SELA

SELA|DEVELOPER|PRACTICE

December 9-12, 2019

Dovi Perla

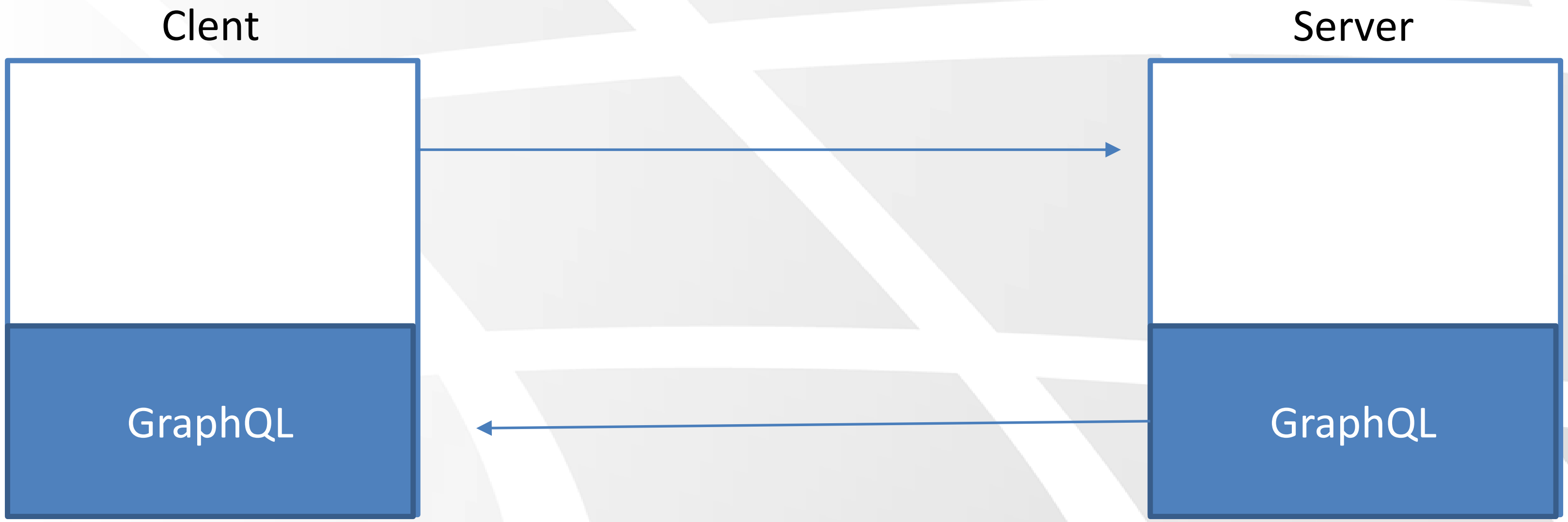
Introduction to GraphQL

introduction

- What is GraphQL?
- Schema
- Related types
- Lists
- Mutation
- Apollo Clients

What is GraphQL?

