DataStax

Introduction to GraphQL

LEVEL UP with the DataStax Developers

DatastaxDevs: Developer Advocates Team

Cedrick Lunven

Aleksandr Volochnev

Jack Fryer

Kirsten Hunter

Stefano Lottini

David Gilardi

Ryan Welford

Rags Srinivas

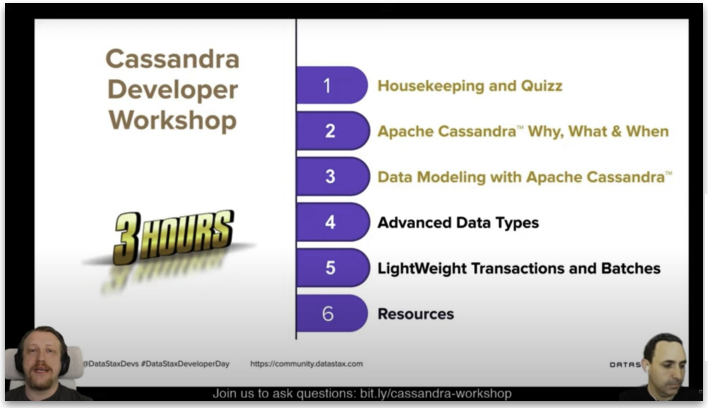
L E S T S

DataStax Developers

Housekeeping #1: Attending the workshop

Live and interactive

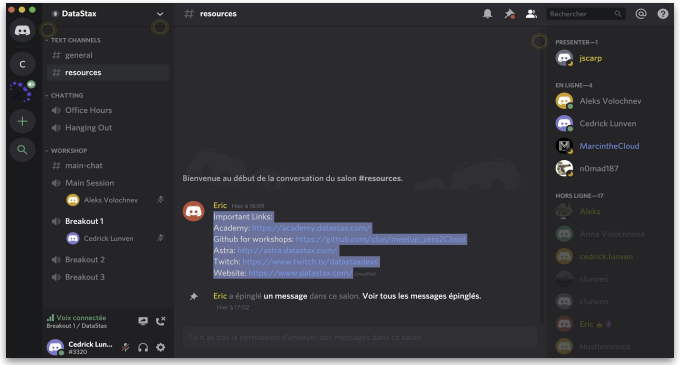
Livestream: youtube.com/DataStaxDevs



YouTube

Twitch

Questions: https://dtsx.io/discord

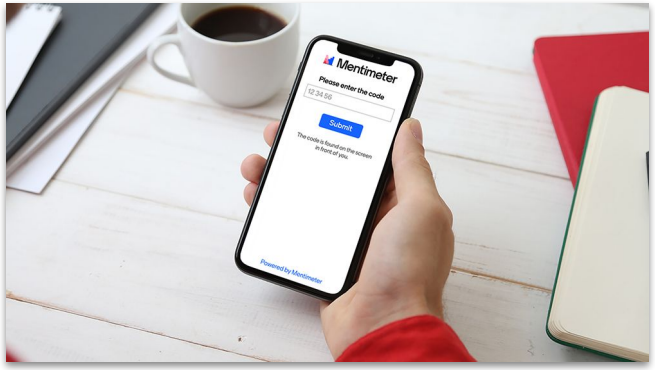


Discord

YouTube

Available on the iPhone App Store

Games menti.com



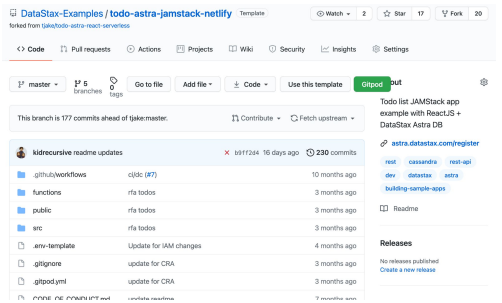
Mentimeter

GET IT ON Google play

Housekeeping #2: Doing Hands-On

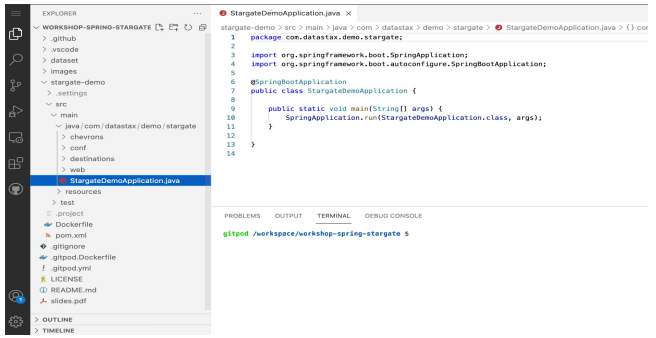
Nothing to install !

