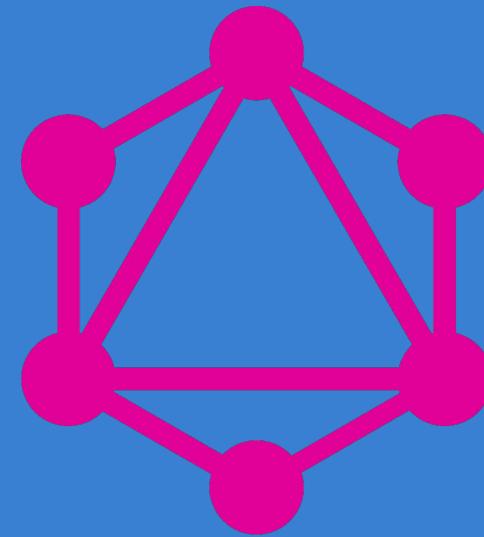
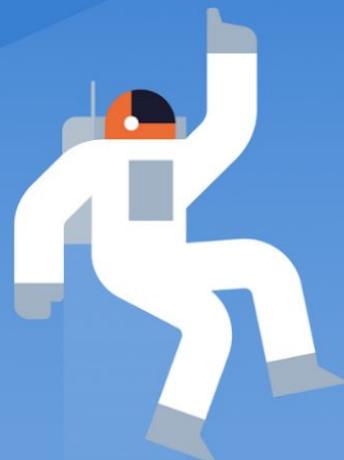


DataStax

# 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



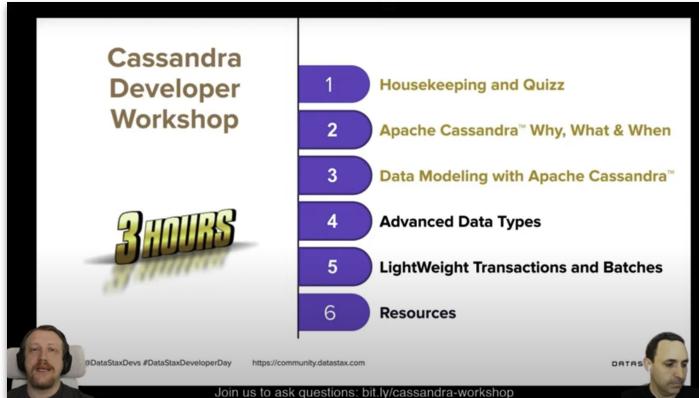
Rags  
Srinivas

DataStax Developers

# Housekeeping #1: Attending the workshop

Live and interactive

**Livestream:** [youtube.com/DataStaxDevs](https://youtube.com/DataStaxDevs)

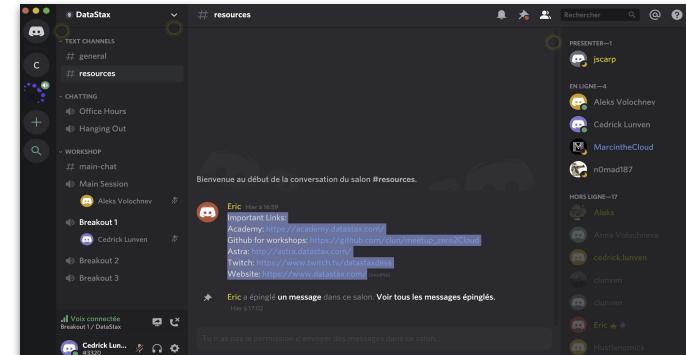


**YouTube**



**Twitch**

**Questions:** <https://dtsx.io/discord>



**Discord**

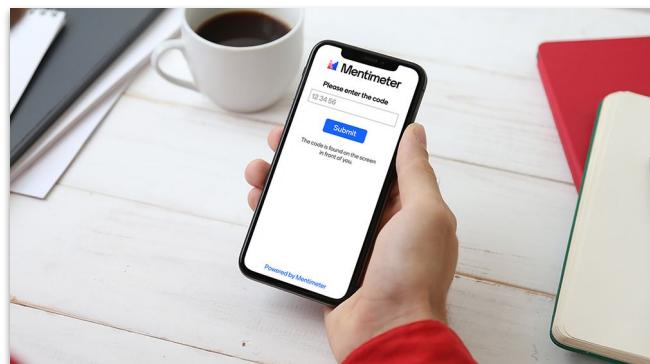


**YouTube**



Available on the iPhone  
App Store

**Games** [menti.com](https://menti.com)



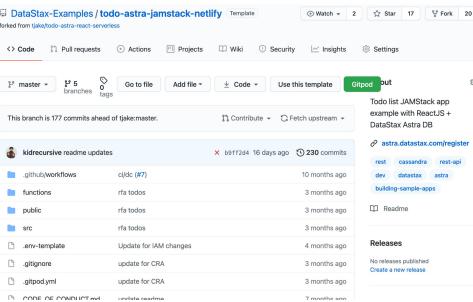
**Mentimeter**



Nothing to install !

