GraphQL Makes Your APIs Better

The Revolution in Efficiency and Real-time Performance
I’m Obsessed with GraphQL

Michael Watson
Developer Relations at Apollo

https://discord.gg/graphos
GraphQL Makes Your APIs Better
Welcome to the Apollo Showcase demo

This demo app provides a playground to test and learn about various Apollo features to understand how Apollo can be used in a moderately complex app.

Getting started

To use this app, you will need a Spotify account. This will allow the app to make calls to the Spotify API using this app's GraphQL API.

Running this app locally

If you would like to hook this up to your own application, you will first need a Spotify developer token. Once obtained, you will need to add these credentials to the app.

Create a Spotify application

First, visit the [Spotify developer portal](https://developer.spotify.com) and, if necessary, log in. This requires a Spotify user account.

Create and register a new application to generate credentials. Follow the [App settings guide](https://developer.spotify.com/) to learn more. We recommend this Spotify app is unique to this demo app.

In the `edit Settings` dialogue, add this app's redirect URL for this app to allow this app to sign into your Spotify account.

```
http://localhost:3000/match/finalize
```

Configure this application

Copy your app's `Client ID` and set it as the `VITE_SPOTIFY_CLIENT_ID` environment variable in `client/.env.development.local` file. This file should look as follows:

```
VITE_SPOTIFY_CLIENT_ID: [your-spotify-client-id]
```
The Initial Dream

- API

- Database

- Server
Then came mobile 🚀
But the mobile team needed something different…
But the mobile team needed something different…
What about vNext?
Add GraphQL to existing APIs
The Router

- Observability
- GraphQL for all
  anyone can write a query
- Real-time
- More efficient query execution
  Caching, query plans and @defer
- Security built in

https://www.apollographql.com/docs/router
Observability
Real-Time Data
Pushing the boundaries with Real-time

WebSockets

HTTP Callback
HTTP Callback Protocol

Incoming Subscription

Apollo Router

Register subscription and callback URL over HTTP
HTTP Callback Protocol

Register subscription and callback URL over HTTP

Incoming Subscription

Apollo Router

New Data
HTTP Callback Protocol

Register subscription and callback URL over HTTP

Incoming Subscription

Apollo Router

Sends new data via HTTP to callback URL

New Data
HTTP Callback Protocol

Register subscription and callback URL over HTTP

Apollo Router

Sends new data via HTTP to callback URL

New Data

Federated Subscriptions

Incoming Subscription
HTTP Callback Protocol

- **Incoming Subscription**
- **Apollo Router**
  - Register subscription and callback URL over HTTP
  - Sends new data via HTTP to callback URL
- **New Data**
- **Federated Subscriptions**
Subscriptions in your flavor 🍦

```
subscription:
  enabled: true
  mode:
    passthrough:
      subgraphs:
        reviews:
          path: /ws
          protocol: graphql_transport_ws
    preview_callback:
      listen: 0.0.0.0:4000
      path: /callback
      subgraphs:
        - accounts
```

It’s just config for the router
Feature request:

On the Tracks page, we should progressively load the data. You don’t need the album or artist details to start interacting with the page.
The Power of @defer
The Power of @defer
The Power of @defer
The Power of @defer
GraphQL makes your APIs better

You can add GraphQL to your existing API today
Thanks!

Michael Watson
Developer Relations at Apollo

https://discord.gg/graphos