
database administrators

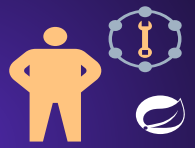
Agenda (6H)

01 

Apache Cassandra™
Les fondamentaux

02 

Apache Cassandra™
Modélisation

03 

Apache Cassandra™
Développements

04 

Kubernetes
Les fondamentaux

05 

Kubernetes
Stateful & Cassandra

05 

K8ssandra
Tour d'horizon

III - Apache Cassandra™ Développement

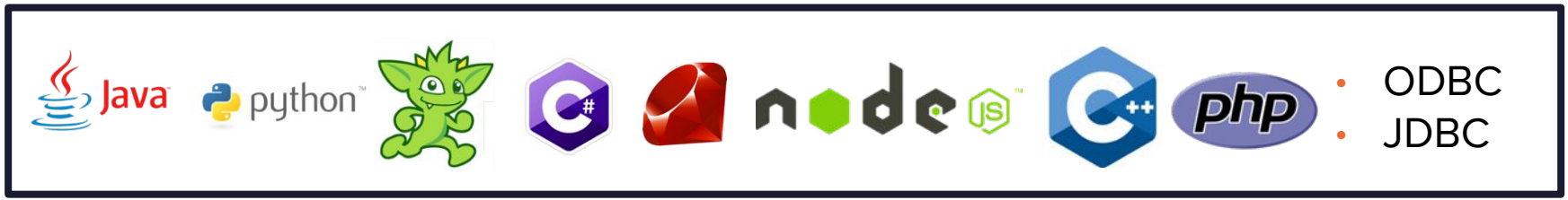
1. **Connectivity**
2. Build your queries
3. Execute Statements
4. Object Mapping
5. Asynchronous APIs
6. Reactive APIs

DataStax Drivers

- OSS Driver Features
  - CQL Support
  - Sync / Async API
  - Load Balancing Policies
  - Retry Policies
  - Reconnection Policies

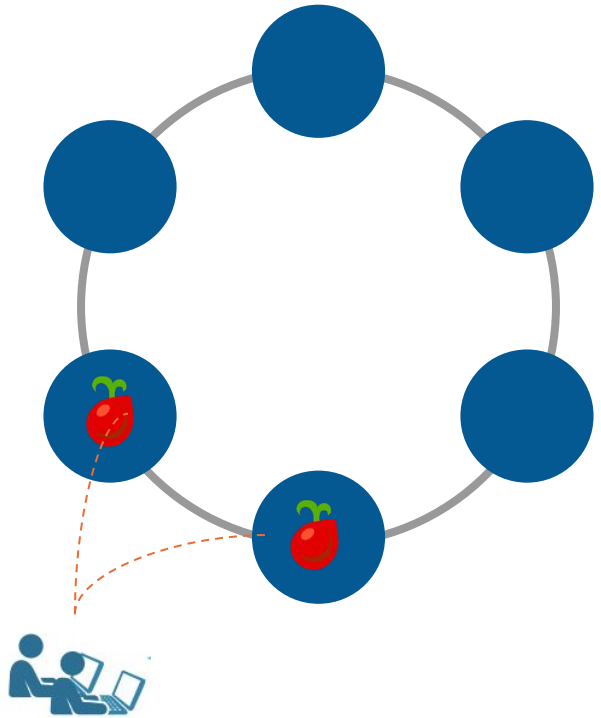
- Connection Pooling
- SSL
- Compression
- Query Builder
- Object Mapper

- Enterprise Driver Features
  - OSS Driver features, plus…
  - DSE Advanced Security, Unified Authentication
  - DSE Graph Fluent API
  - DSE Geometric Types



Contact Points

- Only one necessary
- Unless that node is down
- More are good



File-based Configuration

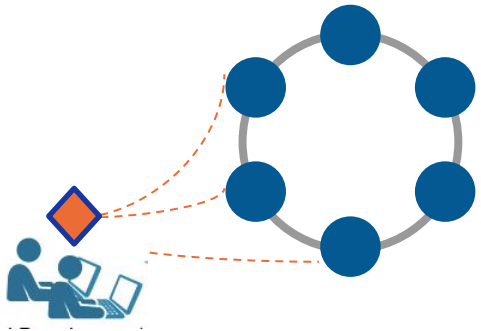
- Based on Typesafe Config
- Attributes are grouped into basic and advanced categories
- A reference file (*reference.conf*) provide default values embedded in the jar file. Can be override with key in application.conf.
- Driver searches `application.conf` in the classpath

application.conf

```
datastax-java-driver {
  basic {
    request.timeout     = 5 seconds
    request.consistency = LOCAL_QUORUM
  }
}
```

Load-Balancing

- Used to create query plans for each statement executed
- Default policy is token-aware, round robin
- Requests are routed to nodes in the “local” data center only



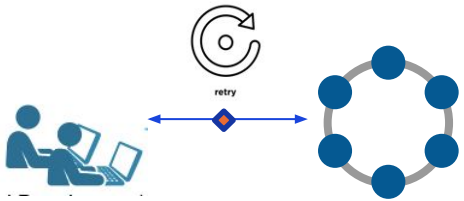
application.conf

```
datastax-java-driver {
 basic {
  load-balancing-policy {

    # The class of the policy.
    class = DefaultLoadBalancingPolicy

    # The datacenter that is considered "local"
    # The default policy will only include nodes from
    # this datacenter in its query plans.
    local-datacenter = datacenter1

    # A custom filter to include/exclude nodes
    // filter.class=
   }
 }
}
```

Retry Policy

Determines when queries are retried on failure

- `DefaultRetryPolicy`
  - Default
  - Retries once onReadTimeout or onWriteTimeout
  - Enough replicas for your consistency level must be online
  - Only retries idempotent mutations



application.conf

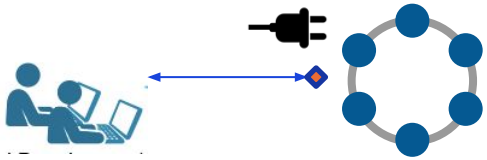
```
datastax-java-driver {

  # The policy that controls if the driver retries
  # requests that have failed on one node.
  advanced.retry-policy {

   # The class of the policy
   class = DefaultRetryPolicy

  }
}
```

Reconnection Policy

Reconnects driver to a downed node

Two options:

- `ConstantReconnectionPolicy`
  - Check every N milliseconds
- `ExponentialReconnectionPolicy`
  - Increases every interval
  - Caps out at a max



application.conf

```
datastax-java-driver {

  # Whether to schedule reconnection attempts
  # if all contact points are unreachable at init
  advanced.reconnect-on-init = false

  advanced.reconnection-policy {

    # The class of the policy
    class = ExponentialReconnectionPolicy

    # Parameters
    base-delay = 1 second
    max-delay = 60 seconds
  }
}
```

Important to know about CqlSession

- **CqlSession** is a stateful object handling communications with each node
- **CqlSession** should be unique in the Application (*Singleton*)
- **CqlSession** should be closed at application shutdown (*shutdown hook*) in order to free opened TCP sockets (*stateful*)

```
@PreDestroy
public void cleanup() {
 if (null != cqlSession) {
   cqlSession.close();
 }
}
```

III - Apache Cassandra™ Développement

1. Connectivity
2. **Build your queries**
3. Execute Statements
4. Object Mapping
5. Asynchronous APIs
6. Reactive APIs

How to to execute queries ?

- First job of **CqlSession** is to execute queries using, well, execute method.

```
cqlSession.execute("SELECT * FROM killrvideo.users");
```

SimpleStatement

```
Statement statement = …

// (1) Explicit SimpleStatement Definition
SimpleStatement.newInstance("select * from t1 where c1 = 5");

// (2) Externalize Parameters (no name)
SimpleStatement.builder("select * from t1 where c1 = ?")
               .addPositionalValue(5);

// (3) Externalize Parameters (name)
SimpleStatement.builder("select * from t1 where c1 = :myVal")
               .addNamedValue("myVal”, 5);

cqlSession.execute(statement);
```

Prepared and Bound Statements

- Compiled once on each node automatically as needed
- Prepare each statement only once per application
- Use one of the many bind variations to create a BoundStatement

```
PreparedStatement ps = cqlSession.prepare("SELECT * from t1 where c1 = ?");

BoundStatement bound = ps.bind(5);

cqlSession.execute(bound);
```


Query
Builder

Query Builder

- Fluent API for building CQL string queries programmatically
- Contains methods to build SELECT, UPDATE, INSERT and DELETE statements
- Generates a Statement as per the earlier techniques

OSS Driver (current version 4.2.0)

```
<dependency>
 <groupId>com.datastax.oss</groupId>
 <artifactId>
  java-driver-query-builder
 </artifactId>
</dependency>
```

DSE Driver (current version 2.2.0)

```
<<dependency>
 <groupId>com.datastax.dse</groupId>
 <artifactId>
  dse-java-driver-query-builder
 </artifactId>
</dependency>
```

Query Builder

```
import static com.datastax.oss.driver.api.querybuilder.QueryBuilder.bindMarker;
import static com.datastax.oss.driver.api.querybuilder.QueryBuilder.deleteFrom;
import static com.datastax.oss.driver.api.querybuilder.QueryBuilder.selectFrom;
import static com.datastax.oss.driver.api.querybuilder.relation.Relation.column;

// Simple SELECT using QueryBuilder
Statement stmtSelect = selectFrom(”killrvideo”, ”videos_by_users”)
  .column(”userid”).column(”commentid”)
  .function(”toTimestamp”, Selector.column(”commentid”)).as("comment_timestamp")
  .where(column(”userid”).isEqualTo(bindMarker(”userid”)))
  .build()

// Simple DELETE using QueryBuilder
Statement stmtDelete = deleteFrom(”killrvideo”, ”videos_by_users”)
   .where(column(”userid”).isEqualTo(bindMarker(”userid”)))
   .build()
```

Query Builder

- Can also use **QueryBuilder** to create **PreparedStatements** and later execute at runtime
- Note use of **bindMarker() t**o designate parameters that will be provided later

```
// Prepared QueryBuilder statiements as any statement
PreparedStatement psStmt = cqlSession.prepare(
  deleteFrom(”killrvideo”, ”videos_by_users”)
   .where(column(”userid”).isEqualTo(bindMarker(”userid”)))
   .build()));

// Binding
BoundStatement bsStmt = psStmt.bind(”e7a8ac9f-c12d-415c-a526-4137815df573”);

// Execute
cqlSession.execute(bsStmt);
```

III - Apache Cassandra™ Développement

1. Connectivity
2. Build your queries
3. **Execute Statements**
4. Object Mapping
5. Asynchronous APIs
6. Reactive APIs

ResultSet

- ResultSet is the object returned for executing query. It contains **ROWS** (data) and **EXECUTION INFO**.
- ResultSet is **iterable** and as such you can navigate from row to row.
- Results are **always paged** for you (avoiding memory and response time issues)

```
ResultSet rs = cqlSession.execute(myStatement);

// Plumbery
ExecutionInfo info = rs.getExecutionInfo();
int executionTime = info.getQueryTrace().getDurationMicros();

// Data: NOT ALL DATA RETRIEVED IMMEDIATELY (only when needed .next())
Iterator<Row> iterRow = rs.iterator();
int itemsFirstCall = rs.getAvailableWithoutFetching();
```

Parsing ResultSet

```
// We know there is a single row (eg: count)
Row singleRow = resultSet.one();

// We know there are not so many results we can get all (fetch all pages)
List<Row> allRows = resultSet.all();

// Browse iterable
for(Row myRow : resultSet.iterator()) {
   // .. Parsing rows
}

// Use Lambda
rs.forEach(row -> { row.getColumnDefinitions(); });

// Use for LWT
boolean isQueryExecuted = rs.wasApplied();
```

Parsing Rows

```
// Sample row
Row row = resultSet.one();

// Check null before read
Boolean isUsernNameNull = row.isNull("userName");

// Reading Values from row
String userName1 = row.get("username", String.class);
String userName2 = row.getString("username");
String userName3 = row.getString(CqlIdentifier.fromCql("username"));

// Tons of types available
row.getUuid("userid");
row.getBoolean(”register");
row.getCqlDuration(”elapsed");
…
```

Paging

- ResultSet contains up to "pageSize" items. When browsing records you may hit this number that will trigger fetching next "pageSize" items.
- To fetch anything else that first page you must provide a PagingState.

```
// Enforce few items per page (often = UI requirements)
myStatement = myStatement.setPageSize(10);
ResultSet page1 = cqlSession.execute(myStatement);

// Paging State
ByteBuffer pagingState = page1.getExecutionInfo().getPagingState();
myStatement = myStatement.setPageState(pagingState);

// Very same statement with pagingState provided
ResultSet page2 = cqlSession.execute(myStatement);
```

Batches – What you need to know

- Batches about **data integrity** between tables
- Not Atomic & not used for mass query optimization
- Used to keep denormalized data in sync

- There is no guarantee that a batch will complete all operations.
- There are still edge cases where things can fail out.
- There is no rollback if something fails
- This is where upserts come into play as you can simply re-fire the batch.

Batch Example

```
// Sample statements (insert same data in multiple tables)
Statement stmt1 = SimpleStatement
 .builder(”INSERT INTO users_by_group(groupid,userid) values(?,?)")
 .addPositionalValue(groupname, username);
Statement stmt2 = SimpleStatement
 .builder(”INSERT INTO groups_by_user(userid,groupid) values(?,?)")
 .addPositionalValue(username, groupname);

// Group as a Batch
BatchStatement batchStmt = BatchStatement
  .builder(DefaultBatchType.LOGGED)
  .addStatement(stmt1).addStatement(stmt2).build();

// Execute
cqlSession.execute(batchStmt);
```

Profiles

Override parameters for dedicated request

```
SimpleStatement
  .newInstance("select...")
  .setPageSize(10)
  // here is the magic 
  .setExecutionProfileName("dse_search");
```

application.conf

```
datastax-java-driver {
   profiles{

    # DSE Search type of queries
    dse_search {
      basic {
       request.consistency = LOCAL_ONE
       request.timeout     = 5 seconds
      }
    } # The class of the policy
    fast_query {
       basic.request.consistency = ONE
       basic.request.timeout     = 1 second
    }
  }
}
```

III - Apache Cassandra™ Développement

1. Connectivity
2. Build your queries
3. Execute Statements
4. **Object Mapping**
5. Asynchronous APIs
6. Reactive APIs

Object Mapper

- WHAT ?
  - Abstracts details of mapping Java attributes to/from CQL types and UDTs
  - Packaged separately from the driver – pom.xml update required
  - This slide shows the runtime dependency – will show compile-time shortly

- HOW ?
  - Some annotation processors will GENERATE Mapper, Dao, and Entity implementations for you
  - At each update in the files, the IDE (eclipse, intelliJ) will use annotation processor
  - Compiler plugin must be updated to define the annotation processor

Object Mapper

OSS Driver

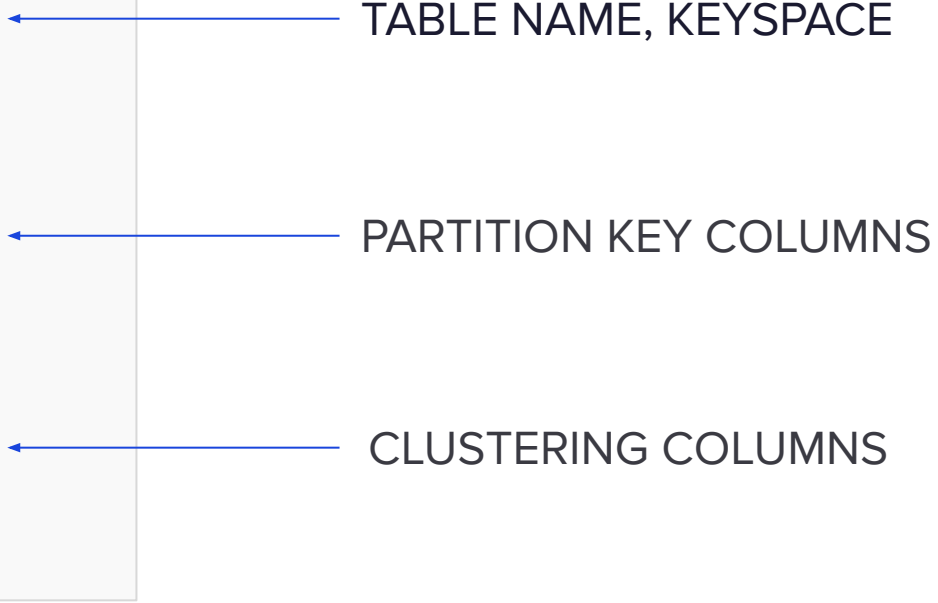
```
<dependency>
 <groupId>com.datastax.oss</groupId>
 <artifactId>java-driver-mapper-runtime</artifactId>
</dependency>

<!-- X -->

<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <configuration>
    <release>11</release>
 <annotationProcessorPaths>
 <path>
 <groupId>com.datastax.oss</groupId>
 <artifactId>java-driver-mapper-processor</artifactId>
 </path>
 </annotationProcessorPaths>
   </configuration>
</plugin>
```

DSE Driver

```
<dependency>
 <groupId>com.datastax.dse</groupId>
 <artifactId>dse-java-driver-mapper-runtime</artifactId>
</dependency>

<!-- X -->

<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <configuration>
    <release>11</release>
 <annotationProcessorPaths>
 <path>
 <groupId>com.datastax.dse</groupId>
 <artifactId>dse-java-driver-mapper-processor</artifactId>
 </path>
 </annotationProcessorPaths>
   </configuration>
</plugin>
```

Annotate Entities

```
@Entity
@CqlName("user_v")
public class UserVideo {

 @PartitionKey
 @CqlName("userid")
 private UUID userid;

 @ClusteringColumn(1)
 @CqlName(”added")
 private UUID videoid;

}
```



Annotate DAO Interface (1/2)

```
@Dao
public interface VideoDao {

  @Select
  Optional<UserVideo> findUserById(UUID userid);

  @Query("SELECT * FROM ${tableId}")
  PagingIterable<UserVideo> findAll();

  @Select(customWhereClause = "videoid = : videoid")
  PagingIterable<UserVideo>
    findUserByVideoId(@CqlName("videoid") UUID vid);
```

Annotate DAO Interface (2/2)

```
// Save a bean
@Insert
void save(UserVideo userVideo);

// Userid id is PK
@Delete
void delete(UUID userid);

// Custom implementations
@QueryProvider(
  providerClass = MySampleQueryProvider.class,
  entityHelpers = { UserVideo.class })
String doSomething(String abc);
```

Annotate Mapper Interface

```
@Mapper
public interface MyApplicationMapper {

    @DaoFactory
    VideoDao videoDao(@DaoKeyspace CqlIdentifier keyspace);

}
```

Sample QueryProvider

```
public class MySampleQueryProvider {

  // Constructor, getting session
  public MySampleQueryProvider(
     MapperContext context,
     EntityHelper<UserVideo > helperUser} {}

  // Custom implementation method
  public String doSomething(String abc) {
  }
}
```

Object Mapper

- WHAT ?
  - Abstracts details of mapping Java attributes to/from CQL types and UDTs
  - Packaged separately from the driver – pom.xml update required
  - This slide shows the runtime dependency – will show compile-time shortly

- HOW ?
  - Some annotation processors will GENERATE Mapper, Dao, and Entity implementations for you
  - At each update in the files, the IDE (eclipse, intelliJ) will use annotation processor
  - Compiler plugin must be updated to define the annotation processor

Object Mapper

OSS Driver

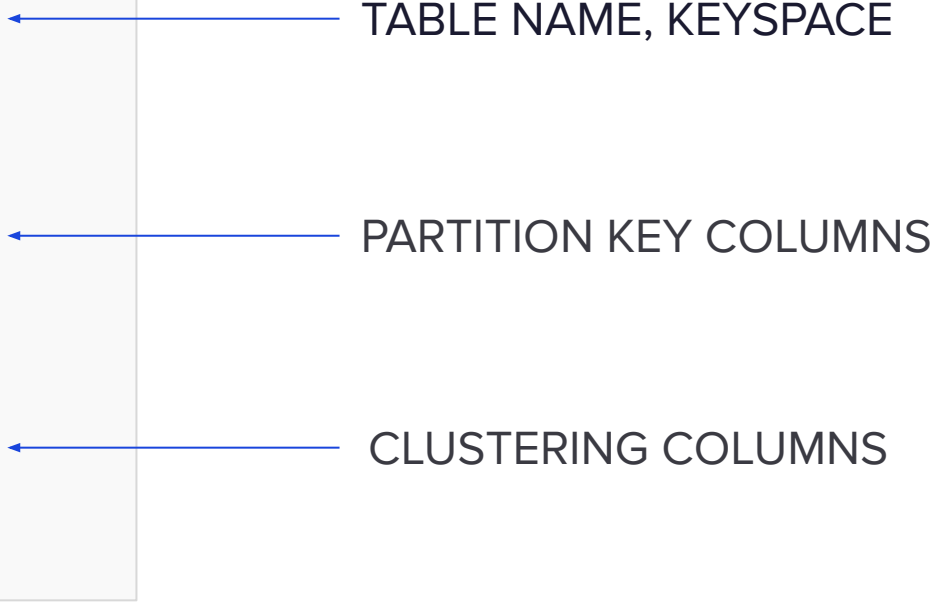
```
<dependency>
 <groupId>com.datastax.oss</groupId>
 <artifactId>java-driver-mapper-runtime</artifactId>
</dependency>

<!-- X -->

<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <configuration>
    <release>11</release>
 <annotationProcessorPaths>
 <path>
 <groupId>com.datastax.oss</groupId>
 <artifactId>java-driver-mapper-processor</artifactId>
 </path>
 </annotationProcessorPaths>
   </configuration>
</plugin>
```

DSE Driver

```
<dependency>
 <groupId>com.datastax.dse</groupId>
 <artifactId>dse-java-driver-mapper-runtime</artifactId>
</dependency>

<!-- X -->

<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <configuration>
    <release>11</release>
 <annotationProcessorPaths>
 <path>
 <groupId>com.datastax.dse</groupId>
 <artifactId>dse-java-driver-mapper-processor</artifactId>
 </path>
 </annotationProcessorPaths>
   </configuration>
</plugin>
```

Annotate Entities

```
@Entity
@CqlName("user_v")
public class UserVideo {

 @PartitionKey
 @CqlName("userid")
 private UUID userid;

 @ClusteringColumn(1)
 @CqlName(”added")
 private UUID videoid;

}
```



Annotate DAO Interface (1/2)

```
@Dao
public interface VideoDao {

  @Select
  Optional<UserVideo> findUserById(UUID userid);

  @Query("SELECT * FROM ${tableId}")
  PagingIterable<UserVideo> findAll();

  @Select(customWhereClause = "videoid = : videoid")
  PagingIterable<UserVideo>
    findUserByVideoId(@CqlName("videoid") UUID vid);
```

Annotate DAO Interface (2/2)

```
// Save a bean
@Insert
void save(UserVideo userVideo);

// Userid id is PK
@Delete
void delete(UUID userid);

// Custom implementations
@QueryProvider(
  providerClass = MySampleQueryProvider.class,
  entityHelpers = { UserVideo.class })
String doSomething(String abc);
```

Annotate Mapper Interface

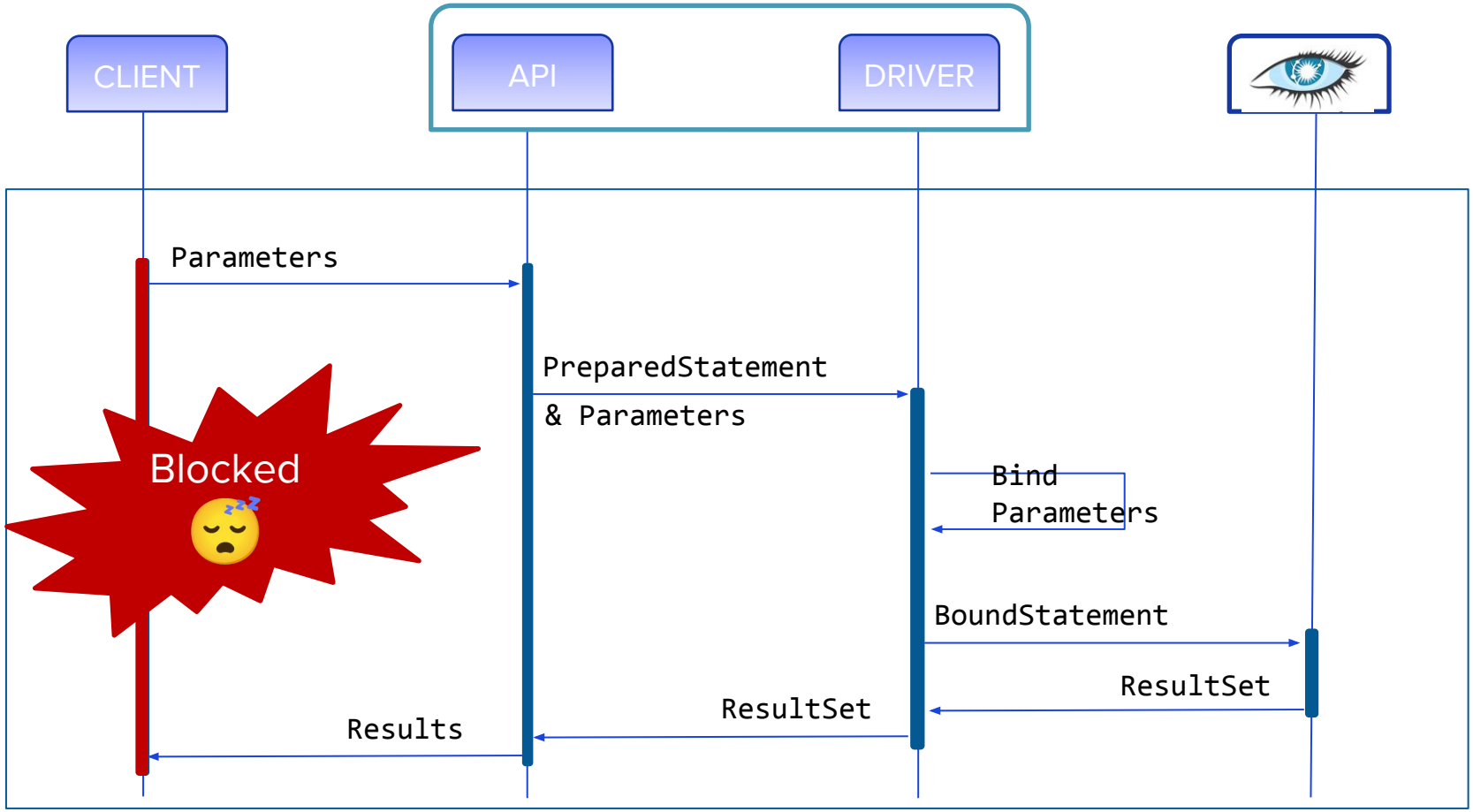
```
@Mapper
public interface MyApplicationMapper {

    @DaoFactory
    VideoDao videoDao(@DaoKeyspace CqlIdentifier keyspace);

}
```

Sample QueryProvider

```
public class MySampleQueryProvider {

  // Constructor, getting session
  public MySampleQueryProvider(
     MapperContext context,
     EntityHelper<UserVideo > helperUser} {}

  // Custom implementation method
  public String doSomething(String abc) {
  }
}
```

