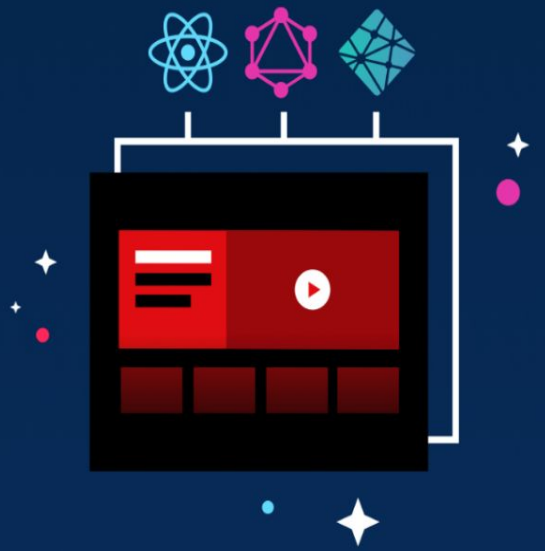
LIVE HANDS-ON WORKSHOP

Introduction to GraphQL

November 23, Wednesday
8am PT | 4pm GMT | 5pm CET | 8.30pm IST



DataStax

01

Housekeeping

Live and Hands-on

02

Database Setup

Data model & Astra DB

03

GraphQL

Design & Toolings

04

Backend

Expose GQL endpoint

05

Frontend

Consume GQL Endpoint

06

What's next?

Quiz, Homework, ...

Agenda

Housekeeping

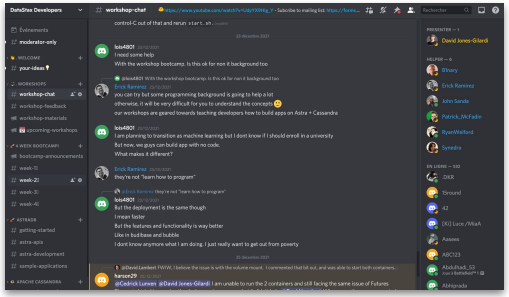
Live and Hands-on

Livestream: youtube.com/DataStaxDevs

YouTube

Twitch

Questions: https://dtsx.io/discord

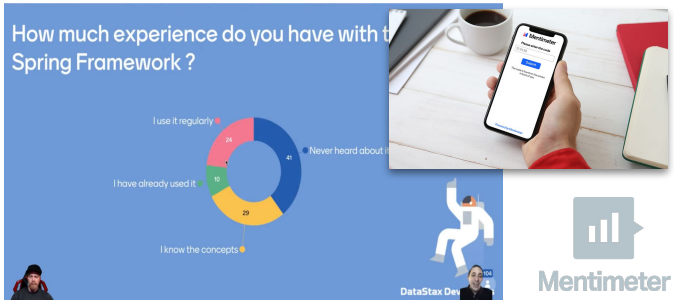


YouTube

Discord

Live and interactive

Games and quizzes: menti.com

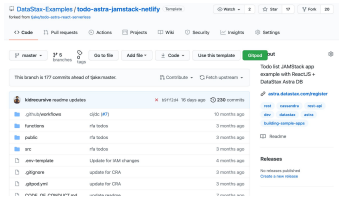


Mentimeter

Live Sessions

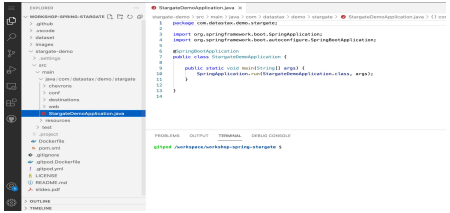
Nothing to install !

Source code, exercises, slides



GitHub

IDE

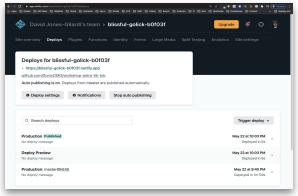


Gitpod

Database + GraphQL + PlayGround

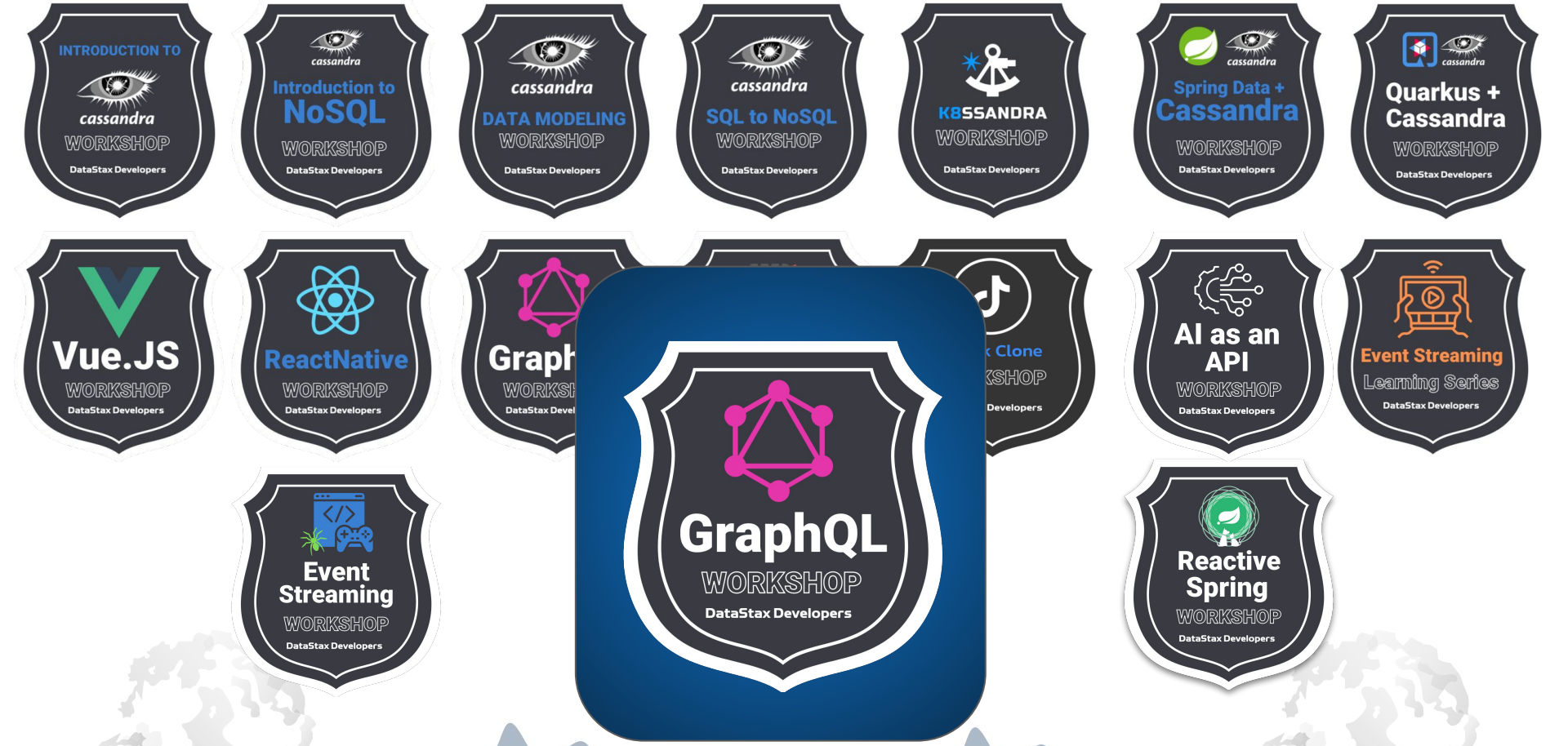
DataStax
Astra DB

Deploy + Run + Host



netlify

Hands-On Housekeeping



Get your badge for today

DataStax Developers

01

Housekeeping

Live and Hands-on

02

Database Setup

Data model & Astra DB

03

GraphQL

Design & Toolings

04

Backend

Expose GQL endpoint

05

Frontend

Consume GQL Endpoint

06

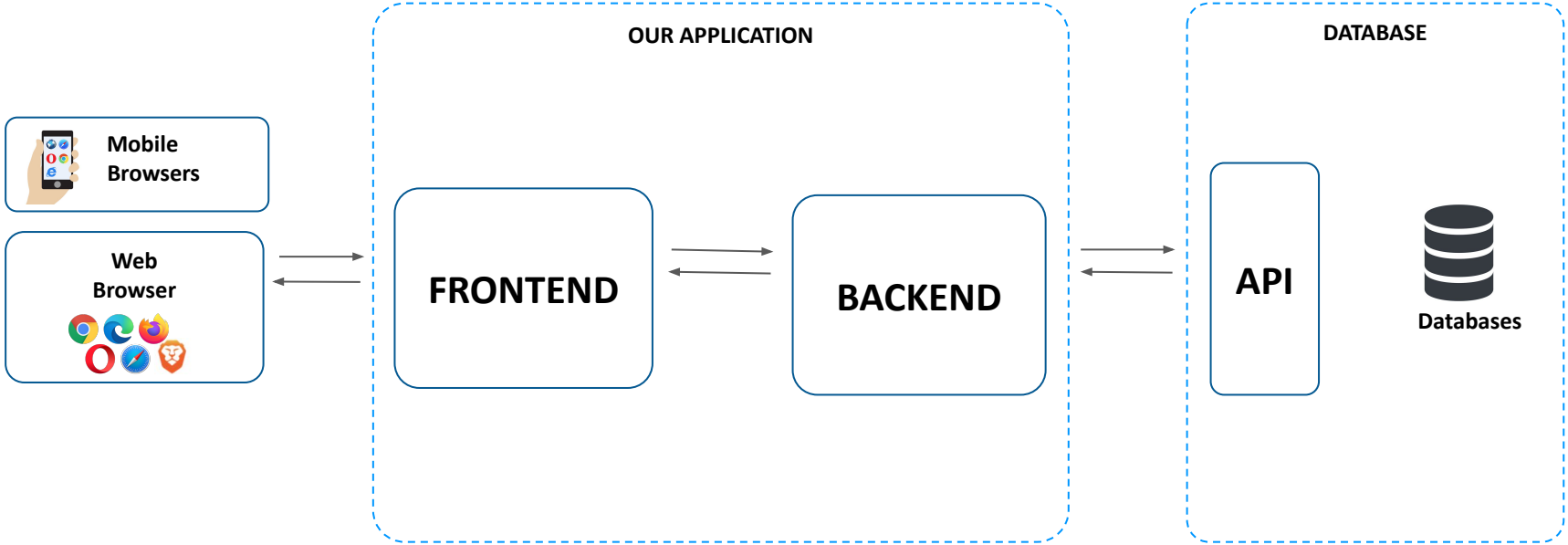
What's next?