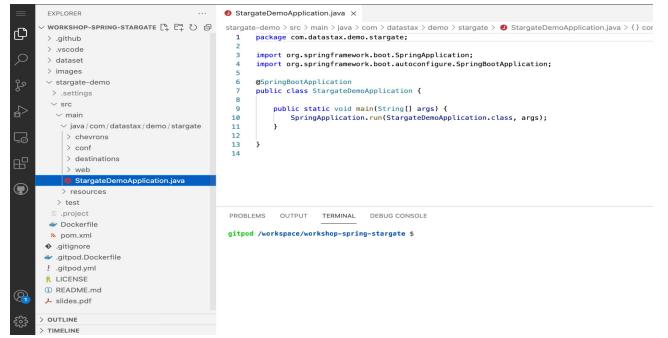
# Housekeeping #2: Doing Hands-On

Source code + exercises + slides



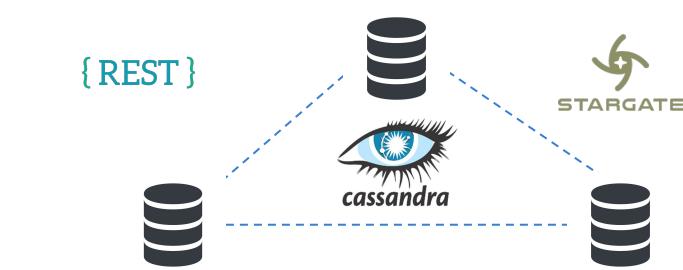
 GitHub

IDE



 Gitpod

Database + GraphQL + PlayGround



DataStax  
**Astra DB**

DataStax Developers

# A Badge when completing homeworks. No certificate of attendance.

New Badge !



## Introduction to GraphQL

Awarded to **test** • test12313213213213@test.ru  
Issued on **Sep 13, 2021**

Upgrade Complete! This badge is to certify successful completion of the DataStax Cassandra Workshop: "Introduction to GraphQL".

DataStax Developers

**EARNING CRITERIA**  
Recipients must complete the earning criteria to earn this Badge

To earn this badge, individuals must complete the following steps during the Introduction to GraphQL