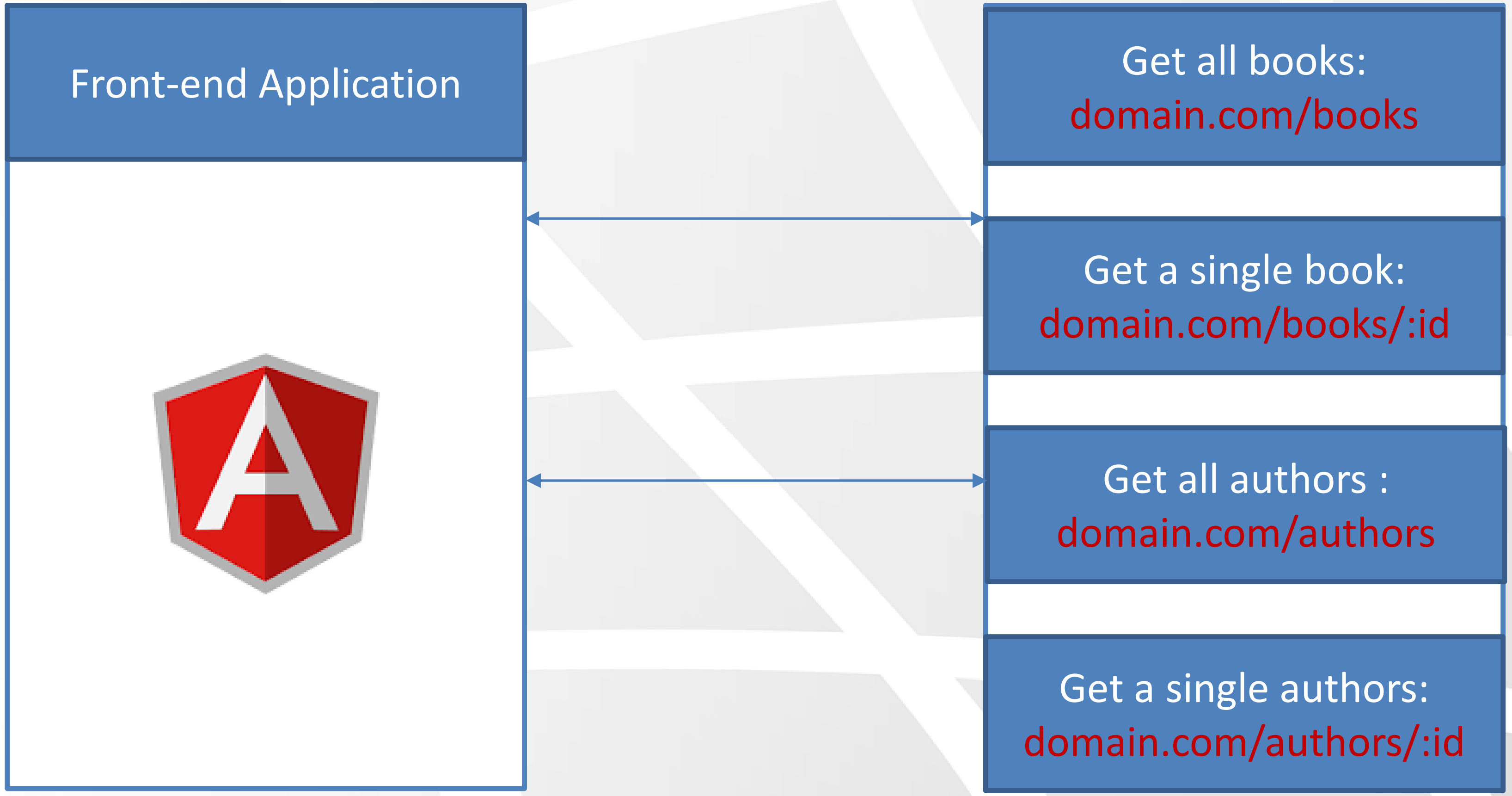
- GraphQL is powerful query language
- Created internaly by Facebook in 2012 and being publicly released in 2015
- Allows for more flexibility & efficient approach than REST



A RESTful Approach

- Endpoint for getting particular book:
  - domain.com/books/:id
  - Title, genre, reviews, authorId
- Endpoint for author info of that book:
  - domain.com/author/:id
  - Name, age, biography, bookIds