Source code + exercises + slides



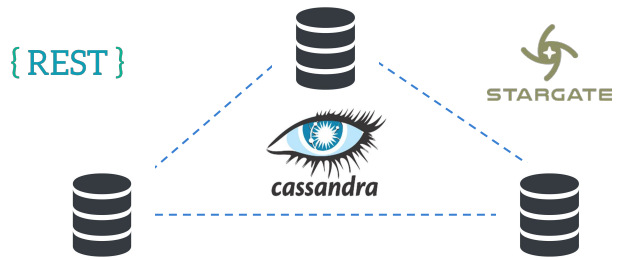
GitHub

IDE



Gitpod

Database + GraphQL + PlayGround

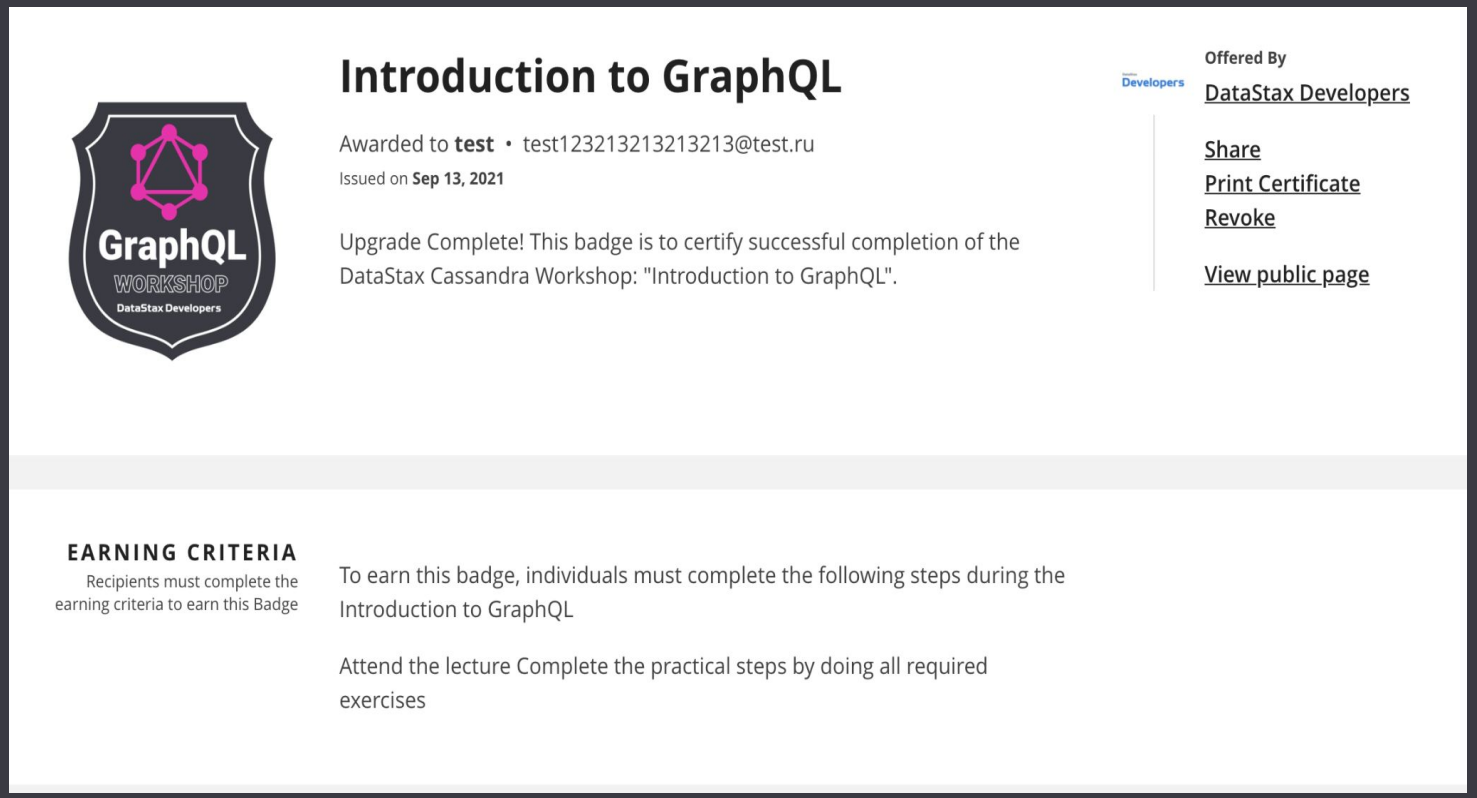


DataStax Astra DB

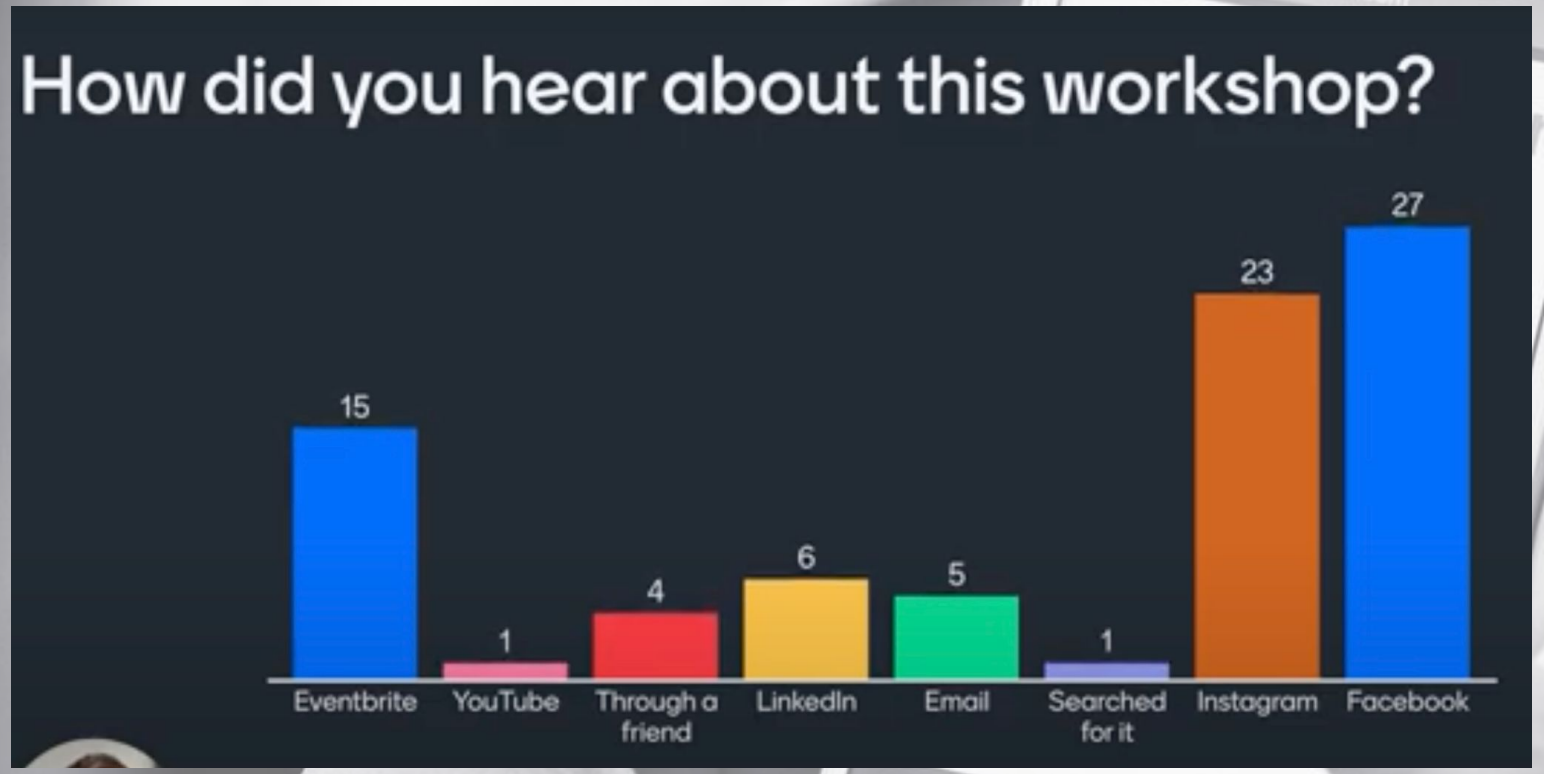
DataStax Developers

A Badge when completing homeworks. No certificate of attendance.

