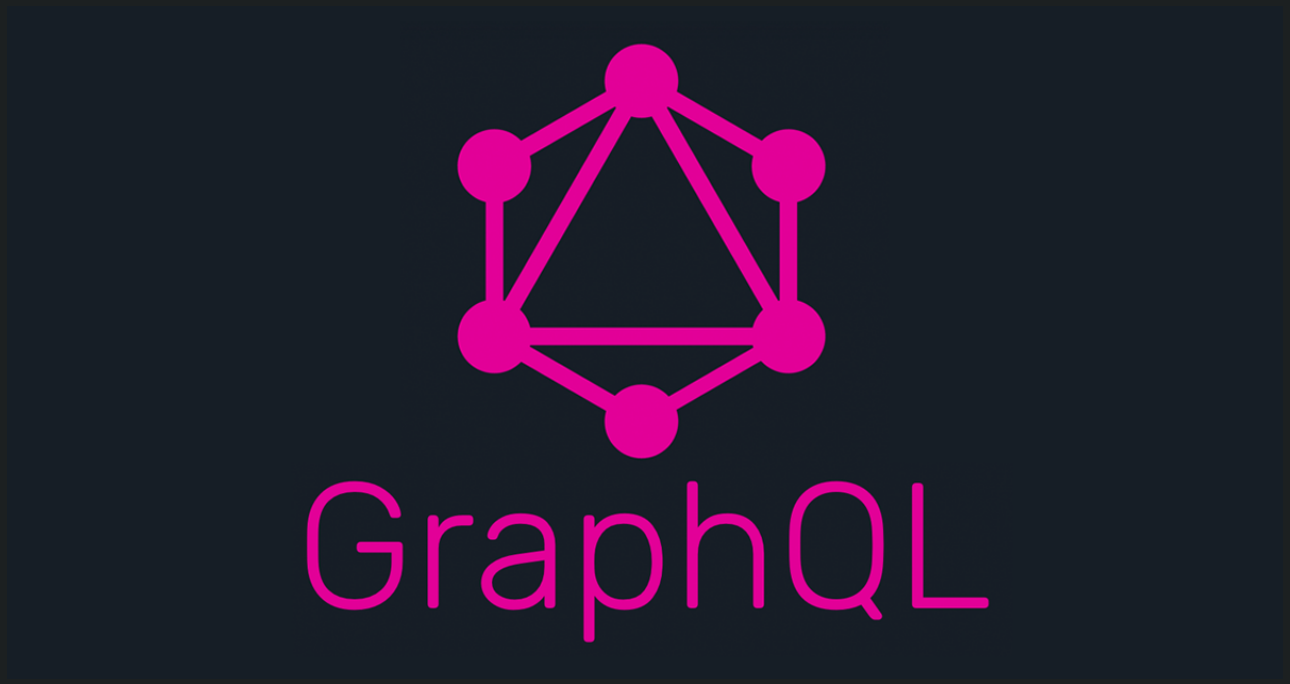


Lunch & Learn

Nafeu Nasir - Full Stack Developer (Console Team)

March 22nd, 2019

What is your experience with GraphQL:

a – never heard of it

b – heard of it but have never used it

c – used it in a sandbox or pet project

d – used it in production

What is GraphQL?

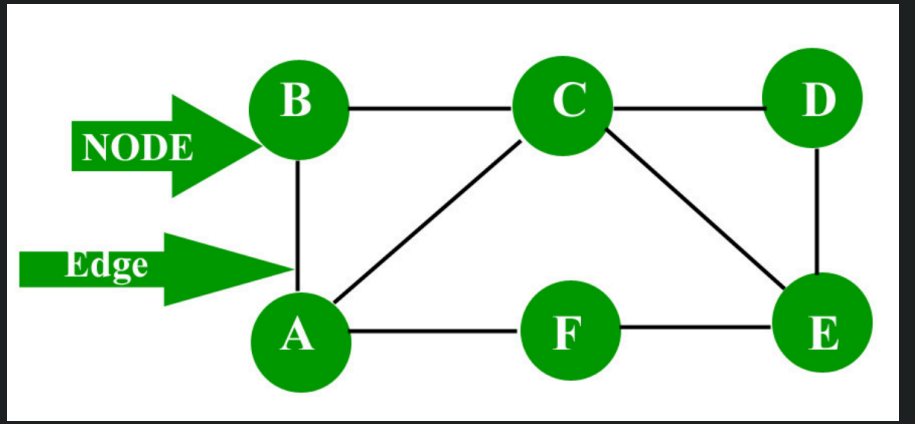
First lets get some context:

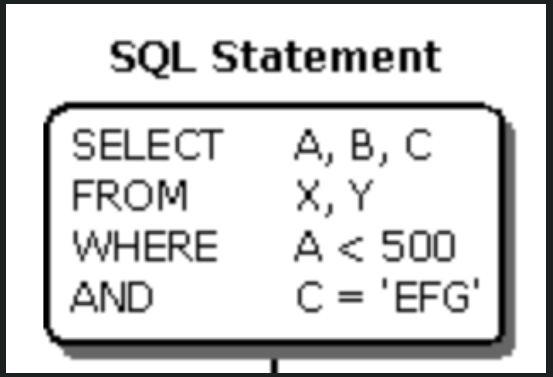
Graph => thinking of a problem in terms of nodes and edges

QL => query language (a way of asking for data)

GraphQL => a way of asking for data in terms of nodes and edges

Imagine querying an API in terms of nodes and edges instead of endpoints.





Describe your data

```
type Project {
  name: String
  tagline: String
  contributors: [User]
}
```

Ask for what you want

```
{
  project(name: "GraphQL") {
    tagline
  }
}
```

Get predictable results

```
{
  "project": {
    "tagline": "A query language for APIs"
  }
}
```

It can be your API, or a wrapper around your existing API, or an API gateway to manage communications to all your microservices.

It is a process that sits between your application logic and your database resources, but at its core its just a paradigm for designing your API(s).

Why should you care?

Maintaining and evolving REST APIs in the long run is:

a - Trivial

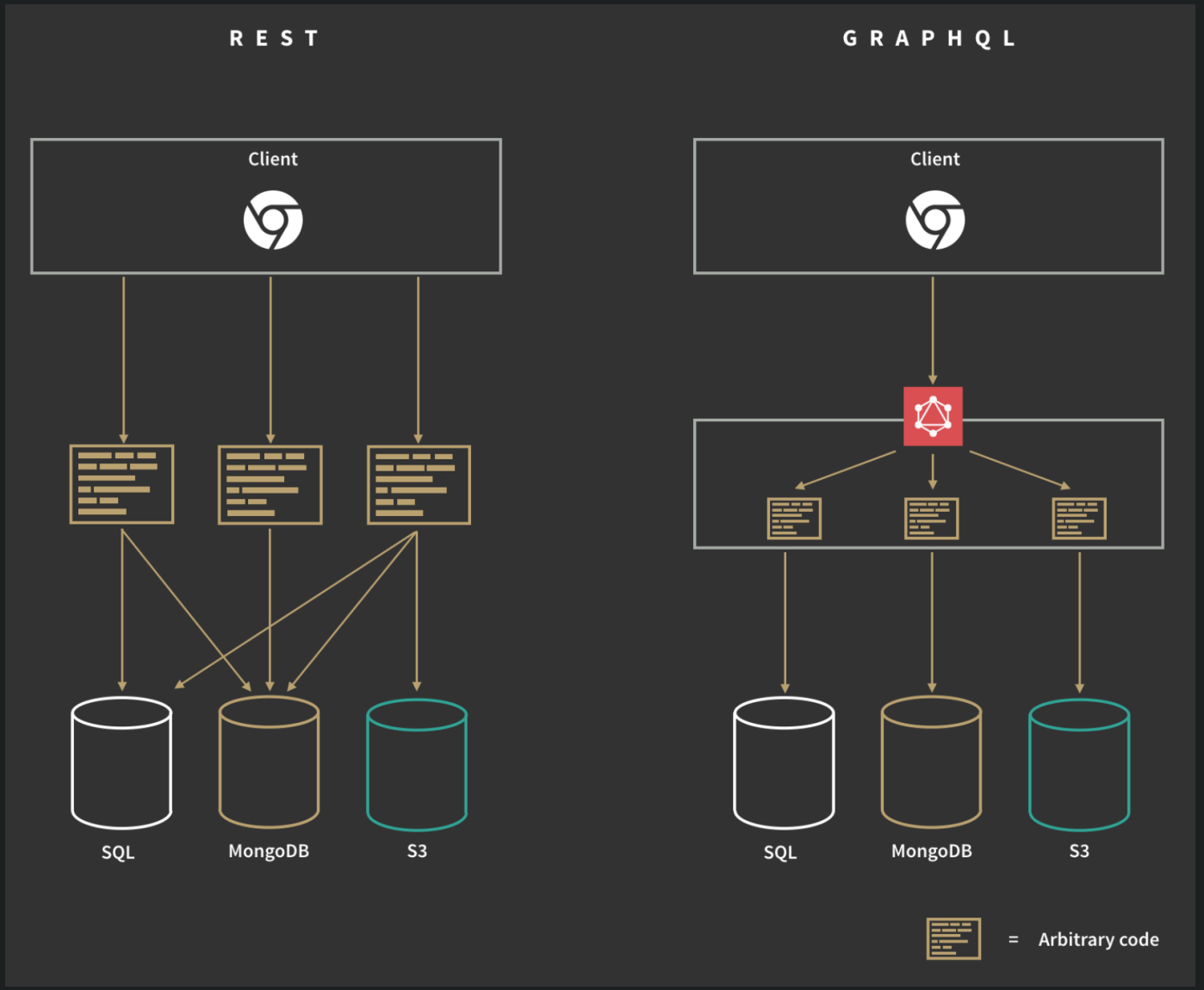
b - Meh

c - Reasonably difficult

d - Very difficult

It replaces REST APIs! (or can be used to manage multiple separate REST APIs)

- it can be used in conjunction with existing technology
- it is shown to be highly performant, more maintainable, more scalable and more efficient in place of REST



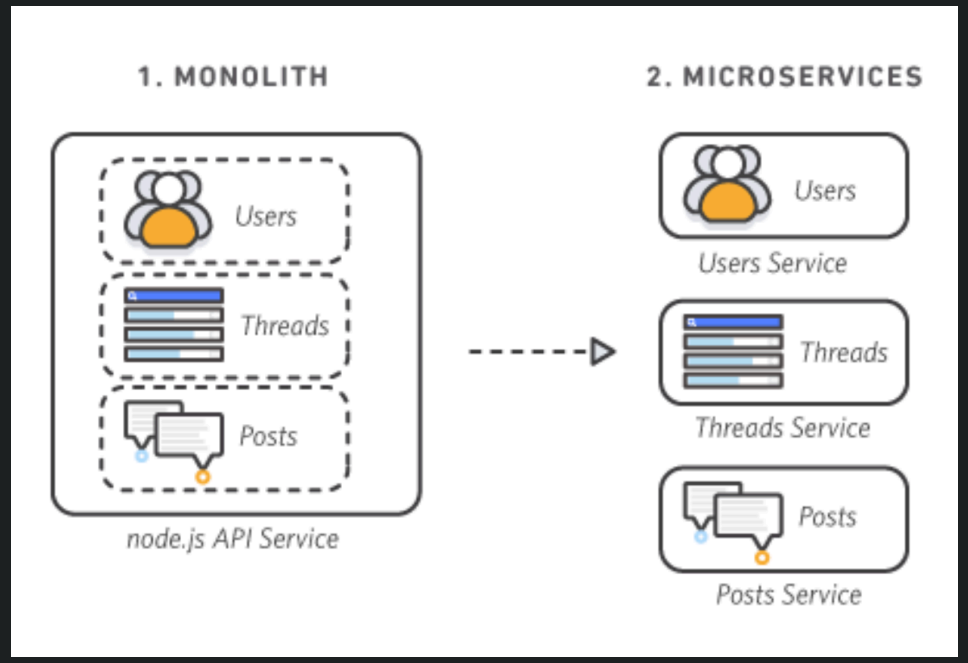
Who's using GraphQL?

Facebook's mobile apps have been powered by GraphQL since 2012.
A GraphQL spec was open sourced in 2015 and is now available in
many environments and used by teams of all sizes.




Goodbye Monolith

- the industry is moving towards microservice and hexagonal architecture over monolithic architecture (and for good reason)
- we have a huge variation in experiences now (mobile, tablet, IOT device, desktop, voice assistant) all querying the same resources
- we have Third Party APIs that do certain challenging things really well so that we don't have to anymore
  - Twilio: handles texting & automated voice calls
  - Braintree: handles monetary transactions
  - Auth0: handles authentication and user account management
  - Goole/Amazon/IBM/Microsoft Cloud: various AI related APIs



Our problems, in layman's:

Our apps are talking to a lot of things, and then those things are talking to other things. Some of those things are inside a database that we own and some aren’t. Some of those things get searched through multiple times even when its not necessary. Some of those lookups hog more resources than they need to. Sometimes we need to change the types of things we are looking for while simultaneously NOT breaking how we are already looking for things.

Sometimes we do not get all of the things and we don’t know why.

