

*n*Lighten™

nLighten 4 API Guide



Overview

The nLighten Application Programming Interface (API) is an interface to share nLighten data with other applications. It is implemented with REST and GraphQL and currently returns JavaScript Object Notation (JSON) objects that represent the requested data from nLighten. This API is protected using API Keys and token security model and IP address whitelisting. This document gives a developer who wants to access the API, the information needed to complete this task.

Assumptions

This document assumes that the user has already been given an nLighten account to access the desired instance, has activated this account, and has a whitelisted IP address to access the portal. If an account has not been created for nLighten access, it should be completed prior to using this document.

Access Token Authentication

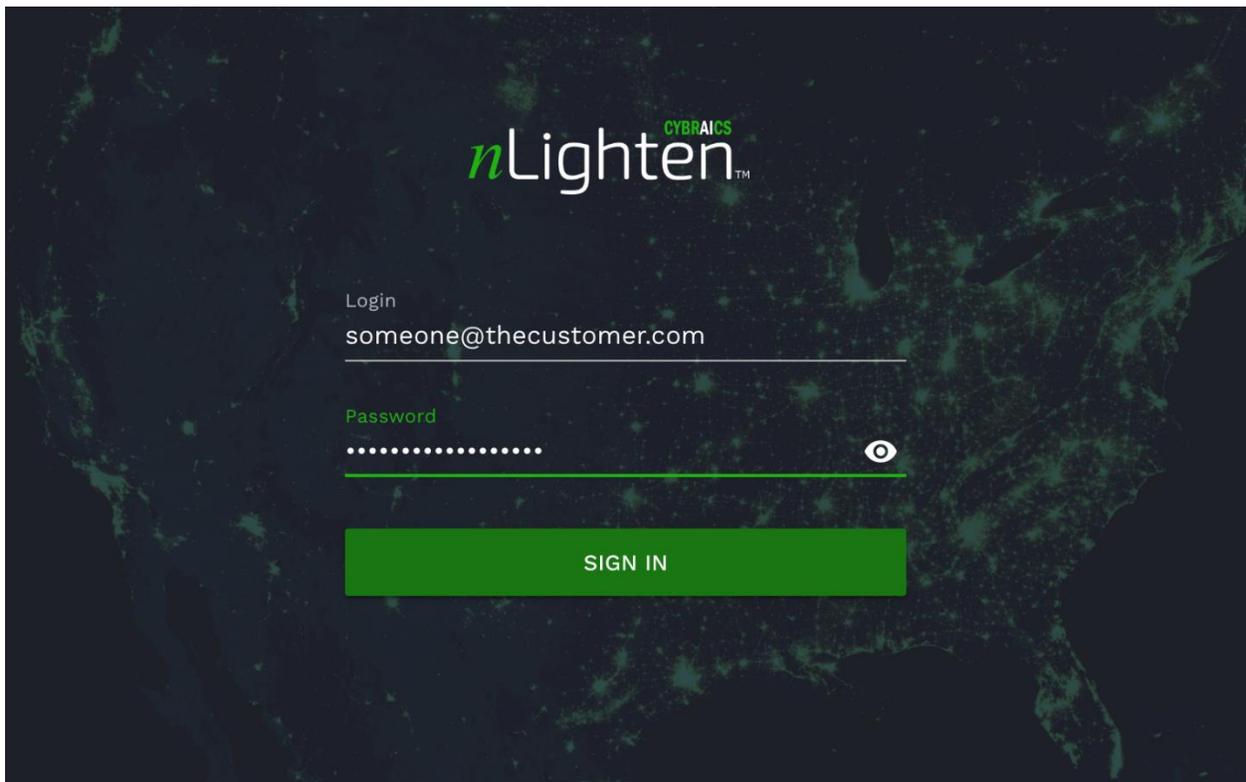
HTTP requests made to nLighten must be authenticated or the response will be [401 Unauthorized](#). If the request is authenticated but the requestor doesn't have the required permissions, the response will be [403 Forbidden](#).

To authenticate, the requestor must supply an access token in an [Authorization HTTP header](#). Below is an example. The long string of characters is the access token.

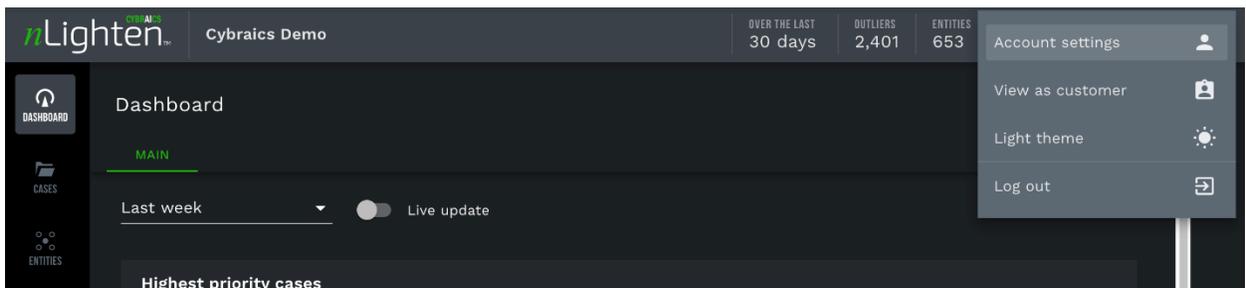
```
{ "Authorization": "Bearer 123456789012345678" }
```

You can create an access token for an instance of nLighten by following steps below.

