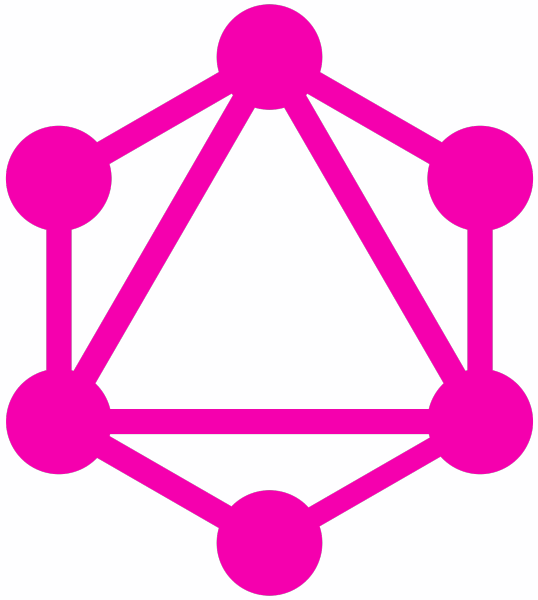
OSF | Global Services



GraphQL

What is a schema and how GraphQL works

OSF Global Services

Software Developer

Highly-skilled software development professional bringing more than five years of experience in software design, development and integration.

OSF Global Services

What's GraphQL?

- New **API standard** by Facebook
- GraphQL server exposes **single endpoint**
- Over by it **over your** existing **backends**
- **Declarative** way of **fetching** & **updating** data

OSF Global Services

The Five Design Principles

OSF Global Services

Hierarchical

A GraphQL query is hierarchical. **Fields are nested within other fields** and the query is shaped like the data that it returns.

OSF Global Services

Product Centric

GraphQL is **driven by the data needs of the client** and the language and runtime that support the client.

OSF Global Services

Strong Typing

A GraphQL server is backed by the GraphQL type system. In schema, **each data point has a specific type** against which it will be validated.

Client-Specified Queries

A GraphQL server **provides the capabilities** that the client are allowed to consume.

