Continuous Evolution of GraphQL Schemas
Marc Giroux 🇨🇦

Montreal, Canada 🇨🇦

@__xuorig__
Continuous Evolution?
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.

| type Film { |
| title: String |
| episode: Int |
| releaseDate: String |
| openingCrawl: String |
| director: String |
| directedBy: Person |
| } |

| type File { |
| title: String |
| episode: Int |
| releaseDate: String |
| openingCrawl: String |
| director: String |
| @deprecated |
| directed: Person |
| } |

| type Person { |
| name: String |
| directed: [File] |
| actedIn: [File] |
| } |

| type Person { |
| name: String |
| directed: [File] |
| actedIn: [File] |
| } |
REST

What is the best practice for versioning a REST API?

DON'T

Versioning an interface is just a "polite" way to kill deployed applications.
The Sunset HTTP Header

draft-wilde-sunset-header-04

Abstract

This specification defines the Sunset HTTP response header field, which indicates that a URI is likely to become unresponsive at a specified point in the future. It also defines a sunset link relation type that allows linking to resources providing information about an upcoming resource or service sunset.
API evolution is the concept of striving to maintain the “I” in API, the request/response body, query parameters, general functionality, etc., only breaking them when you absolutely, absolutely, have to.

- PHIL STURGEON
Our Mission 🚀
1. Avoid breaking changes the best we can.
2. Make it easy for engineers to evolve our schemas.
3. Make it easy for users to evolve their clients.
Avoiding Breaking Changes
Accidental Breaking Changes 😳
Checked-in GraphQL IDL

```
$ bin/dump-graphql-schema
> config/schema.public.graphql

$ git add config/schema.public.graphql
```
# A user is an individual's account on GitHub that owns repositories and can make new content.

```typescript
interface User implements Actor & Node & RepositoryOwner & UniformResourceLocatable {
  avatarUrl()
    // The size of the resulting square image.
    size: Int
    // URI!

  bio: String

  // The user's public profile bio as HTML.
  bioHTML: HTML!

  commits{
    // A list of commit comments made by this user.
    commitComments{
      // Returns the elements in the list that come after the specified global ID.
      after: String

      // Returns the elements in the list that come before the specified global ID.
      before: String

      // Returns the first _n_ elements from the list.
      first: Int

      // Returns the last _n_ elements from the list.
      last: Int
    }:
    // CommitCommentConnection
  }
```
Checked-in GraphQL IDL

github-lint-schema

- **Failure**  Re-run

- built **just now in 2 minutes**
- 45cc31c by @xuorig
```ruby
lib/platform/objects/pull_request_review_comment.rb

@@ -58,8 +58,6 @@ def self.async_viewer_can_see?(permission, object)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>58</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td></td>
</tr>
</tbody>
</table>

- pr_review_comment.async_update_path_uri
- end

- field :pull_request, Objects::PullRequest, method: :async_pull_request, description: "The pull request associated with the comment.", null: false

- field :pull_request_review, PullRequestReview, method: :async_pull_request_review, description: "The pull request review associated with the comment.", null: false

- field :commit, Commit, description: "Identifies the commit associated with the comment.", null: false
```
<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7278</td>
<td># Identifies when the comment was published at.</td>
</tr>
<tr>
<td>7279</td>
<td>publishedAt: DateTime</td>
</tr>
<tr>
<td>7280</td>
<td></td>
</tr>
<tr>
<td>7281</td>
<td># The pull request associated with this review comment.</td>
</tr>
<tr>
<td>7282</td>
<td>pullRequest: PullRequest!</td>
</tr>
<tr>
<td>7283</td>
<td></td>
</tr>
<tr>
<td>7284</td>
<td># The pull request review associated with this review comment.</td>
</tr>
<tr>
<td>7285</td>
<td>pullRequestReview: PullRequestReview</td>
</tr>
<tr>
<td>7286</td>
<td></td>
</tr>
<tr>
<td>File</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>bin</td>
<td>improved CLI</td>
</tr>
<tr>
<td>lib/graphql</td>
<td>6.0</td>
</tr>
<tr>
<td>test</td>
<td>Better test coverage</td>
</tr>
<tr>
<td>.gitignore</td>
<td>gitignore</td>
</tr>
<tr>
<td>.travis.yml</td>
<td>Update travis.yml</td>
</tr>
<tr>
<td>CHANGELOG.md</td>
<td>6.0</td>
</tr>
<tr>
<td>CODE_OF_CONDUCT.md</td>
<td>First</td>
</tr>
<tr>
<td>Gemfile</td>
<td>First</td>
</tr>
<tr>
<td>LICENSE.txt</td>
<td>First</td>
</tr>
<tr>
<td>README.md</td>
<td>Update Readme.md</td>
</tr>
<tr>
<td>Rakefile</td>
<td>Add Result tests</td>
</tr>
</tbody>
</table>
COMPARE(SCHEMA_A, SCHEMA_B) => SET<CHANGE>
$ bin/testrb test/fast/linting/graphql_breaking_changes_test.rb
Run options: --seed 54244

# Running:

E

Finished in 2.333004s, 0.4285 runs/s, 0.0000 assertions/s.