Quiz, Homework, ...

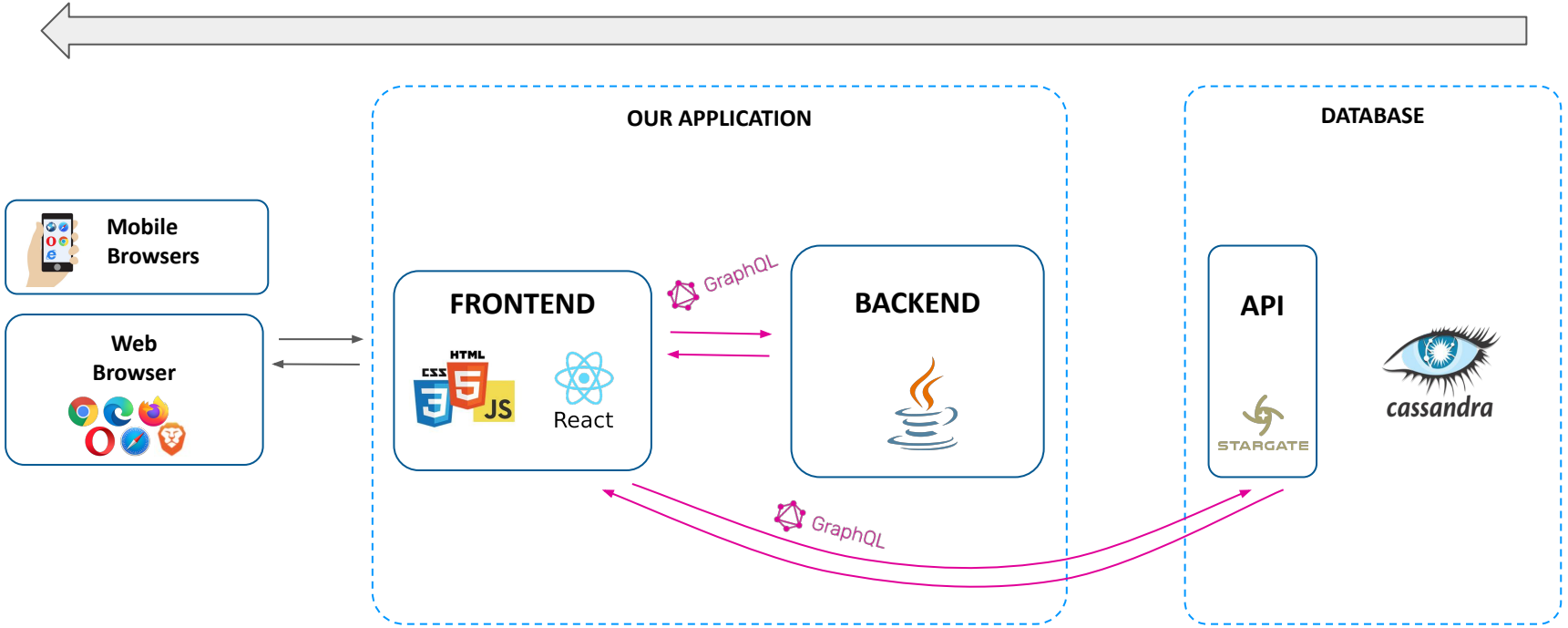
Agenda

Our Goal

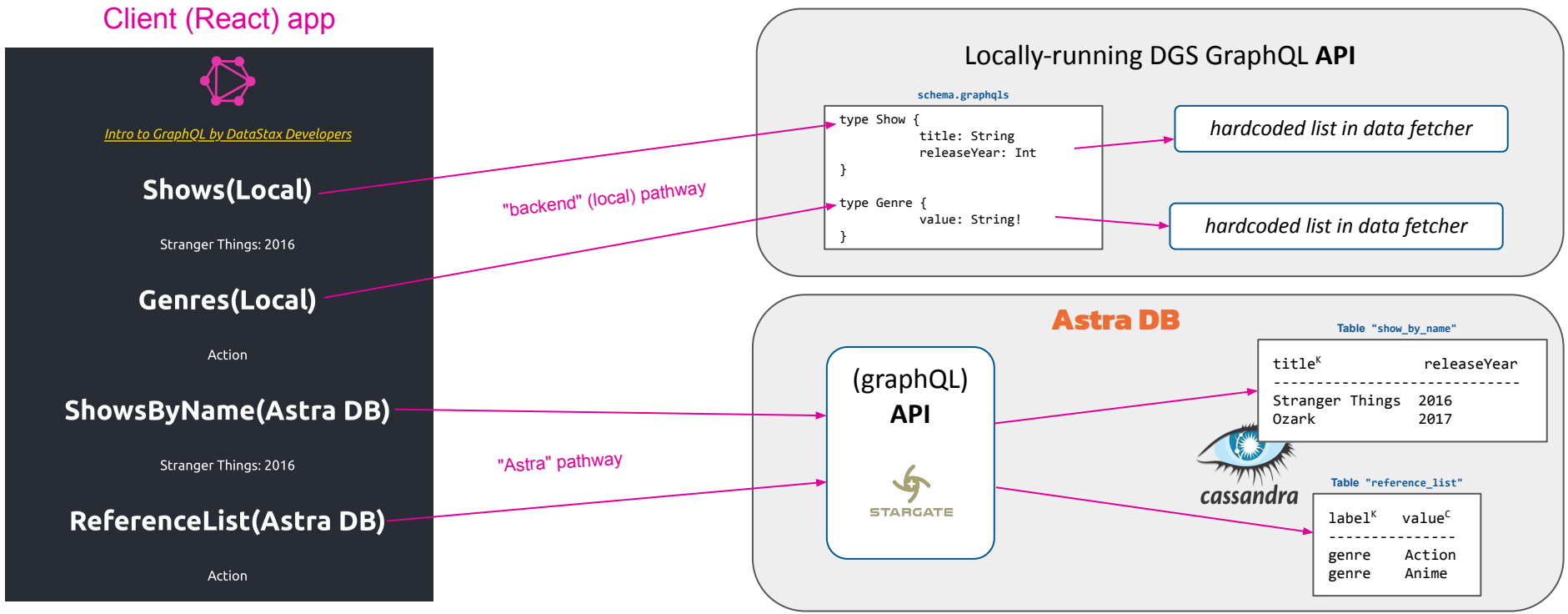
Overall Architecture



Building a Full-stack application



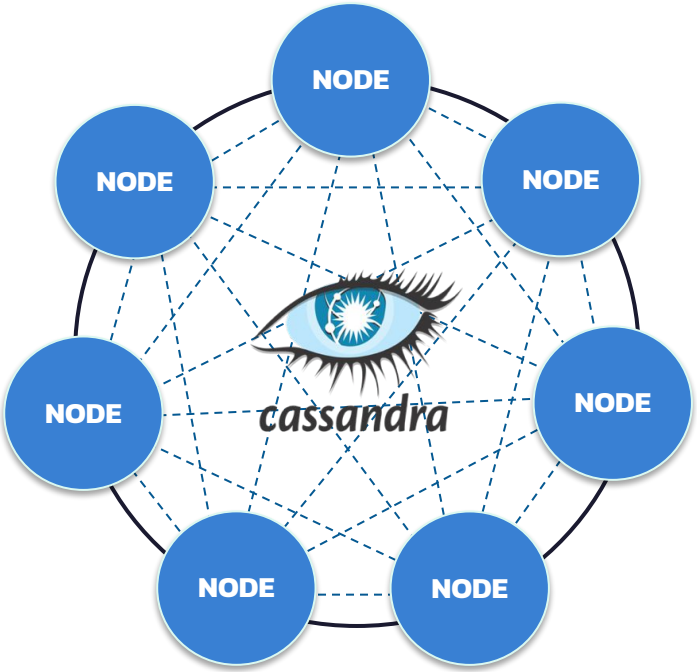
Building a Full-stack application



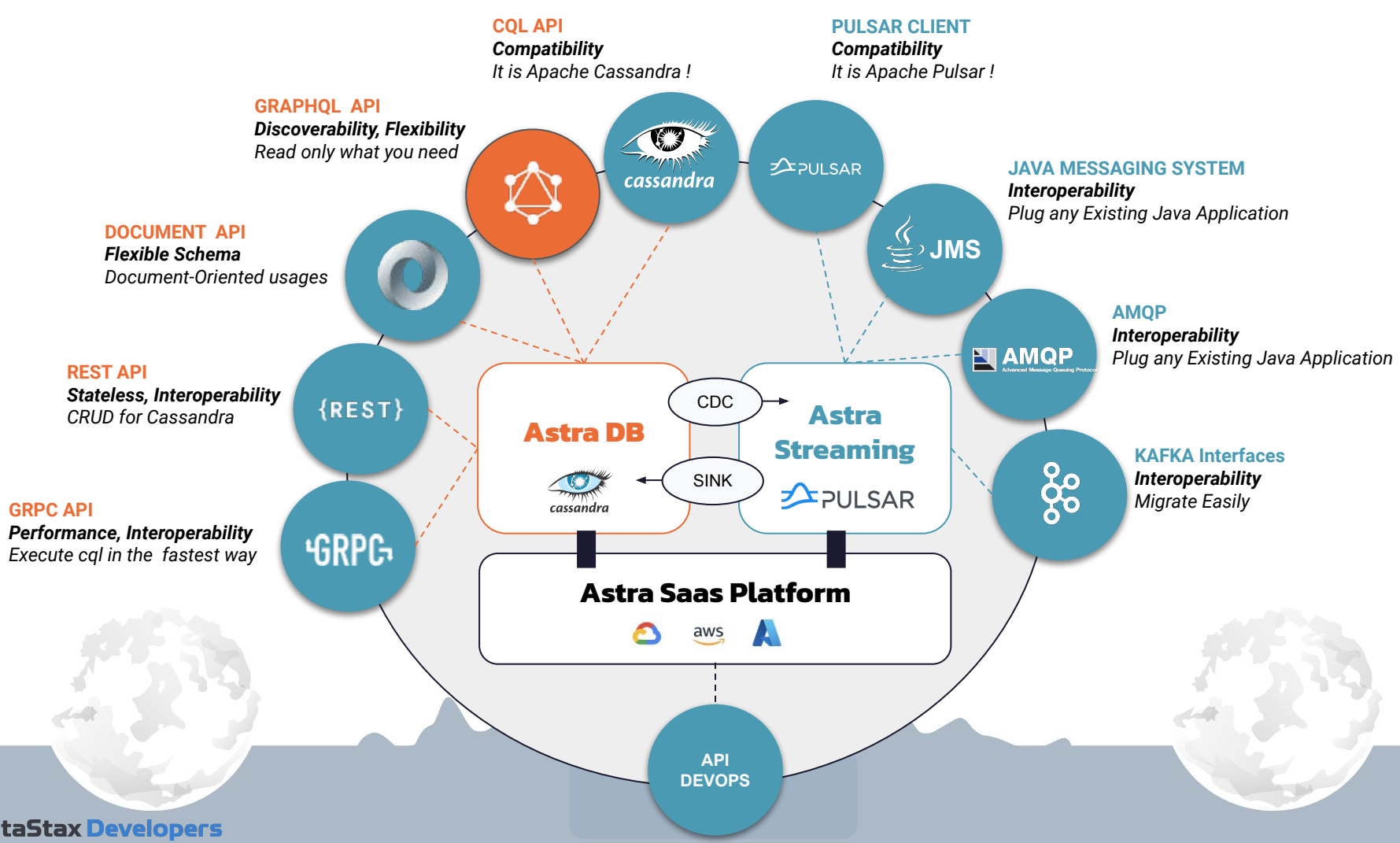
Today's target app

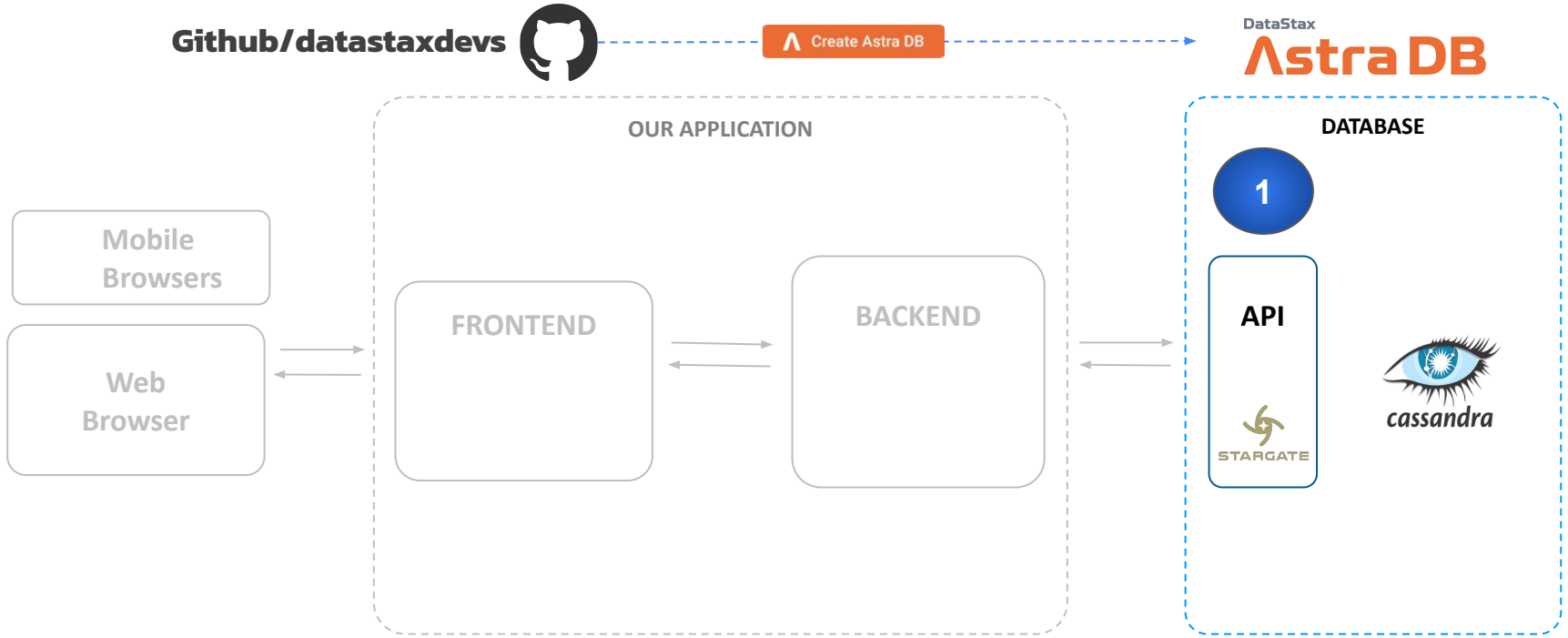
Database setup

Data model & Astra DB

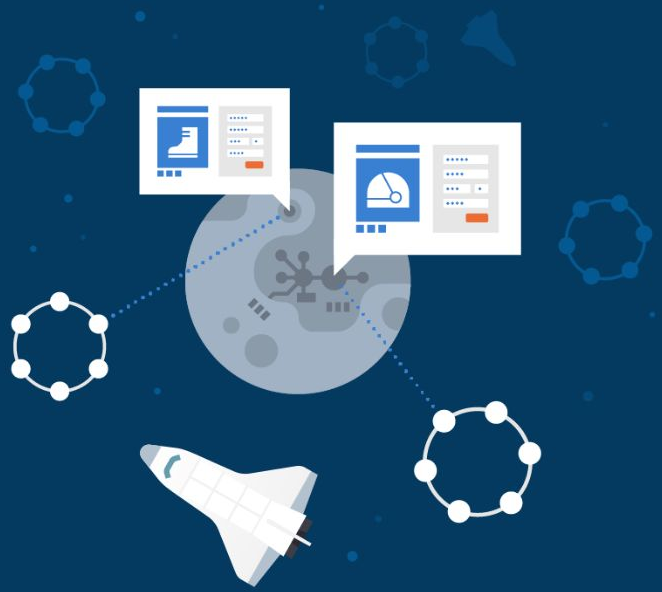


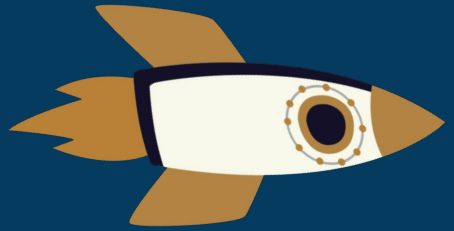
- Big Data Ready
- Read / Write Performance
- Linear Scalability
- Highest Availability
- Self-Healing and Automation
- Geographical Distribution
- Platform Agnostic
- Vendor Independent


CQL API
Compatibility
It is Apache Cassandra !
PULSAR CLIENT
Compatibility
It is Apache Pulsar !
GRAPHQL API
Discoverability, Flexibility
Read only what you need
JAVA MESSAGING SYSTEM
Interoperability
Plug any Existing Java Application
DOCUMENT API
Flexible Schema