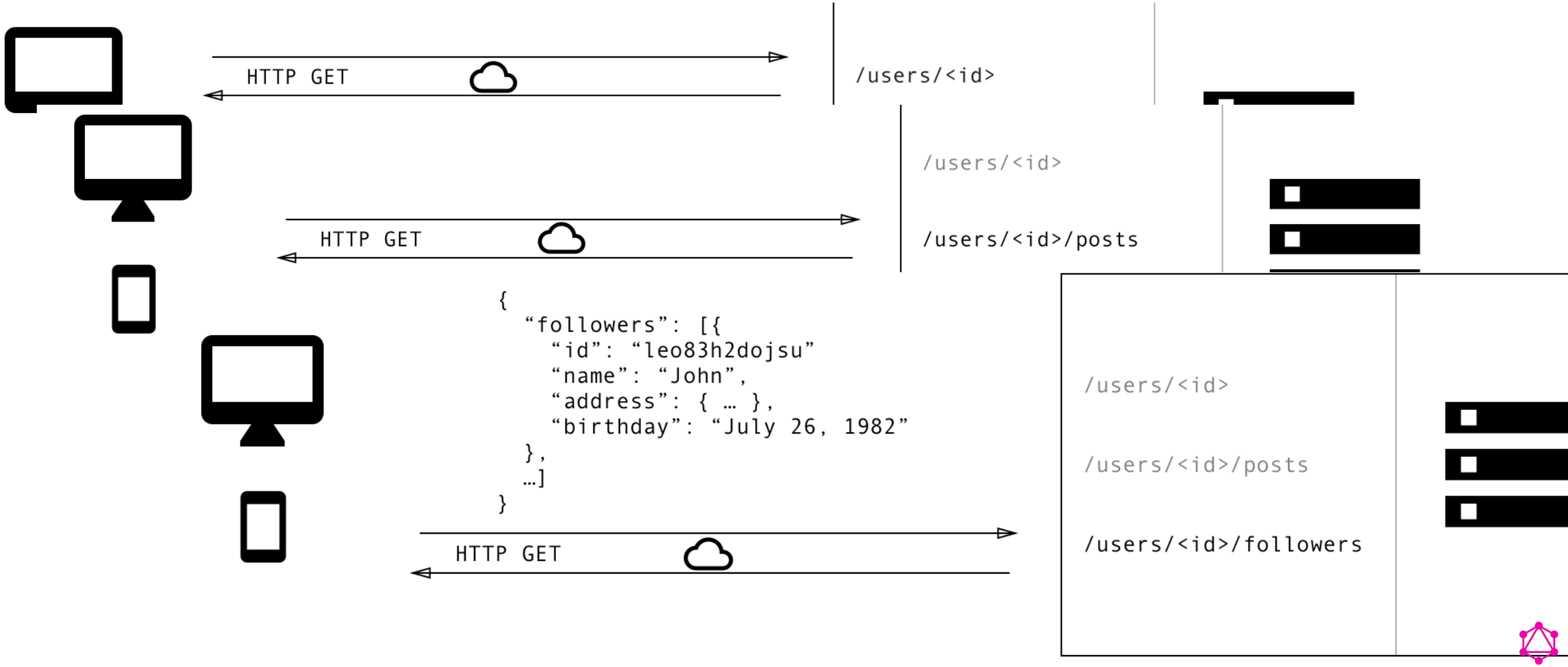
OSF Global Services

Introspective

The GraphQL **Language is able to query the** GraphQL server's **type system.** In other words, it knows itself.

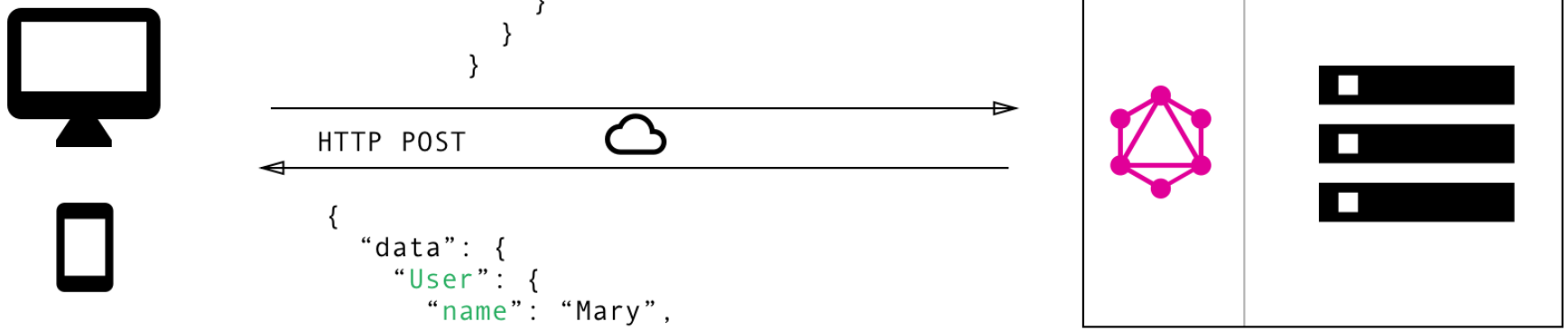
OSF Global Services

No more **OVER** fetching



No more **UNDER** fetching

```
query {
  User(id: “er3tg439frjw”) {
    name
    posts {
      title
    }
    followers(last: 3) {
      name
    }
  }
}
```




```
{
  “data”: {
    “User”: {
      “name”: “Mary”,
      “posts”: [
        { title: “Learn GraphQL today” }
      ],
      “followers”: [
        { name: “John” },
        { name: “Alice” },
        { name: “Sarah” },
      ]
    }
  }
}
```

Schema

GraphQL uses a strong type system to define the capabilities of an API. They are exposed in an API using a **Schema Definition Language (SDL)**.

```
type Person {
   id: ID!
   name: String!
   age: Int!
   name: [Post!]!
}

type Post {
   title: String!
   author: Person!
}
```

OSF Global Services

Schema

Schema Definition Language
express the **relationships**
and the shape of the data
to return