New Badge !



menti.com

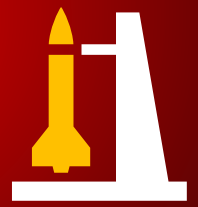


Mentimeter

Agenda

01 DataStax Astra

DB Setup and Credentials



02

Introduction to GraphQL Design

03

GraphQL Tooling

04 DGS

Backend

Expose GQL Endpoint

05

Frontend

Consume GQL Endpoint

06

Resources

Demo:

- Let's see the apps

GitHub

https://github.com/datastaxdevs/workshop-intro-to-graphql

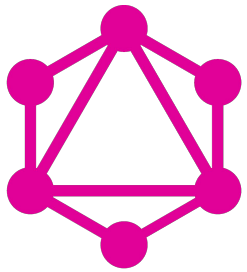
History

2012 Created by Facebook
Used internally for mobile apps

2015 Facebook give talk at ReactJs Conf

2015 Facebook open sourced graphQL

2016 Github announced move to GQL



GraphQL

Definition

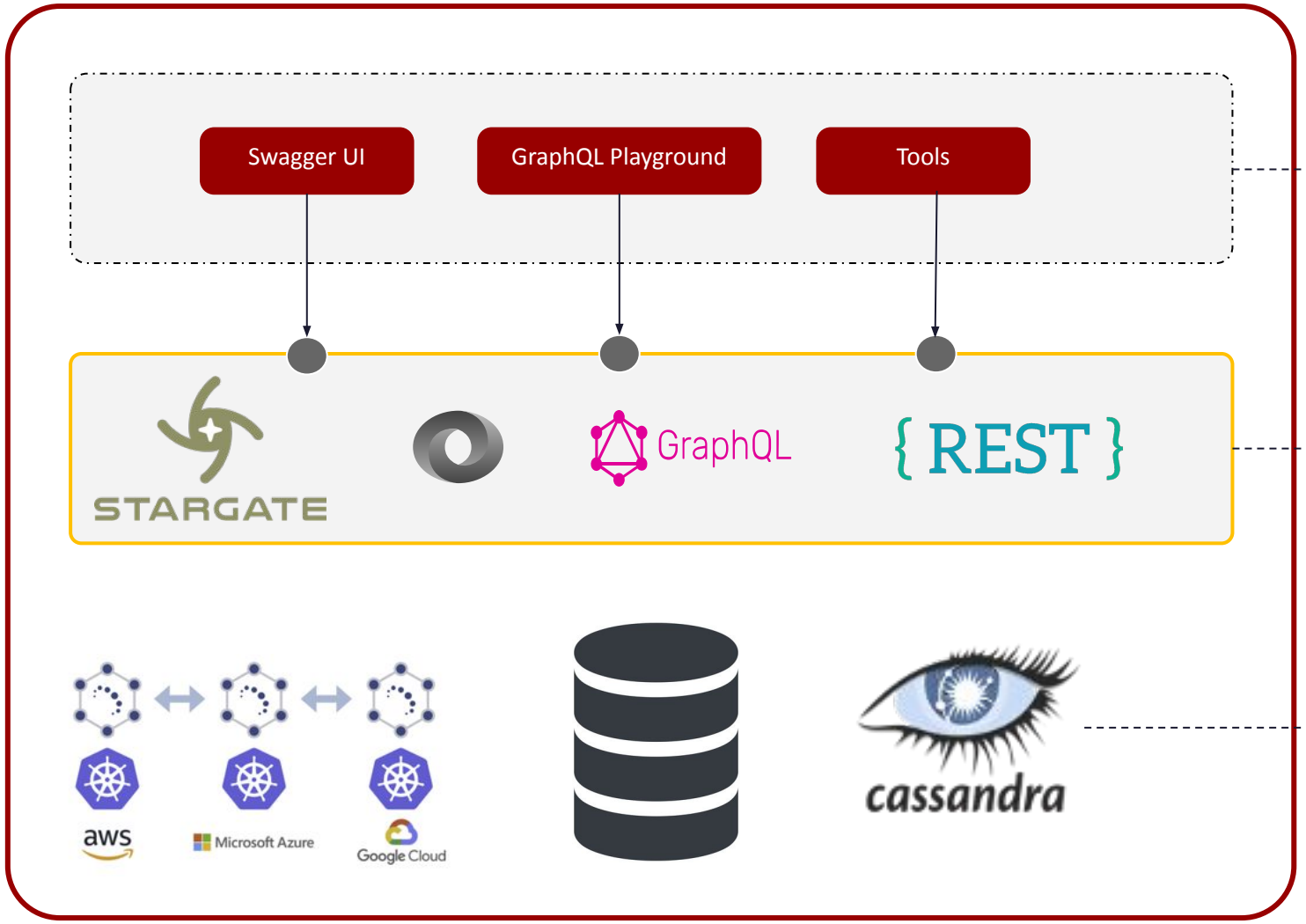
GraphQL is an application programming interface (API) query language and server-side runtime that prioritises giving customers precisely the data they request.

Why ?


GraphQL
Describe your data
type Project {
name: String
tagline: String
contributors: [User]
}
Ask for what you want
{
project(name: "GraphQL") {
tagline
}
}
Get predictable results
{
"project": {
"tagline": "A query language for APIs"
}
}

DataStax

Astra DB



User Interface
Web Utils

OSS Stargate.io
Gateway

STARGATE

Apache Cassandra

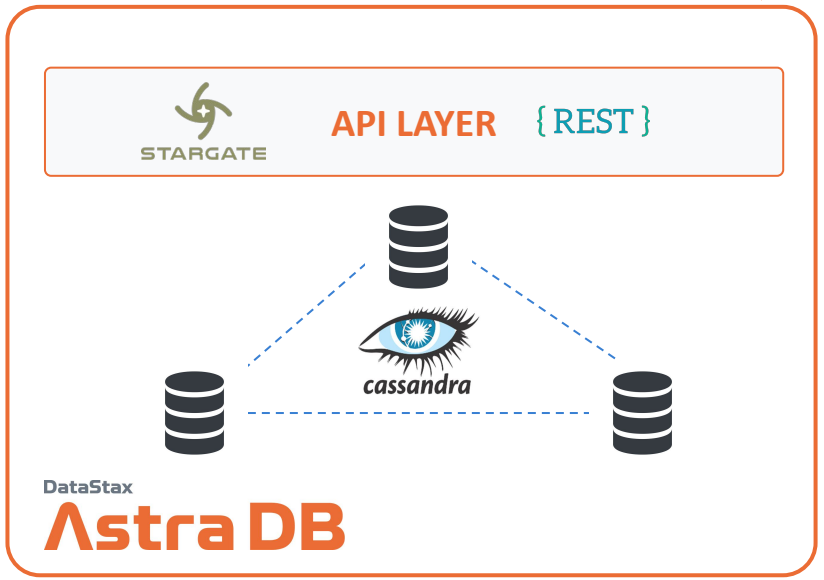
cassandra

DataStax Developers

Create Astra DB

Create Instance + token

DataStax Repository



Hands-on #1:

- Create Astra Instance
- Create security token
- Launch Gitpod

GitHub

https://github.com/datastaxdevs/workshop-intro-to-graphql

Agenda

01 DataStax Astra

DB Setup and Credentials

02

Introduction to GraphQL Design

03

GraphQL Tooling

GraphiQL

GraphQL Playground

04 DGS

Backend

Expose GQL Endpoint

05

Frontend

Consume GQL Endpoint

06

Resources

What is graphQL ?

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

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

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

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

https://graphql.org/learn/schema/#type-language

What is graphQL ?

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
}
```

https://graphql.org/learn/schema/#type-language

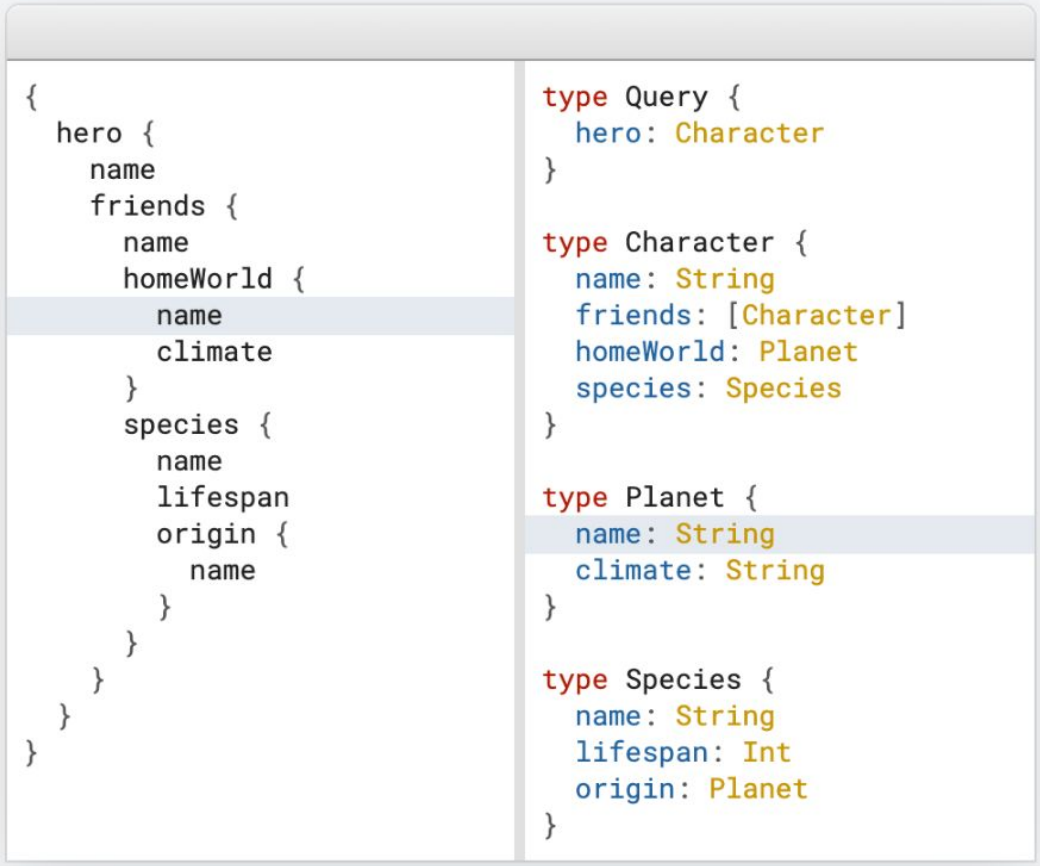
GraphQL Type System – Built in scalar type and UDT

- **Int:** An integer type, example 10
- **Float:** Floating point number, example 3.43
- **String:** A sequence of characters, example “Hello World”
- **Boolean:** A boolean example true
- **ID:** An object identifier

- **User Defined types**
  - Create your structure with constraints

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

type Speaker {
    name: String!
    email: String
}
```

Schema Definition



```
{
  hero {
    name
    friends {
      name
      homeWorld {
        name
        climate
      }
      species {
        name
        lifespan
        origin {
          name
        }
      }
    }
  }
}
```

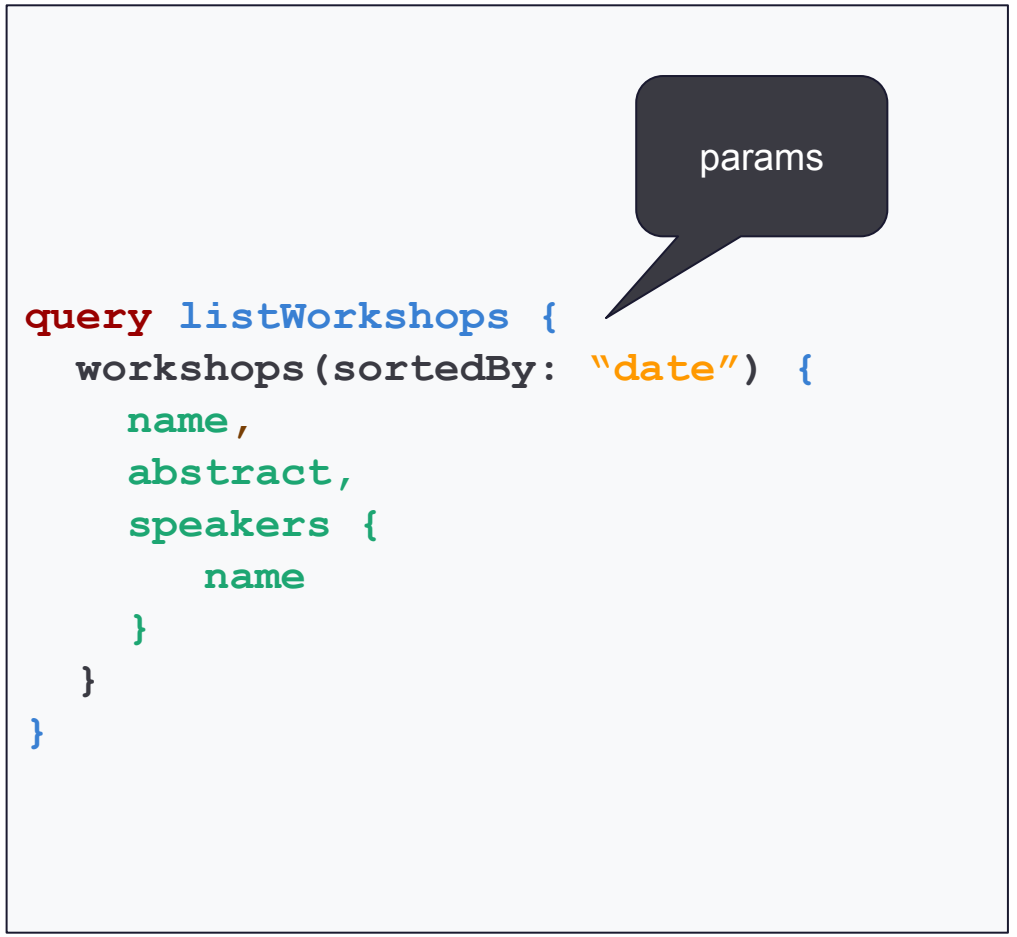
```
type Query {
  hero: Character
}

type Character {
  name: String
  friends: [Character]
  homeWorld: Planet
  species: Species
}

type Planet {
  name: String
  climate: String
}

type Species {
  name: String
  lifespan: Int
  origin: Planet
}
```

