APOLLO

GraphQL Makes Your APIs Better

The Revolution in Efficiency and Real-time Performance

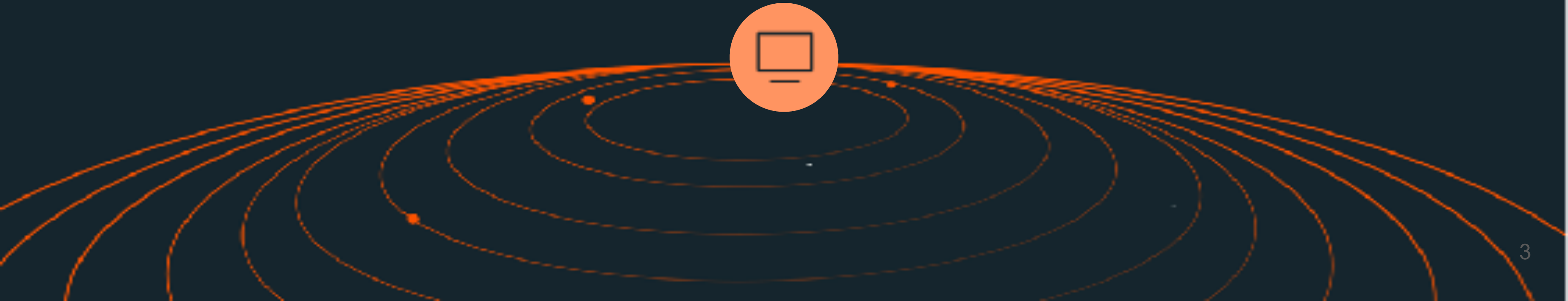
I’m **Obsessed** with GraphQL

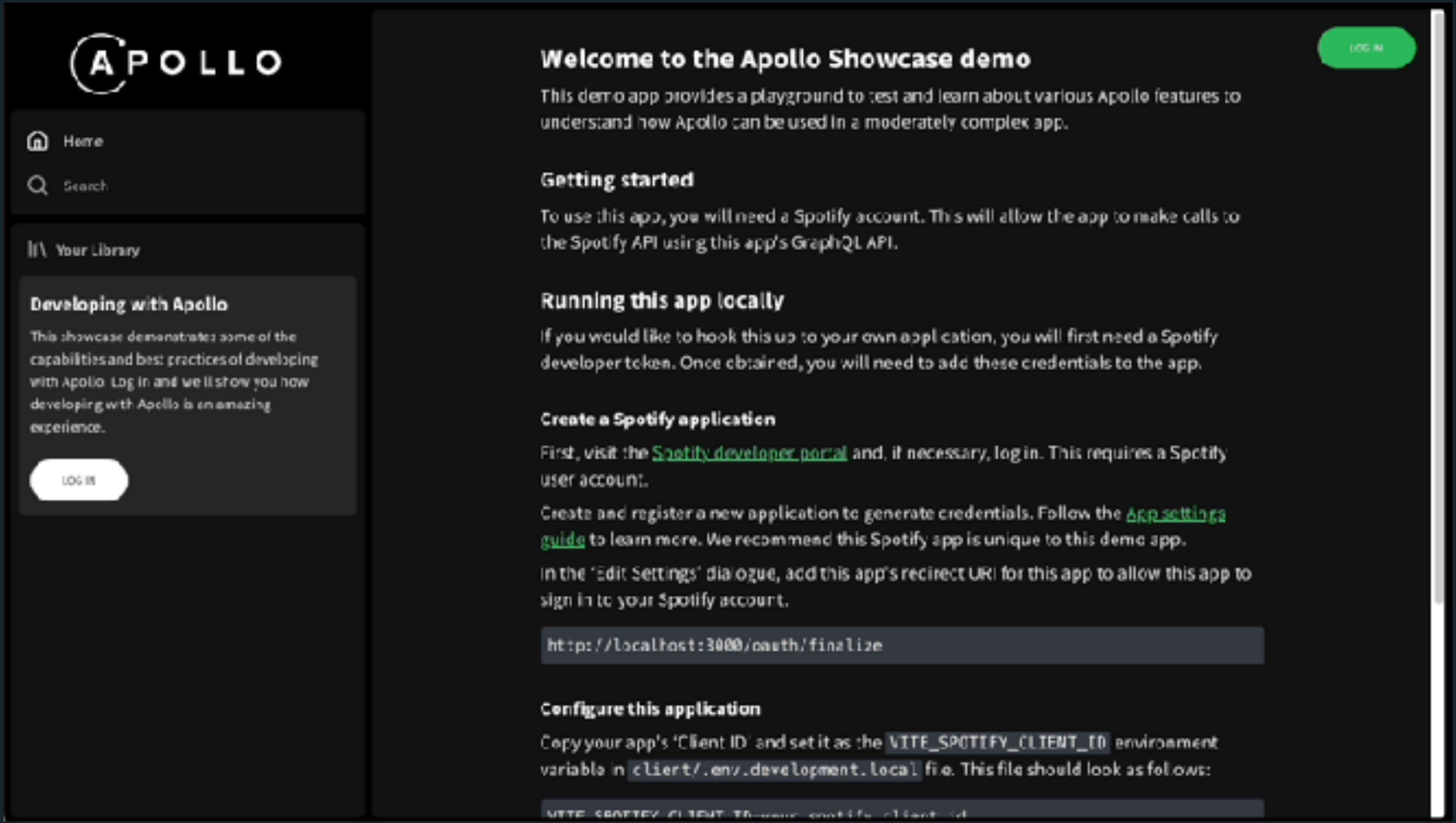
Michael Watson

Developer Relations at Apollo

https://discord.gg/graphos

GraphQL Makes Your APIs Better





Developing with Apollo

This showcase demonstrates some of the capabilities and best practices of developing with Apollo. Log in and we'll show you how developing with Apollo is an amazing experience.

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 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 to learn more. We recommend this Spotify app is unique to this demo app.

In the 'Edit Settings' dialogue, add this app's redirect URI for this app to allow this app to sign in to your Spotify account.

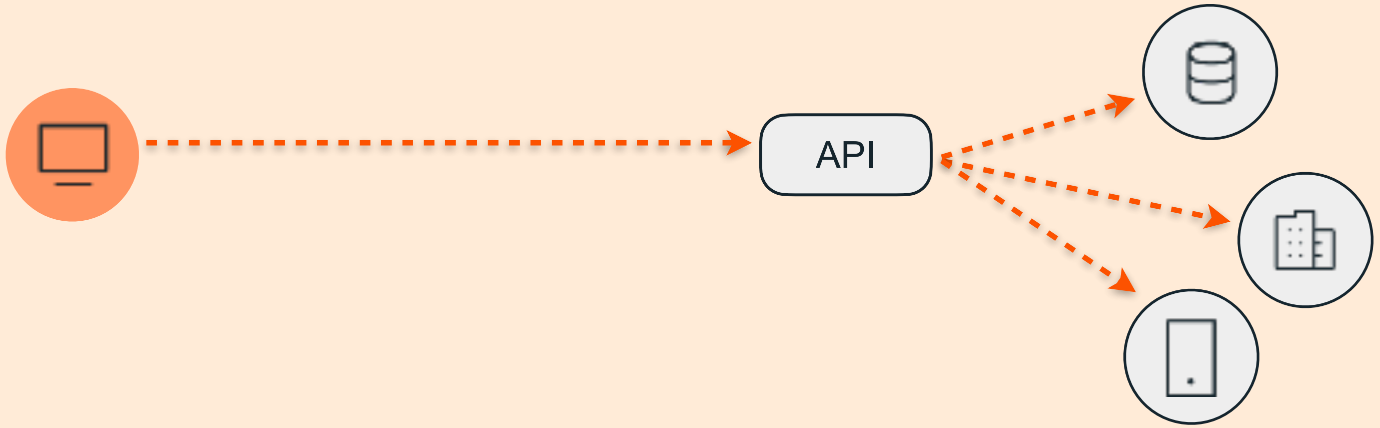
```
http://localhost:3000/oauth/finalize
```