OSF Global Services

Queries

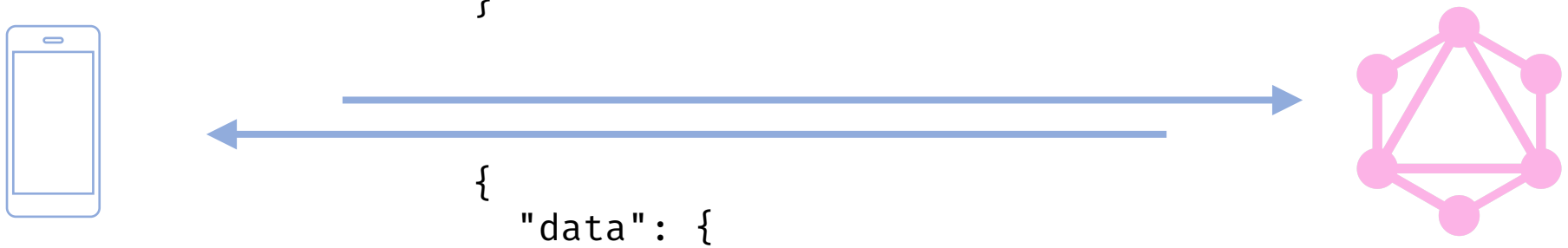
Queries are used to get data it needs from the server.

It's known as a **Read operation.**

```
type Query {
    personById(id: Int) : Person!
}
```

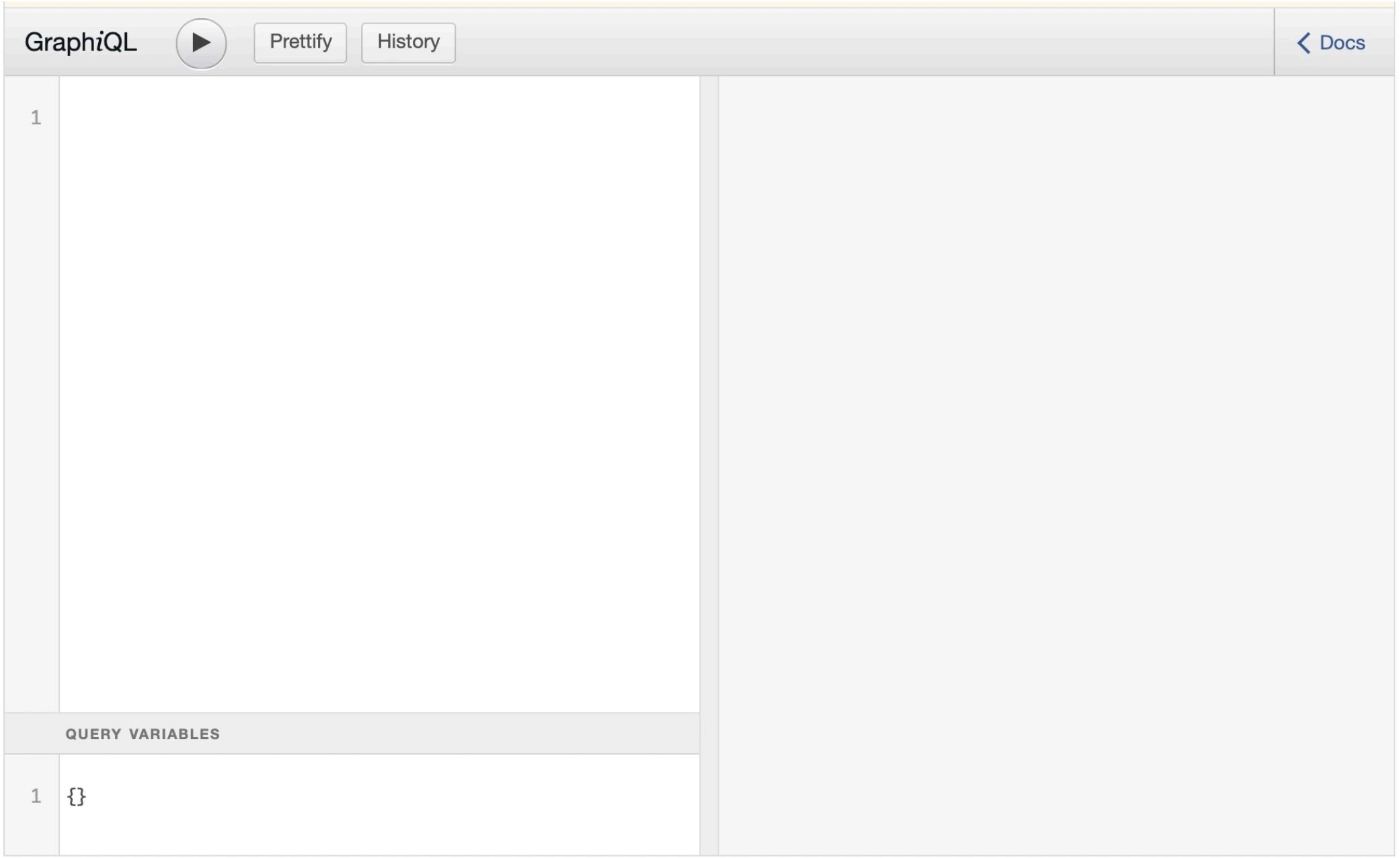
Querying in GraphQL

```
{
    personById(id: 1234) {
        name,
        age
    }
}
```



```
{
  "data": {
    "personById": {
      "name": "Lysandro",
      "age": 25
    }
  }
}
```