Document-Oriented usages
AMQP
Interoperability
Plug any Existing Java Application
REST API
Stateless, Interoperability
CRUD for Cassandra
KAFKA Interfaces
Interoperability
Migrate Easily
GRPC API
Performance, Interoperability
Execute cql in the fastest way
cassandra
PULSAR
JMS
AMQP
{REST}
GRPC
Astra DB
cassandra
CDC
SINK
Astra Streaming
PULSAR
Astra Saas Platform
aws
API
DEVOPS



Setup Database





Hands-on (!github)

#1 DB SETUP

✅ Register & Create your Database

✅ Create a security token if needed

01

Housekeeping

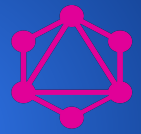
Live and Hands-on

02

Database Setup

Data model & Astra DB

03



GraphQL

Design & Toolings

04

Backend

Expose GQL endpoint

05

Frontend

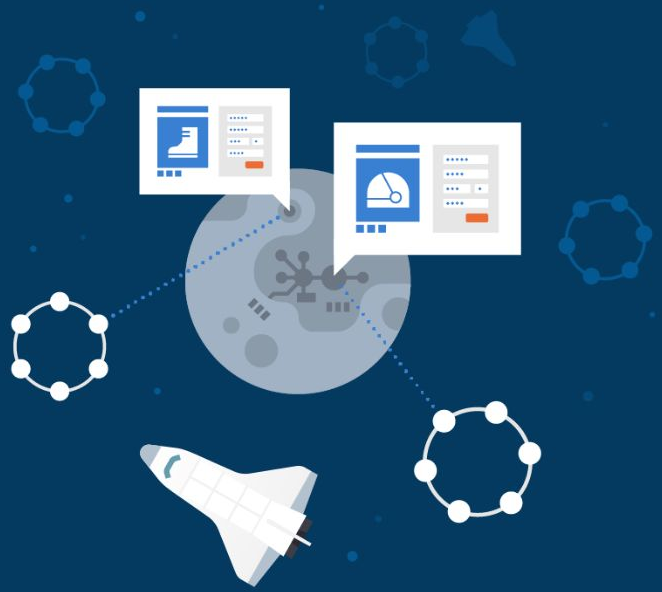
Consume GQL Endpoint

06

What's next?

Quiz, Homework, ...

Agenda



GraphQL

GraphQL

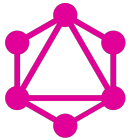
Learn & Design

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.

