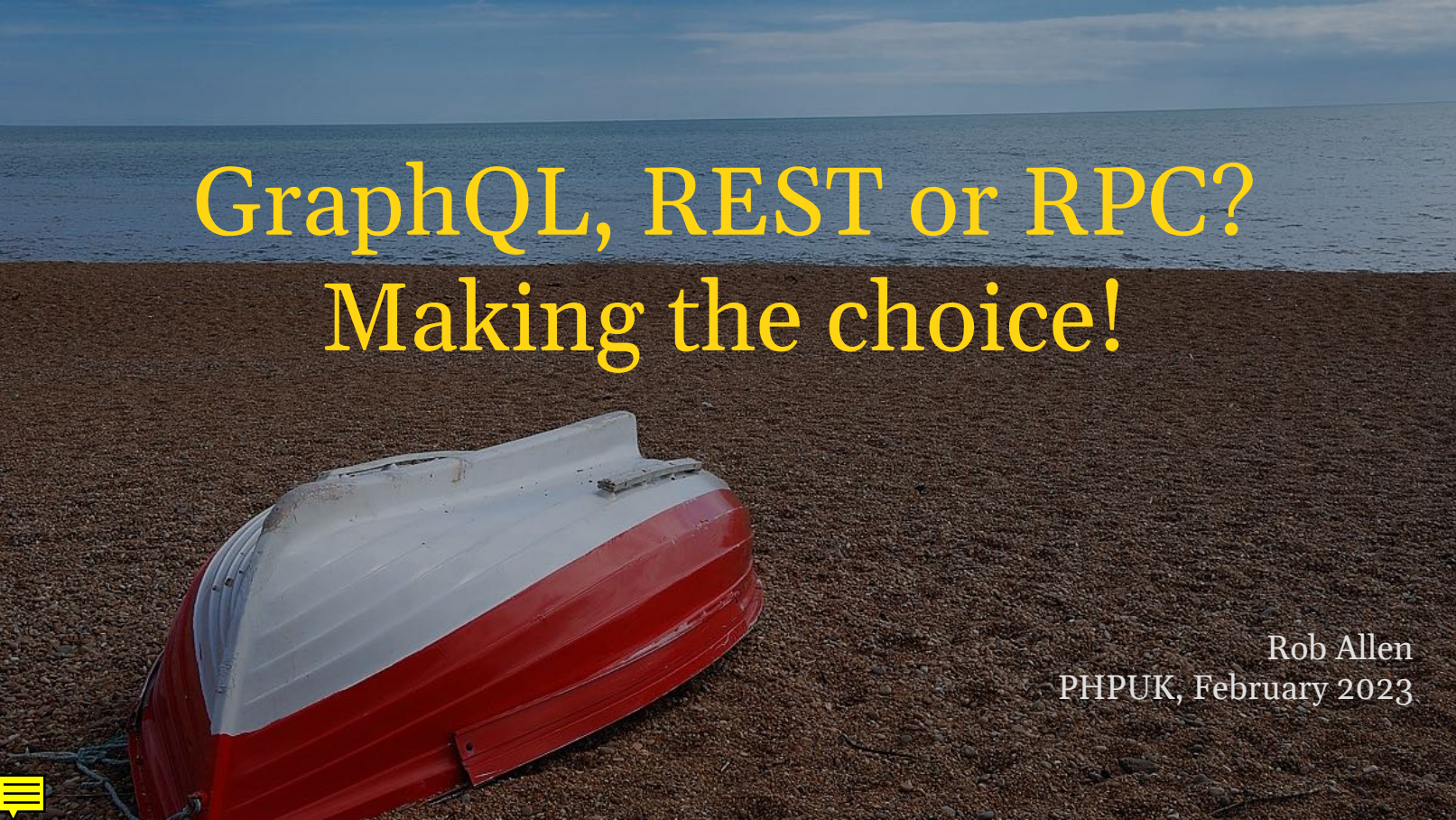
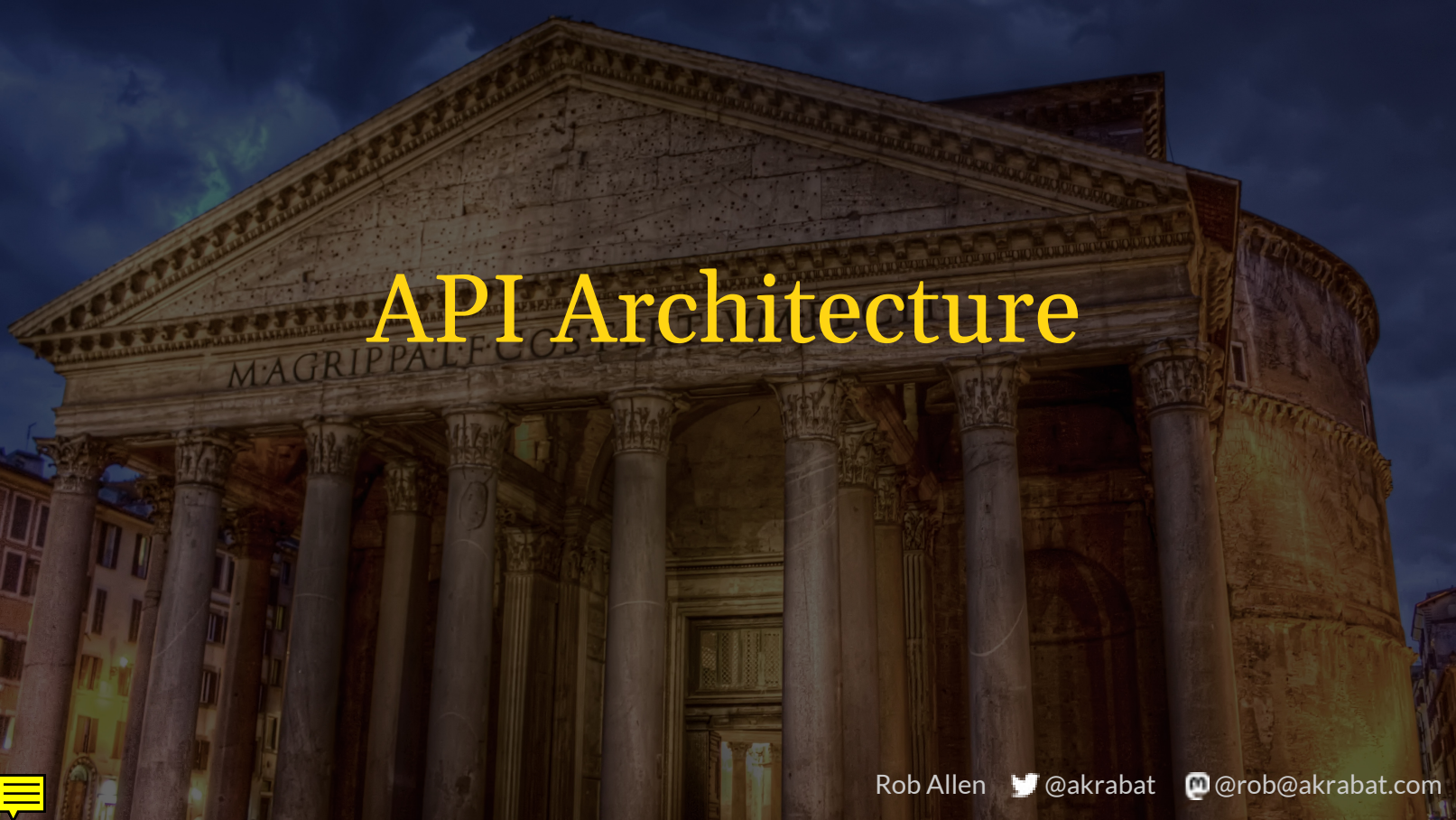
GraphQL, REST or RPC?
Making the choice!



