



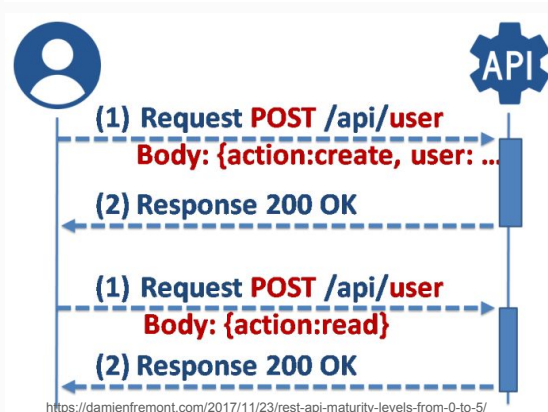
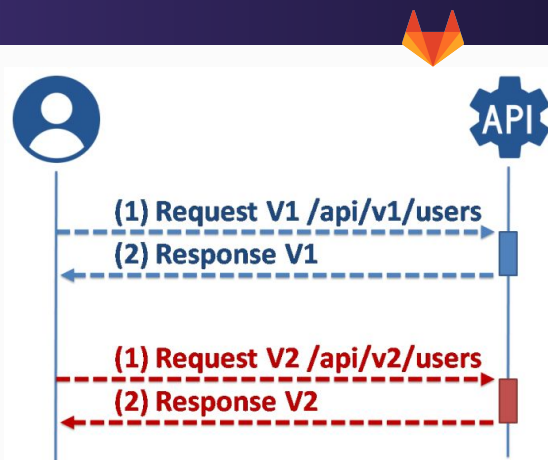
# GitLab

---

GraphQL Deep Dive - CREATE  
2019-03-20

# GraphQL - a new kind of API

- APIs are for for programmatic interaction with a web service
  - Many types of API (SOAP, REST, etc)
  - REpresentation State Transfer defined by Roy Fielding in ~2000
  - We currently have a Level Four Rest API™
  - Our frontend is increasingly a Vue-based web application for it
  - Central promise of REST is a universal client for every API
- GraphQL is a Level Zero Rest API™
  - Rejects ~20 years of Best Practice™
  - We give up on the single-client-for-everything aspiration
  - Anything (else) REST can do, GraphQL can do... better?
  - Focus on making an API that can serve dedicated clients better
- GraphQL optimizes for flexible queries, served efficiently



<https://damienfremont.com/2017/11/23/rest-api-maturity-levels-from-0-to-5/>

# “Why would you do that?”



- 19th March: [https://gitlab.com/gitlab-org/gitlab-ee/merge\\_requests/9760](https://gitlab.com/gitlab-org/gitlab-ee/merge_requests/9760)
- MR tries to improve performance of elasticsearch-backed commit search
- Existing REST API intervenes...
  - Fixed set of fields, returned for every request. Projections difficult to add
  - Compatibility guarantee, even though it's likely nobody even uses these fields
  - We can't actually tell if anyone even uses these fields
  - Hard to introduce lazy evaluation to resolve underlying N+1 issue
- If this were GraphQL...
  - Dynamic set of fields, selected only if client needs them. No need for projections
  - Living API - we could deprecate & remove these fields if we absolutely had to
  - Instrumentation can give us metrics on field use to inform deprecation decisions
  - Lazy evaluation built-in from the start

# GraphQL at Facebook and GitLab



- Started in 2012 following shift from web to native applications on mobile
- Address challenges encountered using REST: “Slow, Fragile, Tedious”
  - A great talk by Lee Byron: <https://www.youtube.com/watch?v=F-OizdRJh1U>
- React.js: open-sourced in 2012
- Relay + GraphQL: open-sourced in 2015. Specification + reference implementation
- We set a plan for adoption in 2017
  - <https://gitlab.com/gitlab-org/gitlab-ce/issues/34754>
- Issues with lack of patent grant in specification raised & resolved in 2017
  - <https://github.com/facebook/graphql/issues/351>
  - <https://facebook.github.io/graphql/June2018/>
- **Alpha** support merged into GitLab in 2018
  - [https://gitlab.com/gitlab-org/gitlab-ce/merge\\_requests/19008](https://gitlab.com/gitlab-org/gitlab-ce/merge_requests/19008)
- 12+ independent implementations by 2019.
- GraphQL foundation forming in March 2019
- Issue suggestions feature uses GraphQL!

