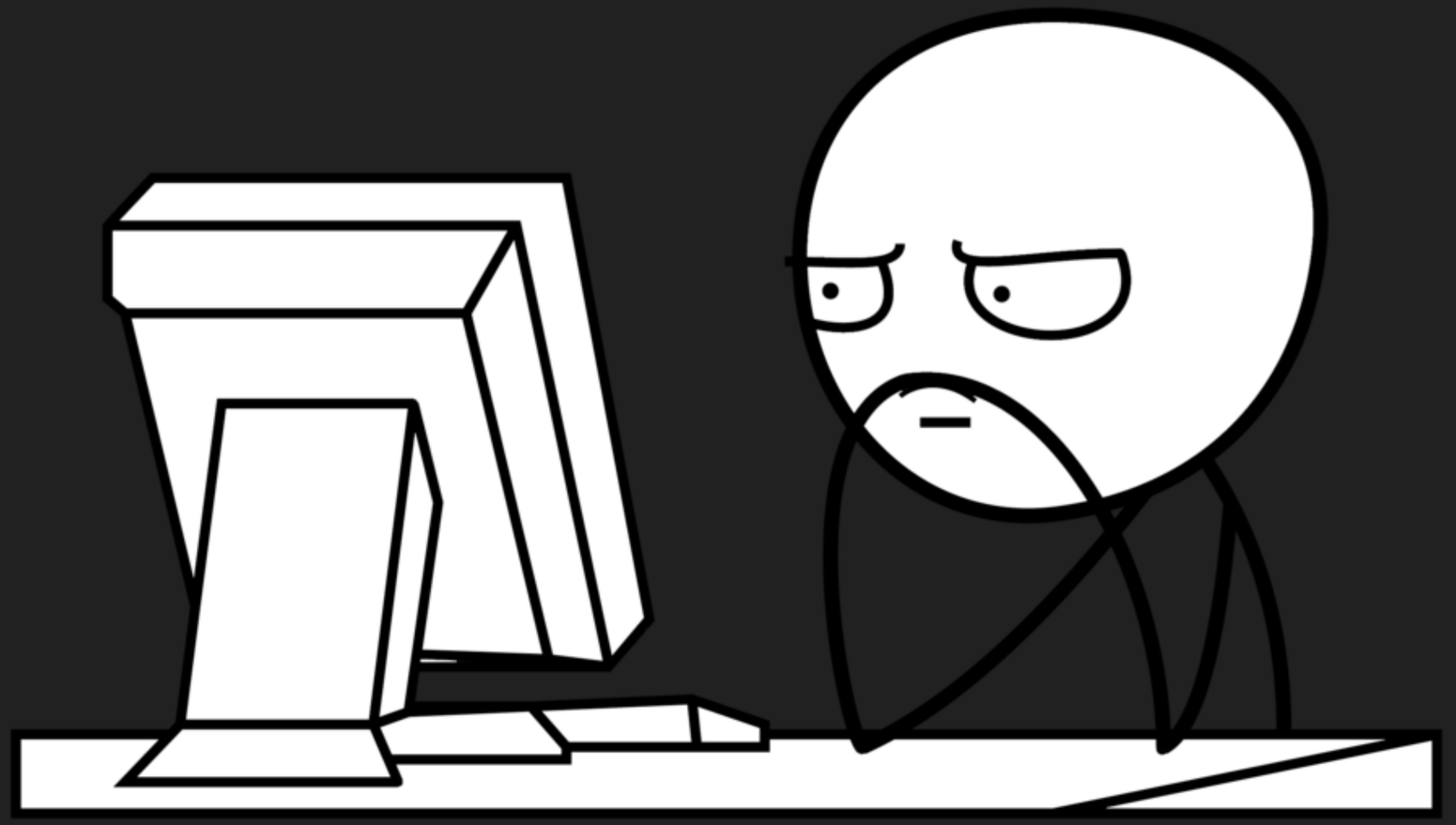
Apollo Federation 101

Why GraphQL?



Fan favorites >

This week's top TV and movies

★7.5 ☆

The Suicide Squad

+ Watchlist

▶ Trailer

★8.8 ☆

Shershaah

Watch options

▶ Trailer

★7.6 ☆

Free Guy

+ Watchlist

▶ Trailer

★8.4 ☆

Loki

+ Watchlist

▶ Trailer

★7.3 ☆

A Quiet Place Part II

+ Watchlist

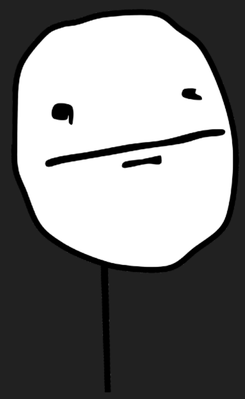
▶ Trailer

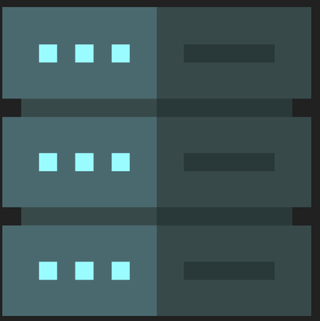
★7.7 ☆

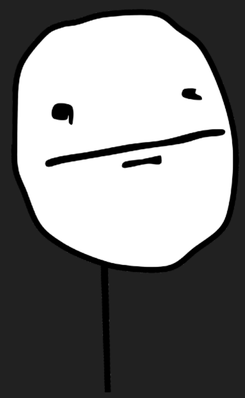
The White Lotus

+ Watchlist

▶ Trailer

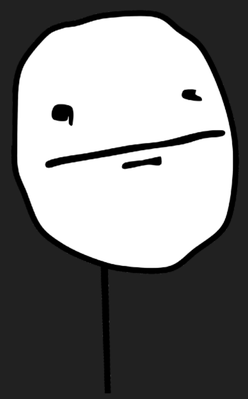






/movies

[{ id: 1, … }, { id : 2, … }, …]



/movies

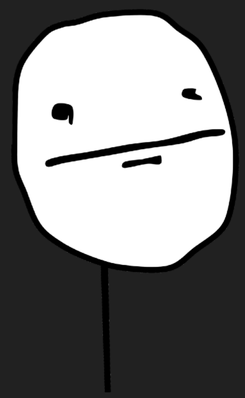
[{ id: 1, … }, { id : 2, … }, …]

/reviews/1

```
[{
  id: 1
},
… ]
```

.
.
.


THAT'S A LOT OF REQUESTS
imgflip.com

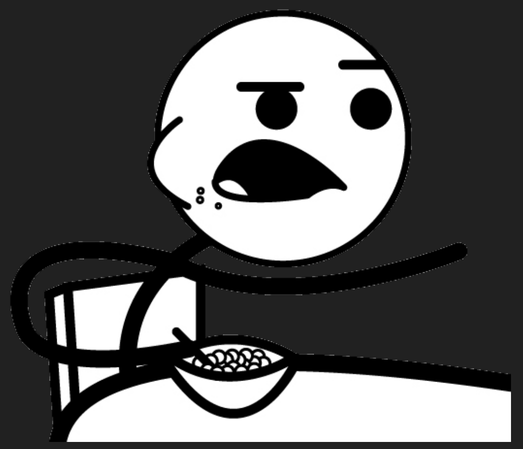


/movies

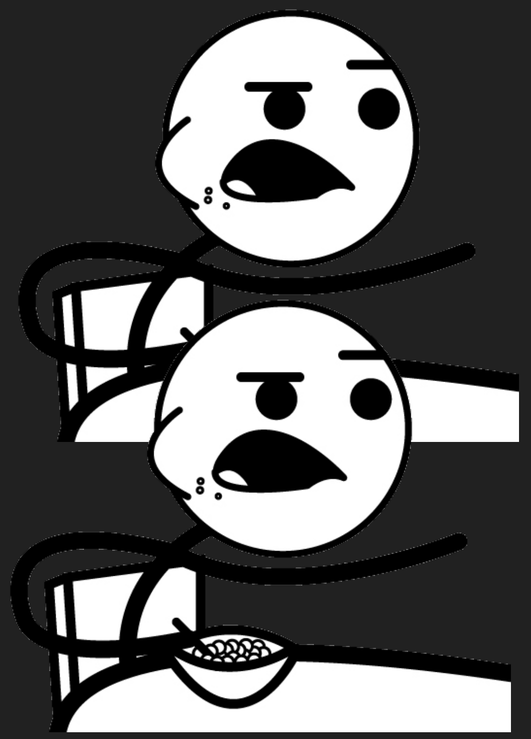
[{ id: 1, … }, { id : 2, … }, …]

/reviews/[movieIds]

```
[{
  id: 1
},
… ]
```

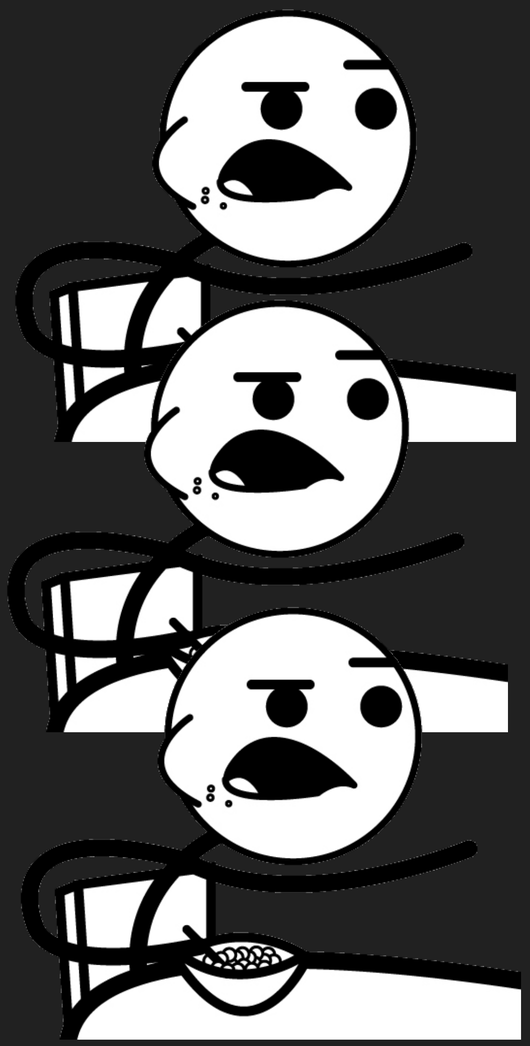


I want movies' title with reviews



I want movies' title with reviews

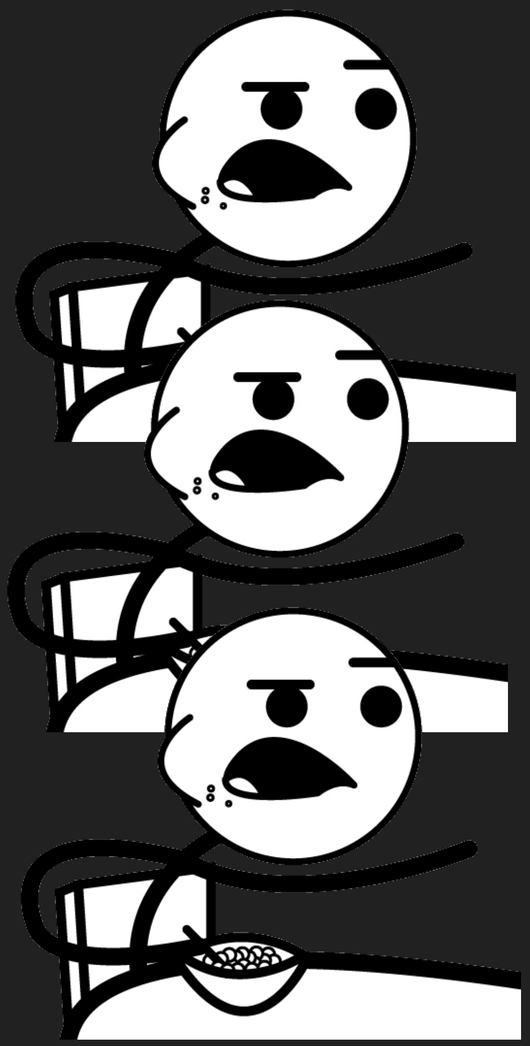
I want movies' title and comment without reviews



I want movies' title with reviews

I want movies' title and comment without reviews

I want movies' review with only review rating



I want movies' title with reviews

I want movies' title and comment without reviews

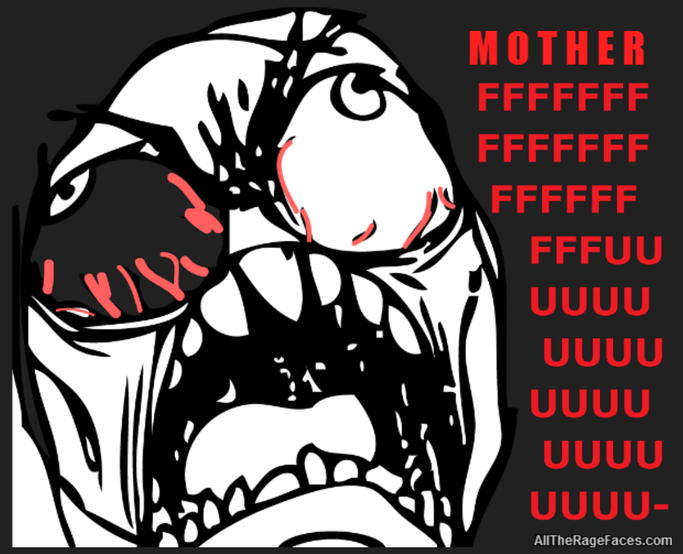
I want movies' review with only review rating


Bernie
Frontend guy
I am once again asking
What's the type of the response?
imgflip.com