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

query sample

```
query getShows {
    shows {
        title
        releaseYear
    }
}
```

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

schema snippet

```
type Query {
     shows: [Show]
}

type Show {
    title: String!
    blurb: String
    actors: [Actor]
    releaseYear: Int
}
```

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

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

query sample

```
query getShows {
    shows {
        title
    }
}
```

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

query sample

```
query getShows {
    shows {
        title
        blurb
        actors: {
            name
        }
        releaseYear
    }
}
```

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

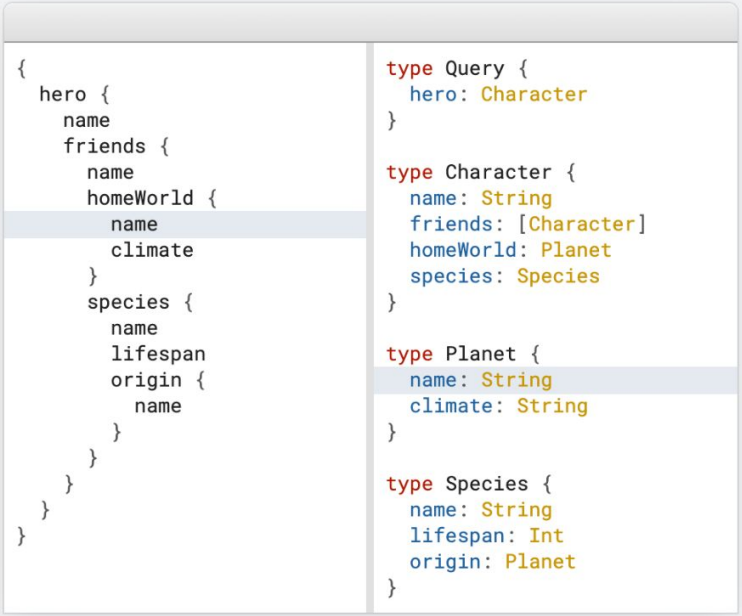
- **Int:** An integer type (example: 10)
- **Float:** Floating point number, example 3.43
- **String:** A sequence of characters (“Hello World”)
- **Boolean:** A boolean example true
- **ID:** An object identifier

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

schema snippet

```
type Show {
    title: String!
    blurb: String
    actors: [Actor]
    releaseYear: Int
}

type Actor {
    name: String!
    age: Int
}
```



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.

https://graphql.org

- 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
- **All runnable queries must appear in schema!**

query sample

```
query getSortedShows {
    shows(titleFilter: "Ozark") {
        title
        blurb
        releaseYear
    }
}
```

optional params

schema snippet

```
type Query {
     shows(titleFilter: String): [Show]
}

type Show {
    ...
}
```

What is a GraphQL Query ?

```
{
  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.

https://graphql.org

Declarative Query

- 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
- All runnable mutations must appear in schema!

query sample

