

DEVELOPERWEEK™

AUSTIN

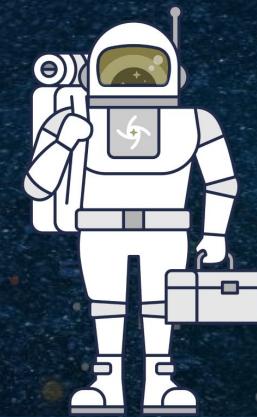
• VIRTUAL •

Expose **Rest, Graphql, Grpc**
Apis on Top for Your
Databases

DataStax Developers



STARGATE



Cédrick Lunven

Director Developer Relations



SDK



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



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



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

Agenda

01

Data Api
What, Why

02

Rest, GraphQL, gRPC
SWOT

03

Data Gateway
Why and How

04

Stargate.io
Architecture

05

Demo
Api and Tooling

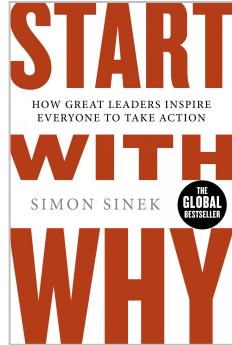
05

What's NEXT ?
Resources

Data API : Start with Why

- Developers

- Do you like learning query languages (CQL, N1QL, GQL, cypher, gremlin....)
 - No. Save my JSON, give it to me back when I need it
 - ORM / Spring Data are so popular nowadays
- Do you care about how your data is stored ?
 - Physical data model part of the interface...yikes
 - Create structures based on queries
- Do you like installing and running databases locally
 - especially distributed databases
 - especially with datasets and integration tests



Data API : Start with Why

- Operators and Databases Administrators...
 - Do you allow developers to execute direct queries against your DB ?
 - Do you like opening port ranges like 0-65536 to allow communications with applications, especially in the cloud.
 - Do you like creating dedicated projects and hiring people just to create APIs to expose an existing treatment (digital transformation FTW)
 - Security, Monitoring, Throttling

Agenda

01

Api and Databases
What, Why and How

02

Rest, GraphQL, gRPC
SWOT

03

Data Gateway
Why and How

04

Stargate.io
Architecture

05

Demo
Api and Tooling

05

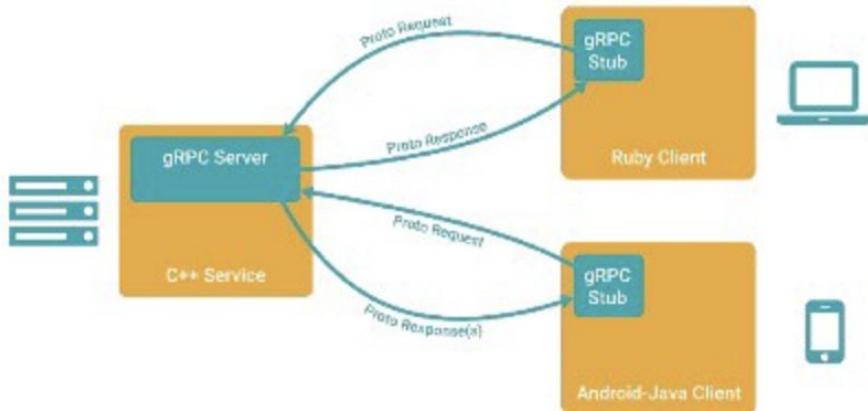
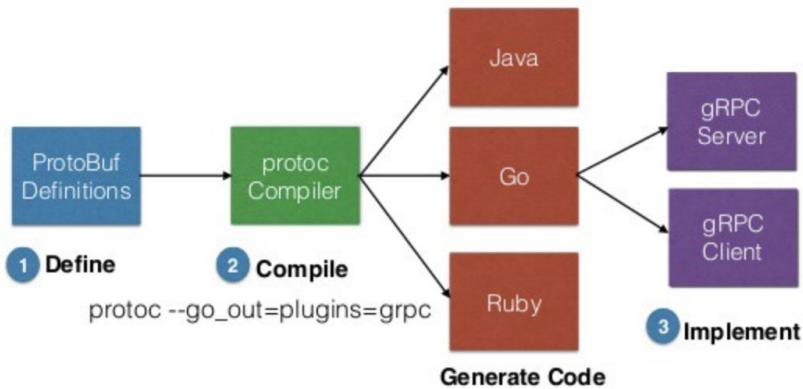
What's NEXT ?
Resources

{ REST }

The screenshot shows the Swagger UI interface for a REST API. At the top, there's a navigation bar with the Swagger logo, the URL '/swagger.json', and a green 'Explore' button. Below the navigation is a section titled 'documents'. The main content area lists various API endpoints with their methods, URLs, and descriptions. The endpoints are color-coded by method: GET (blue), POST (green), PUT (orange), DELETE (red), and PATCH (light green). The descriptions provide details like 'List collections in namespace' or 'Create a new document'.

Method	URL	Description
GET	/v2/namespaces/{namespace-id}/collections	List collections in namespace
POST	/v2/namespaces/{namespace-id}/collections	Create a new empty collection in a namespace
GET	/v2/namespaces/{namespace-id}/collections/{collection-id}	Search documents in a collection
POST	/v2/namespaces/{namespace-id}/collections/{collection-id}	Create a new document
DELETE	/v2/namespaces/{namespace-id}/collections/{collection-id}	Delete a collection in a namespace
POST	/v2/namespaces/{namespace-id}/collections/{collection-id}/upgrade	Upgrade a collection in a namespace
POST	/v2/namespaces/{namespace-id}/collections/{collection-id}/batch	Write multiple documents in one request
GET	/v2/namespaces/{namespace-id}/collections/{collection-id}/{document-id}	Get a document
PUT	/v2/namespaces/{namespace-id}/collections/{collection-id}/{document-id}	Create or update a document with the provided document-id
DELETE	/v2/namespaces/{namespace-id}/collections/{collection-id}/{document-id}	Delete a document
PATCH	/v2/namespaces/{namespace-id}/collections/{collection-id}/{document-id}	Update data at the root of a document
GET	/v2/namespaces/{namespace-id}/collections/{collection-id}/{document-id}/{document-path}	Get a path in a document
PUT	/v2/namespaces/{namespace-id}/collections/{collection-id}/{document-id}/{document-path}	Replace data at a path in a document
DELETE	/v2/namespaces/{namespace-id}/collections/{collection-id}/{document-id}/{document-path}	Delete a path in a document
PATCH	/v2/namespaces/{namespace-id}/collections/{collection-id}/{document-id}/{document-path}	Update data at a path in a document
GET	/v2/namespaces/{namespace-id}/collections/{collection-id}/json-schema	Get a JSON schema from a collection

gRPC





It is a web protocol (HTTP) that specifies the way we build and query remote APIs using a **Tree/JSON Syntax**.

Based on a **strongly typed** language named GraphQL SDL (Schema Definition Language)

```
incomingWorkshops {  
  title  
  abstract  
  speakers  
}
```

```
type Workshop {  
  title: String!  
  abstract: String  
  speakers: [Speaker]  
  releaseYear: int  
}
```



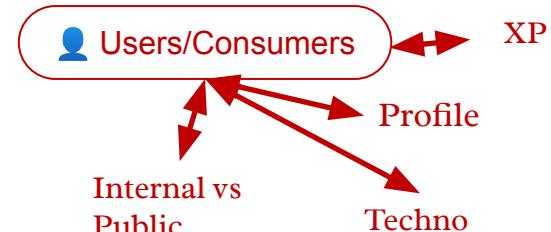
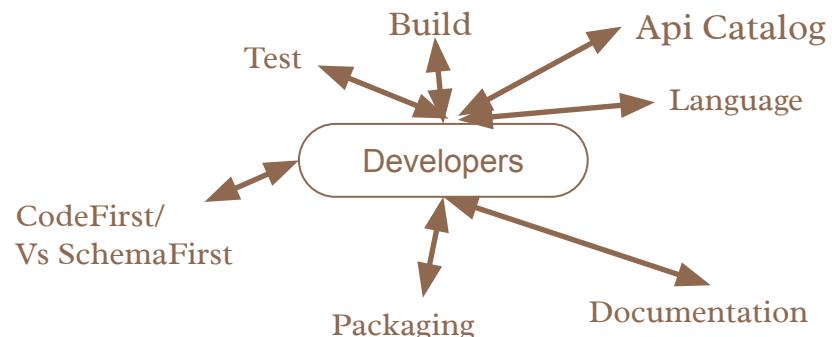
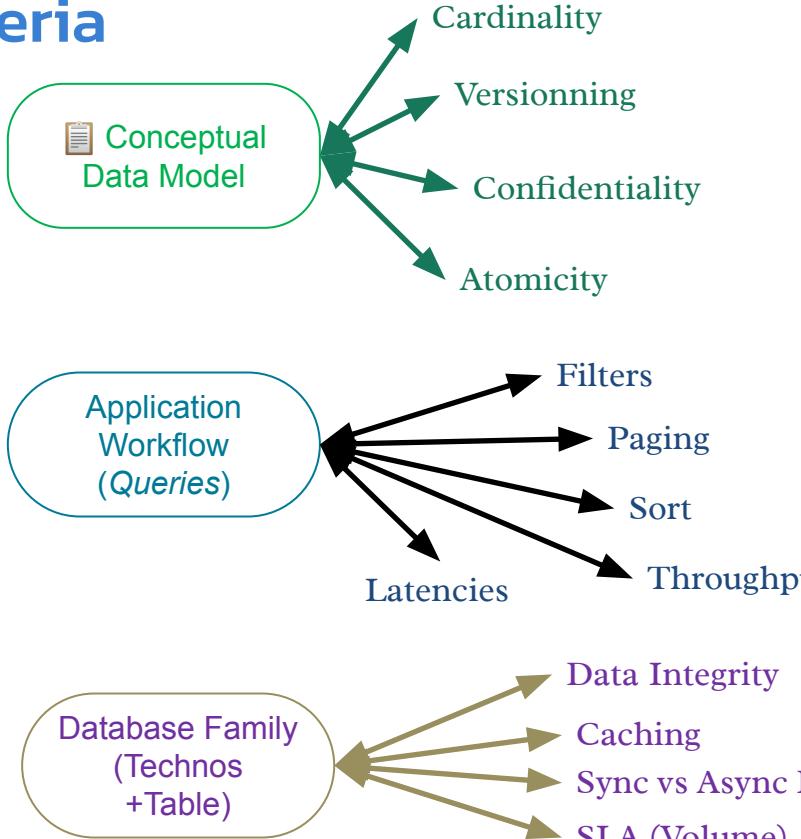
A protocol that allows the client to specify exactly **what** data it needs from a model

```
incomingWorkshops {  
    title  
}
```