Configure this application

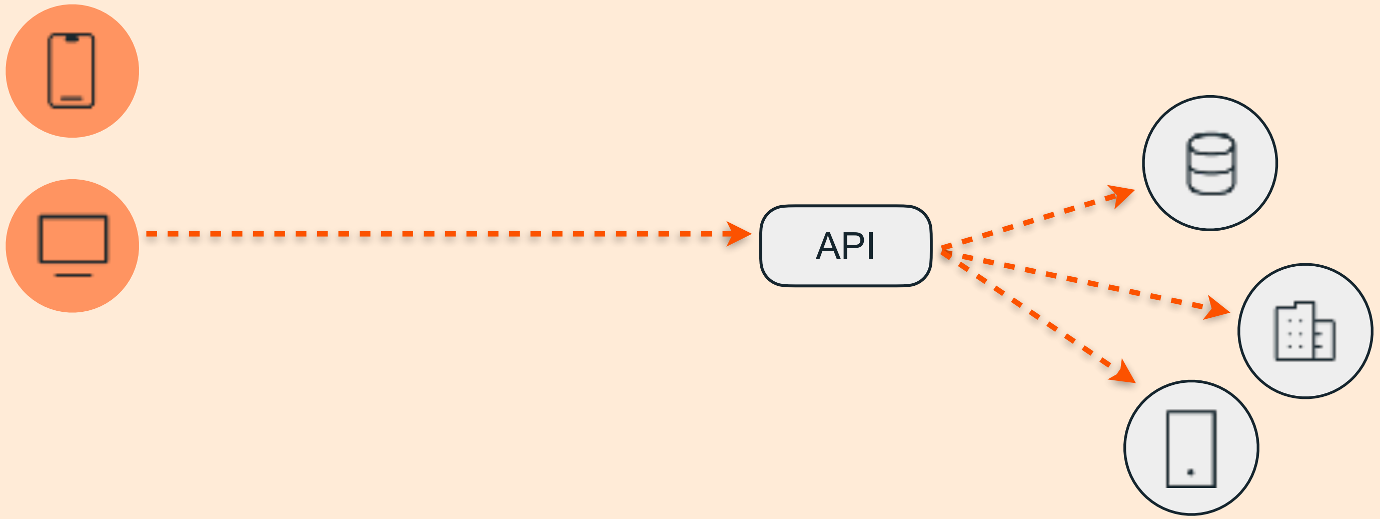
Copy your app's 'Client ID' and set it as the `VITE_SPOTIFY_CLIENT_ID` environment variable in `client/.env.development.local` file. This file should look as follows:

```
VITE_SPOTIFY_CLIENT_ID=your_spotify_client_id
```

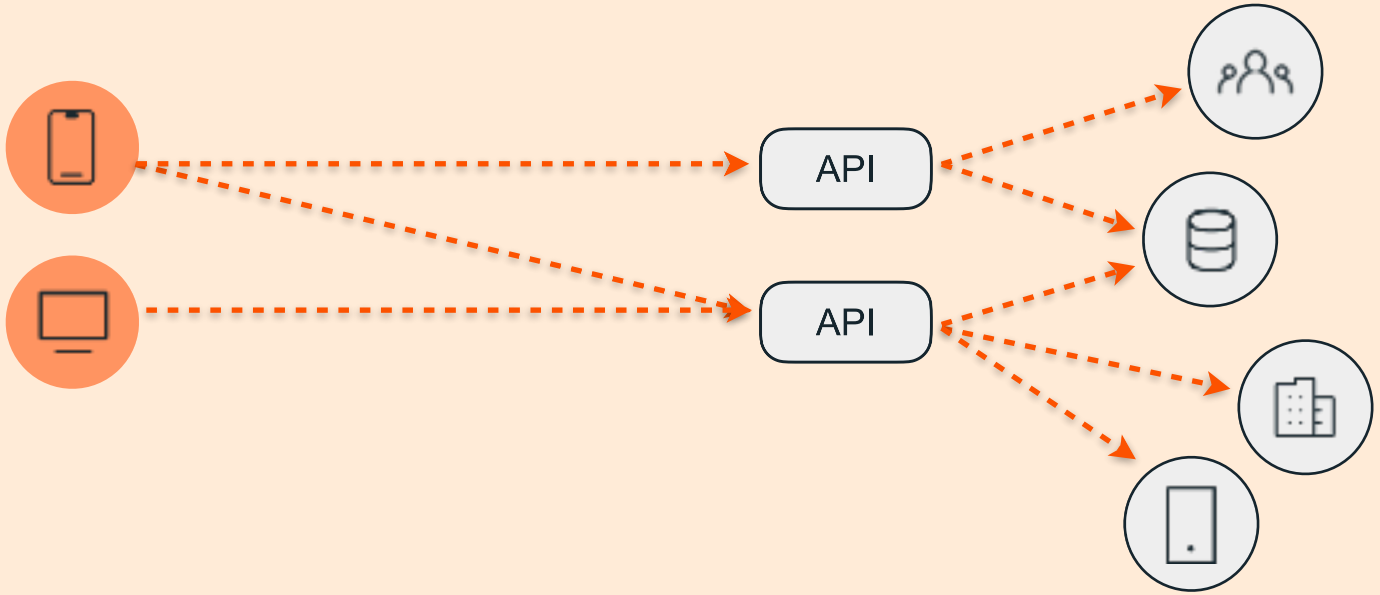
The Initial Dream 🙌



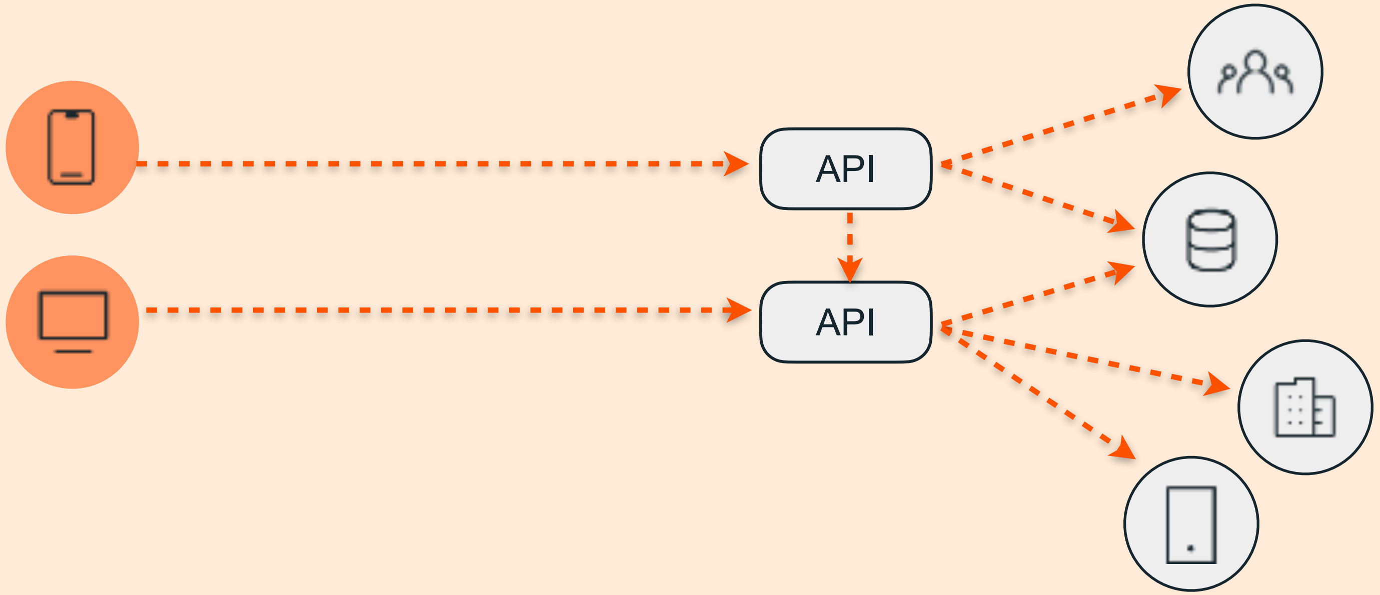
Then came mobile 🚀



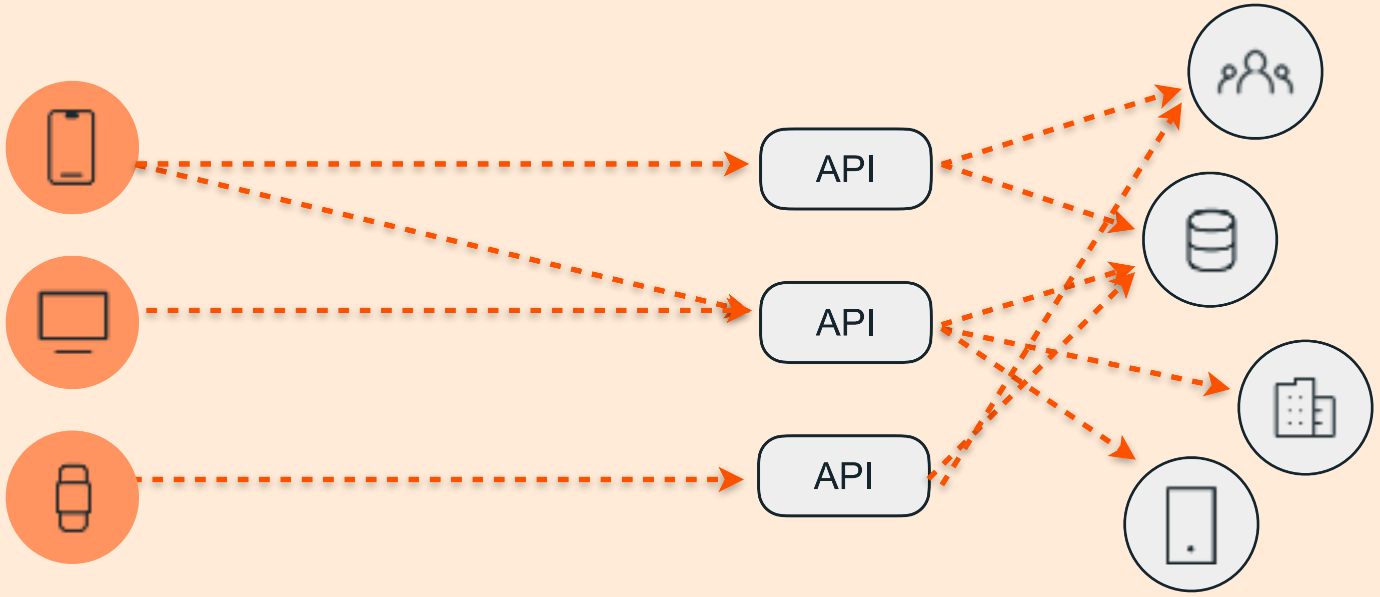
But the mobile team needed something different...