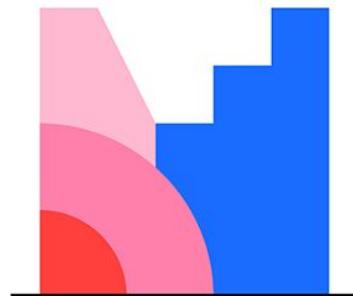
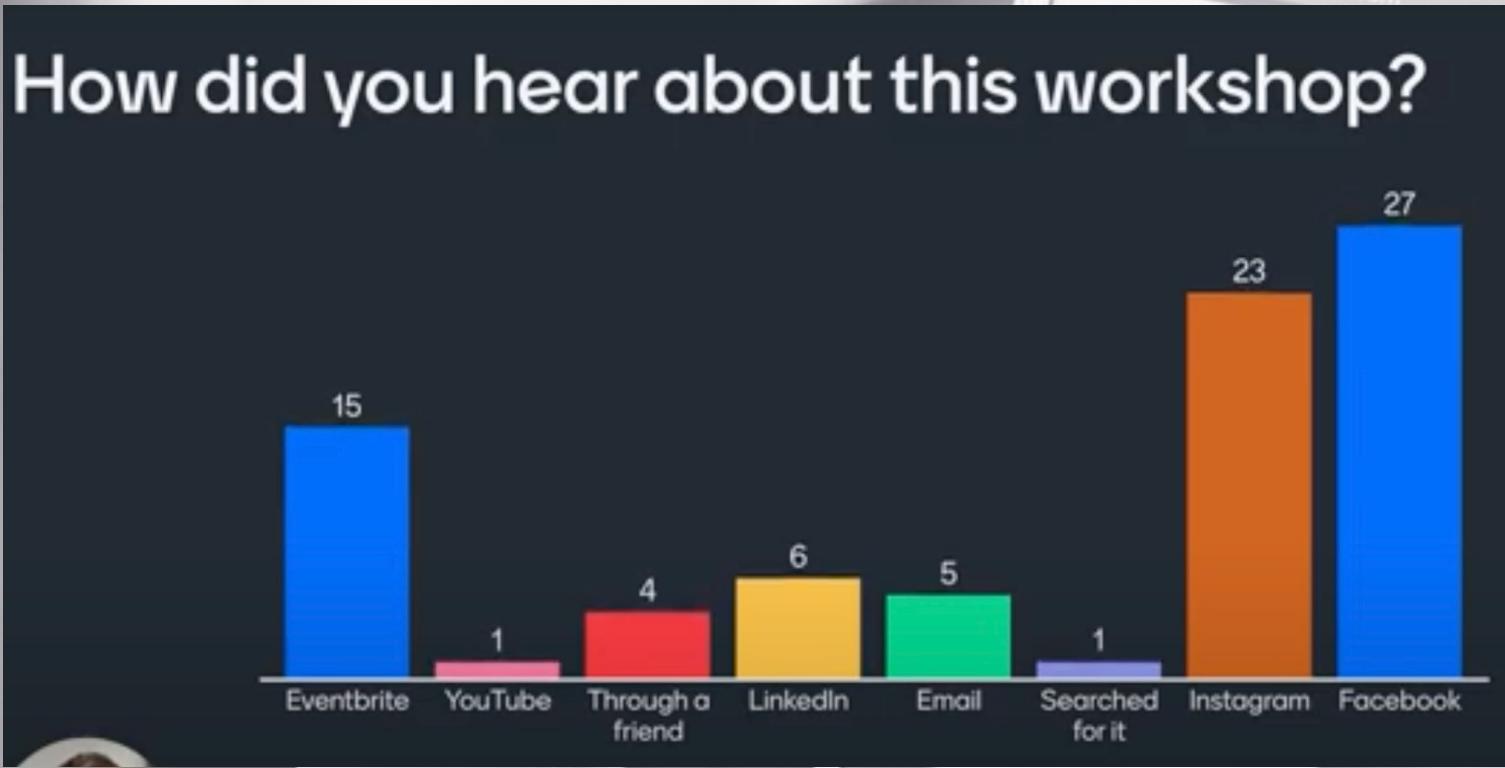
Attend the lecture Complete the practical steps by doing all required exercises

Offered By  
[DataStax Developers](#)

[Share](#)  
[Print Certificate](#)  
[Revoke](#)  
[View public page](#)