Facebook's Solution:

Instead of having multiple "dumb" endpoints, have a single "smart" endpoint that can take in complex queries, and then massage the data output into whatever shape the client requires.

“I want a list of all the users but sometimes I just need a list of their emails, or just their phone numbers”

With REST:

Multiple endpoints? or additional query params?

`api/users/getAll`

`api/users/getAllEmails`

`api/users/getAll?filter=email`

…

With GraphQL:

One endpoint: /graphql?query=‘…’

```
`{
  User {
    emails
  }
}`
```

```
{
  hero {
    name
    height
    mass
  }
}
```

```
{
  "hero": {
    "name": "Luke Skywalker",
    "height": 1.72,
    "mass": 77
  }
}
```

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.

You worked on a production level API that you felt was making more requests and serving more data than it needed to:

a - Disagree

b – Mildly Disagree

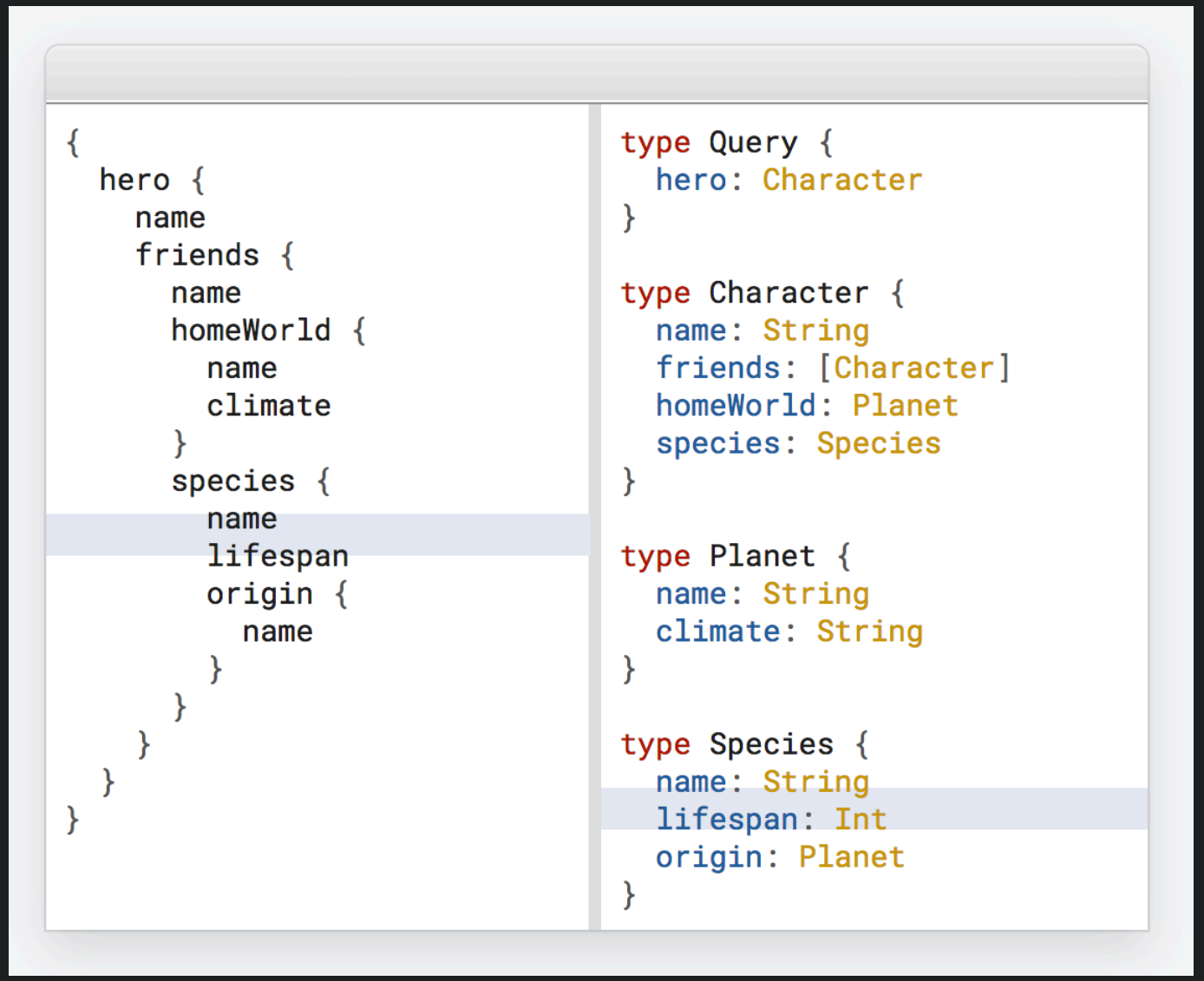
c - Agree

d – Strongly Agree

GraphQL makes it easy to stich together results from various combined external resources.