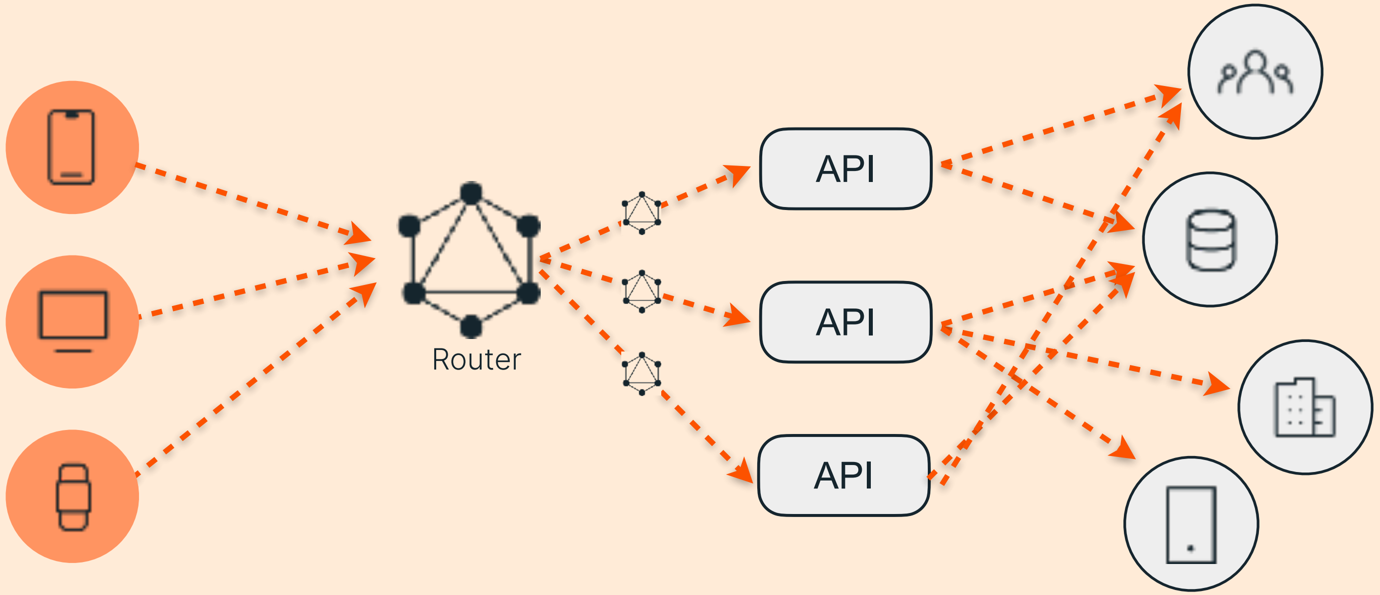
But the mobile team needed something different...



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

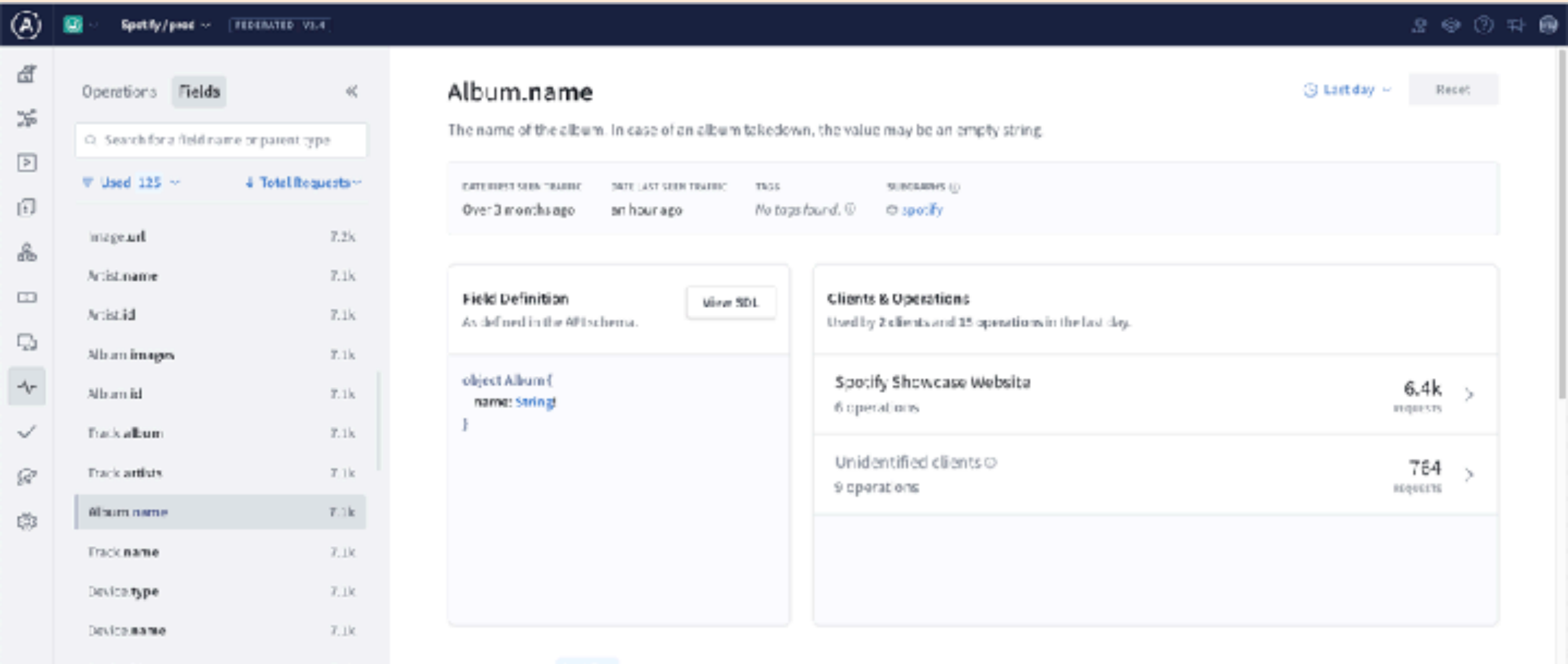


Image.url	7.2k
Artist.name	7.1k
Artist.id	7.1k
Album.images	7.1k
Album.id	7.1k
Track.album	7.1k
Track.artists	7.1k
Album.name	7.1k
Track.name	7.1k
Device.type	7.1k
Device.name	7.1k

Album.name

The name of the album. In case of an album takedown, the value may be an empty string.

DATE FIRST SEEN TRAFFIC	DATE LAST SEEN TRAFFIC	TAGS	SUBGRAPHS
Over 3 months ago	an hour ago	*No tags found.*	spotify

Field Definition

As defined in the API schema.