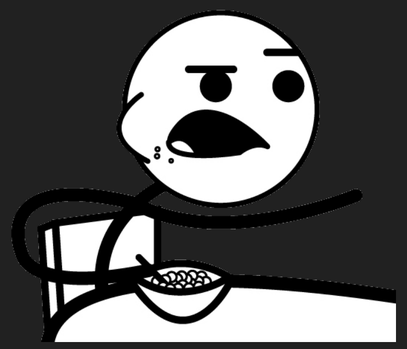
Over/Under-fetching

Poor performance

Poor communication/
documentation



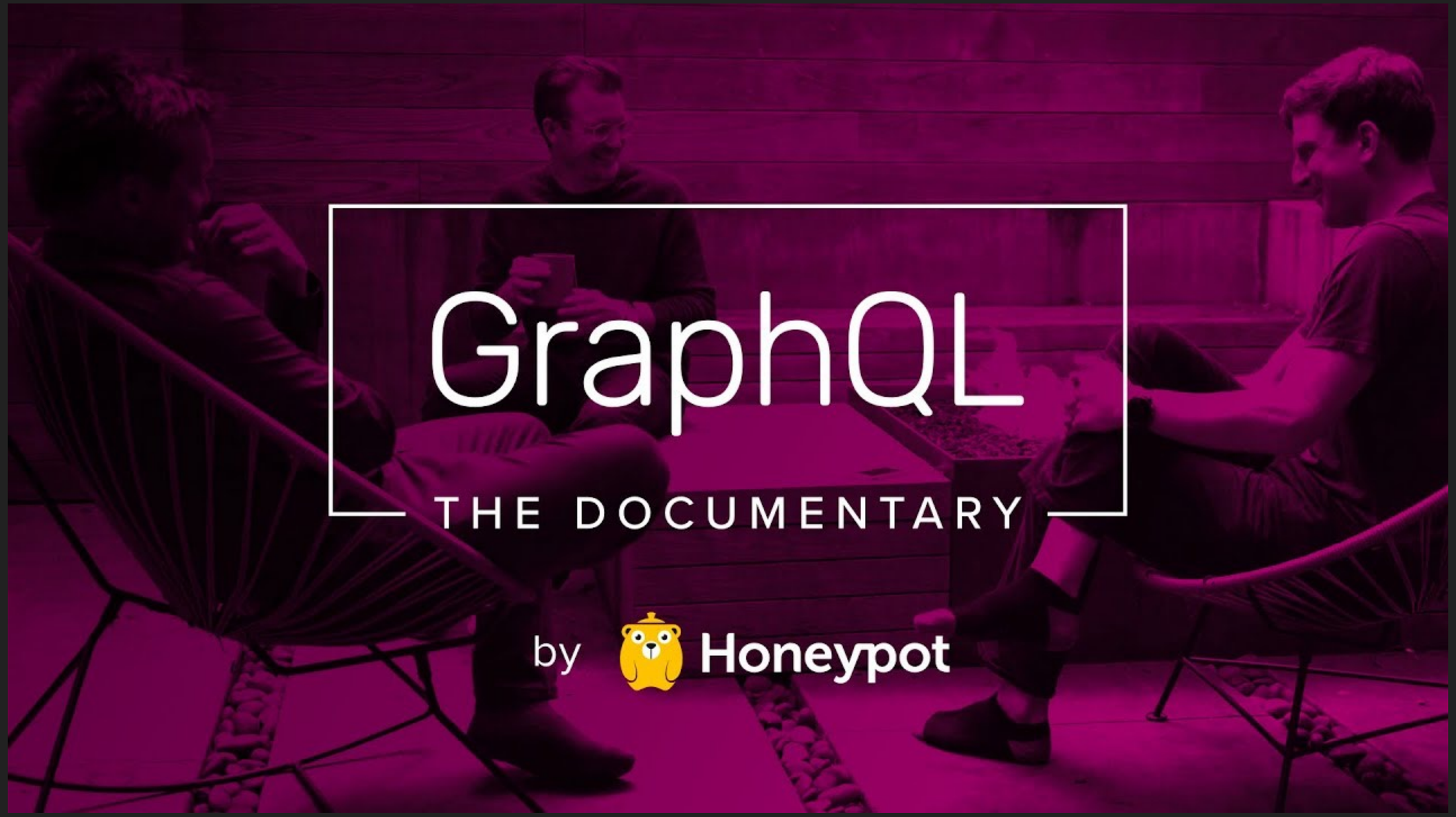

REST
GraphQL
imgflip.com



```graphql
query MoviesWithReviews {
  movies {
    title
    reviews {
      rating
      comment
    }
  }
}
```

```graphql
type Review {
  id: ID!
  rating: Int!
  comment: String
}

type Movie {
  id: ID!
  title: String!
  description: String
  reviews: [Review!]!
  avgRating: Float!
}

type Query {
  movies: [Movie!]!
  movie(id: ID!): Movie
}
```


⚠TRADE OFFER⚠
i receive:
what you want
you receive:
what you want
backend
imgflip.com

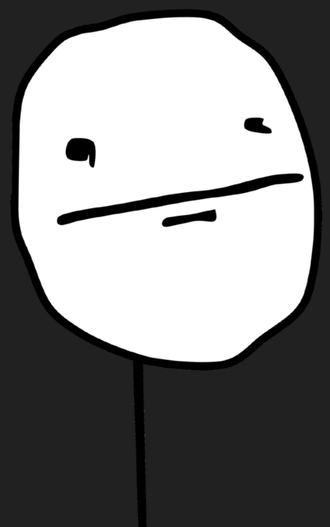


https://www.youtube.com/watch?v=783ccP_No8

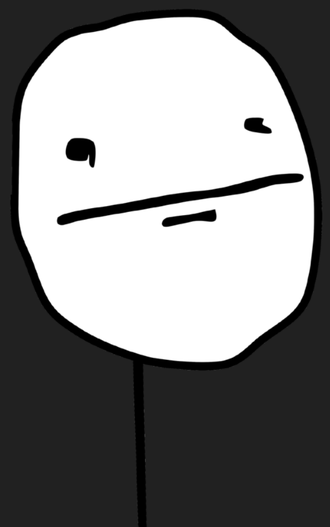
Quiz no.1

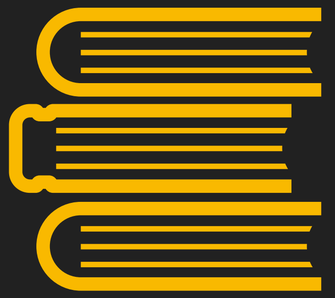
Why Microservices?

me

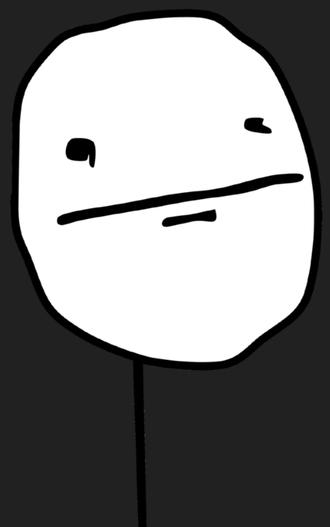


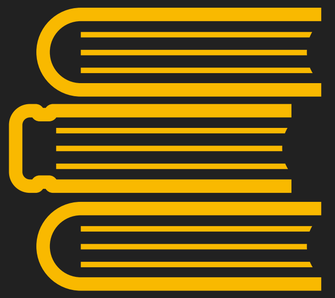
me

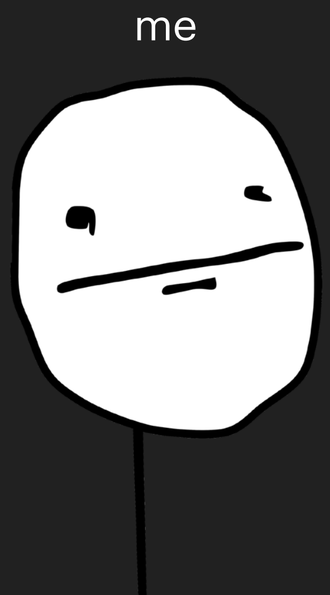


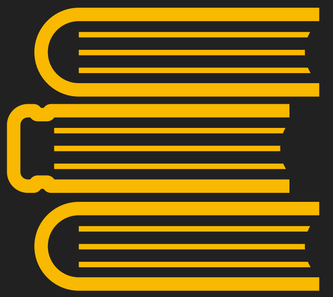


me

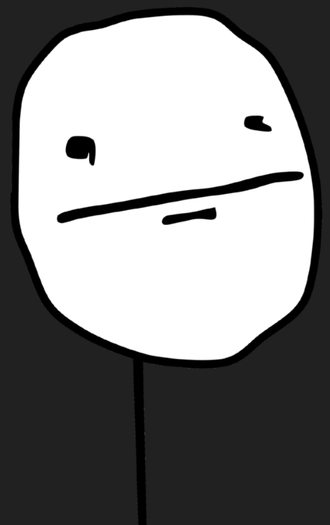


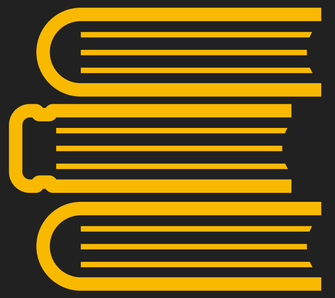


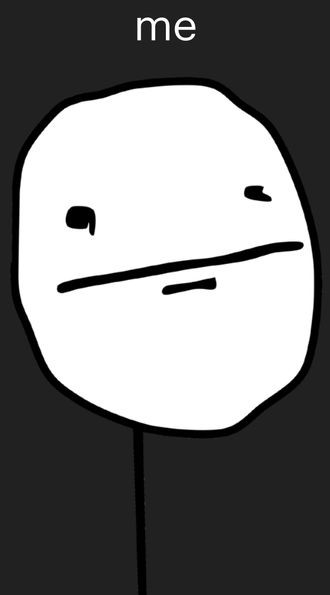

me

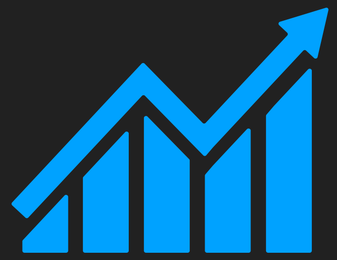


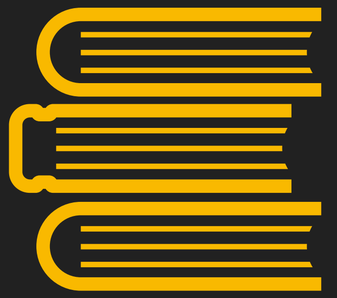
me

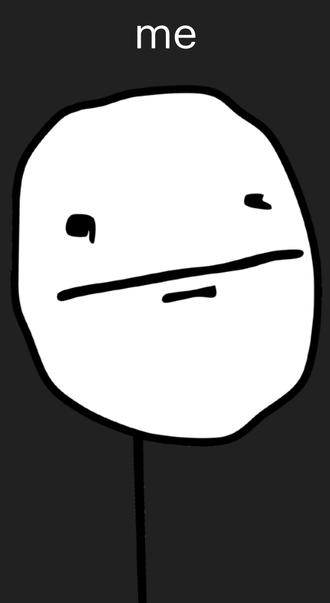


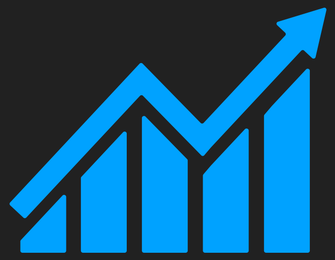


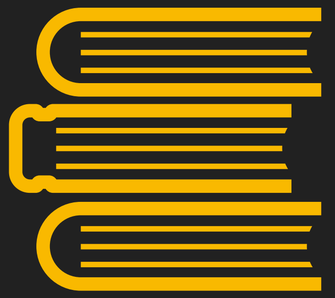

me



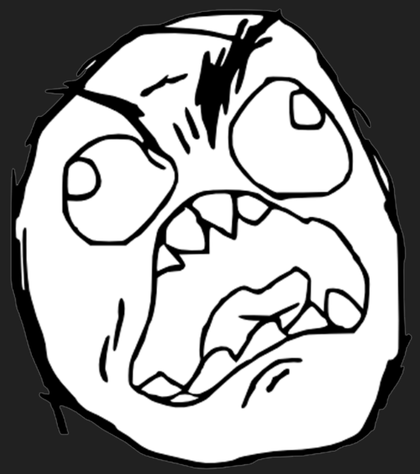


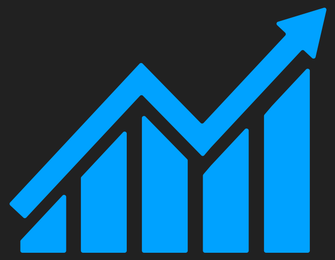

me

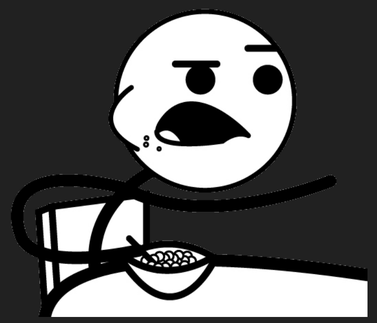




me



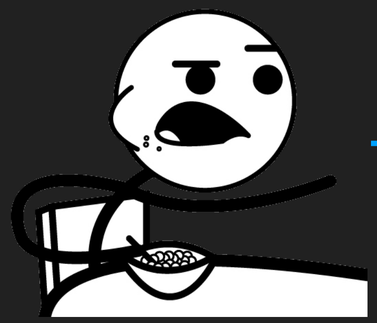




Gateway

Movie

Review

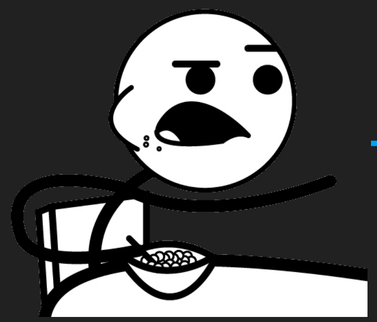


```
query MoviesWithReviews {
  movies {
    title
    reviews {
      rating
      comment
    }
  }
}
```

Gateway

Movie

Review



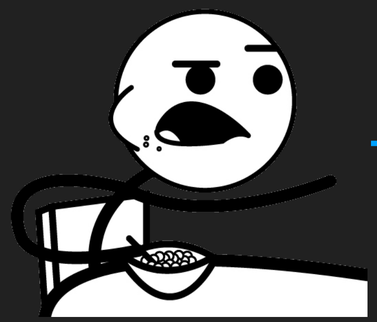
```
query MoviesWithReviews {
  movies {
    title
    reviews {
      rating
      comment
    }
  }
}
```