```
mutation insertMyFavoriteShow {
  createShow({
    title: "Anna Karenina"
    blurb: "Anna; train; splat!"
    actors: [
        {
            name: "Greta Garbo"
            age: 30
        },
        {
            name: "Fredric March"
            age: 38
        }
    ]
    releaseYear: 111
  })
  {
    title
  }
}
```

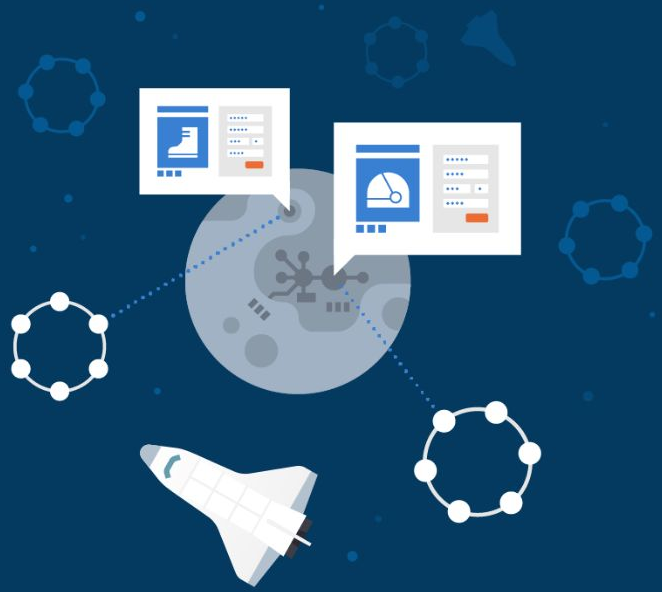
What is a GraphQL Mutation ?

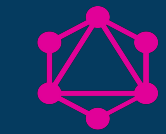
- Discoverability, introspection
- Declarative Data Fetching
- Schema stitching
- Customize responses
- Flexible versioning
- Match standards (Json | Http)

- 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, "federation" (*joins*)
- When bandwidth matters (*mobile phones*)

GraphQL S.W.O.T. analysis





GraphQL

Tooling

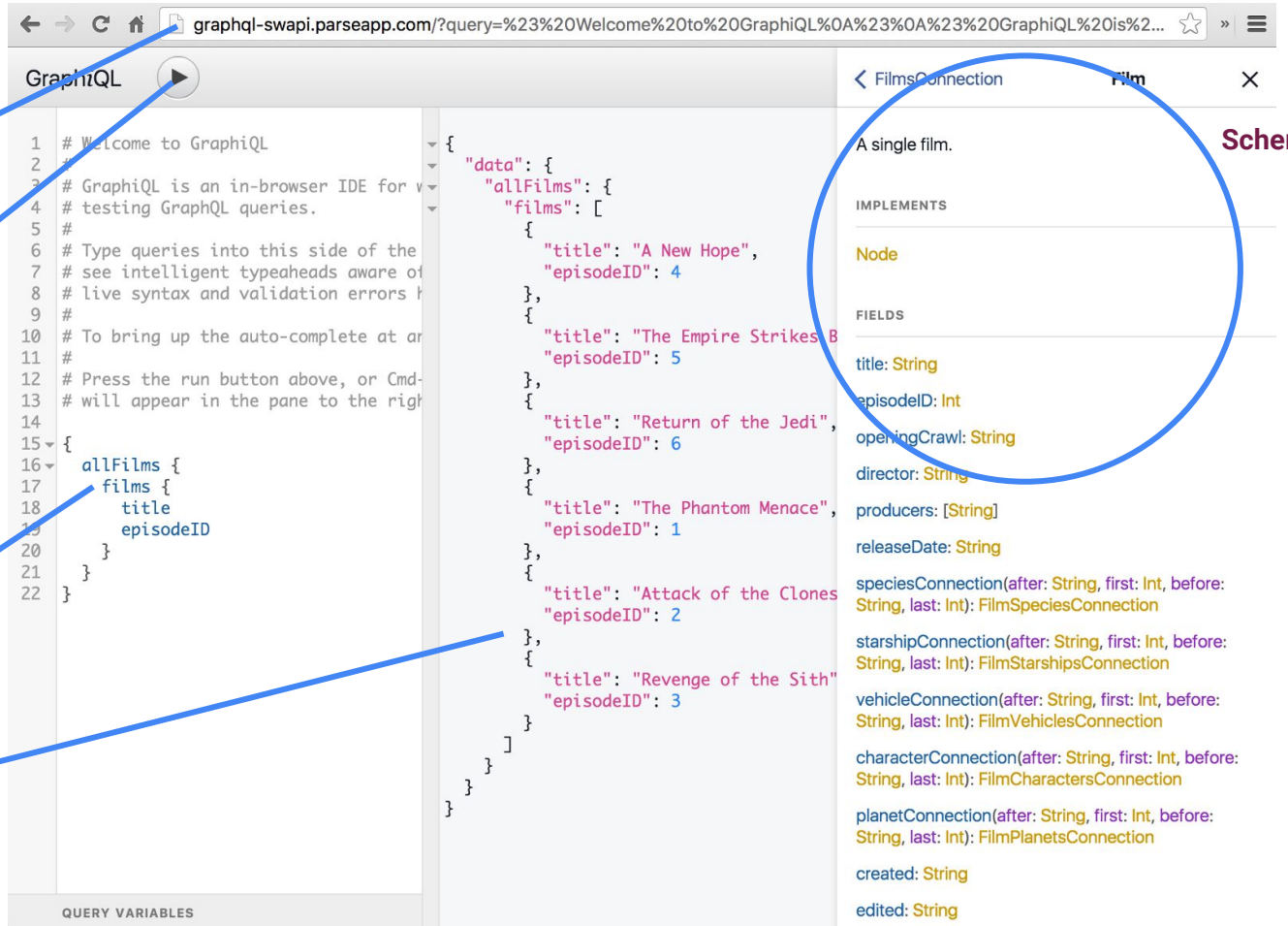
GraphiQL

ENDPOINT URL

EXECUTE

QUERY

RESULTS



Schema

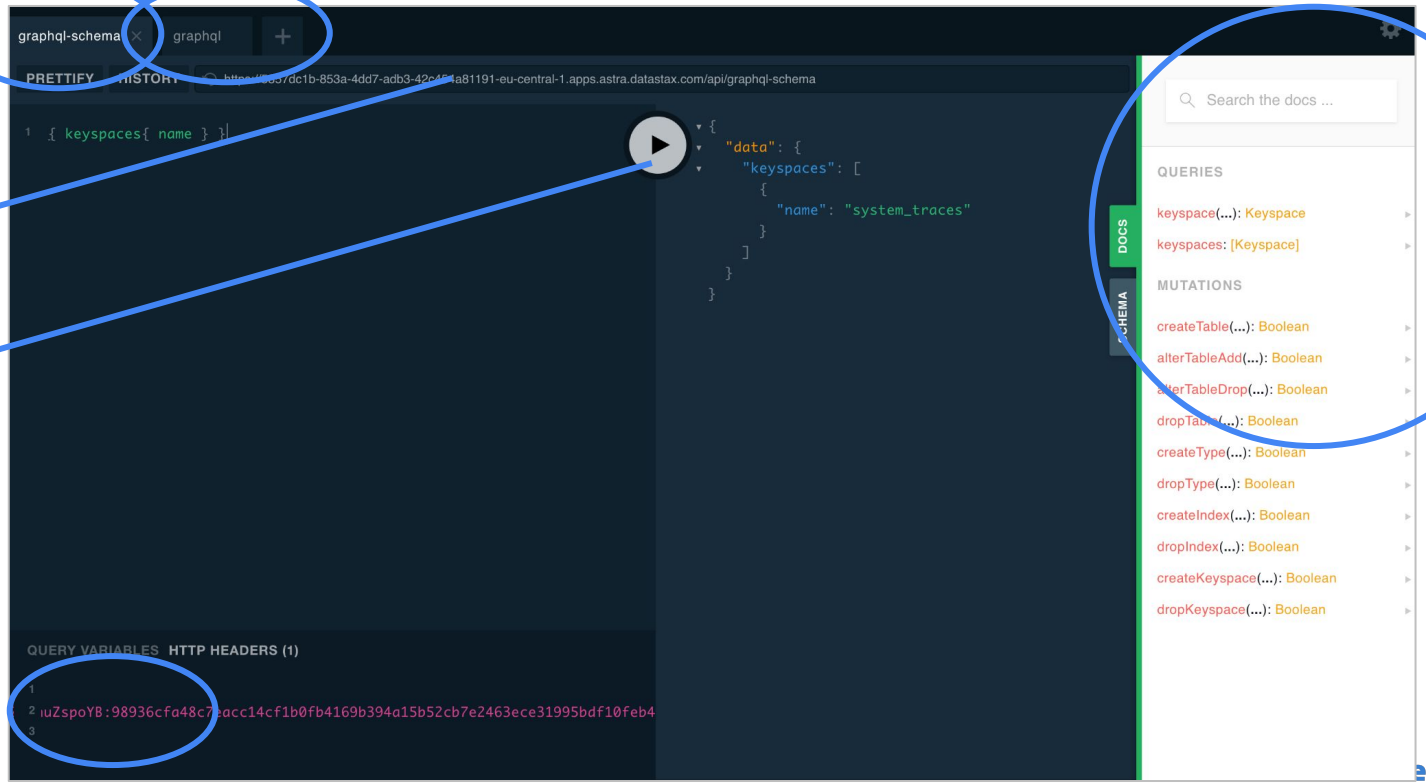
DataStax Developers

GraphQL Playground

DDL: DATA DEFINITION
Create Table
Create Schema

DML: DATA MANAGEMENT
INSERT/RETRIEVE DATA

Schema

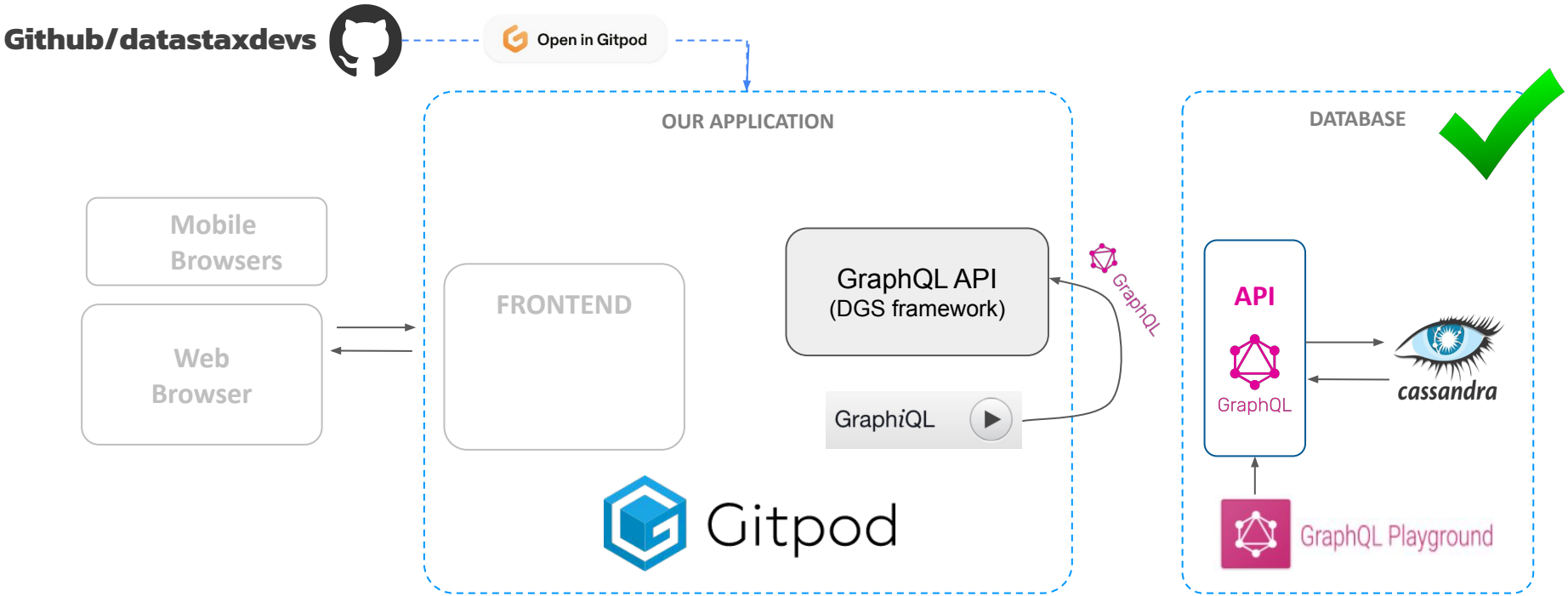


ENDPOINT URL

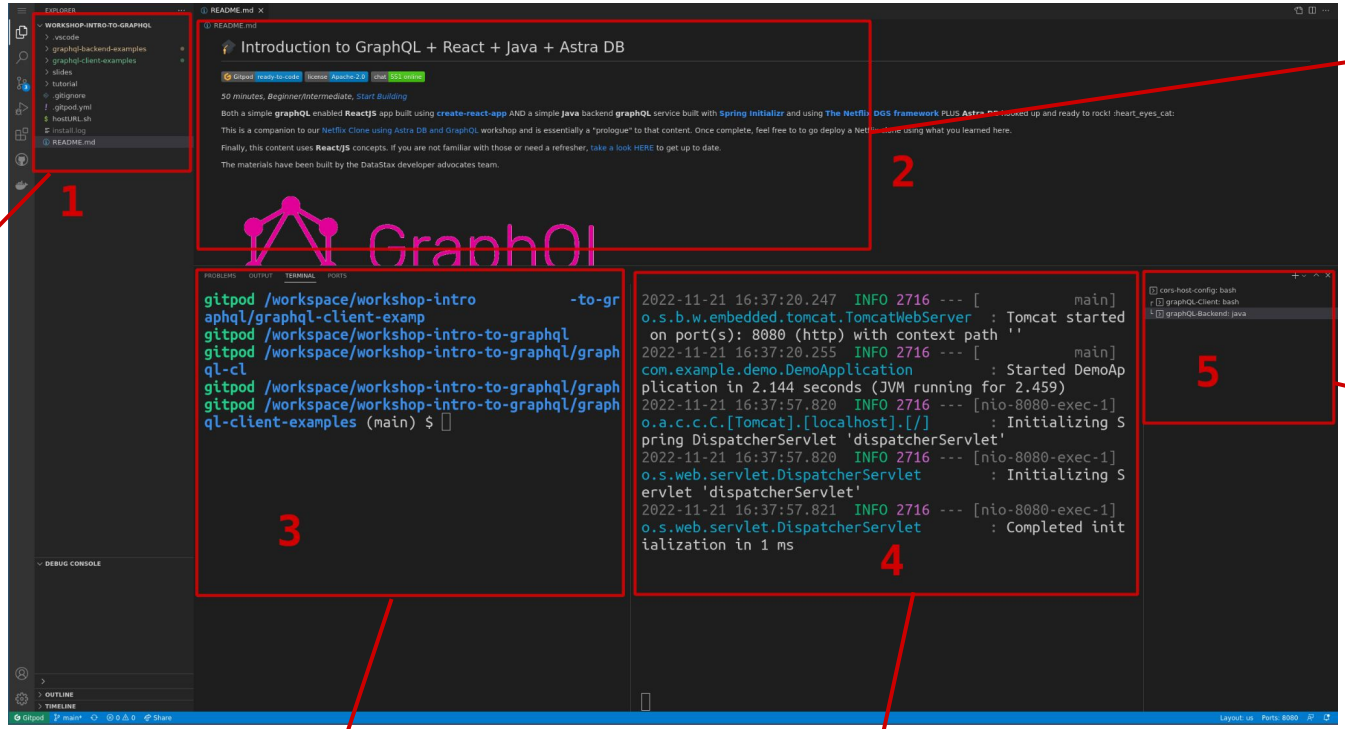
EXECUTE

HEADER

evelopers



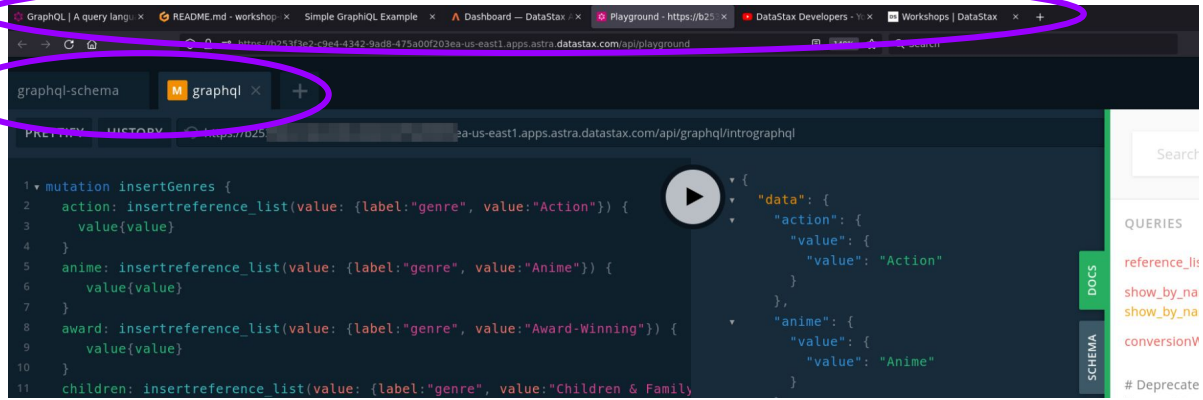
What's coming in next Hands-on ?



Gitpod UI cheat sheet

In your GraphQL Playground ...