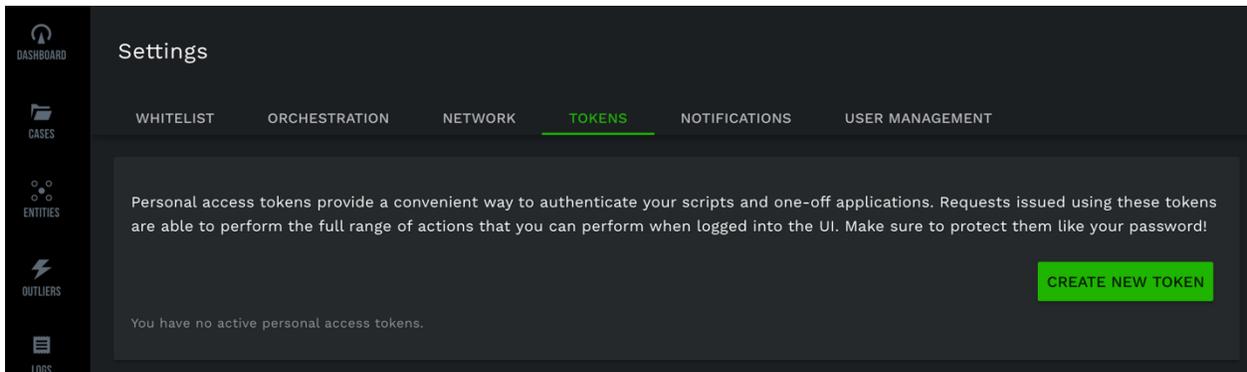
1. Sign into the nLighten instance (<https://<pea>.nl.cybraics.com>).



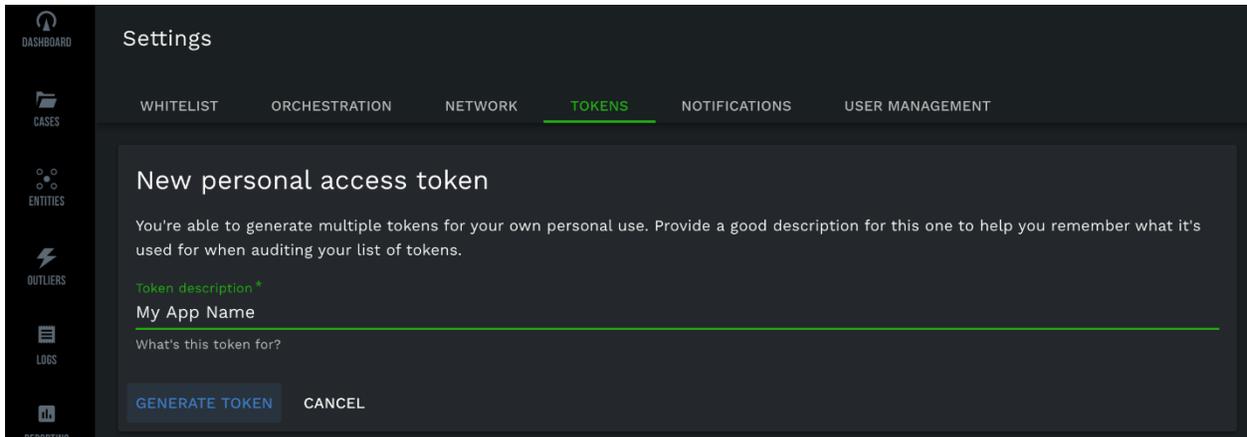
2. At the top right of nLighten, click on your name to get a dropdown menu, then click on "Account Settings".