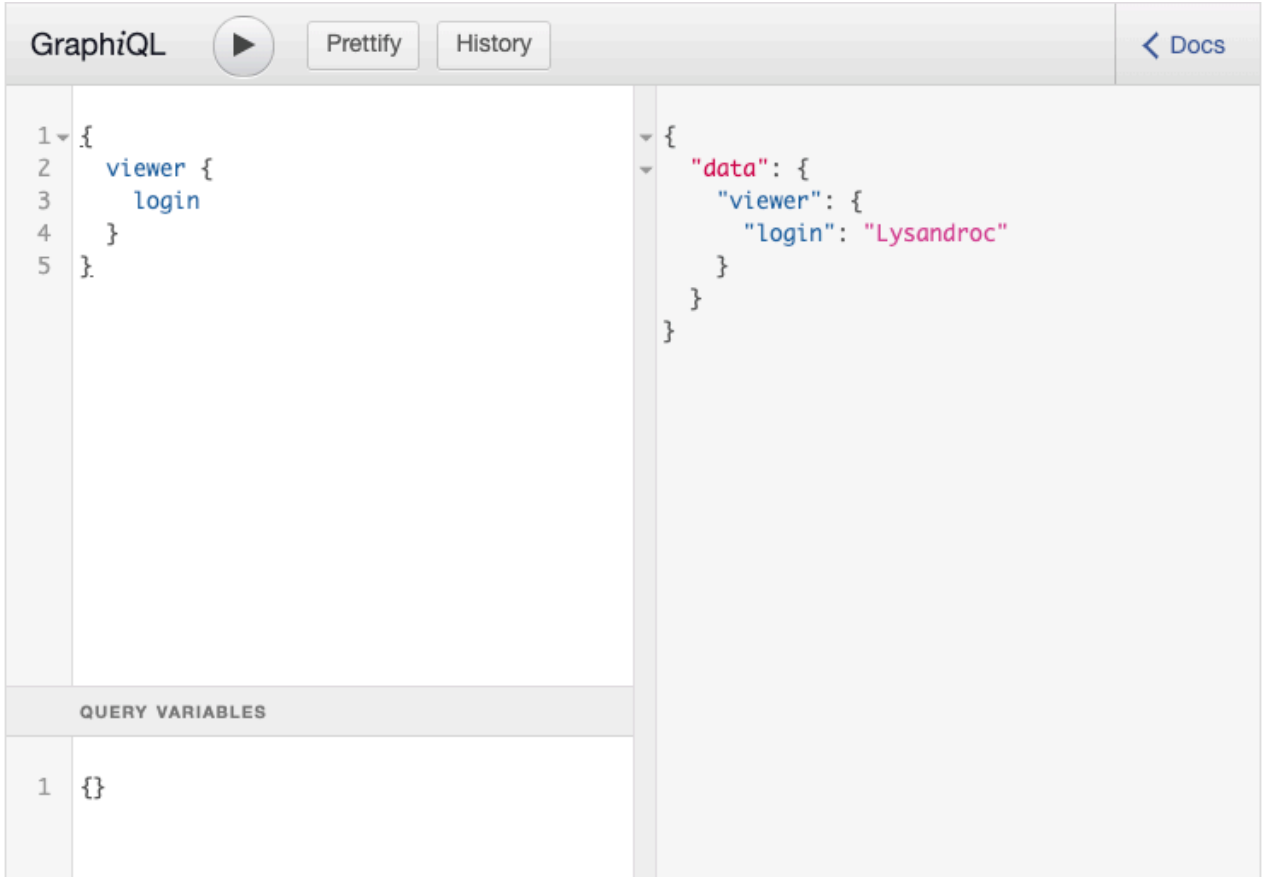
OSF Global Services



OSF Global Services

GraphiQL

GraphiQL offers syntax highlighting, code completion, and error warning, and it **lets you run and view query results directly in the browser.**



OSF Global Services

Mutations

Mutations are used to change data it needs from the server. It's known as a **Create, Update and Delete operation.**

```
type Mutation {
   createPerson(name: String!) : Person!
}
```

Mutating in GraphQL

```
mutation {
    createPerson(name: "Lysandro", age: 25) {
        Id
    }
}
```

```
{
  "data": {
    "createPerson": {
      “id”: "10b64872-5a2c-426f-872c-
    c138829bf525"
    }
  }
}
```

Subscriptions

Subscription allows us to listen to the GraphQL API for **real-time data changes.**

```
type Subscription {
    newPerson : Person!
}
```