It allows one to aggregate data from **multiple relations** in a single query

```
incomingWorkshops {  
    title: String!  
    abstract: String  
    speakers: {  
        name  
    }  
    releaseYear: int  
}
```

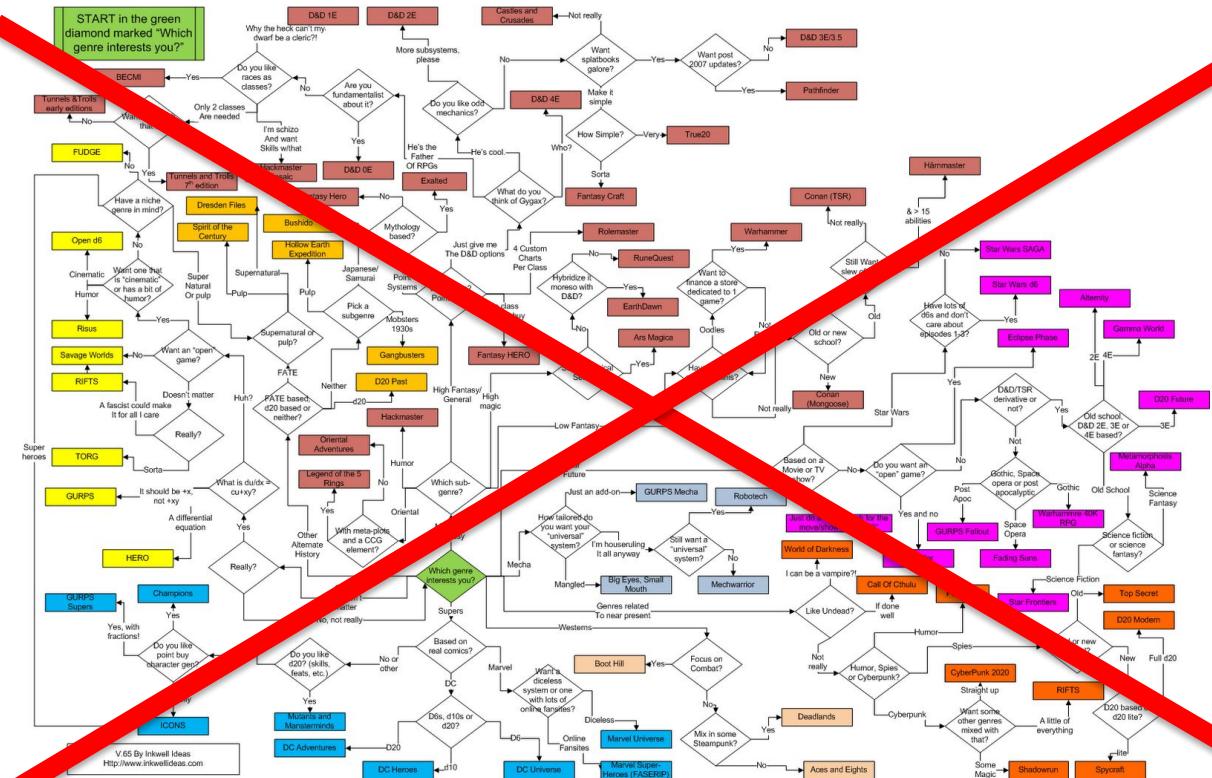
Criteria



Matrix Analysis

Concept	GraphQL	REST	gRPC
Modèle conceptuel de données (Entités, Relations)			
Cardinalité	1	5	3
Versionning	1	8	9
Confidentialité	6	9	1
Atomicité	6	7	0
Parcours Applicatifs			
Filtres	2	3	4
Pagination	3	3	6
Tris	4	4	6
Throughput	4	3	6
Latence	6	4	6
Base de données	2	3	6
Intégrité	6	4	7
Cache	7	0	6
Sync vs Async vs Reactive	1	2	4
SLA	6	3	4
Development			
Language	1	4	5
Api Catalog	6	7	8
Test	3	2	0
Build	4	5	6
Packaging	4	6	7
Code first / Schema Frist	5	7	8
Documentation	2	5	1
Client			
XP	1	4	7
Profil	3	3	9
Techno	4	5	7
Interne vs Extern	4	5	8

Decision Tree



{ REST }



- ❖ Decoupling Client / Server (*Schema on read*)
- ❖ Api Lifecycle (*Versioning*)
- ❖ Tooling (*API Management, Serverless*)



- ❖ Verbose payloads (*json, xml*)
- ❖ No discoverability
- ❖ Not suitable for command-like (functions) API



- ❖ CRUD superstar
- ❖ Relevant for mutations (OLTP)
- ❖ Public and web APIs
- ❖ Limited Business Scope



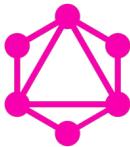
- ❖ High Performances (*http/2 – binary serialisation*)
- ❖ Multiple stubs : Sync, Async, Streaming
- ❖ Multi languages - Interoperability



- ❖ Strongly coupled (*schema with proto files*)
- ❖ No discoverability
- ❖ *Protobuf* serialization format



- ❖ Distributed network of services (no waits)
- ❖ High throughput & streaming use cases
- ❖ Command-like (eg: *slack*)



GraphQL



- ❖ Discoverability, documentation
- ❖ Custom payloads
- ❖ Match standards (Json | Http)



- ❖ Single endpoint (*versioning, monitoring, security*)
- ❖ Complex implementation (*tooling, still young*)
- ❖ Nice for customers nasty for DB (*N+1 select*)



- ❖ Backend for frontend (JS)
- ❖ Service aggregation | composition (*joins*)
- ❖ When volume matters (*mobile phones*)

Agenda

01

Api and Databases
What, Why and How

02

Rest, GraphQL, gRPC
SWOT

03

Data Gateway
Why and How

04

Stargate.io
Architecture

05

Demo
Api and Tooling

05

What's NEXT ?
Resources

Data Gateway



Developers want the option to use modern APIs and development gateways.

Cassandra is a database designed for the new standard and the associated APIs that facilitate the first choice of developers.