Gateway

```
query Movies {
  movies {
    id
    title
  }
}
```

Movie

Review



```
query MoviesWithReviews {
  movies {
    title
    reviews {
      rating
      comment
    }
  }
}
```

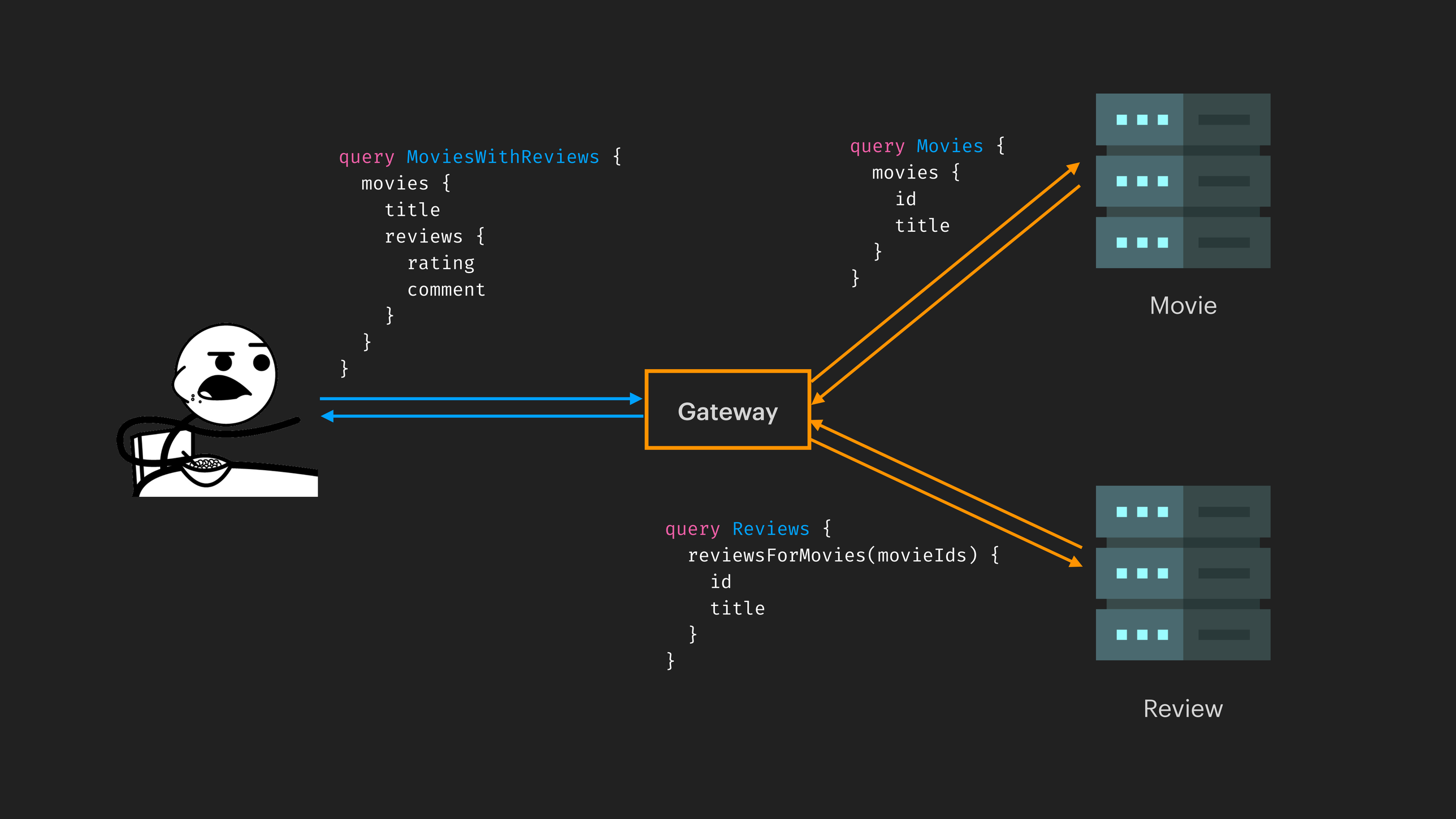
Gateway

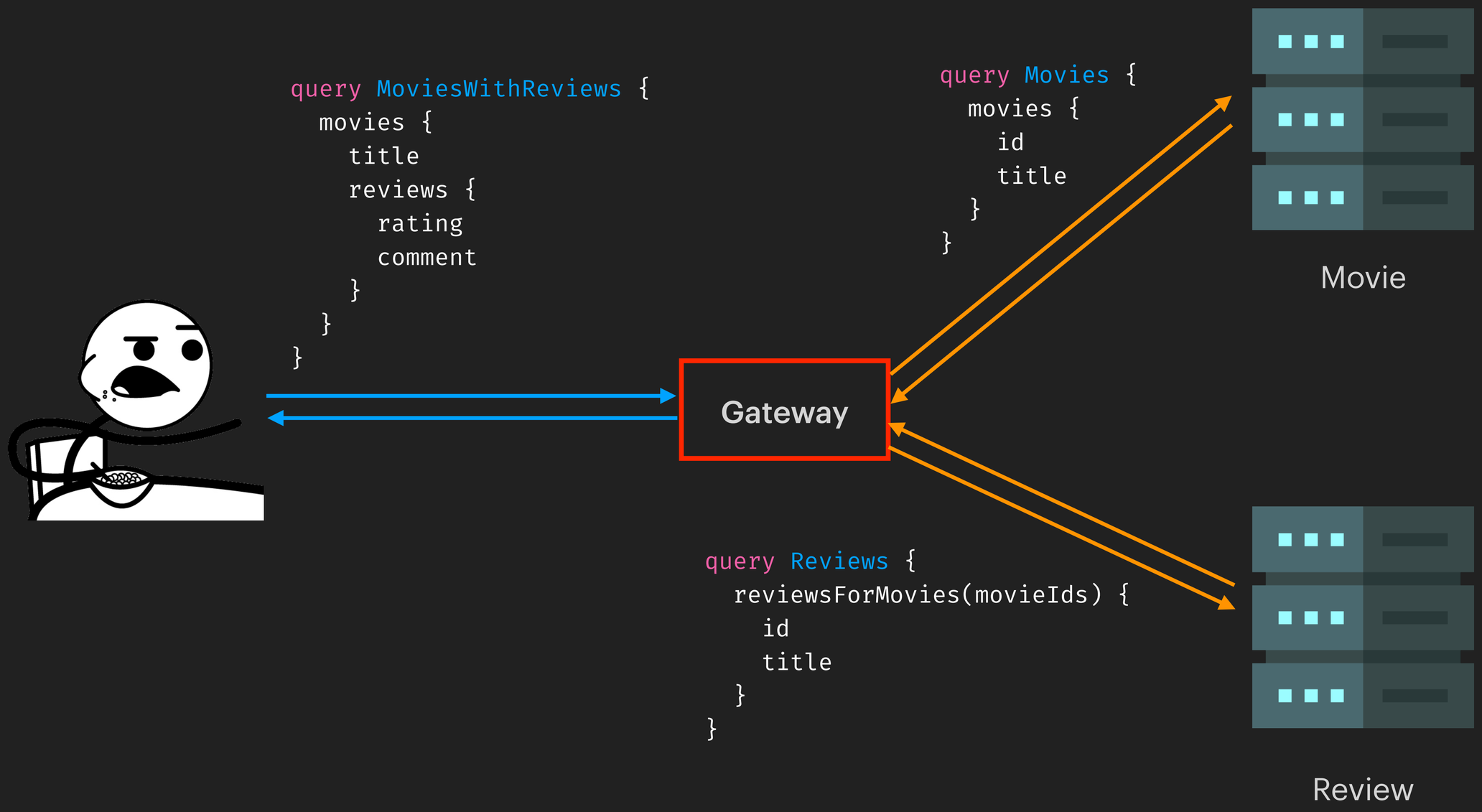
```
query Movies {
  movies {
    id
    title
  }
}
```

Movie

```
query Reviews {
  reviewsForMovies(movieIds) {
    id
    title
  }
}
```

Review


query MoviesWithReviews {
movies {
title
reviews {
rating
comment
}
}
}
query Movies {
movies {
id
title
}
}
Gateway
Movie
query Reviews {
reviewsForMovies(movieIds) {
id
title
}
}
Review


query MoviesWithReviews {
movies {
title
reviews {
rating
comment
}
}
}
query Movies {
movies {
id
title
}
}
Gateway
Movie
query Reviews {
reviewsForMovies(movieIds) {
id
title
}
}
Review

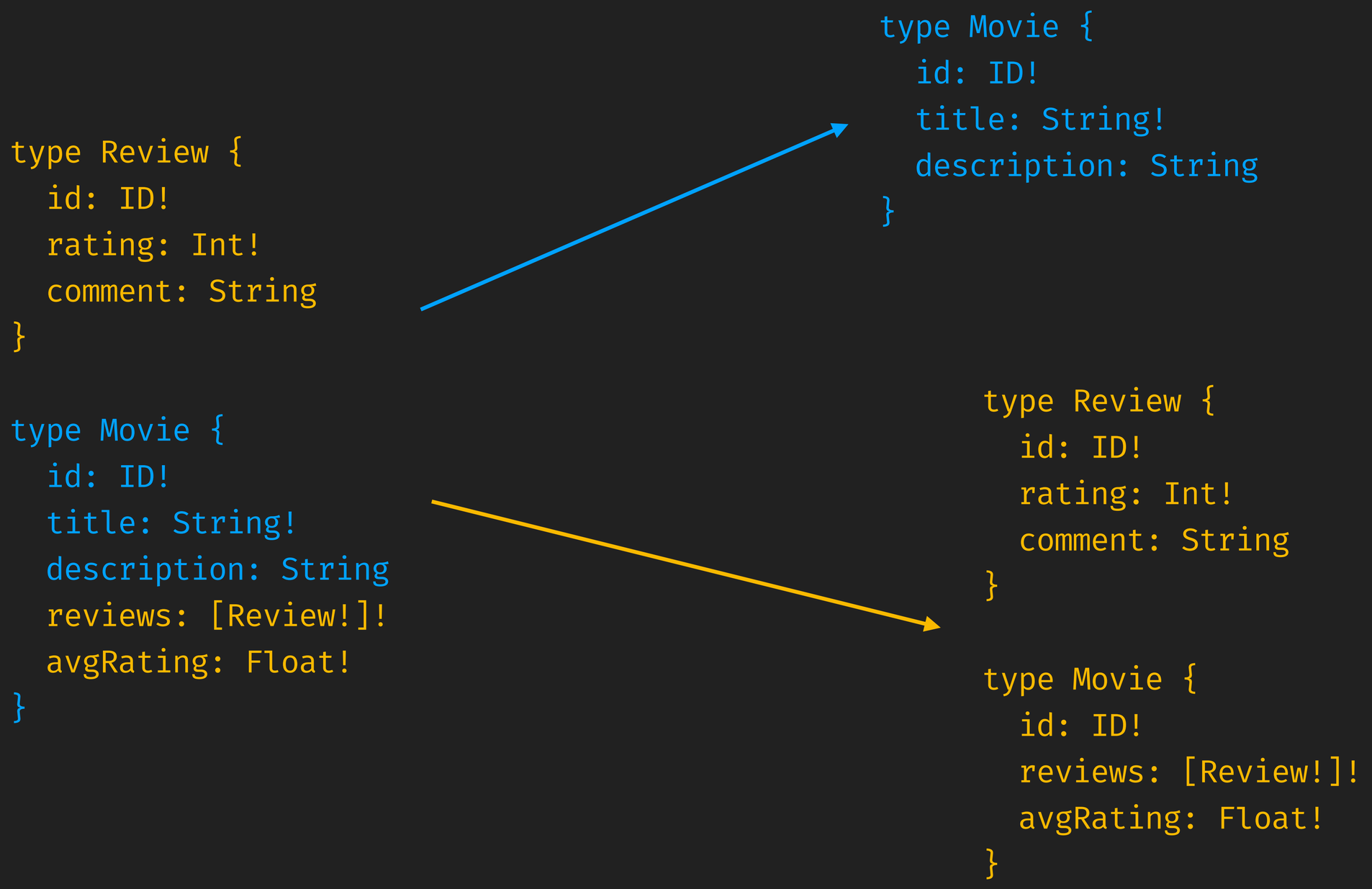
Quiz no.2

What is Apollo Federation?

"Dividing a big graph into many subgraphs across services"

TL;DR

```
type Review {
  id: ID!
  rating: Int!
  comment: String
}

type Movie {
  id: ID!
  title: String!
  description: String
  reviews: [Review!]!
  avgRating: Float!
}
```


type Review {
id: ID!
rating: Int!
comment: String
}
type Movie {
id: ID!
title: String!
description: String
}
type Movie {
id: ID!
title: String!
description: String
reviews: [Review!]!
avgRating: Float!
}
type Review {
id: ID!
rating: Int!
comment: String
}
type Movie {
id: ID!
reviews: [Review!]!
avgRating: Float!
}

Why Apollo Federation?