III - Apache Cassandra™ Développement

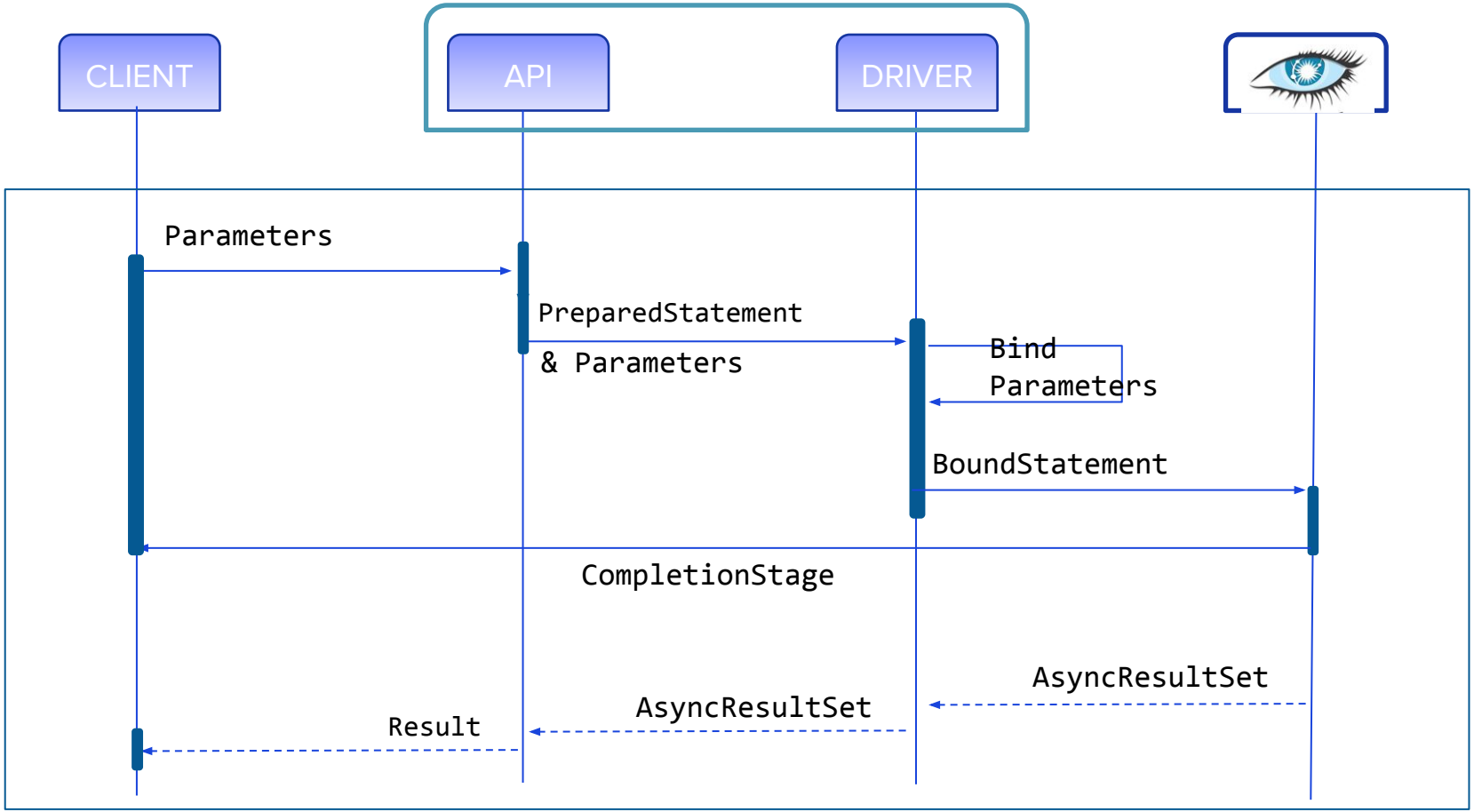
1. Connectivity
2. Build your queries
3. Execute Statements
4. Object Mapping
5. **Asynchronous APIs**
6. Reactive APIs



Capgemini

Synchronous Queries

DataStax Developers



Capgemini

Asynchronous Queries

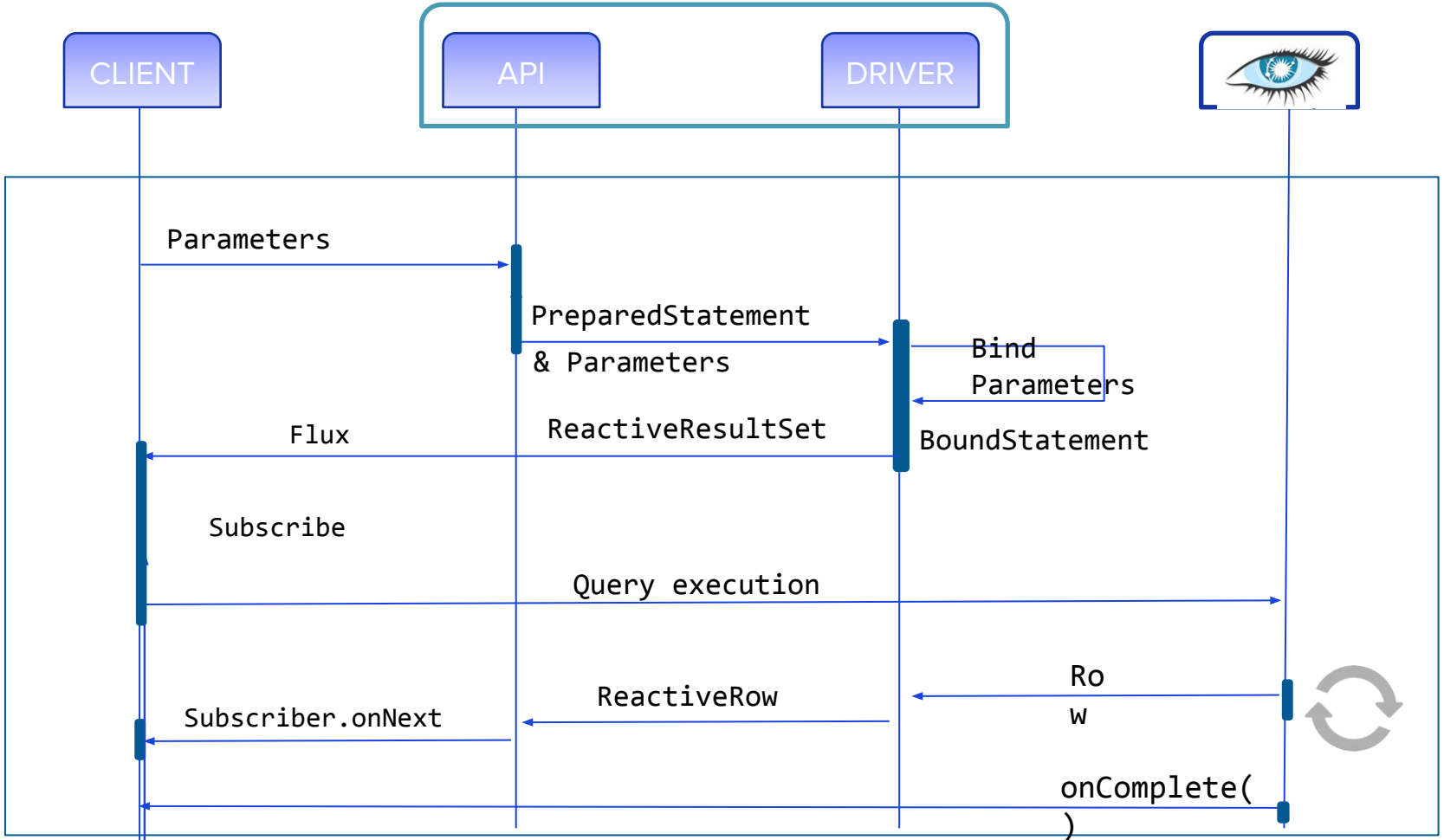
DataStax Developers

Asynchronous Queries

```
// From Synchronous
ResultSet resSync = cqlSession.execute(myStatement);

// to Asynchronous
CompletionStage<AsyncResultSet> resAsync =
                  cqlSession.executeAsync(myStatement);

resAsync.thenApply(AsyncPagingIterable::one)
        .thenApply(Optional::ofNullable)
        .thenApply(optional -> optional.map(rowMapper))
        .then..
```

III - Apache Cassandra™ Développement

1. Connectivity
2. Build your queries
3. Execute Statements
4. Object Mapping
5. Asynchronous APIs
6. **Reactive APIs**



Capgemini

Reactive Queries

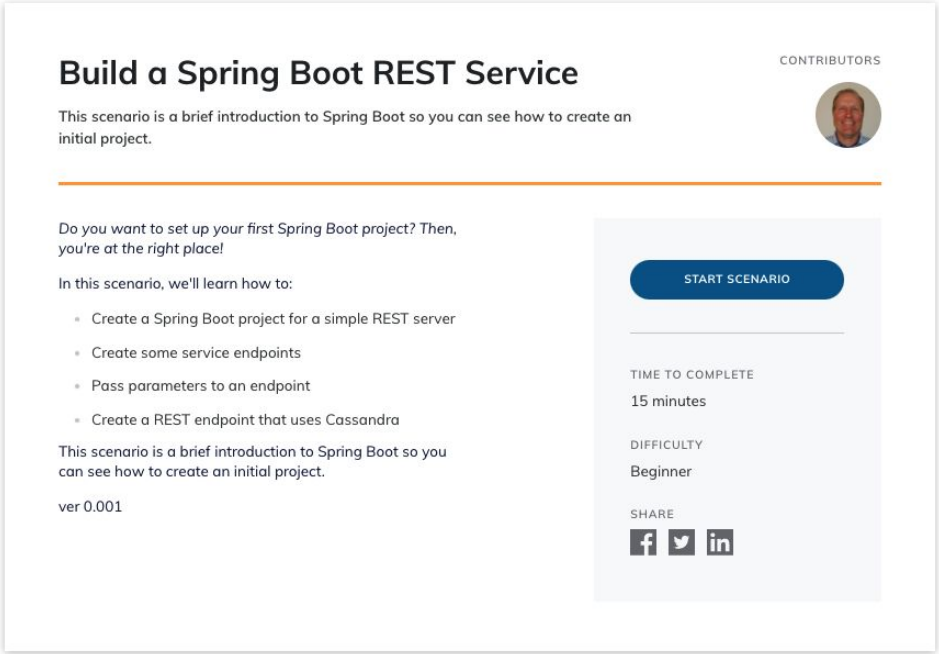
DataStax Developers

Reactive Queries

```
private final Function<Row, MyBean> rowMapper = ..;

// Execute in reactive way
ReactiveResultSet rs = session.executeReactive(myStatement);

// Return Flux for lists
Flux<ReactiveRow> flux = Flux.from(rs);
Flux<MyBean> res1= flux.skip(offset).take(limit).map(rowMapper);

// Return Mono for single bean
Mono<MyBean> res2 = Mono.fromDirect(rs).map(rowMapper);
```

Cassandra and Spring

https://www.datastax.com/dev/spring



Build a Spring Boot REST Service

CONTRIBUTORS

This scenario is a brief introduction to Spring Boot so you can see how to create an initial project.

Do you want to set up your first Spring Boot project? Then, you're at the right place!

In this scenario, we'll learn how to:

- Create a Spring Boot project for a simple REST server
- Create some service endpoints
- Pass parameters to an endpoint
- Create a REST endpoint that uses Cassandra

This scenario is a brief introduction to Spring Boot so you can see how to create an initial project.

ver 0.001

START SCENARIO

TIME TO COMPLETE

15 minutes

DIFFICULTY

Beginner

SHARE

Hands-on

Sample Codes

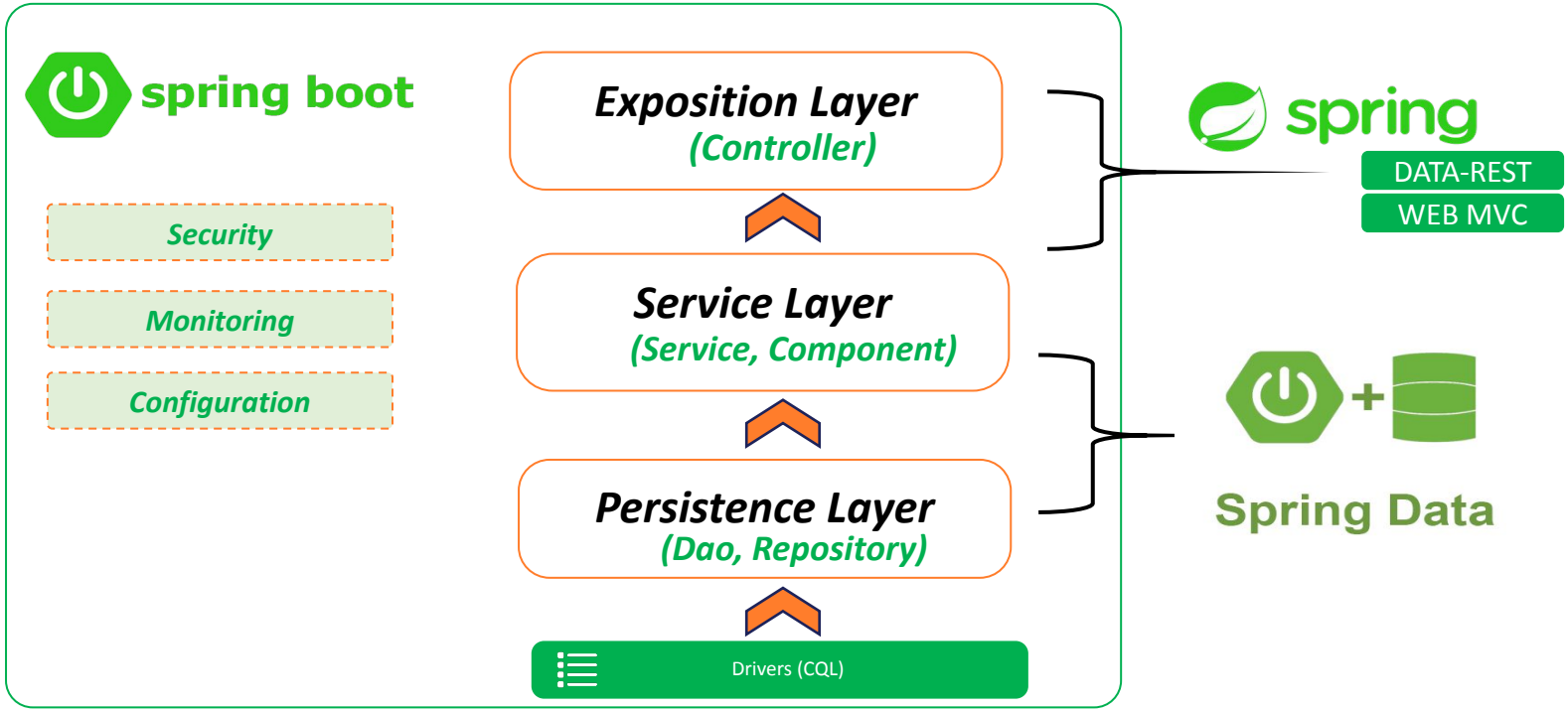
https://github.com/DataStax-Examples/java-cassandra-driver-from3x-to4x

https://github.com/datastaxdevs/conference-2020-javafest-api

https://github.com/DataStax-Academy/microservices-java-workshop-online

Cassandra and Spring Data

https://github.com/datastaxdevs/workshop-spring-stargate



Cassandra and Spring Data

```
Documents
```

Spring Data Cassandra & DataStax.pdf

Spring Data Cassandra 3.x and Drivers 4.x

Agenda (6H)

01 

Apache Cassandra™
Les fondamentaux

02 

Apache Cassandra™
Modélisation

03 

Apache Cassandra™
Développements

04 

Kubernetes
Les fondamentaux

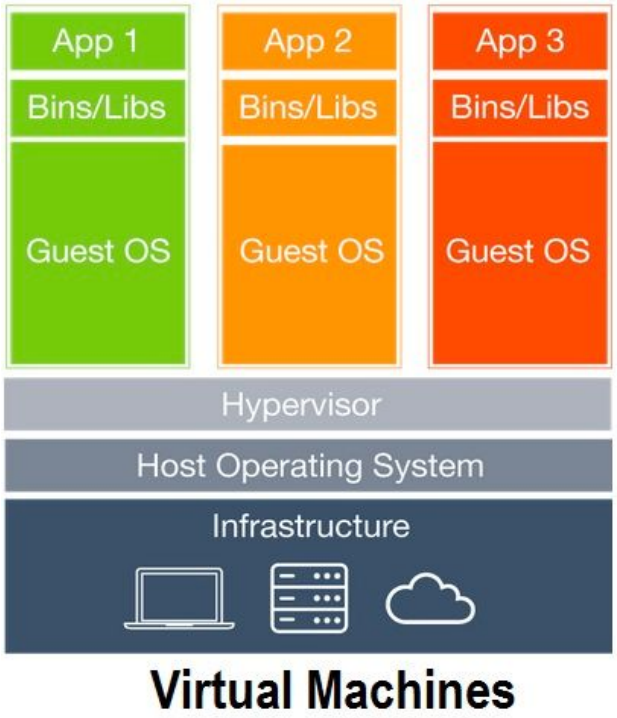
05 

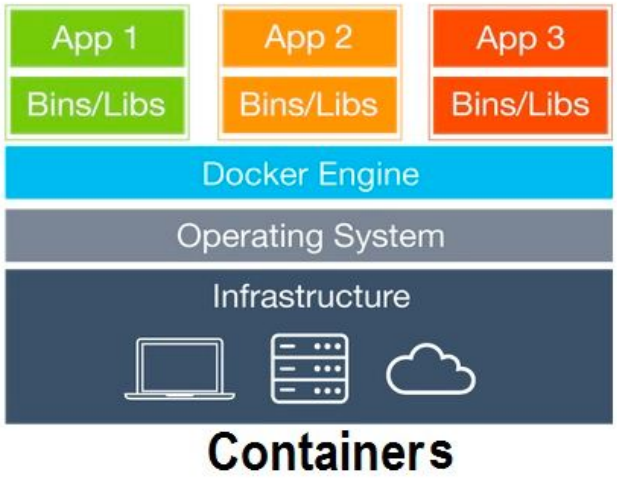
Kubernetes
Stateful & Cassandra

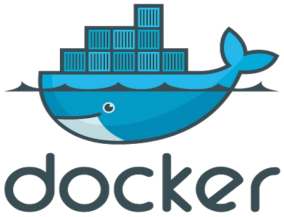
05 

K8ssandra
Tour d'horizon

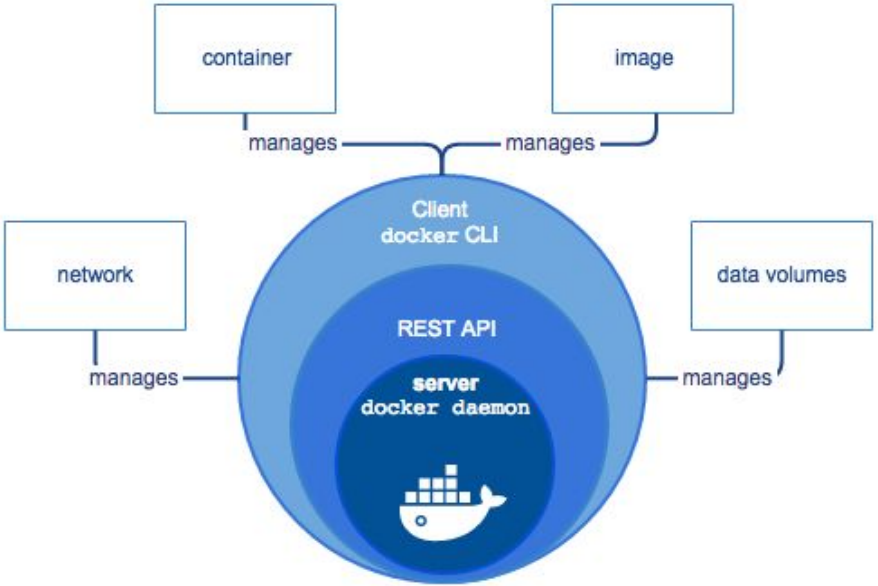
Why containers?







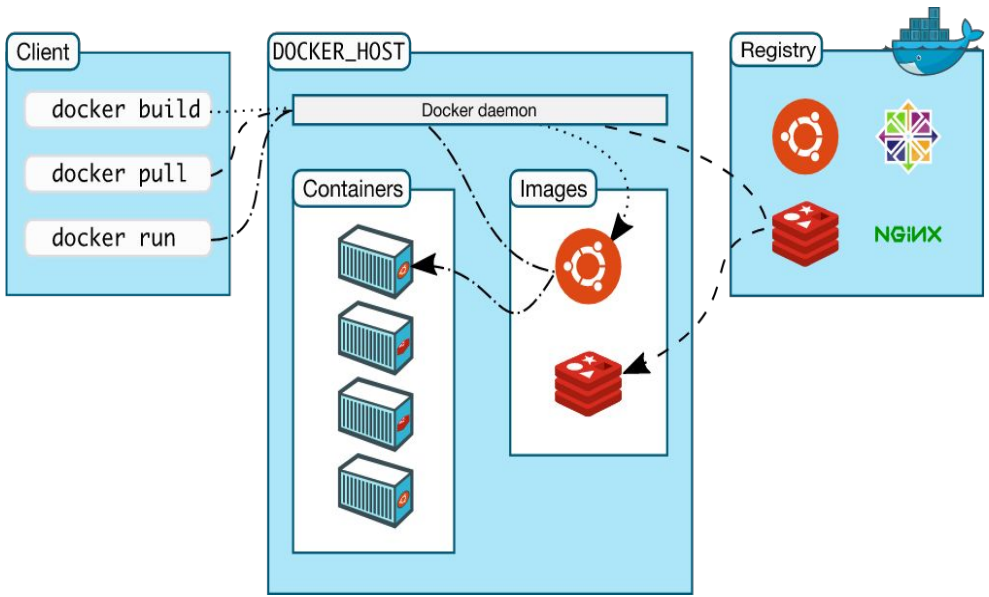
reference documentation : https://docs.docker.com/get-started/overview/



Docker Engine is Client-server application with :

- A server which is a type of long-running program called a daemon process **(the dockerd command).**
- **A REST API** which specifies interfaces that programs can use to talk to the daemon and instruct it what to do.
- **A command line interface (CLI)** client (the docker command).

Docker Architecture



Some Docker client commands

- We move immutable **images** (build/pull/push)
- Docker Daemon start containers based on images (run)
  - *Adding Env Var*
  - *Adding Storage (volumes)*
  - *Adding Network*
  - *Exposing port*
- We interact with **running containers** (ps, exec…)

cassandra ☆

Docker Official Images

Apache Cassandra is an open-source distributed storage system.

100M+

Container | Linux | PowerPC 64 LE | ARM | ARM 64 | 386 | x86-64 | Databases | Official Image

Linux - x86 (latest)

Copy and paste to pull this image

```
docker pull cassandra
```

View Available Tags

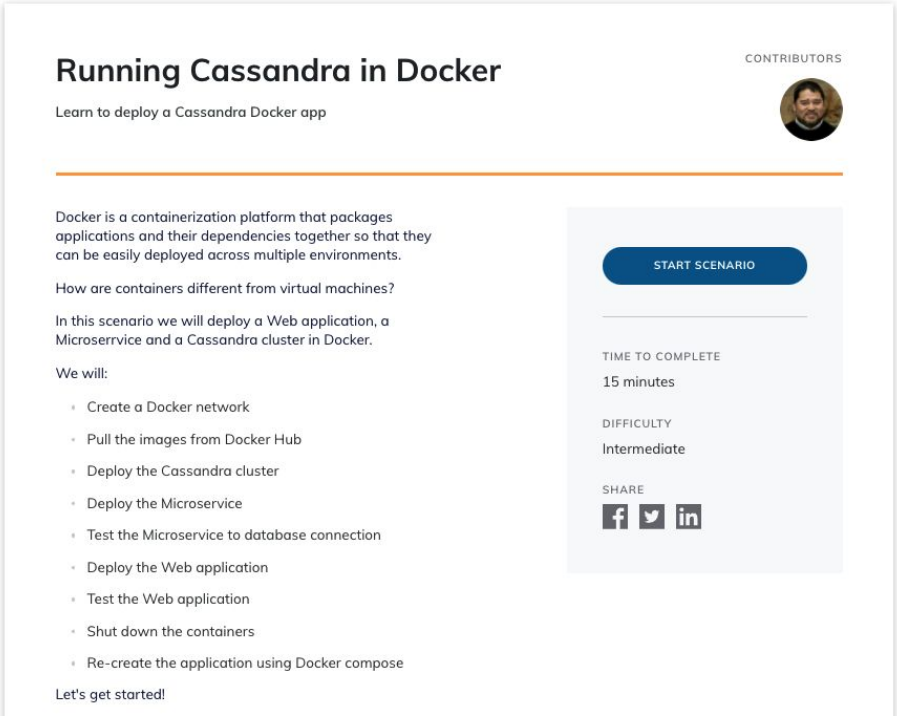
Running Cassandra in Docker

- Define a proper network
- **Env variables** can be defined to override keys in cassandra.yaml.
- Export ports **7000, 9042, ...**
- Define volumes to stores data
  - **/var/lib/cassandra**

```
$ docker run

  --name some-cassandra -d \

  -e CASSANDRA_BROADCAST_ADDRESS=10.42.42.42 \

  -p 7000:7000,9042:9042

  -v /my/own/datadir:/var/lib/cassandra \

  cassandra:tag
```

Hands-ON

https://github.com/datastaxdevs/conference-2021-apachecon-stargate



Running Cassandra in Docker

Learn to deploy a Cassandra Docker app

CONTRIBUTORS

Docker is a containerization platform that packages applications and their dependencies together so that they can be easily deployed across multiple environments.

How are containers different from virtual machines?

In this scenario we will deploy a Web application, a Microserrvice and a Cassandra cluster in Docker.

We will:

- Create a Docker network
- Pull the images from Docker Hub
- Deploy the Cassandra cluster
- Deploy the Microservice
- Test the Microservice to database connection
- Deploy the Web application
- Test the Web application
- Shut down the containers
- Re-create the application using Docker compose

Let's get started!

START SCENARIO

TIME TO COMPLETE

15 minutes

DIFFICULTY

Intermediate

SHARE

Docker-Compose

```
docker-compose -f docker-compose.yml up -d --scale cassandra-node=2
```

Define and run multi-container Docker applications through the use of a **YAML** file to configure your applications

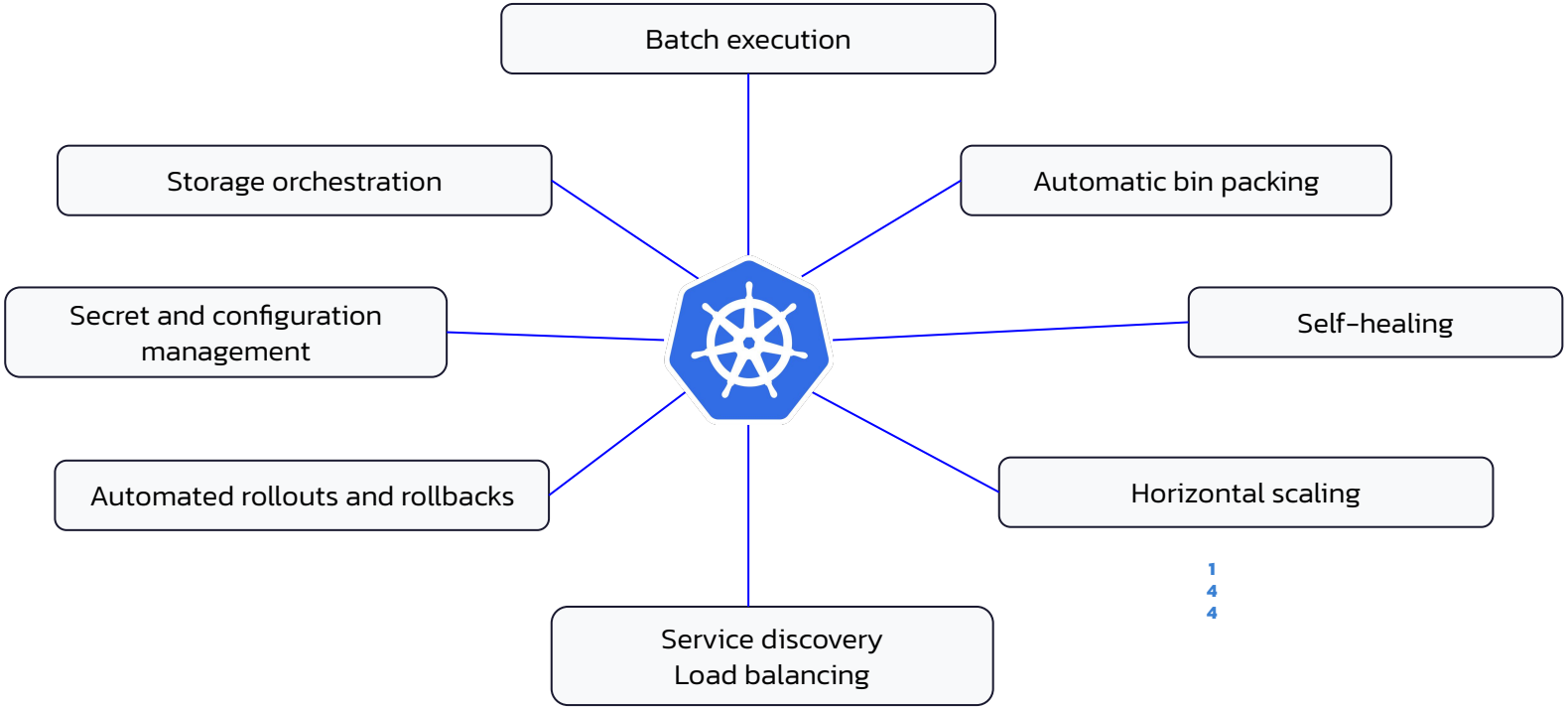
```
version: '2'
services:

  cassandra-seed:
    container_name: cassandra-seed-node
    image: cassandra: 3.11.6
    ports:
      - "9042:9042"   # Native transport
      - "7199:7199"   # JMX
      - "9160:9160"   # Thrift clients

  cassandra-node:
    image: cassandra: 3.11.6
    command: /bin/bash -c "echo 'Waiting for seed node' && sleep 30 && /docker-entrypoint.sh cassandra -f"
    environment:
      - "CASSANDRA_SEEDS=cassandra-seed-node"
    depends_on:
      - "cassandra-seed"
```

kubernetes

"Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications."

1
4
3

Why Kubernetes ?