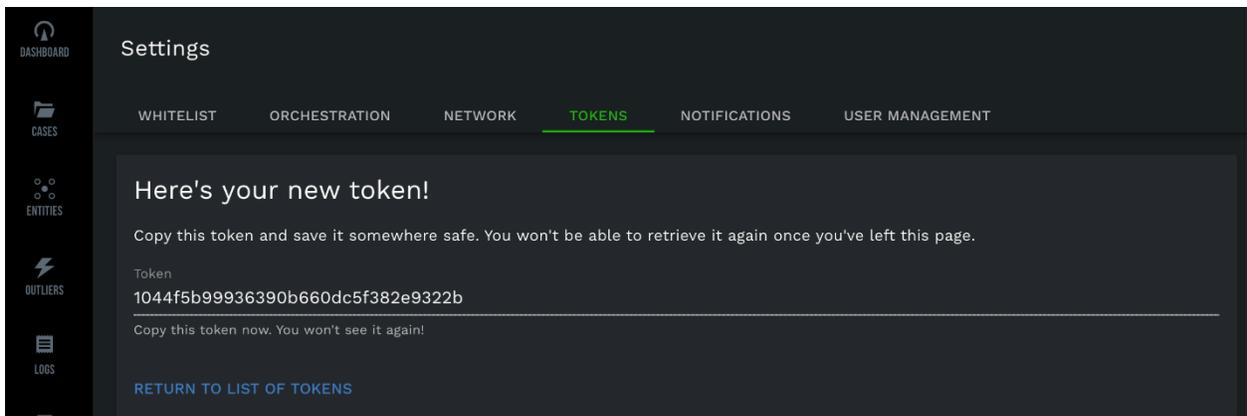
3. On the "Settings" page click on the "TOKENS" tab.



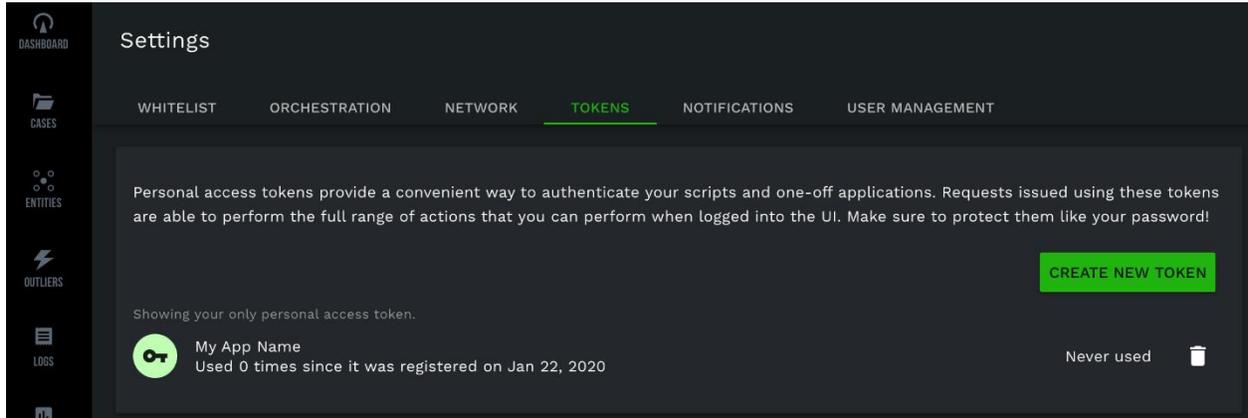
4. Under the "TOKENS" tab click on the "CREATE NEW TOKEN" button.
Add a token description. A useful description is the name of the application that will use the token.



5. Click on the "GENERATE TOKEN" button and the token will be displayed. The long string of characters is the access token.
Be certain to copy and save the access token somewhere safe! Some place that you will be able to remember/find when you need to use the token. This is the only time nLighten will display the access token!



6. If you click on "RETURN TO LIST OF TOKENS" you will see a list of all your access tokens and when they were last used. You can click on the trashcan icon at the right of each token to delete it. If you lost a token you previously created, you can always delete that token and create a new one.



REST API

Media Type

The REST API uses an `application/json` media type.

Request HTTP header:

```
{ "Accept": "application/json" }
```

Response HTTP header:

```
{ "Content-Type": "application/json; charset=utf-8" }
```

Response Body Structure

The REST API responds with the [JSON:API v1.0 document structure](#). Note however, as stated above, the REST API does *not* adhere to the JSON:API requirement of media type `application/vnd.api+json`.

For example, a request for `/api/cases/477` will return an JSON object similar to the following.