# GraphQL basics



- Fields
  - a. Everything is a *field*
  - b. *Fields* can take arguments
- Types
  - a. Every *field* has a **type**
  - b. Some built-in, some user-defined
  - c. **QueryType** and **MutationType** are special
  - d. **Types** (can) have *fields*, forming a graph
- Schema
  - a. Specifies available *fields* and **types**
  - b. Is a *field*. I wasn't kidding about this bit
- Queries
  - a. Special mini-language, think SQL
  - b. Specifies the fields the client wants
  - c. Can be parameterized

< Schema

Query

×

🔍 Search Query...

No Description

FIELDS

echo(text: String): String!

Testing endpoint to validate the API with

metadata: Metadata

Metadata about GitLab

project(fullPath: ID!): Project

Find a project

< Schema

Mutation

×

🔍 Search Mutation...

No Description

FIELDS

mergeRequestSetWip(input: MergeRequestSetWipInput!): MergeRequestSetWipPayload

<https://gitlab.com/-/graphql-explorer>

# Example query - basic



```
1 {
2   project(fullPath:"gitlab-com/www-gitlab-com") {
3     id
4     issue(iid:1) {
5       webUrl
6     }
7   }
8
9   gitlab_ce:project(fullPath:"gitlab-org/gitlab-ce") {
10     id
11   }
12   gitlab_ee:project(fullPath:"gitlab-org/gitlab-ee") {
13     id
14   }
15 }
```

```
{
  "data": {
    "project": {
      "id": "7764",
      "issue": {
        "webUrl": "https://gitlab.com/gitlab-com/www-gitlab-com/issues/1"
      }
    },
    "gitlab_ce": {
      "id": "13083"
    },
    "gitlab_ee": {
      "id": "278964"
    }
  }
}
```

- *project* is a field of **QueryType**, type **Project**
- *gitlab\_ce:project* is the same, renamed
- *fullPath* is an argument selecting one project
  - Specify exactly the fields you want
- *issue* is a field of **Project**, type **Issue**
  - Specify exactly the fields you want
- Turned into JSON and transmitted to server
- Can load all 3 projects in one query(ish)

- Response is JSON
- Closely mirrors request semantics
- Only what I asked for is there
- Schema guarantees the server has what I want
- We can restrict the complexity of queries

# Example query - pagination & caching



- *issues* is of type **IssueConnection**
- Includes *pageInfo*, *edges*
- Each *edge* has its own *cursor* too
- Support offset, keyset, etc, pagination
- *endCursor* to page forwards
- *startCursor* to page backwards
- REST API puts this into headers instead

```
1 {
2   project(fullPath:"gitlab-org/gitlab-ce") {
3     issues(first:2, sort:created_asc) {
4       pageInfo {
5         hasNextPage
6         endCursor
7       }
8       edges {
9         node {
10          iid
11          title
12        }
13      }
14    }
15  }
16 }
```

```
{
  "data": {
    "project": {
      "issues": {
        "pageInfo": {
          "hasNextPage": true,
          "endCursor": "MjAxMyOxMiOyNSAyMDo1NCBVVEM="
        },
        "edges": [
          {
            "node": {
              "iid": "1",
              "title": "GitLab is only useable in English"
            },
            "cursor": "MjAxMyOxMiOyNSAyMDo1NCBVVEM="
          }
        ]
      }
    }
  }
}
```