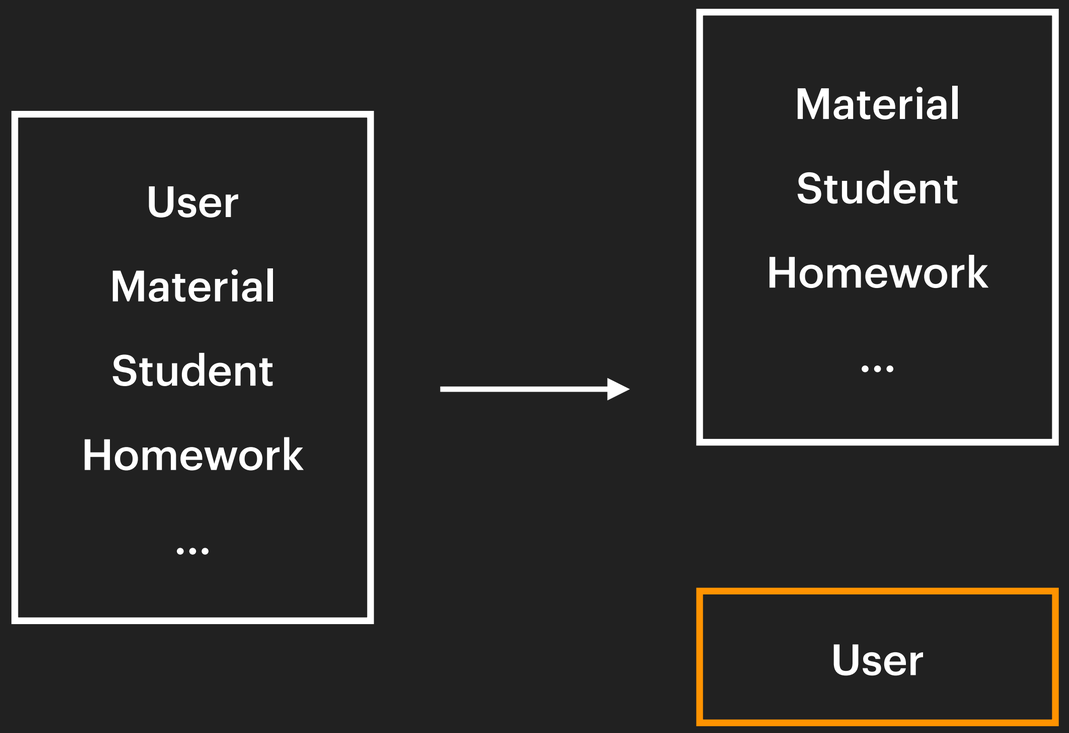
User

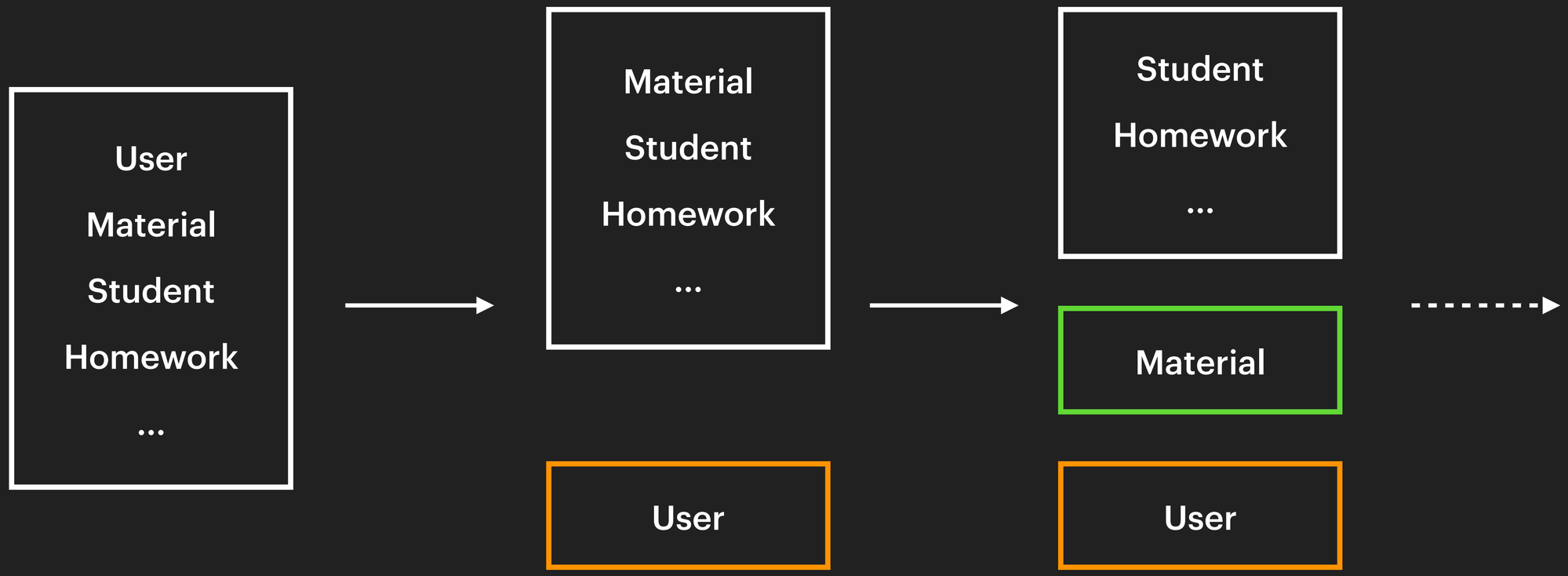
Material

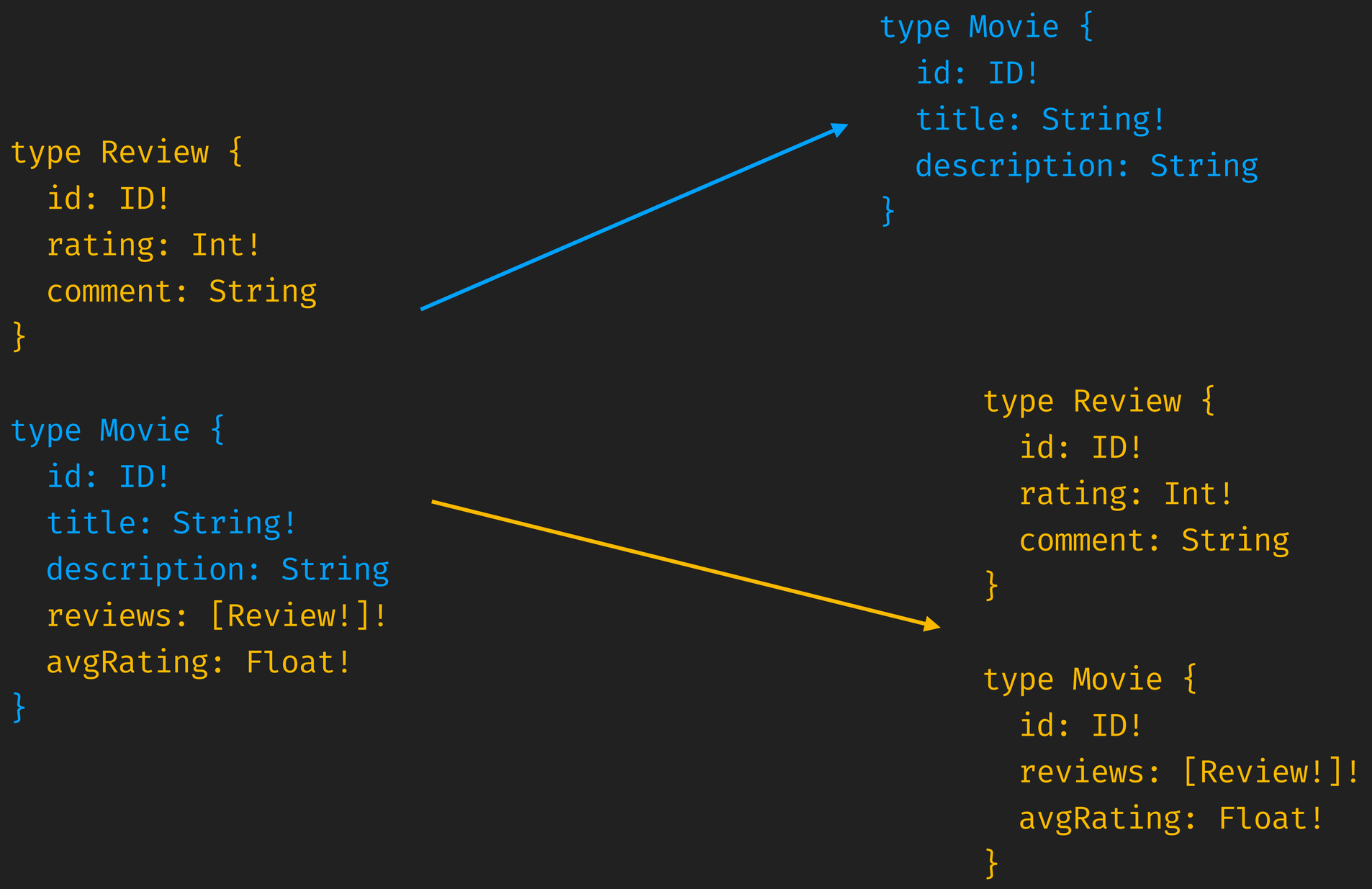
Student

Homework

...


User
Material
Student
Homework
...
Material
Student
Homework
...
User


User
Material
Student
Homework
...
Material
Student
Homework
...
User
Student
Homework
...
Material
User


type Review {
id: ID!
rating: Int!
comment: String
}
type Movie {
id: ID!
title: String!
description: String
}
type Movie {
id: ID!
title: String!
description: String
reviews: [Review!]!
avgRating: Float!
}
type Review {
id: ID!
rating: Int!
comment: String
}
type Movie {
id: ID!
reviews: [Review!]!
avgRating: Float!
}

How to Apollo Federation?

Setup

Movie service

Review service

Compose Subgraph

Gateway

Setup

Movie service

Review service

Compose Subgraph

Gateway

Setup Database

EXPLORER

OPEN EDITORS
- schema.prisma prisma
- .env
- package.json M

APOLLO-FEDERATION-101
- gateway
  - index.ts
- generated
- lib
- node_modules
- prisma
  - schema.prisma
- services
- .env
- .env.example
- .gitignore
- .prettierrc.json
- codegen.yml
- docker-compose.yml
- index.ts
- package.json M
- supergraph-config.yml
- supergraph.graphql
- tsconfig.json
- yarn-error.log
- yarn.lock

OUTLINE
TIMELINE
NPM SCRIPTS

schema.prisma | .env | package.json M

package.json > { } devDependencies

```
    "@apollo/rover": "^0.1.10",
    "@graphql-codegen/cli": "^2.0.1",
    "@graphql-codegen/typescript": "^2.0.0",
    "@graphql-codegen/typescript-resolvers": "^2.0.0",
    "dotenv": "^10.0.0",
    "dotenv-cli": "^4.0.0",
    "prisma": "^2.29.1",
    "ts-node": "^10.2.0",      You, 15 hours ago • chore: initial commit
    "typescript": "^4.3.5"
  },
  Debug
  "scripts": {
    "start": "ts-node index.ts",
    "gateway:start": "ts-node gateway/index.ts",
    "pregateway:start": "yarn supergraph",
    "db:migrate": "dotenv -- yarn prisma migrate dev --preview-feature",
    "db:migrate:reset": "dotenv -- yarn prisma migrate reset",
    "codegen:graphql": "graphql-codegen",
    "codegen:prisma": "prisma generate",
    "codegen": "yarn codegen:graphql && yarn codegen:prisma",
    "supergraph": "rover supergraph compose --config ./supergraph-config.yml > supergraph.graphql"
  }
}
```

TERMINAL | PROBLEMS 11 | OUTPUT | DEBUG CONSOLE

zsh

~/Projects/apollo-federation-101 master !1

master* | 0 0 11 | Live Share | json | package.json | You, 15 hours ago | Ln 24, Col 26 | Spaces: 2 | UTF-8 | LF | JSON | Prettier

Setup

Movie service

Review service

Compose Subgraph

Gateway

Setup

Movie service

Review service

Compose Subgraph

Gateway

Schema

services/movie/schema.ts

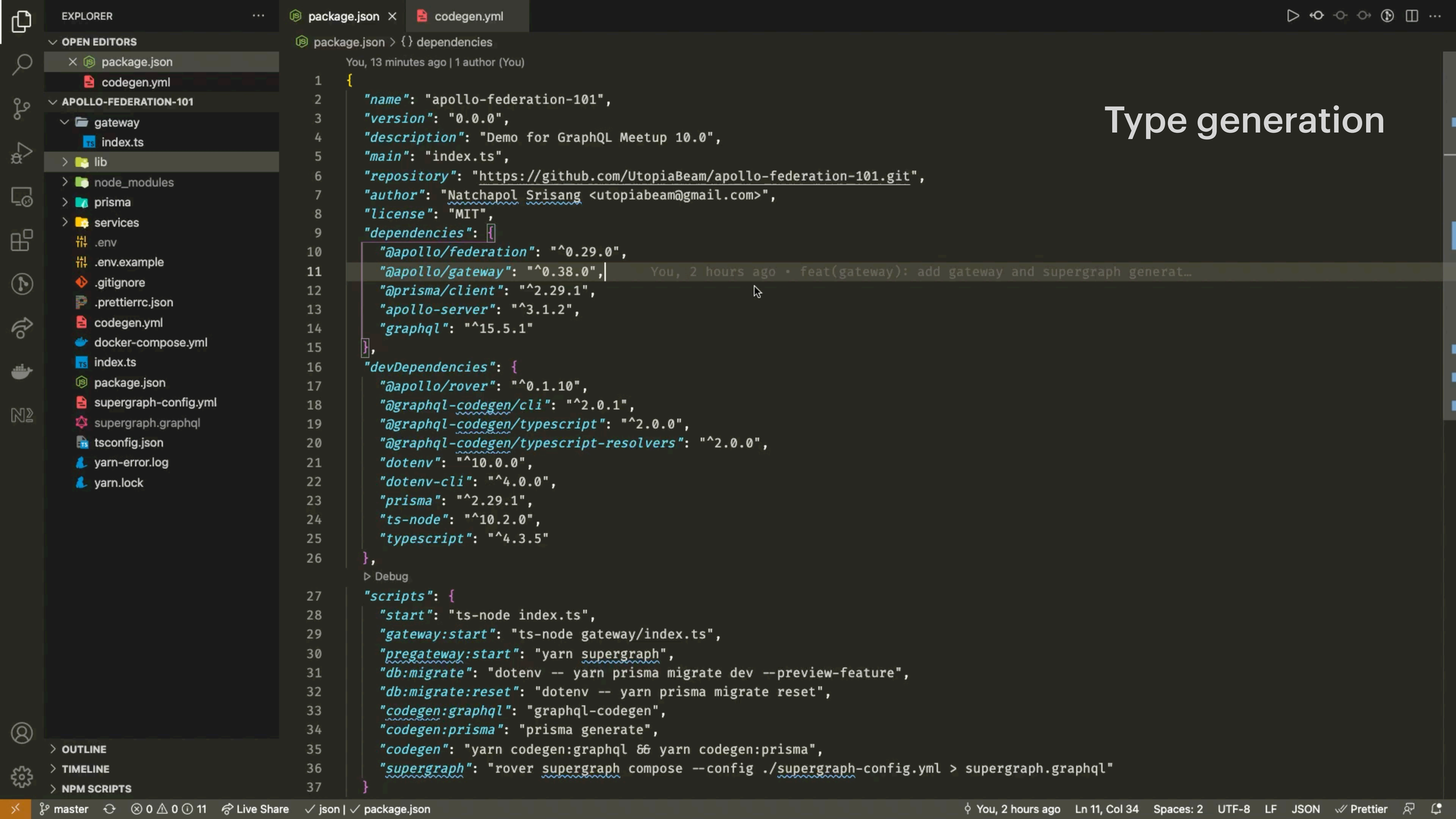
```
export const typeDefs = gql`
  type Movie @key(fields: "id") {
    id: ID!
    title: String!
    description: String
  }

  input CreateMovieInput {
    title: String!
    description: String
  }
```