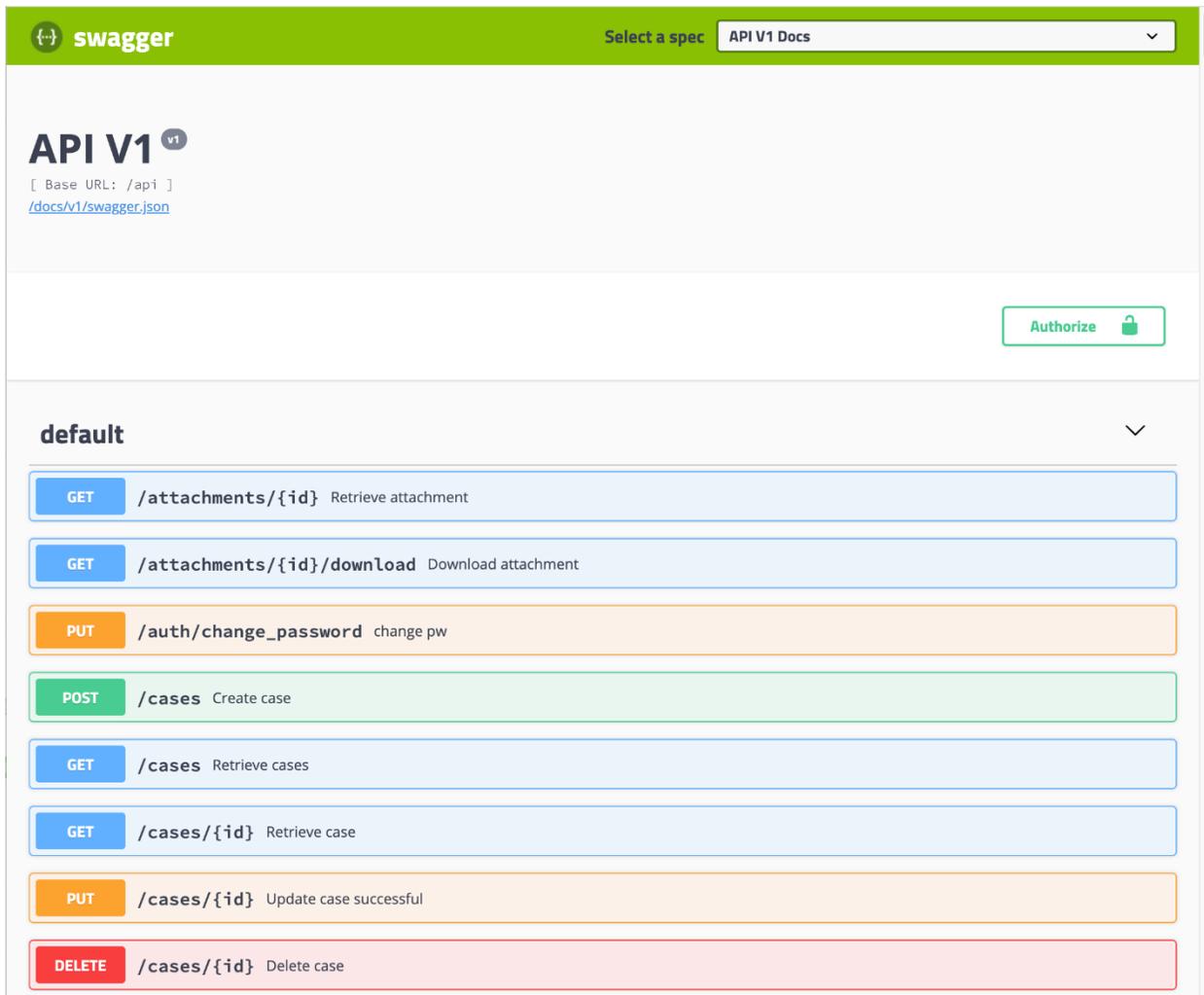
```
{
  "data": {
    "id": "477",
    "type": "cases",
    "attributes": {
      "id": 477,
      "title": "Suspicious VPN Connection Detected",
      "author_id": 4,
      // ... redacted remaining attributes
    },
    "relationships": {
      "author": {
        "data": {
          "id": "4",
          "type": "users"
        }
      },
      // ... redacted remaining relationships
    },
    "links": {
      "comments": "/api/cases/477/comments"
    }
  }
}
```

Interactive Documentation

Interactive documentation for the REST API is generated by [Swagger](#) and is made available at the following URL.

```
https://<pea>.nl.cybraics.com/docs/swagger-ui/index.html
```

When loaded, the page should look something like the image below.



You can click on one of the listed actions to see the accepted parameters. For example, POST /cases looks like the image below. Accepted parameters are title, summary, risk, and threat_type_id as shown by the "Example Value".

POST /cases Create case Try it out

Parameters

Name	Description
case_create_attrs <i>(body)</i>	<p>Example Value Model</p> <pre>{ "title": "string", "summary": "string", "risk": 0, "threat_type_id": 0 }</pre> <p>Parameter content type: application/json</p>

Responses Response content type: application/json

Click on the "Model" link and you will get a description of each parameter.