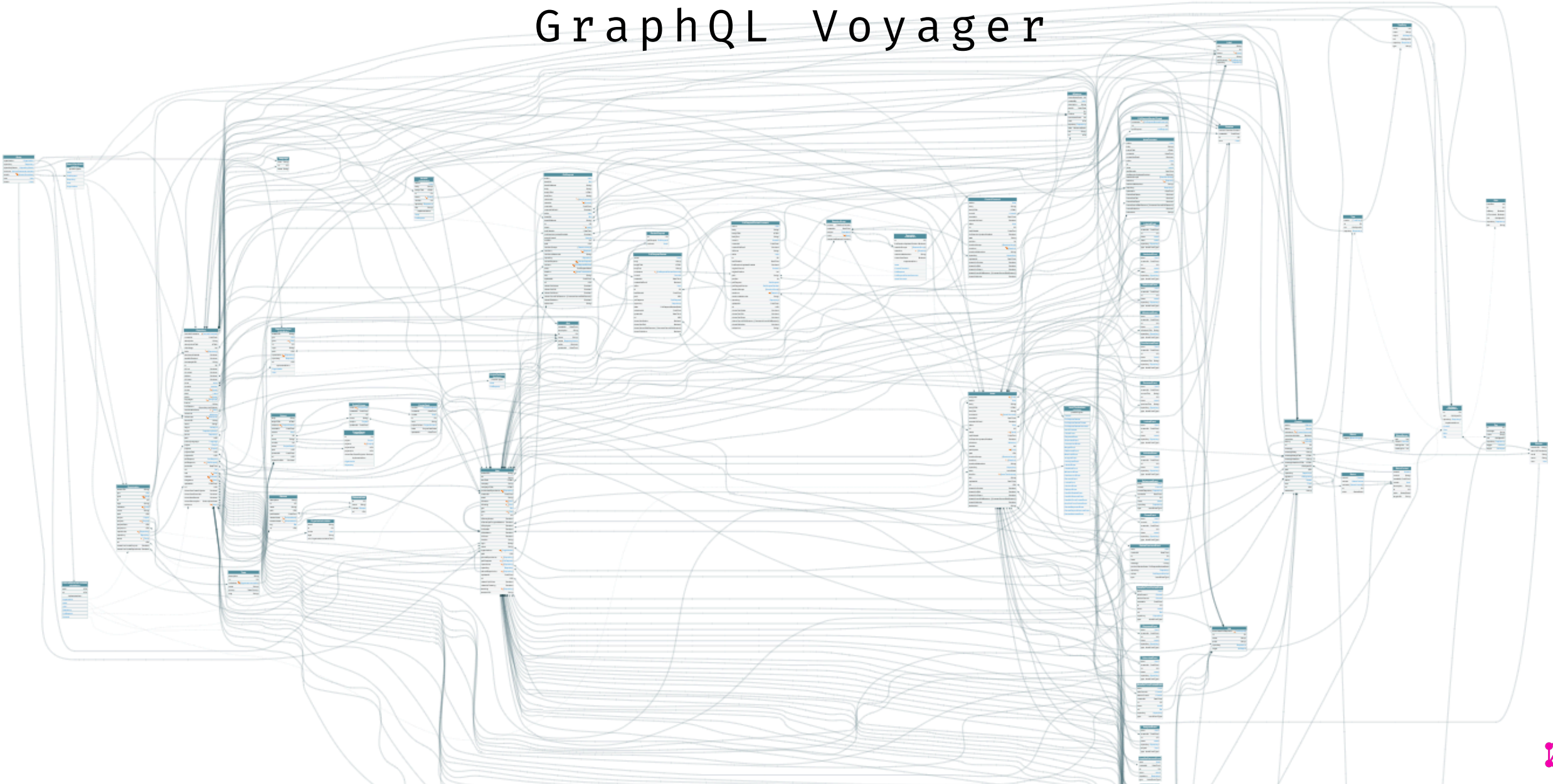
Introspection

Introspection is the ability to **query which resources are available in** the current **API schema**. We can see the queries, types, fields it supports.

```
{
    __schema {
        queryType {
            name
            description
        }
    }
}
```

OSF Global Services

GraphQL Voyager



OSF Global Services

Resolver