Schema

services/movie/schema.ts

```
export const typeDefs = gql`
  type Movie @key(fields: "id") {
    id: ID!
    title: String!
    description: String
  }

  input CreateMovieInput {
    title: String!
    description: String
  }
```

Type generation



Server

services/movie/index.ts

```
import { buildSubgraphSchema } from '@apollo/federation'
import { typeDefs } from './schema'
import { resolvers } from './resolvers'
import { ApolloServer } from 'apollo-server'
import { createContext } from '../../lib/context'

export async function createMovieService(): Promise<void> {
  const schema = buildSubgraphSchema({ typeDefs, resolvers })
  const server = new ApolloServer({
    schema,
    context: ({ req }) => createContext(req),
  })

  const { url } = await server.listen(4001)

  console.log(`Movie service running at ${url}`)
}
```

Server

services/movie/index.ts

```
import { buildSubgraphSchema } from '@apollo/federation'
import { typeDefs } from './schema'
import { resolvers } from './resolvers'
import { ApolloServer } from 'apollo-server'
import { createContext } from '../../lib/context'

export async function createMovieService(): Promise<void> {
  const schema = buildSubgraphSchema({ typeDefs, resolvers })
  const server = new ApolloServer({
    schema,
    context: ({ req }) => createContext(req),
  })

  const { url } = await server.listen(4001)

  console.log(`Movie service running at ${url}`)
}
```

Server

services/movie/index.ts

```
import { buildSubgraphSchema } from '@apollo/federation'
import { typeDefs } from './schema'
import { resolvers } from './resolvers'
import { ApolloServer } from 'apollo-server'
import { createContext } from '../../lib/context'

export async function createMovieService(): Promise<void> {
  const schema = buildSubgraphSchema({ typeDefs, resolvers })
  const server = new ApolloServer({
    schema,
    context: ({ req }) => createContext(req),
  })

  const { url } = await server.listen(4001)

  console.log(`Movie service running at ${url}`)
}
```

Server

services/movie/index.ts

```
import { buildSubgraphSchema } from '@apollo/federation'
import { typeDefs } from './schema'
import { resolvers } from './resolvers'
import { ApolloServer } from 'apollo-server'
import { createContext } from '../../lib/context'

export async function createMovieService(): Promise<void> {
  const schema = buildSubgraphSchema({ typeDefs, resolvers })
  const server = new ApolloServer({
    schema,
    context: ({ req }) => createContext(req),
  })

  const { url } = await server.listen(4001)

  console.log(`Movie service running at ${url}`)
}
```

Server

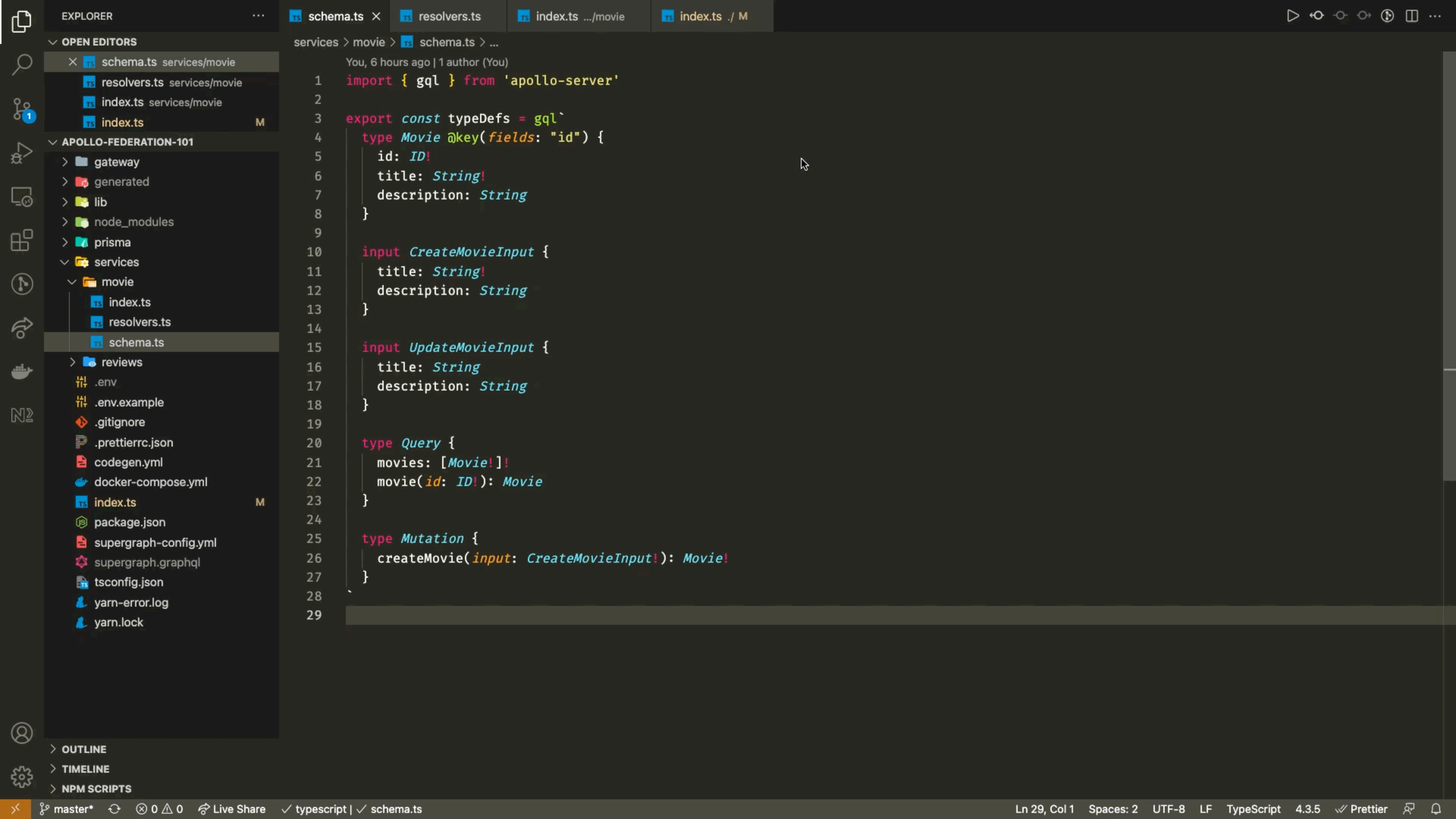
services/movie/index.ts

```
import { buildSubgraphSchema } from '@apollo/federation'
import { typeDefs } from './schema'
import { resolvers } from './resolvers'
import { ApolloServer } from 'apollo-server'
import { createContext } from '../../lib/context'

export async function createMovieService(): Promise<void> {
  const schema = buildSubgraphSchema({ typeDefs, resolvers })
  const server = new ApolloServer({
    schema,
    context: ({ req }) => createContext(req),
  })

  const { url } = await server.listen(4001)

  console.log(`Movie service running at ${url}`)
}
```

Context

lib/context.ts

```ts
import { PrismaClient } from '../generated/prisma'
import { Request } from 'express'

export interface Context {
  prisma: PrismaClient
}

export function createContext(_req: Request) {
  return {
    prisma: new PrismaClient(),
  }
}
```


EXPLORER
OPEN EDITORS
schema.ts services/movie
resolvers.ts services/movie
index.ts services/movie
index.ts M
APOLLO-FEDERATION-101
gateway
generated
lib
node_modules
prisma
services
movie
index.ts
resolvers.ts
schema.ts
reviews
.env
.env.example
.gitignore
.prettierrc.json
codegen.yml
docker-compose.yml
index.ts M
package.json
supergraph-config.yml
supergraph.graphql
tsconfig.json
yarn-error.log
yarn.lock
OUTLINE
TIMELINE
NPM SCRIPTS
schema.ts
resolvers.ts
index.ts .../movie
index.ts ./ M
services > movie > schema.ts > ...
You, 6 hours ago | 1 author (You)
import { gql } from 'apollo-server'
export const typeDefs = gql`
type Movie @key(fields: "id") {
id: ID!
title: String!
description: String
}
input CreateMovieInput {
title: String!
description: String
}
input UpdateMovieInput {
title: String
description: String
}
type Query {
movies: [Movie!]!
movie(id: ID!): Movie
}
type Mutation {
createMovie(input: CreateMovieInput!): Movie!
}
`
master*
0 0
Live Share
typescript | schema.ts
Ln 29, Col 1
Spaces: 2
UTF-8
LF
TypeScript
4.3.5
Prettier

Setup

Movie service

Review service

Compose Subgraph

Gateway

Setup

Movie service

Review service

Compose Subgraph

Gateway

Schema

services/review/schema.ts

```
export const typeDefs = gql`
  type Review {
    id: ID!
    rating: Int!
    comment: String
    movie: Movie!
  }

  extend type Movie @key(fields: "id") {
    id: ID! @external
    reviews: [Review!]!
    avgRating: Float!
  }
```

Schema

services/review/schema.ts

```
export const typeDefs = gql`
  type Review {
    id: ID!
    rating: Int!
    comment: String
    movie: Movie!
  }

  extend type Movie @key(fields: "id") {
    id: ID! @external
    reviews: [Review!]!
    avgRating: Float!
  }
```

Schema

services/review/schema.ts

```
export const typeDefs = gql`
  type Review {
    id: ID!
    rating: Int!
    comment: String
    movie: Movie!
  }

  extend type Movie @key(fields: "id") {
    id: ID! @external
    reviews: [Review!]!
    avgRating: Float!
  }
```

Schema

services/review/schema.ts

```
export const typeDefs = gql`
  type Review {
    id: ID!
    rating: Int!
    comment: String
    movie: Movie!
  }

  extend type Movie @key(fields: "id") {
    id: ID! @external
    reviews: [Review!]!
    avgRating: Float!
  }
```

Resolvers

services/review/resolvers.ts

```ts
export const resolvers: Resolvers<Context> = {
  Query: { ... },
  Mutation: { ... },
  Movie: {
    avgRating: async (parent, _args, ctx) => {
      ...
    },
    reviews: async (parent, _args, ctx) => {
      ...
    },
  },
  Review: {
    movie: (parent, _args, _ctx) => {
      return { __typename: 'Movie', id: parent.id } as Movie
    },
  },
}
```

Resolvers

services/review/resolvers.ts

```
export const resolvers: Resolvers<Context> = {
  Query: { ... },
  Mutation: { ... },
  Movie: {
    avgRating: async (parent, _args, ctx) => {
      ...
    },
    reviews: async (parent, _args, ctx) => {
      ...
    },
  },
  Review: {
    movie: (parent, _args, _ctx) => {
      return { __typename: 'Movie', id: parent.id } as Movie
    },
  },
}
```

Resolvers

services/review/resolvers.ts

```
export const resolvers: Resolvers<Context> = {
  Query: { ... },
  Mutation: { ... },
  Movie: {
    avgRating: async (parent, _args, ctx) => {
      ...
    },
    reviews: async (parent, _args, ctx) => {
      ...
    },
  },
  Review: {
    movie: (parent, _args, _ctx) => {
      return { __typename: 'Movie', id: parent.id } as Movie
    },
  },
```

Resolvers

services/review/resolvers.ts

```
export const resolvers: Resolvers<Context> = {
  Query: { ... },
  Mutation: { ... },
  Movie: {
    avgRating: async (parent, _args, ctx) => {
      ...
    },
    reviews: async (parent, _args, ctx) => {
      ...
    },
  },
  Review: {
    movie: (parent, _args, _ctx) => {
      return { __typename: 'Movie', id: parent.id } as Movie
    },
  },
```