Rob Allen
PHPUK, February 2023


Fit for Purpose
Rob Allen @akrabat @rob@akrabat.com



API Architecture

Rob Allen @akrabat @rob@akrabat.com

APIs can be realised in any style
but, which makes the most sense?

RPC APIs

Rob Allen @akrabat @rob@akrabat.com

RPC APIs

- Call a function on a remote server

RPC APIs

- Call a function on a remote server
- Common implementations: JSON-RPC, SOAP, gRPC

RPC APIs

- Call a function on a remote server
- Common implementations: JSON-RPC, SOAP, gRPC
- Tends to require a schema (WSDL, ProtoBuf Defintion)

Ethereum JSON-RPC

Request:

```
POST / HTTP/1.1
Host: localhost:8545

{
  "jsonrpc":"2.0",
  "id":1,
  "method":"net_peerCount",
  "params":[]
}
```

Ethereum JSON-RPC

Response:

```
{
  "id":1,
  "jsonrpc": "2.0",
  "result": "0x2"
}
```

gRPC

Interact via PHP library:

```
$client = new RouteGuideClient('localhost:50051');

$p = new Routeguide\Point();
$p->setLatitude(409146138);
$p->setLongitude(-746188906);
list($feature, $status) = $client->GetFeature($p)->wait();
```

RESTful APIs

Rob Allen @akrabat @rob@akrabat.com

RESTful APIs

- Operate on a representation of the state of a resource though HTTP verbs

RESTful APIs

- Operate on a representation of the state of a resource though HTTP verbs
- HTTP native

RESTful APIs

- Operate on a representation of the state of a resource though HTTP verbs
- HTTP native
- Uniform interface

RESTful APIs

- Operate on a representation of the state of a resource though HTTP verbs
- HTTP native
- Uniform interface
- Hypermedia controls

RESTful APIs

```
PUT /users/ba60c99fd3
Content-Type: application/json
Accept: application/json

{
    "name": "Rob Allen"
    "email": "rob@akrabat.com"
}
```

RESTful APIs

```
PUT /users/ba60c99fd3
Content-Type: application/json
Accept: application/json

{
    "name": "Rob Allen"
    "email": "rob@akrabat.com"
}
```

RESTful APIs

```
PUT /users/ba60c99fd3
Content-Type: application/json
Accept: application/json

{
    "name": "Rob Allen"
    "email": "rob@akrabat.com"
}
```

RESTful APIs

```
PUT /users/ba60c99fd3
Content-Type: application/json
Accept: application/json

{
    "name": "Rob Allen"
    "email": "rob@akrabat.com"
}
```

RESTful APIs: Response

```
HTTP/1.1 201 Created
Content-Type: application/hal+json
ETag: dfb9f2ab35fe4d17bde2fb2b1cee88c1

{
    "name": "Rob Allen"
    "email": "rob@akrabat.com",
    "_links": {
      "self": "https://api.example.com/user/ba60c99fd3"
    }
}
```

RESTful APIs

```
HTTP/1.1 201 Created
Content-Type: application/hal+json
ETag: dfb9f2ab35fe4d17bde2fb2b1cee88c1

{
    "name": "Rob Allen"
    "email": "rob@akrabat.com",
    "_links": {
      "self": "https://api.example.com/user/ba60c99fd3"
    }
}
```

RESTful APIs

```
HTTP/1.1 201 Created
Content-Type: application/hal+json
ETag: dfb9f2ab35fe4d17bde2fb2b1cee88c1

{
    "name": "Rob Allen"
    "email": "rob@akrabat.com",
    "_links": {
      "self": "https://api.example.com/user/ba60c99fd3"
    }
}
```

RESTful APIs

```
HTTP/1.1 201 Created
Content-Type: application/hal+json
ETag: dfb9f2ab35fe4d17bde2fb2b1cee88c1

{
    "name": "Rob Allen"
    "email": "rob@akrabat.com",
    "_links": {
      "self": "https://api.example.com/user/ba60c99fd3"
    }
}
```

GraphQL APIs

Rob Allen @akrabat @rob@akrabat.com

GraphQL APIs

- Retrieve only the data you need on consumer side

GraphQL APIs

- Retrieve only the data you need on consumer side
- Reduce the number of calls to retrieve data with embedded resources

GraphQL APIs

- Retrieve only the data you need on consumer side
- Reduce the number of calls to retrieve data with embedded resources
- Self-describing schema

Queries

```
query {
  author(name: "Anne McCaffrey") {
    id, name
    books(first: 5) {
      totalCount
      edges {
        node {
          id, title
        }
      }
    }
  }
}
```