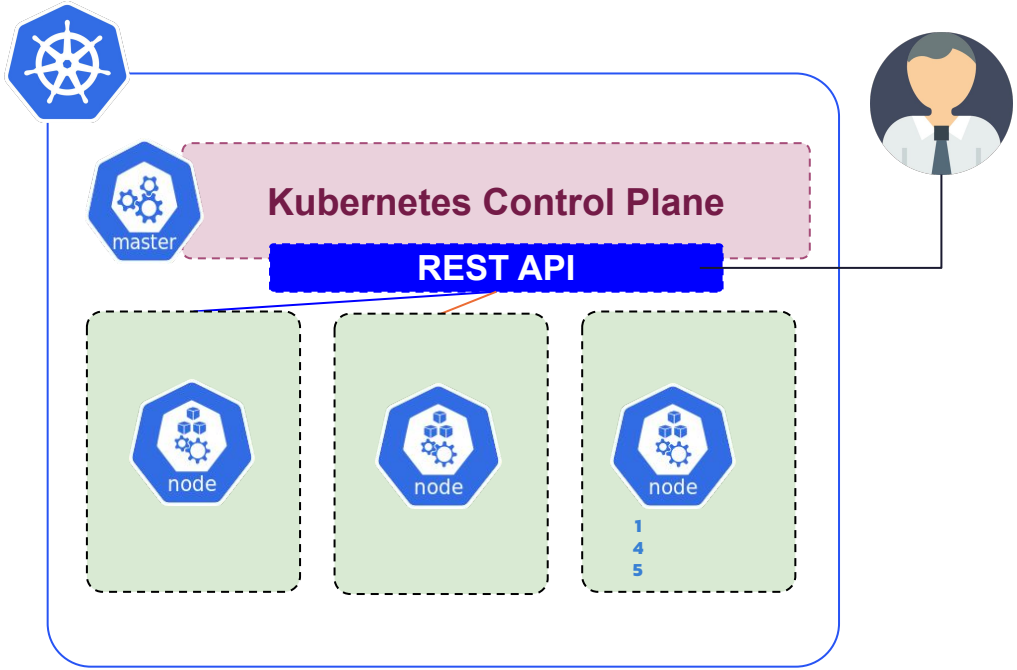


K8s Infrastructure

Cluster:
Kubernetes cluster.

Master:
Kubernetes Control Plane.

Node:
Worker machine



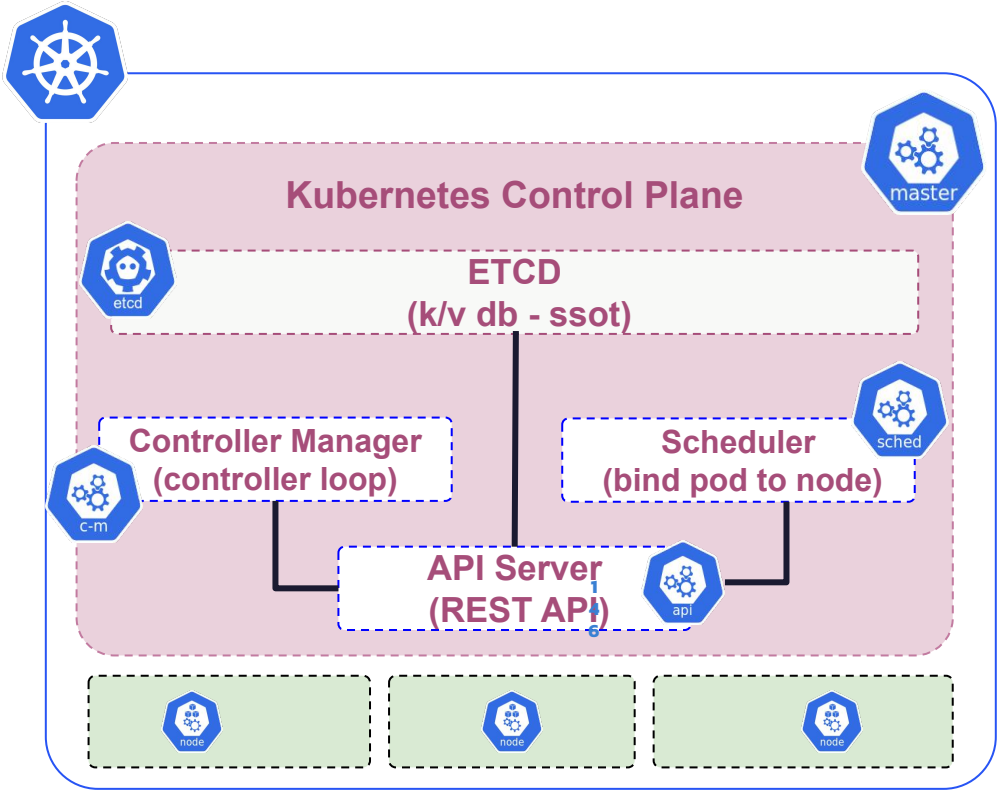
K8s Infrastructure – Control Plane

K8s API Server
Kubernetes API

Controller Manager
Kubernetes controller manager

Scheduler
In charge of placing pods

ETCD
Kubernetes' backing store.



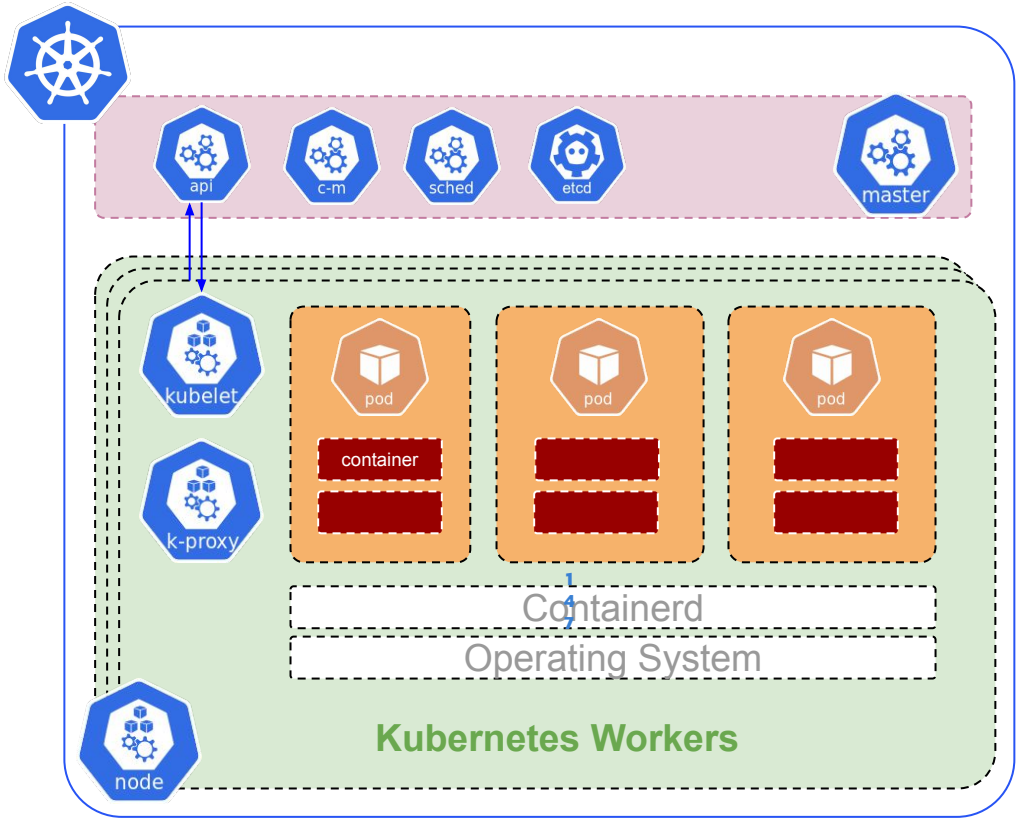
K8s Infrastructure - Worker Node

Kubelet:
The kubelet is the primary "node agent" that runs on each node.

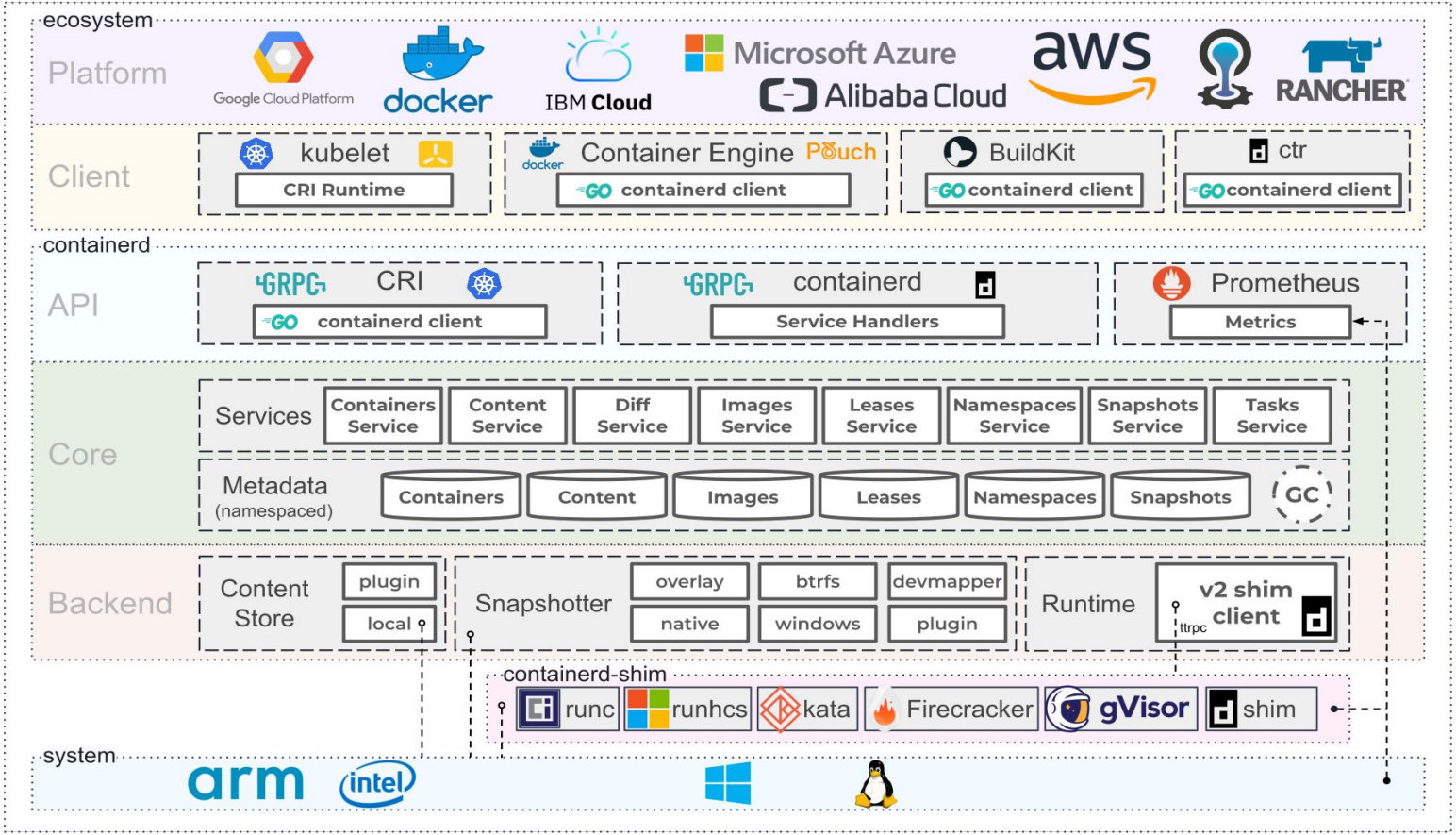
Kube-proxy
The Kubernetes network proxy runs on each node. This reflects services as defined in the Kubernetes API on each node.

POD
Collection of containers that can run on a host.

This resource is created by clients and scheduled onto hosts.



Open specification they said...



Kubernetes Distributions

Clouds

Local Environment

Azure

- Azure Kubernetes
  - Service (AKS)

vmware

Pivotal Container Service

ORACLE CLOUD

Red Hat CodeReady Containers

Amazon EKS

DigitalOcean

Alibaba Cloud | Worldwide Cloud Services Partner

kind

CLOUD FOUNDRY CONTAINER RUNTIME

OpenShift

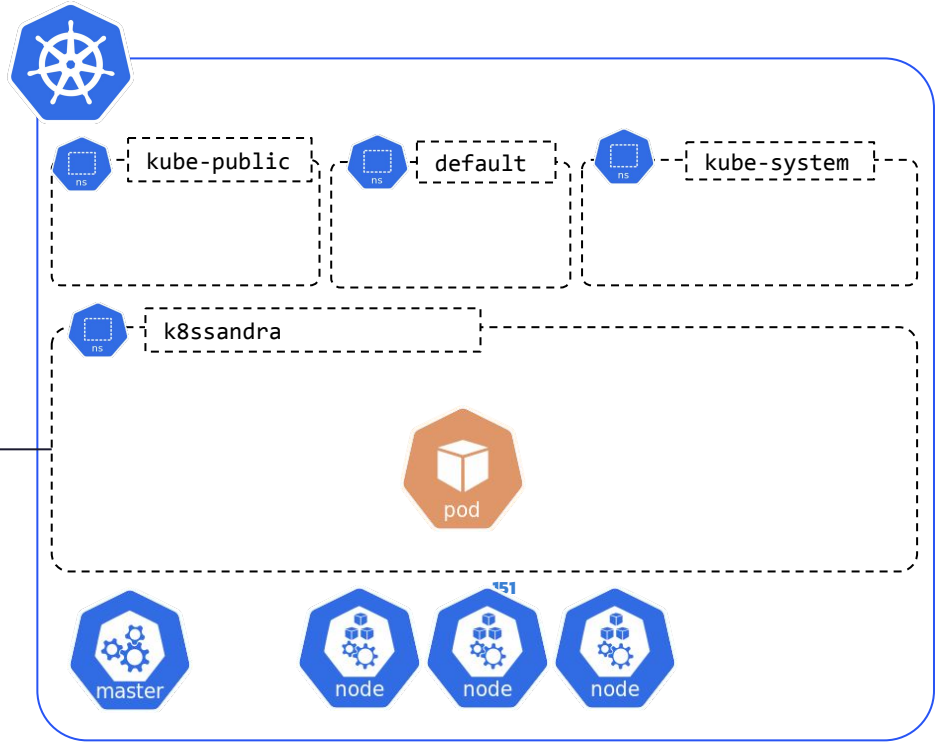
- Google Kubernetes
  - Engine (GKE)

IBM


I KNOW
KUBERNETES
imgflip.com

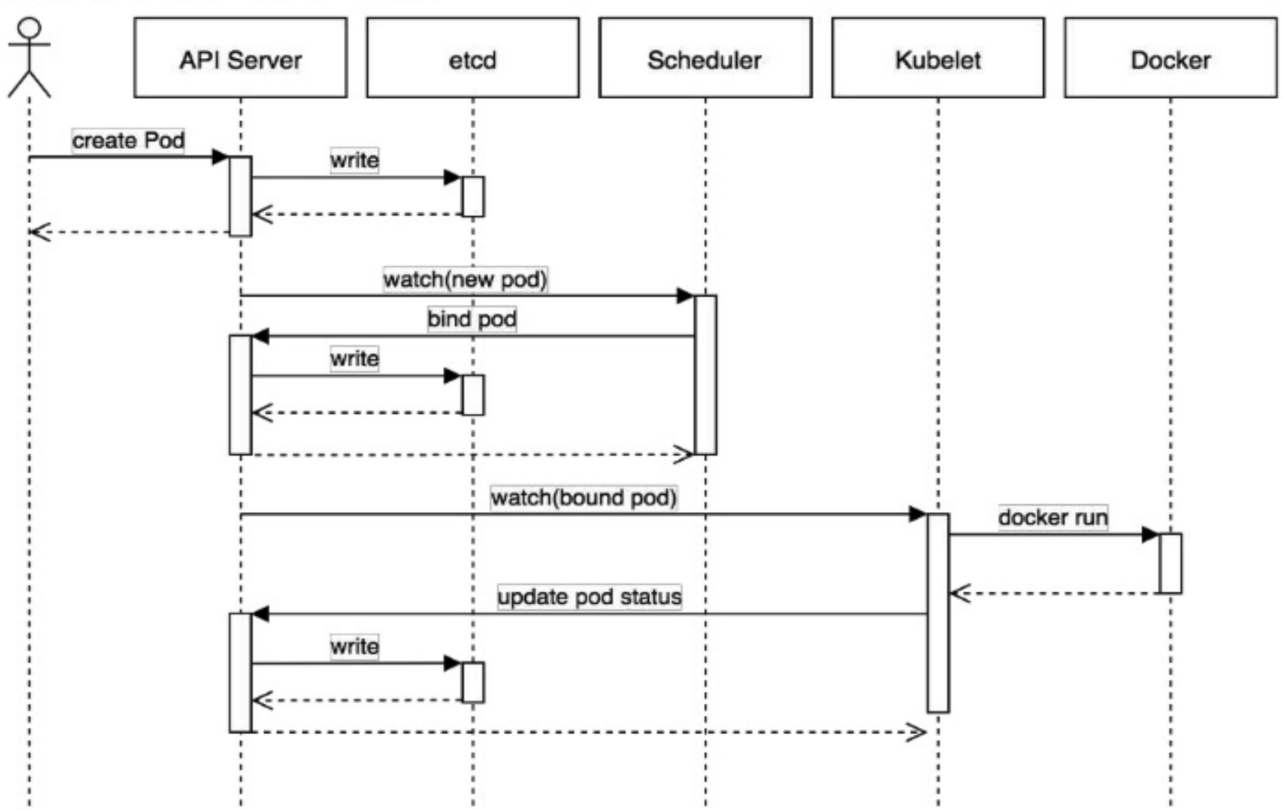
Kubernetes Namespaces

Namespace: Namespace provides a scope for Names. Use of multiple namespaces is optional.



We create resources in namespaces that span across nodes.

Pod Lifecycle

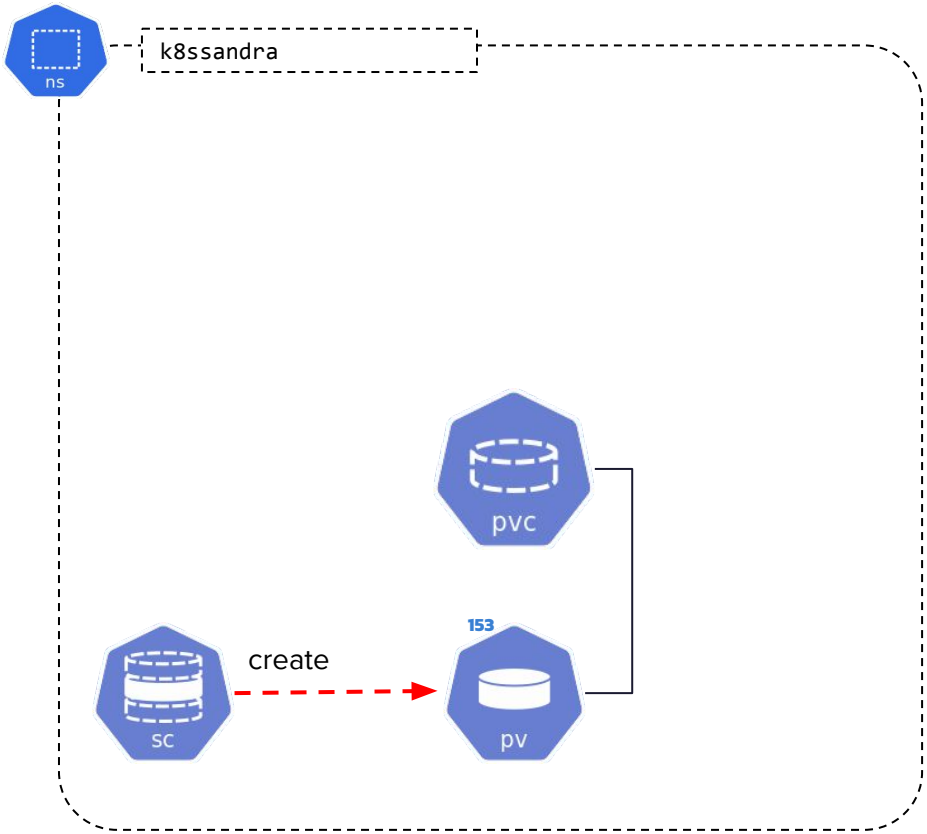


K8s Primitives : Storage

PersistentVolume: is a storage resource provisioned by an administrator or dynamic provisioner.

PersistentVolumeClaim: PersistentVolumeClaim is a user's request for and claim to a persistent volume.

StorageClass: StorageClass describes the parameters for a class of storage for which *PersistentVolumes* can be dynamically provisioned.



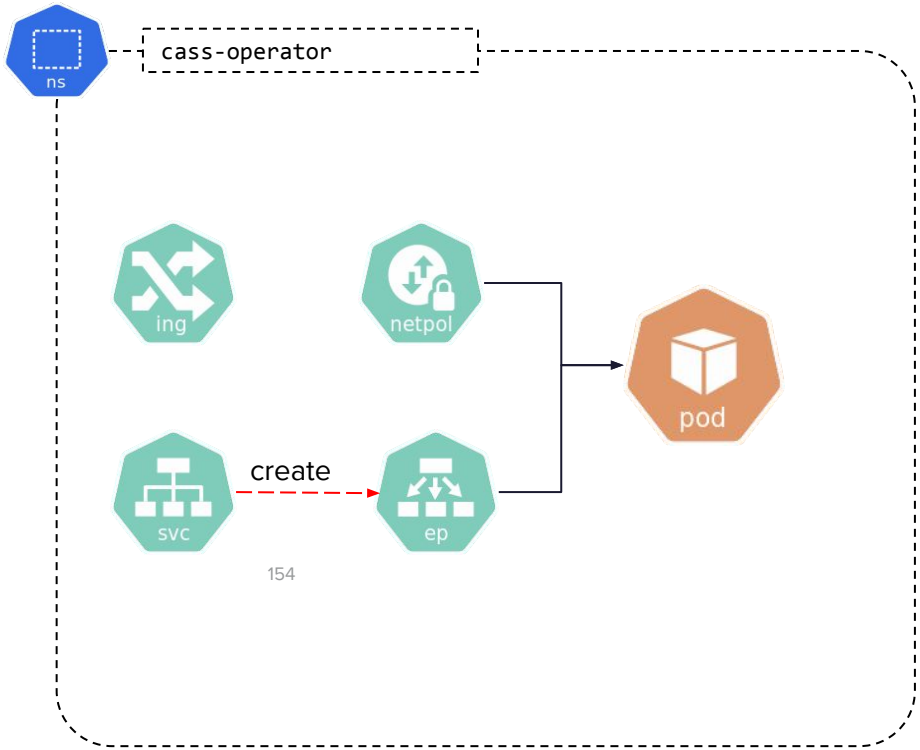
K8s Primitives : Service

Ingress is a collection of rules that allow inbound connections to reach the endpoints defined by a backend.

Service is a **named** abstraction of software service with ports to listen on and selector to determine which pods will answer requests.

EndPoint is a collection of endpoints that implement the actual service..

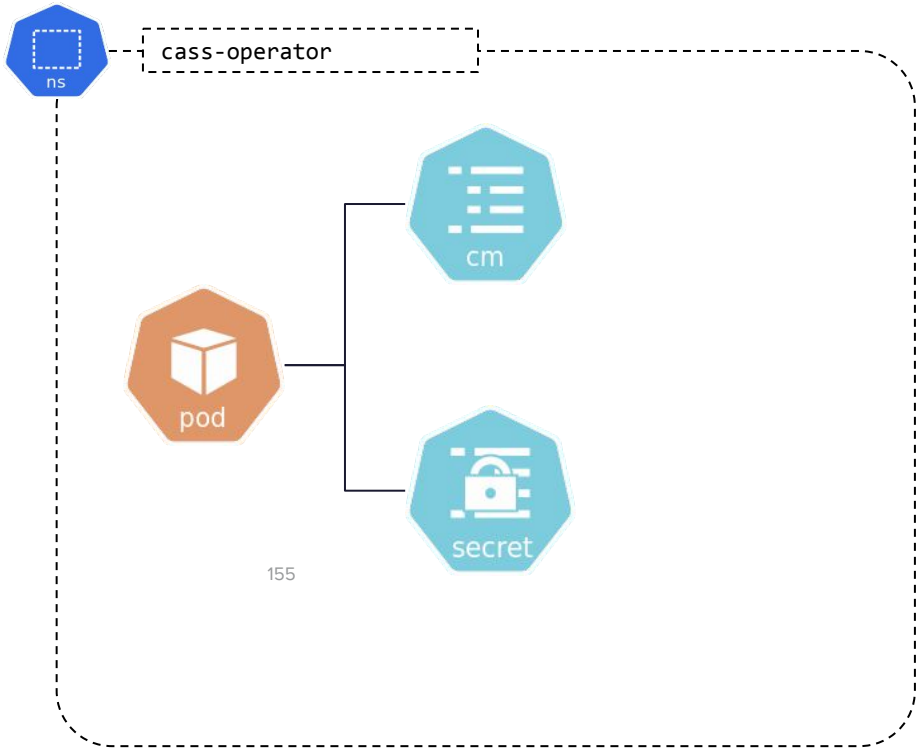
NetworkPolicy: Describes what network traffic is allowed for a set of Pods.



K8s Primitives : Configuration

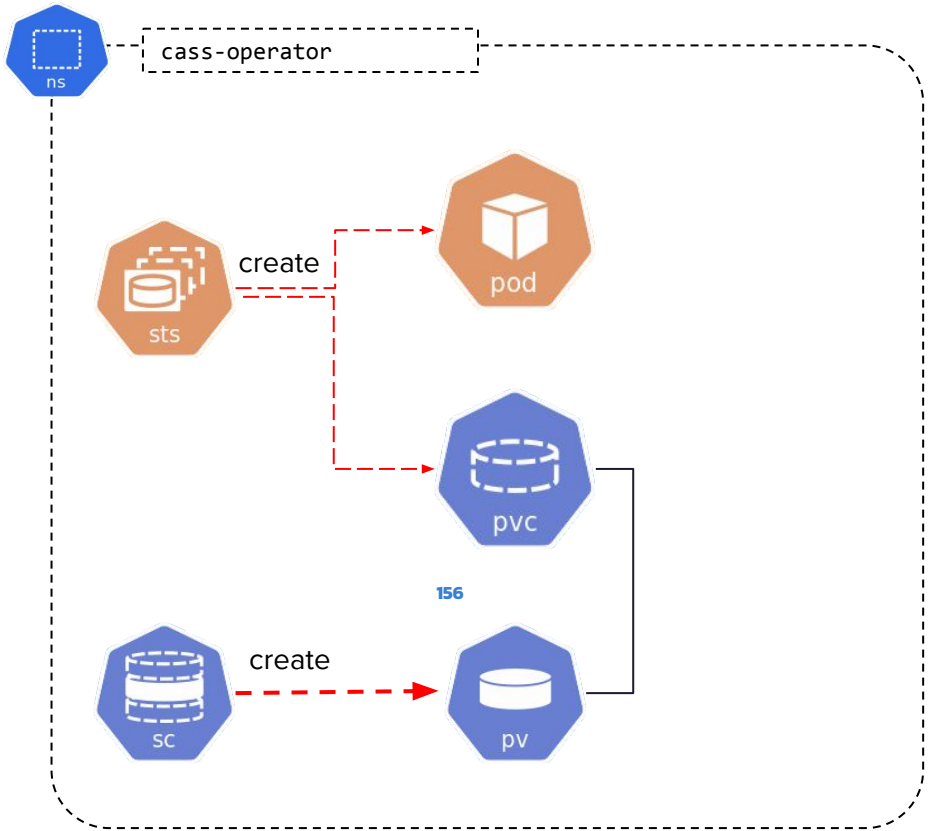
ConfigMap: ConfigMap holds configuration data for pods to consume..

Secret: Secret holds secret data of a certain type..