Describe what's possible with a type system

GraphQL APIs are organized in terms of types and fields, not endpoints. Access the full capabilities of your data from a single endpoint. GraphQL uses types to ensure Apps only ask for what's possible and provide clear and helpful errors. Apps can use types to avoid writing manual parsing code.

What is a GraphQL Query ?

- Used for **READ ONLY** requests.
- Used to **fetch data** from the server using the graphQL SDL syntax
- Describe what data the requester wishes to fetch from whoever is fulfilling the GraphQL Query



Declarative Query

graphql.org

```
{
  hero {
    name
    height
    mass
  }
}
```

```
{
  "hero": {
      "name": "Luke Skywalker",
      "height": 1.72,
      "mass": 77
  }
}
```

Ask for what you need, get exactly that

Send a GraphQL query to your API and get exactly what you need, nothing more and nothing less. GraphQL queries always return predictable results. Apps using GraphQL are fast and stable because they control the data they get, not the server.

What is a GraphQL Mutation ?

- Used to change resources data or execute actions on the server

- Client specifies the arguments and the action to be executed and then receives a response or a resource updated

```
mutation createWorkshop {
  createWorkshops({
    name: "Intro to GraphQL"
    abstract: "TODO"
    speakers: [
       {name: "Gilardi"},
       {name: "Hunter"}
    ]})
  {
    name
  }
}
```

GraphQL (personal) S.W.O.T Analysis

- Discoverability, introspection
- Declarative Data Fetching
- Schema stitching
- Flexible versioning
- Match standards (Json | Http)

- Single endpoint (*versioning, monitoring, security*)
- Complex implementation (*tooling, still young*)
- Nice for customers nasty for DB (*N+1 select*)
- Rate limiting is required
- Caching is hard (Apollo is a good solution here https://www.apollographql.com) <- this is what Netflix uses

- BFF : Backend for frontend
- Service aggregation | composition (*joins*)
- When bandwidth matters (*mobile phones*)

