Red Hat

API

What is GraphQL?

GraphQL is a query language and server-side runtime for application programming interfaces (APIs) that prioritizes giving clients exactly the data they request and no more.

GraphQL is designed to make APIs fast, flexible, and developer-friendly. It can even be deployed within an integrated development environment (IDE) known as GraphiQL. As an alternative to REST, GraphQL lets developers construct requests that pull data from multiple data sources in a single API call.

Additionally, GraphQL gives API maintainers the flexibility to add or deprecate fields without impacting existing queries. Developers can build APIs with whatever methods they prefer, and the GraphQL specification will ensure they function in predictable ways to clients.

7 keys to effective API management

Schemas, resolvers, and other common GraphQL terms

API developers use GraphQL to create a **schema** to describe all the possible data that clients can query through that service.

A GraphQL schema is made up of object types, which define which kind of object you can request and what fields it has.

As **queries** come in, GraphQL validates the queries against the schema. GraphQL then executes the validated queries.

The API developer attaches each field in a schema to a function called a **resolver**. During execution, the resolver is called to produce the value.

Apart from defining and validating syntax for API queries (outlined in the graphql-spec repository), GraphQL leaves most other decisions to the API designer. GraphQL does not provide any direction for how to store data or what programming language to use—developers can use PHP (graphql-php), Scala (Sangria), Python (Graphene Python), Ruby (graphql-ruby), JavaScript (graphql.js), and more. GraphQL offers no requirements for the network, authorization, or pagination.

From the point of view of the client, the most common GraphQL operations are likely to be **queries** and **mutations**. If we were to think about them in terms of the *create, read, update and delete* (CRUD) model, a query would be equivalent to *read*. All the others (*create, update,* and *delete*) are handled by mutations.
API design best practices

Advantages and disadvantages of GraphQL in corporate environments

Thinking about trying GraphQL in a business or enterprise environment? It comes with both pros and cons.

Advantages

- A GraphQL schema sets a single source of truth in a GraphQL application. It offers an organization a way to federate its entire API.
- GraphQL calls are handled in a single round trip. Clients get what they request with no overfetching.
- Strongly defined data types reduce miscommunication between the client and the server.
- GraphQL is introspective. A client can request a list of data types available. This is ideal for auto-generating documentation.
- GraphQL allows an application API to evolve without breaking existing queries.
- Many open source GraphQL extensions are available to offer features not available with REST APIs.
- GraphQL does not dictate a specific application architecture. It can be introduced on top of an existing REST API and can work with existing API management tools.

Disadvantages

- GraphQL presents a learning curve for developers familiar with REST APIs.
- GraphQL shifts much of the work of a data query to the server side, which adds complexity for server developers.
- Depending on how it is implemented, GraphQL might require different API management strategies than REST APIs, particularly when considering rate limits and pricing.
- Caching is more complex than with REST.
- API maintainers have the additional task of writing maintainable GraphQL schema.

Tips to incorporate APIs throughout your company

An example GraphQL query

The best way to appreciate GraphQL is to look at some sample queries and responses. Let's look at 3 examples adapted from the GraphQL project website, graphql.org.

The first example shows how a client can construct a GraphQL query, asking an API to return specific fields in a shape you've specified.

```
{
  me {
    name
  }
}
```

A GraphQL API would return a result like this in JSON format:

```
{
  "me": {
    "name": "Dorothy"
  }
}
```

A client can also pass arguments as part of a GraphQL query, as seen in this example:

```
{
  human(id: "1000") {
    name
    location
  }
}
```

The result:

```
{
  "data": {
    "human": {
      "name": "Dorothy,
      "location": "Kansas"
    }
  }
}
```

From here, things get more interesting. GraphQL gives users the ability to define reusable fragments and assign variables.

Suppose you need to request a list of IDs, then request a series of records for each ID. With GraphQL, you could construct a query that pulls everything you want with a single API call.

So this query:

```
query HeroComparison($first: Int = 3) {
  leftComparison: hero(location: KANSAS) {
    ...comparisonFields
  }
  rightComparison: hero(location: OZ) {
    ...comparisonFields
  }
}

fragment comparisonFields on Character {
  name
  friendsConnection(first: $first) {
    totalCount
    edges {
      node {
        name
      }
    }
  }
}
```

Might produce this result:

```
{
  "data": {
    "leftComparison": {
      "name": "Dorothy",
      "friendsConnection": {
        "totalCount": 4,
        "edges": [
          {
            "node": {
              "name": "Aunt Em"
            }
          },
          {
            "node": {
              "name": "Uncle Henry"
            }
```


GraphQL and open source

GraphQL was developed by Facebook, which first began using it for mobile applications in 2012. The GraphQL specification was open sourced in 2015. It is now overseen by the GraphQL Foundation.

There are a variety of open source projects that involve GraphQL. The list below is not exhaustive, but includes projects designed to facilitate the adoption of GraphQL.

- Apollo, a GraphQL platform that includes a frontend client library (Apollo Client) and backend server framework (Apollo Server).
- Offix, an offline client that allows GraphQL mutations and queries to execute even when an application is unreachable.
- Graphback, a command line-client for generating GraphQL-enabled Node.js servers.
- OpenAPI-to-GraphQL, a command-line interface and library for translating APIs described by OpenAPI Specifications or Swagger into GraphQL.

Here's a guide to deploying APIs

Get a 60-day self-serve experience to explore the benefits of a fully managed API service.

Get started

Featured resource

E-book: What is an API?

Read more about APIs

- Topic: Understanding APIs
- Article: What is an API?
- Article: What is API management?
- Article: What is API design?
- Article: What can you do with APIs?
- Article: What is API security?
- Article: Why choose Red hat for API management?

APIs and Red Hat

Make it easy to share, secure, distribute, control, and monetize your APIs for internal or external users.

Learn more

A distributed, cloud-native integration platform that connects APIs—on-premise, in the cloud, and anywhere in between.

Learn more

Red Hat OpenShift API Management

A hosted and managed API management service delivered as an add-on to Red Hat OpenShift Dedicated.

Learn more

Get started with Red Hat OpenShift API Management

Try it | **Talk to a Red Hatter**

ABOUT

We're the world's leading provider of enterprise open source solutions, using a community-powered approach to deliver high-performing Linux, cloud, container, and Kubernetes technologies. We help you standardize across environments, develop cloud-native applications, and integrate, automate, secure, and manage complex environments with award-winning support, training, and consulting services.

PRODUCTS

Red Hat Ansible Automation Platform

Red Hat Enterprise Linux

Red Hat OpenShift

Red Hat OpenShift Data Foundation

Red Hat OpenStack Platform

See all products

TOOLS

My account

Customer support

Partner resources

Developer resources

Training and certification

Red Hat Ecosystem Catalog

Resource library

TRY, BUY, SELL

Product trial center

Red Hat Store

Red Hat Marketplace

Find a partner

Contact sales

Contact training

Contact consulting

COMMUNICATE

Contact us

Feedback

Social

Red Hat newsletter