Queries

```
query {
  author(name: "Anne McCaffrey") {
    id, name
    books(first: 5) {
      totalCount
      edges {
        node {
          id, title
        }
      }
    }
  }
}
```

Queries

```
query {
  author(name: "Anne McCaffrey") {
    id, name
    books(first: 5) {
      totalCount
      edges {
        node {
          id, title
        }
      }
    }
  }
}
```

Queries

```
query {
  author(name: "Anne McCaffrey") {
    id, name
    books(first: 5) {
      totalCount
      edges {
        node {
          id, title
        }
      }
    }
  }
}
```

Queries

```
query {
  author(name: "Anne McCaffrey") {
    id, name
    books(first: 5) {
      totalCount
      edges {
        node {
          id, title
        }
      }
    }
  }
}
```

Queries

```
query {
  author(name: "Anne McCaffrey") {
    id, name
    books(first: 5) {
      totalCount
      edges {
        node {
          id, title
        }
      }
    }
  }
}
```

Queries

```
query {
  author(name: "Anne McCaffrey") {
    id, name
    books(first: 5) {
      totalCount
      edges {
        node {
          id, title
        }
      }
    }
  }
}
```

Queries: Result


```
"data": {
  "author": {
    "id": "MXxBdXRob3J8ZjA",
    "name": "Anne McCaffrey",
    "books": {
      "totalCount": 6,
      "edges": [
        {
          "node": {
            "id": "MXxCb29rfGYwNzU",
            "title": "Dragonflight"
          }
        },
```


Queries: Result


```
"data": {
  "author": {
    "id": "MXxBdXRob3J8ZjA",
    "name": "Anne McCaffrey",
    "books": {
      "totalCount": 6,
      "edges": [
        {
          "node": {
            "id": "MXxCb29rfGYwNzU",
            "title": "Dragonflight"
          }
        },
```


Queries: Result


```
"data": {
  "author": {
    "id": "MXxBdXRob3J8ZjA",
    "name": "Anne McCaffrey",
    "books": {
      "totalCount": 6,
      "edges": [
        {
          "node": {
            "id": "MXxCb29rfGYwNzU",
            "title": "Dragonflight"
          }
        },
```


Queries: Result


```
"data": {
  "author": {
    "id": "MXxBdXRob3J8ZjA",
    "name": "Anne McCaffrey",
    "books": {
      "totalCount": 6,
      "edges": [
        {
          "node": {
            "id": "MXxCb29rfGYwNzU",
            "title": "Dragonflight"
          }
        },
```


Mutations

```
mutation {
  createAuthor(
    name: "Mary Shelley", dateOfBirth: "1797-08-30"
  ) {
    returning {
      id, name
    }
  }
}
```

Mutations

```
mutation {
  createAuthor(
    name: "Mary Shelley", dateOfBirth: "1797-08-30"
  ) {
    returning {
      id, name
    }
  }
}
```

Mutations

```
mutation {
  createAuthor(
    name: "Mary Shelley", dateOfBirth: "1797-08-30"
  ) {
    returning {
      id, name
    }
  }
}
```

Mutations

```
mutation {
  createAuthor(
    name: "Mary Shelley", dateOfBirth: "1797-08-30"
  ) {
    returning {
      id, name
    }
  }
}
```

Mutations: Response

Response:

```
"data": {
  "createAuthor": {
    "returning": [
      {
        "id": "e3388cbea4e840a",
        "name": "Mary Shelly",
      }
    ]
  }
}
```


Which to pick?

Lamborghini or Ferrari?

Rob Allen @akrabat @rob@akrabat.com

Lamborghini or Truck?

Rob Allen @akrabat @rob@akrabat.com

Considerations

- What is it to be used for?
- Response customisation requirements
- HTTP interoperability requirements

API Uses

- Do you control both server and client?
- How many users are expected?
- What is the skill level of your integrators?

Response customisation

- GraphQL is a query-first language
- REST tends towards less customisation
- With RPC you get what you're given!

(None will fix your database layer's ability to efficiently retreive the data requested!)