# menti.com



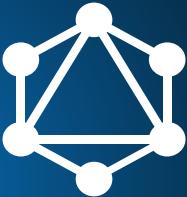
# Mentimeter

# Agenda

**01**   
DB Setup  
and Credentials



**02**  
Introduction to  
GraphQL Design



**03**  
GraphQL  
Tooling


**04**   
Backend  
Expose GQL Endpoint

**05**  
Frontend  
Consume GQL Endpoint



**06**  
Resources



# Demo:

- Let's see the apps

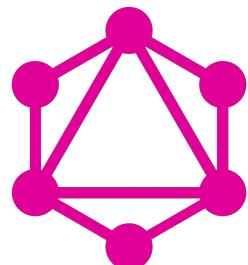


<https://github.com/datastaxdevs/workshop-intro-to-graphql>



# 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]  
}
```

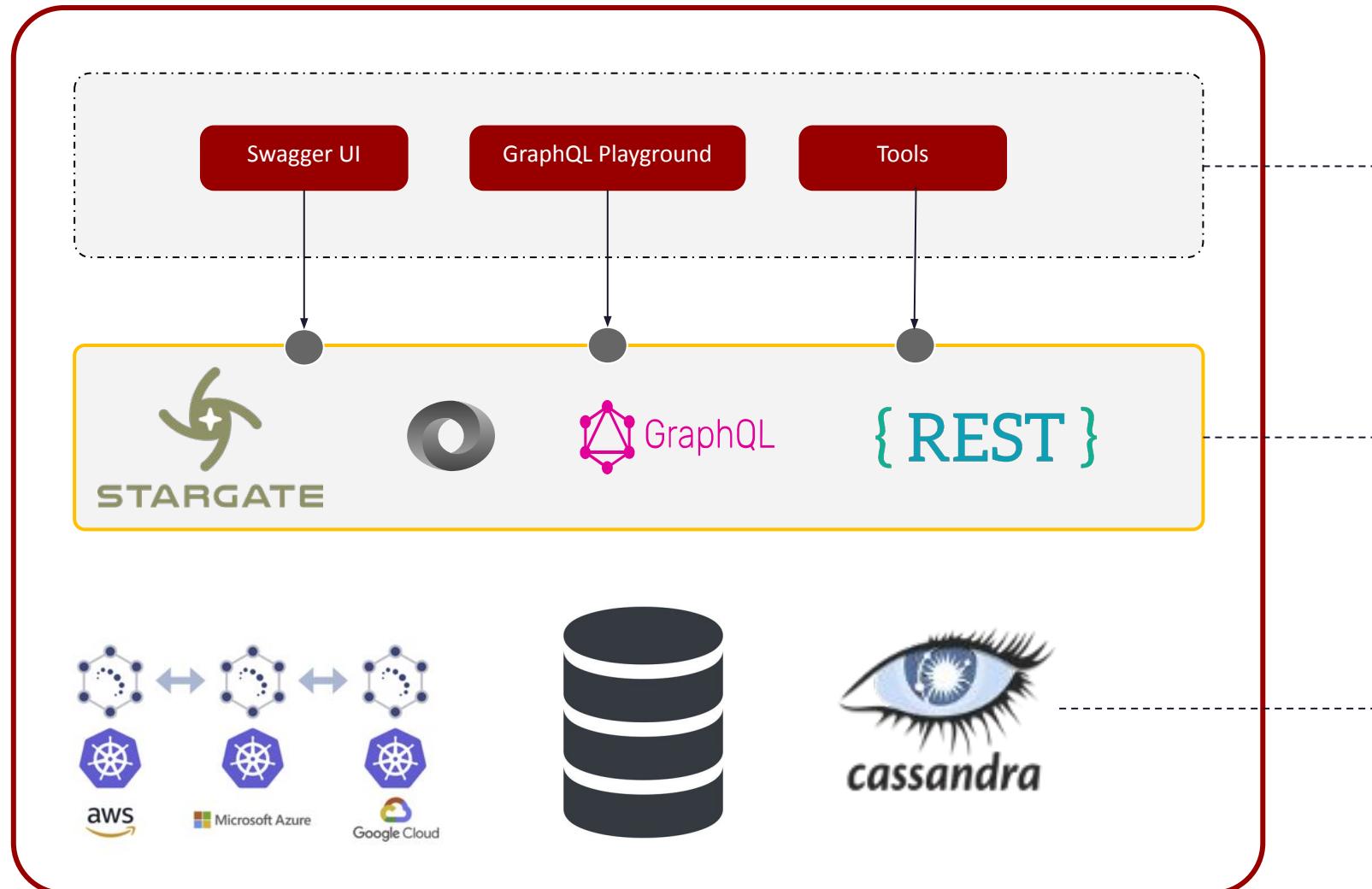
Ask for what you want

```
{  
    project(name: "GraphQL") {  
        tagline  
    }  
}
```

Get predictable results

```
{  
    "project": {  
        "tagline": "A query language for APIs"  
    }  
}
```

# Astra DB



**User Interface**  
Web Utils

OSS Stargate.io  
Gateway



Apache Cassandra

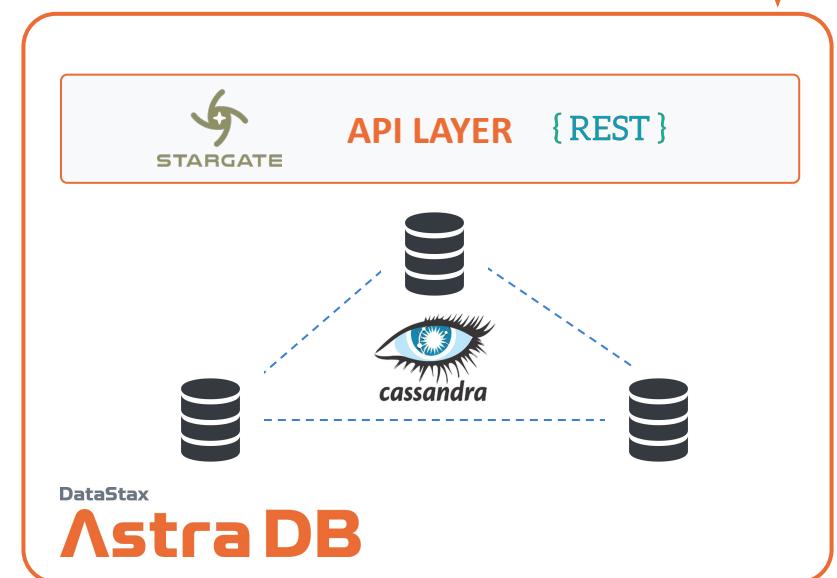


 Create Astra DB

Create Instance + token



## DataStax Repository



DataStax Developers

# Hands-on #1:

- Create Astra Instance
- Create security token
- Launch Gitpod

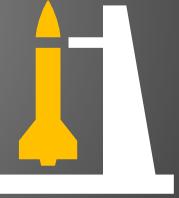


<https://github.com/datastaxdevs/workshop-intro-to-graphql>

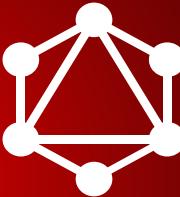


# Agenda

**01**   
DB Setup  
and Credentials



**02**  
Introduction to  
GraphQL Design



**03**  
GraphQL  
Tooling

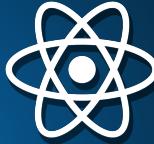


GraphQL 

GraphQL Playground

**04**   
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  
}
```