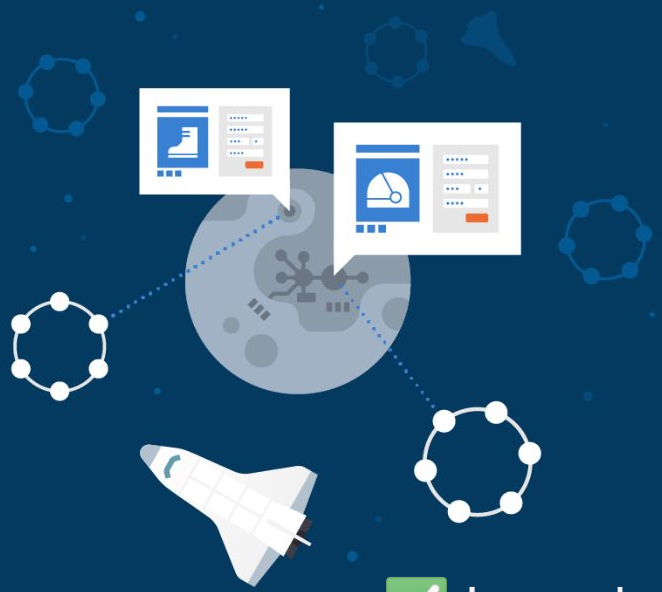
These are your browser's tabs

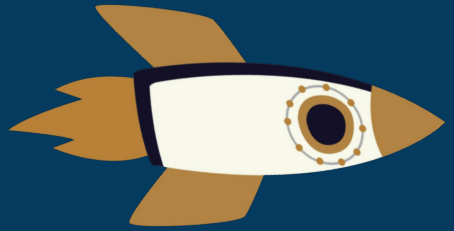


These are the playground tabs ("graphql-schema" & "graphql")

Tabs, tabs, tabs: know the difference

DataStax Developers





Hands-on (!github)

#2 First Touch

- ✅ Launch Gitpod
- ✅ Experiment with Graph*i*QL
- ✅ Demo with GraphQL Playground

01

Housekeeping

Live and Hands-on

02

Database Setup

Data model & Astra DB

03

GraphQL

GraphQL

Design & Toolings

04

Backend

Expose GQL endpoint

05

Frontend

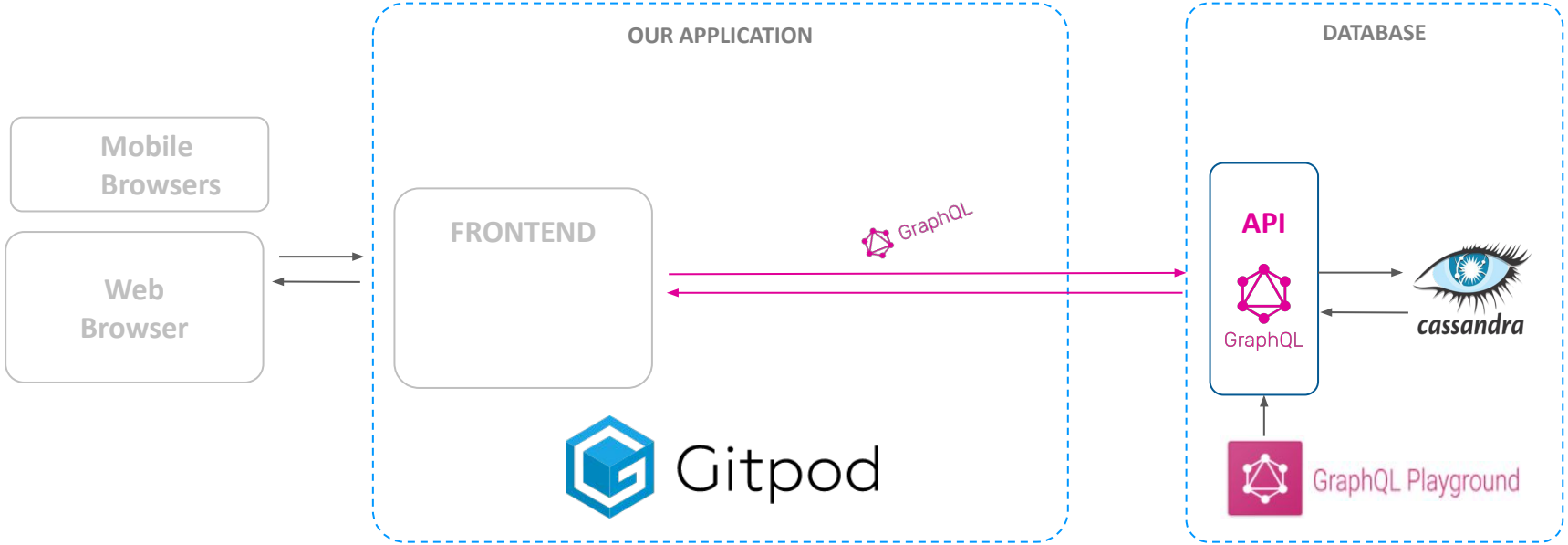
Consume GQL Endpoint

06

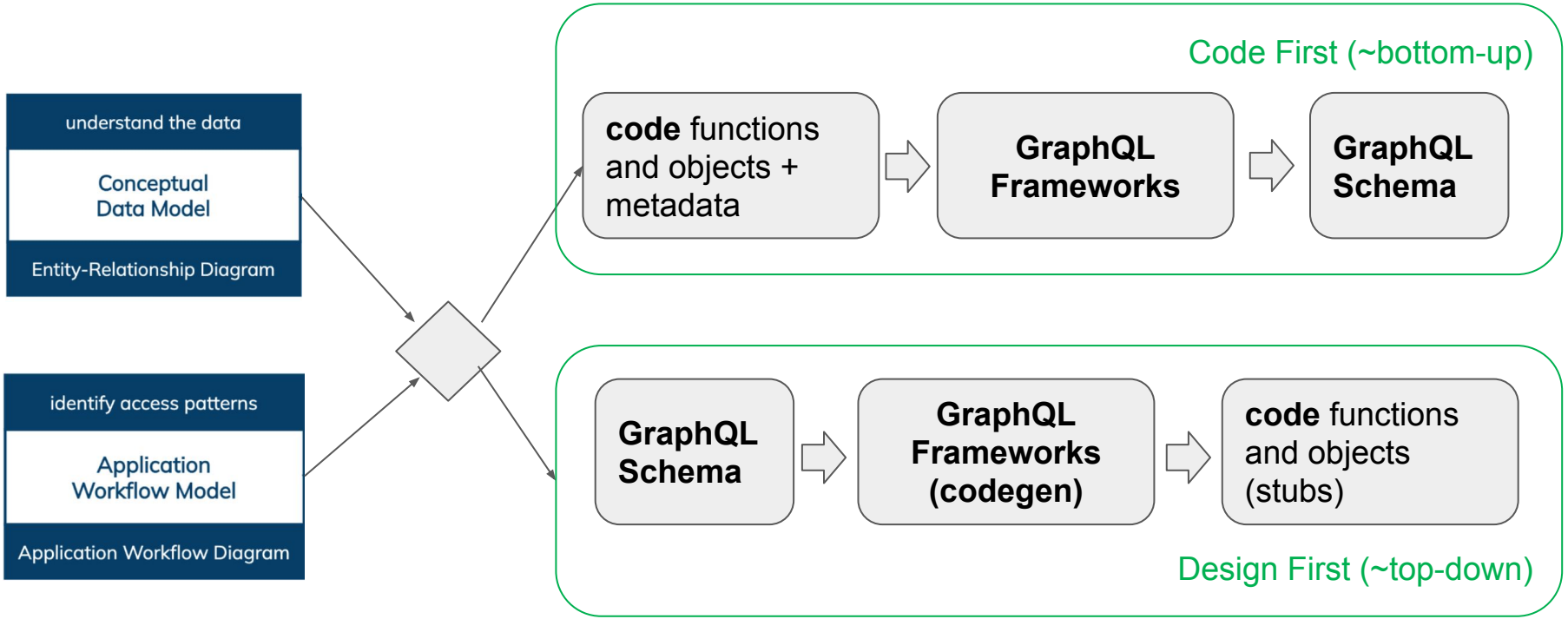
What's next?

Quiz, Homework, ...

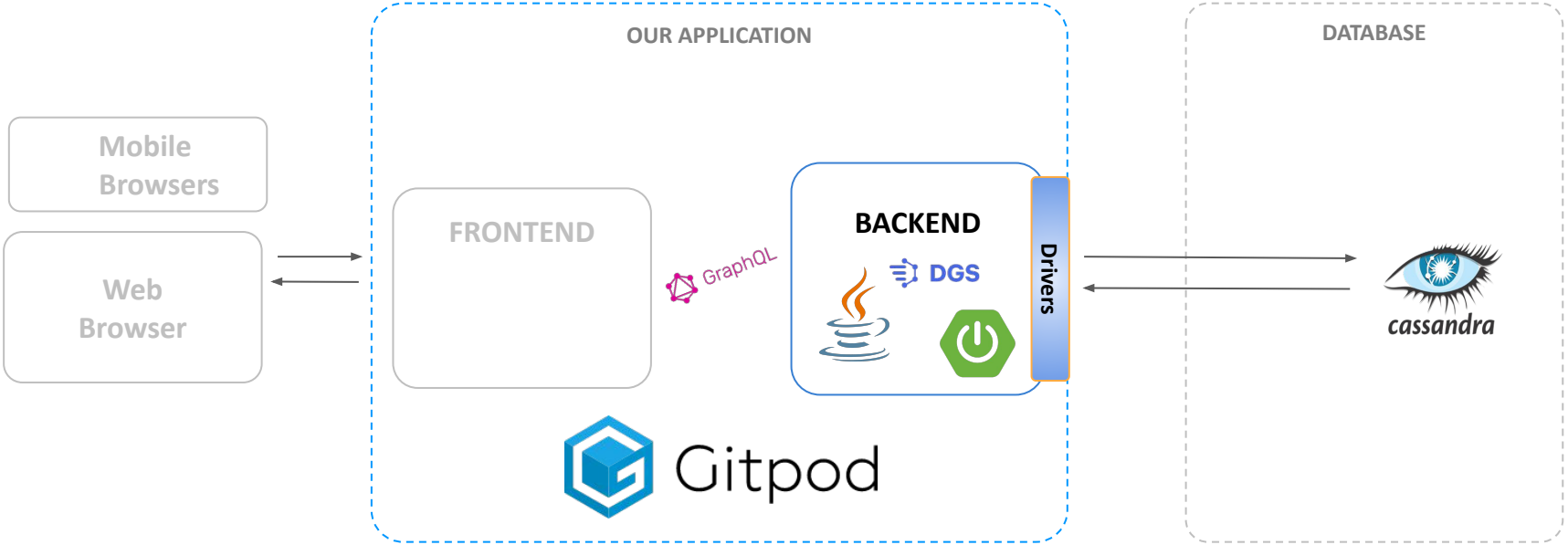
Agenda



Sooo Connect the Front end to Graphql



Build a GraphQL API



Introduction to DGS by Netflix

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

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

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

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`

Exposing a GQL Endpoint with

01

Housekeeping

Live and Hands-on

02

Database Setup

Data model & Astra DB

03

GraphQL

GraphQL

Design & Toolings

04

Backend

Expose GQL endpoint

05

Frontend

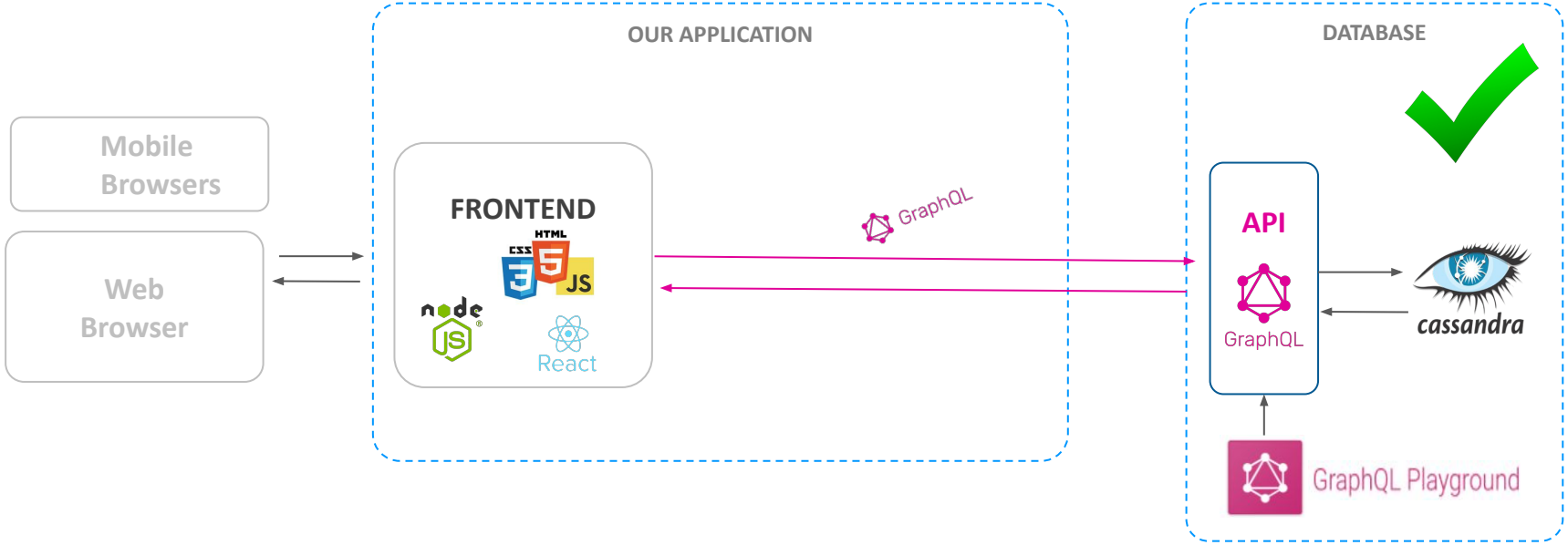
Consume GQL Endpoint

06