An example of this will be shown during the demo.

How about maintainability?

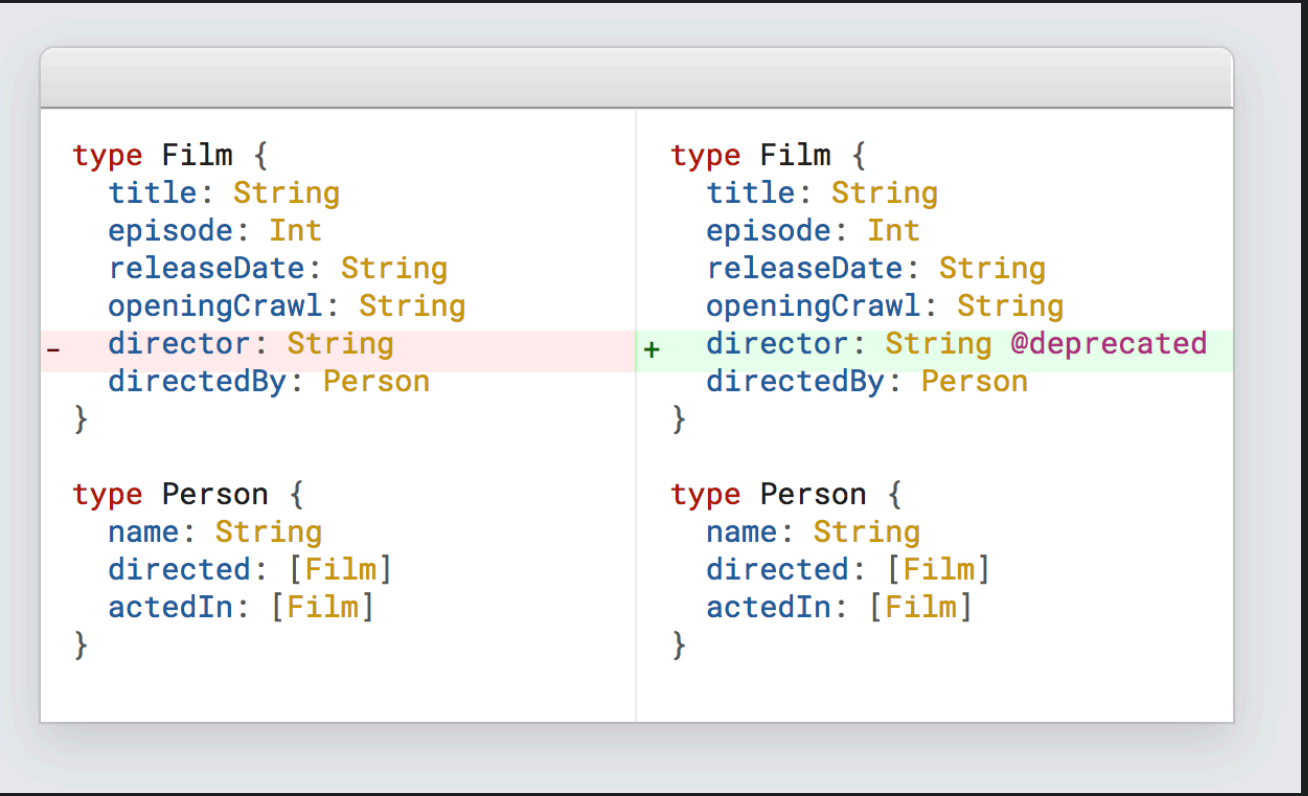


Type System

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.

Update and evolve your API without versions

Add new fields and types to your GraphQL API without impacting existing queries. Aging fields can be deprecated and hidden from tools. By using a single evolving version, GraphQL APIs give apps continuous access to new features and encourage cleaner, more maintainable server code.



There are some awesome devtools that come with GraphQL right out of the box which will be shown in the demo.

DEMO TIME!

source code available at:

github.com/nafeu-pelmorex/pa-graphql-lnl

Discussion Time:

How can we potentially utilize GraphQL to improve our apps, apis and other technical offerings at Pelmorex?

What are some current bottlenecks on an API level?