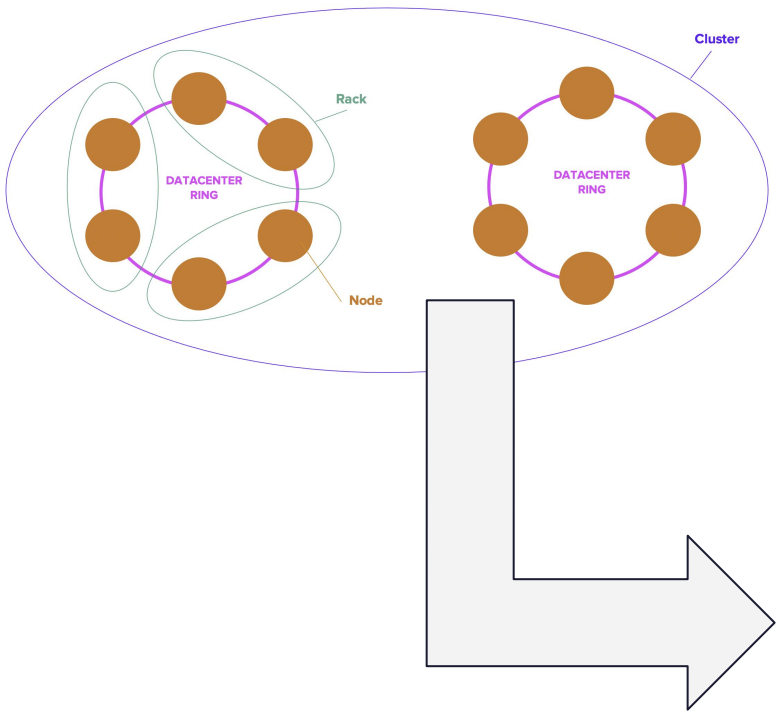


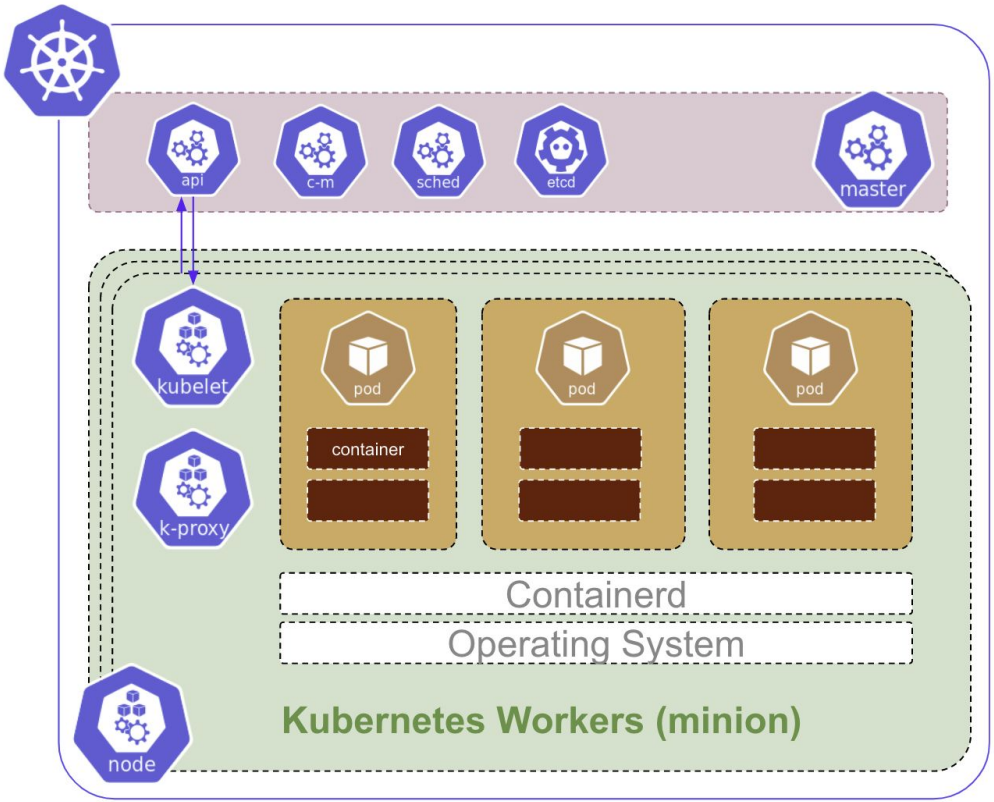
K8s Primitives : StatefulSet

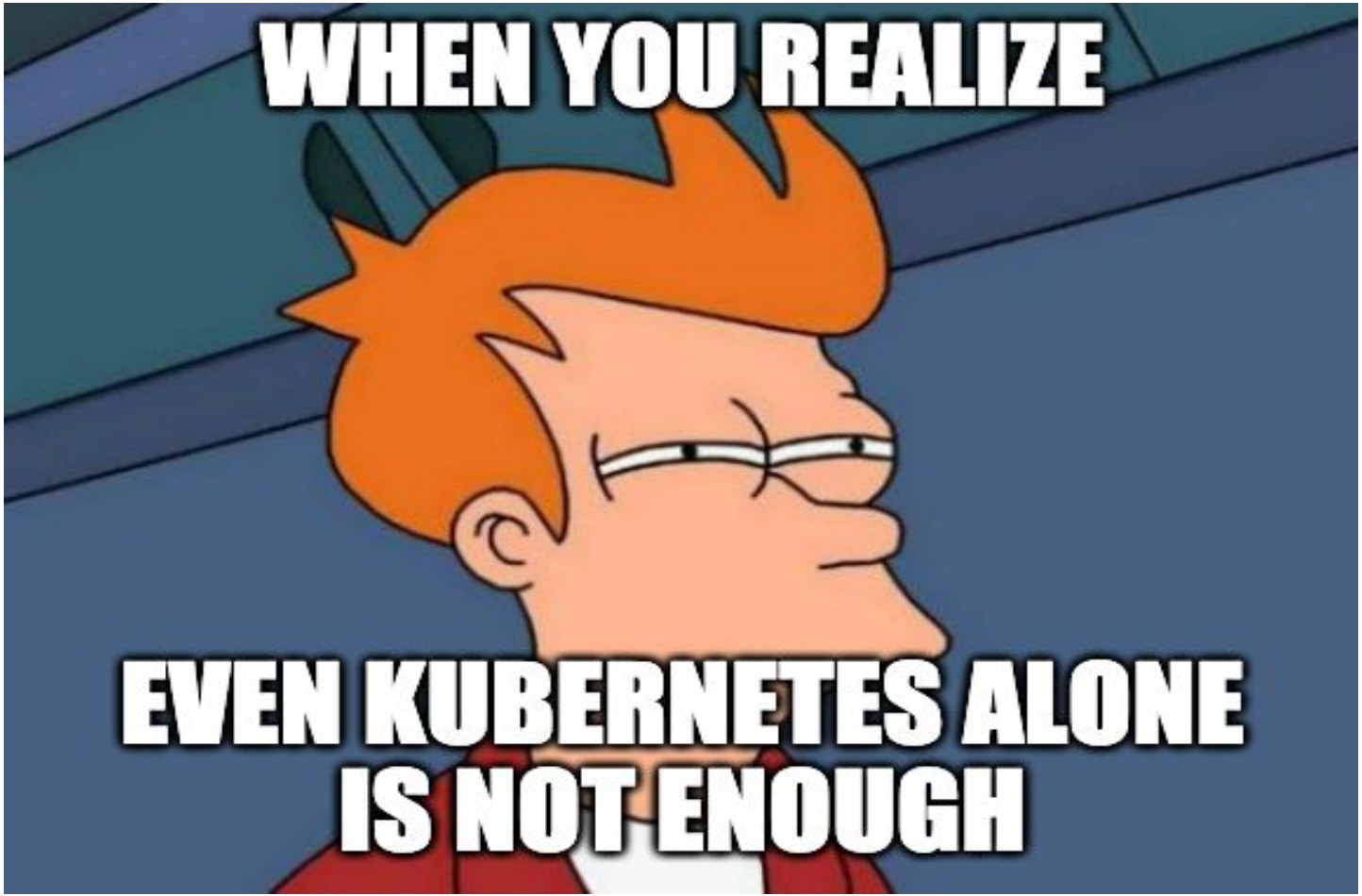
StatefulSet: StatefulSet represents a set of pods with consistent identities. Identities are defined as: network, storage.



Vous mission si vous décidez de l'accepter....






WHEN YOU REALIZE
EVEN KUBERNETES ALONE
IS NOT ENOUGH

Agenda (6H)

01

Apache Cassandra™
Les fondamentaux

02

Apache Cassandra™
Modélisation

03

Apache Cassandra™
Développements

04

Kubernetes
Les fondamentaux

05

Kubernetes
Stateful & Cassandra

05

K8ssandra
Tour d'horizon

K8s Primitives : Custom Resources

Custom Resource Definition :
Extensions of the Kubernetes API.
Customization making K8s more modular

- **Spec** declares the desired state of a resource
  - **Configuration settings** provided by the user
  - **Default values** expanded by the system
  - **Other properties** initialized by other internal components after resource creation.

- **Status** : describes the object's current, observed state.
  - Kubernetes API server provides a REST API to clients. A Kubernetes object or resource is a REST resource.
  - The status of a Kubernetes resource is typically implemented as a **REST subresource** that can only be modified by internal, system components

```yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: cassandradatacenters.cassandra.datastax.com
spec:
  group: cassandra.datastax.com
  names:
    kind: CassandraDatacenter
    listKind: CassandraDatacenterList
    plural: CassandraDatacenters
    shortNames:
      - cassdc
      - cassdcs
    singular: CassandraDatacenter
  scope: Namespaced
subresources:
  status: {}
...
```

- **Examples used in K8ssandra:**
  - `CassandraDataCenter` **(cass-operator)**
  - `CassandraBackup`, `CassandraRestore` **(cassandra-medusa)**
  - `Reaper` **(reaper)**
  - `GrafanaDataSource` **(grafana)**

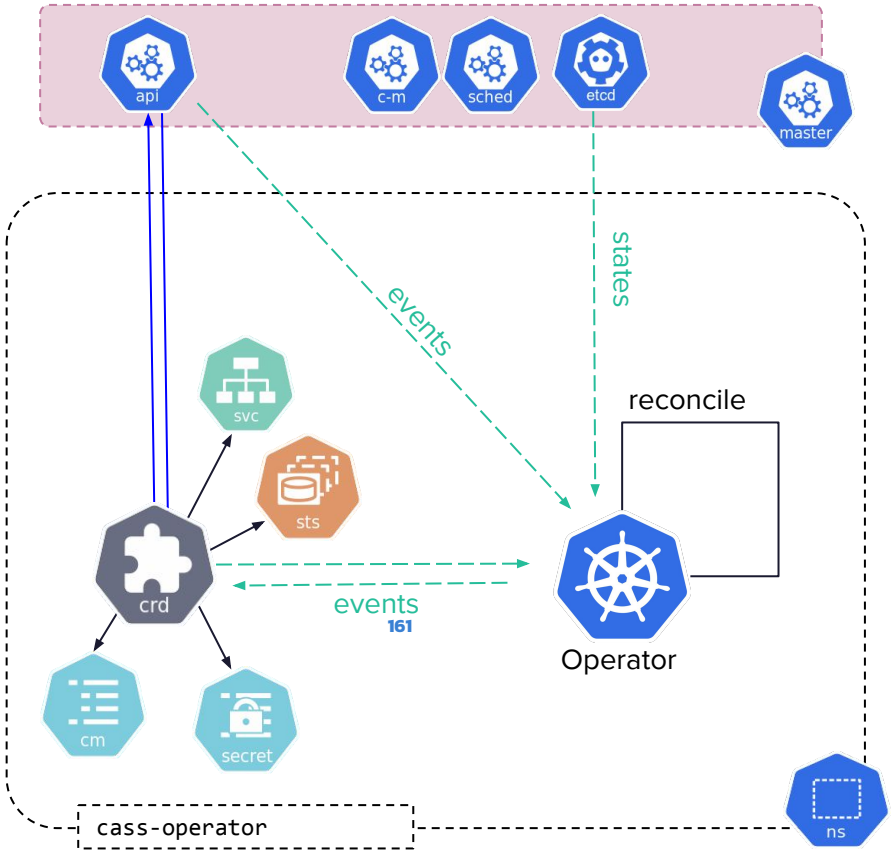
K8s Primitives : Operator

Building an application and driving an application on top of Kubernetes, behind Kubernetes APIs

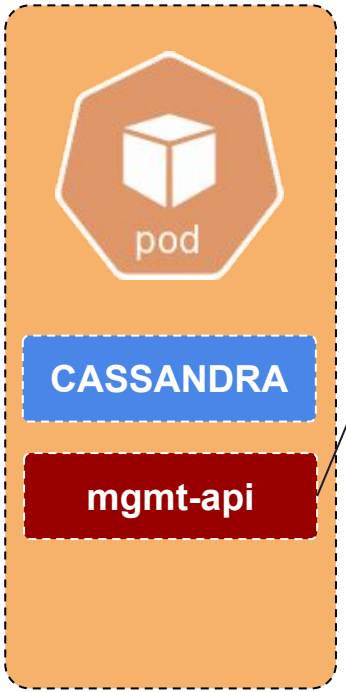
A Kubernetes Operator helps extend the types of applications that can run on Kubernetes by allowing developers to provide additional knowledge to applications that need to maintain state." –Jonathan S. Katz

- Reconcile CRD instances which states defined within the "**spec**" attribute.
- Listen for **events** and **status evolution** to react accordingly.

- **Examples used in K8ssandra:**
  - `cass-operator` (cass-operator)
  - `reaper-operator` (cassandra-reaper)
  - `PrometheusOperator` (Prometheus)



Kubernetes Probes (readiness, liveness)



Cassandra Management API Service

https://github.com/datastax/management-api-for-apache-cassandra

Management API for Apache Cassandra

0.1 OAS3

https://raw.githubusercontent.com/datastax/management-api-for-apache-cassandra/master/management-api-server/doc/openapi.json

This is a Restful service for operating Apache Cassandra. You can find out more about the Management API on Github

Apache 2.0

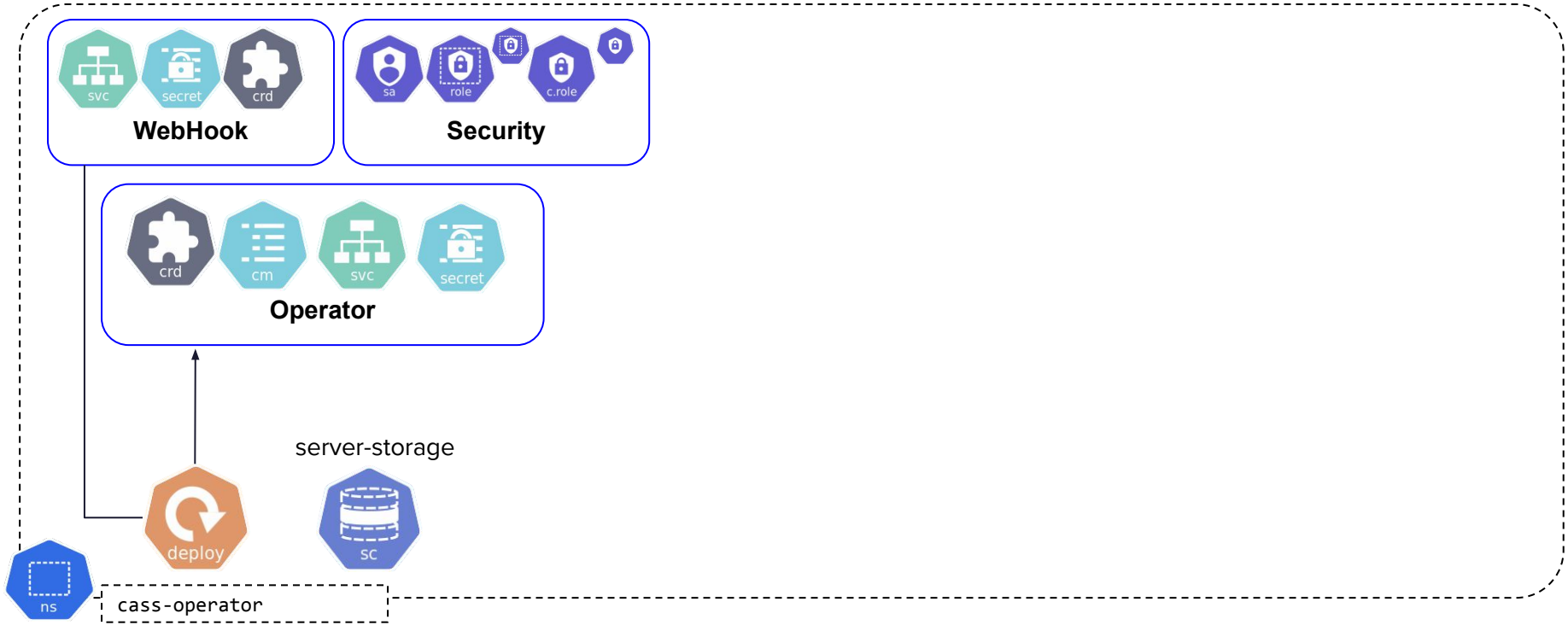
default

POST	/api/v0/ops/auth/role	Creates a new user role
GET	/api/v0/probes/liveness	Indicates whether this service is running
GET	/api/v0/probes/readiness	Indicates whether the Cassandra service is ready to service requests
GET	/api/v0/probes/cluster	Indicated whether the Cassandra cluster is able to achieve the specified consistency
POST	/api/v0/ops/seeds/reload	
POST	/api/v0/ops/keyspace/refresh	Load newly placed SSTables to the system without restart
POST	/api/v0/ops/keyspace/cleanup	Triggers the immediate cleanup of keys no longer belonging to a node. By default, clean all keyspaces
POST	/api/v0/lifecycle/start	
POST	/api/v0/lifecycle/stop	
POST	/api/v0/lifecycle/configure	
GET	/api/v0/lifecycle/pid	
GET	/api/v0/metadata/versions/release	Returns the Cassandra release version
GET	/api/v0/metadata/endpoints	Returns this nodes view of the endpoint states of nodes
POST	/api/v0/ops/node/drain	Drain the node (stop accepting writes and flush all tables)

Cass Operator : Features

- Proper token ring initialization, with only one node bootstrapping at a time
- Seed node management -
  - one per rack, or three per datacenter, whichever is more
- Server configuration integrated into the CassandraDatacenter CRD
  - Rolling reboot nodes by changing the CRD
  - Store data in a rack-safe way - one replica per cloud AZ
  - Scale up racks evenly with new nodes
  - Replace dead/unrecoverable nodes
- Multi DC clusters (limited to one Kubernetes namespace)

Installing the Cass Operator Manifest

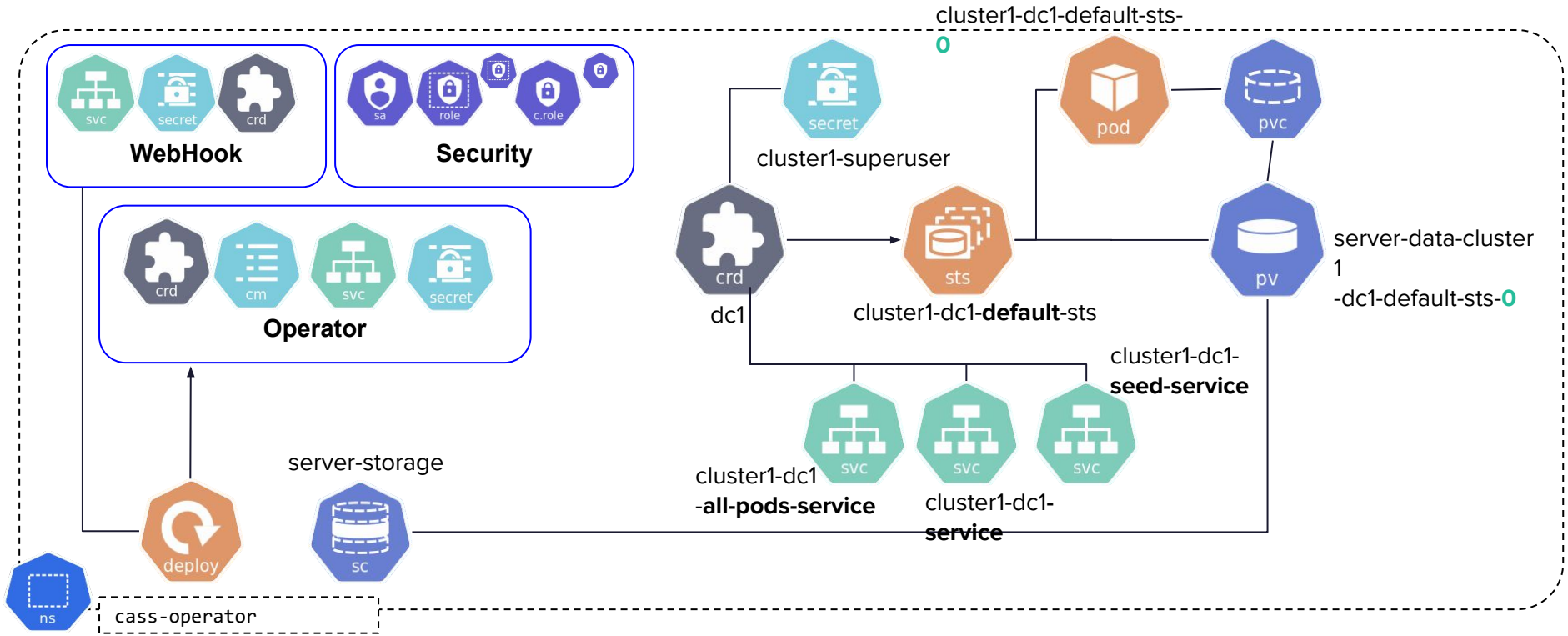


CRD CassandraDataCenter

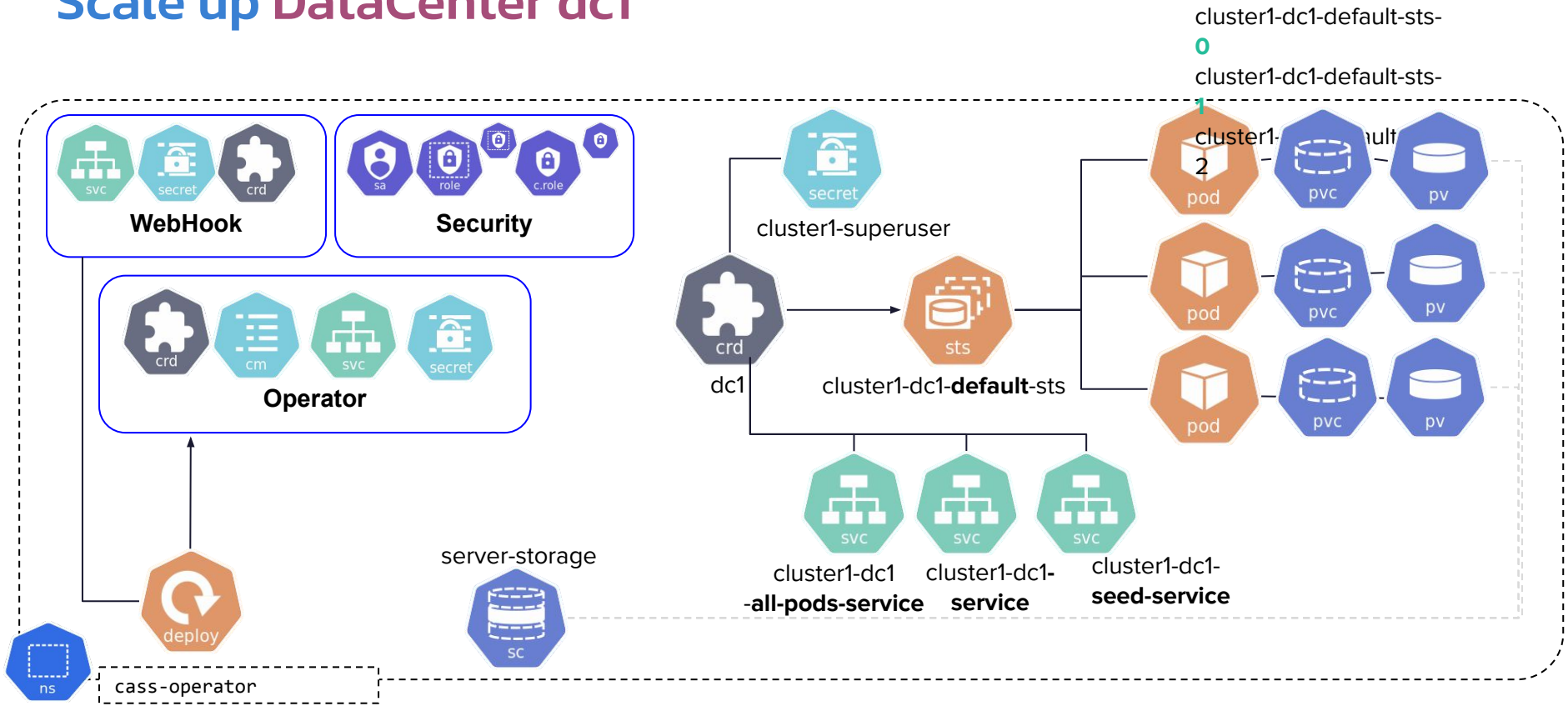
```yaml
apiVersion: cassandra.datastax.com/v1beta1
kind: CassandraDatacenter
metadata:
  name: dc1
spec:
  clusterName: cluster1
  serverType: cassandra
  serverVersion: "3.11.6"
  managementApiAuth:
    insecure: {}
  size: 1
  storageConfig:
    cassandraDataVolumeClaimSpec:
      storageClassName: server-storage
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 5Gi
  config:
    cassandra-yaml:
      authenticator: org.apache.cassandra.auth.PasswordAuthenticator
      authorizer: org.apache.cassandra.auth.CassandraAuthorizer
      role_manager: org.apache.cassandra.auth.CassandraRoleManager
    jvm-options:
      initial_heap_size: "800M"
      max_heap_size: "800M"
```

```yaml
apiVersion: cassandra.datastax.com/v1beta1
kind: CassandraDatacenter
metadata:
  name: multi-rack
spec:
  clusterName: multi-rack
  serverType: cassandra
  serverVersion: 3.11.6
  managementApiAuth:
    insecure: {}
  size: 9
  racks:
  - name: us-east1-b
    zone: us-east1-b
  - name: us-east1-c
    zone: us-east1-c
  - name: us-east1-d
    zone: us-east1-d
  storageConfig:
    cassandraDataVolumeClaimSpec:
      storageClassName: standard
      accessModes:
      - ReadWriteOnce
      resources:
        requests:
          storage: 5Gi
```

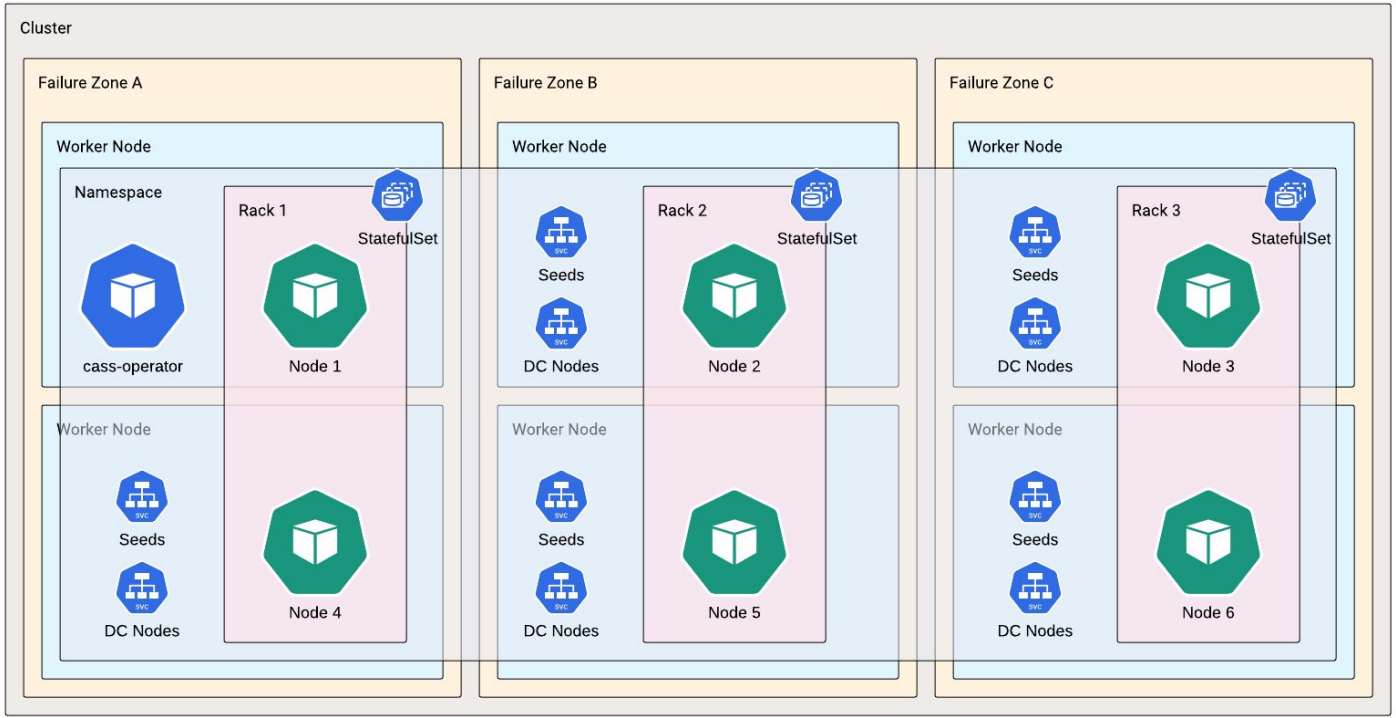
Creating DataCenter dc1



Scale up DataCenter dc1

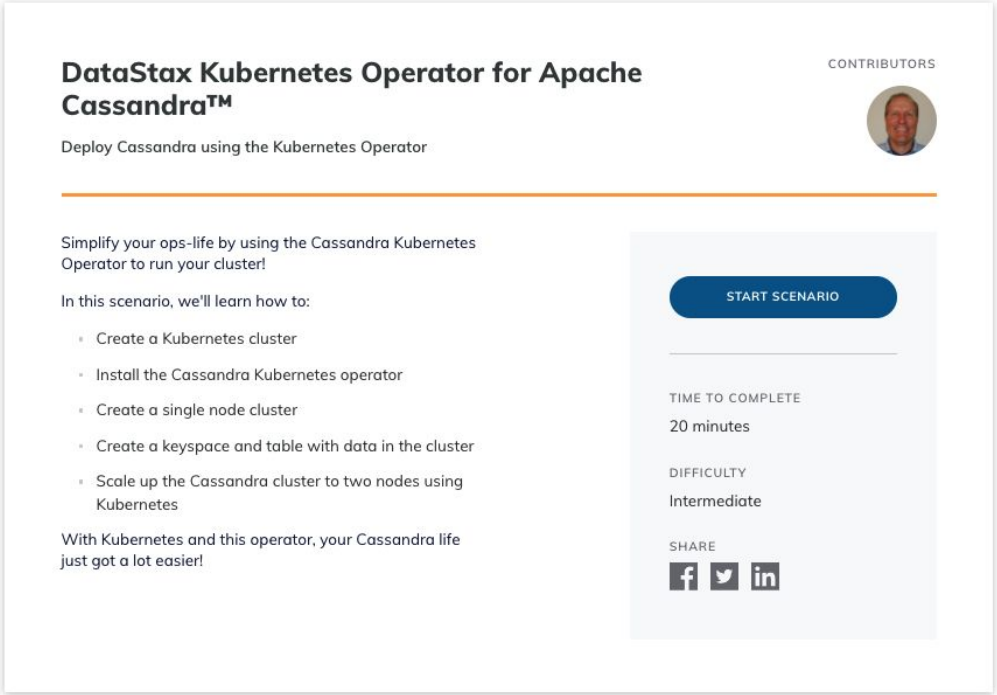


Example Deployment Across Multiple K8s Workers



Hands-ON

https://www.datastax.com/dev/kubernetes



DataStax Kubernetes Operator for Apache Cassandra™

Deploy Cassandra using the Kubernetes Operator

CONTRIBUTORS

Simplify your ops-life by using the Cassandra Kubernetes Operator to run your cluster!