Performance

- REST and RPC puts server performance first
- GraphQL puts client performance first

Caching

- GraphQL and RPC can only cache at application layer
- REST can additionally cache at HTTP layer

Data Transfer

GraphQL:

```
query {
  avatar(userId: "1234")
}

{
  "data": {
    "avatar": "(base64 data)"
    "format": "image/jpeg"
  }
}}
```

RPC:

```
POST /api
{
  "method": "getAvatar",
  "userId": "1234"
}

{
  "result": "(base64 data)"
}
```

Data Transfer

REST:

```
GET /user/1234/avatar
Accept: image/jpeg

HTTP/1.1 200 OK
{jpg image data}
```

REST:

```
GET /user/1234/avatar
Accept: application/json

HTTP/1.1 200 OK
{"data": "(base64 data)"}
```

Versioning

- RPC, GraphQL and REST can all version via evolution as easily as each other

Versioning

- RPC, GraphQL and REST can all version via evolution as easily as each other
- GraphQL is very good for deprecation of specific fields

Design considerations

It's *always* hard!

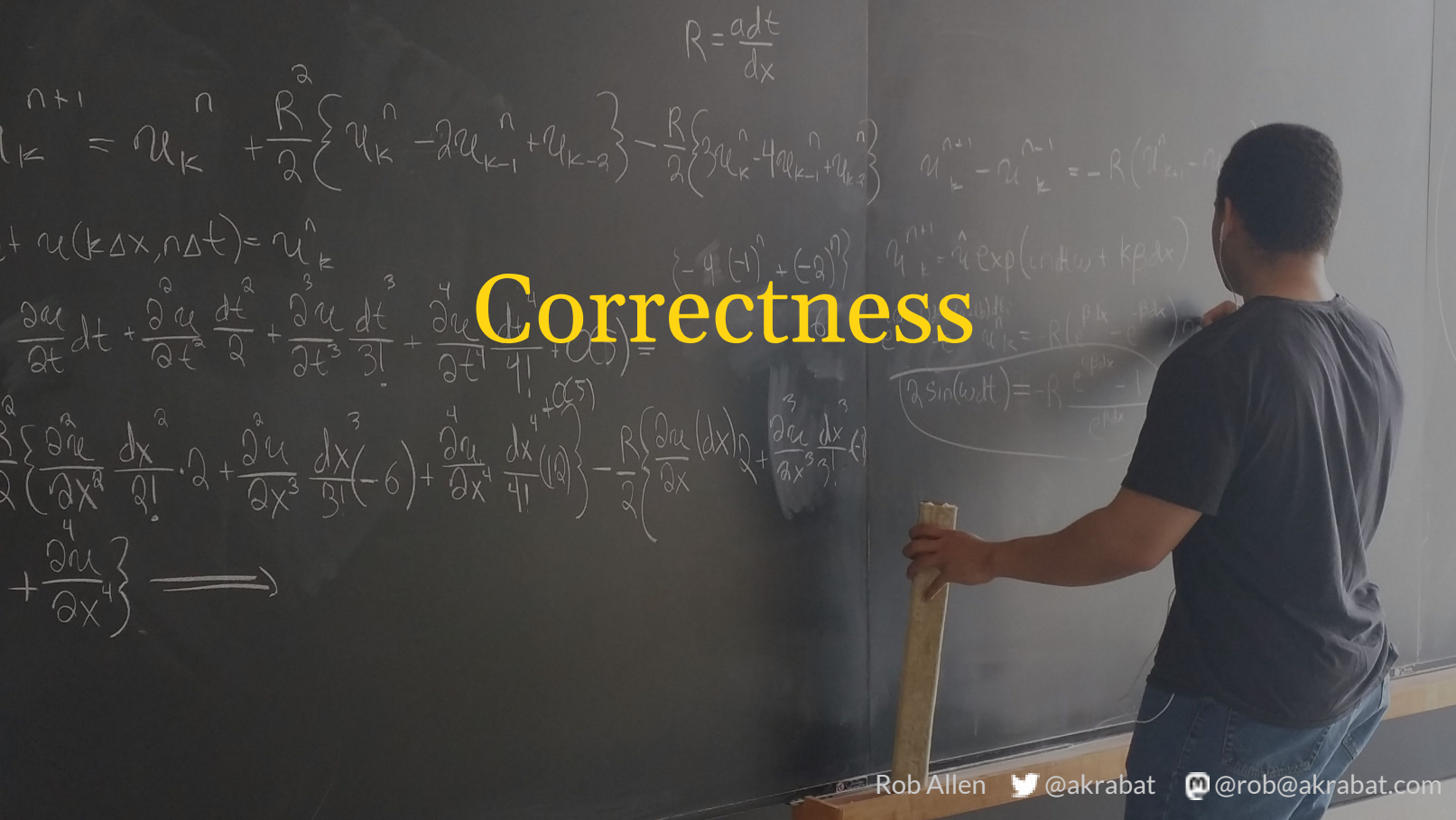
Design considerations

It's *always* hard!

It's your choice

Developer Experience

Rob Allen @akrabat @rob@akrabat.com



Correctness

Correctness

RPC: Functions!

Correctness

RPC: Functions!