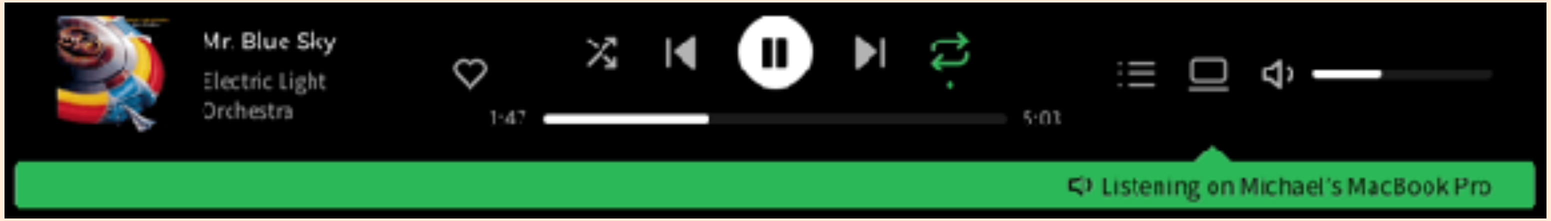
```
object Album {
  name: String!
}
```

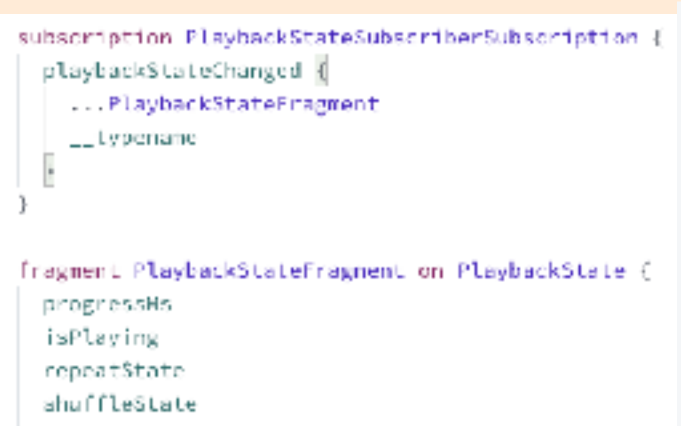
Clients & Operations

Used by 2 clients and 15 operations in the last day.

Spotify Showcase Website 6 operations	6.4k requests
Unidentified clients 9 operations	764 requests

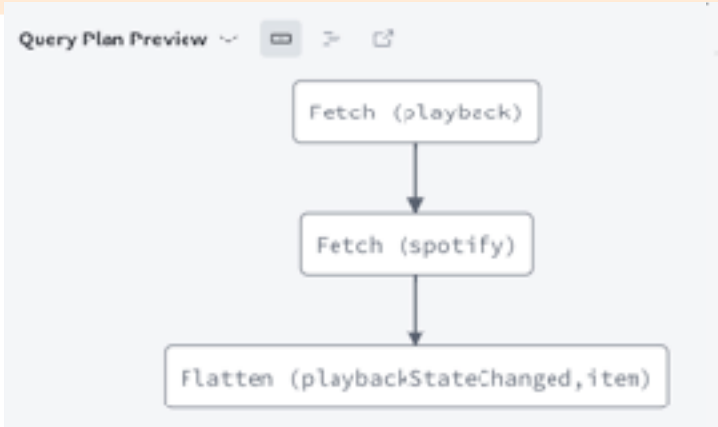
Real-Time Data

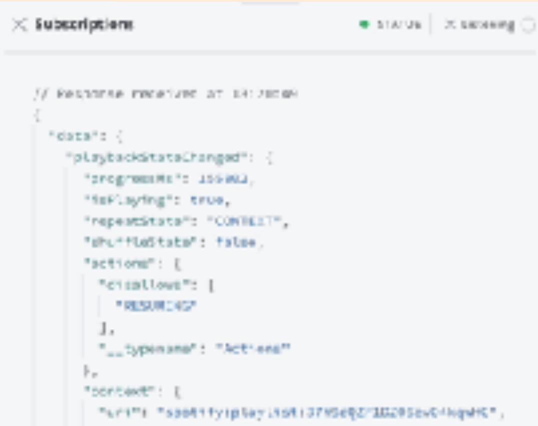




```
subscription PlaybackStateSubscriberSubscription {
  playbackStateChanged {
    ...PlaybackStateFragment
    __typename
  }
}

fragment PlaybackStateFragment on PlaybackState {
  progressMs
  isPlaying
  repeatState
  shuffleState
```





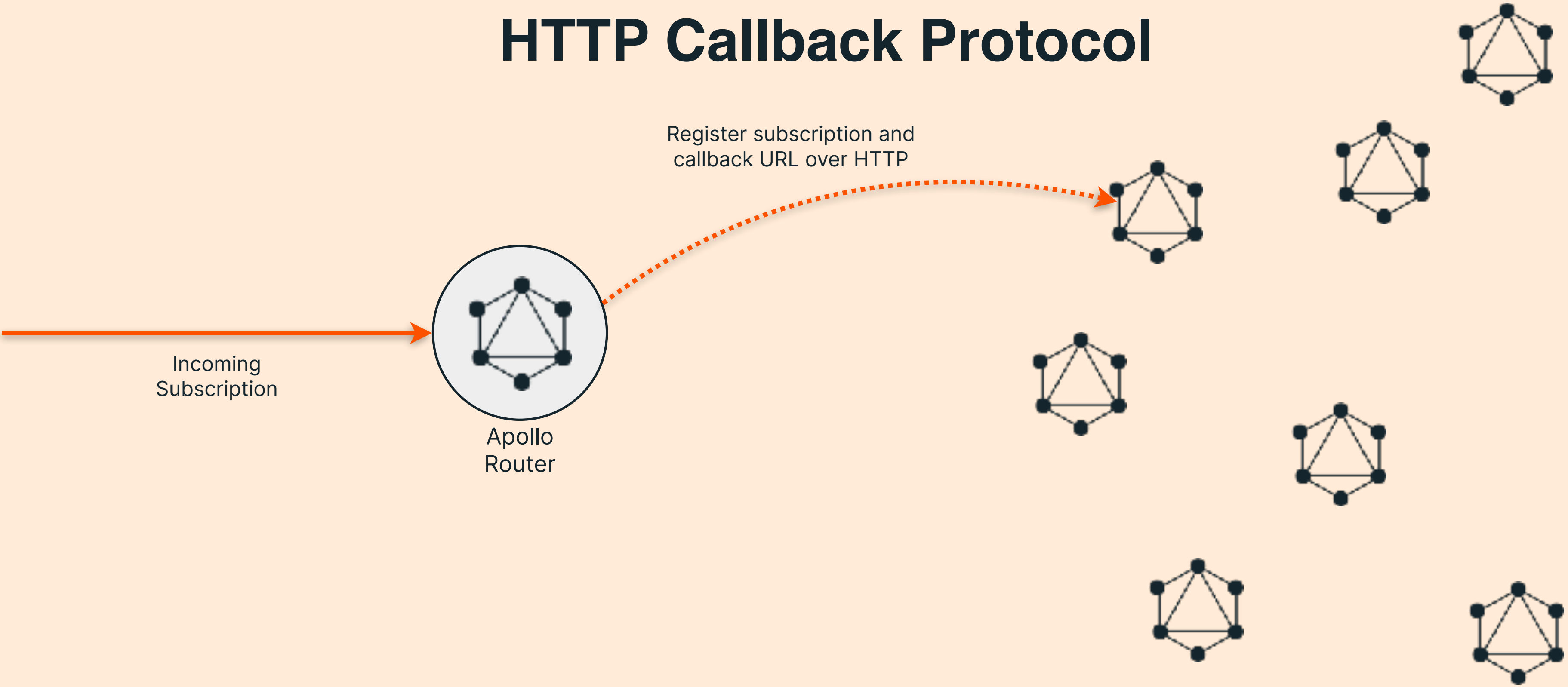
```
// Response received at [illegible]
{
  "data": {
    "playbackStateChanged": {
      "progressMs": [illegible],
      "isPlaying": true,
      "repeatState": "CONTEXT",
      "shuffleState": false,
      "actions": {
        "disallows": [
          "RESUMING"
        ],
        "__typename": "Actions"
      },
      "context": {
        "uri": "spotify:playlist:[illegible]",
```