# What is graphQL ?

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

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

```
incomingWorkshops {  
    title  
}
```

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

# 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

```
query listWorkshops {  
  workshops(sortedBy: "date") {  
    name,  
    abstract,  
    speakers {  
      name  
    }  
  }  
}
```

params

# 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)



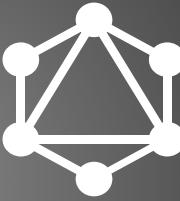
- ❖ Single endpoint (*versioning, monitoring, security*)
- ❖ Complex implementation (*tooling, still young*)
- ❖ Nice for customers nasty for DB (*N+1 select*)
- ❖ Rate limiting is required
- ❖ Caching is hard (Apollo is a good solution here  
<https://www.apollographql.com>) <- this is what Netflix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



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

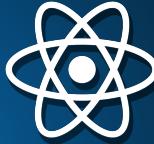
# Agenda

**01**  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

# GraphQL

ENDPOINT URL

EXECUTE

QUERY

RESULTS

The screenshot shows a GraphQL IDE interface with the following components:

- Left Panel (QUERY):** Contains the GraphQL query code:

```
1 # Welcome to GraphQL
2 #
3 # GraphQL is an in-browser IDE for v
4 # testing GraphQL queries.
5 #
6 # Type queries into this side of the
7 # see intelligent typeheads aware of
8 # live syntax and validation errors h
9 #
10 # To bring up the auto-complete at an
11 #
12 # Press the run button above, or Cmd-
13 # will appear in the pane to the right
14
15 {
16   allFilms {
17     films {
18       title
19       episodeID
20     }
21   }
22 }
```
- Middle Panel (RESULTS):** Displays the JSON results of the query:

```
{
  "data": {
    "allFilms": {
      "films": [
        {
          "title": "A New Hope",
          "episodeID": 4
        },
        {
          "title": "The Empire Strikes Back",
          "episodeID": 5
        },
        {
          "title": "Return of the Jedi",
          "episodeID": 6
        },
        {
          "title": "The Phantom Menace",
          "episodeID": 1
        },
        {
          "title": "Attack of the Clones",
          "episodeID": 2
        },
        {
          "title": "Revenge of the Sith"
          "episodeID": 3
        }
      ]
    }
  }
}
```
- Right Panel (Schema):** Shows the schema definition for the `Film` type, which implements the `Node` interface.

```
< FilmsConnection
Film X
A single film.

IMPLEMENTS
Node

FIELDS
title: String
episodeID: Int
openingCrawl: String
director: String
producers: [String]
releaseDate: String
speciesConnection(after: String, first: Int, before: String, last: Int): FilmSpeciesConnection
starshipConnection(after: String, first: Int, before: String, last: Int): FilmStarshipsConnection
vehicleConnection(after: String, first: Int, before: String, last: Int): FilmVehiclesConnection
characterConnection(after: String, first: Int, before: String, last: Int): FilmCharactersConnection
planetConnection(after: String, first: Int, before: String, last: Int): FilmPlanetsConnection
created: String
edited: String
```

# GraphQL Playground

## DDL: DATA DEFINITION

Create Table

Create Schema

The screenshot shows the GraphQL Playground interface with the title "graphql-schema". The main area displays a GraphQL query to list keyspaces:

```
1 { keyspaces{ name } }
```

Below the query is a large orange circle highlighting the "PRETTIFY" and "HISTORY" tabs. A play button icon is positioned above the results area, which contains the JSON response:

```
{ "data": { "keyspaces": [ { "name": "system_traces" } ] } }
```

At the bottom, there are sections for "QUERY VARIABLES" and "HTTP HEADERS (1)". The "HEADER" section is highlighted with an orange circle, showing the value: "1 uZspoYB:98936cfa48c7eacc14cf1b0fb4169b394a15b52cb7e2463ece31995bdf10feb4".