The principal unit of execution in GraphQL API is the *resolver*. It's a **Function with single purpose, providing data for single GraphQL Field.**

```
(root, args, context, info) ⇒ {
  //return data from anywhere
};
```

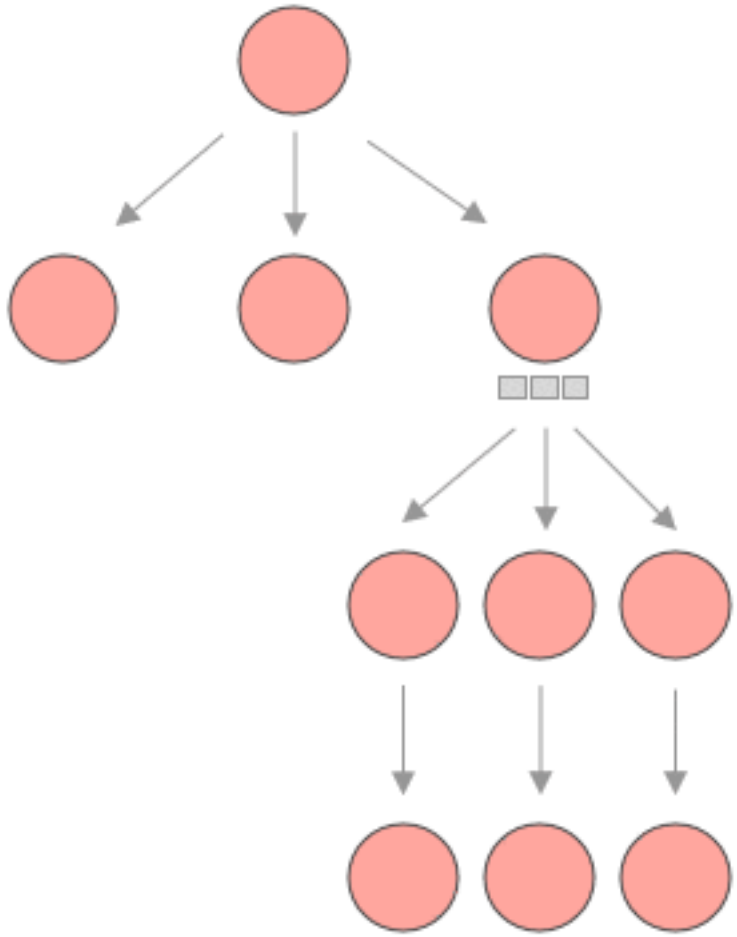
OSF Global Services

Defining a resolver

```
const resolvers = {
  Query: {
   personById: async (_, { id }) ⇒ await personRepo.findById(id);
  },
};
```