Pushing the boundaries with Real-time

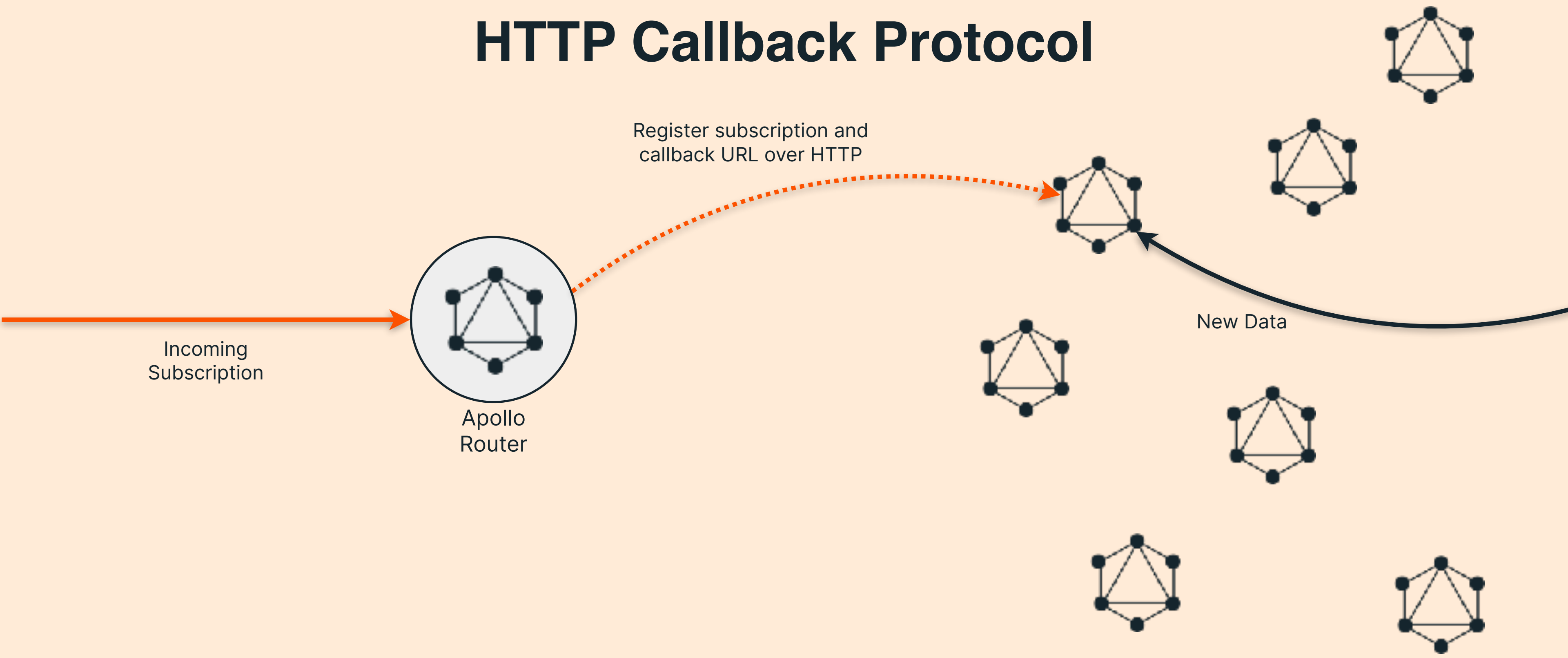
WebSockets

HTTP Callback

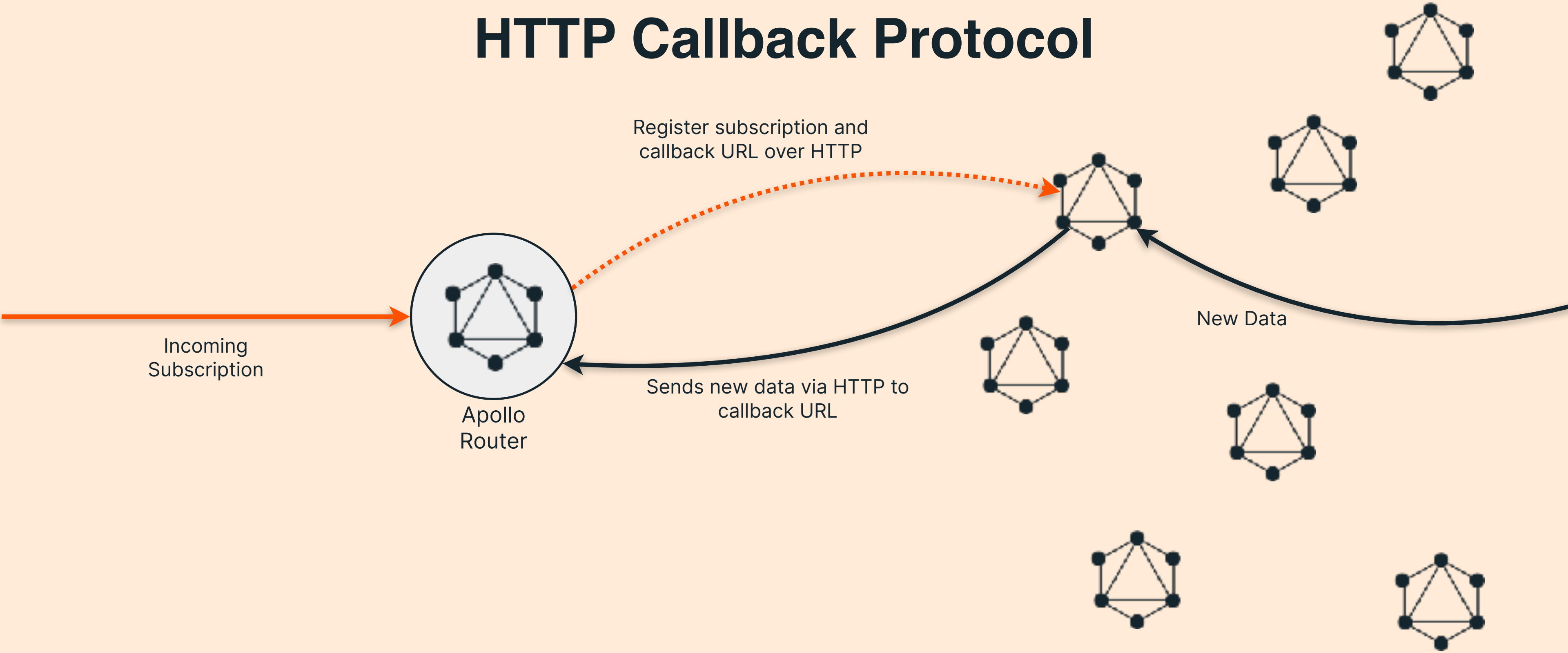
HTTP Callback Protocol



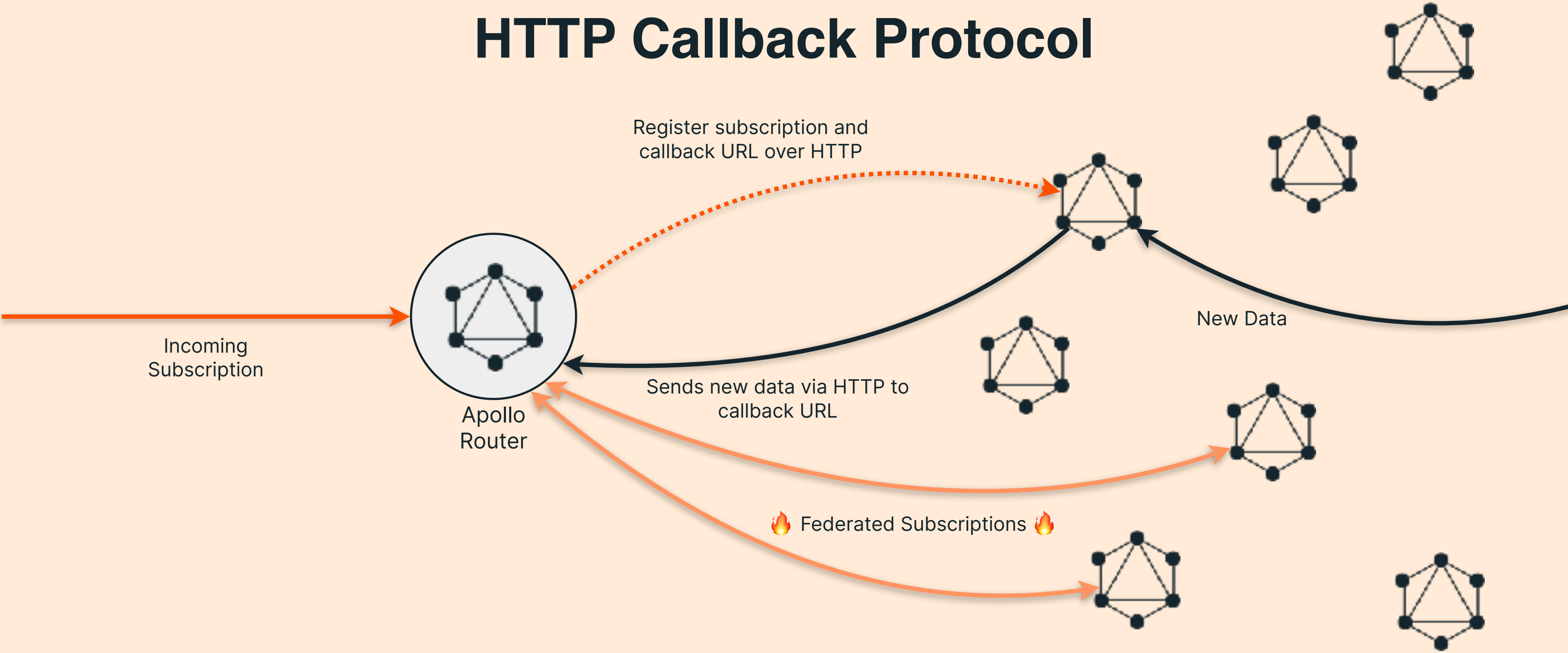
HTTP Callback Protocol



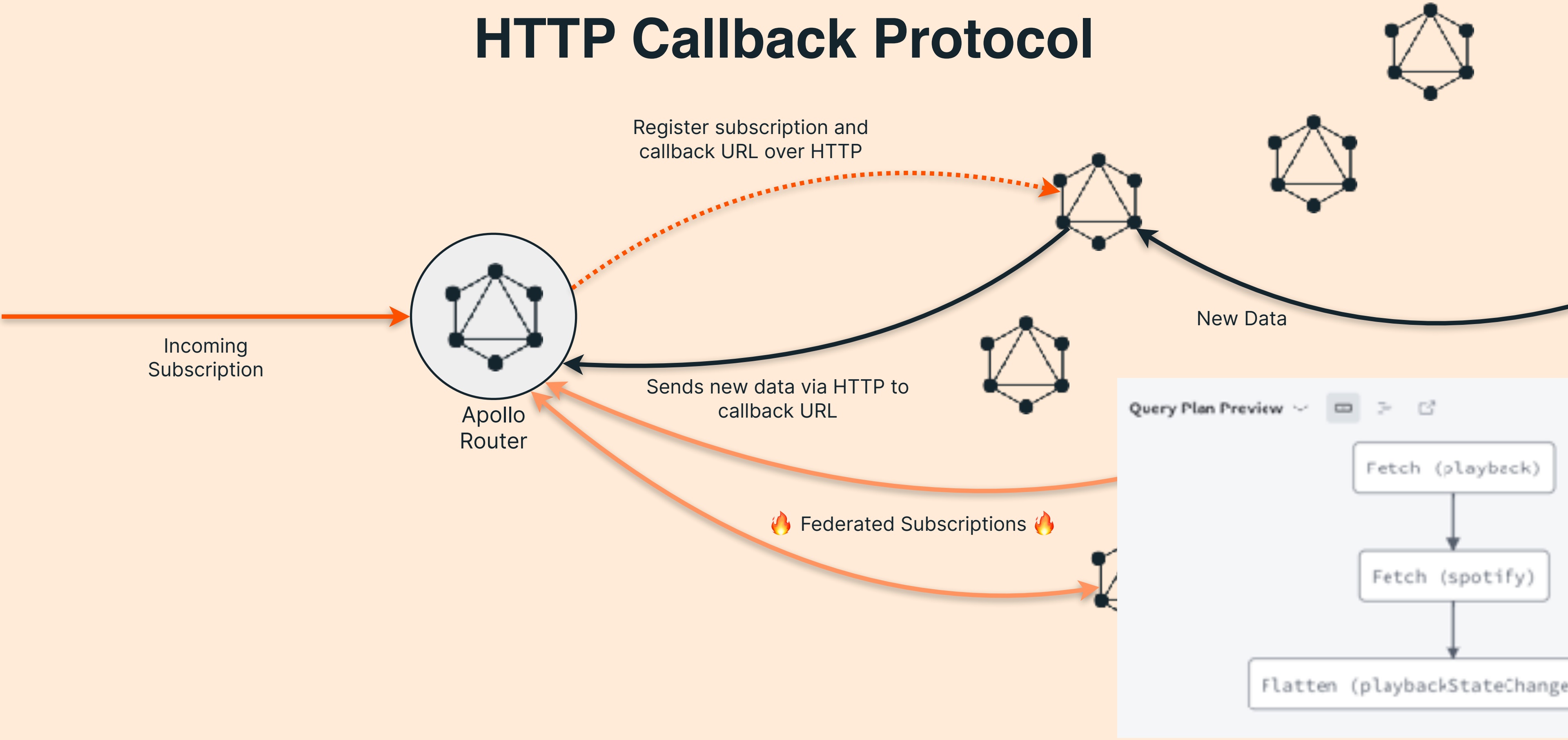
HTTP Callback Protocol


Register subscription and callback URL over HTTP
Incoming Subscription
Apollo Router
New Data
Sends new data via HTTP to callback URL

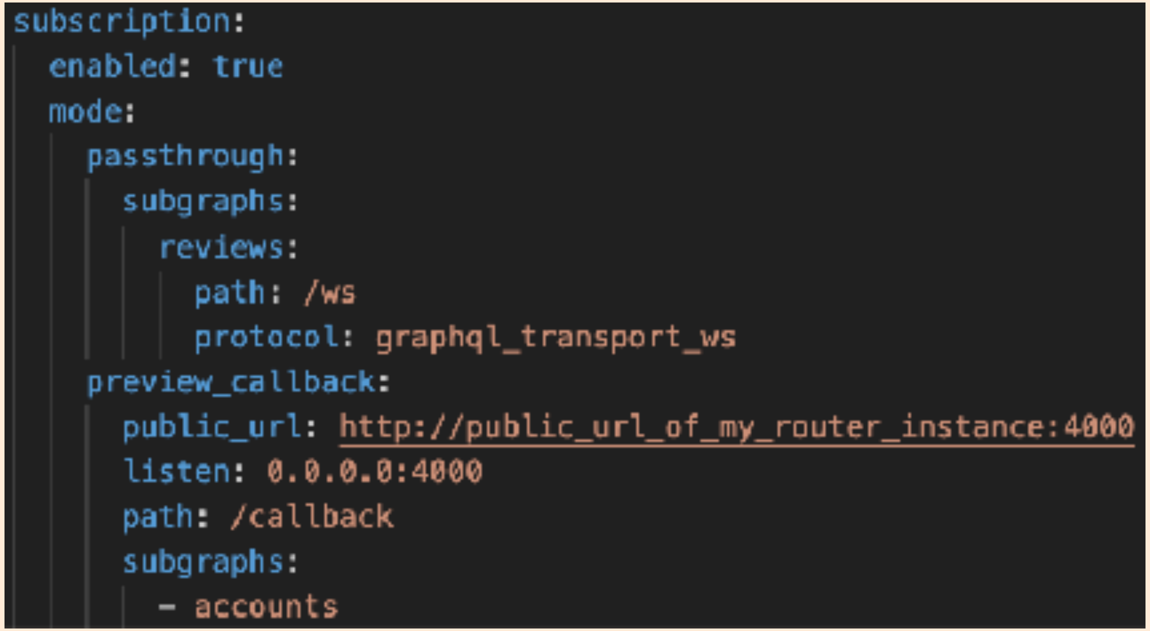
HTTP Callback Protocol


Register subscription and callback URL over HTTP
Incoming Subscription
Apollo Router
New Data
Sends new data via HTTP to callback URL
🔥 Federated Subscriptions 🔥

HTTP Callback Protocol


Register subscription and callback URL over HTTP
Incoming Subscription
Apollo Router
New Data
Sends new data via HTTP to callback URL
🔥 Federated Subscriptions 🔥
Query Plan Preview
Fetch (playback)
Fetch (spotify)
Flatten (playbackStateChange