POST /cases Create case Try it out

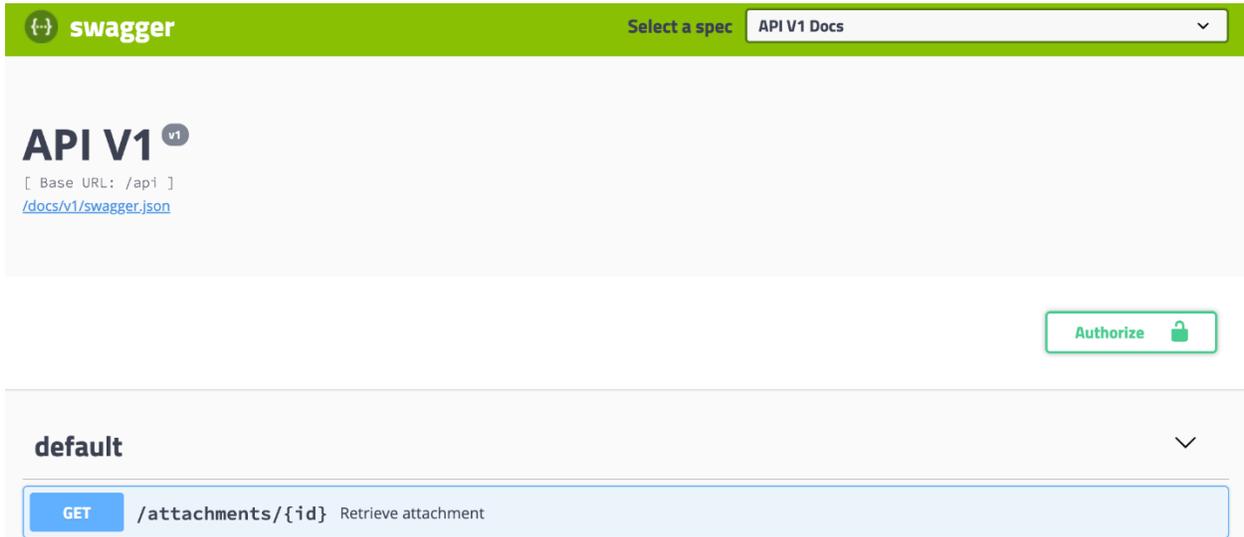
Parameters

Name	Description
case_create_attrs <i>(body)</i>	<p>Example Value Model</p> <pre>▼ { title string The title that the case will be created with. summary string A short summary of this case, suitable for displaying in lists and headers. risk integer The severity of the problem that this case documents, 0 being least severe and 4 being most severe. Enum: > Array [5] threat_type_id integer The id of the threat type that best represents this case. Must be a valid threat type id. }</pre>

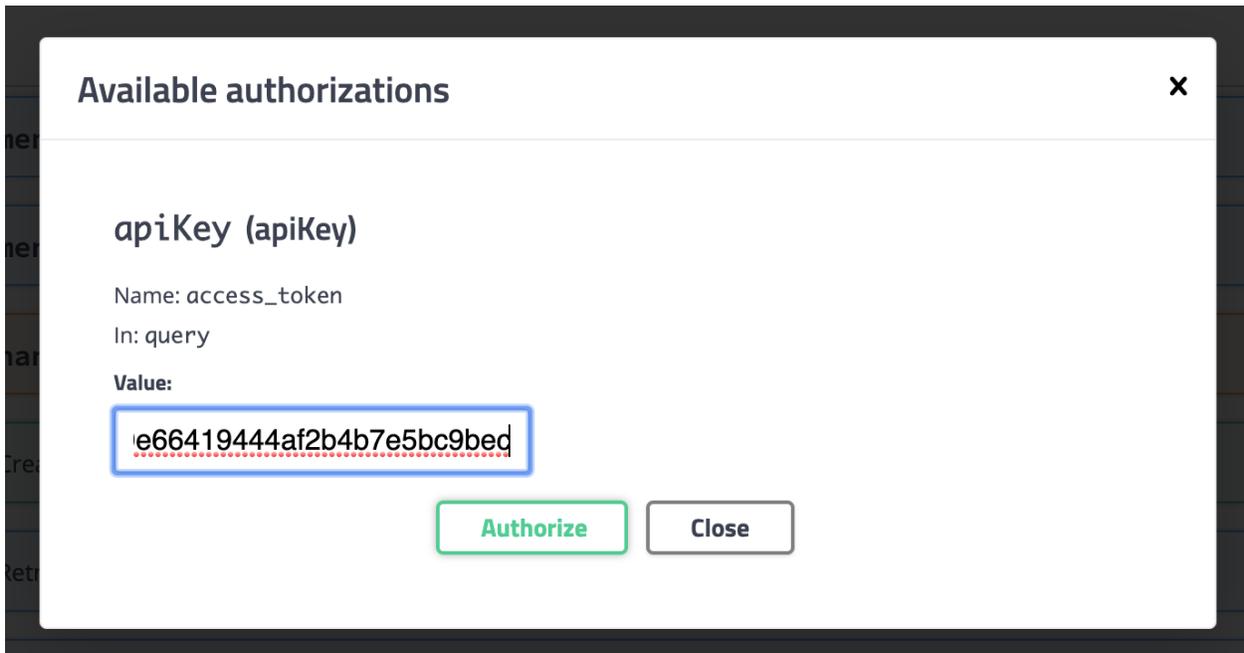
Responses Response content type: application/json

You can experiment with the REST API directly from this page using the following steps.