Agenda

01 DataStax Astra

DB Setup and Credentials

02

Introduction to GraphQL Design

03

GraphQL Tooling

GraphiQL

GraphQL Playground

04 DGS

Backend

Expose GQL Endpoint

05

Frontend

Consume GQL Endpoint

06

Resources

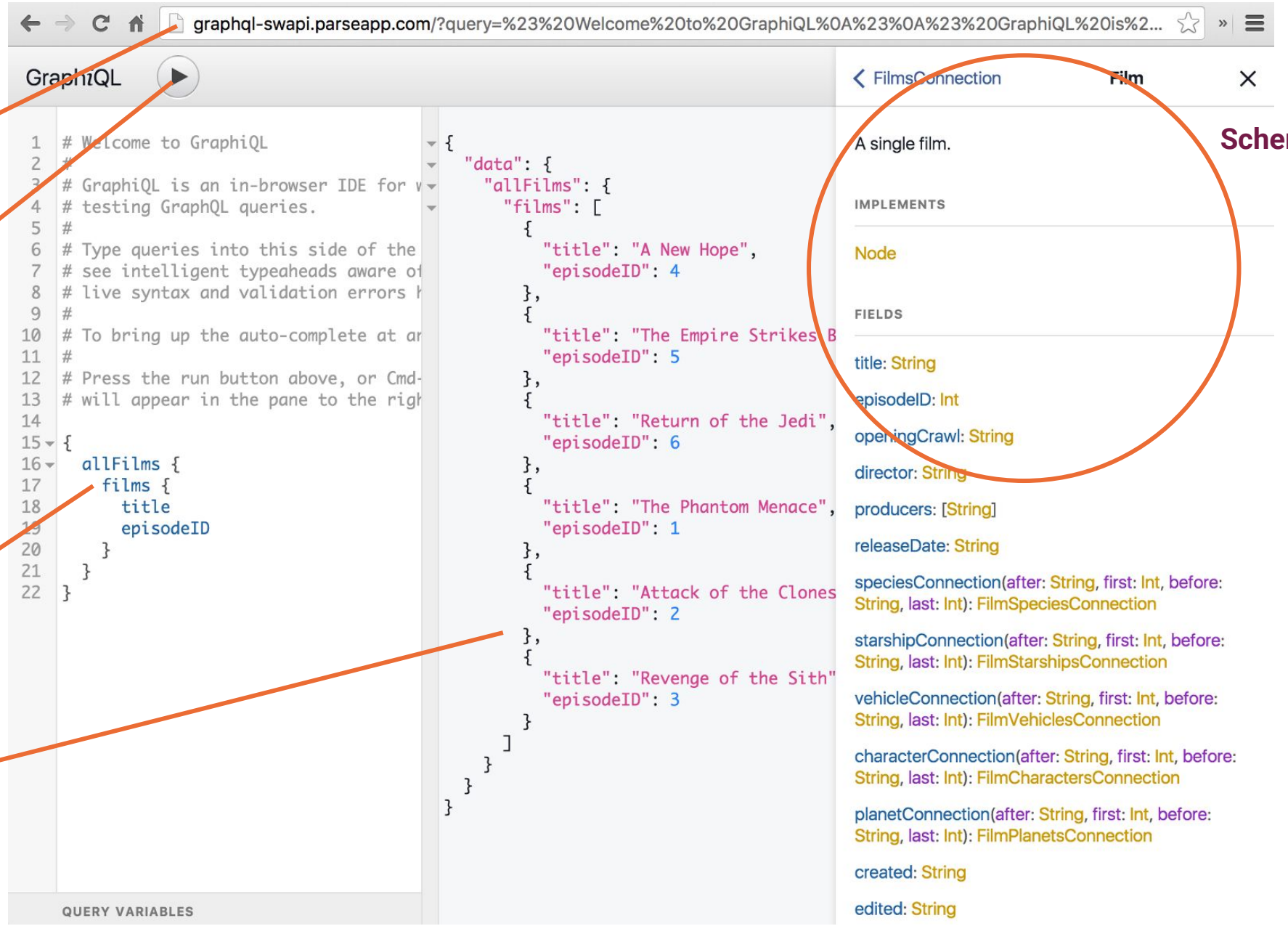
GraphiQL

ENDPOINT URL

EXECUTE

QUERY

RESULTS



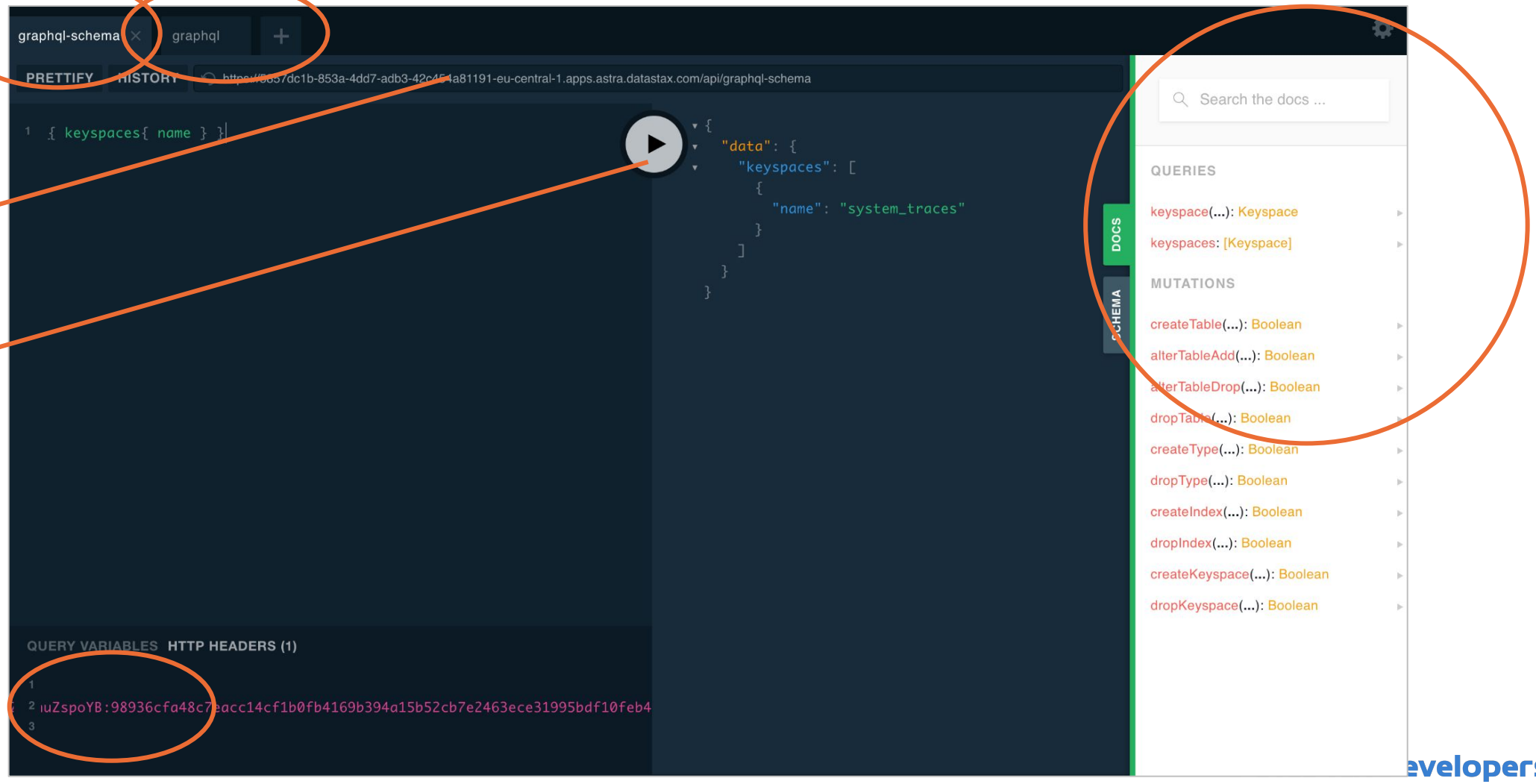
Schema

GraphQL Playground

DDL: DATA DEFINITION
Create Table
Create Schema

DML: DATA MANAGEMENT
INSERT DATA

Schema



ENDPOINT URL

EXECUTE

HEADER

evelopers

Agenda

01 DataStax Astra

DB Setup and Credentials

02

Introduction to GraphQL Design

03

GraphQL Tooling

GraphiQL

GraphQL Playground

04 DGS

Backend

Expose GQL Endpoint

05

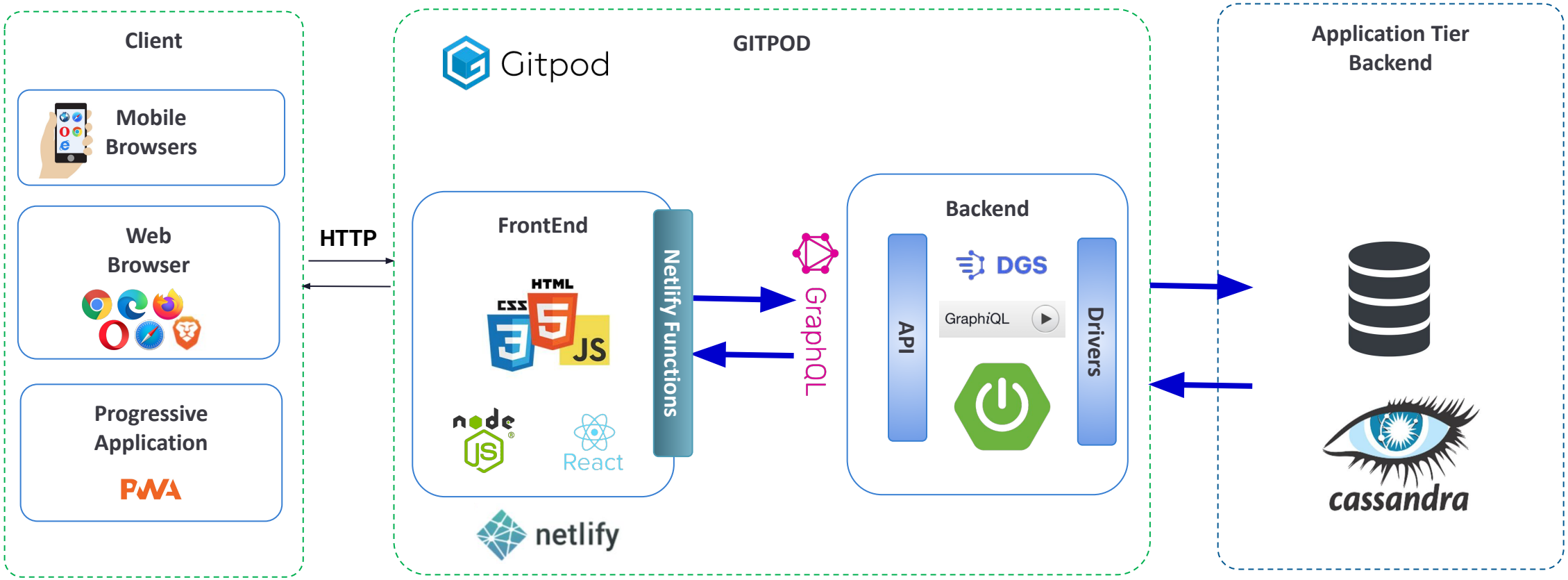
Frontend

Consume GQL Endpoint

06

Resources

Our Application Today



Exposing a GQL Endpoint with DGS

```java
public class Genre {
    private final String value;

    public Genre(String value) {
        this.value = value;
    }

    public String getValue() {
        return value;
    }
}
```

POJO Bean

DataFetcher as `@DgsComponent`

```java
@DgsComponent
public class GenresDatafetcher {
    Logger logger = Logger.getLogger(GenresDatafetcher.class.getName());

    private final List<Genre> genres = List.of(
            new Genre("Action"),