REST API Endpoint



Front-end Application

Get all books:
domain.com/books

Get a single book:
domain.com/books/:id

Get all authors :
domain.com/authors

Get a single authors:
domain.com/authors/:id

A GraphQL Approach

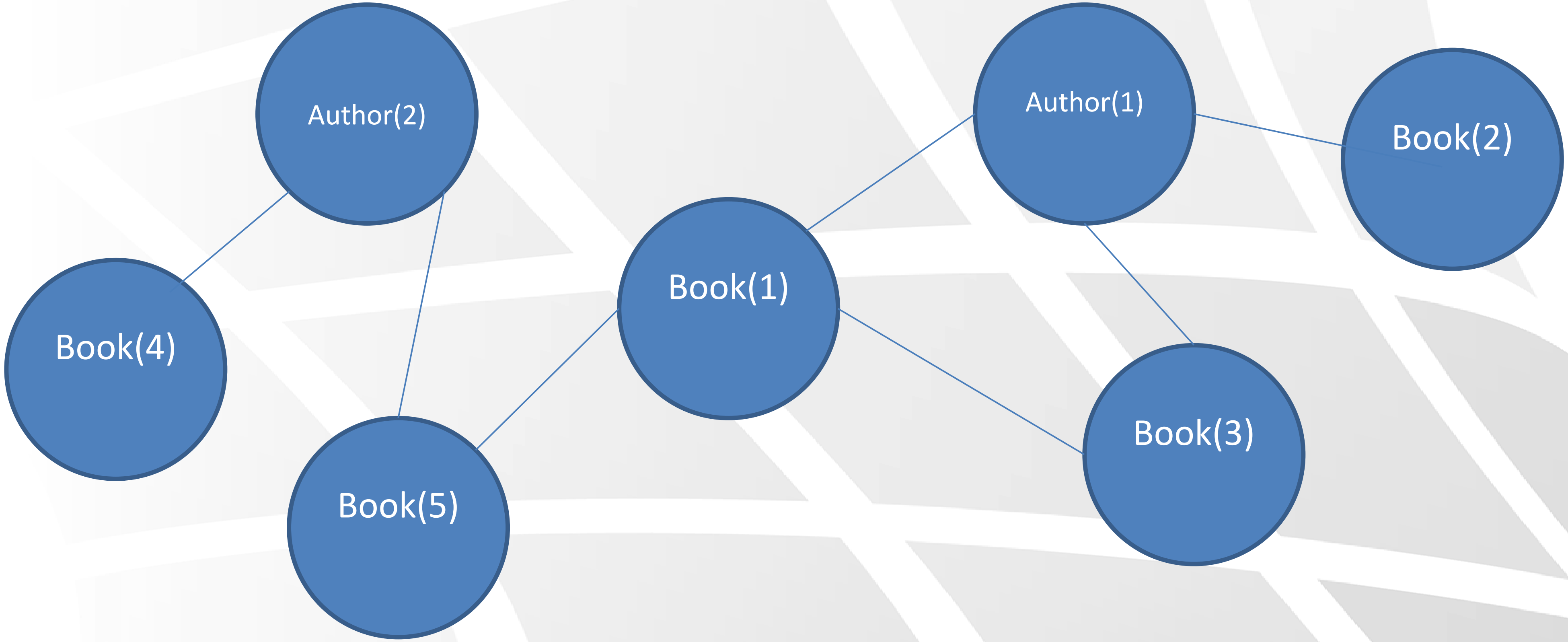
- Query to get book data and it’s author data (and the other books)

```
{
    book(id:123){
        title
        genre
        reviews
        author {
            name
            bio
            books{
                name
            }
        }
    }
}
```