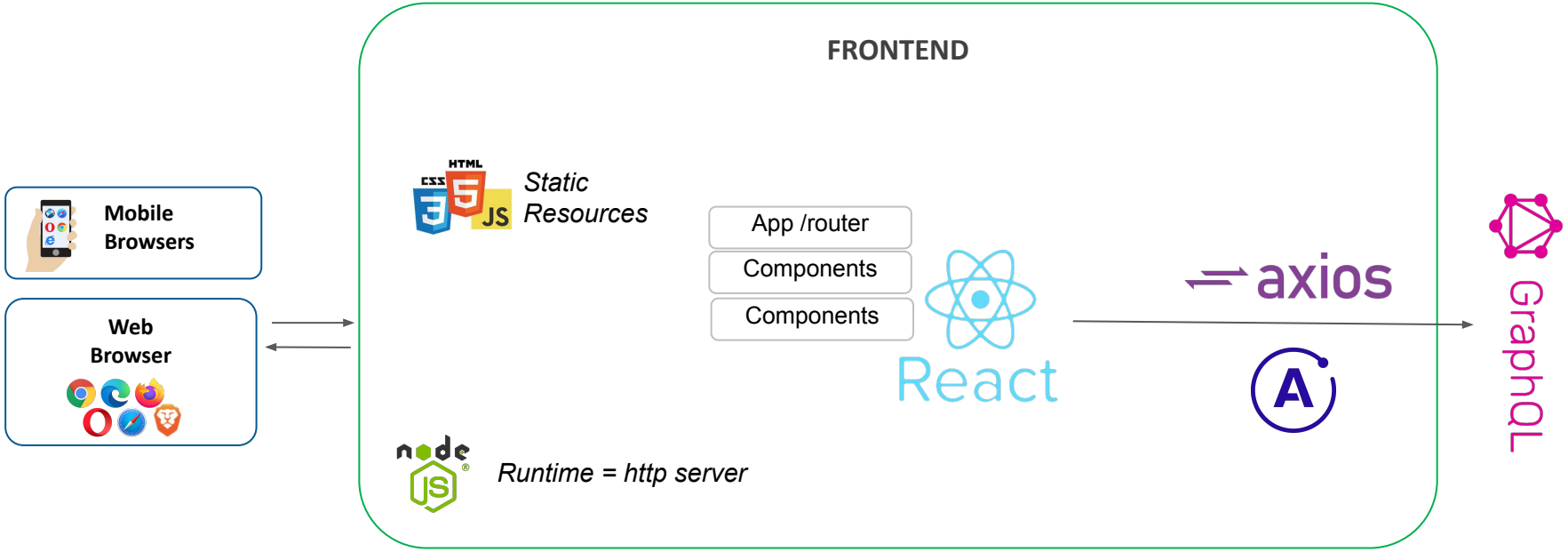
What's next?

Quiz, Homework, ...

Agenda



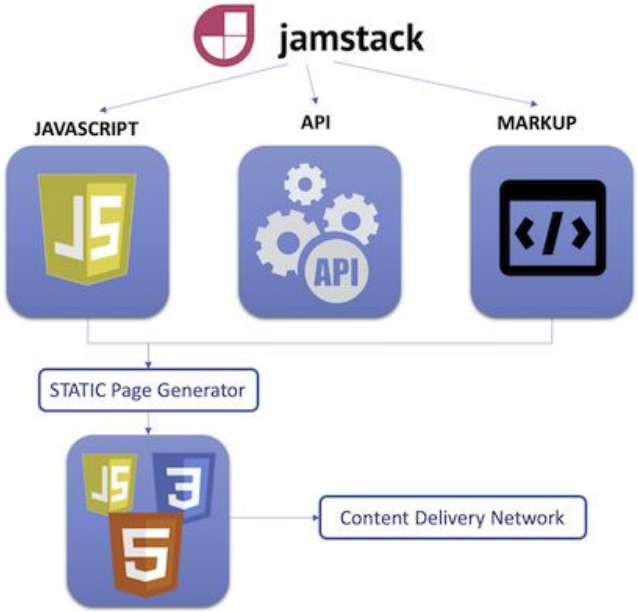
Sooo Connect the Front end to Graphql

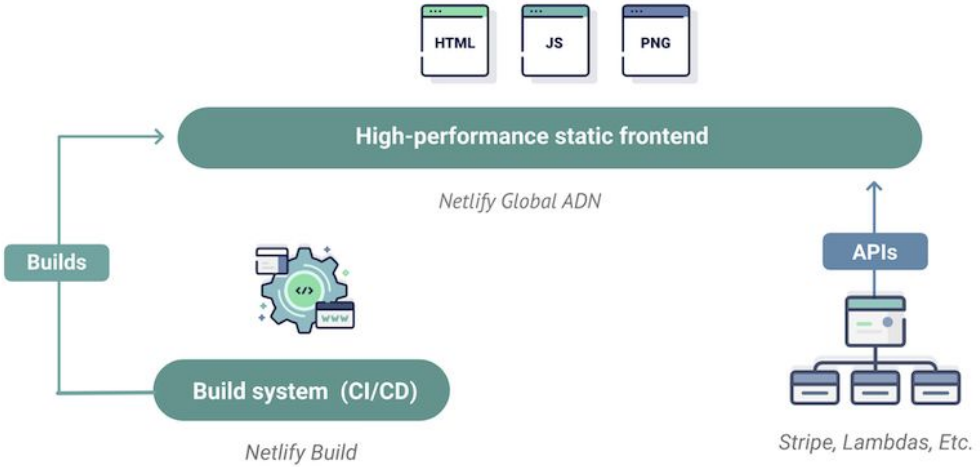


(*) Check out Apollo's `@apollo/client` library as well …

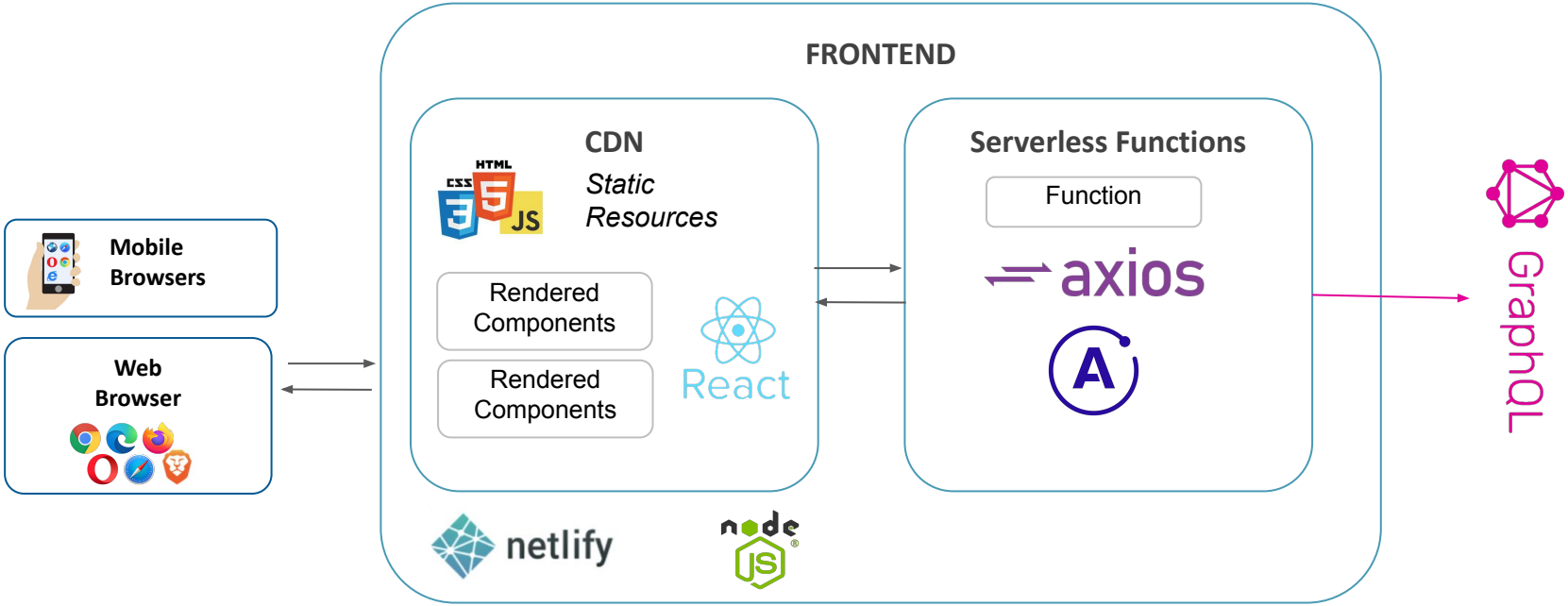
Our Front End

Cops are coming for you

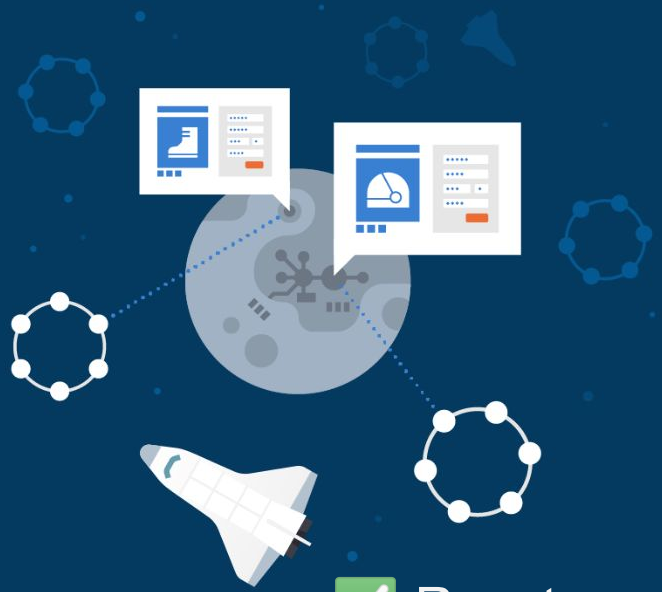


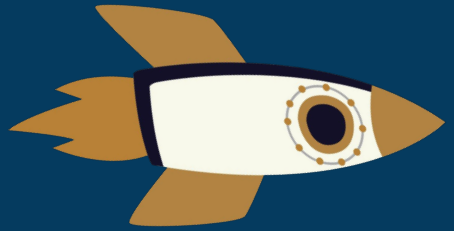


Introduction to Netlify



Our Front End





Hands-on (!github)

#3 Run the app

- ✅ React (start client, examine code)
- ✅ DB Connectivity (link app and DB)
- ✅ Run the final app (connected to DB, last touch)

01

Housekeeping

Live and Hands-on

02

Database Setup

Data model & Astra DB

03

GraphQL

GraphQL

Design & Toolings

04

Backend

Expose GQL endpoint

05

Frontend

Consume GQL Endpoint

06

What's next?

Quiz, Homework, ...

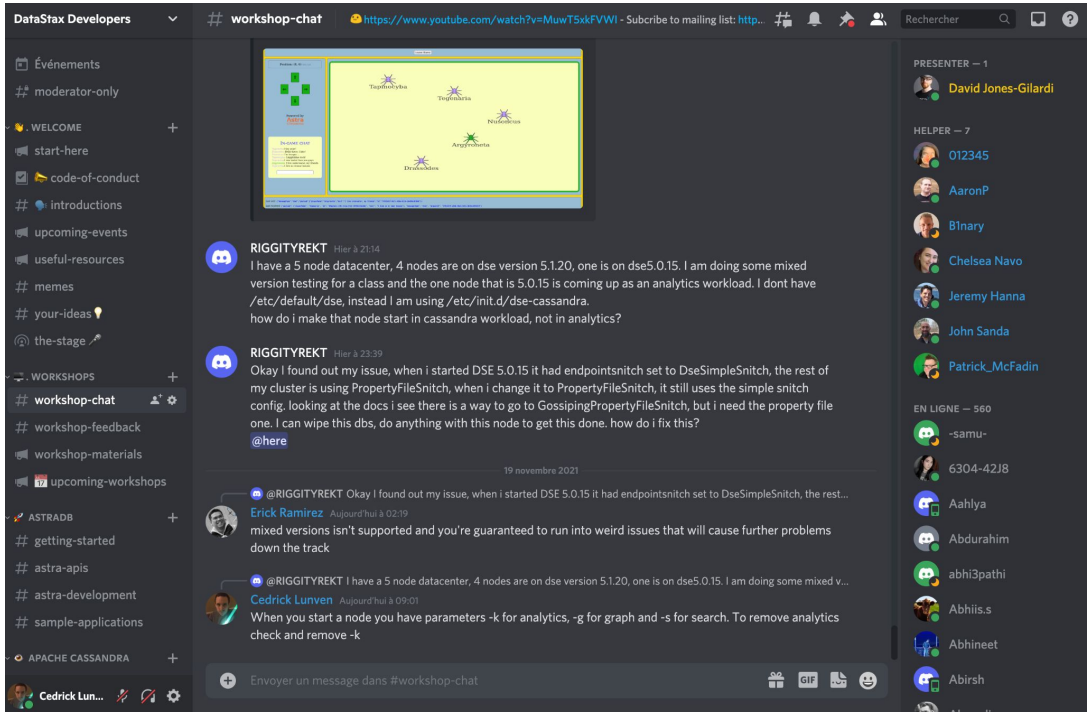
Agenda

What's next?

Quiz, Homework, …

- **Homework**
  - Complete the practice steps after step #9
  - Insert (mutate) a show or genre of your choice in the database.
  - Submit screenshot to us ... and wait for your **badge**!
- Optionally
  - Watch "Netflix Clone GraphQL" workshop recording
- Do reach us for anything: dtsx.io/discord !

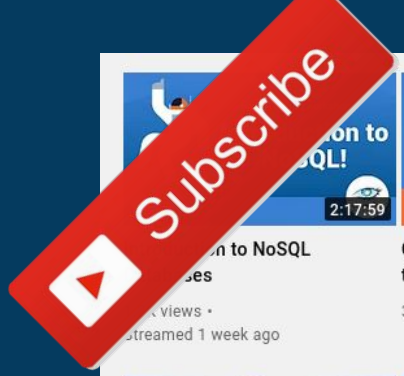
All info on Github - `!homework`