Subscriptions in your flavor 🍦



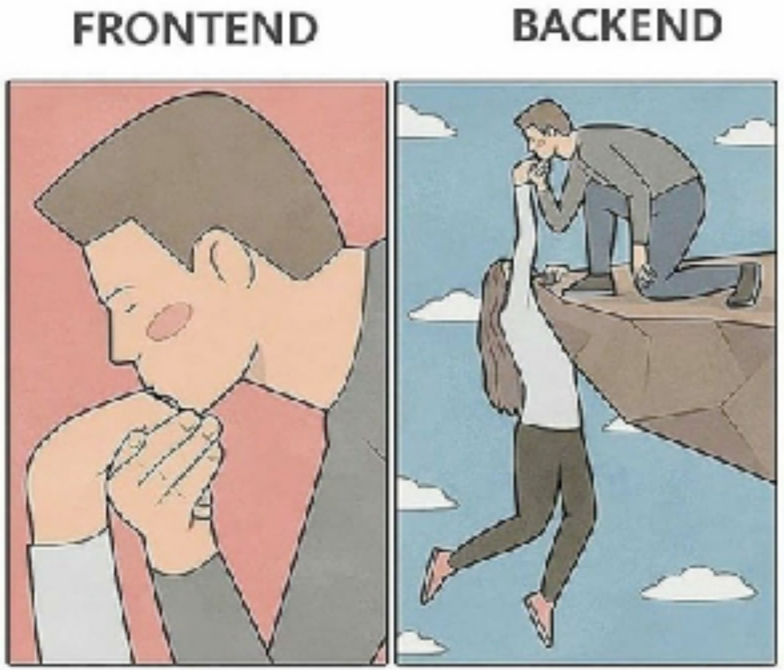
```yaml
subscription:
  enabled: true
  mode:
    passthrough:
      subgraphs:
        reviews:
          path: /ws
          protocol: graphql_transport_ws
    preview_callback:
      public_url: http://public_url_of_my_router_instance:4000
      listen: 0.0.0.0:4000
      path: /callback
      subgraphs: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

It's just config for the router

Feature request:

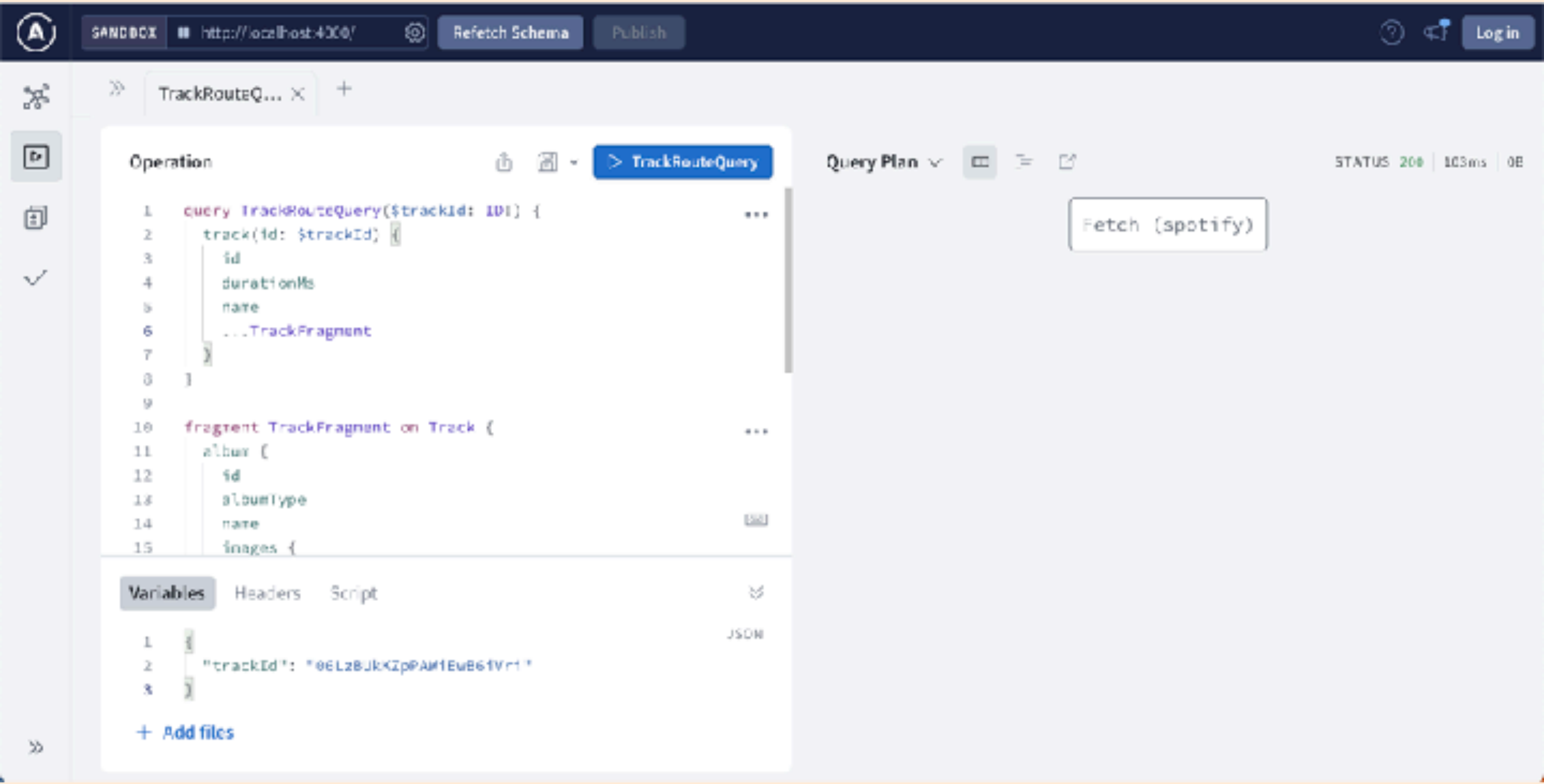
On the Tracks page, we should progressively load the data. You don't need the album or artist details to start interacting with the page.

```
query TrackRouteQuery($trackId: ID!) {
  track(id: $trackId) {
```



```
      topTracks {
        id
        name
```

The Power of @defer



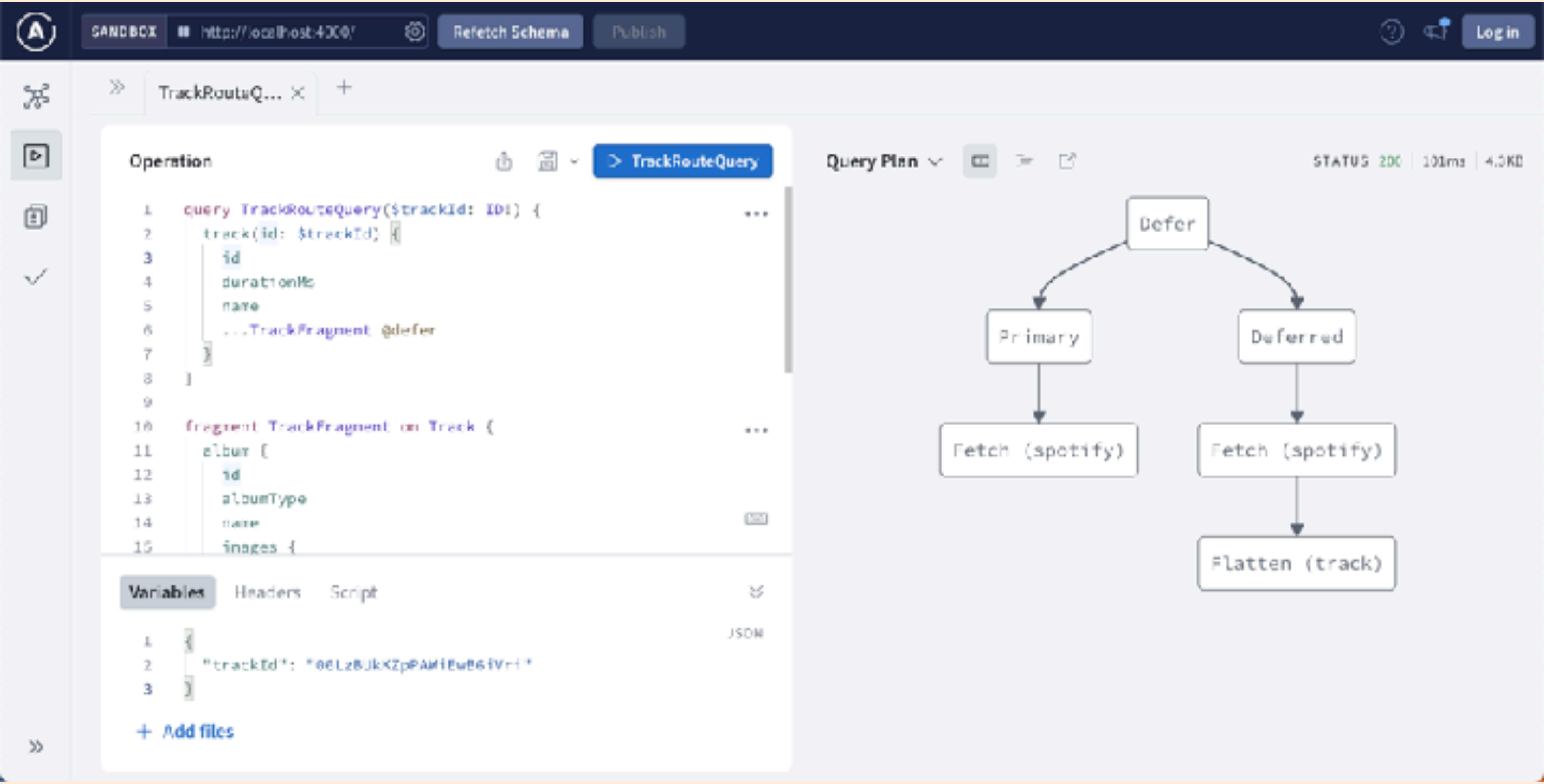
```graphql
query TrackRouteQuery($trackId: ID!) {
  track(id: $trackId) {
    id
    durationMs
    name
    ...TrackFragment
  }
}

fragment TrackFragment on Track {
  album {
    id
    albumType
    name
    images {
```