## DML: DATA MANAGEMENT

INSERT DATA

The screenshot shows the GraphQL Playground interface with the title "graphql". The main area displays a GraphQL mutation to create a keyspace:

```
1 mutation CreateKeyspace { 2   createKeyspace(name: "test") { 3     name 4   } 5 }
```

A large orange circle highlights the "SCHEMA" tab on the right sidebar. The sidebar also includes "DOCS" and "HISTORY" tabs. The "SCHEMA" tab shows the schema definitions:

- QUERIES
  - keyspace(...): Keyspace
  - keyspaces: [Keyspace]
- MUTATIONS
  - createTable(...): Boolean
  - alterTableAdd(...): Boolean
  - alterTableDrop(...): Boolean
  - dropTable(...): Boolean
  - createType(...): Boolean
  - dropType(...): Boolean
  - createIndex(...): Boolean
  - dropIndex(...): Boolean
  - createKeyspace(...): Boolean
  - dropKeyspace(...): Boolean

At the bottom right, the word "developers" is visible.

Schema

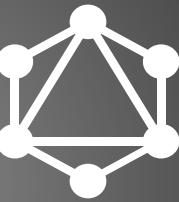
ENDPOINT URL

EXECUTE

HEADER

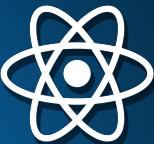
# Agenda

**01**  DB Setup and Credentials 

**02**  Introduction to GraphQL Design

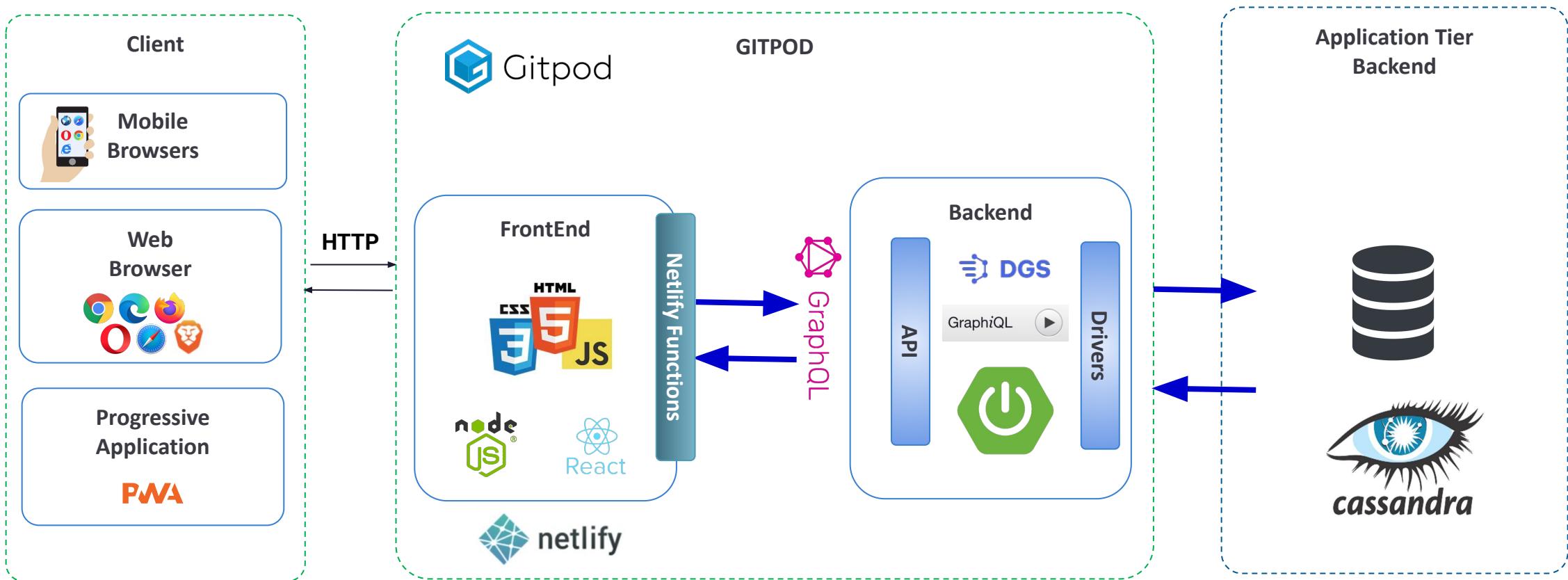
**03**  GraphQL Tooling 

**04**  DGS Backend Expose GQL Endpoint

**05**  Frontend Consume GQL Endpoint

**06**  Resources

# Our Application Today



# Exposing a GQL Endpoint with



```
public class Genre {  
    private final String value;  
  
    public Genre(String value) {  
        this.value = value;  
    }  
  
    public String getValue() {  
        return value;  
    }  
}
```