In this scenario, we'll learn how to:

- Create a Kubernetes cluster
- Install the Cassandra Kubernetes operator
- Create a single node cluster
- Create a keyspace and table with data in the cluster
- Scale up the Cassandra cluster to two nodes using Kubernetes

With Kubernetes and this operator, your Cassandra life just got a lot easier!

START SCENARIO

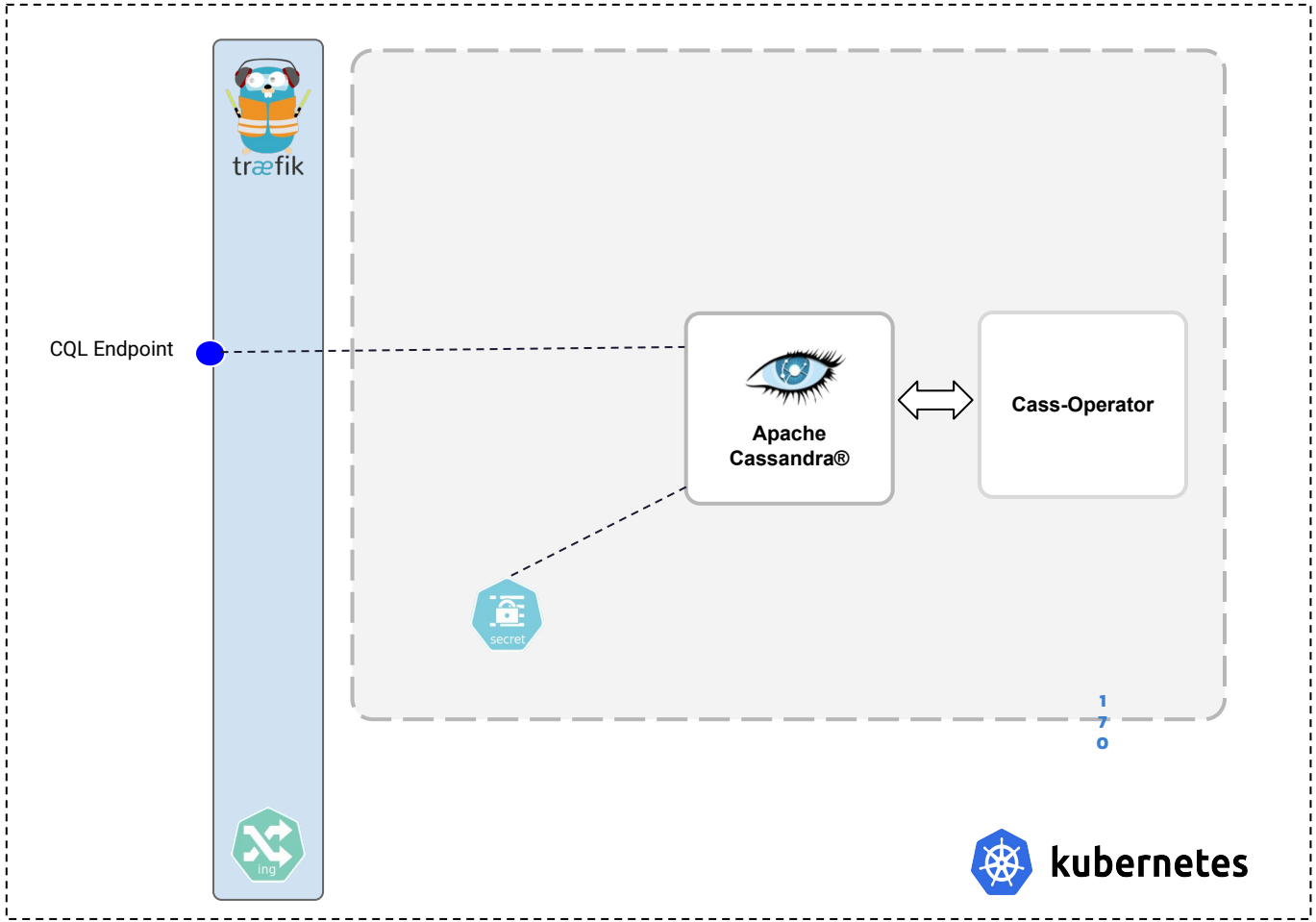
TIME TO COMPLETE

20 minutes

DIFFICULTY

Intermediate

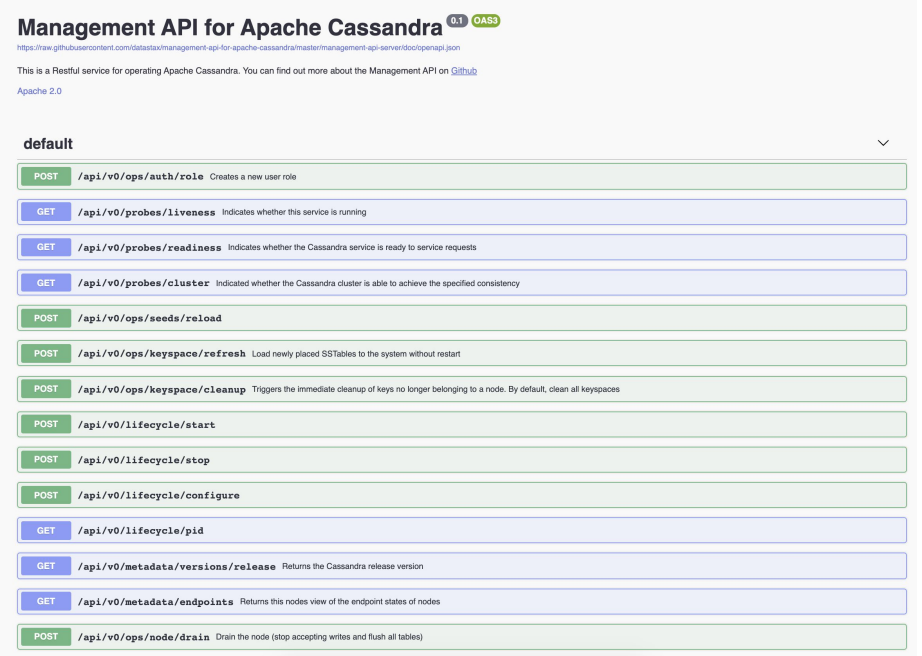
SHARE


træfik
CQL Endpoint
Apache Cassandra®
Cass-Operator
secret
ing
kubernetes

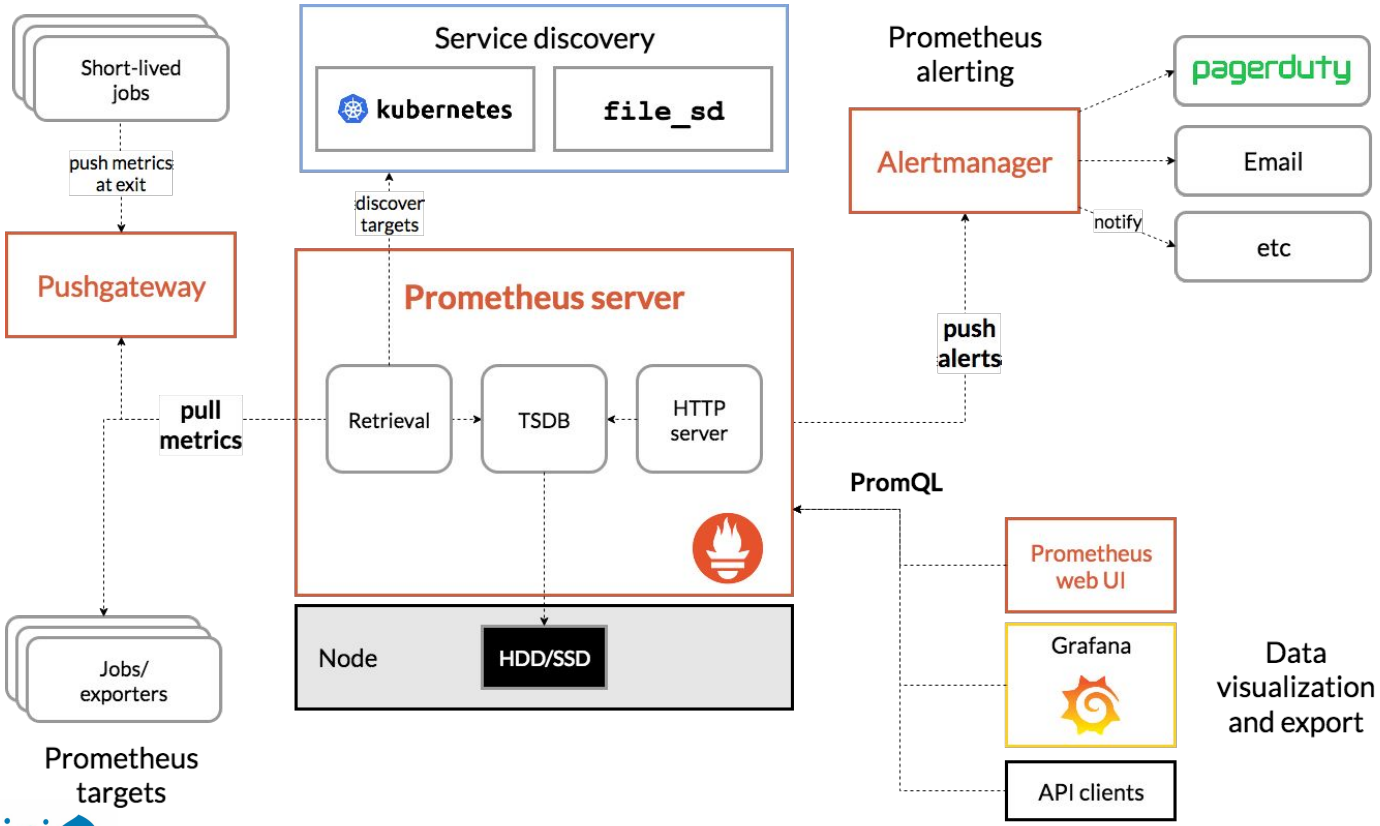
MCAC = Cassandra Management API SideCar

The Management API is a sidecar service layer that attempts to build a well supported set of operational actions on Cassandra® nodes that can be administered centrally. It currently works with official Apache Cassandra® 3.11.x an 4.0 via a drop in java agent.

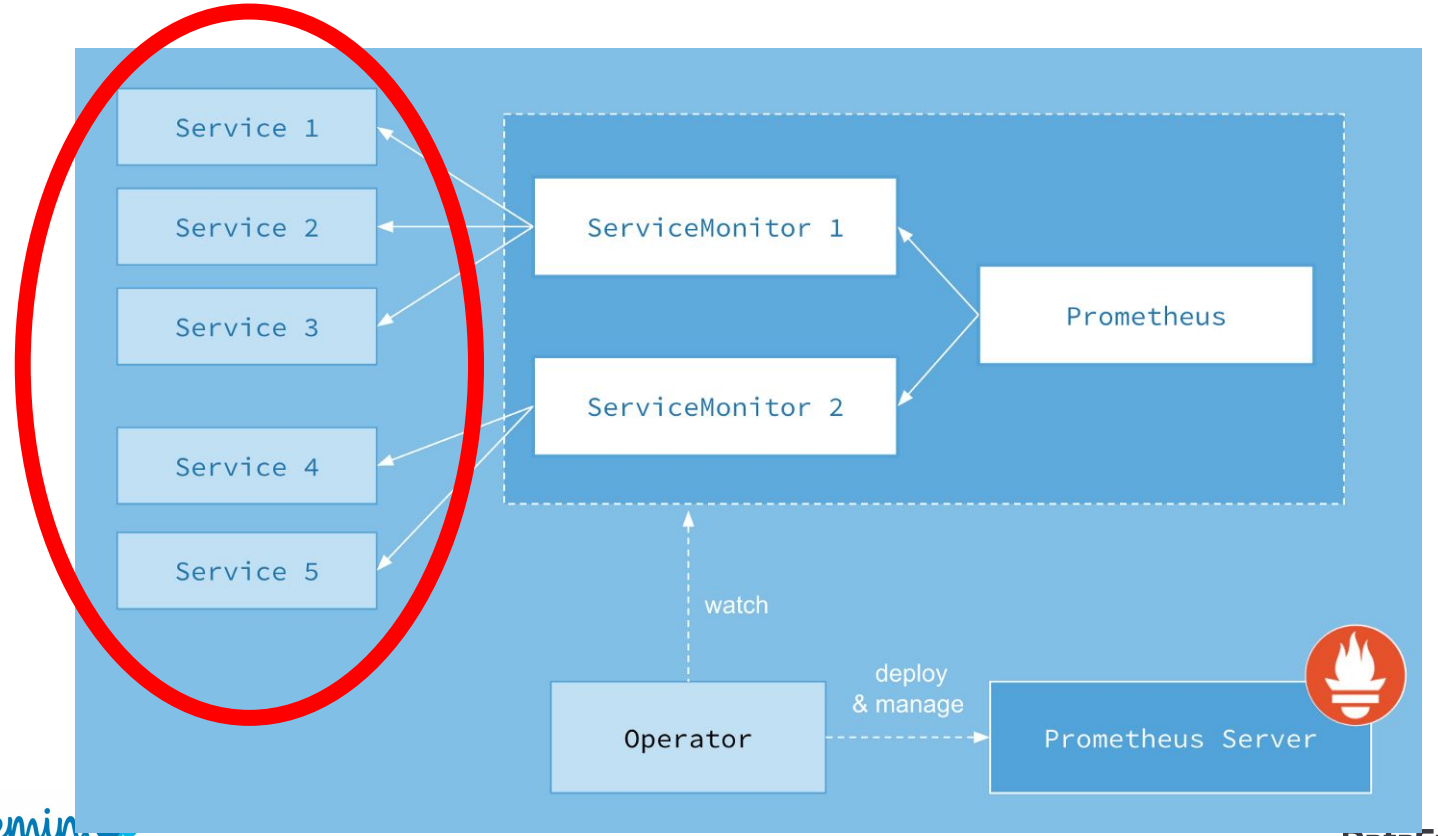
- Lifecycle Management
  - Start Node
  - Stop Node
- Configuration Management (alpha)
  - Change YAML
  - Change jvm-opts
- Health Checks
  - Kubernetes liveness/readiness checks
  - Consistency level checks
- Per node actions
  - All nodetool commands

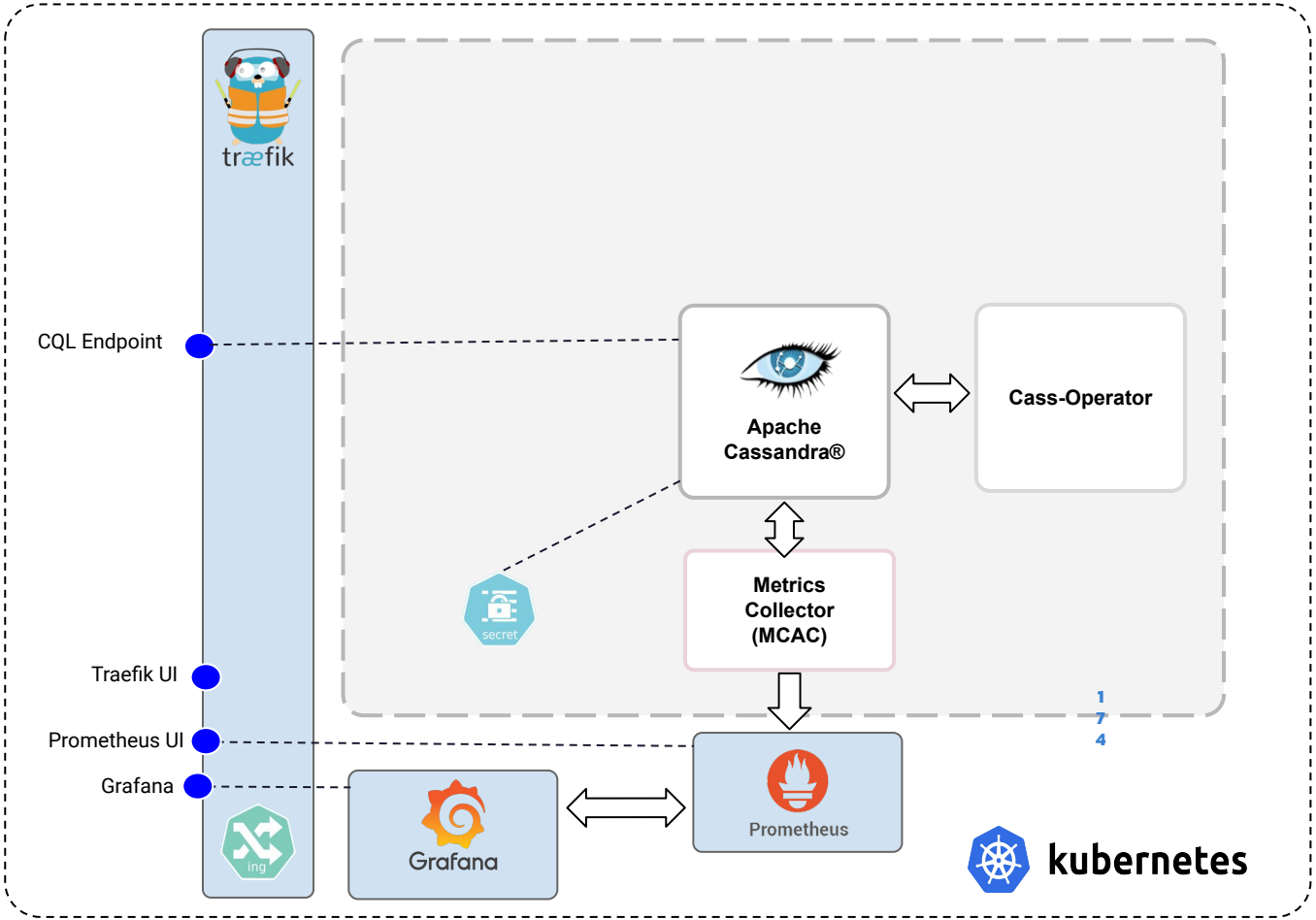


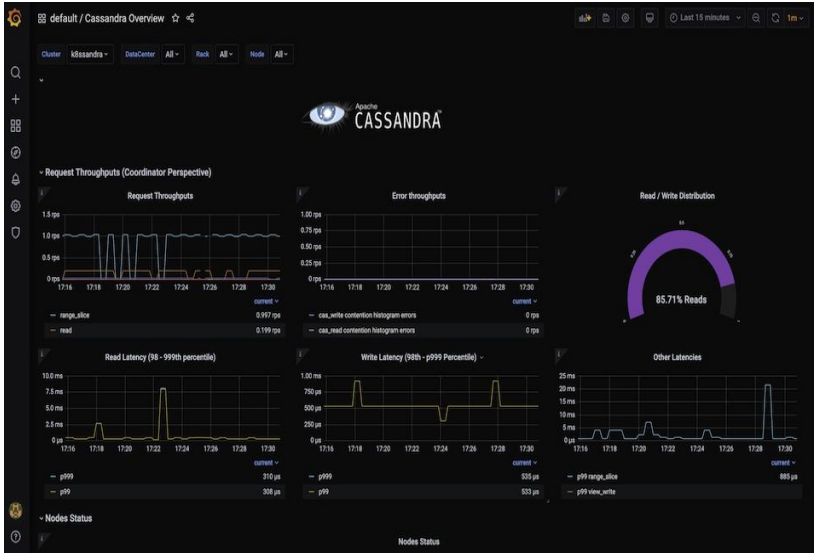
Prometheus + Grafana

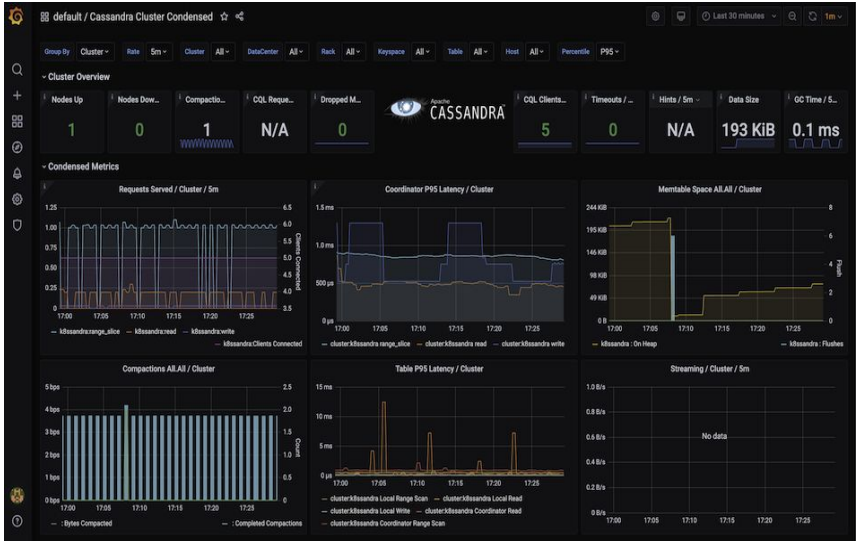


Prometheus metrics collection

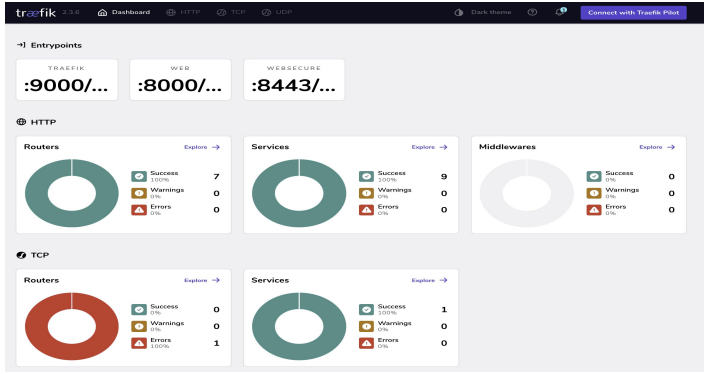



træfik
CQL Endpoint
Traefik UI
Prometheus UI
Grafana
ing
secret
Apache Cassandra®
Cass-Operator
Metrics Collector (MCAC)
Grafana
Prometheus
kubernetes


default / Cassandra Overview
Request Throughputs (Coordinator Perspective)
Request Throughputs
Error throughputs
Read / Write Distribution
85.71% Reads
Read Latency (98 - 99th percentile)
Write Latency (98th - p999 Percentile)
Other Latencies
Nodes Status


default / Cassandra Cluster Condensed
Cluster Overview
Nodes Up
Nodes Dow...
Compactio...
CQL Reque...
Dropped M...
CQL Clients...
Timeouts / ...
Hints / 5m
Data Size
GC Time / 5...
1
0
1
N/A
0
5
0
N/A
193 KiB
0.1 ms
Condensed Metrics
Requests Served / Cluster / 5m
Coordinator P95 Latency / Cluster
Memtable Space All.All / Cluster
Compactions All.All / Cluster
Table P95 Latency / Cluster
Streaming / Cluster / 5m
No data


default / System & Node Metrics
Basic CPU / Mem / Disk Gauge
CPU Busy
Memory Used
Swap Used
Disk Used
CPU System Load (1m avg)
CPU System Load (5m avg)
10%
68%
N/A
N/A
17%
19%
Basic CPU / Mem / Disk Info
CPU Cores
Total RAM
Total Swap
Total RootFS
System Load (1m avg)
System Uptime
4
15.5 GiB
0 B
N/A
0.70
2.4 hour
Basic CPU / Mem Graph
CPU Basic
Basic memory usage
Basic Network / Disk Graph
Network Traffic / Second
Network Packets / Second


traefik
Dashboard
Connect with Traefik Pilot
Entrypoints
TRAEFIK
:9000/...
WEB
:8000/...
WEBSECURE
:8443/...
HTTP
Routers
Services
Middlewares
TCP
Routers
Services

Agenda (6H)

