IBM StepZen Graph Server

Build, optimize and scale GraphQL APIs quickly and easily

Highlights

Build APIs quickly

Create foundational building blocks with one line of code using introspective techniques

Get high performance effortlessly with an automatic in-memory query optimization engine

Access diverse data sources easily and confidently with multiple safeguards

API abstraction enables backend sources to change without breaking API consumers

The adoption of new systems and tools in recent years has led to data being spread across multiple silos. Application delivery is often impeded because developers lack access to the correct data when they need it. This has led to an explosion in the usage of APIs, but we often see progress hindered because the right API is unavailable.

Siloed data across multiple systems force application developers to deal with different protocols, structures, authentications, performance and more—these activities drain productivity and add little value.

What is needed is a universal, federated API that can be built to abstract the backend complexities and deliver a simple, intuitive, consistent view to the application developer. GraphQL is emerging as the new standard for APIs that delivers this consistent and intuitive view. It layers upon investments in REST APIs and data systems.

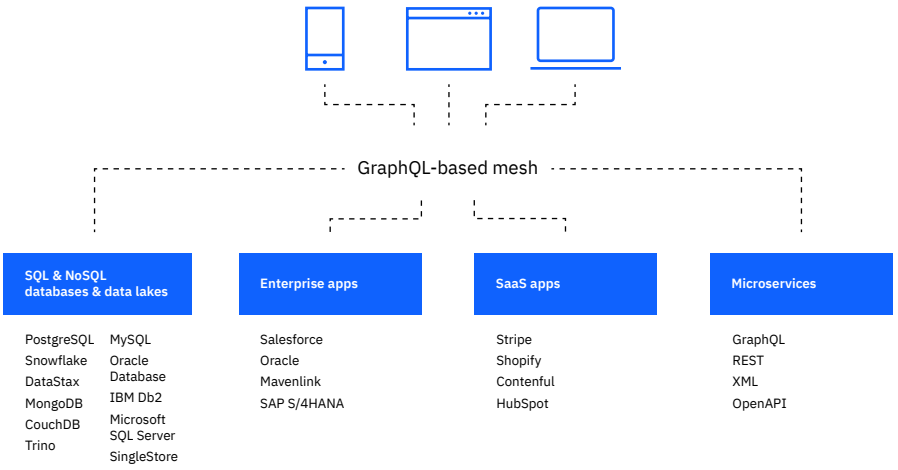

Figure 1. GraphQL unlocks siloed data for consumption.

GraphQL makes it significantly easier for application teams to consume data, however it represents a challenge to API teams who are tasked with building it. The team not only has to build the API, but deploy, protect, optimize, evolve and scale it. This process can take months and often requires specialized skills—such as GraphQL query optimization, that many API teams don't have in abundance.

IBM StepZen Graph Server enables the rapid creation of a data mesh across enterprise data—SOAP XML, REST, SQL or NoSQL backends—and an execution runtime for the data mesh with low latency, high throughput and security baked in. The secret is declarative APIs. Similar to how databases changed the application landscape, from requiring programmatic data manipulation to declarative creation and access, IBM StepZen is doing the same to the GraphQL API landscape. With declarative APIs, the code is more straightforward and faster to write. IBM StepZen Graph Server's underlying runtime automatically handles deployment, protection, scale and optimization so that specialized skills aren't required for success.

What we do

IBM StepZen Graph Server is a GraphQL server with a special architecture that enables developers to build their APIs quickly using declarative configurations. Because these APIs run in a Golang-based in-memory GraphQL engine and are deployed in Kubernetes, they're highly responsive to the application's needs. With IBM StepZen, you'll build clean, consistent APIs using declarative building blocks, while IBM StepZen Graph Server takes care of scaling and optimizing them.

Developers build graphs by composing declarative building blocks. They build foundational building blocks by connecting to backends, linking those building blocks together in a single graph and building supergraphs by composing GraphQL graphs. With powerful introspection capabilities, the GraphQL model for enterprise data can be constructed and composed with only a few lines of code.

Since the API is built declaratively, IBM StepZen Graph Server can understand the API and make the right optimization decisions at runtime, resulting in low latency and high throughput without changing the backend. In addition, IBM StepZen Graph Server is delivered as a cloud service which can connect to public, private or on-premises data sources and services.