MICROSERVICES
DEVELOPERS



GraphQL

{REST:API}



OPENAPI
INITIATIVE



JSON



GRPC



Ubiquitous, API-based Consumption



Stargate Overview

An open source API framework for data

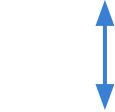
Stargate makes it easy to use a database for any application workload by adding plugin support for new APIs, data types, and access methods

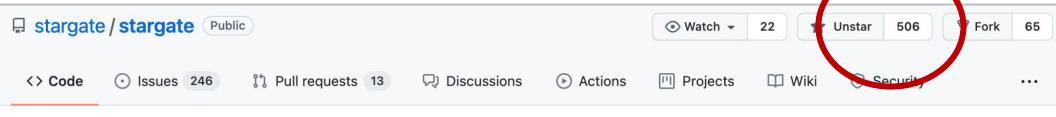


MICROSERVICES
DEVELOPERS



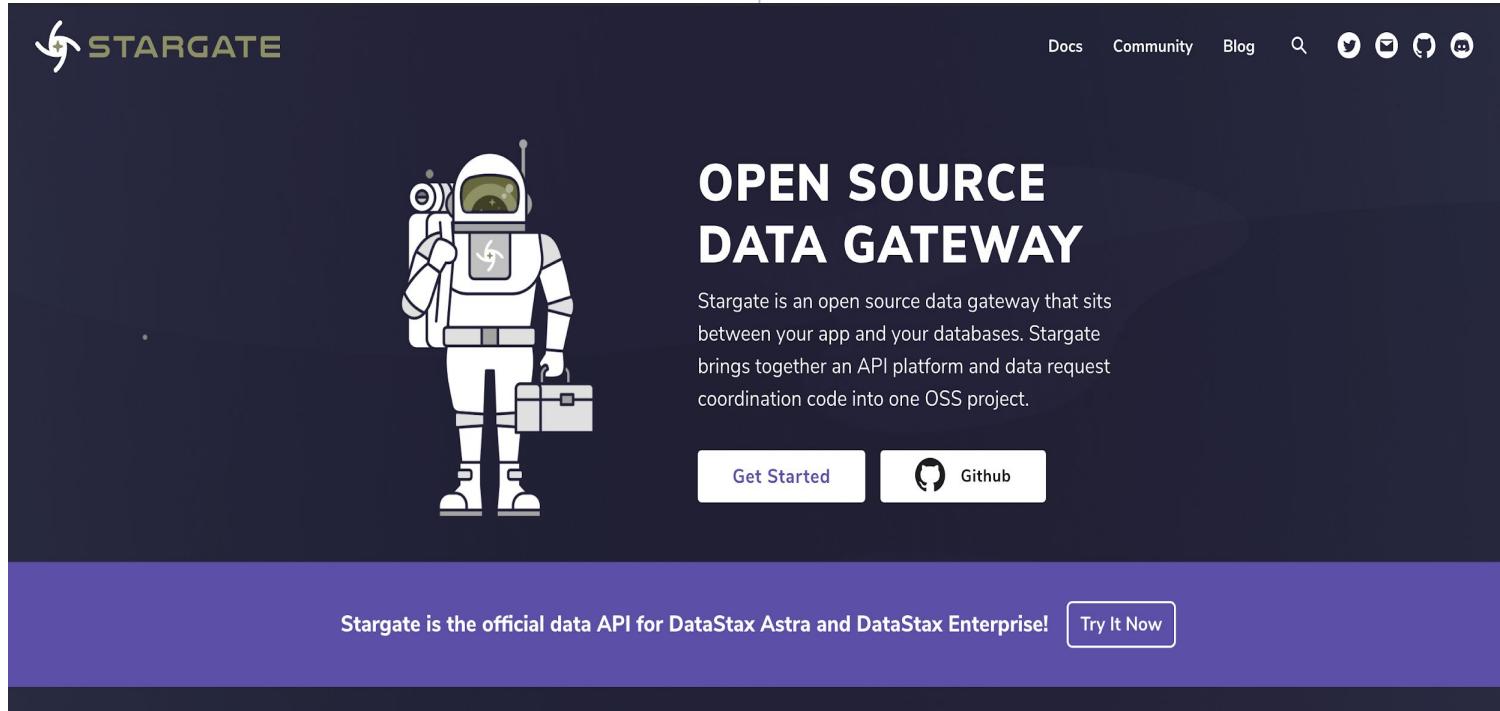
STARGATE





stargate / stargate Public

Code Issues 246 Pull requests 13 Discussions Actions Projects Wiki Security ...



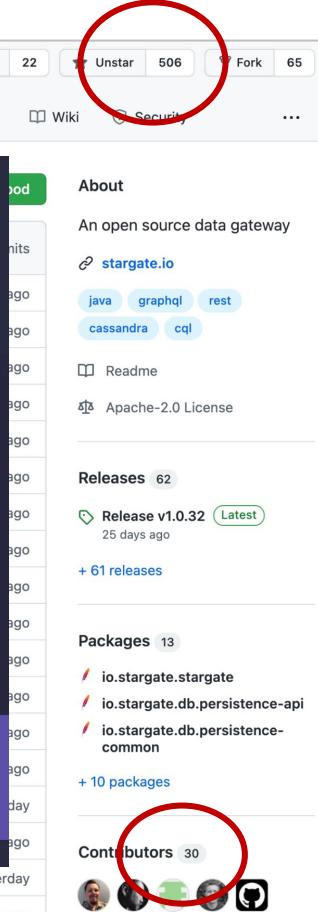
OPEN SOURCE DATA GATEWAY

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 

Stargate is the official data API for DataStax Astra and DataStax Enterprise! Try It Now

grpc Reference example for gRPC java client (#1236) yesterday



About
An open source data gateway

stargate.io
java graphql rest
cassandra cql

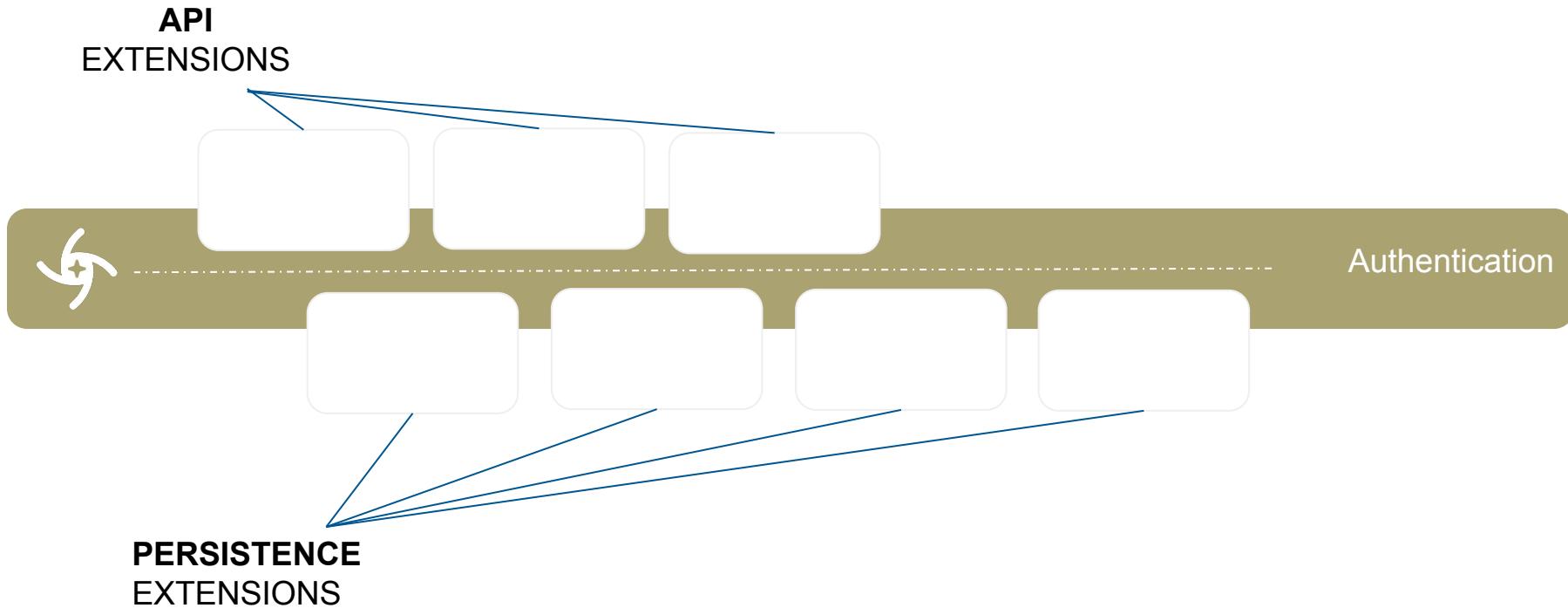
Readme
Apache-2.0 License

Releases 62
Release v1.0.32 Latest 25 days ago + 61 releases

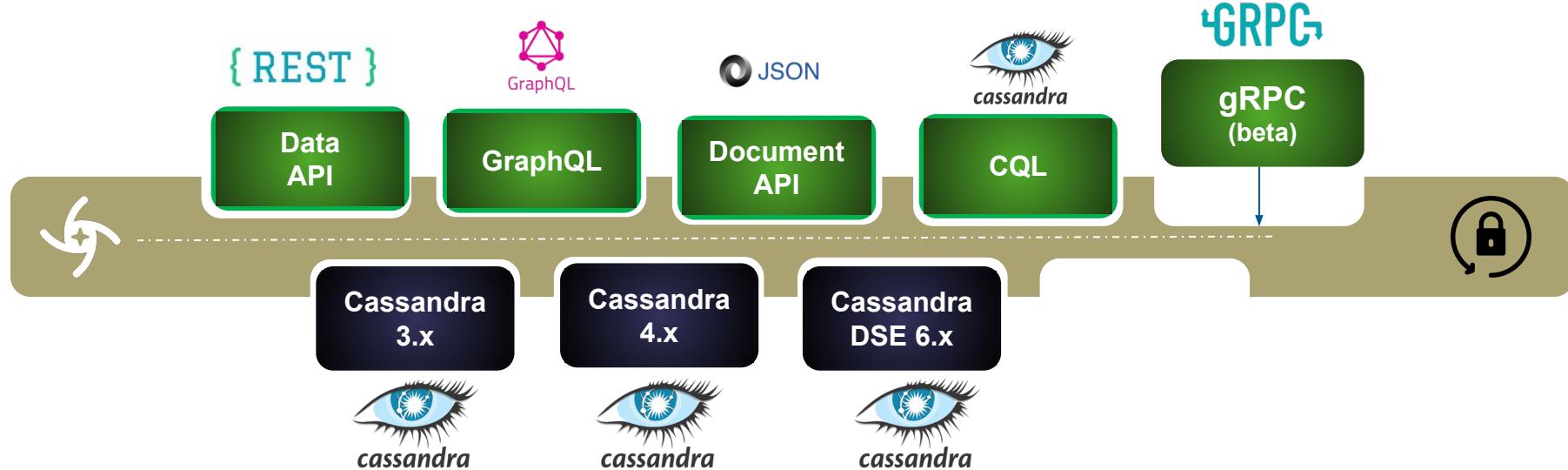
Packages 13
io.stargate.stargate
io.stargate.db.persistence-api
io.stargate.db.persistence-common
+ 10 packages