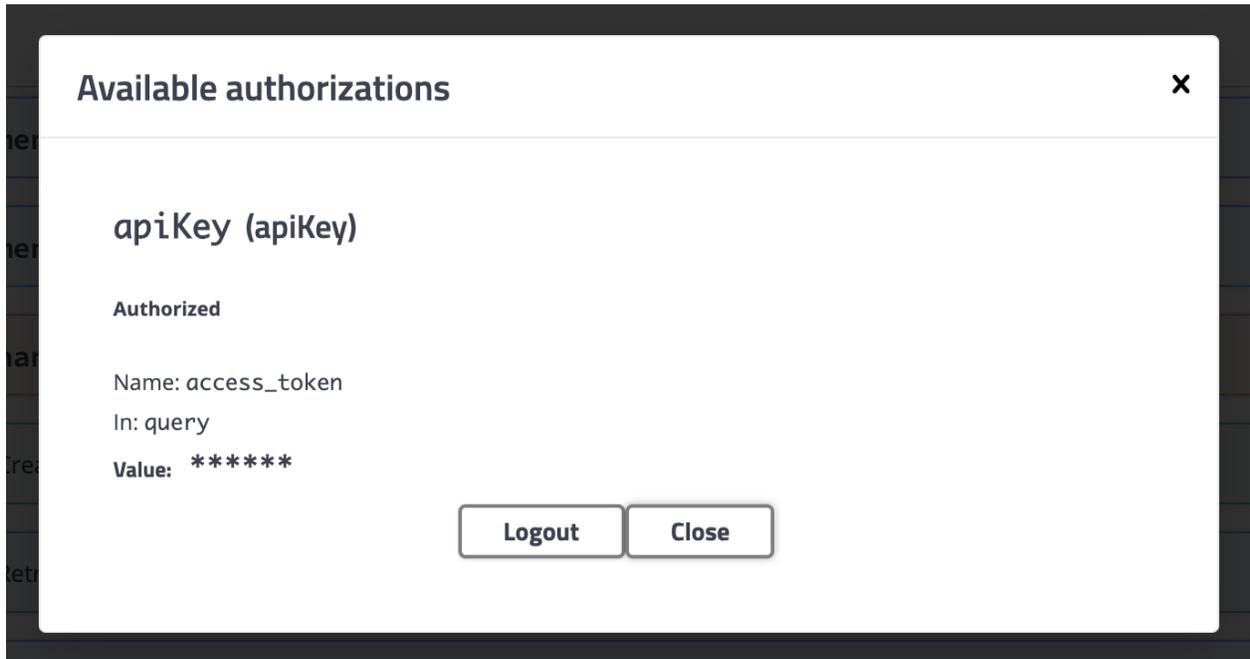
1. Click on the "Authorize" button at the top right.



2. Enter the access token you generated earlier and click on the "Authorize" button.



3. Click on the "Close" button to get back to the REST API.
Whenever you are done experimenting with the API, you can again click the "Authorize" button to display this window, then click on the "Logout" button to remove your token.



- Click on one of the listed actions, for example, GET /cases.

GET /attachments/{id}/download Download attachment

PUT /auth/change_password change pw

POST /cases Create case

GET /cases Retrieve cases

Parameters Try it out

No parameters

Responses Response content type application/json

Code	Description
200	<p><i>only includes published cases</i></p> <p>Example Value Model</p> <p>application/json</p> <pre>{ "data": [{ "id": "1359", "type": "cases", "attributes": { "id": 1359, "title": "case title", "author_id": 2201, "summary": "", "created_at": "2017-01-02T03:04:05.000Z", "updated_at": "2017-01-02T03:04:05.000Z", "status": "open", "deleted_at": null, "risk": 0, "opened_at": "2017-01-02T03:04:05.000Z", "threat_type_id": 2, "external_ticket_url": null, "external_ticket_title": null, "closed_at": null, "indexed_at": "2017-01-02T03:04:05.000Z", "analytics_used": [], "data_sources_used": [], "timewindow_start": null, "timewindow_end": null, "observed_behaviors": [] } }] }</pre>

GET /cases/{id} Retrieve case

- Click on the "Try it out" button, then click on the "Execute" button that appears. The request will be made, and the results will be displayed in the page.