            new Genre("Anime"),
            new Genre("Award-Winning"),
            new Genre("Children & Family"),
            new Genre("Comedies")
    );

    @DgsQuery
    public List<Genre> genres(@InputArgument String labelFilter) {
        logger.info("QUERY executed - Genres value is: " + genres.get(0).getValue());

        if(labelFilter == null) {
            return genres;
        }

        return genres.stream().filter(s -> s.getValue().contains(labelFilter)).collect(Collectors.toList());
    }
```

Expose query with `@DgsQuery`

Hands-on #2:

- Experiment with GraphiQL and the Java back-end

GitHub

https://github.com/datastaxdevs/workshop-intro-to-graphql

DataStax Developers

Agenda

01 DataStax Astra

DB Setup and Credentials

02

Introduction to GraphQL Design

03

GraphiQL

GraphQL Playground

GraphQL Tooling

04 DGS

Backend

Expose GQL Endpoint

05

Frontend

Consume GQL Endpoint

06

Resources

Hands-on #3:

- Startup React client
- Digest graphQL code

GitHub

https://github.com/datastaxdevs/workshop-intro-to-graphql

DataStax Developers

Hands-on #4:

- Hook up the data layer with Astra DB
- Insert data in the Table with GraphQL
- Retrieving list of values

https://github.com/datastaxdevs/workshop-intro-to-graphql

Hands-on #5:

- Hook the database up to our React/JS app

https://github.com/datastaxdevs/workshop-intro-to-graphql

Agenda

01 DataStax Astra

DB Setup and Credentials

02

Introduction to GraphQL Design

03

GraphQL Tooling

GraphiQL

GraphQL Playground

04 DGS

Backend

Expose GQL Endpoint

05

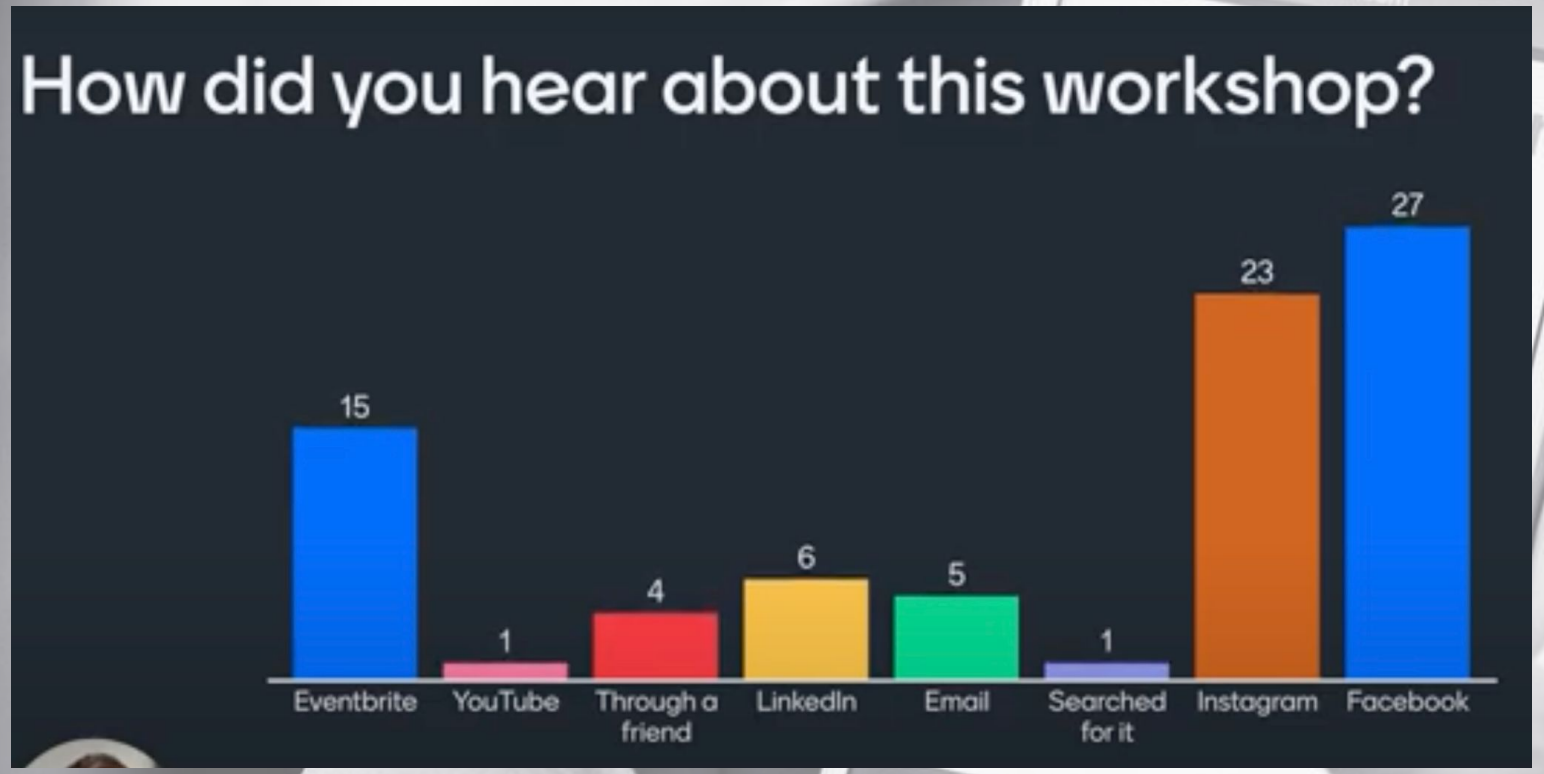
Frontend

Consume GQL Endpoint

06

Resources

menti.com First Questions !



Mentimeter

Swag Winners 

Congratulations to 1st, 2nd and 3rd place on the menti quiz!

To claim your prize, please send an email to:

jack.fryer@datastax.com

** Include a screenshot of your menti screen

Claim your FREE Voucher 🎉

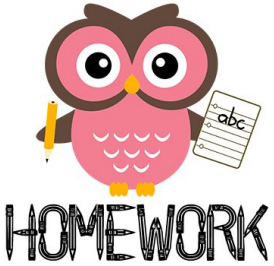
Vouchers (normally 145$ each)

- valid for 3 months,
- valid for 2 attempts

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

Claim using the link:

http://dtsx.io/workshop-voucher



Get your badge!
(and brag on Linkedin)

#1 - [Complete hands-on]

Run the React application in Gitpod connected to AstraDB,
insert (mutate) a new show/genre of your choice,
then *upload a screenshot*

All needed info in github repo

https://github.com/datastaxdevs/workshop-intro-to-graphql

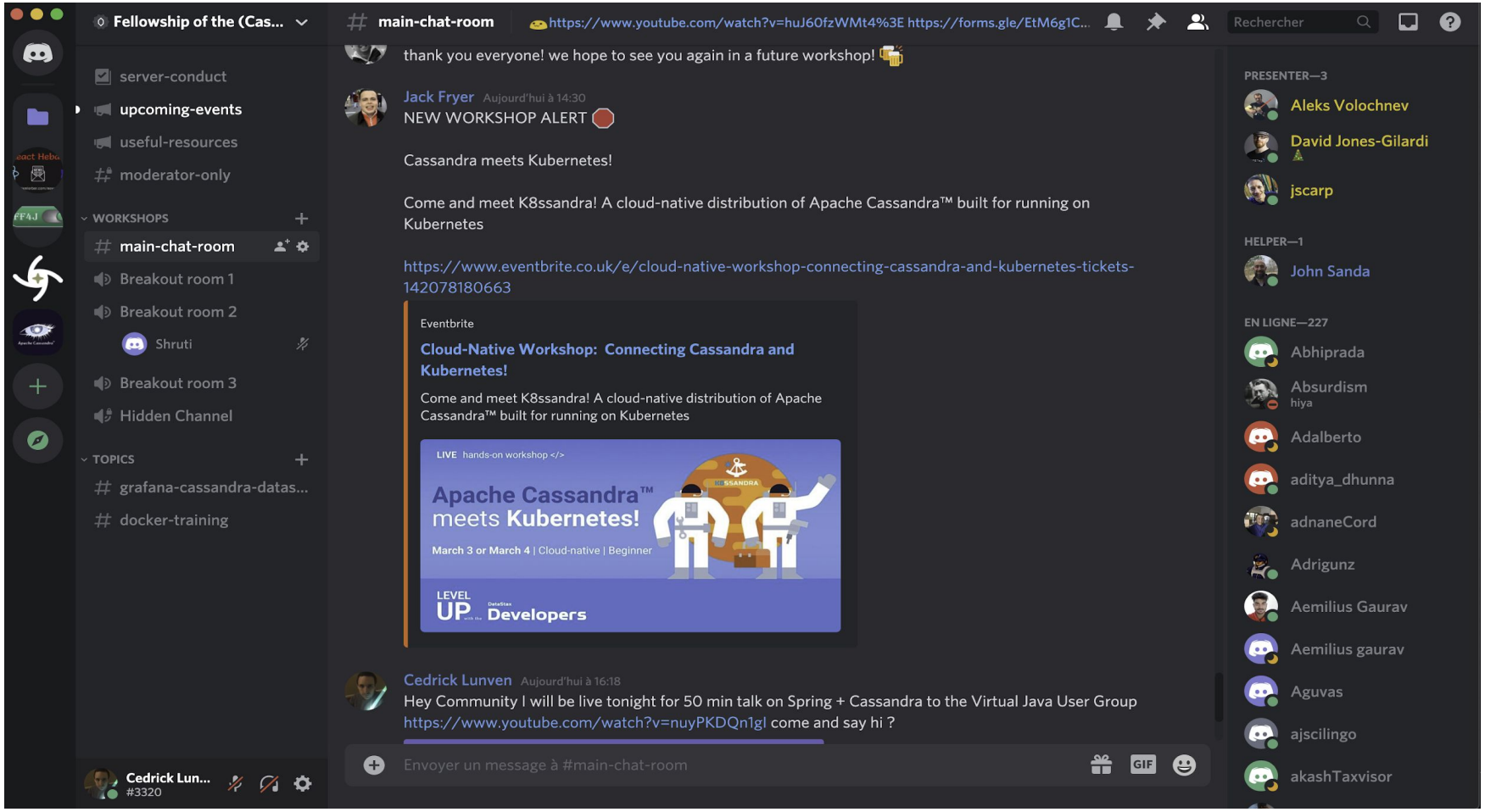
What's coming ?

datastax.com/workshops

9/16 **9/21** **9/22**

DataStax Developers

Join our 16k Discord Community
The Fellowship of the RINGS
dtsx.io/discord



DataStax Developers

Subscribe

Thank you!

Thank you!

Subscribe