Contributors 30

API and Persistence Extensions



API Extensions and Persistence Extensions



Agenda

01

Api and Databases
What, Why and How

02

Rest, GraphQL, gRPC
SWOT

03

Data Gateway
Why and How

04

Stargate.io
Architecture

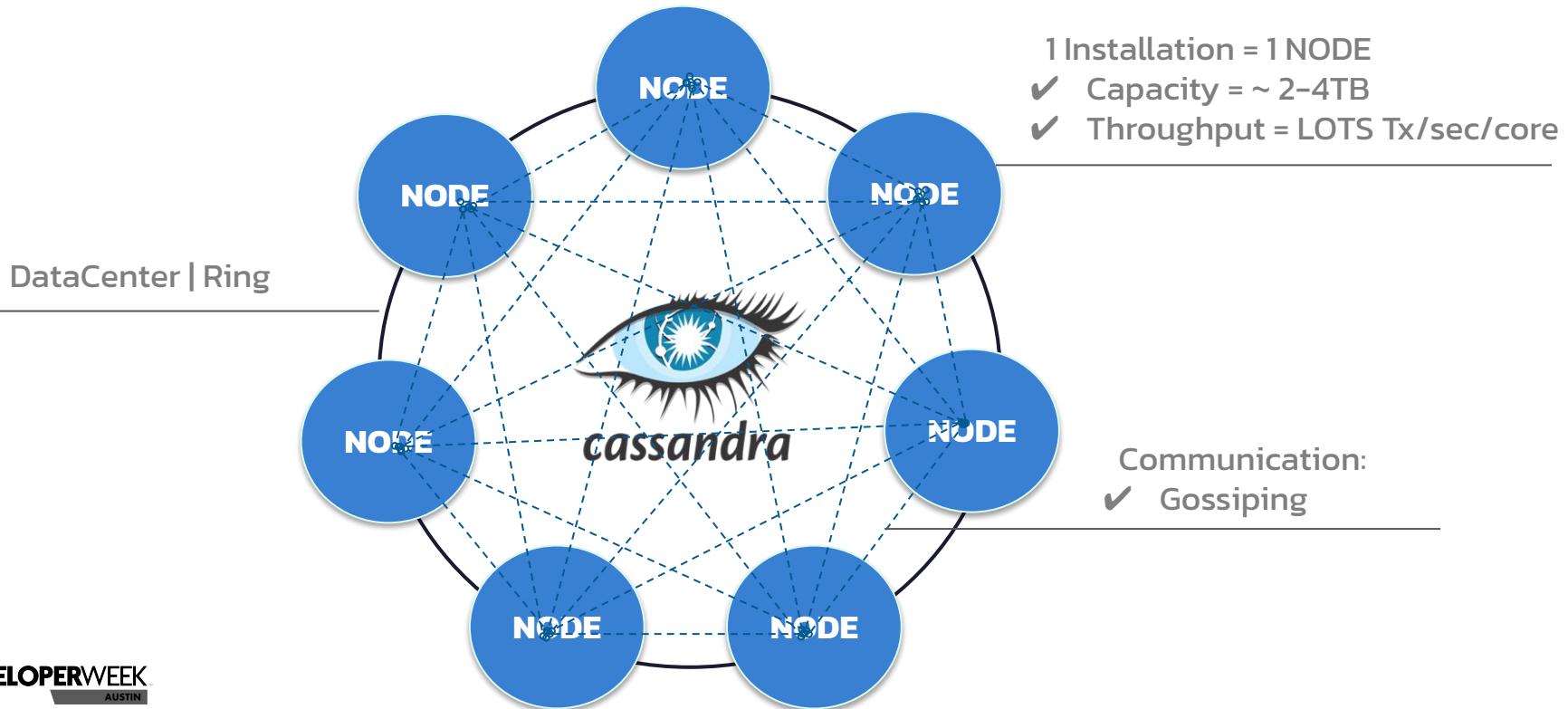
05

Demo
Api and Tooling

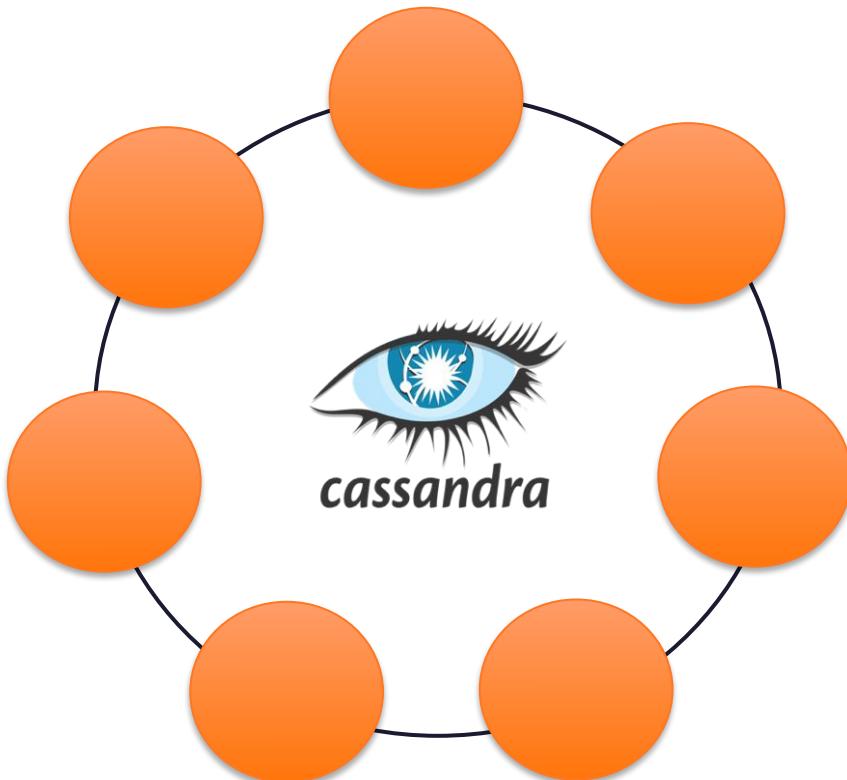