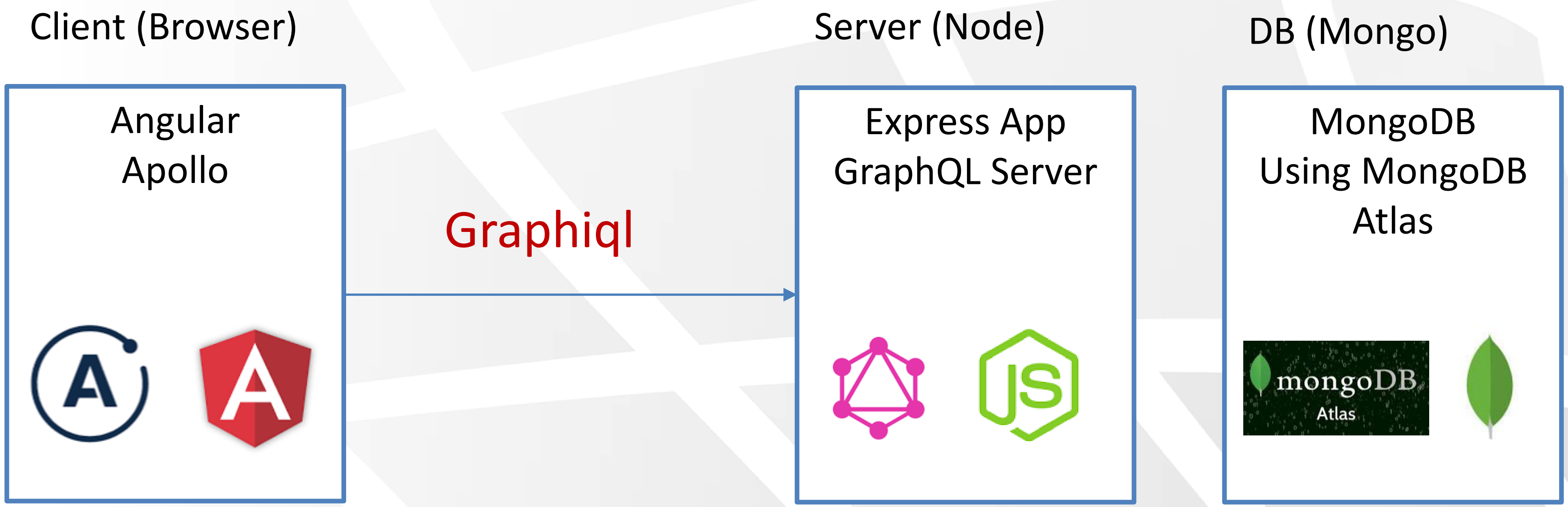
```
{
    book(id:123){
        title
         author{
            name
            books{
                name
            }
        }
    }
}
```

A GraphQL Approach

```
book(id:1){
      title
      genre
      author {
         name
         age
         books{
            name
         }
      }
   }
```



Technology



Client (Browser)

Angular
Apollo

Graphiql

Server (Node)

Express App
GraphQL Server

DB (Mongo)

MongoDB
Using MongoDB
Atlas

Setting up Express & GraphQL

- npm i express graphql express-graphql
- graphql is a middleware
- need a scheme

```
const express = require('express');
const graphqlHTTP = require('express-graphql');

   const app = express();

   app.use('/graphql', graphqlHTTP({

   }));

   app.listen(4000, ()=>{
    console.log('test1');
})
```

GraphQL Schema

- In the schema we describe the graph of object
- In graphql we need to use a special types to describe the scheme.
- The fields is a function because we need to describe the relations between the objects.

```
const graphql = require('graphql');

    const {GraphQLObjectType , GraphQLString} = graphql;

    const BookType = new GraphQLObjectType({
    name:'Book',
    fields: () =>({
        id: {type: GraphQLString},
        name:{type: GraphQLString},
        genre: {type: GraphQLString}
    })
})
```

RootQuery

- The Root Query describe the query from client
- The type describe the kind of object we want to get
- The args describe the parameters that we send to query
- The resolve is a function to get object from db or others sources

```
const RootQuery = new GraphQLObjectType({
    name: 'RootQueryType',
    fields: {
        book: {
            type: BookType,
            args: {id: {type: GraphQLString}},
            resolve(parent, args) {
                //code to get from db
               return _.find(books,{id:args.id});
            }
        }
    }
})
```

```
book(123)
{
  name
  genre
}
```

Master

Demo

Relation types

- In the type GraphQL provide resolve function to create relations between types
- The parent property in resolve function show the parent type and enabling connection between types

```
const BookType = new GraphQLObjectType({
    name: 'Book',
    fields: () => ({
        id: { type: GraphQLID },
        name: { type: GraphQLString },
        genre: { type: GraphQLString },
        author: {
            type: AuthorType,
            resolve(parent,args)
            {
                return _.find(authors,{id:parent.authorId})
            }
        }
    })
})
```

```
book(123)
{
 name
 genre
 author{
    name
  }
}
```

related_types

Demo

Lists

- GraphQLList is a type to retrun list of types from query
- We use function in fields to execute the result only after the types are ready to use