- No REST, so normal caching is out
- We haven't handled this yet
- GraphQL way:
  - Everything gets a GUID
  - Clients build their own caches

```
1 {
2   project(fullPath:"gitlab-org/gitlab-ce") {
3     issues(first:2, sort:created_asc, after:"MjAxMyOxMiOyNSAyMDo1NCBVVEM=") {
4       pageInfo {
5         hasNextPage
6         endCursor
7       }
8       edges {
9         node {
10          iid
11          title
12        }
13      }
14    }
15  }
16 }
```

```
{
  "data": {
    "project": {
      "issues": {
        "pageInfo": {
          "hasNextPage": true,
          "endCursor": "MjAxNC0wMS0wOCAxNzo0NzowMSBVVEM="
        },
        "edges": [
          {
            "node": {
              "iid": "3",
              "title": "Sort tags according to SemVer"
            },
            "cursor": "MjAxNC0wMS0wOCAxNzo0NzowMSBVVEM="
          }
        ]
      }
    }
  }
}
```

## Example query - parameters and fragments



```
1 fragment projectFields on Project {
2   id
3   forksCount
4 }
5
6 query ProjectsWithGitLab($path:ID!) {
7   project(fullPath:$path) {
8     ...projectFields
9
10    issue(iid:1) {
11      iid
12      createdAt
13    }
14  }
15
16  gitlab_ce:project(fullPath:"gitlab-org/gitlab-ce") {
17    ...projectFields
18  }
19
20  gitlab_ee:project(fullPath:"gitlab-org/gitlab-ee") {
21    ...projectFields
22  }
23 }
```

### QUERY VARIABLES

```
1 {
2   "path":"gitlab-com/www-gitlab-com"
3 }
```

```
{
  "data": {
    "project": {
      "issue": {
        "iid": "1",
        "createdAt": "2013-09-12T10:43:54Z"
      },
      "id": "7764",
      "forksCount": 813
    },
    "gitlab_ce": {
      "id": "13083",
      "forksCount": 5129
    },
    "gitlab_ee": {
      "id": "278964",
      "forksCount": 693
    }
  }
}
```



# Example query - directives



- Make some *fields* conditional.
  - a. include if
  - b. include unless
  - c. skip if
  - d. skip unless
- They're in the schema too
- Fewer, more-general, **static** queries

```
1 query ReadProject($path:ID!, $withIssues:Boolean!) {  
2   project(fullPath:$path) {  
3     id  
4  
5     issues(last:10) @include(if: $withIssues) {  
6       edges { node { iid , title } }  
7     }  
8   }  
9 }
```

## QUERY VARIABLES

```
1 {  
2   "path": "gitlab-org/gitlab-ce",  
3   "withIssues": false  
4 }
```

```
{  
  "data": {  
    "project": {  
      "id": "13083"  
    }  
  }  
}
```

```
1 query ReadProject($path:ID!, $withIssues:Boolean!) {  
2   project(fullPath:$path) {  
3     id  
4  
5     issues(last:10) @include(if: $withIssues) {  
6       edges { node { iid , title } }  
7     }  
8   }  
9 }
```

## QUERY VARIABLES

```
1 {  
2   "path": "gitlab-org/gitlab-ce",  
3   "withIssues": true  
4 }
```

```
{  
  "data": {  
    "project": {  
      "id": "13083",  
      "issues": {  
        "edges": [  
          {  
            "node": {  
              "iid": "10",  
              "title": "Linked issues"  
            }  
          },  
          {  
            "node": {  
              "iid": "11",  
              "title": "Linked issues"  
            }  
          },  
          {  
            "node": {  
              "iid": "12",  
              "title": "Linked issues"  
            }  
          },  
          {  
            "node": {  
              "iid": "13",  
              "title": "Linked issues"  
            }  
          },  
          {  
            "node": {  
              "iid": "14",  
              "title": "Linked issues"  
            }  
          },  
          {  
            "node": {  
              "iid": "15",  
              "title": "Linked issues"  
            }  
          },  
          {  
            "node": {  
              "iid": "16",  
              "title": "Linked issues"  
            }  
          },  
          {  
            "node": {  
              "iid": "17",  
              "title": "Linked issues"  
            }  
          },  
          {  
            "node": {  
              "iid": "18",  
              "title": "Linked issues"  
            }  
          },  
          {  
            "node": {  
              "iid": "19",  
              "title": "Linked issues"  
            }  
          }  
        ]  
      }  
    }  
  }  
}
```