01 

Apache Cassandra™
Les fondamentaux

02 

Apache Cassandra™
Modélisation

03 

Apache Cassandra™
Développements

04 

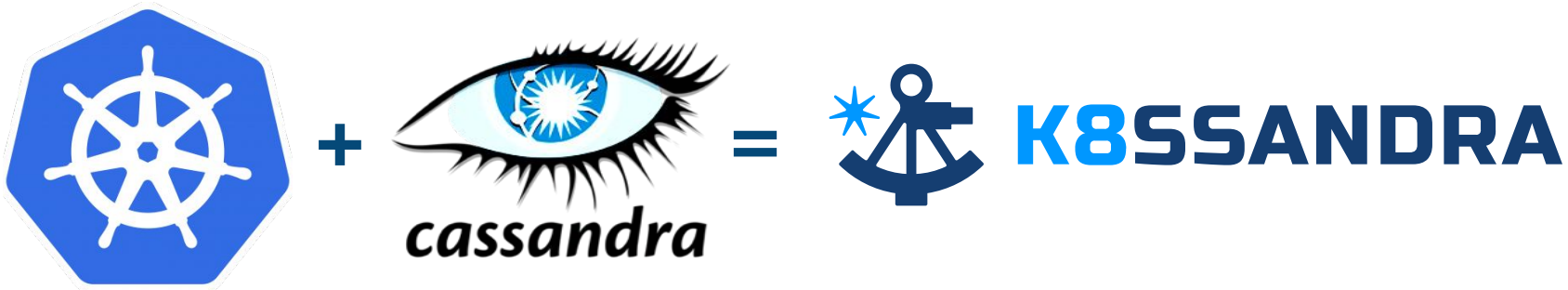
Kubernetes
Les fondamentaux

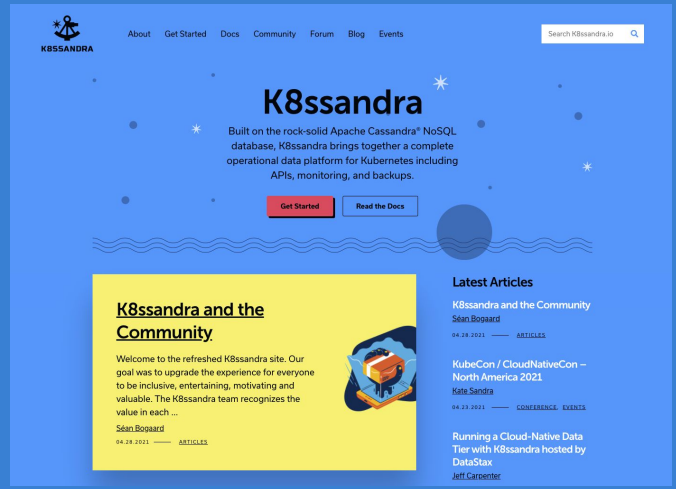
05 

Kubernetes
Stateful & Cassandra

05 

K8ssandra
Tour d'horizon

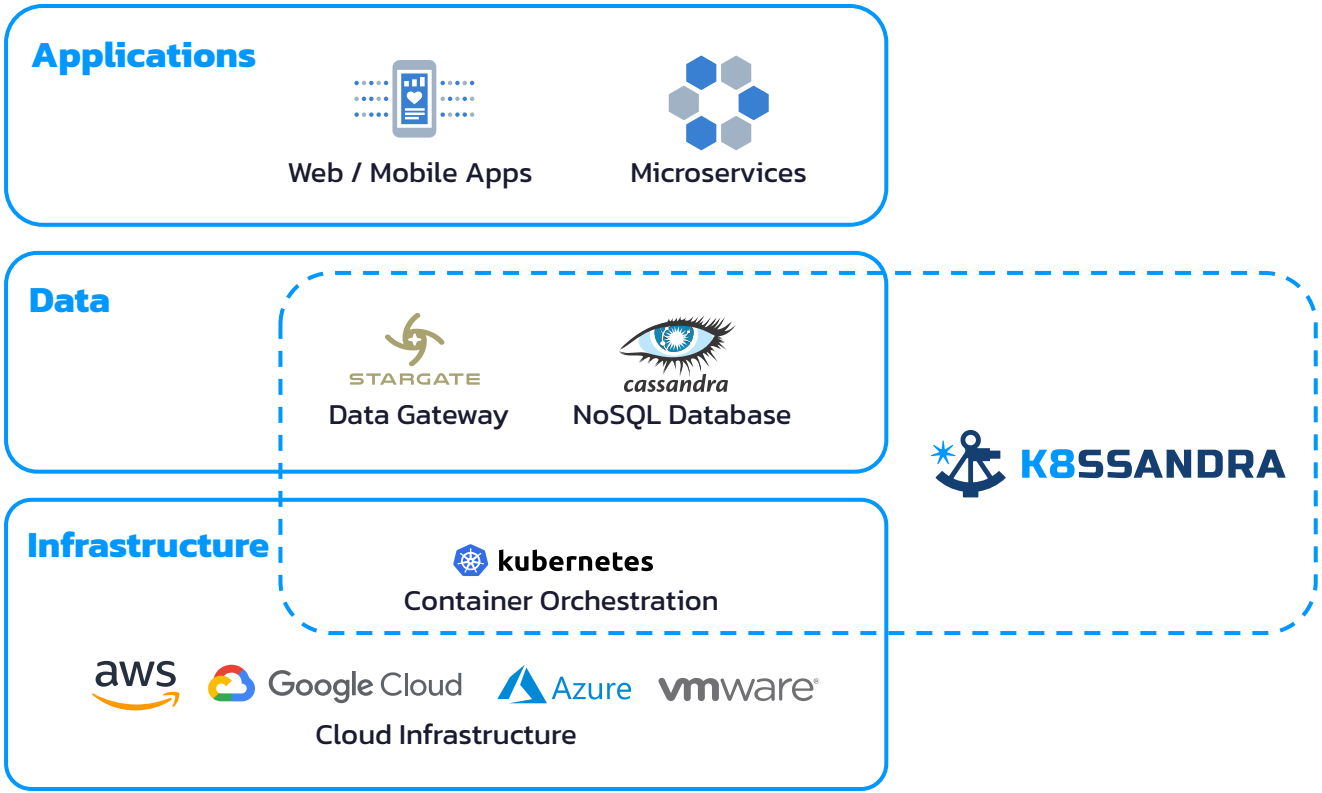

+
cassandra
=
K8SSANDRA



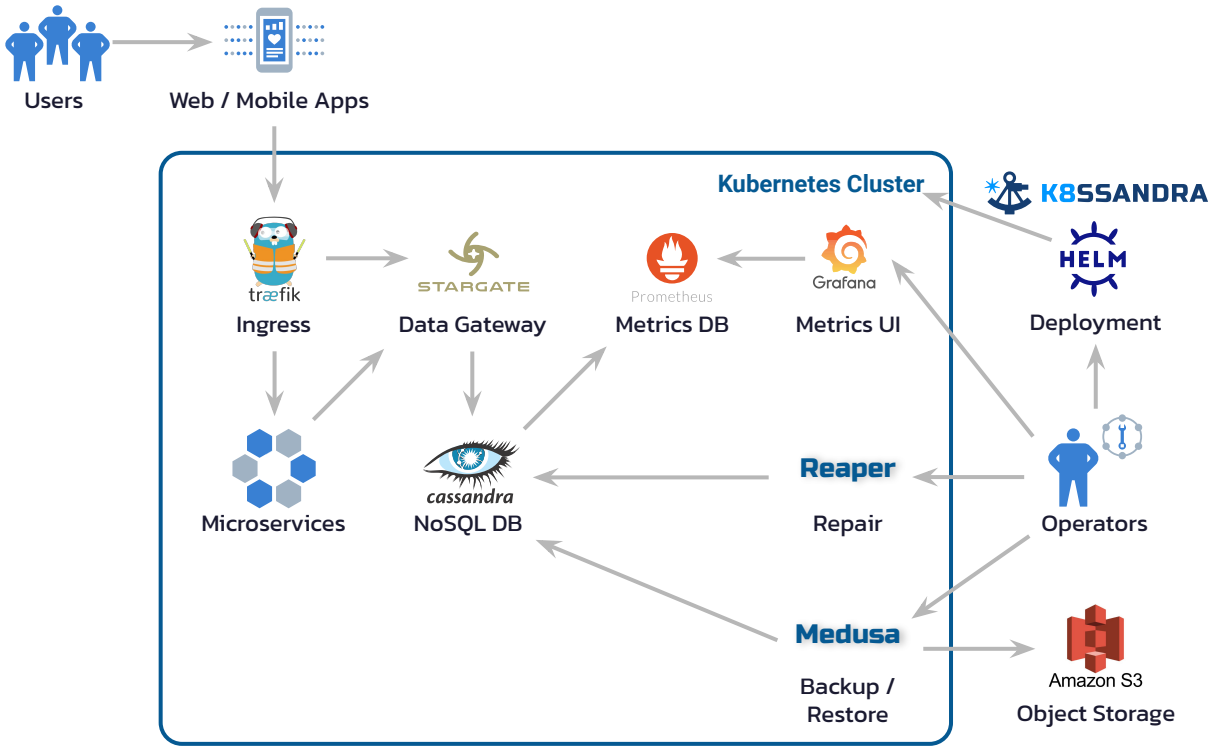
K8SSANDRA

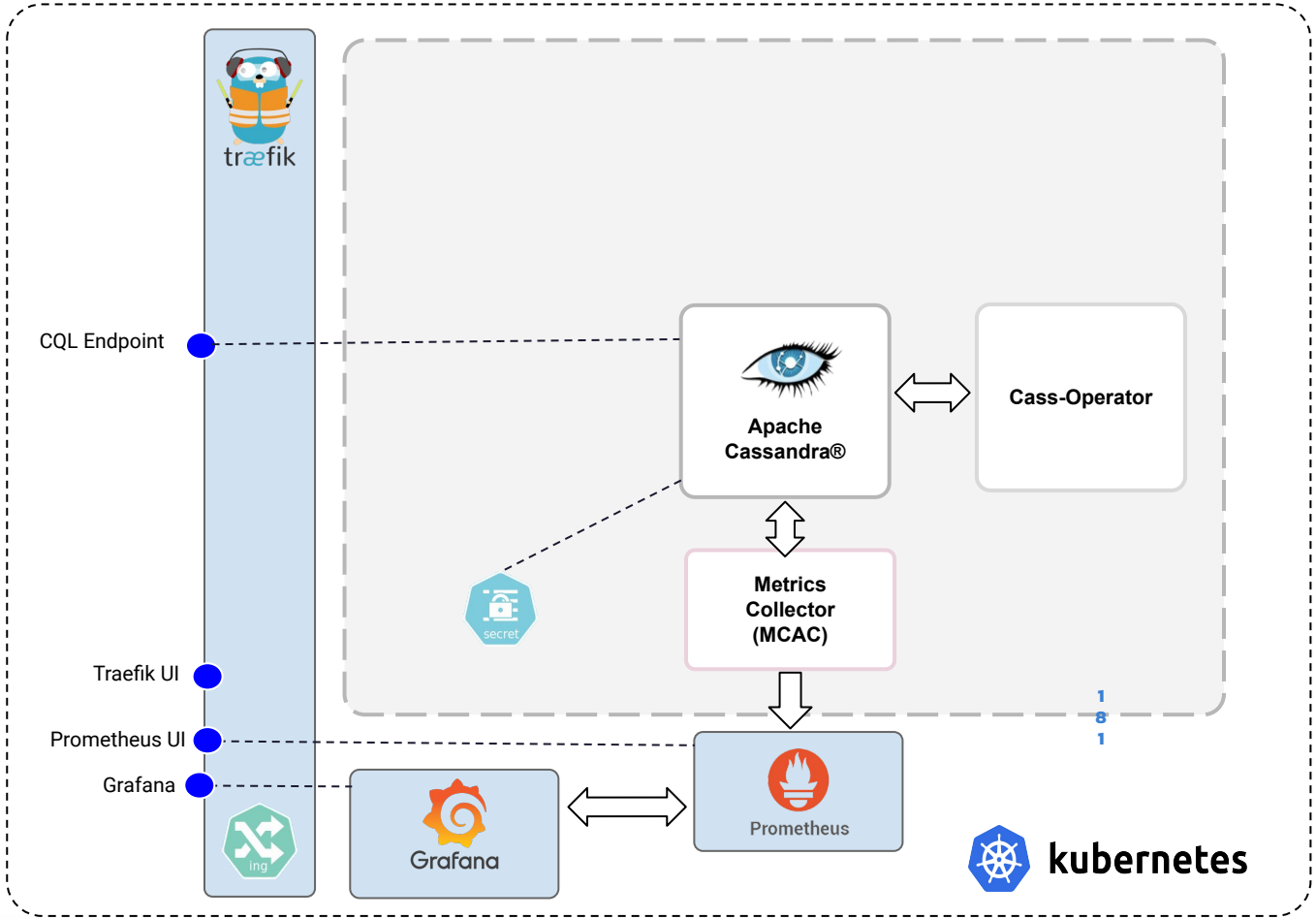
Cloud native, scalable data tier with administration tools and easy data access

Highly Scalable Cloud Application Architecture



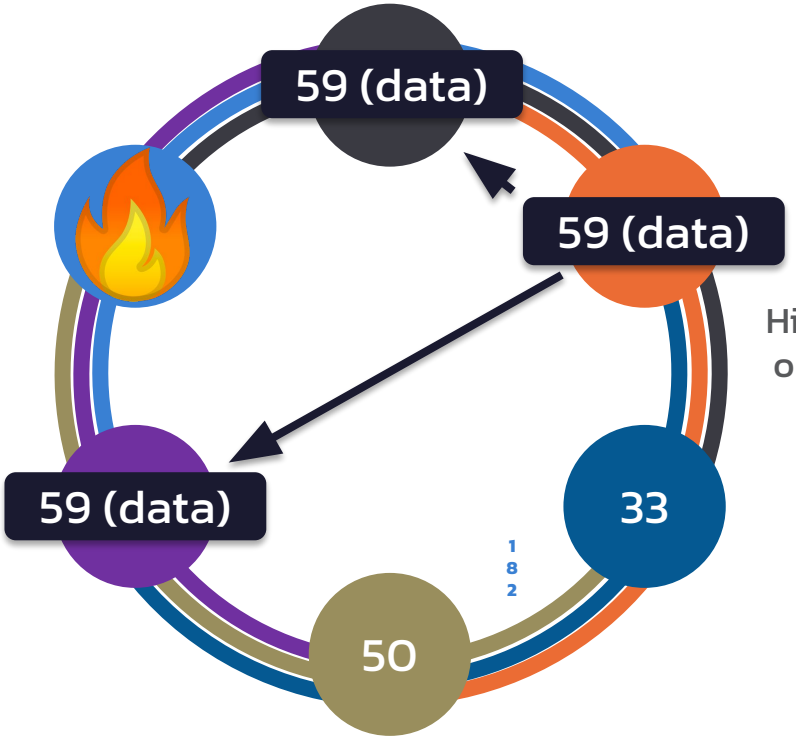
K8ssandra Components in Context




træfik
CQL Endpoint
Traefik UI
Prometheus UI
Grafana
ing
Apache Cassandra®
Cass-Operator
secret
Metrics Collector (MCAC)
Grafana
Prometheus
kubernetes

Repairs in Cassandra

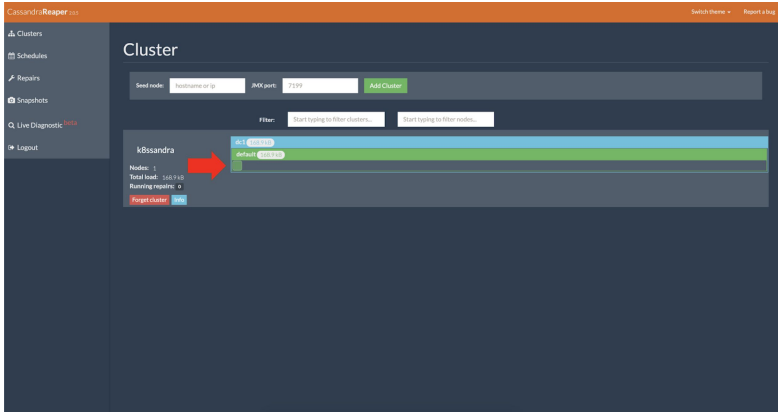
- Repairs are the "last resort" to keep data across a Cassandra cluster in sync
- If repairs don't happen the health of the data will slowly degrade over time due to Entropy

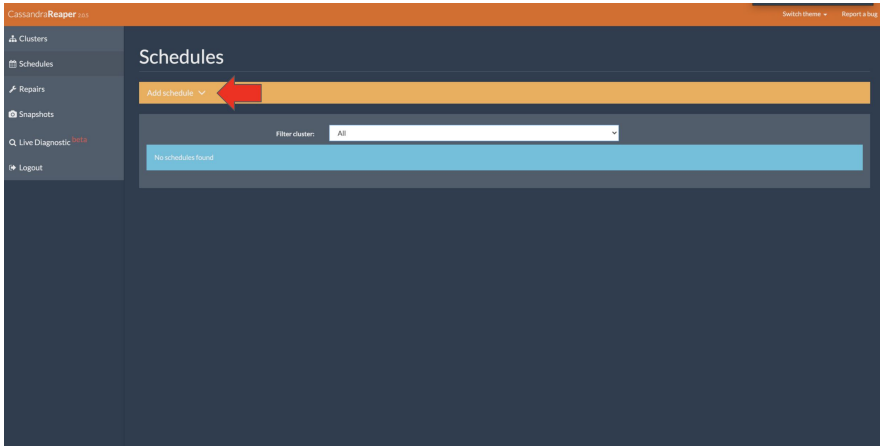


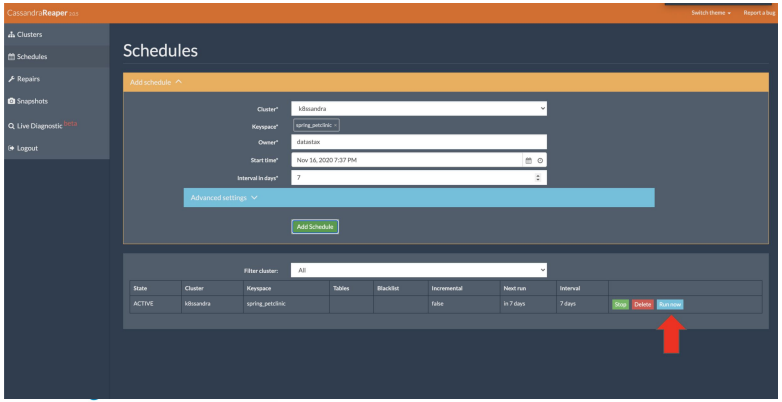
Hint times out... now what?

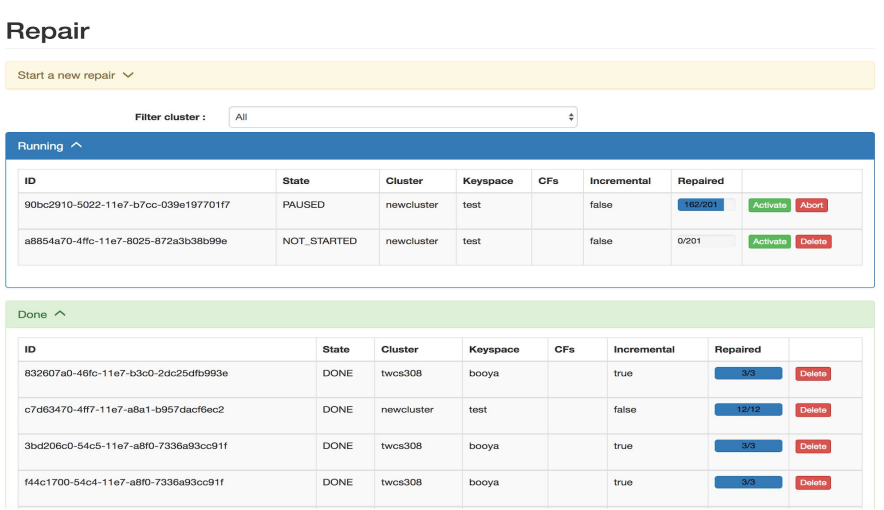
"Without repairs, your database is *hopefully consistent*"
– Vinay Kumar Chella, Principal Cloud Database Architect, Netflix

Cassandra Reaper

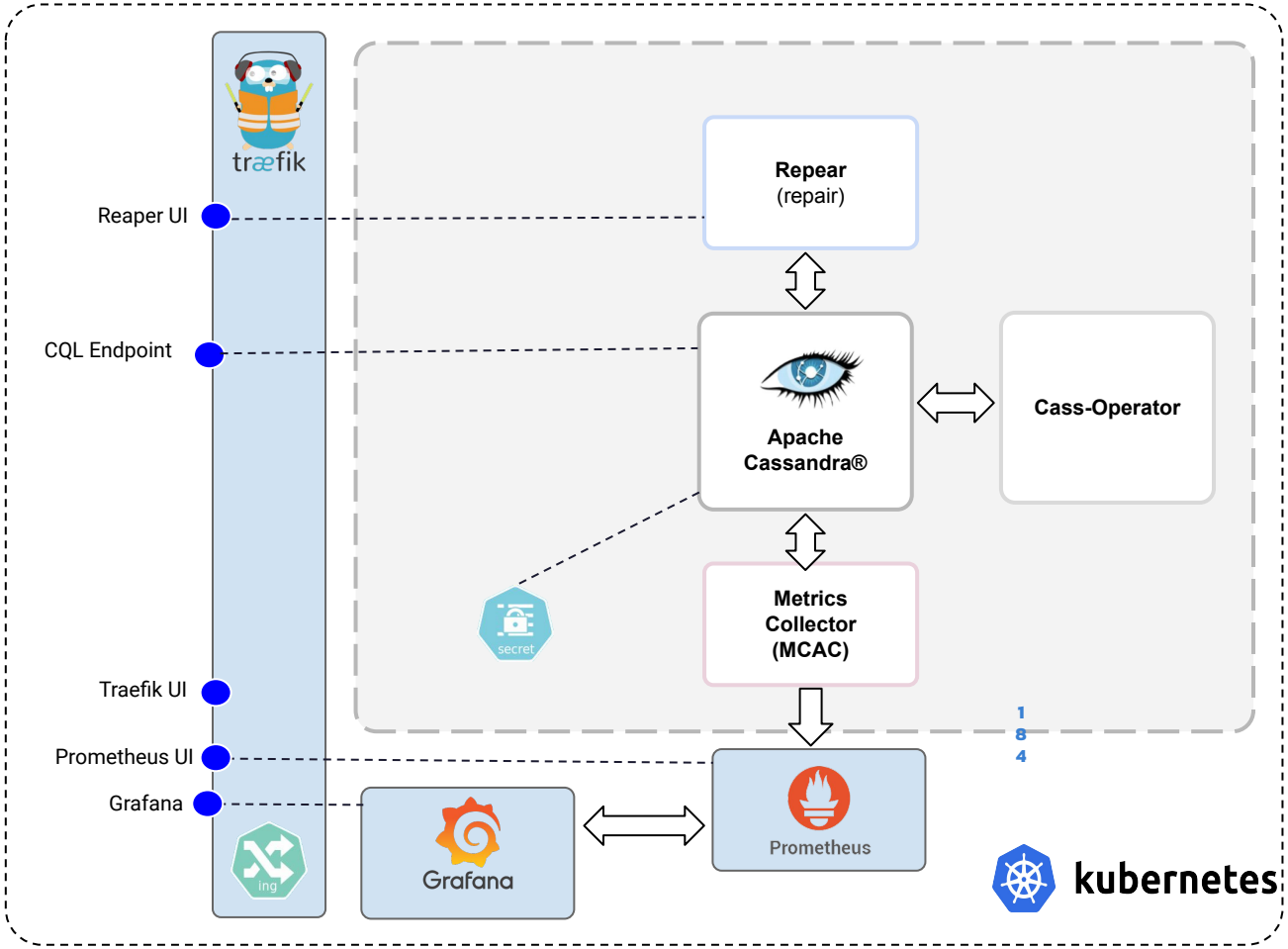








Repair


træfik
Reaper UI
CQL Endpoint
Traefik UI
Prometheus UI
Grafana
ing
Repear
(repair)
Apache
Cassandra®
Cass-Operator
Metrics
Collector
(MCAC)
secret
Grafana
Prometheus
kubernetes

Medusa

Orchestrated backup and restore for Apache Cassandra and DSE.

Designed by 

Implemented by 

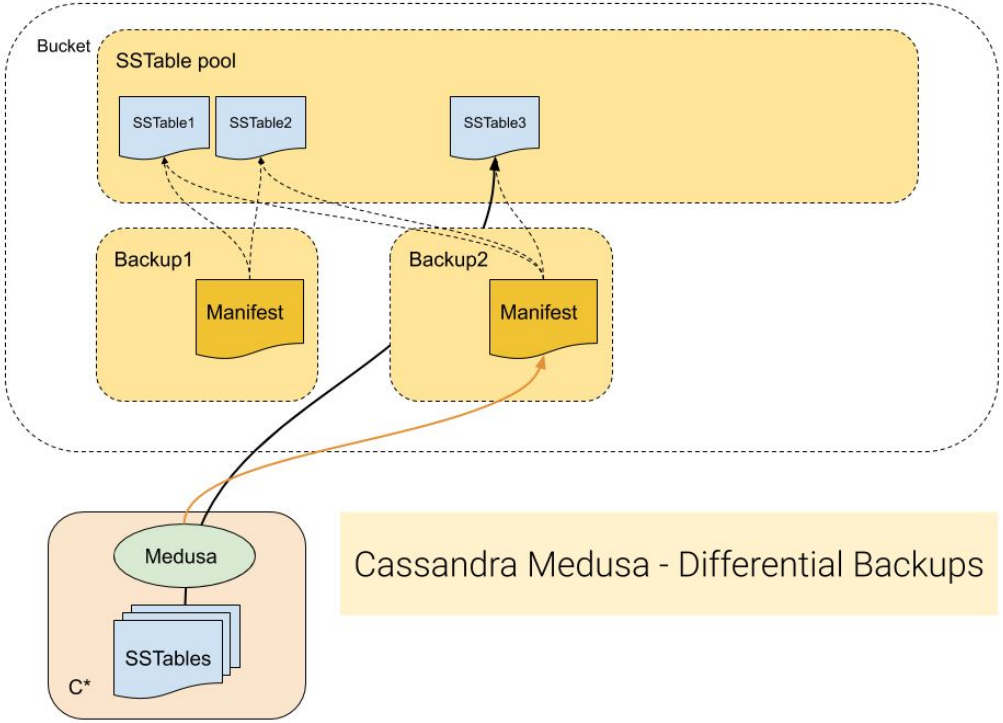
Maintained and supported by 

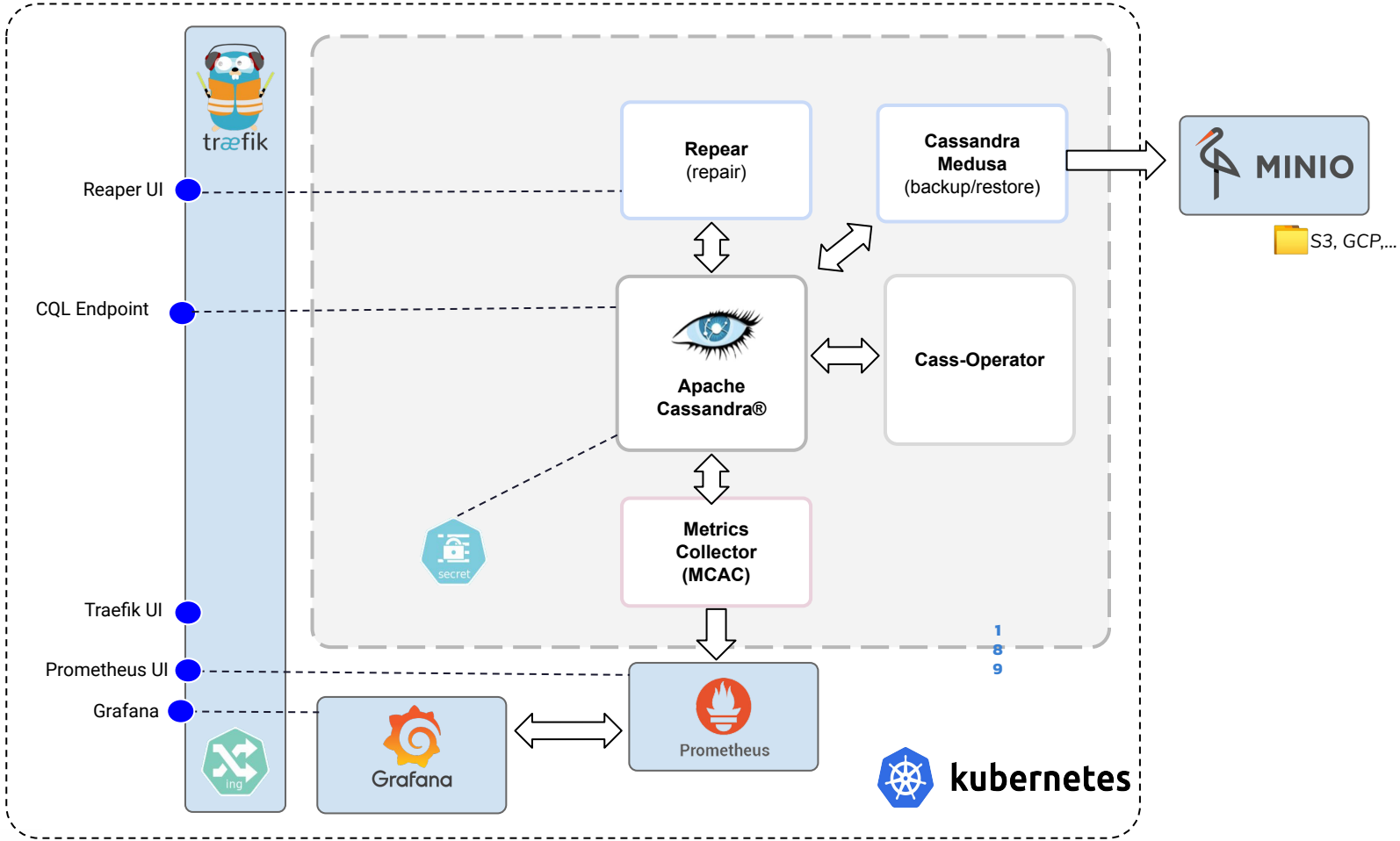
Storage backends

- NFS shares
- AWS S3
- Google Cloud Storage
- IBM Cloud Object Storage
- Azure Blob Storage

Differential backups (default)

SSTables are uploaded only **once** in their lifetime to save **space** and **bandwidth**

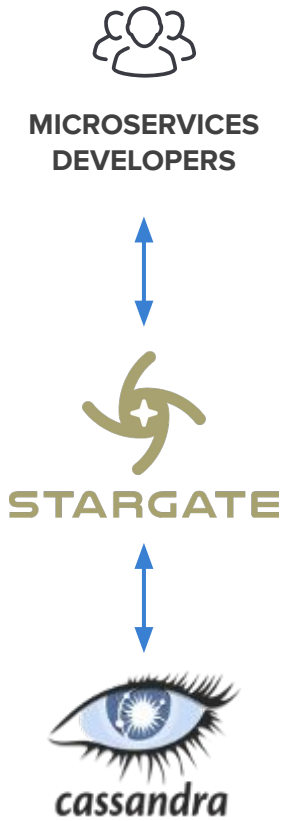



træfik
Reaper UI
CQL Endpoint
Traefik UI
Prometheus UI
Grafana
ing
Repear
(repair)
Cassandra
Medusa
(backup/restore)
MINIO
S3, GCP,...
Apache
Cassandra®
Cass-Operator
secret
Metrics
Collector
(MCAC)
1
8
9
Grafana
Prometheus
kubernetes