1) Error:
GraphQLBreakingChangesTest#test_no_breaking_changes:
RuntimeError:

🚨🚨🚨 *** BREAKING CHANGES DETECTED! *** 🚨🚨🚨

Making a breaking change to our GraphQL schema can potentially impact all of our API integrators.
Our GraphQL schema is public to the world, and it looks like you've made an irreversible change.
Please make sure to get explicit 👍 from @github/ecosystem-api before continuing.
For more information, read https://graphql-docs.githubapp.com/guides/schema-evolution

* Field `pullRequest` was removed from object type `PullRequestReviewComment`
  /Users/xuorig/github/github/test/fast/linting/graphql_breaking_changes_test.rb:169:in `test_no_breaking_changes'

1 runs, 0 assertions, 0 failures, 1 errors, 0 skips
GraphQL Doctor
It looks like your pull request includes changes to our GraphQL schema that requires your attention. Our GraphQL Style Guide also includes information that might be helpful.

Happy shipping!

```
<table>
<thead>
<tr>
<th>config/schema.public.graphql</th>
</tr>
</thead>
<tbody>
<tr>
<td>... ...</td>
</tr>
<tr>
<td>7309 7309</td>
</tr>
<tr>
<td>7310 7310</td>
</tr>
<tr>
<td>7311 7311</td>
</tr>
<tr>
<td>7312 -</td>
</tr>
<tr>
<td>7313 -</td>
</tr>
<tr>
<td>@ type PullRequestReviewComment implements Comment &amp;</td>
</tr>
<tr>
<td># Identifies when the comment was published at.</td>
</tr>
<tr>
<td>publishedAt: DateTime</td>
</tr>
<tr>
<td>pullRequest: PullRequest!</td>
</tr>
</tbody>
</table>
```

Removing a field is a breaking change. It is preferable to deprecate the field before removing it.
Have-to Breaking Changes
Prevention First!
Evolvable Schema Design
# Style Guide

## Table of Contents

1. Naming
2. Object Types
3. Enum Types
4. Abstract Types
# Ordering options for projects returned from the connection

```javascript
orderBy: ProjectOrder
```

github-graphql-doctor [bot] 12 days ago

Adding an `orderBy` argument with a default value to fields that return a `Connection` type is usually preferred over setting an undocumented default that integrators need to guess.
Common Design Issues
Scalar $\Rightarrow$ Object Type
Null $\Rightarrow$ Non-Null 😡
Make it **easy** for **engineers** to evolve our schemas
What GraphQL gives us
3.13.3 @deprecated

directive @deprecated(
    reason: String = "No longer supported"
) on FIELD_DEFINITION | ENUM_VALUE

The @deprecated directive is used within the type system definition language to indicate deprecated portions of a GraphQL service’s schema, such as deprecated fields on a type or deprecated enum values.

Deprecations include a reason for why it is deprecated, which is formatted using Markdown syntax (as specified by CommonMark).

In this example type definition, `oldField` is deprecated in favor of using `newField`.

```
Example № 85
type ExampleType {
    newField: String
    oldField: String @deprecated(reason: "Use `newField`.")
}
```
type ExampleType {
    newField: String
    oldField: String @deprecated(reason: "Use `newField`.")
}

Problem: free form text, hard to be consistent.

type ExampleType {
  newField: String
  oldField: String @deprecated(reason: "Use `newField`."
}

Ruby API

defprecated(
    start_date: Date.new(2018, 1, 15),
    reason: "We do not use databaseID anymore.",
    superseded_by: "Use `Node.id` instead."
)
type Repository {
  databaseId: ID! @deprecated(reason: "databaseId will be removed. Use Node.id instead. Removal on 2017-07-01 UTC.")
}
3.13.3 @deprecated

directive @deprecated(
    reason: String = "No longer supported"
) on FIELD_DEFINITION | ENUM_VALUE

The @deprecated directive is used within the type system definition language to indicate deprecated portions of a GraphQL service's schema, such as deprecated fields on a type or deprecated enum values.

Deprecations include a reason for why it is deprecated, which is formatted using Markdown syntax (as specified by CommonMark).

In this example type definition, oldField is deprecated in favor of using newField.