REST: HTTP matters!

Correctness

RPC: Functions!
REST: HTTP matters!
GraphQL: Think in terms of relationships!

Correctness

RPC: Functions!
REST: HTTP matters!
GraphQL: Think in terms of relationships!

Errors

Rob Allen @akrabat @rob@akrabat.com

REST Errors

```
HTTP/1.1 503 Service Unavailable
Content-Type: application/problem+json
Content-Language: en

{
    "status": 503,
    "type": "https://example.com/service-unavailable",
    "title": "Could not authorise user.”,
    "detail": "Auth service is down for maintenance.”,
    "instance": “https://example.com/maintenance/2023-02-15",
    "error_code": "AUTHSERVICE_UNAVAILABLE"
}
```

GraphQL Errors

- Always returns 200, unless infrastructure failure
- Common to see cryptic messages from GraphQL service
- Top level 'errors' key for exceptional scenarios
- Domain errors should be within the schema
- Consider using an error type per node and a union:

```
union CreateUserResult = UserCreated | UserCreationErrors
```


Documentation
Rob Allen
@akrabat
@rob@akrabat.com

API Reference

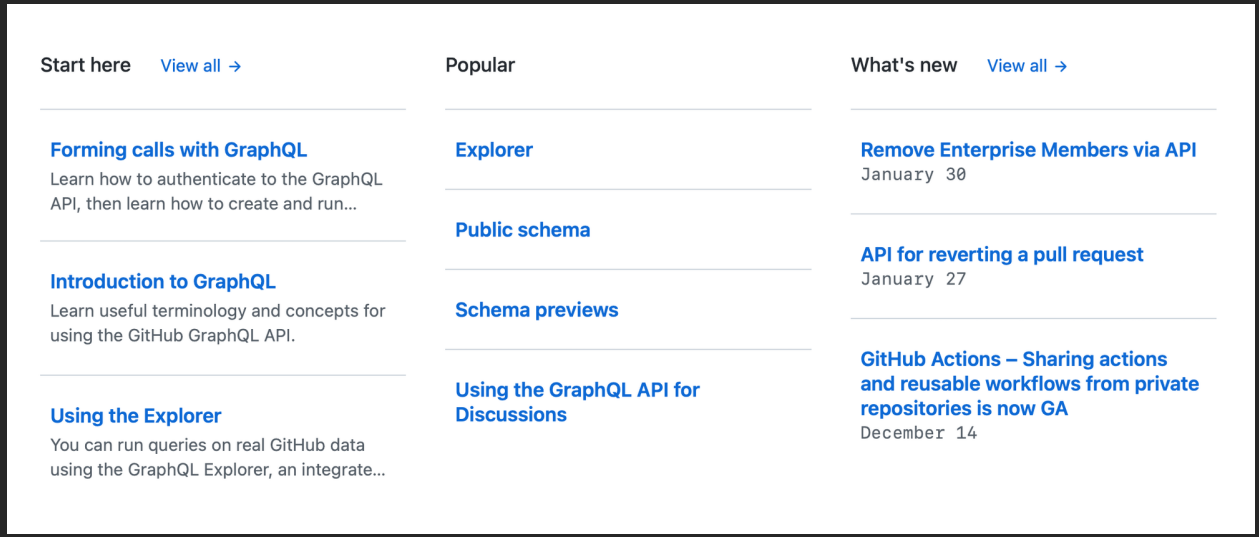
- GraphQL: Built-in introspection
- REST: OpenAPI Specification

Both allow generation of a API reference website.