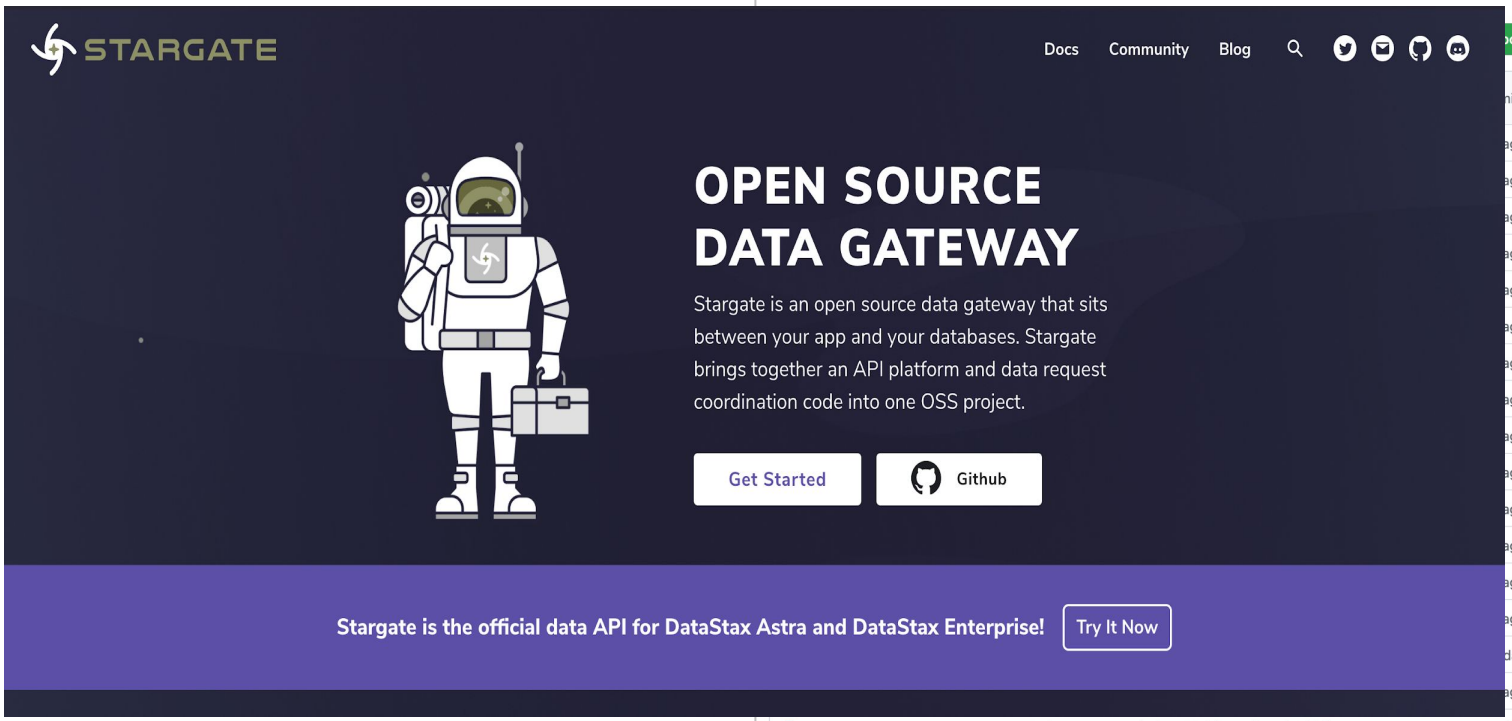
Stargate Overview

An open source API framework for data

Stargate makes it easy to use a database for any application workload by adding plugin support for new APIs, data types, and access methods

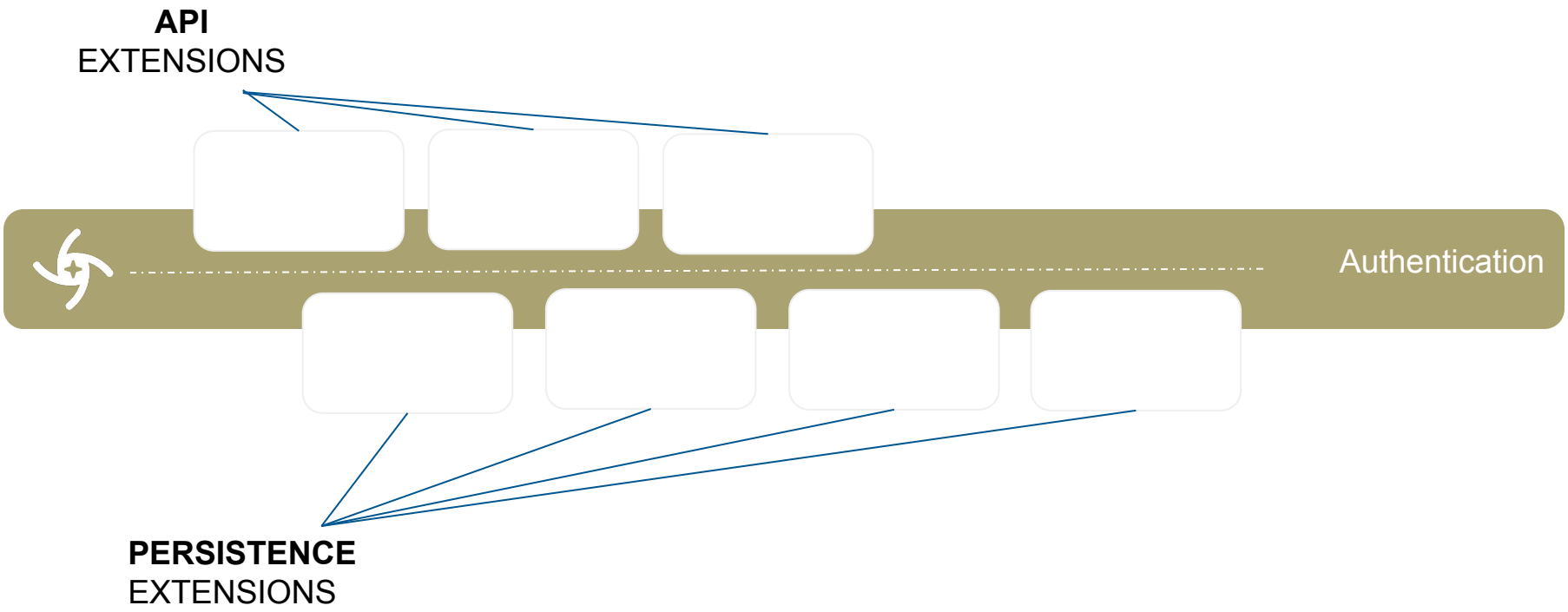


Stargate.io

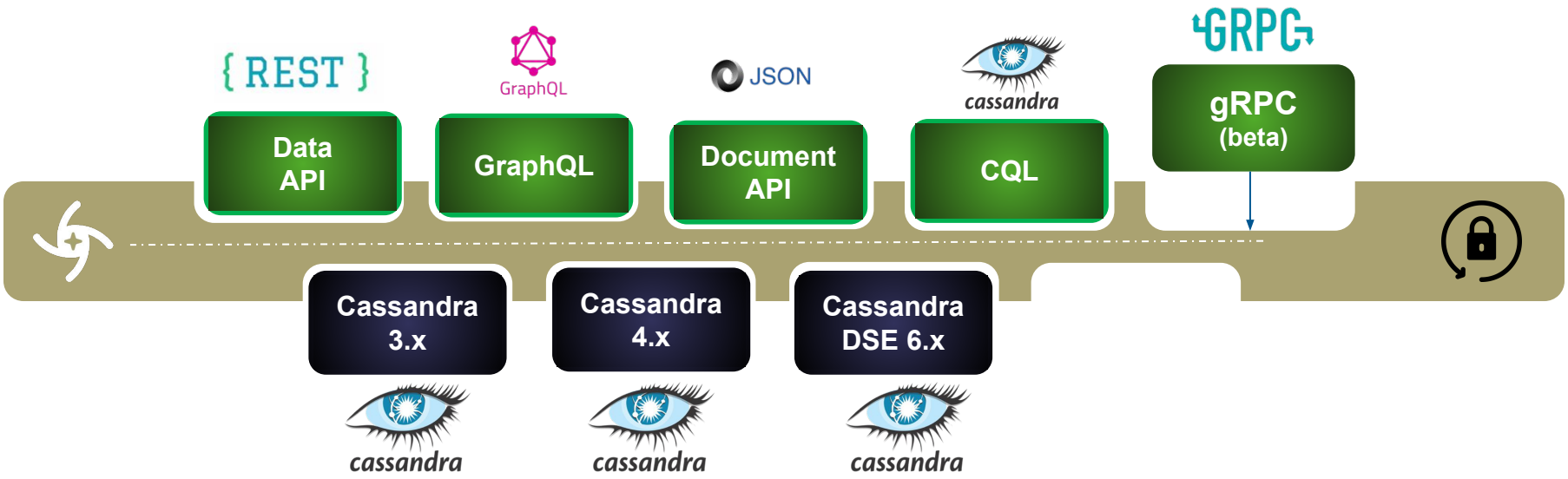

STARGATE
Docs
Community
Blog
OPEN SOURCE
DATA GATEWAY
Stargate is an open source data gateway that sits between your app and your databases. Stargate brings together an API platform and data request coordination code into one OSS project.
Get Started
Github
Stargate is the official data API for DataStax Astra and DataStax Enterprise!
Try It Now

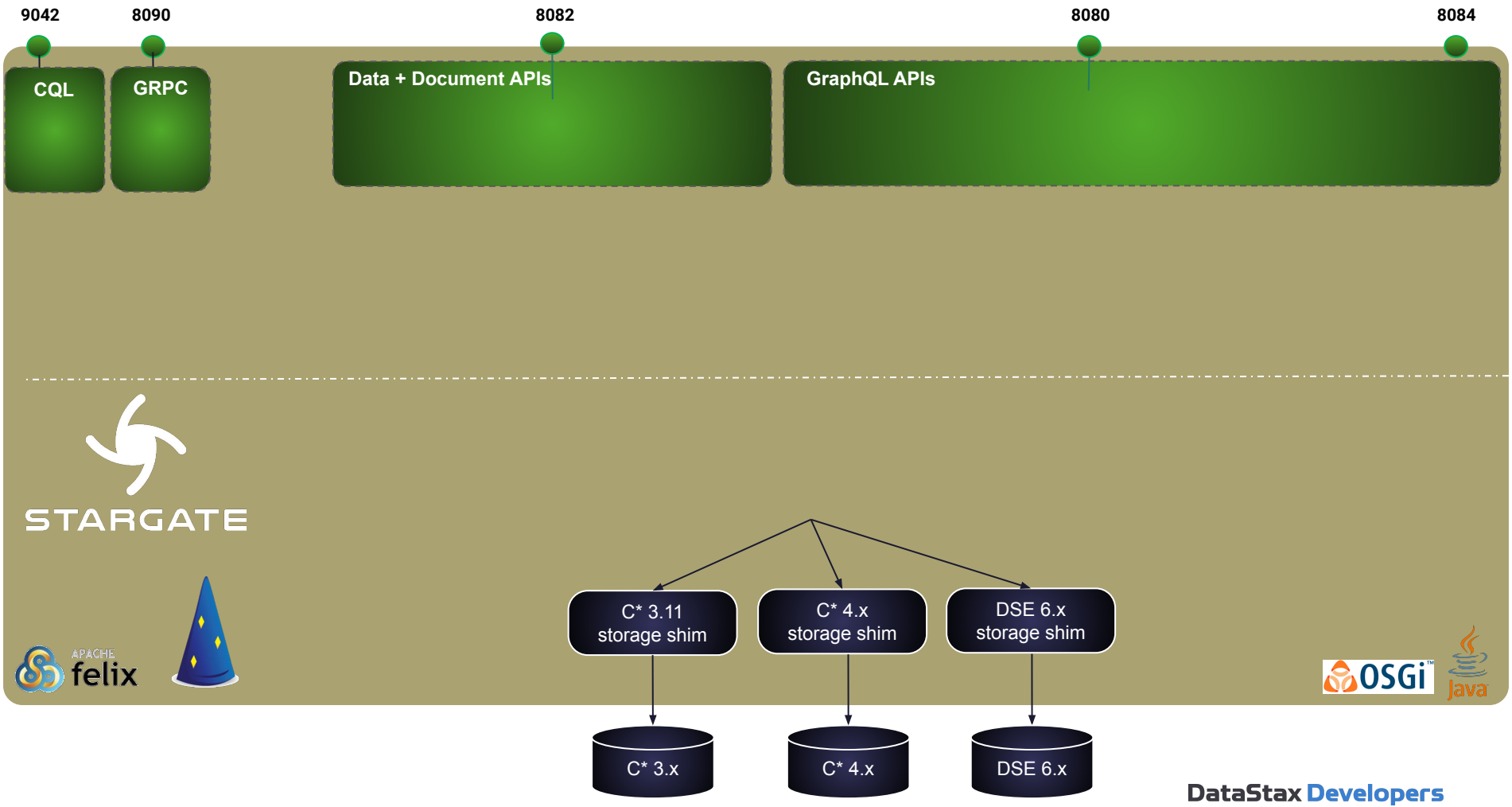
stargate / stargate
Public
Watch 22
Unstar 506
Fork 65
Code
Issues 246
Pull requests 13
Discussions
Actions
Projects
Wiki
About
An open source data gateway
stargate.io
java
graphql
rest
cassandra
cql
Readme
Apache-2.0 License
Releases 62
Release v1.0.32 Latest
25 days ago
+ 61 releases
Packages 13
io.stargate.stargate
io.stargate.db.persistence-api
io.stargate.db.persistence-common
+ 10 packages
Contributors 30
grpc
Reference example for gRPC java client (#1236)
yesterday

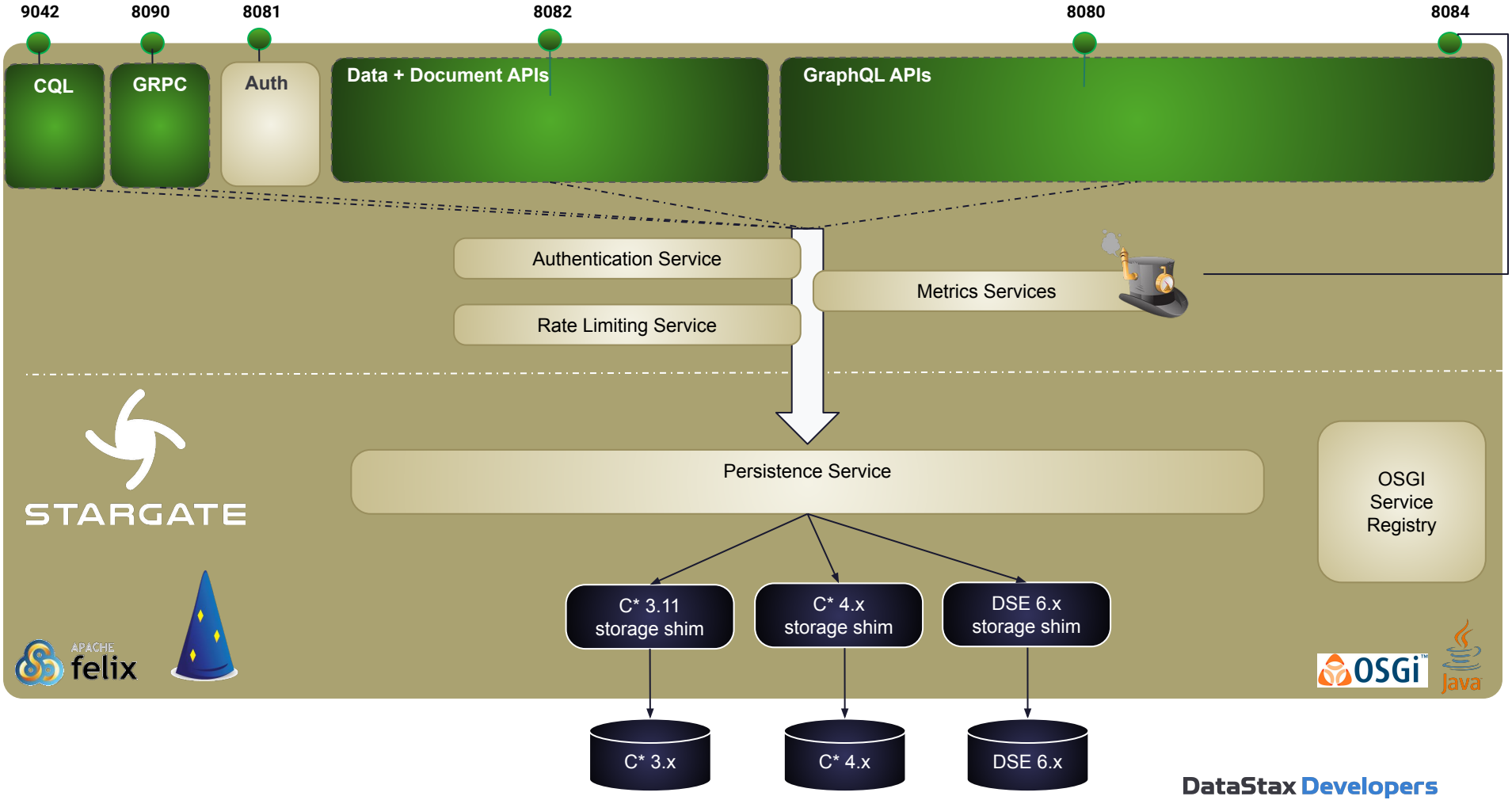
API and Persistence Extensions

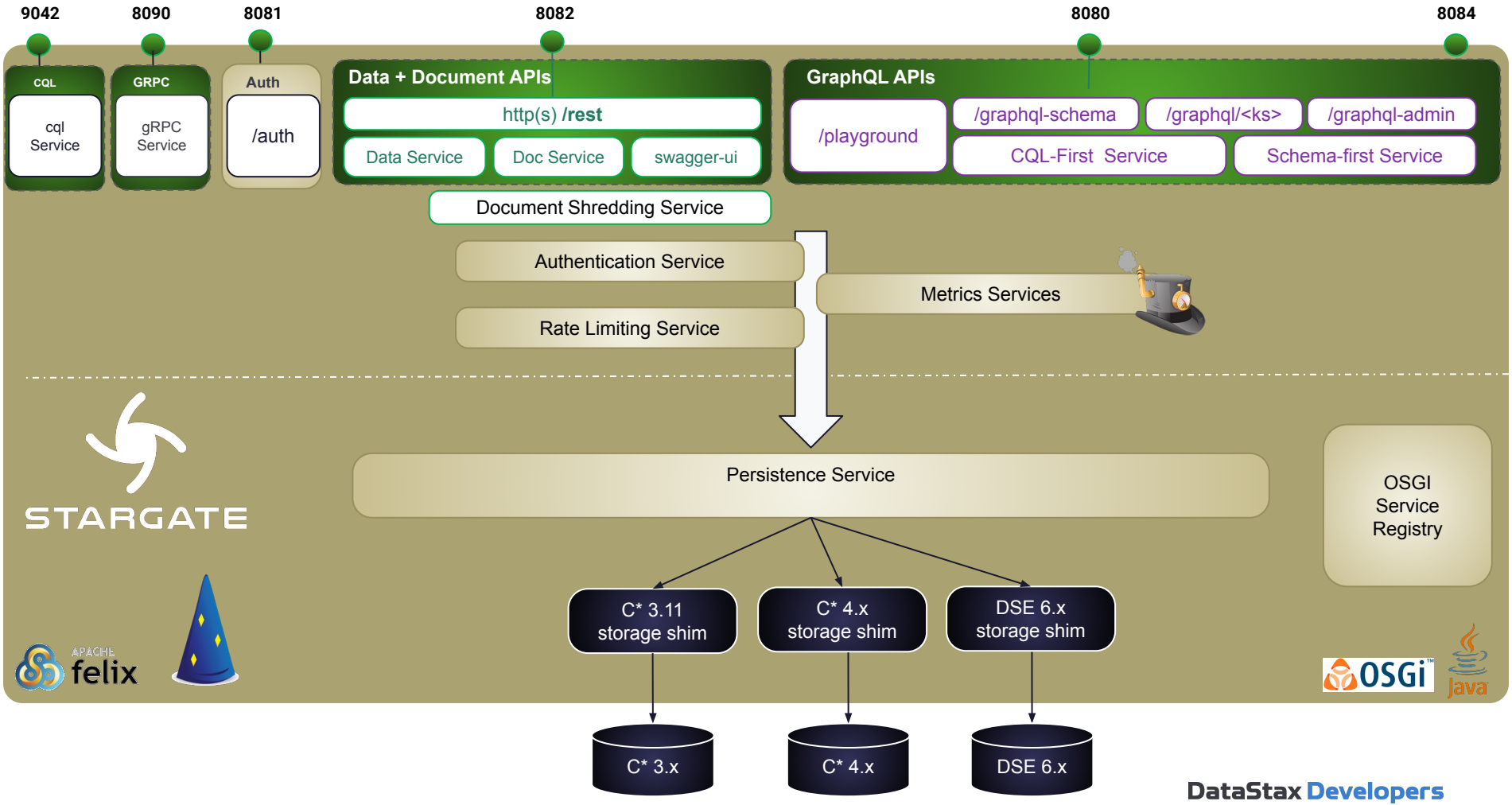


API Extensions and Persistence Extensions

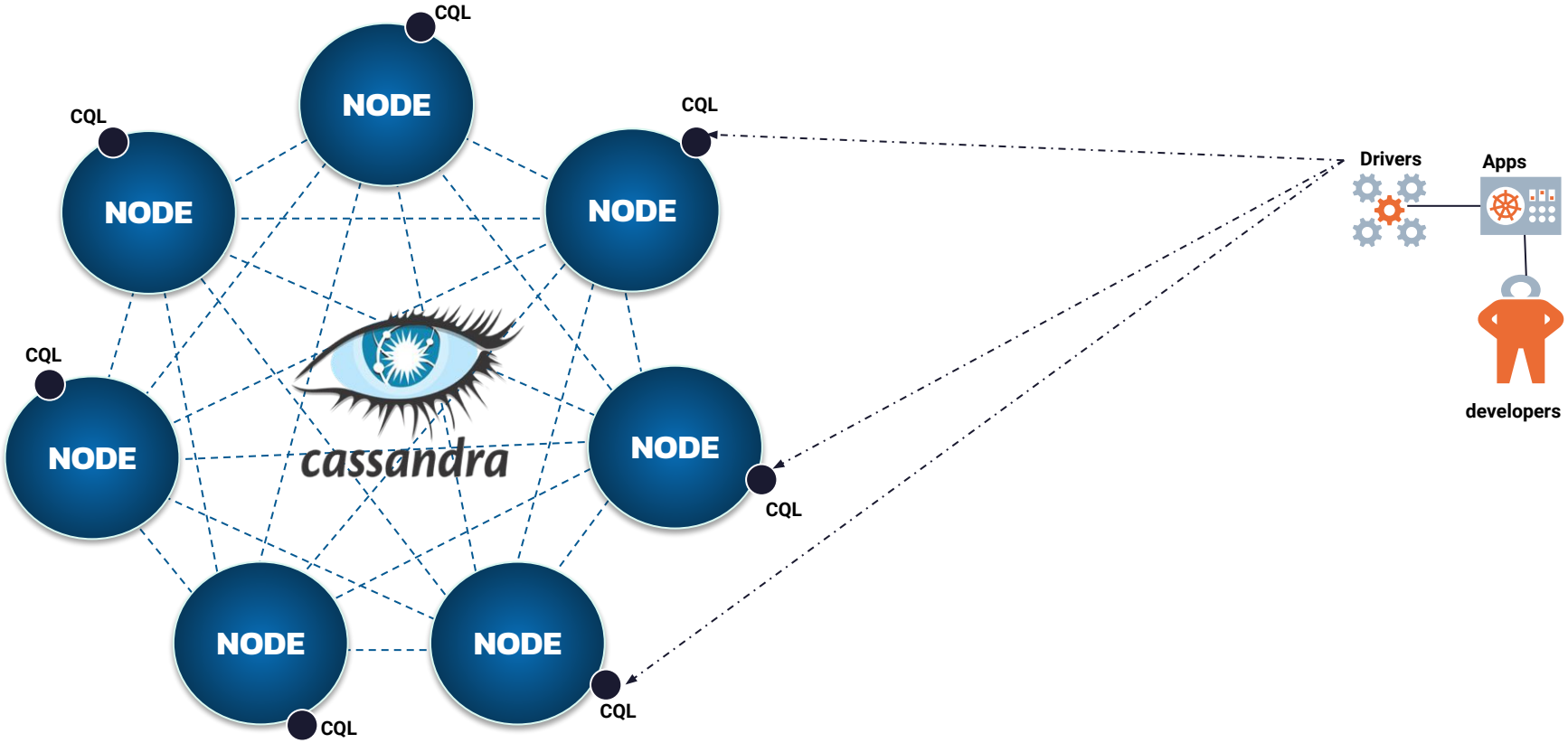



9042
8090
8082
8080
8084
CQL
GRPC
Data + Document APIs
GraphQL APIs
STARGATE
APACHE
felix
C* 3.11
storage shim
C* 4.x
storage shim
DSE 6.x
storage shim
C* 3.x
C* 4.x
DSE 6.x
OSGi
Java
DataStax Developers

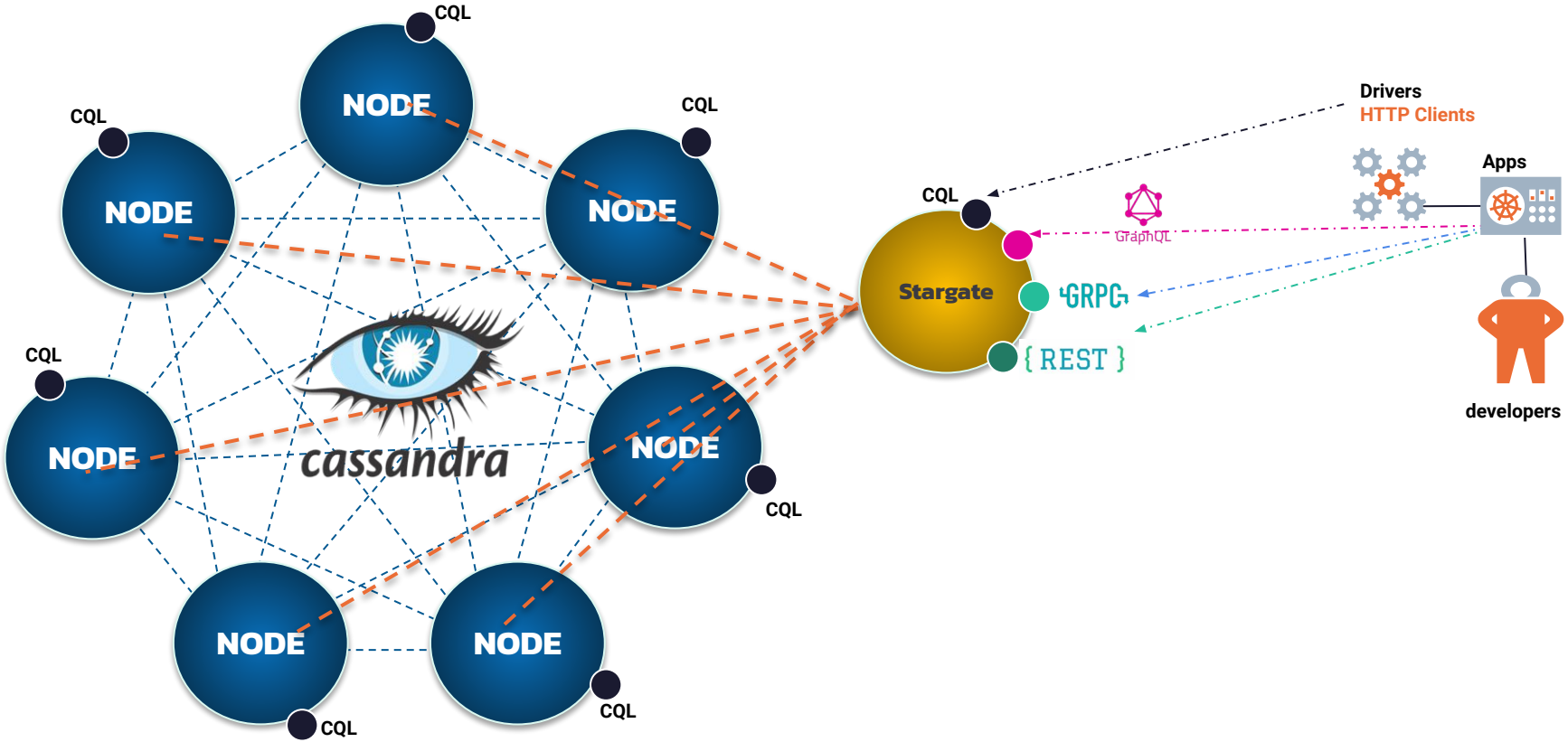

9042
8090
8081
8082
8080
8084
CQL
GRPC
Auth
Data + Document APIs
GraphQL APIs
Authentication Service
Rate Limiting Service
Metrics Services
STARGATE
Persistence Service
OSGI
Service
Registry
C* 3.11
storage shim
C* 4.x
storage shim
DSE 6.x
storage shim
APACHE
felix
OSGi
Java
C* 3.x
C* 4.x
DSE 6.x
DataStax Developers