Building the API
In StepZen, a GraphQL API is composed using declarative building blocks. Each building block connects to a data source and defines a mapping to the GraphQL type system. There are three directives used to declare the foundational building blocks:

- **`@rest`** building block connects to a REST, SOAP and OData backend, or a NoSQL database.
- **`@dbquery`** building block connects to a SQL backend.
- **`@graphql`** building block connects to a GraphQL backend, enabling connections to systems like Shopify that expose a GraphQL interface, and the building of a graph of graphs in larger organizations.

Another directive handles the composition of these building blocks:

- **`@materializer`** links building blocks, enabling the developer to express a request for all related data within a single request, regardless of the source backend.

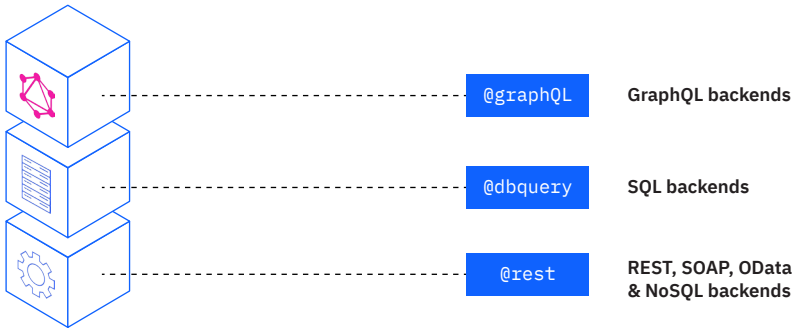

Figure 2. StepZen uses three directives to declare the foundational building blocks.

IBM StepZen Graph Server also has powerful import capabilities using sophisticated introspection techniques to generate schemas from backends. This mechanism often reduces the task of creating foundational building blocks to one line of code, for example **`stepzen import curl`** or **`stepzen import database`**.

IBM has deep experience with APIs and API management, and IBM StepZen Graph Server has been exposed to many of the best practices for APIs. We believe that:

- APIs must be designed for consumption and, therefore, curated.
- The curated APIs must offer the flexibility for the API consumer to ask for specific pieces of data, expressed as query.
- The consumption syntax should be as simple and intuitive as possible.
- The GraphQL API exposed to developers should be consistent across all backends, so they don't need to be aware of the backend source to express their query.

Optimizations and scale
We know that a significant advantage of a declarative approach is the ability to optimize the execution of the queries. IBM StepZen Graph Server has built an in-memory, Golang-based GraphQL query optimization engine that does all the following for developers:

- Autoparallelism
- Pushdowns
- Caching at various levels, controllable by configuration
- N+1 optimizations
- Memorization

With standard GraphQL libraries, the development team must perform all of these optimizations; however, with IBM StepZen Graph Server, they can do it without having to write a single line of code.

Security
GraphQL makes data access easier—we believe that it "makes data liquid." However, with it comes the challenge of helping to ensure that data isn't exposed when it shouldn't be. We take security of your data seriously and offer the following:

- TLS 1.2+ helps protect all inbound and outbound traffic.
- API keys and JWT help protect access to queries and mutations. Developers can write simple rules that check against the claims and allow or disallow access to the queries and mutations. These rules can check for many things, such as valid query arguments, roles and more.
- When accessing backend databases, IBM StepZen Graph Server prepares the statement before execution, leading to zero overhead because the preparation is done once and cached.
- When accessing HTTP backends, IBM StepZen Graph Server supports the protection mechanism the backends uses—for example, key-based, basic auth, OAuth or JWT. In addition, it passes the JWT context down to the respective backends to provide further checks and balances.

Change and the development process
How do developers manage the inevitable changing backends, the need to evolve GraphQL APIs and the need for new protection rules? They do so with their continuous integration continuous delivery (CI/CD) process.

Now, imagine that the API was just code, written in a favorite GraphQL library, with resolvers as functions that are opaque to any analysis. How would the developer perform an impact analysis for their change? It would be nearly impossible. This is why a declarative approach works much better. If a **`type Customer`** is returned by a **`@rest`** call, it's easy to know exactly what may be affected if the backend changes. IBM StepZen Graph Server takes it further. Since **`stepzen import`** is a core part of how the API is set up, IBM StepZen Graph Server supports all changes made during the import step in a separate **`extend`** file, which clearly identifies the deltas from the import. Now, when one reimports for any reason, all the previous work isn't lost; it can be reapplied.