POJO Bean

```
@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")  
    );
```

DataFetcher as `@DgsComponent`

```
@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:

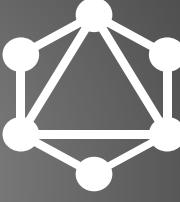
- Experiment with GraphiQL  
and the Java back-end



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

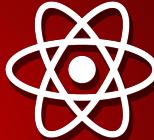
# Agenda

**01**  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 

# Hands-on #3:

- Startup React client
- Digest graphQL code



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



# 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

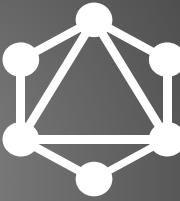


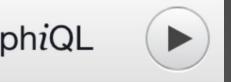
<https://github.com/datastaxdevs/workshop-intro-to-graphql>



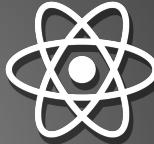
# Agenda

**01**  DB Setup and Credentials 

**02**  Introduction to GraphQL Design

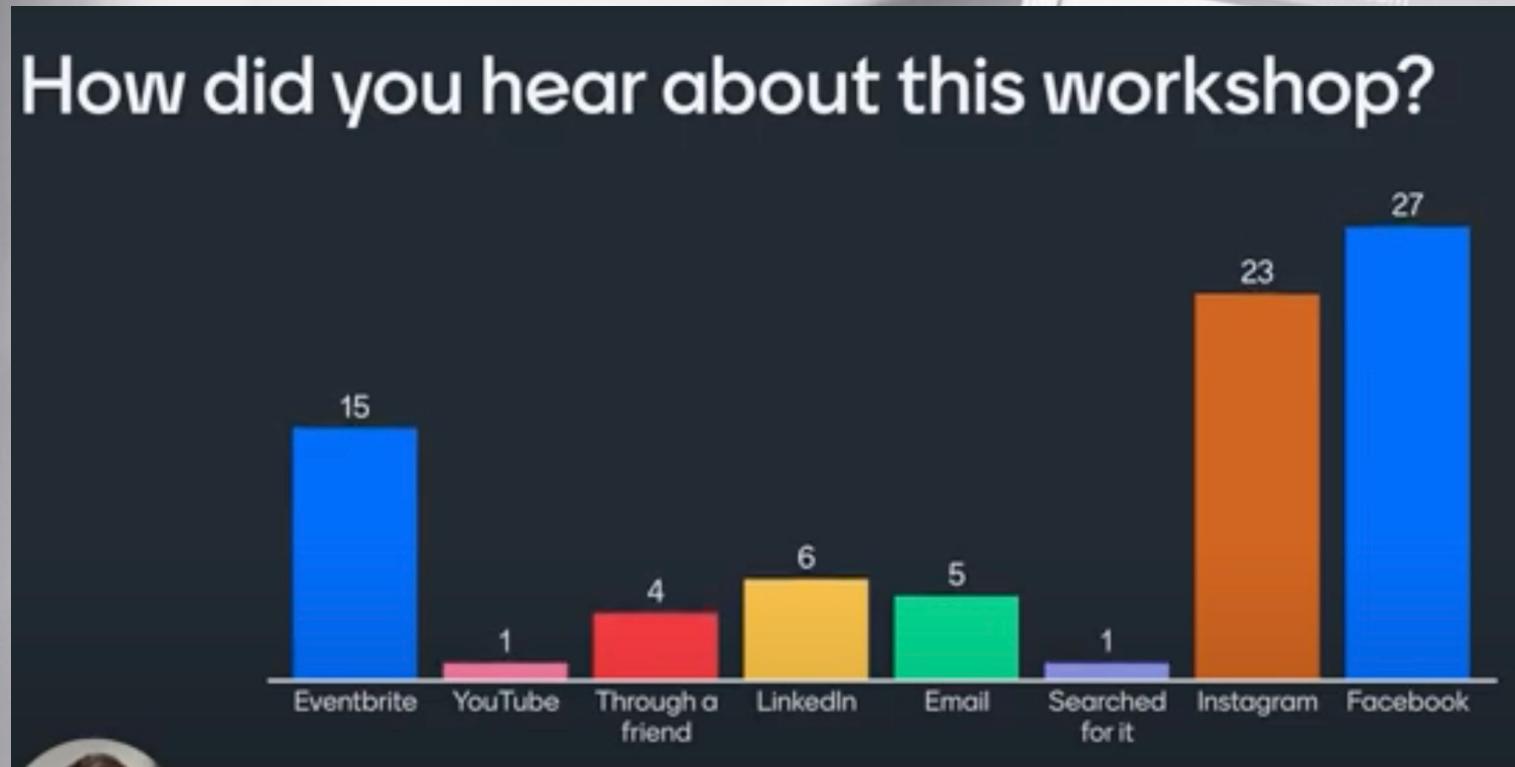
**03**  GraphQL Tooling 

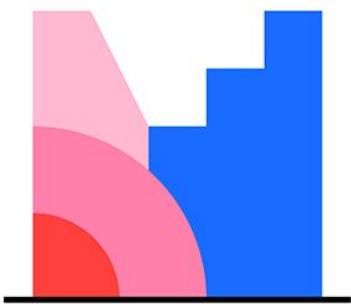
**04**  DGS Backend Expose GQL Endpoint

**05**  Frontend Consume GQL Endpoint

**06**  Resources

# 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](mailto: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>





HOMEWORK

## #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>

Get your badge!  
(and brag on LinkedIn)



# What's coming ?

[datastax.com/workshops](http://datastax.com/workshops)

 [Subscribe](#)



Part of  CityJS™

## Introduction to NoSQL Databases

**LEVEL UP** with the DataStax Developers

 **September 16**  
1pm BST

THU SEP 16 2021  
Workshop: Introduction to NoSQL Databases!

[Register Now](#)

With special guests  ANANT

## Part 1: Designing + Planning a Cloud Migration

**LEVEL UP** with the DataStax Developers

 **September 21**  
9am PT | 12pm ET | 6pm CEST | 9.30pm IST

TUE SEP 21 2021  