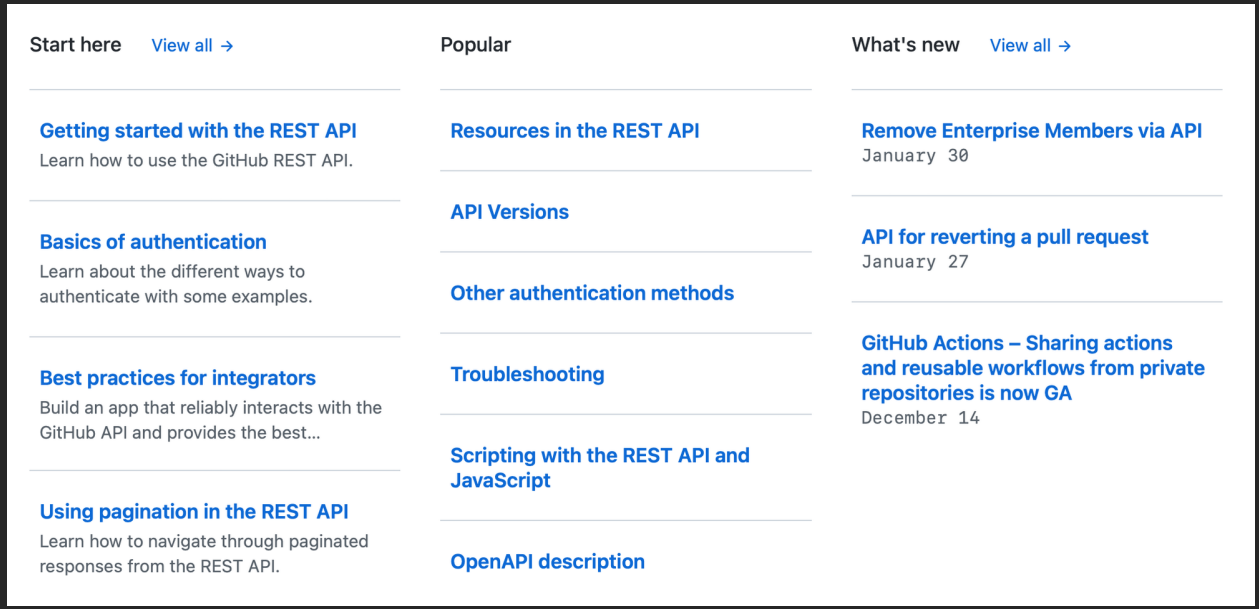
Tutorials

The API Reference is the *what*, the tutorials are the *how* and *why*

GitHub GraphQL



GitHub REST



Start here View all →

Getting started with the REST API
Learn how to use the GitHub REST API.

Basics of authentication
Learn about the different ways to authenticate with some examples.

Best practices for integrators
Build an app that reliably interacts with the GitHub API and provides the best...

Using pagination in the REST API
Learn how to navigate through paginated responses from the REST API.

Popular

Resources in the REST API

API Versions

Other authentication methods

Troubleshooting

Scripting with the REST API and JavaScript

OpenAPI description

What's new View all →

Remove Enterprise Members via API
January 30

API for reverting a pull request
January 27

GitHub Actions – Sharing actions and reusable workflows from private repositories is now GA
December 14

To sum up

If you suck at providing a REST API,
you will suck at providing a GraphQL API

Arnaud Lauret, API Handyman


Thank you!
Rob Allen @akrabat @rob@akrabat.com

Photo credits

- Architecture: https://www.flickr.com/photos/shawnstilwell/4335732627
- Choose Pill: https://www.flickr.com/photos/eclib/4905907267
- Lamborghini & Ferrari: https://akrab.at/3w0yFmg
- Lamborghini & Truck: https://akrab.at/3F4kAZk
- '50s Computer: https://www.flickr.com/photos/9479603@N02/49755349401
- Blackboard: https://www.flickr.com/photos/bryanalexander/17182506391
- Crash Test: https://www.flickr.com/photos/astrablog/4133302216