Resolvers

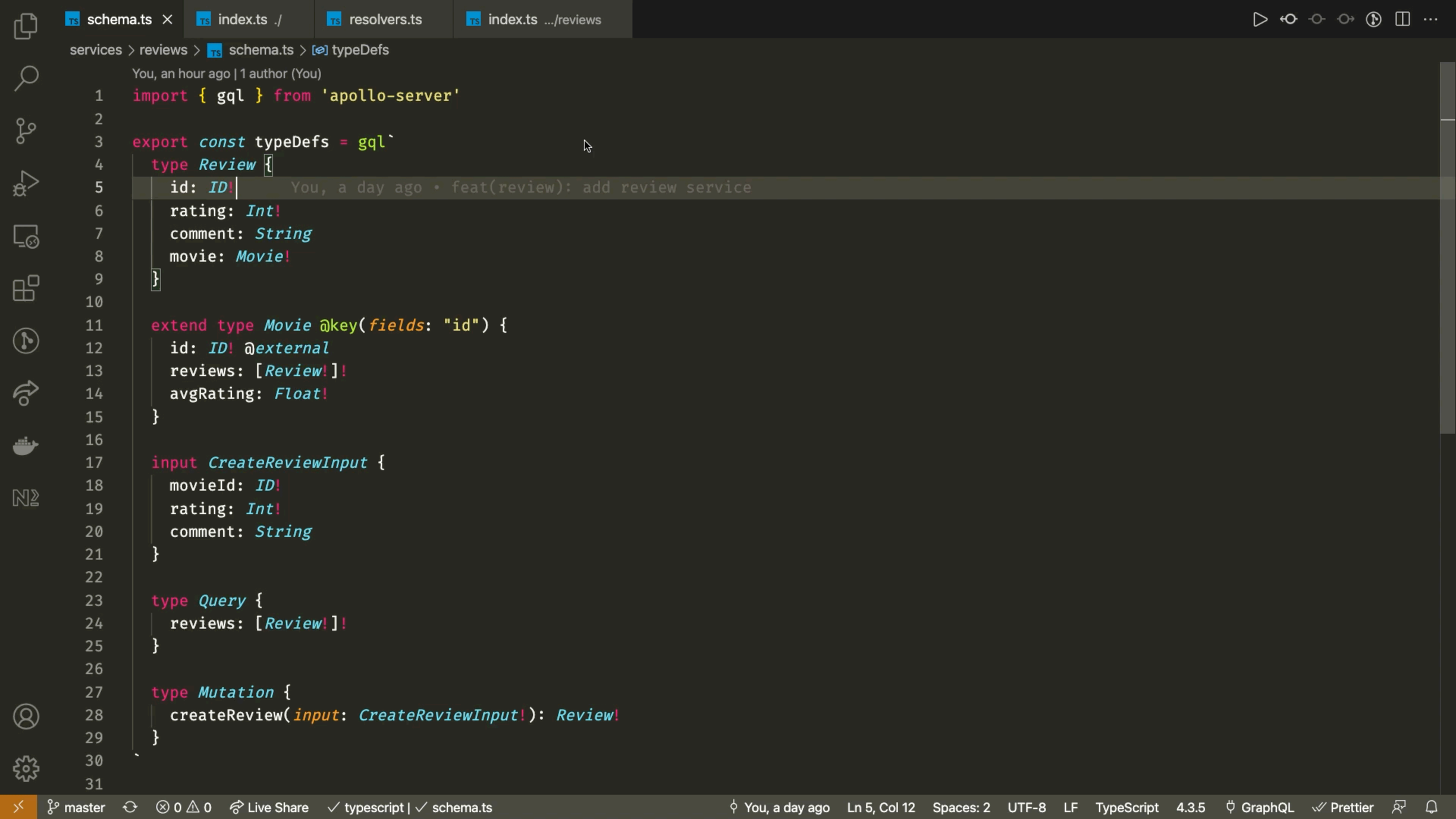
services/movie/resolvers.ts

```ts
export const resolvers: Resolvers<Context> = {
  Query: { ... },
  Mutation: { ... },
  Movie: {
    __resolveReference: async (parent, ctx) => {
      return ctx.prisma.movie.findUnique({
        where: { id: parent.id },
      }) as unknown as Promise<Movie>
    },
  },
}
```

Resolvers

services/movie/resolvers.ts

```
export const resolvers: Resolvers<Context> = {
  Query: { ... },
  Mutation: { ... },
  Movie: {
    __resolveReference: async (parent, ctx) => {
      return ctx.prisma.movie.findUnique({
        where: { id: parent.id },
      }) as unknown as Promise<Movie>
    },
  },
}
```


schema.ts
index.ts ./
resolvers.ts
index.ts .../reviews
services > reviews > schema.ts > typeDefs
You, an hour ago | 1 author (You)
import { gql } from 'apollo-server'
export const typeDefs = gql`
type Review {
id: ID! You, a day ago • feat(review): add review service
rating: Int!
comment: String
movie: Movie!
}
extend type Movie @key(fields: "id") {
id: ID! @external
reviews: [Review!]!
avgRating: Float!
}
input CreateReviewInput {
movieId: ID!
rating: Int!
comment: String
}
type Query {
reviews: [Review!]!
}
type Mutation {
createReview(input: CreateReviewInput!): Review!
}
`
master
0 0
Live Share
typescript | schema.ts
You, a day ago
Ln 5, Col 12
Spaces: 2
UTF-8
LF
TypeScript
4.3.5
GraphQL
Prettier

Setup

Movie service

Review service

Compose Subgraph

Gateway

Setup

Movie service

Review service

Compose Subgraph

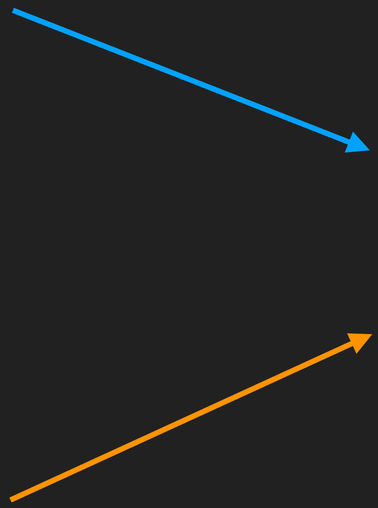
Gateway

```
type Movie @key(fields: "id") {
  id: ID!
  title: String!
  description: String
}
```

```
type Review {
  id: ID!
  rating: Int!
  comment: String
}

extend type Movie @key(fields: "id") {
  id: ID! @external
  reviews: [Review!]!
  avgRating: Float!
}
```

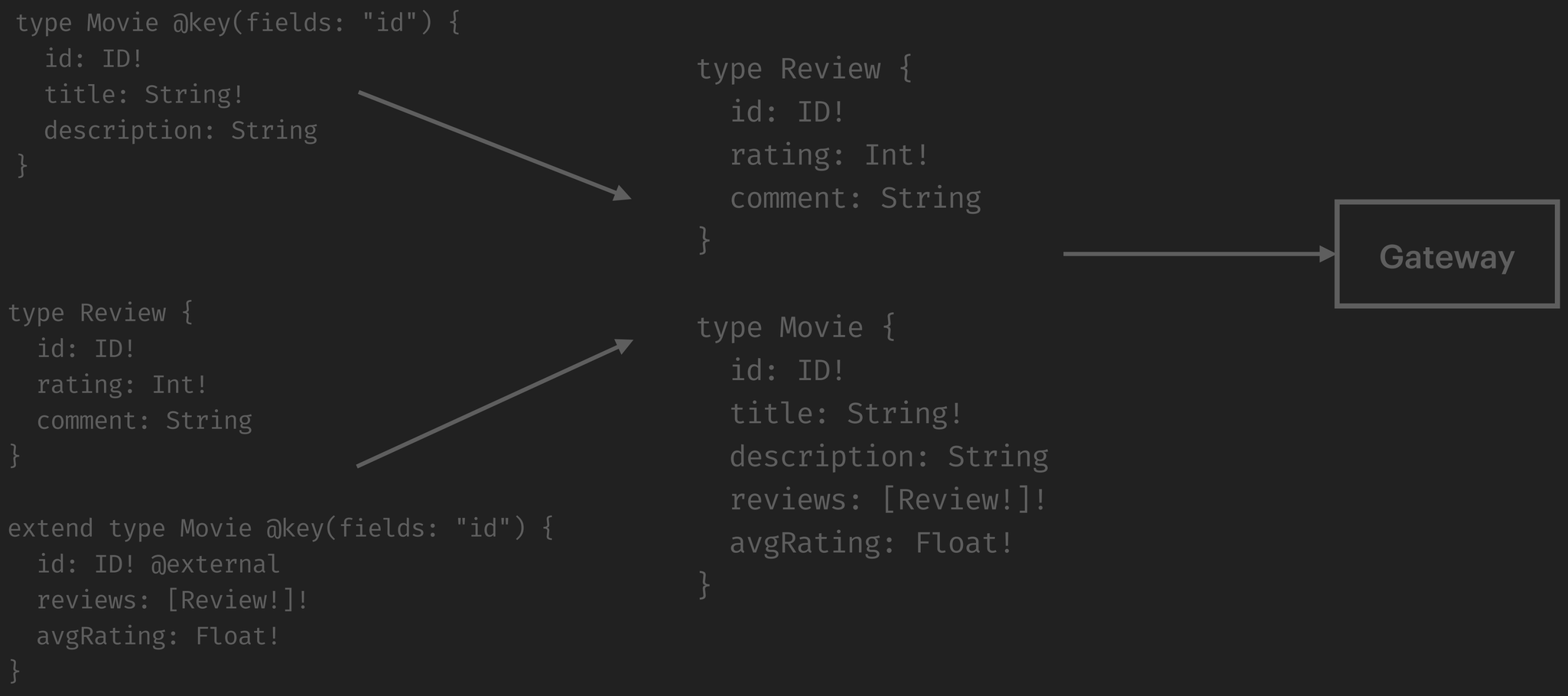
```
type Movie @key(fields: "id") {
  id: ID!
  title: String!
  description: String
}
```

```
type Review {
  id: ID!
  rating: Int!
  comment: String
}

extend type Movie @key(fields: "id") {
  id: ID! @external
  reviews: [Review!]!
  avgRating: Float!
}
```




```
type Review {
  id: ID!
  rating: Int!
  comment: String
}

type Movie {
  id: ID!
  title: String!
  description: String
  reviews: [Review!]!
  avgRating: Float!
}
```

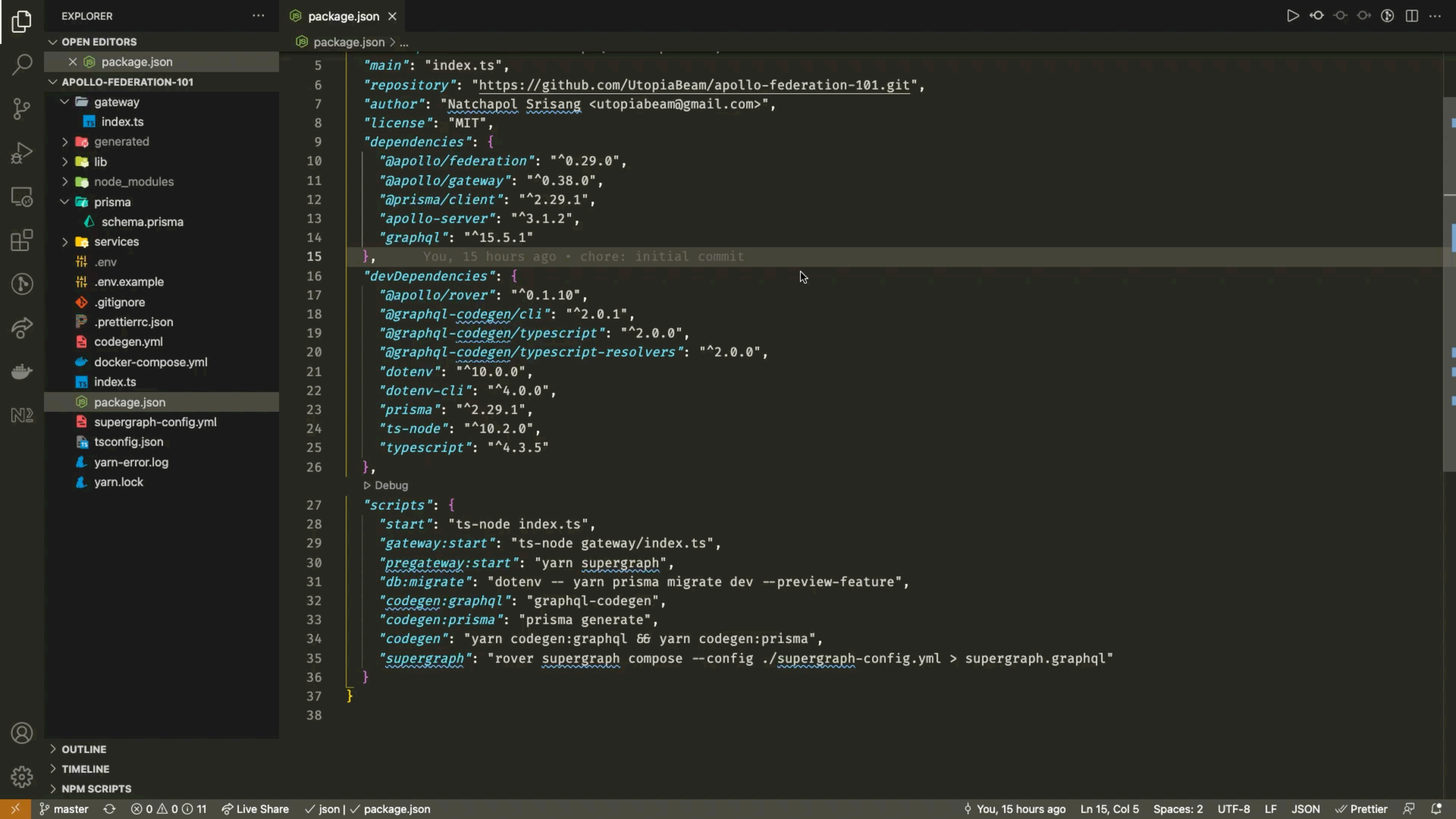

Gateway




```
rover supergraph compose --config ./supergraph-config.yml > supergraph.graphql
```

subgraphl-config.yml

```yaml
subgraphs:
  movies:
    routing_url: http://localhost:4001/graphql
    schema:
      subgraph_url: http://localhost:4001/graphql
  reviews:
    routing_url: http://localhost:4002/graphql
    schema:
      subgraph_url: http://localhost:4002/graphql
```