05

What's NEXT ?
Resources

Apache Cassandra™ = NoSQL Distributed Database



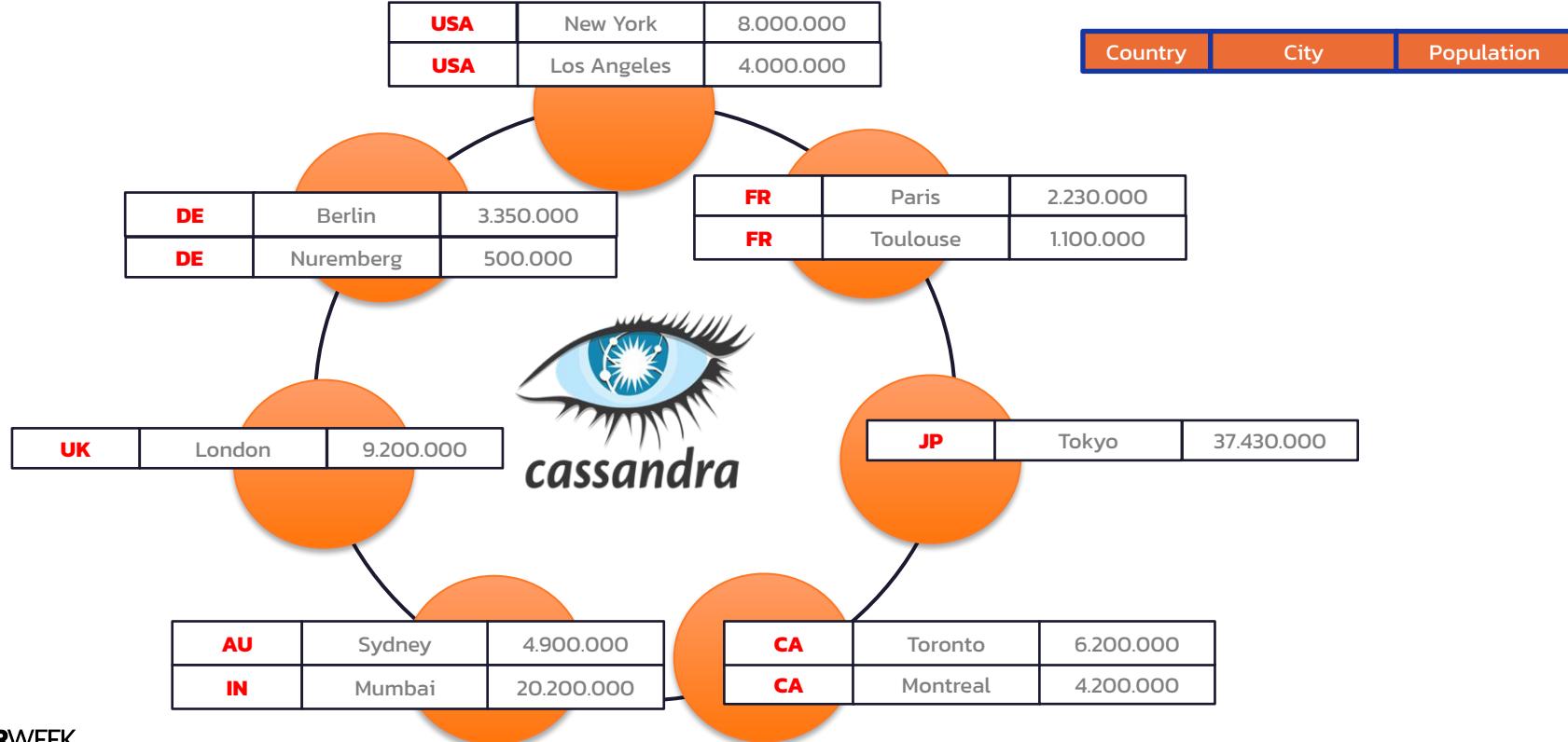
Data is Distributed



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

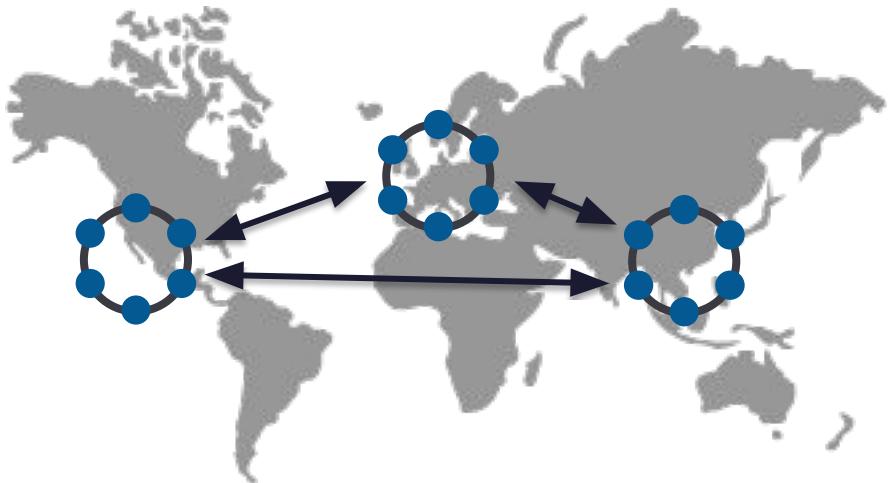
Partition Key

Data is Distributed

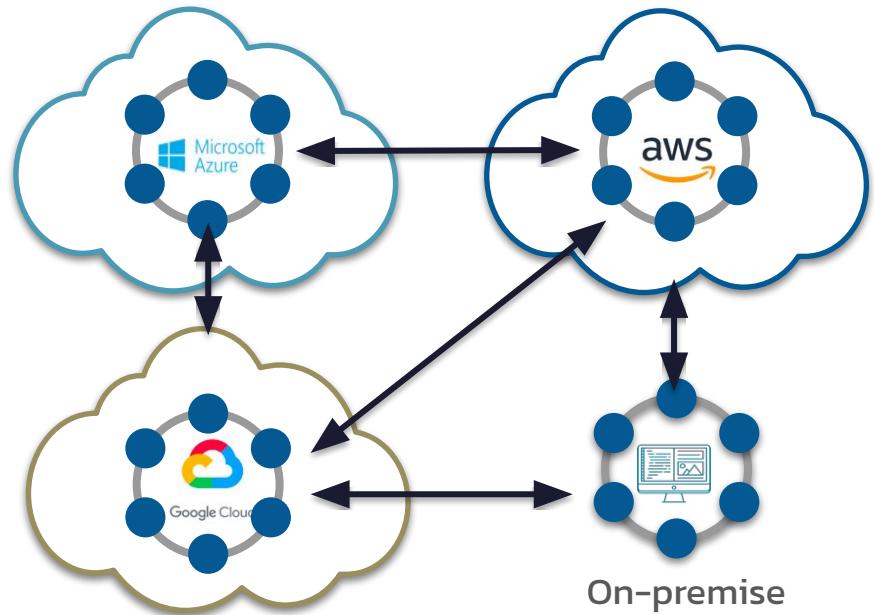