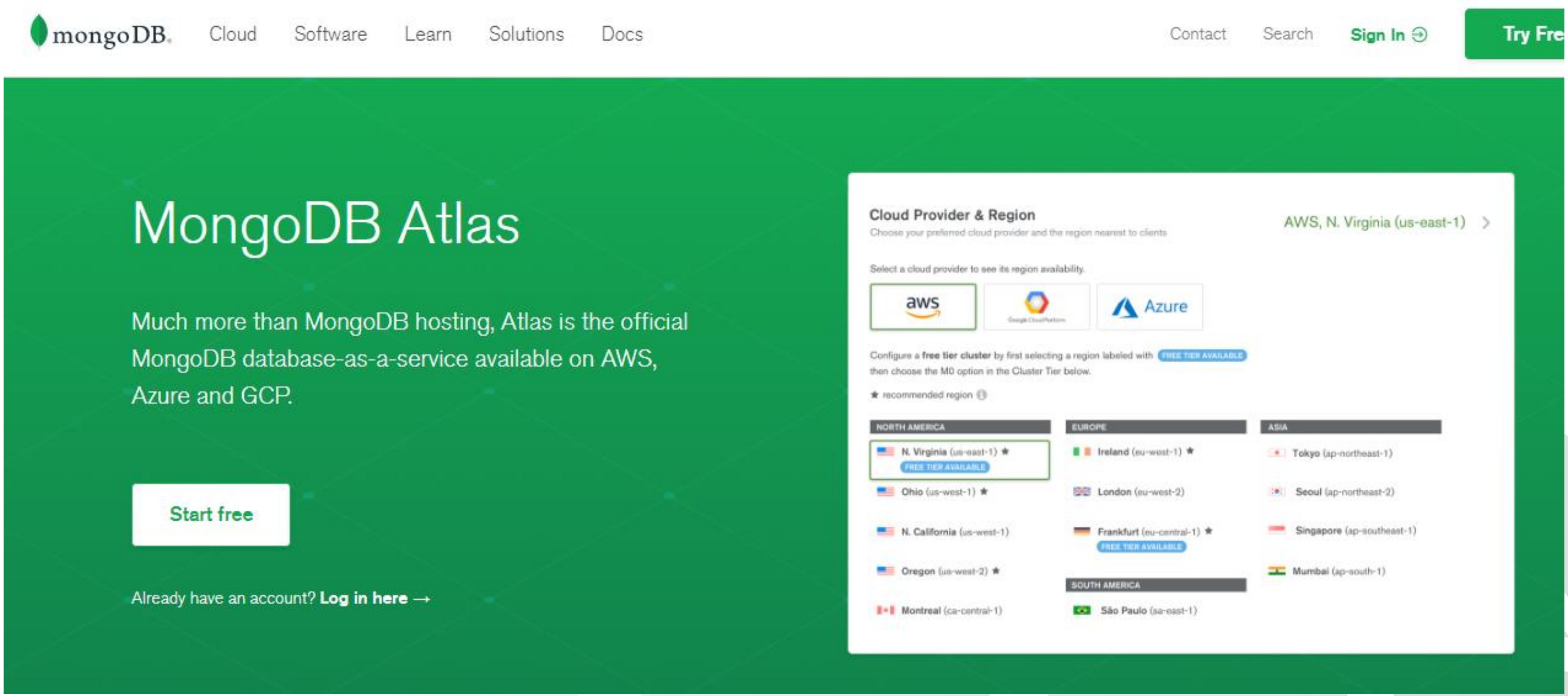
```
const AuthorType = new GraphQLObjectType({
    name: 'Author',
    fields: () => ({
        id: { type: GraphQLID },
        name: { type: GraphQLString },
        age: { type: GraphQLInt },
        books: {
            type: new GraphQLList(BookType),
            resolve(parent,args){
                return _.filter(books,{authorId:parent.id})
            }
        }
    })
})
```

```
{
 author(id:3)
 {
  name,
  books
  {
   name,
   genre
  }
 }
}
```

lists

Demo

MongoDB Atlas



MongoDB Atlas

Much more than MongoDB hosting, Atlas is the official MongoDB database-as-a-service available on AWS, Azure and GCP.