EXPLORER
package.json
OPEN EDITORS
package.json
APOLLO-FEDERATION-101
gateway
index.ts
generated
lib
node_modules
prisma
schema.prisma
services
.env
.env.example
.gitignore
.prettierrc.json
codegen.yml
docker-compose.yml
index.ts
package.json
supergraph-config.yml
tsconfig.json
yarn-error.log
yarn.lock
OUTLINE
TIMELINE
NPM SCRIPTS
package.json > ...
"main": "index.ts",
"repository": "https://github.com/UtopiaBeam/apollo-federation-101.git",
"author": "Natchapol Srisang <utopiabeam@gmail.com>",
"license": "MIT",
"dependencies": {
"@apollo/federation": "^0.29.0",
"@apollo/gateway": "^0.38.0",
"@prisma/client": "^2.29.1",
"apollo-server": "^3.1.2",
"graphql": "^15.5.1"
},
You, 15 hours ago • chore: initial commit
"devDependencies": {
"@apollo/rover": "^0.1.10",
"@graphql-codegen/cli": "^2.0.1",
"@graphql-codegen/typescript": "^2.0.0",
"@graphql-codegen/typescript-resolvers": "^2.0.0",
"dotenv": "^10.0.0",
"dotenv-cli": "^4.0.0",
"prisma": "^2.29.1",
"ts-node": "^10.2.0",
"typescript": "^4.3.5"
},
Debug
"scripts": {
"start": "ts-node index.ts",
"gateway:start": "ts-node gateway/index.ts",
"pregateway:start": "yarn supergraph",
"db:migrate": "dotenv -- yarn prisma migrate dev --preview-feature",
"codegen:graphql": "graphql-codegen",
"codegen:prisma": "prisma generate",
"codegen": "yarn codegen:graphql && yarn codegen:prisma",
"supergraph": "rover supergraph compose --config ./supergraph-config.yml > supergraph.graphql"
}
}
master
0 0 11
Live Share
json | package.json
You, 15 hours ago
Ln 15, Col 5
Spaces: 2
UTF-8
LF
JSON
Prettier

supergraph.graphql

```
enum join__Graph {
  MOVIES @join__graph(name: "movies" url: "http://localhost:4001/graphql")
  REVIEWS @join__graph(name: "reviews" url: "http://localhost:4002/graphql")
}

type Movie
  @join__owner(graph: MOVIES)
  @join__type(graph: MOVIES, key: "id")
  @join__type(graph: REVIEWS, key: "id")
{
  avgRating: Float! @join__field(graph: REVIEWS)
  description: String @join__field(graph: MOVIES)
  id: ID! @join__field(graph: MOVIES)
  reviews: [Review!]! @join__field(graph: REVIEWS)
  title: String! @join__field(graph: MOVIES)
}
```

supergraph.graphql

```
enum join__Graph {
  MOVIES @join__graph(name: "movies" url: "http://localhost:4001/graphql")
  REVIEWS @join__graph(name: "reviews" url: "http://localhost:4002/graphql")
}

type Movie
  @join__owner(graph: MOVIES)
  @join__type(graph: MOVIES, key: "id")
  @join__type(graph: REVIEWS, key: "id")
{
  avgRating: Float! @join__field(graph: REVIEWS)
  description: String @join__field(graph: MOVIES)
  id: ID! @join__field(graph: MOVIES)
  reviews: [Review!]! @join__field(graph: REVIEWS)
  title: String! @join__field(graph: MOVIES)
}
```

supergraph.graphql

```
enum join__Graph {
  MOVIES @join__graph(name: "movies" url: "http://localhost:4001/graphql")
  REVIEWS @join__graph(name: "reviews" url: "http://localhost:4002/graphql")
}

type Movie
  @join__owner(graph: MOVIES)
  @join__type(graph: MOVIES, key: "id")
  @join__type(graph: REVIEWS, key: "id")
{
  avgRating: Float! @join__field(graph: REVIEWS)
  description: String @join__field(graph: MOVIES)
  id: ID! @join__field(graph: MOVIES)
  reviews: [Review!]! @join__field(graph: REVIEWS)
  title: String! @join__field(graph: MOVIES)
}
```

supergraph.graphql

```
enum join__Graph {
  MOVIES @join__graph(name: "movies" url: "http://localhost:4001/graphql")
  REVIEWS @join__graph(name: "reviews" url: "http://localhost:4002/graphql")
}

type Movie
  @join__owner(graph: MOVIES)
  @join__type(graph: MOVIES, key: "id")
  @join__type(graph: REVIEWS, key: "id")
{
  avgRating: Float! @join__field(graph: REVIEWS)
  description: String @join__field(graph: MOVIES)
  id: ID! @join__field(graph: MOVIES)
  reviews: [Review!]! @join__field(graph: REVIEWS)
  title: String! @join__field(graph: MOVIES)
}
```

supergraph.graphql

```
enum join__Graph {
  MOVIES @join__graph(name: "movies" url: "http://localhost:4001/graphql")
  REVIEWS @join__graph(name: "reviews" url: "http://localhost:4002/graphql")
}

type Movie
  @join__owner(graph: MOVIES)
  @join__type(graph: MOVIES, key: "id")
  @join__type(graph: REVIEWS, key: "id")
{
  avgRating: Float! @join__field(graph: REVIEWS)
  description: String @join__field(graph: MOVIES)
  id: ID! @join__field(graph: MOVIES)
  reviews: [Review!]! @join__field(graph: REVIEWS)
  title: String! @join__field(graph: MOVIES)
}
```

supergraph.graphql

```
enum join__Graph {
  MOVIES @join__graph(name: "movies" url: "http://localhost:4001/graphql")
  REVIEWS @join__graph(name: "reviews" url: "http://localhost:4002/graphql")
}

type Movie
  @join__owner(graph: MOVIES)
  @join__type(graph: MOVIES, key: "id")
  @join__type(graph: REVIEWS, key: "id")
{
  avgRating: Float! @join__field(graph: REVIEWS)
  description: String @join__field(graph: MOVIES)
  id: ID! @join__field(graph: MOVIES)
  reviews: [Review!]! @join__field(graph: REVIEWS)
  title: String! @join__field(graph: MOVIES)
}
```

Setup

Movie service

Review service

Compose Subgraph

Gateway

Setup

Movie service

Review service

Compose Subgraph

Gateway

gateway/index.ts

```
export async function createGateway(): Promise<void> {
  const supergraphSdl = readFileSync(
    resolve(__dirname, '../supergraph.graphql')
  ).toString()

  const gateway = new ApolloGateway({
    supergraphSdl,
  })
  const server = new ApolloServer({
    gateway,
  })

  const { url } = await server.listen(4000)

  console.log(`🚀 Gateway ready at ${url}`)
}
```

gateway/index.ts

```
export async function createGateway(): Promise<void> {
  const supergraphSdl = readFileSync(
    resolve(__dirname, '../supergraph.graphql')
  ).toString()

  const gateway = new ApolloGateway({
    supergraphSdl,
  })
  const server = new ApolloServer({
    gateway,
  })

  const { url } = await server.listen(4000)

  console.log(`🚀 Gateway ready at ${url}`)
}
```

gateway/index.ts

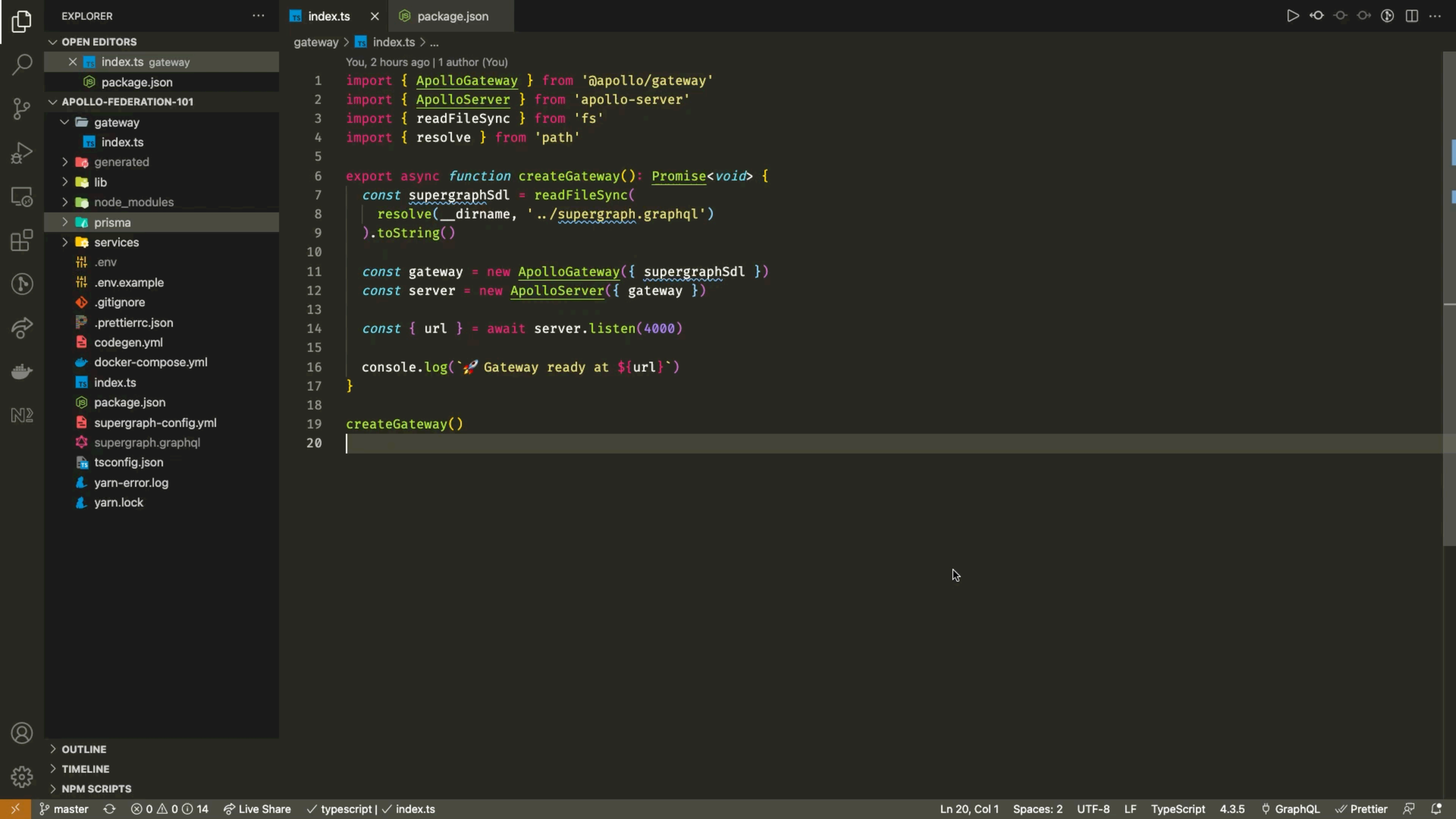
```
export async function createGateway(): Promise<void> {
  const supergraphSdl = readFileSync(
    resolve(__dirname, '../supergraph.graphql')
  ).toString()

  const gateway = new ApolloGateway({
    supergraphSdl,
  })
  const server = new ApolloServer({
    gateway,
  })

  const { url } = await server.listen(4000)

  console.log(`🚀 Gateway ready at ${url}`)
}
```

gateway/index.ts

```
export async function createGateway(): Promise<void> {
  const supergraphSdl = readFileSync(
    resolve(__dirname, '../supergraph.graphql')
  ).toString()

  const gateway = new ApolloGateway({
    supergraphSdl,
  })
  const server = new ApolloServer({
    gateway,
  })

  const { url } = await server.listen(4000)

  console.log(`🚀 Gateway ready at ${url}`)
}
```

gateway/index.ts

```
export async function createGateway(): Promise<void> {
  const supergraphSdl = readFileSync(
    resolve(__dirname, '../supergraph.graphql')
  ).toString()

  const gateway = new ApolloGateway({
    supergraphSdl,
  })
  const server = new ApolloServer({
    gateway,
  })

  const { url } = await server.listen(4000)

  console.log(`🚀 Gateway ready at ${url}`)
}
```


EXPLORER
OPEN EDITORS
index.ts gateway
package.json
APOLLO-FEDERATION-101
gateway
index.ts
generated
lib
node_modules
prisma
services
.env
.env.example
.gitignore
.prettierrc.json
codegen.yml
docker-compose.yml
index.ts
package.json
supergraph-config.yml
supergraph.graphql
tsconfig.json
yarn-error.log
yarn.lock
OUTLINE
TIMELINE
NPM SCRIPTS
index.ts
package.json
gateway > index.ts > ...
You, 2 hours ago | 1 author (You)
import { ApolloGateway } from '@apollo/gateway'
import { ApolloServer } from 'apollo-server'
import { readFileSync } from 'fs'
import { resolve } from 'path'
export async function createGateway(): Promise<void> {
const supergraphSdl = readFileSync(
resolve(__dirname, '../supergraph.graphql')
).toString()
const gateway = new ApolloGateway({ supergraphSdl })
const server = new ApolloServer({ gateway })
const { url } = await server.listen(4000)
console.log(`🚀 Gateway ready at ${url}`)
}
createGateway()
master
0 0 14
Live Share
typescript | index.ts
Ln 20, Col 1
Spaces: 2
UTF-8
LF
TypeScript
4.3.5
GraphQL
Prettier


CONGRATULATIONS.
NOW GET BACK TO WORK
memegenerator.net

Quiz no.3

Authentication/ Authorization

gateway/index.ts

```
export async function createGateway(): Promise<void> {
  const supergraphSdl = readFileSync(
    resolve(__dirname, '../supergraph.graphql')
  ).toString()

  const gateway = new ApolloGateway({
    supergraphSdl,
    buildService: ({ url }) => new AuthenticationSource({ url }),
  })
  const server = new ApolloServer({
    gateway,
    context: ({ req }) => ({
      authorization: req.headers.authorization,
    }),
  })

  const { url } = await server.listen(4000)
```

gateway/index.ts

```
export async function createGateway(): Promise<void> {
  const supergraphSdl = readFileSync(
    resolve(__dirname, '../supergraph.graphql')
  ).toString()

  const gateway = new ApolloGateway({
    supergraphSdl,
    buildService: ({ url }) => new AuthenticationSource({ url }),
  })
  const server = new ApolloServer({
    gateway,
    context: ({ req }) => ({
      authorization: req.headers.authorization,
    }),
  })

  const { url } = await server.listen(4000)
```

Authentication source

gateway/authentication-source.ts

```ts
import { RemoteGraphQLDataSource } from '@apollo/gateway'

export class AuthenticationSource extends RemoteGraphQLDataSource {
  willSendRequest({ request, context }: any) {
    if (context.authorization) {
      request.http.headers.set('authorization', context.authorization)
    }
  }
}
```

Authentication source

gateway/authentication-source.ts

```
import { RemoteGraphQLDataSource } from '@apollo/gateway'

export class AuthenticationSource extends RemoteGraphQLDataSource {
  willSendRequest({ request, context }: any) {
    if (context.authorization) {
      request.http.headers.set('authorization', context.authorization)
    }
  }
}
```

gateway/index.ts

```
export async function createGateway(): Promise<void> {
  const supergraphSdl = readFileSync(
    resolve(__dirname, '../supergraph.graphql')
  ).toString()

  const gateway = new ApolloGateway({
    supergraphSdl,
    buildService: ({ url }) => new AuthenticationSource({ url }),
  })
  const server = new ApolloServer({
    gateway,
    context: ({ req }) => ({
      authorization: req.headers.authorization,
    }),
  })

  const { url } = await server.listen(4000)
```