Example № 85

type ExampleType {
    newField: String
    oldField: String @deprecated(reason: "Use `newField`.")
}

Another Problem: Only fields and enum values...

type Repository {
  issues(states: [IssueState!]) @deprecated(reason: "invalid"): [Issue!]!
}

Invalid according to spec
Fake It Till You Make It™

type Repository {
  issues(
    # Deprecated: `states` will be removed. Use `newStates` instead. Removal on 2018-07-01 UTC.
    states: [IssueState!]
  ): [Issue!]!
}

Making the change
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7278</td>
<td>7278</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7279</td>
<td>7279</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7280</td>
<td>7280</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7281</td>
<td>7281</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7282</td>
<td>7282</td>
<td>-</td>
<td></td>
<td>pullRequest: PullRequest!</td>
<td></td>
</tr>
<tr>
<td>7283</td>
<td>7283</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7284</td>
<td>7281</td>
<td></td>
<td></td>
<td></td>
<td>pullRequestReview: PullRequestReview</td>
</tr>
<tr>
<td>7285</td>
<td>7282</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7286</td>
<td>7283</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- `publishedAt: DateTime`

# Identifies when the comment was published at.

- `pullRequest: PullRequest!`

# The pull request associated with this review comment.

# The pull request review associated with this review comment.
Who is using this field?
Has this field been used in past $x$ days?
query {
  viewer {
    pullRequests {
      title
      description
    }
  }
}
<query> → GitHub API → kafka
Query Analytics
query {
  viewer {
    pullRequests {
      title
      description
    }
  }
}

Query Analytics
Query Analytics

Type: Query
Field: viewer
App: xxx

Type: User
Field: pullRequests
App: xxx
I have found the things
.graphql usage Repository.name --schema public --since 5

hubot  9:02 PM
One moment while I crunch these numbers for you @xuorig. I'll ping you when the results are ready!
I am running 1 queries for you:

service.github.net/analytics_queries/195

Your results are ready @xuorig! GraphQL Analytics Query #195, with schema PUBLIC_SCHEMA since 2018-09-18 01:02:33 UTC:

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td></td>
<td>123456</td>
</tr>
</tbody>
</table>

Message @hubot
It looks like your pull request includes some risky changes to our GraphQL schema.

I've commented on those lines to let you know the impact of those changes. If you have questions, please ping @github/graphql-api-code-review.

Our GraphQL Style Guide also includes information that might be helpful.

Happy shipping! 🎉 🎈

```graphql
schema.graphql
... ... @@ -14,7 +14,7 @@ type Query {
  ... ...
  type Repository implements Node {
    homepageUri: URI
    id: ID!
    - nameWithOwner: String!
  }

graphql-doctor-dev bot 7 days ago
Removing a field is a breaking change. It is preferable to deprecate the field before removing it.

The Repository.nameWithOwner field has been used 666 times in the past day.
```
Make it **easy** for **users** to evolve their clients.
GitHub GraphQL API Schema Changes

The GraphQL schema change log is a list of recent and upcoming changes to our GraphQL API schema. It includes backwards-compatible changes, schema previews, and upcoming breaking changes.

**Breaking changes** include changes that will break existing queries or could affect the runtime behavior of clients. For a list of breaking changes and when they will occur, see our breaking changes log.

GraphQL Schema Changes for 2018-09-19

September 19, 2018  hubot

The Issues Preview preview includes these changes:

- Field `convertProjectCardNoteToIssue` was added to object type `Mutation`
- Type `ConvertProjectCardNoteToIssueInput` was added
- Type `ConvertProjectCardNoteToIssuePayload` was added
Breaking Changes

1. Changes scheduled for 2019-01-01

Breaking changes are any changes that might require action from our integrators. We divide these changes into two categories:

- **Breaking**: Changes that will break existing queries to the GraphQL API. For example, removing a field would be a breaking change.
- **Dangerous**: Changes that won’t break existing queries but could affect the runtime behavior of clients. Adding an enum value is an example of a dangerous change.

We strive to provide stable APIs for our integrators. When a new feature is still evolving, we release it behind a schema preview.

We’ll announce upcoming breaking changes at least three months before making changes to the GraphQL schema, to give integrators time to make the necessary adjustments. Changes go into effect on the first day of a quarter (January 1st, April 1st, July 1st, or October 1st). For example, if we announce a change on January 15th, it will be made on July 1st.

Changes scheduled for 2019-01-01

- **Breaking**: A change will be made to `AcceptTopicSuggestionPayload.topic`.
  
  **Description**: Type for `topic` will change from `Topic` to `Topic`.

  **Reason**: In preparation for an upcoming change to the way we report mutation errors, non-nullable payload fields are becoming nullable.

- **Breaking**: A change will be made to `AddCommentPayload.commentEdge`. 

  **Description**: Type for `commentEdge` will change from `CommentEdge` to `Comment`.
Schema Previews

- Deployments
- Checks
- Team discussions
- Branch Protection Rules
- Protected Branch: Required Signatures
- Hoverscards
- Protected Branch: Multiple Required Approving Reviews
- MergeInfoPreview - More detailed information about a pull request's merge state.
- Access to a Repositories Dependency Graph
- Repository Vulnerability Alerts
- Temporary Cloning Token for Private Repositories
- Resolvable Threads Preview
- Project Event Details
- Issues Preview

Schema previews let you try out new features and changes to our GraphQL schema before they become part of the official GitHub API.
Schema usage data: personalised deprecation **warnings!**
Learnings & Next Steps 🧠
Deprecating things is hard.
Built-In Deprecations: A double-edged sword
How can we make API evolution even better?
Automatic Sunsets?
Smart(er) Clients?
Auto Upgrades?
Thank you! 💚

Marc Giroux 🐱

@__xuorig__