Data Distributed Everywhere

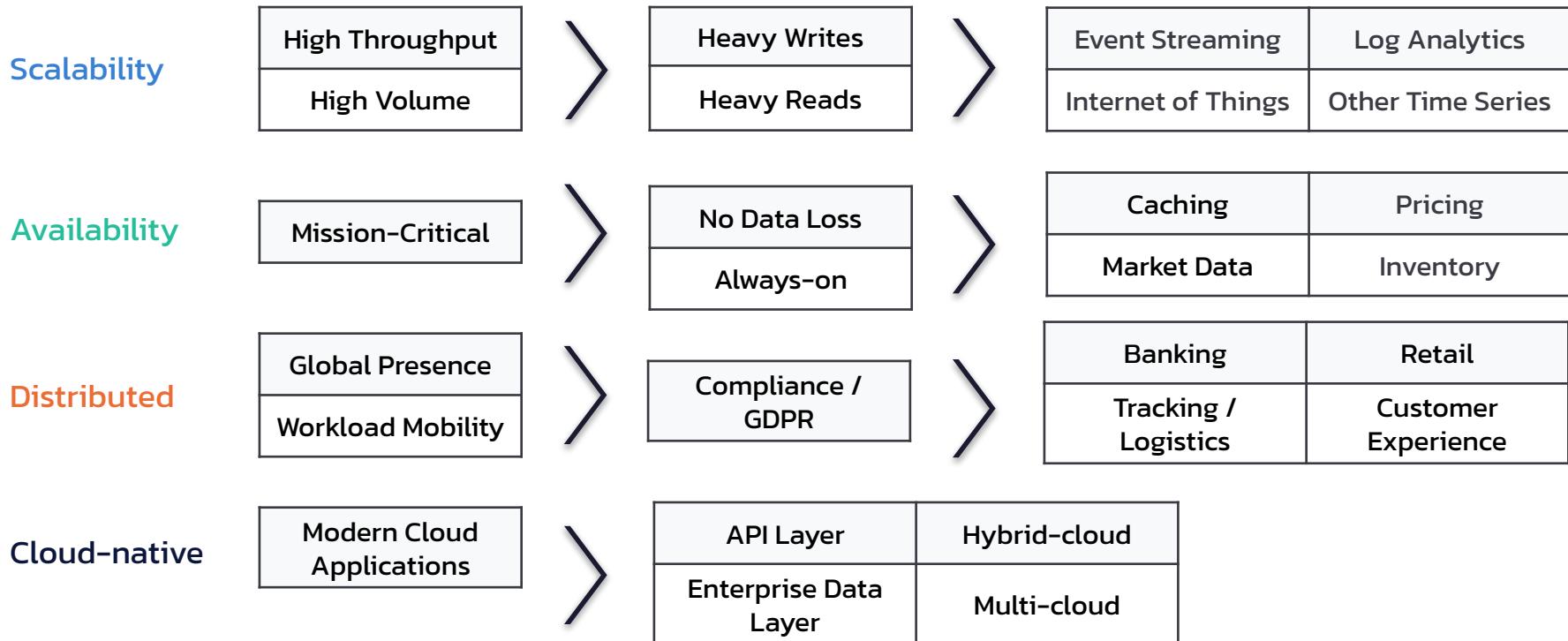
- Geographic Distribution



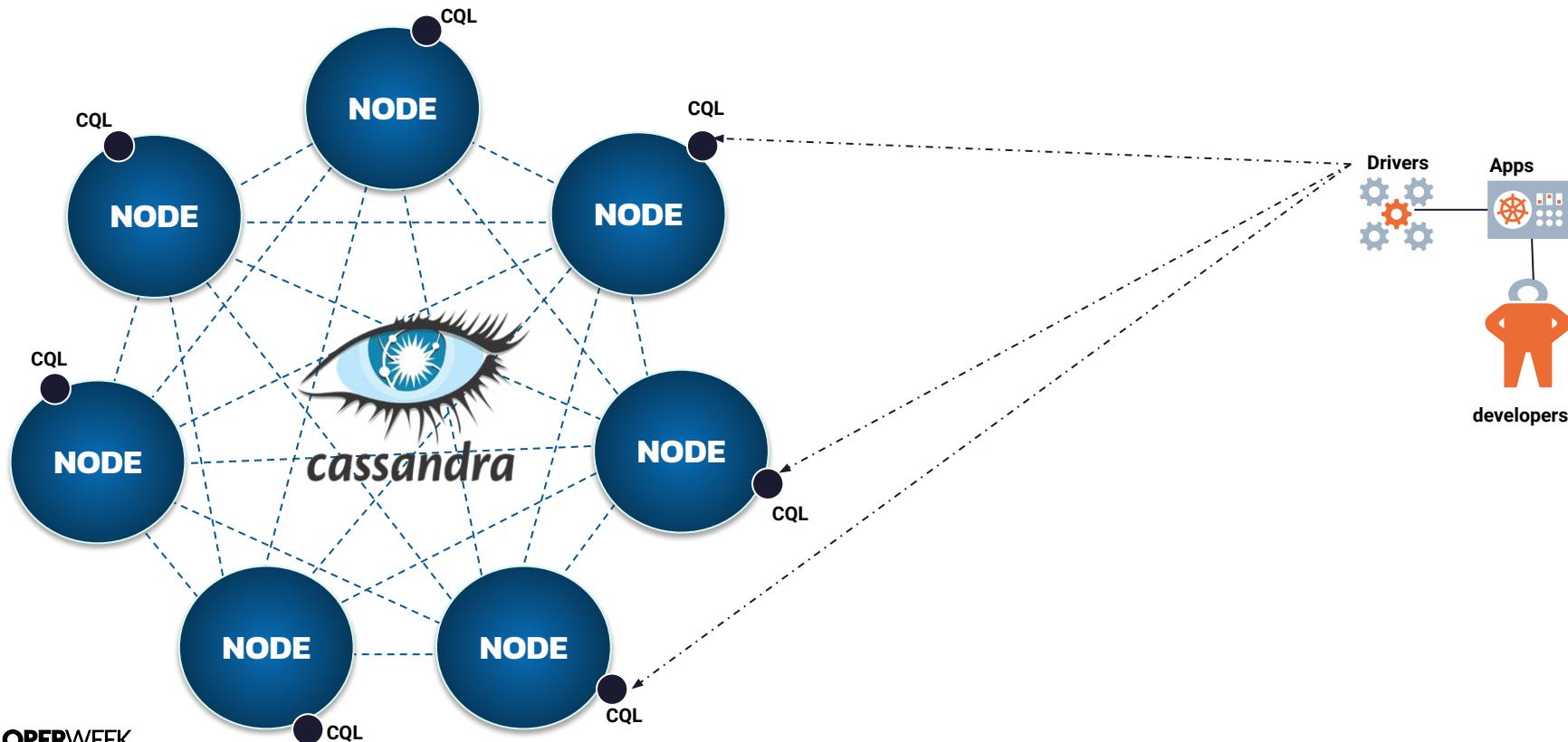
- Hybrid-Cloud and Multi-Cloud



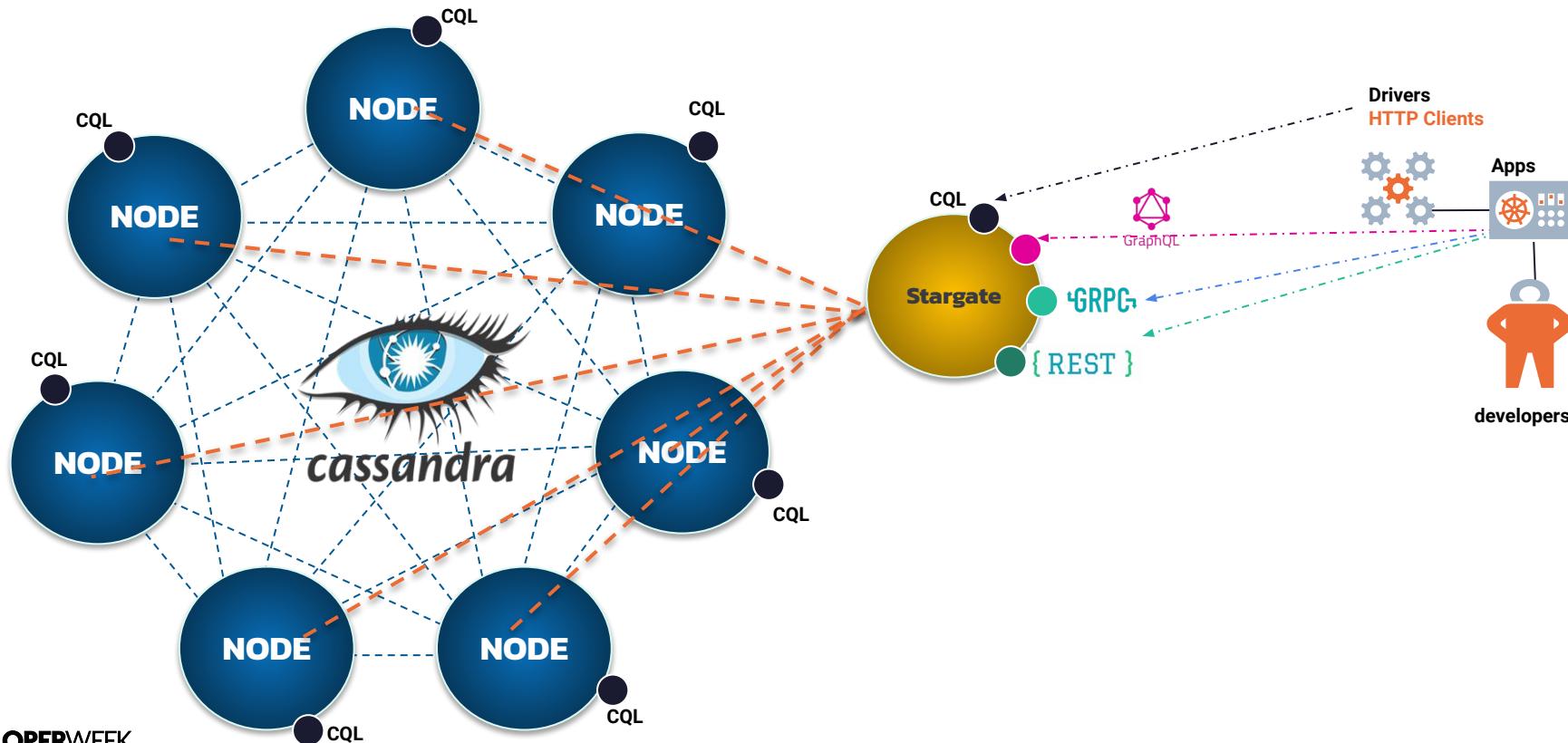
Understanding Use Cases



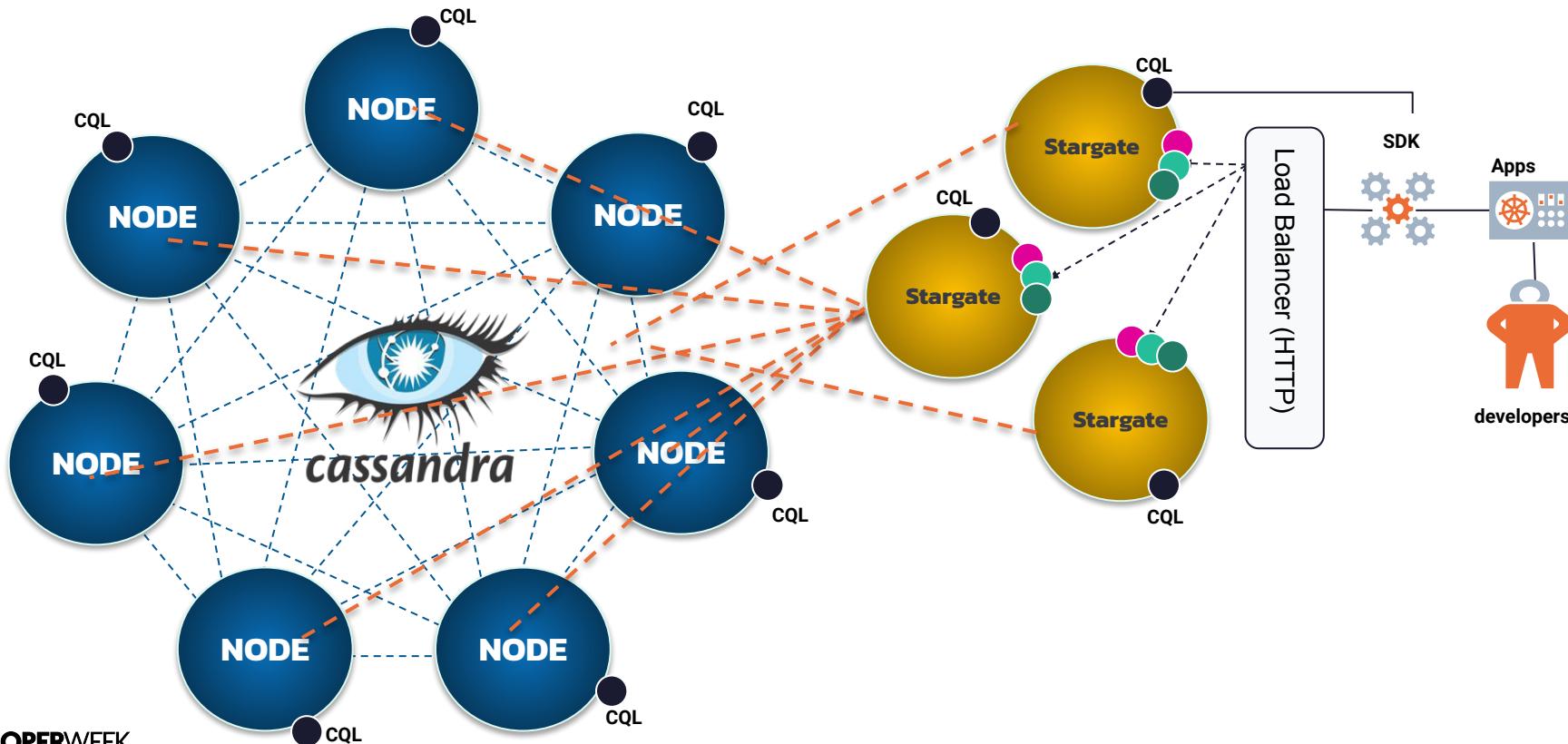
Connecting to your cluster (Before)