OSF Global Services

```
query {
  name
  age
  friends(first: 3) {
    bestFriend {
      name
    }
  }
}
```




```
type Query {

}

type Mutation {

}

type Subscription {

}
```

OSF Global Services

Sample Query Implementation

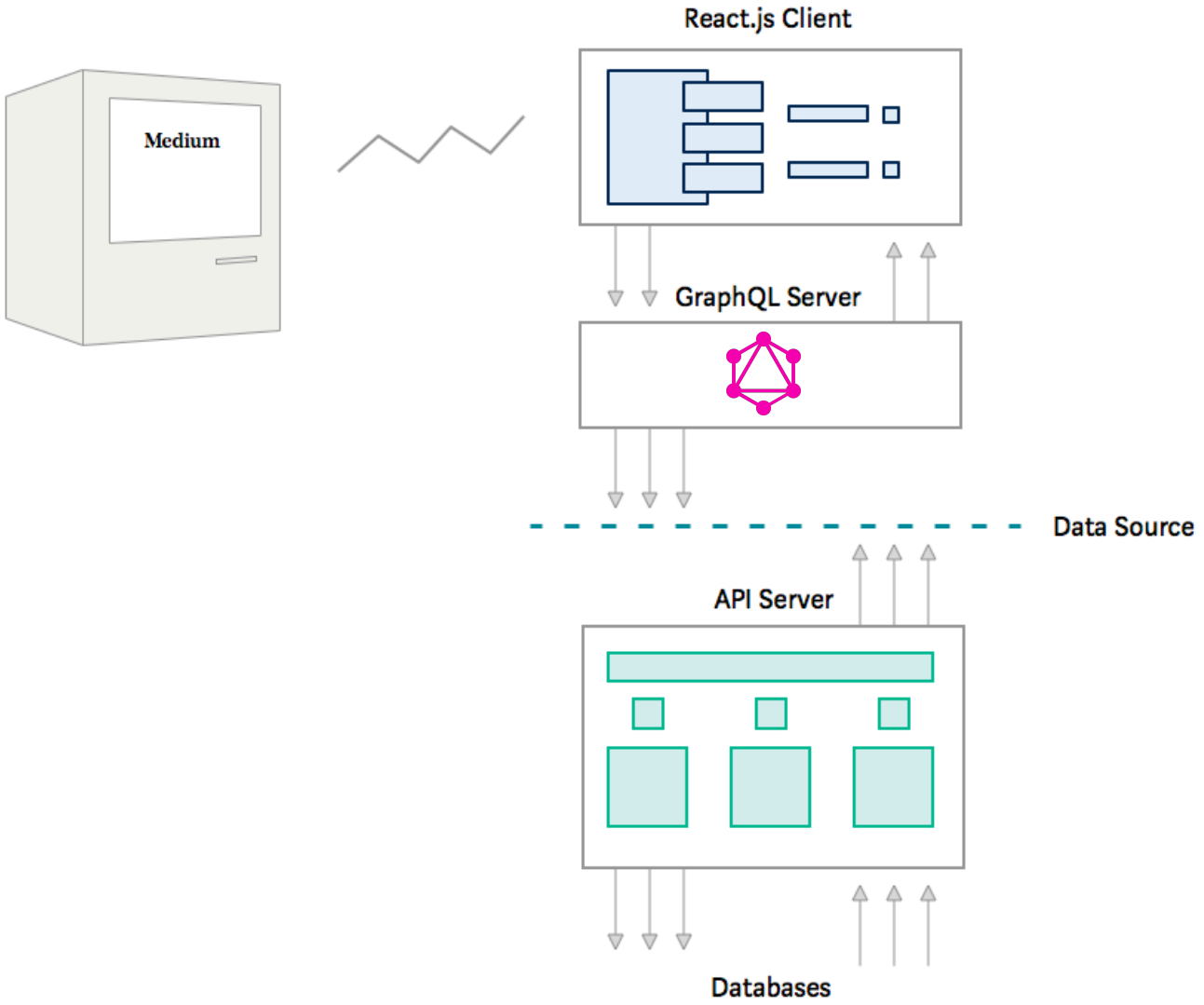
```
const typeDefs = `
 type Query {
   personById(id: Int) : Person!
 }

 type Person {
   id: ID!
   name: String!
   age: Int!
   Post: [Post]!
 }
`;
```

```
const resolvers =  {
  Query: {
    personById: async (root, { id }) ⇒
      await fetch(`{url}/{id}`);
  }
};
```