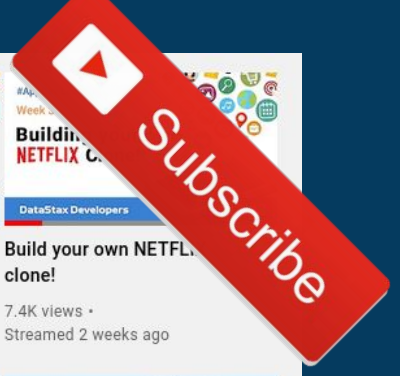


!discord

dtsx.io/discord

Datastax Developers Discord (19k+)





…n to NoSQL …ses

…k views •
…Streamed 1 week ago

Crash Course | Introduction to Cassandra for Developers

331 views • 1 week ago

Introduction to NoSQL Databases

3.5K views •
Streamed 1 week ago

Introduction to NoSQL Databases

10K views •
Streamed 1 week ago

Build your own NETFLIX clone!

4K views •
Streamed 2 weeks ago

Build your own NETFL… clone!

7.4K views •
Streamed 2 weeks ago

Astra Streaming Demo

177 views • 2 weeks ago

Kubernetes Ingress Management with Traefik…

496 views •
Streamed 2 weeks ago

Build your own TikTok clone!

1.9K views •
Streamed 3 weeks ago

Build your own TikTok Clone!

4K views •
Streamed 3 weeks ago

How to use the Connect Driver in Astra DB

113 views • 4 weeks ago

How to use the CQL Console in Astra DB

39 views • 4 weeks ago

How to create an Authentication Token in…

37 views • 4 weeks ago

How to use the Data Loader in Astra DB

62 views • 4 weeks ago

Astra DB Sample App Gallery

36 views • 4 weeks ago

How to use Secure Connect in Astra DB

42 views • 4 weeks ago

Cassandra Day India: CL Room (Workshops)

2.4K views •
Streamed 4 weeks ago

Cassandra Day India: RF Room (Talks)

1.3K views •
Streamed 1 month ago

Thank You!