Connecting to your cluster (with Stargate)



Connecting to your cluster (with Stargates)



Agenda

01

Api and Databases
What, Why and How

02

Rest, GraphQL, gRPC
SWOT

03

Data Gateway
Why and How

04

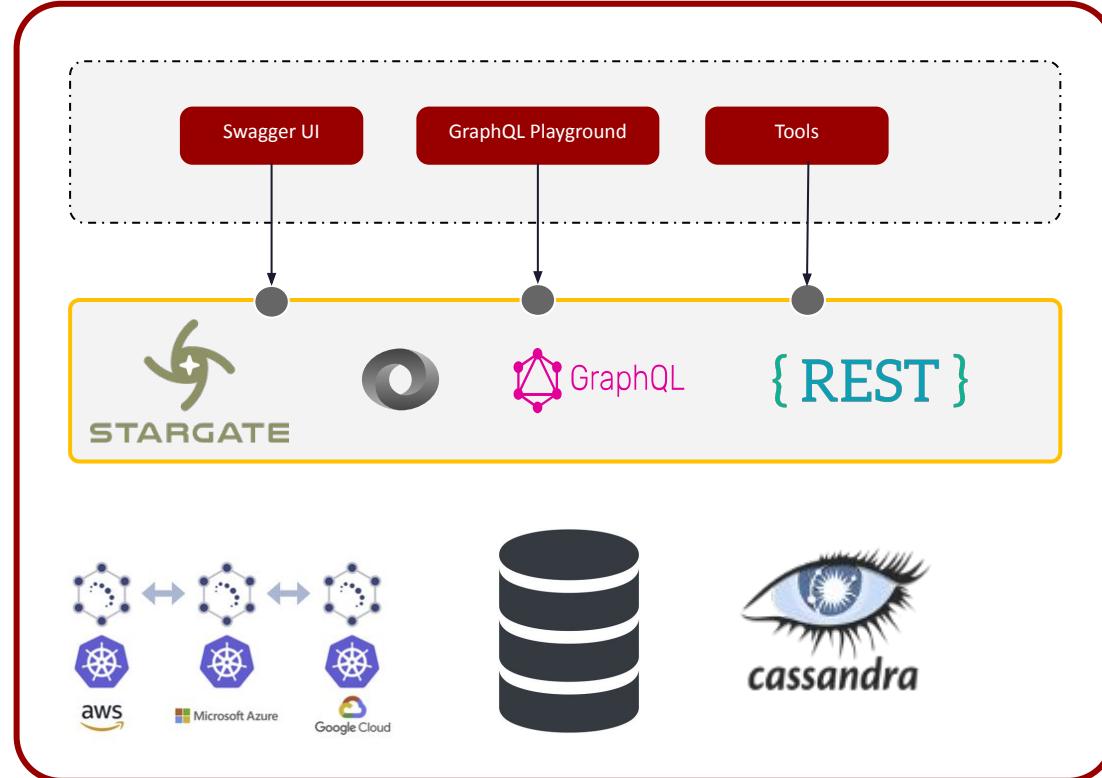
Stargate.io
Architecture

05

Demo
Api and Tooling

05

What's NEXT ?
Resources



HouseKeeping

- Instructions and Slide
 - <https://github.com/datastaxdevs/conference-2021-developer-week-austin>
 -
- Runtime
 - <https://astra.datastax.com>

1. Create Astra Instance

ASTRA is the simplest way to run Cassandra with zero operations at all - just push the button and get your cluster. No credit card required, \$25.00 USD credit every month, roughly 5M writes, 30M reads, 40GB storage monthly - sufficient to run small production workloads.

Register (if needed) and Sign In to Astra <https://astra.datastax.com>: You can use your Github, Google accounts or register with an email.

Make sure to chose a password with minimum 8 characters, containing upper and lowercase letters, at least one number and special character

Create a "pay as you go" plan

Follow this [guide](#), to set up a pay as you go database with a free \$25 monthly credit.

- Select the pay as you go option: Includes \$25 monthly credit - no credit card needed to set up.

You will find below which values to enter for each field.

- For the database name - `free_db`. While Astra allows you to fill in these fields with values of your own choosing, please follow our recommendations to ensure the application runs properly.

- For the keyspace name - `free`. It's really important that you use the name "free" for the code to work.

You can technically use whatever you want and update the code to reflect the keyspace. This is really to get you on a happy path for the first run.

- For provider and region: Choose and provider (either GCP or AWS). Region is where your database will reside physically (choose one close to you or your users).

Hands-on #1/2 Getting your instance

1. Create Astra Instance

Astra is the simplest way to run Cassandra with zero operations at all - just push the button and get your cluster. No credit card required, \$25.00 USD credit every month, roughly 5M writes, 30M reads, 40GB storage monthly - sufficient to run small production workloads.

- Register (if needed) and Sign In to Astra <https://astra.datastax.com>: You can use your Github, Google accounts or register with an email.

Make sure to choose a password with minimum 8 characters, containing upper and lowercase letters, at least one number and special character

- Create a "pay as you go" plan

Follow this [guide](#), to set up a pay as you go database with a free \$25 monthly credit.

- **Select the pay as you go option:** Includes \$25 monthly credit - no credit card needed to set up.

You will find below which values to enter for each field.

- **For the database name** - `free_db`. While Astra allows you to fill in these fields with values of your own choosing, please follow our recommendations to ensure the application runs properly.
- **For the keyspace name** - `keyspace1`. It's really important that you use the name "free" for the

2. Working with Cassandra

- Check that our keyspace exist

```
describe keyspaces;
```

- Create Entities

```
use keyspace1;
```

```
CREATE TYPE IF NOT EXISTS video_format (
    width   int,
    height  int
);
```

```
CREATE TABLE IF NOT EXISTS videos (
    videoid  uuid,
    title    text,
    upload   timestamp,
    email   text,
    url     text,
    tags    set <text>,
    frames  list<int>,
    formats map <text,frozen<video_format>>,
    PRIMARY KEY (videoid)
);
```

```
describe keyspace1;
```