9042
8090
8081
8082
8080
8084
CQL
cql Service
GRPC
gRPC Service
Auth
/auth
Data + Document APIs
http(s) /rest
Data Service
Doc Service
swagger-ui
GraphQL APIs
/playground
/graphql-schema
/graphql/<ks>
/graphql-admin
CQL-First Service
Schema-first Service
Document Shredding Service
Authentication Service
Rate Limiting Service
Metrics Services
STARGATE
Persistence Service
OSGI Service Registry
C* 3.11 storage shim
C* 4.x storage shim
DSE 6.x storage shim
C* 3.x
C* 4.x
DSE 6.x
APACHE felix
OSGi
Java
DataStax Developers

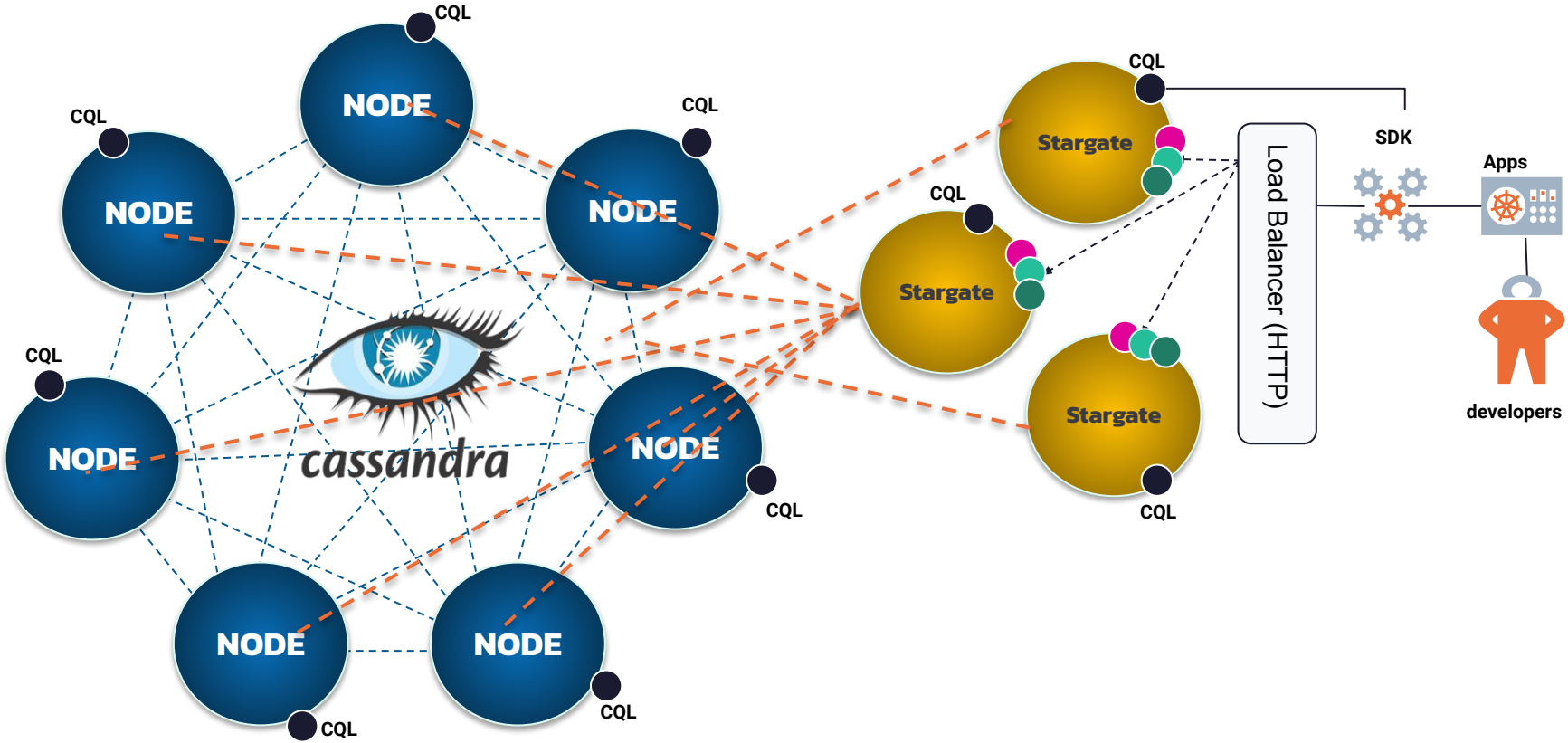
Connecting to your cluster (Before)



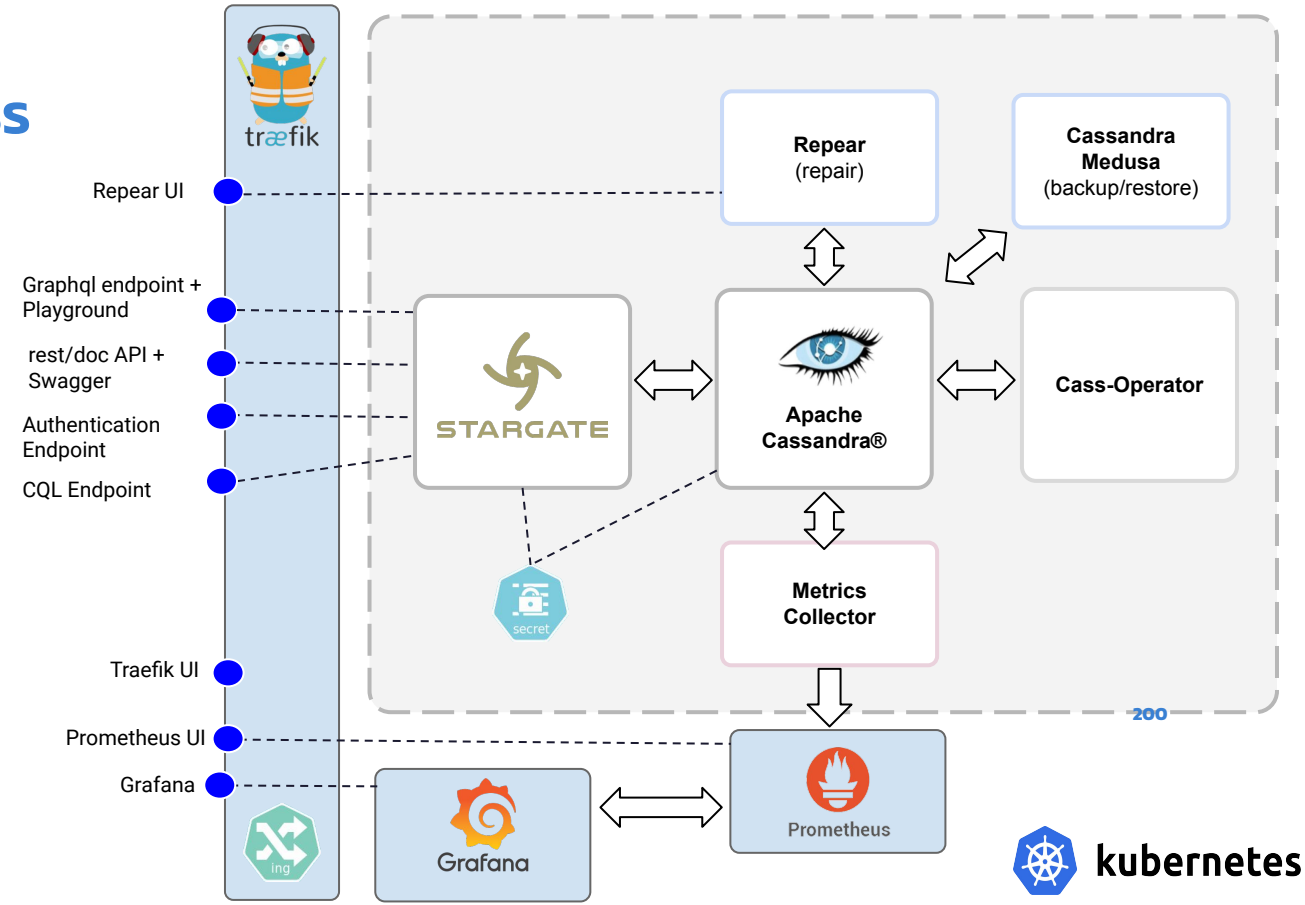
Connecting to your cluster (with Stargate)



Connecting to your cluster (with Stargates)



Data Access


træfik
Repear UI
Graphql endpoint + Playground
rest/doc API + Swagger
Authentication Endpoint
CQL Endpoint
Traefik UI
Prometheus UI
Grafana
ing
Repear
(repair)
Cassandra
Medusa
(backup/restore)
STARGATE
Apache
Cassandra®
Cass-Operator
secret
Metrics
Collector
200
Grafana
Prometheus
kubernetes

Join the K8ssandra Community!

Sign up for newsletter ->

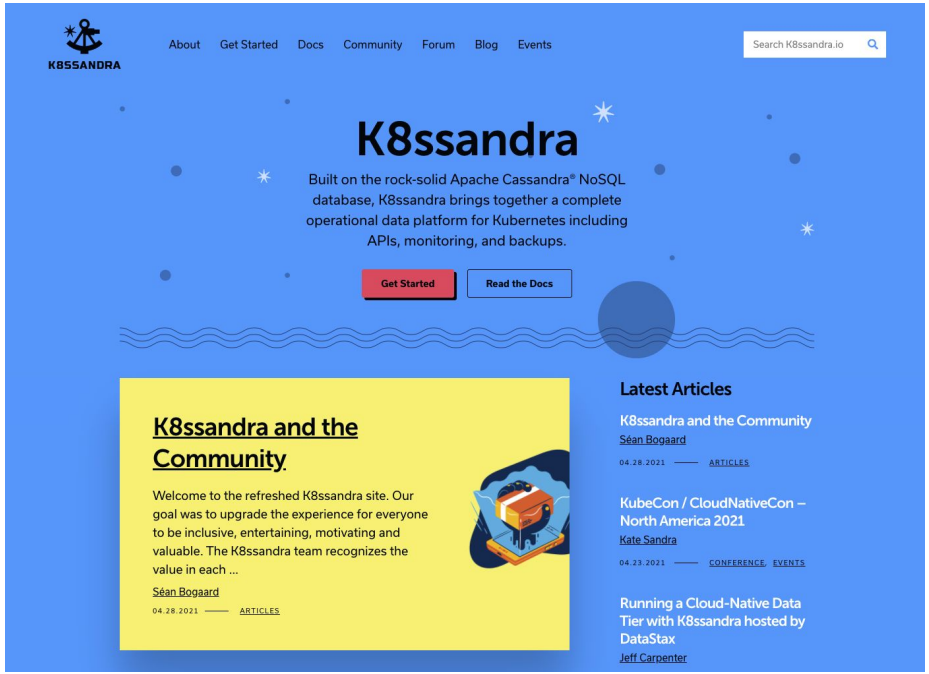
forum.k8ssandra.io

Discord live chat

@k8ssandra

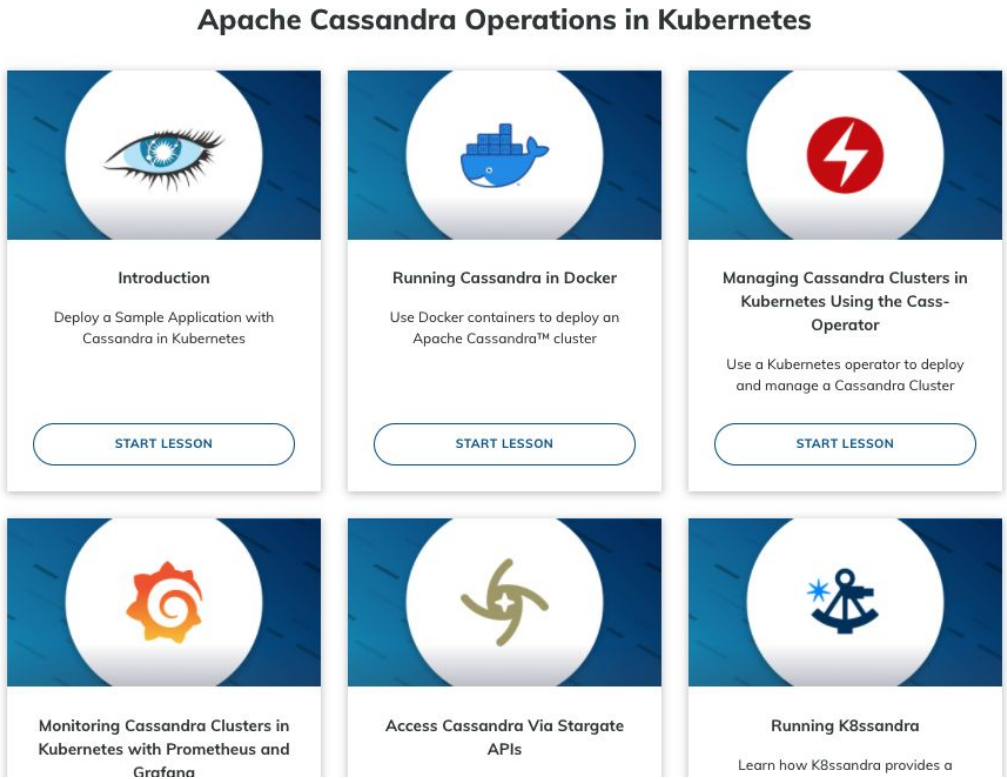
github.com/k8ssandra

k8ssandra.io

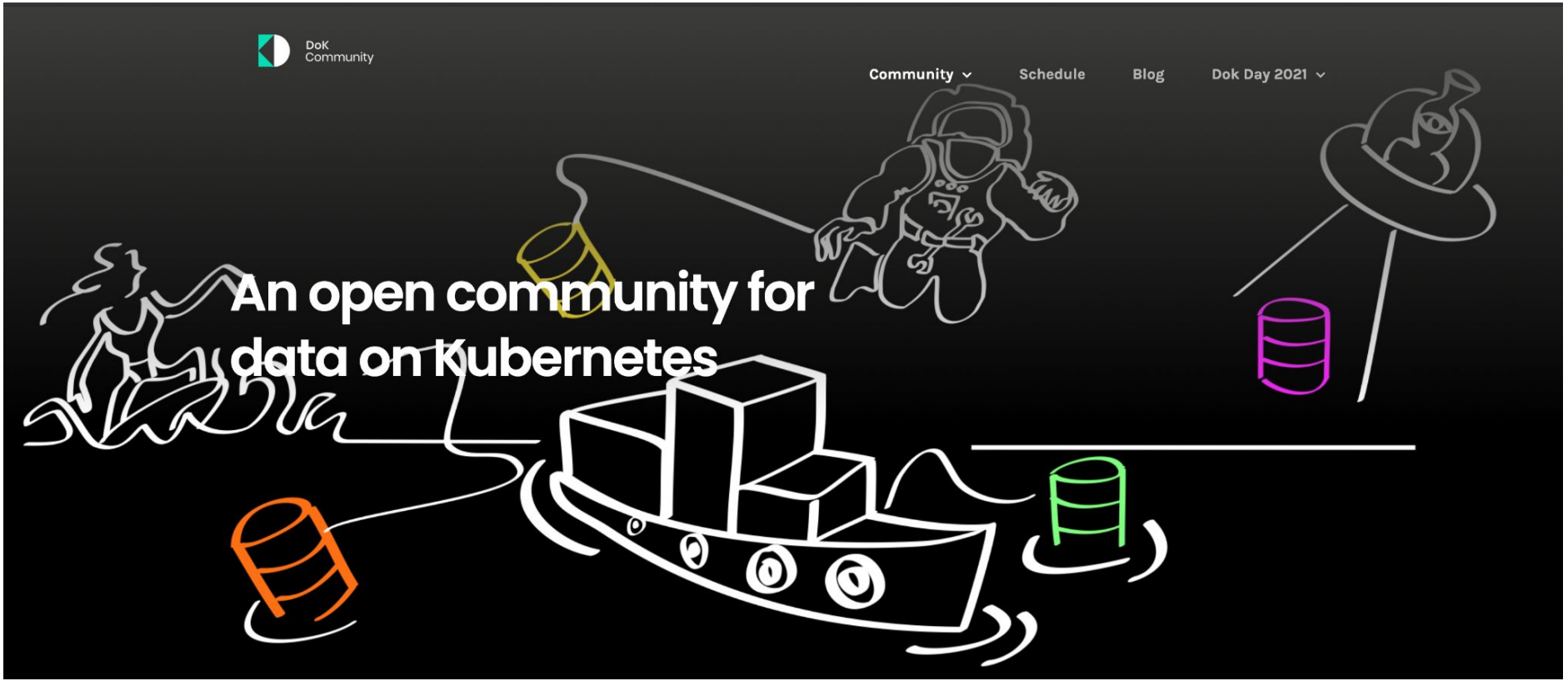


Hands-ON

https://github.com/k8ssandra/workshop



Data on Kubernetes Community – DoK Community



Can you run databases on K8s?

Kubernetes is everywhere. Transactional apps, video streaming services and machine learning workloads are finding a home on this ever-growing platform. But what about databases? If you had asked me this question five years ago, the answer would have been a resounding "**No!**" — based on my experience in development and operations. In the following years, as more resources emerged for stateful applications, my answer would have changed to "*Maybe,*" but always with a qualifier: "It's fine for development or test environments..." or "If the rest of your tooling is Kubernetes-based, and you have extensive experience..."

But how about today? Should you run a database on Kubernetes? With complex operations and the requirements of persistent, consistent data, let's retrace the stages in the journey to my current answer: "In a cloud native environment? **Yes!**"

Christopher Bradford

Christopher has a passion for enabling efficiency through automation. From promoting effortless scaling via Cassandra to DevOps pipelines with infrastructure automation and containers, he is here to get work done and enable operators to rest easy.

https://thenewstack.io/a-case-for-databases-on-kubernetes-from-a-former-skeptic/

Pushing Helm to its limit

DATA / KUBERNETES / OPEN SOURCE / SPONSORED / CONTRIBUTED

We Pushed Helm to the Limit, then Built a Kubernetes Operator

8 Sep 2021 6:04am, by Jeff Carpenter and John Sanda

Jeff Carpenter

Jeff works in developer relations at DataStax. He has worked on large-scale systems in the defense and hospitality industries and is co-author of 'Cassandra: The Definitive Guide.'

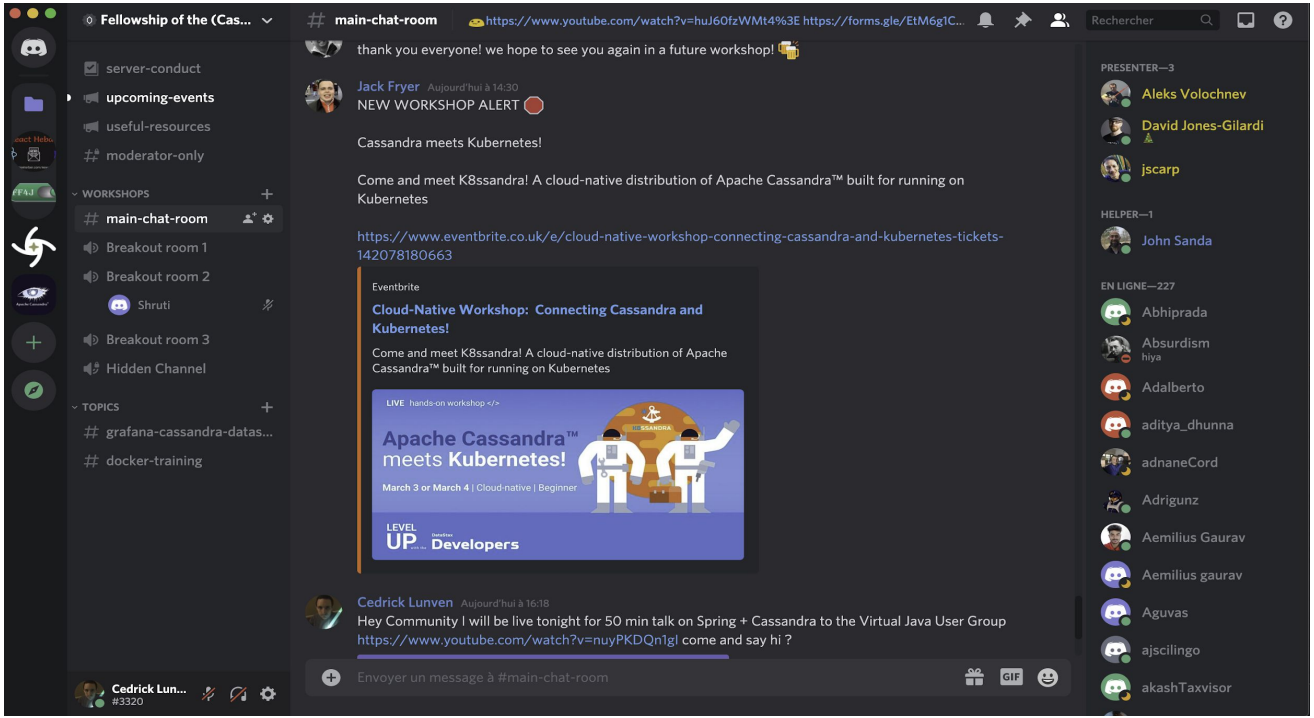
John Sanda

John is a DataStax engineer working on the K8ssandra project. He is passionate about Cassandra and Kubernetes and loves being involved in open source. Prior to joining DataStax, John worked as an Apache Cassandra consultant.

https://thenewstack.io/we-pushed-helm-to-the-limit-then-built-a-kubernetes-operator/

https://bit.ly/cassandra-workshop

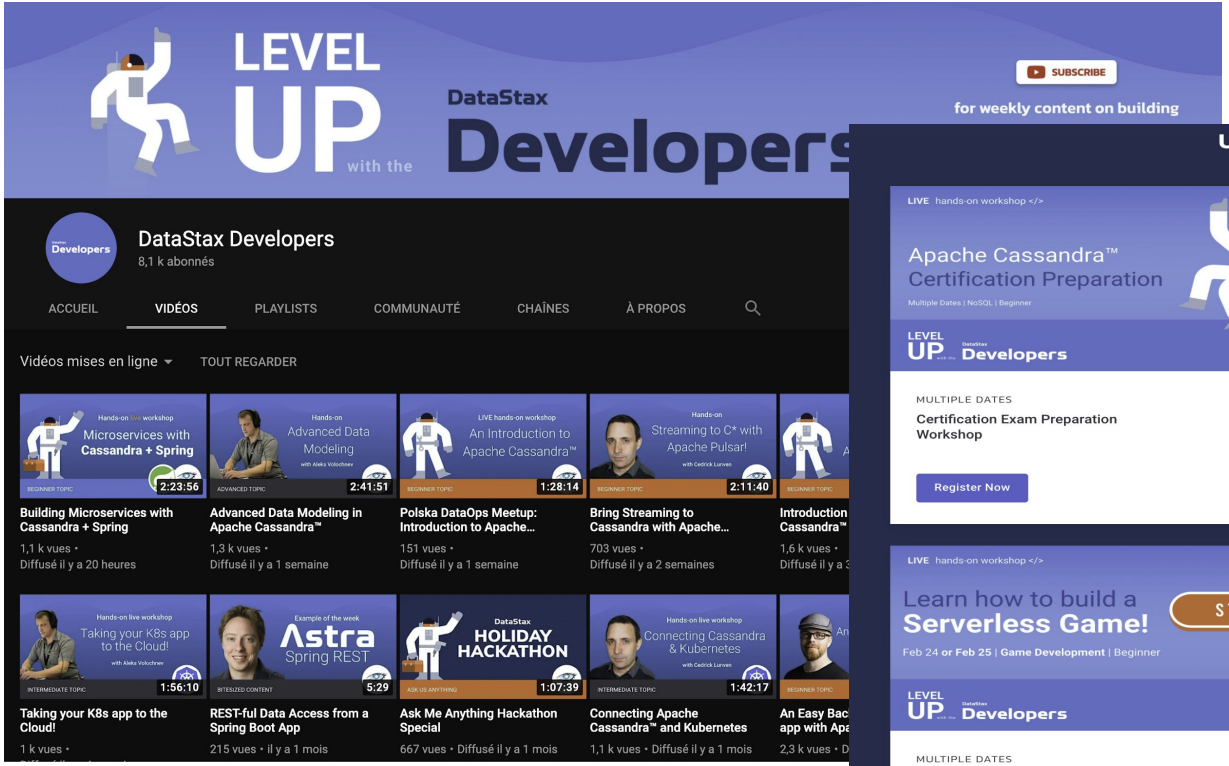
Join our 10k Discord Community
The Fellowship of the RINGS

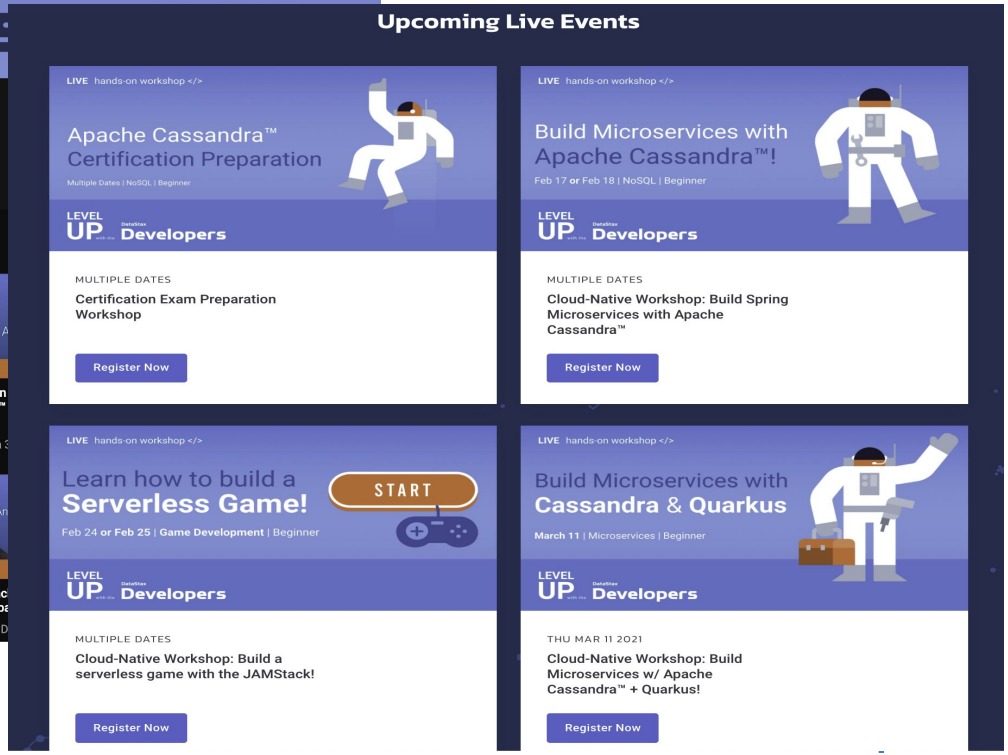


DataStax Developers

Workshops Hebdomadaires

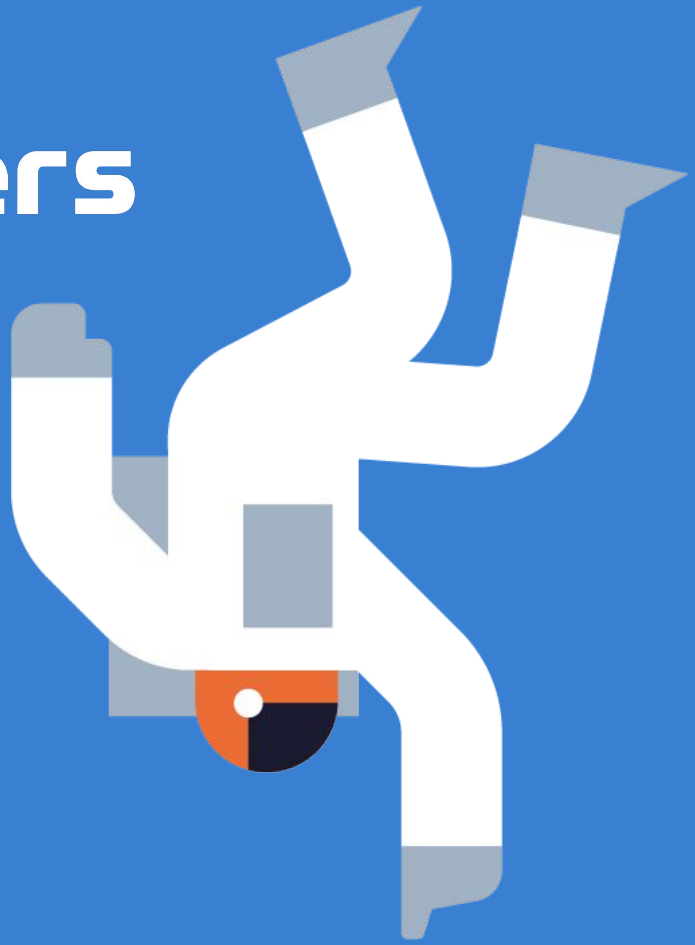
https://www.datastax.com/workshops





DataStax Developers

Thank you!



GitHub @clun

@clunven