The screenshot shows a REST client interface with the following sections:

- Method:** GET
- Path:** /cases Retrieve cases
- Parameters:** No parameters. A "Cancel" button is visible.
- Buttons:** "Execute" (highlighted in blue) and "Clear".
- Responses:** Response content type is set to application/json.
- Curl:** curl -X GET "https://demo.nl.cybraics.com/api/cases" -H "accept: application/json"
- Request URL:** https://demo.nl.cybraics.com/api/cases
- Server response:** Code 200, Details.
- Response body:** A JSON object containing a list of cases. The first case is:

```
{
  "data": [
    {
      "id": "14",
      "type": "cases",
      "attributes": {
        "id": 14,
        "title": "Suspicious VPN Connection Detected - tncyber - 213.74.161.218",
        "author_id": 4,
        "summary": "nLighten detected a suspicious VPN connection. \n**First Event:** 2019-06-26T10:52:47+00:00
\n\n\n**Behavior Name:** VPN Blacklist \n**Detection Source:** firewall \n**Source IP:** 213.74.161.218 \n**Destination IP:** 213.74.161.218 \n**Destination Country:** Turkey \n**Destination ASN:** Tellcom Iletisim Hizmetleri A.s.
\n",
        "created_at": "2019-06-26T10:53:24.395Z",
        "updated_at": "2019-10-11T18:23:04.723Z",
        "status": "closed",
        "deleted_at": null,
        "risk": 3,
        "opened_at": "2019-06-26T10:53:24.394Z",
        "threat_type_id": 4,
        "external_ticket_url": null,
        "external_ticket_title": null,
        "closed_at": "2019-10-11T18:23:04.597Z",
        "indexed_at": "2019-10-11T18:23:04.723Z",
        "analytics_used": [
          "VPNX"
        ]
      }
    }
  ]
}
```
- Response headers:** cache-control: max-age=0, private, must-revalidate



GraphQL API

Media Type

The GraphQL API uses an `application/json` media type.

Request HTTP header:

```
{ "Accept": "application/json" }
```

Response HTTP header:

```
{ "Content-Type": "application/json; charset=utf-8" }
```

Response Body Structure

The GraphQL API responds with the standard [GraphQL response format](#).

For example, using the following query:

```
query {
  outlierByld(id: 68100) {
    id
    analyticName
    timewindowStart
    timewindowEnd
    behavior {
      name
      category
    }
  }
}
```

will return a structure as shown below.

```
{
  "data": {
    "outlierByld": {
      "id": "68100",
      "analyticName": "PAX",
      "timewindowStart": "2019-07-25T05:13:45.000000Z",
      "timewindowEnd": "2019-07-25T05:14:13.000000Z",
      "behavior": {
        "name": "Palo Alto High Alert",
        "category": "unusual_user_behavior"
      }
    }
  }
}
```

The best way to learn the GraphQL API is to browse it using a GraphQL tool, such as [GraphQL Playground](#) or [GraphiQL](#). The following steps describe how to browse the API using GraphQL Playground.

1. Download the GraphQL Playground distribution for your platform from the URL below. The distributions are listed under "Assets" for the release you want to use. Install the distribution.

<https://github.com/prisma-labs/graphql-playground/releases>

The screenshot shows the GitHub release page for GraphQL Playground version 1.8.10. It includes a sidebar with navigation options like 'v1.8.10' and '77064d5', and a 'Compare' button. The main content area displays the version number '1.8.10', the release date 'Feb 23, 2019', and the number of commits '11 commits'. Below this, there is a section for 'Fixes' listing several issues and their resolutions. At the bottom, there is a table of 'Assets' with 13 items, including various platform-specific binaries and source code archives.

Latest release

v1.8.10
77064d5

Compare

1.8.10

huv1k released this on Feb 23, 2019 · 11 commits to master since this release

This release was mostly focused on fixing small bugs.

Fixes

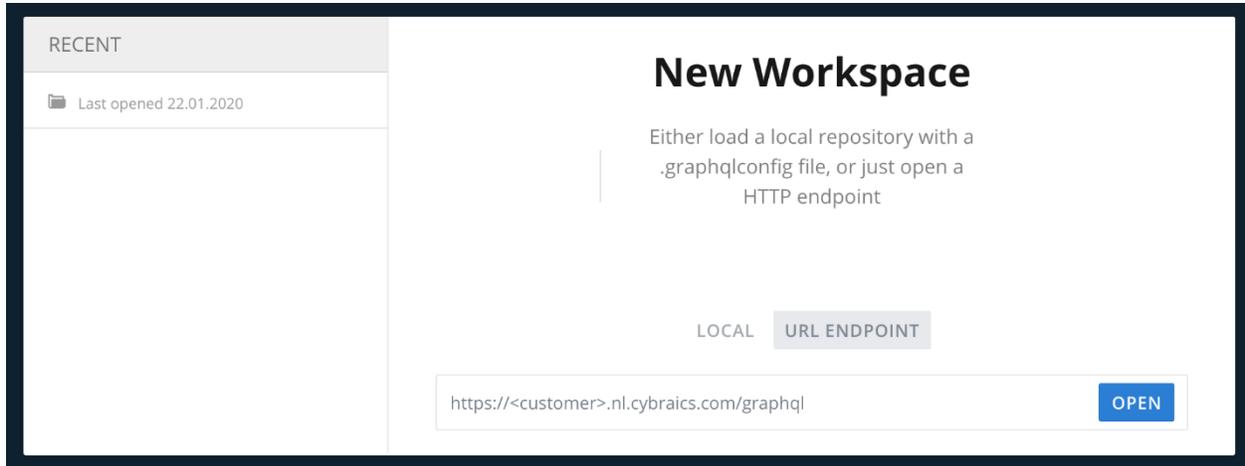
- #956 Thanks @lekterable fixing overflow in project sidenav
- #957 Thanks @yoshiakis for fixing subscription display in the light theme
- #960 Thanks @mmmeff for fixing an issue with license
- #961 Thanks @dncrews for fixing express middleware
- #968 Thanks @yoshiakis for fixing docs overflowing
- #969 Thanks @lekterable for adding hide option to electron app