Authentication source

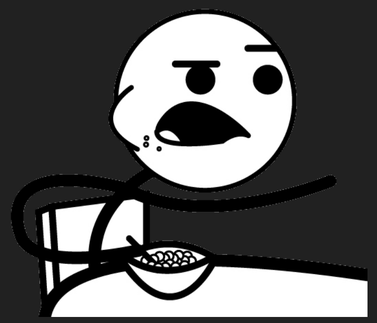
gateway/authentication-source.ts

```
import { RemoteGraphQLDataSource } from '@apollo/gateway'

export class AuthenticationSource extends RemoteGraphQLDataSource {
  willSendRequest({ request, context }: any) {
    if (context.authorization) {
      request.http.headers.set('authorization', context.authorization)
    }
  }
}
```

Authentication source

gateway/authentication-source.ts

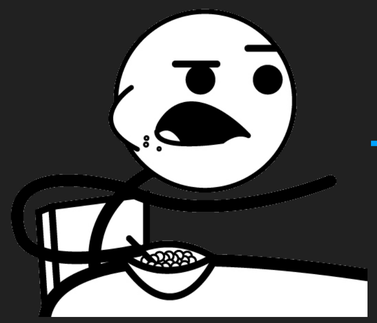
```
import { RemoteGraphQLDataSource } from '@apollo/gateway'

export class AuthenticationSource extends RemoteGraphQLDataSource {
  willSendRequest({ request, context }: any) {
    if (context.authorization) {
      request.http.headers.set('authorization', context.authorization)
    }
  }
}
```



Gateway

Movie

Review

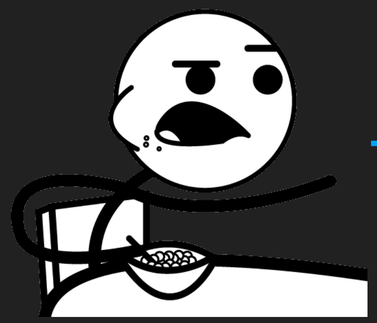


`authorization: 'token'`

Gateway

Movie

Review



authorization: 'token'

Gateway

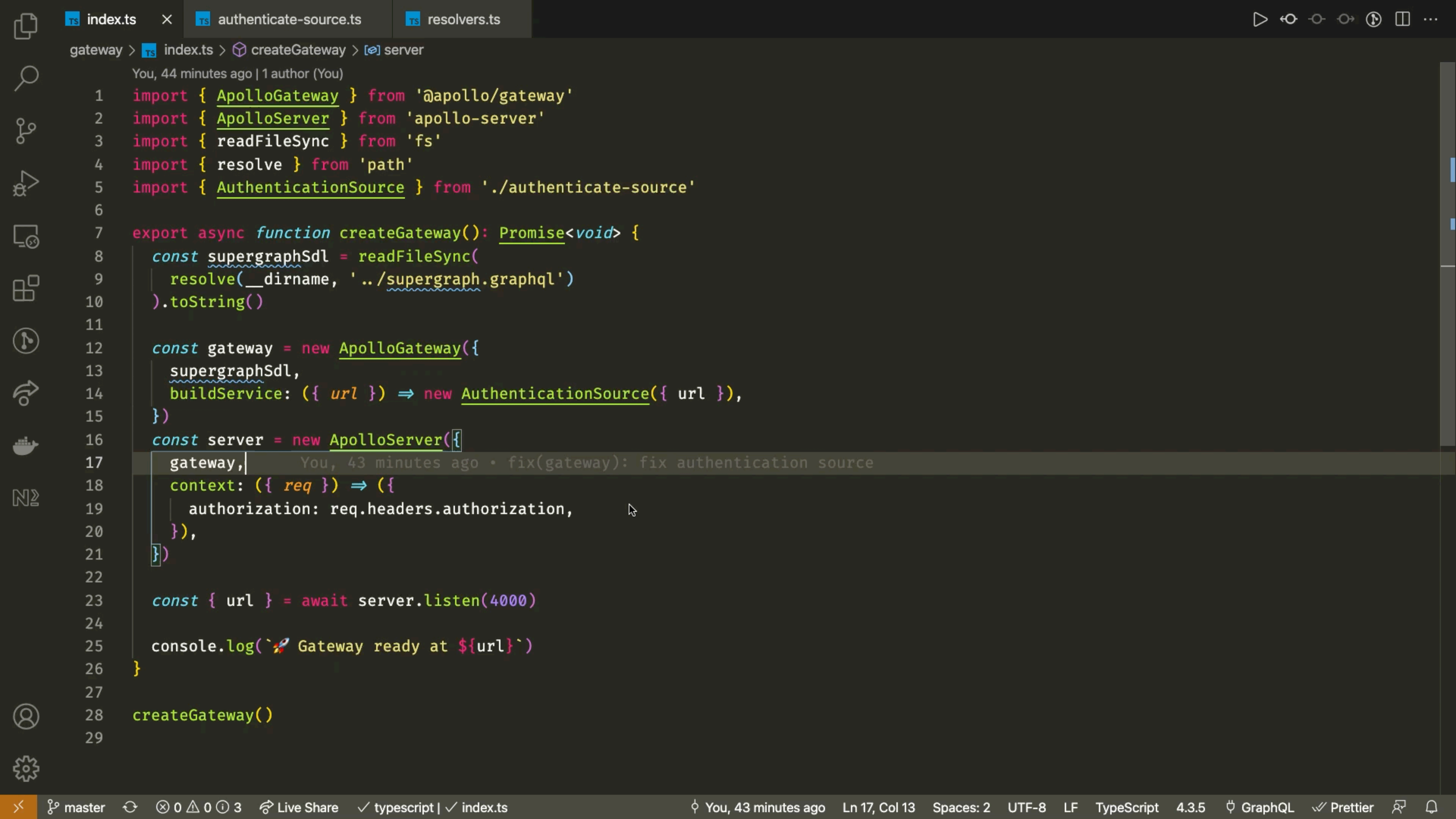
authorization: 'token'

Movie

Review

services/movie/resolvers.ts

```ts
export const resolvers: Resolvers<Context> = {
  Query: {
    movies: async (_parent, _args, ctx) => {
      if (!ctx.isAuthenticated) {
        throw new Error('Unauthenticated')
      }
      return ctx.prisma.movie.findMany() as unknown as Promise<Movie[]>
    },
    movie: (_parent, { id }, ctx) => {
      return ctx.prisma.movie.findUnique({
        where: { id },
      }) as Promise<Movie | null>
    },
  },
  Mutation: {
    createMovie: async (_parent, { input }, ctx) => {
      return ctx.prisma.movie.create({
```


index.ts
authenticate-source.ts
resolvers.ts
gateway > index.ts > createGateway > server
You, 44 minutes ago | 1 author (You)
import { ApolloGateway } from '@apollo/gateway'
import { ApolloServer } from 'apollo-server'
import { readFileSync } from 'fs'
import { resolve } from 'path'
import { AuthenticationSource } from './authenticate-source'
export async function createGateway(): Promise<void> {
const supergraphSdl = readFileSync(
resolve(__dirname, '../supergraph.graphql')
).toString()
const gateway = new ApolloGateway({
supergraphSdl,
buildService: ({ url }) => new AuthenticationSource({ url }),
})
const server = new ApolloServer({
gateway,
You, 43 minutes ago • fix(gateway): fix authentication source
context: ({ req }) => ({
authorization: req.headers.authorization,
}),
})
const { url } = await server.listen(4000)
console.log(`🚀 Gateway ready at ${url}`)
}
createGateway()
master
0 0 3
Live Share
typescript | index.ts
You, 43 minutes ago
Ln 17, Col 13
Spaces: 2
UTF-8
LF
TypeScript
4.3.5
GraphQL
Prettier

What if we want real-time reviews?

Subscription!


Implement Subscription
in Apollo Federation
WE DON'T DO THAT HERE
imgflip.com

https://www.youtube.com/watch?v=C5pSwLpjZM0

federation-subscription-tools

0.1.0 • Public • Published 4 months ago

Readme | Explore BETA | 7 Dependencies | 0 Dependents | 1 Versions

Using Subscriptions with a Federated Data Graph

This demonstration library shows how a decoupled subscription service can run alongside a federated data graph to provide real-time updates to a client. While the subscription service runs a separate non-federated Apollo Server, client applications do not need to perform any special handling of their subscription operations and may send those requests as they would to any GraphQL API that supports subscriptions. The subscription service's API may also specify return types for the `Subscription` fields that are defined in the federated data graph without explicitly redefining them in that service's type definitions.

In brief, the utilities contained within this library will allow you to:

- Create a decoupled, independently scalable subscriptions service to run alongside a unified data graph
- Use types defined in your unified data graph as return types within the subscriptions service's type definitions (without manually redefining those types in the subscriptions service)
- Publish messages to a shared pub/sub implementation from any subgraph service
- Allow clients to write subscription operations just as they would if the `Subscription` fields were defined directly within the unified data graph itself

Install

```
> npm i federation-subscription-tools
```

Weekly Downloads

10

Version
0.1.0

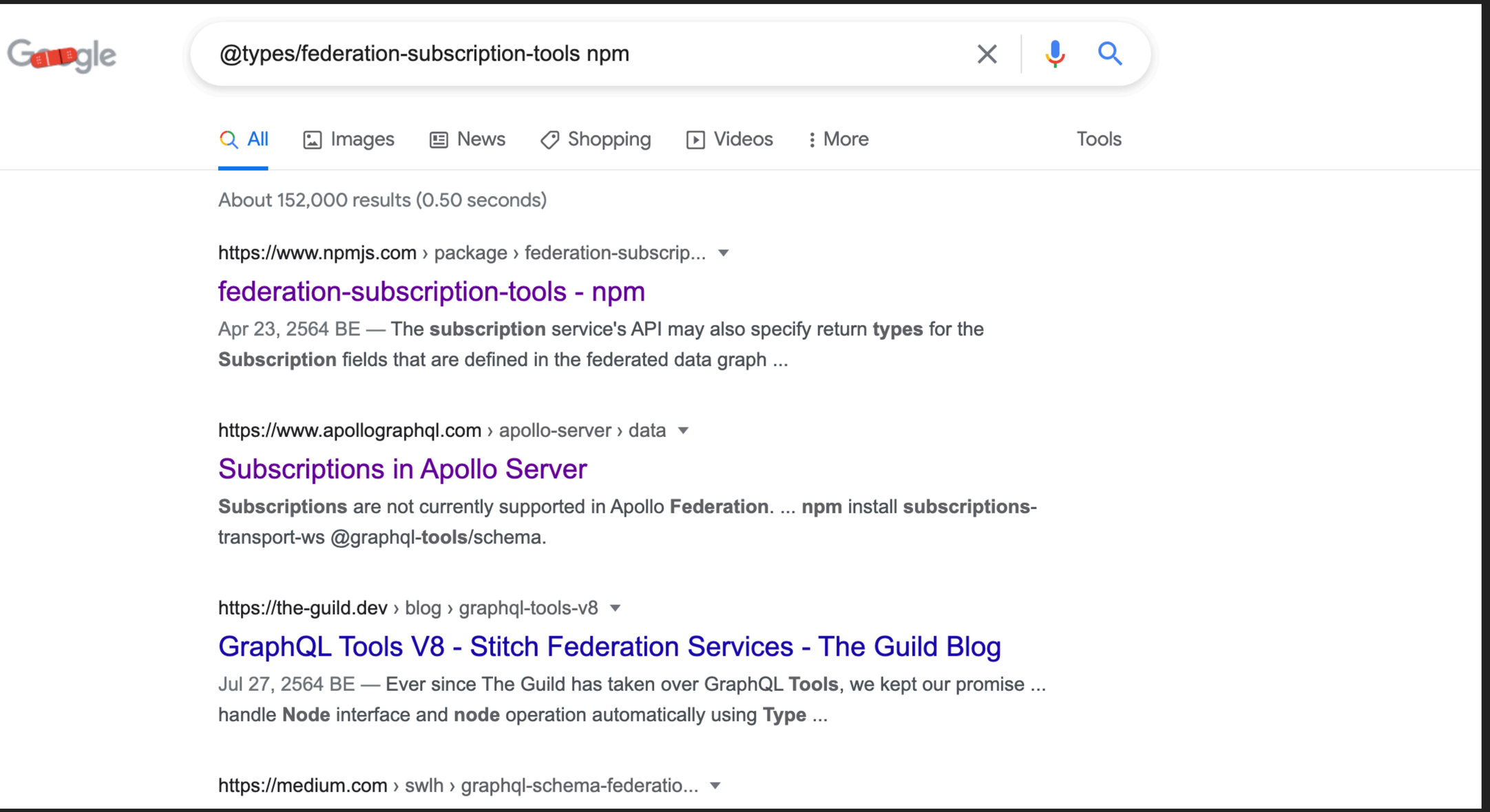
License
ISC

Unpacked Size
143 kB

Total Files
57

Last publish
4 months ago

Collaborators


@types/federation-subscription-tools npm
All
Images
News
Shopping
Videos
More
Tools
About 152,000 results (0.50 seconds)
https://www.npmjs.com › package › federation-subscrip...
federation-subscription-tools - npm
Apr 23, 2564 BE — The subscription service's API may also specify return types for the Subscription fields that are defined in the federated data graph ...
https://www.apollographql.com › apollo-server › data
Subscriptions in Apollo Server
Subscriptions are not currently supported in Apollo Federation. ... npm install subscriptions-transport-ws @graphql-tools/schema.
https://the-guild.dev › blog › graphql-tools-v8
GraphQL Tools V8 - Stitch Federation Services - The Guild Blog
Jul 27, 2564 BE — Ever since The Guild has taken over GraphQL Tools, we kept our promise ... handle Node interface and node operation automatically using Type ...
https://medium.com › swlh › graphql-schema-federatio...

Alternatives

19-20 August 2021

GraphQL Meetup 10.0

Serverless & Schema Stitching

Farhan Poh-Asae

Backend Engineer at BRIKL

Organized by brikl

Summary

What's Apollo Federation?

"Where GraphQL and Microservices combine beautifully"


Thank you

brikl.com/jobs