# Example mutation



- We only have this one right now!
- Very RPC-like, which is by design
- Can run multiple mutations too
- We've got a lot to learn here

```
1
2 mutation {
3   mergeRequestSetWip(input:{
4     projectPath:"nick.thomas/test-project",
5     iid:"1",
6     wip:true
7   }) {
8     errors
9     mergeRequest {
10      workInProgress
11    }
12  }
13 }
14
```

```
{
  "data": {
    "mergeRequestSetWip": {
      "errors": [],
      "mergeRequest": {
        "workInProgress": true
      }
    }
  }
}
```

```
1
2 mutation {
3   foo:mergeRequestSetWip(input:{
4     projectPath:"nick.thomas/test-project",
5     iid:"1",
6     wip:true
7   }) {
8     errors
9     mergeRequest {
10      workInProgress
11    }
12  }
13
14   bar:mergeRequestSetWip(input:{
15     projectPath:"nick.thomas/test-project",
16     iid:"2",
17     wip:true
18   }) {
19     errors
20     mergeRequest {
21      workInProgress
22    }
23  }
24 }
25
```

```
{
  "data": {
    "foo": {
      "errors": [],
      "mergeRequest": {
        "workInProgress": true
      }
    },
    "bar": null
  },
  "errors": [
    {
      "message": "The resource that you are attempting to access does not exist",
      "locations": [
        {
          "line": 14,
          "column": 3
        }
      ],
      "path": [
        "bar"
      ]
    }
  ]
}
```



- Websockety
- I can't really dive deeply into this, I've never touched it
- Support for Relay and ActionCable out of the box, though \o/
- Historically, we've been unable to support long-lived connections to gitlab backend
- Move to Puma might unlock some of these capabilities

# Authorization & Authentication



```
1 {  
2   echo(text:"Hello")  
3  
4   metadata {  
5     version  
6     revision  
7   }  
8  
9   project(fullPath:"gitlab-org/gitlab-ce") {  
10    id  
11    userPermissions {  
12      readProject  
13      adminProject  
14    }  
15  }  
16 }
```

```
{  
  "data": {  
    "echo": "nil says: Hello",  
    "metadata": null,  
    "project": {  
      "id": "13083",  
      "userPermissions": {  
        "readProject": true,  
        "adminProject": false  
      }  
    }  
  }  
}
```

- Authentication
  - Similar to REST API
  - Supports cookies and token-based authentication. CSRF protection for cookie auth
- Authorization
  - Individual fields are authorizable
  - Efficiently authorizing arbitrarily complex queries can be a challenge
  - Requests can partially succeed, leaving unauthorized fields blank, or completely fail
  - You can query your own permissions for a field

# Adding a new GraphQL endpoint



- Important files:
  - `app/assets/javascripts/lib/graphql.js`
  - `app/assets/javascripts/**/*.graphql`
  - `app/controllers/graphql_controller.rb`
  - `app/graphql/gitlab_schema.rb`
  - `app/graphql/functions/*`
  - `app/graphql/mutations/*`
  - `app/graphql/resolvers/*`
  - `app/graphql/types/*`
  - `lib/gitlab/graphql/*`
  - `spec/graphql/*`
  - `spec/requests/api/graphql/*`
- EE-only support is coming in 11.10
- You're adding *fields* (of course)
- New top-level query fields in **QueryType**
- Otherwise added to another type
- Resolvers gather data without N+1 issues
- Functions are simple resolvers for a few fields
- Specs are composable, just like the fields
  - Request specs can be simpler
- Frontend defines a static query in `.graphql` file
- Applies it to the graphql client to do the thing
- Reimplement REST API in terms of GraphQL
- How does that search API look?
  - GitHub's (sneaky peek)
- Ouch. OK, let's look at file templates instead
- <https://graphql.org/>
- <https://graphql-ruby.org/>