Now, imagine building the API with a graphical user interface (GUI). It may somewhat simplify the initial build but messes up the CI/CD process. In IBM StepZen Graph Server, all building blocks and compositions are text files, and deployments are executed through are through the command-line interface (CLI).

Federation

Companies typically want to compartmentalize their GraphQL API development as teams and applications expand. As a result, individual teams develop GraphQL for their functions and data domains that must be brought together or federated to form the organization's data mesh. IBM StepZen Graph Server believes that choice of implementation is important, which is why we support two declarative mechanisms for building federated graphs.

- Compose a supergraph by combining subgraphs using **`@graphql`** and **`@materializer`**. This step is relatively trivial to complete and maintains the independence of the underlying graphs. It doesn't matter whether the graphs being combined are built in IBM StepZen Graph Server or elsewhere.
- Compose a supergraph in Apollo, which has a different form of declarative composition using object-based stitching. Graphs that are built or proxied in IBM StepZen Graph Server using **`@graphql`**, are Apollo Federation compliant. Simply specify the **`@key`** annotation and IBM StepZen Graph Server takes care of the rest, **`creating_service`**, **`_entities`**, even **`@link`**, **`@required`**, and more.

In IBM StepZen Graph Server, our clients are building enterprise-wide GraphQL APIs, and GraphQL APIs that are federated into an enterprise-wide GraphQL API in Apollo. As an enterprise, you can decide which federation model is right, and IBM StepZen Graph Server will support that model. For more, see Declarative GraphQL Federation.

GraphQL as a service

It has been said, "Success is 1% inspiration and 99% perspiration." In this world, the aha moment of getting a GraphQL API up and running is inspiring, but then the hard work of keeping it running, scaling and error-free begins. Since IBM StepZen Graph Server is available as a service it allows you to easily scale and run more services quickly. As a result, your developer teams do not have to worry about downtimes, scaling or errors, and can devote their time to more value-added work, such as solving their enterprise's next business problem.

As a cloud service, IBM StepZen Graph Server delivers more than 99.99% availability, scales at over 5,000 transactions per second (TPS), and latencies of less than 10 milliseconds (ms),[1] without the developers having to manage infrastructure, and with full network address translation (NAT) support for access to private endpoints. IBM StepZen Graph Server is delivered as a cloud service which can connect to public, private or on-premises data sources and services. For more information, see the StepZen website: Run GraphQL on a Golang-based Query Optimization Engine.

Conclusion

IBM StepZen Graph Server is bringing the world of database techniques to the world of APIs:

- We believe that backends that deliver data to GraphQL are diverse and the most precious resources.
- IBM StepZen Graph Server delivers a building block assembly of declarative GraphQL building blocks. This method results in a less code and much faster time to value.
- Because the API is built declaratively, we have built out a special query optimization engine that's more than just a router.
- IBM StepZen Graph Server has simple declarative constructs to secure the GraphQL API.
- With the declarative approach and "everything as text files," APIs built in IBM StepZen Graph Server fit easily and naturally into the CI/CD pipelines.
- In large enterprises, a "graph of graphs" is the right structure for GraphQL and the entire API strategy. IBM StepZen Graph Server enables these supergraphs to be built using declarations in IBM StepZen, or with other alternative technologies like Apollo.
- Finally, IBM StepZen Graph Server's architecture is designed to be high throughput and low latency. You can consume GraphQL as a fully managed SaaS.

Data silos is a universal problem and it cuts across industries. GraphQL is emerging as a new standard for a universal, federated API to provide application developers with a simple, intuitive, and consistent view of data. IBM StepZen Graph Server enables teams to create GraphQL APIs across enterprise data quickly and efficiently. Our focus is on enabling developers to create and query APIs declaratively and on delivering a runtime that automatically handles deployment, protection, scale and optimization.

For more information

For more information, see stepzen.com and stepzen.com/docs.

1. https://stepzen.com/product/run-serverless

IBM Corporation
New Orchard Road
Armonk, NY 10504

Produced in the
United States of America
June 2023