Workshop: Designing & Planning a Cloud Migration!

[Register Now](#)

LIVE hands-on workshop </>

## Build a Netflix Clone with GraphQL, React and a NoSQL DB

**LEVEL UP** with the DataStax Developers

 **September 22**  
9am PT | 12pm ET | 6pm CET | 9.30pm IST

WED SEP 22 2021  
#Frontend2021 - Build a Netflix clone with GraphQL, React and a NoSQL DB

[Register Now](#)

9/16

9/21

9/22

# Join our 16k Discord Community

## The Fellowship of the RINGS

[dtsx.io/discord](https://dtsx.io/discord)

# main-chat-room | <https://www.youtube.com/watch?v=huJ60fzWMt4%3E> <https://forms.gle/EtM6g1C...>

thank you everyone! we hope to see you again in a future workshop! 🍻

Jack Fryer Aujourd'hui à 14:30  
NEW WORKSHOP ALERT

Cassandra meets Kubernetes!

Come and meet K8ssandra! A cloud-native distribution of Apache Cassandra™ built for running on Kubernetes

<https://www.eventbrite.co.uk/e/cloud-native-workshop-connecting-cassandra-and-kubernetes-tickets-142078180663>

Eventbrite

**Cloud-Native Workshop: Connecting Cassandra and Kubernetes!**

Come and meet K8ssandra! A cloud-native distribution of Apache Cassandra™ built for running on Kubernetes

LIVE hands-on workshop </>

**Apache Cassandra™ meets Kubernetes!**

March 3 or March 4 | Cloud-native | Beginner

LEVEL UP DataStax Developers

Cedrick Lunven Aujourd'hui à 16:18  
Hey Community I will be live tonight for 50 min talk on Spring + Cassandra to the Virtual Java User Group  
<https://www.youtube.com/watch?v=nuyPKDQn1gl> come and say hi ?

Envoyer un message à #main-chat-room

PRESENTER—3

- Aleks Volochnev
- David Jones-Gilardi
- jscarp

HELPER—1

- John Sanda

EN LIGNE—227

- Abhiprada
- Absurdism
- hiya
- Adalberto
- aditya\_dhunna
- adnaneCord
- Adrigunz
- Aemilius Gaurav
- Aemilius gaurav
- Aguvas
- ajscilingo
- akashTaxvisor

# Thank you!

Subscribe



Subscribe



# Thank you!