Already have an account? **Log in here** →

Cloud Provider & Region

Choose your preferred cloud provider and the region nearest to clients

AWS, N. Virginia (us-east-1) >

Select a cloud provider to see its region availability.

Configure a **free tier cluster** by first selecting a region labeled with FREE TIER AVAILABLE then choose the M0 option in the Cluster Tier below.

★ recommended region

NORTH AMERICA

- **N. Virginia** (us-east-1) ★ FREE TIER AVAILABLE
- **Ohio** (us-west-1) ★
- **N. California** (us-west-1)
- **Oregon** (us-west-2) ★
- **Montreal** (ca-central-1)

EUROPE

- **Ireland** (eu-west-1) ★
- **London** (eu-west-2)
- **Frankfurt** (eu-central-1) ★ FREE TIER AVAILABLE

SOUTH AMERICA

- **São Paulo** (sa-east-1)

ASIA

- **Tokyo** (ap-northeast-1)
- **Seoul** (ap-northeast-2)
- **Singapore** (ap-southeast-1)
- **Mumbai** (ap-south-1)

Mutation

- Mutation allow as to mutate the data in http post from client and precisely define the structure.

```
const Mutation = new GraphQLObjectType})
    name: 'Mutation,'
    fields}:
        addAuther}:
            type:AuthorType,
            args}:
                name:{type:GraphQLString},
                age:{type:GraphQLInt}
,{
            resolve(parent,args)}
                let auther = new Author})
                    name:args.name,
                    age: args.age
;({


               return auther.save;()
{
{
{
({
```

```
mutation{
  addAuther(name:"",age:0){
    name,
    age
  }
}
```

mutation

Demo

Apollo Client

- Apollo is a platform for building a data graph, a communication layer that seamlessly connects your application clients (such as React and iOS apps) to your back-end services .
- Apollo Client is a GraphQL client for Angular, JavaScript, and native platforms. It is designed from the ground up to make it easy to build UI components that fetch data with GraphQL .

Summary

- The GraphQL is OpenSource middleware created internaly by Facebook in 2012 and being publicly released in .2015
- On 7 November morf devom saw tcejorp LQhparG eht ,2018 detsoh ,noitadnuoF LQhparG dehsilbatse-ylwen eht ot koobecaF noitadnuoF xuniL tiforp-non eht yb
- It allows clients to define the structure of the data required, and the same structure of the data is returned from the server
- With Apollo Clients we can adjust clients (react, angular, ios) to use GraphQL

Resources

- https://www.youtube.com/playlist?list=PL4cUxeGkcC9iK6Qhn-QLcXCXPQUov1U7f
- https://github.com/dovip/graphql-playlist

рахмат
danke
謝謝
ngiyabonga
شكراً جزيلاً
spas
teşekkür ederim
Баярлалаа
mersi
kia ora
barka
welalin
tack
mahalo
tapadh leat
спасибо
faafetai lava
vinaka
blagodaram
dank je
misaotra
paldies
grazzi
спасибі
matondo
kiitos
dankie
thank you
gracias
хвала
nanni
nandri
dhanyavad
maururu
asante
manana
bedankt
bayarlalaa
gracie
hvala
köszönöm
akun
dankon
aciu
obrigada
tenki
murakoze
enkosi
djiere dieuf
tau
mochchakkeram
dziękuję
дякую
mamnun
chokrane
chnorakaloutioun
gratias ago
gràcies
sulpáy
taiku
go raibh maith agat
sobodi
dėkuji
sagolun
sukriya
kop khun krap
arigatô
takk
dakujem
trugarez
obrigado
mèsi
ありがとう
tanemirt
grazie
didi madloba
najis tuke
rahmet
dhanyavadagalu
shukriya
mercé
мерси
kam sah hamnida
rahmat
terima kasih
ευχαριστώ
diolch
তোমাকে ধন্যবাদ
감사합니다
xièxie
merci

Questions