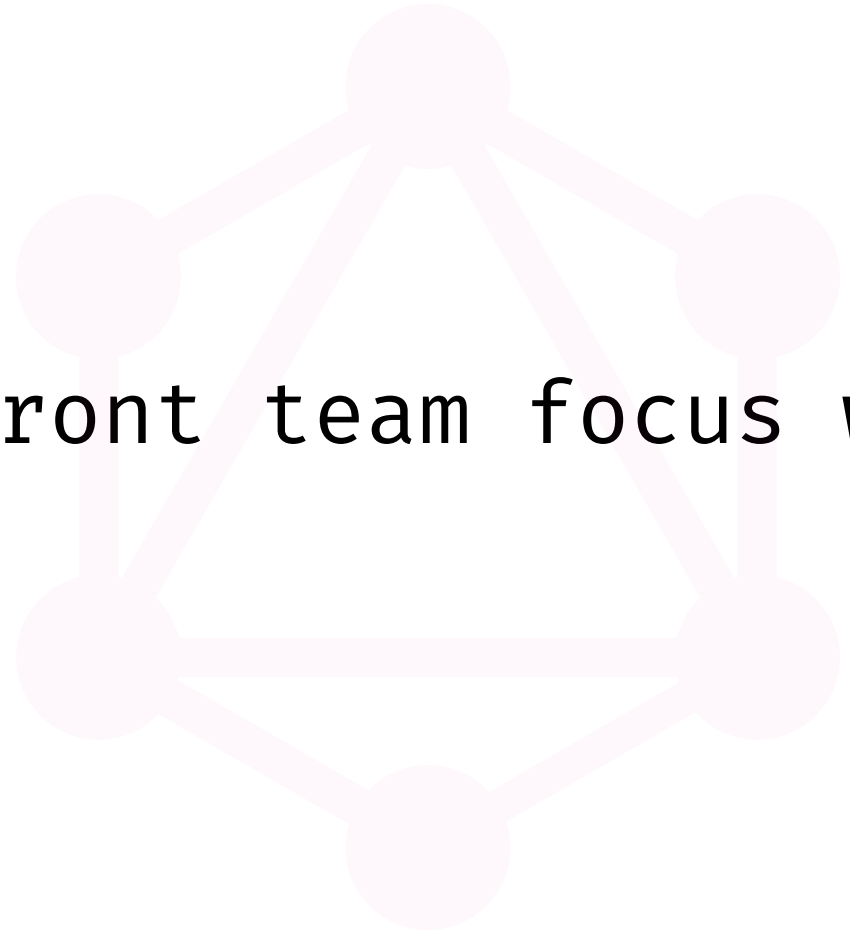
```
const server = new GraphQLServer({
  typeDefs,
  resolvers
});
server.start(options);
```

OSF Global Services

GraphQL Architecture



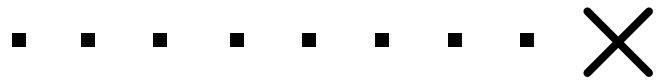
OSF Global Services



Let your front team focus what matter.

OSF | Global Services

THANK YOU



@lysandrocb

We are hiring!