Agenda

01

Api and Databases
What, Why and How

02

Rest, GraphQL, gRPC
SWOT

03

Data Gateway
Why and How

04

Stargate.io
Architecture

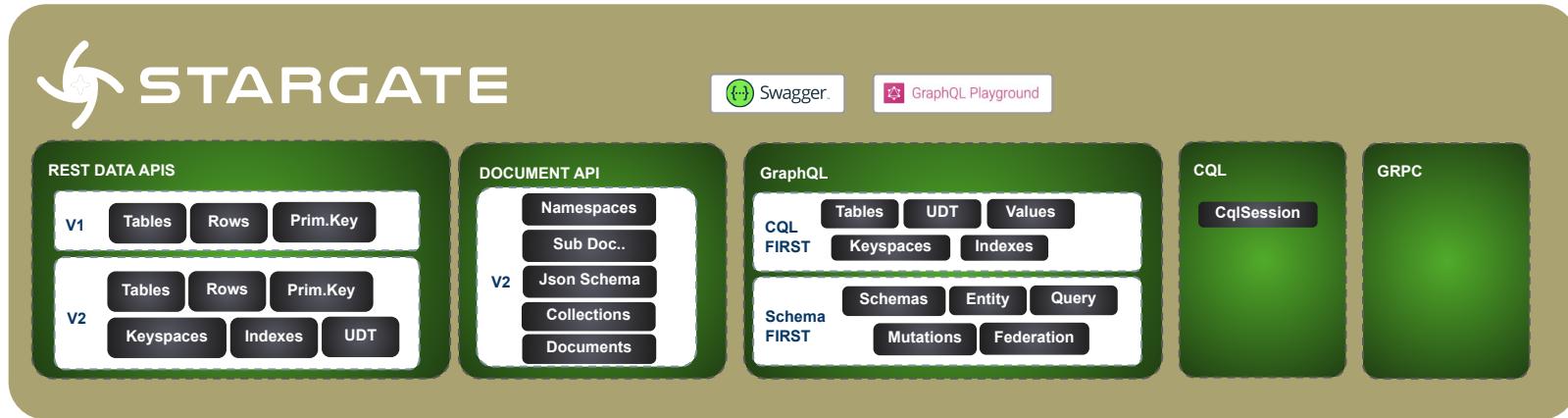
05

Demo
Api and Tooling

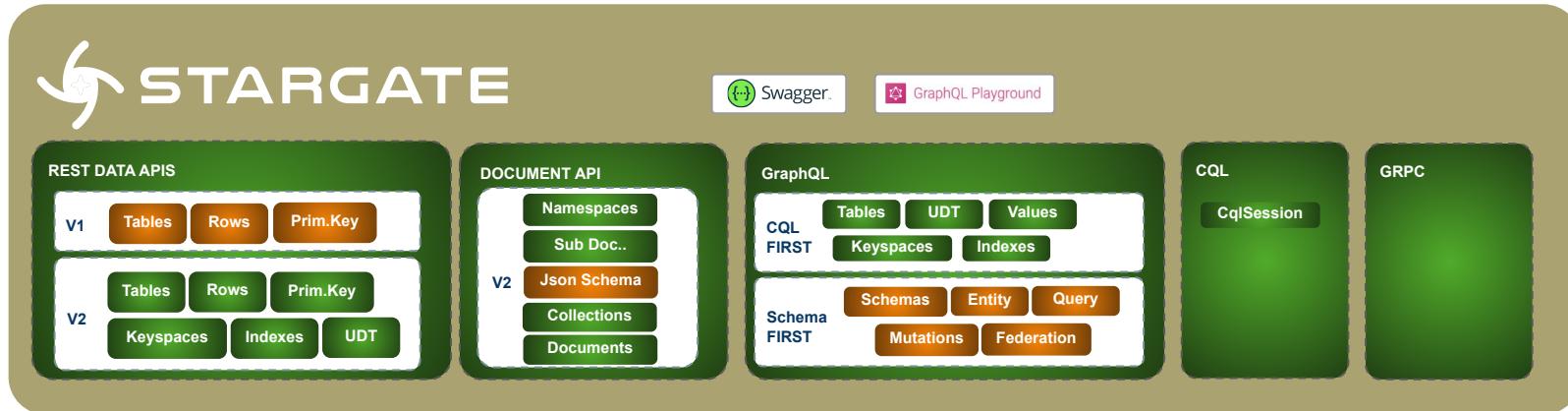
05

What's NEXT ?
Resources

Features Map



SDK in Java, Javascript, Python



Sample Codes



```
StargateClient client = StargateClient.builder()
    .username("k8ssandra-superuser")           // Mandatory username
    .password("JxzrPOnvDGqfEOQ0EySQ")          // Mandatory password
    .endPointAuth("http://localhost:8081")       // Mandatory authentication url, defaulting to http://localhost
    .endPointRest("http://localhost:8082")        // Rest and Document APIs
    .endPointGraphQL("http://localhost:8080")     // GraphQL API

    // Cqlsession Only
    .addCqlContactPoint("127.0.0.", 9042)      // Contact Point
    .localDc("dc1")                            // Local Datacenter is mandatory driver 4xx+
    .keypace("ks1")                            // (optional) Set your keyspace
    .build();
```

Sample Code

```
// Retrieve an object and marshall
Optional<Address> address = colPersonClient
    .document("e8c5021b-2c91-4015-aec6-14a16e449818")
    .findSubDocument("address", Address.class);

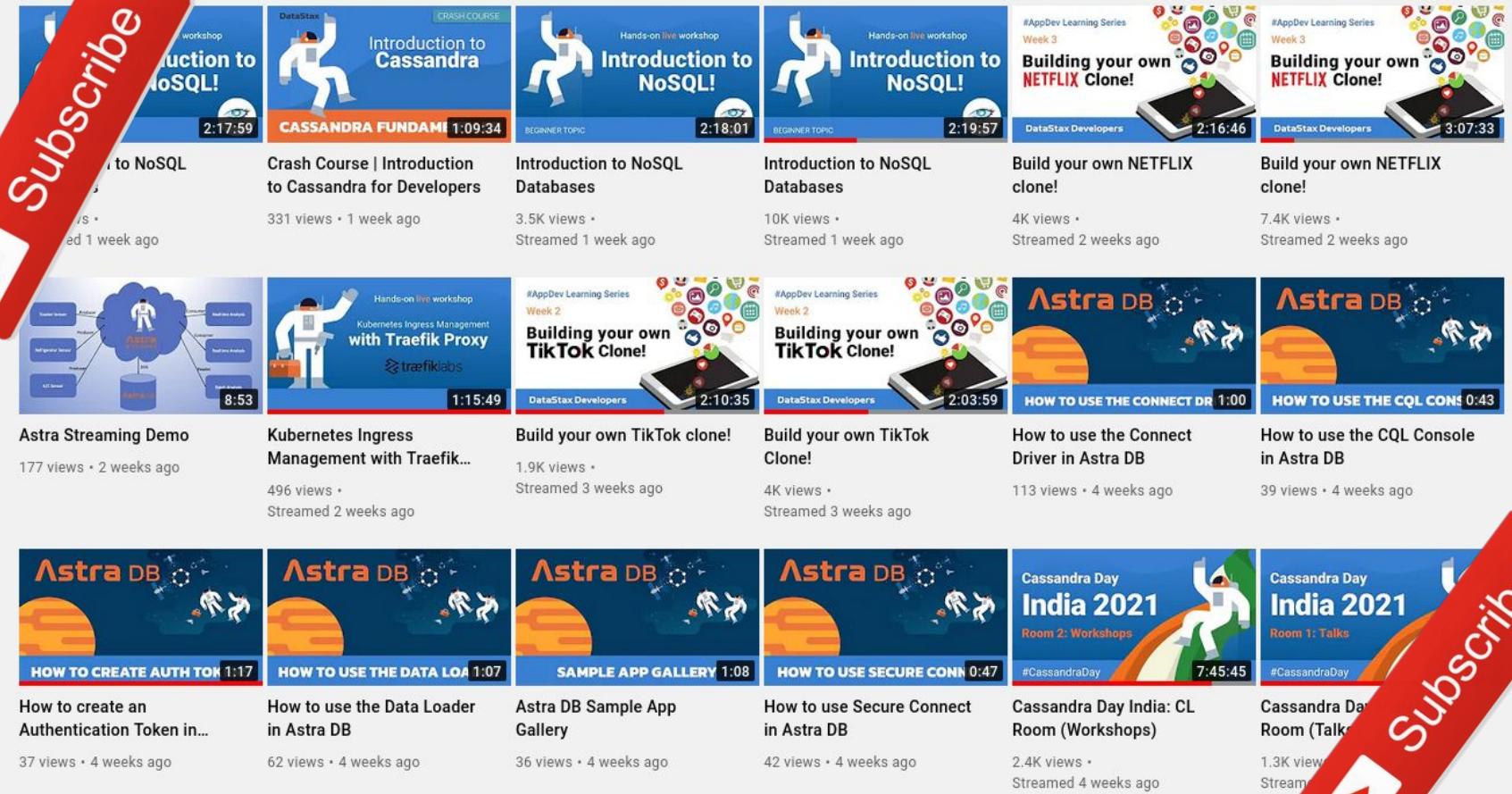
// Building query {"age": {"$gte":30}, "lastname": {"$eq": "PersonAstra2"}}
SearchDocumentQuery query = SearchDocumentQuery.builder()
    .where("age").isGreaterOrEqualsThan(30)      // First filter to use where()
    .and("lastname").isEqualTo("PersonAstra2")   // Any extra filter to use and()
    .withPageSize(10)                            // Default and max pageSize are 20
    .build();

// Retrieve PAGE 1
DocumentResultPage<Person> currentPage = colPersonClient.findPage(query, Person.class);

// Retrieve PAGE 2 (if any)
if (currentPage.getPageState().isPresent()) {
    query.setPageState(currentPage.getPageState().get());
}
DocumentResultPage<Person> nextPage = colPersonClient.findPage(query, Person.class);

// Retrieve all documents in one call (with warning in RED ABOVE)
Stream<Person> allPersons = colPersonClient.findAll(query, Person.class)
```

Subscribe



Subscribe

DataStax Developers

DataStax Developers

Thank you!



GitHub

@clun



@clunven