```json
{
  "trackId": "0eLzBJkKZpPAM1EwB61Vr1"
}
```

```
Fetch (spotify)
```

The Power of @defer

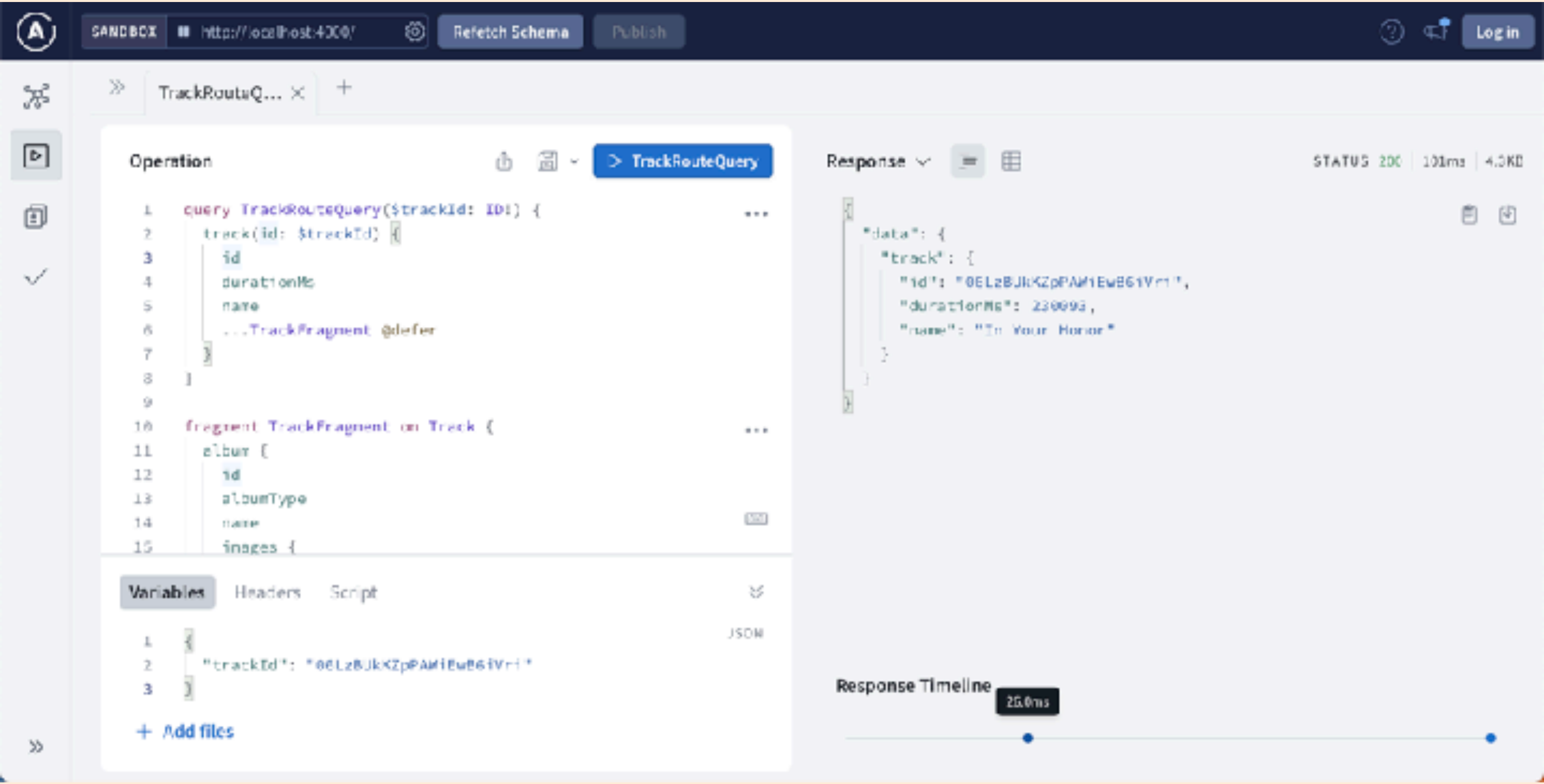


```graphql
query TrackRouteQuery($trackId: ID!) {
  track(id: $trackId) {
    id
    durationMs
    name
    ...TrackFragment @defer
  }
}

fragment TrackFragment on Track {
  album {
    id
    albumType
    name
    images {
```

```json
{
  "trackId": "0cLzBJkKZpPAWiEwB6iVri"
}
```

The Power of @defer



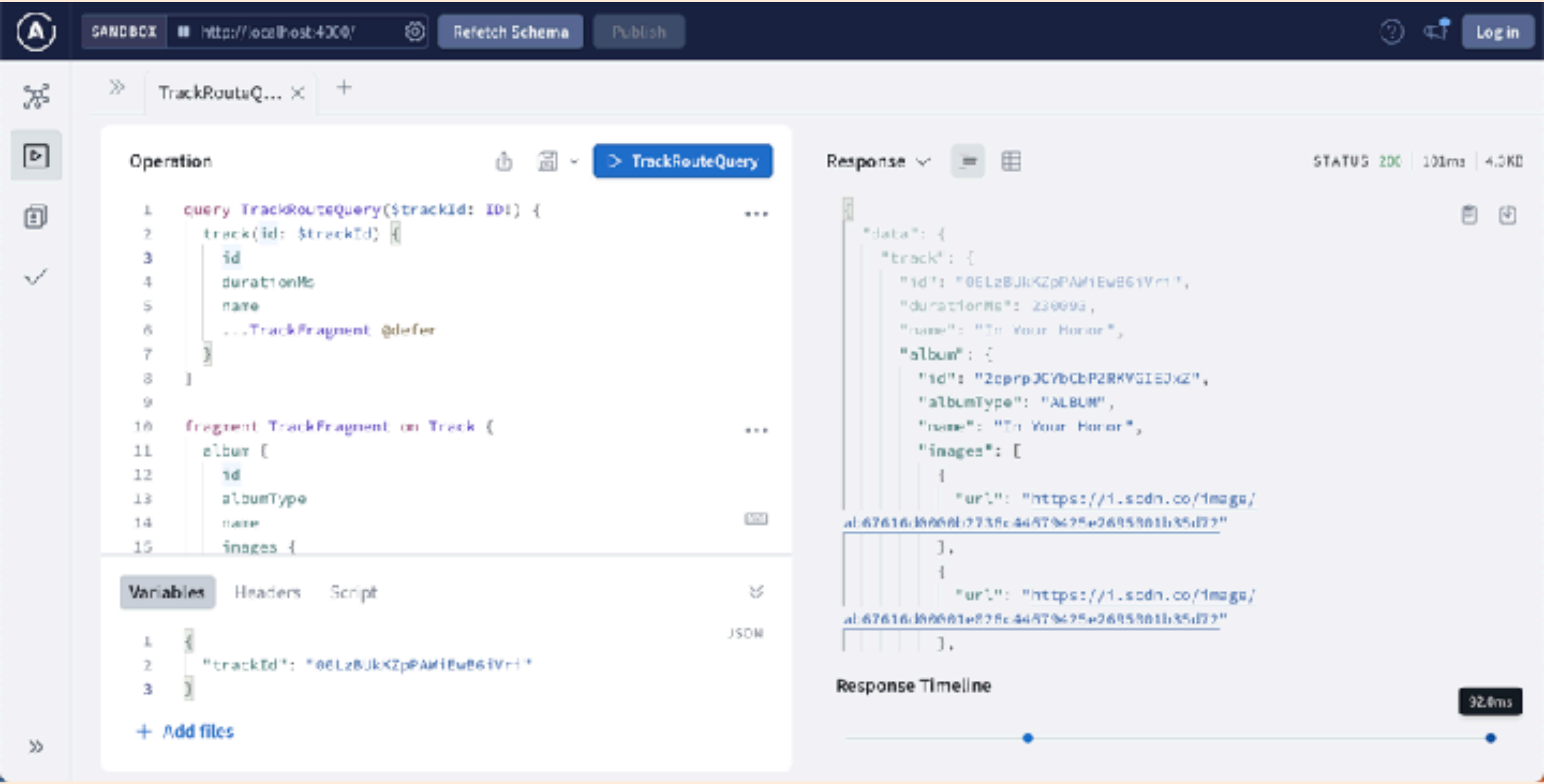
```graphql
query TrackRouteQuery($trackId: ID!) {
  track(id: $trackId) {
    id
    durationMs
    name
    ...TrackFragment @defer
  }
}

fragment TrackFragment on Track {
  album {
    id
    albumType
    name
    images {
```

```json
{
  "trackId": "0ELzBJkKZpPAWiEwB6iVri"
}
```

```json
{
  "data": {
    "track": {
      "id": "0ELzBJkKZpPAWiEwB6iVri",
      "durationMs": 230093,
      "name": "In Your Honor"
    }
  }
}
```

The Power of @defer



```graphql
query TrackRouteQuery($trackId: ID!) {
  track(id: $trackId) {
    id
    durationMs
    name
    ...TrackFragment @defer
  }
}

fragment TrackFragment on Track {
  album {
    id
    albumType
    name
    images {
```

```json
{
  "trackId": "0ELzBJkKZpPAWiEwB61Vri"
}
```

STATUS 200 | 101ms | 4.3KB

```json
{
  "data": {
    "track": {
      "id": "0ELzBJkKZpPAWiEwB61Vri",
      "durationMs": 230093,
      "name": "In Your Honor",
      "album": {
        "id": "2cprpJCYbCbP2RKVGIEJxZ",
        "albumType": "ALBUM",
        "name": "In Your Honor",
        "images": [
          {
            "url": "https://i.scdn.co/image/
ab67616d0000b2738c4457962 5e2695801b35d72"
          },
          {
            "url": "https://i.scdn.co/image/
ab67616d00001e028c4457962 5e2695801b35d72"
          },
```

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