Assets 13

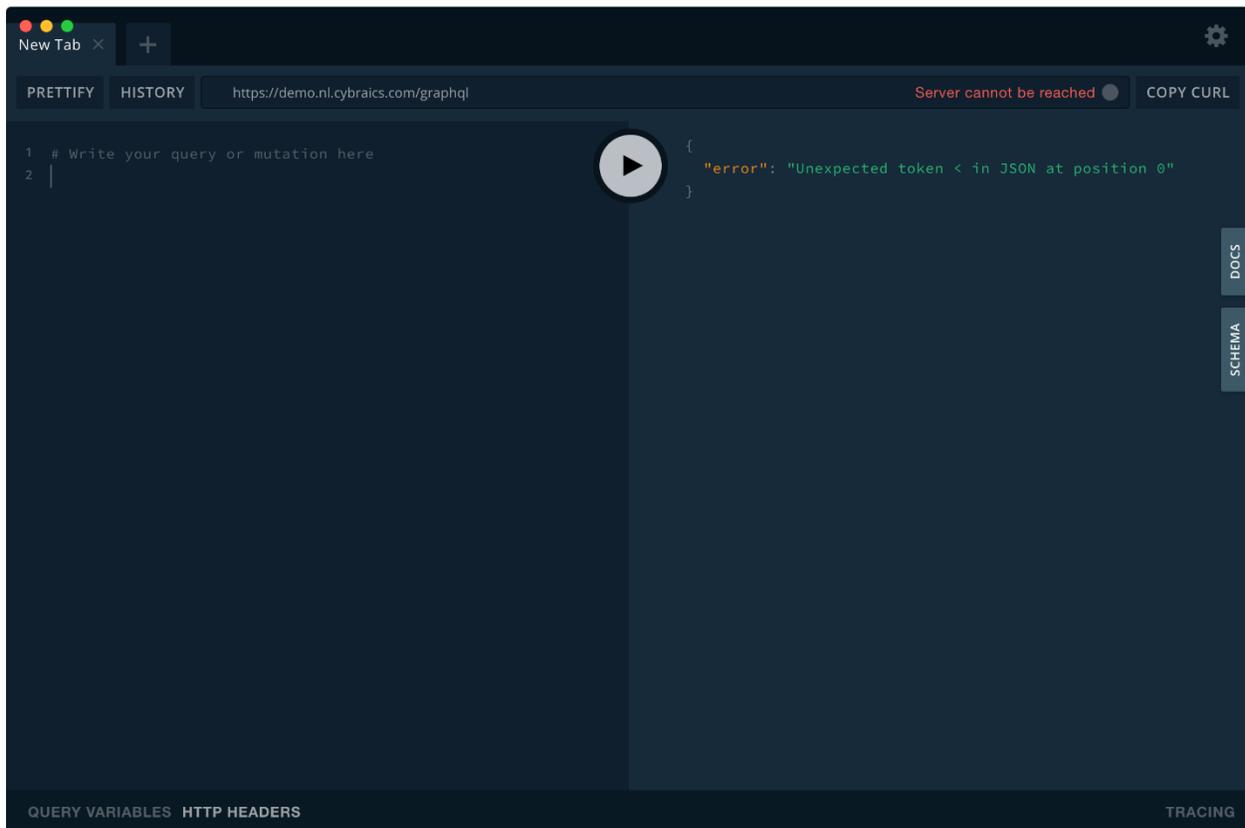
graphql-playground-electron-1.8.10-mac.zip	70.8 MB
graphql-playground-electron-1.8.10-x86_64.AppImage	74 MB
graphql-playground-electron-1.8.10.dmg	73.5 MB
graphql-playground-electron-1.8.10.dmg.blockmap	79.9 KB
graphql-playground-electron-setup-1.8.10.exe	52.5 MB
graphql-playground-electron-setup-1.8.10.exe.blockmap	57 KB
graphql-playground-electron_1.8.10_amd64.deb	44 MB
graphql-playground-electron_1.8.10_amd64.snap	125 MB
latest-linux.yml	421 Bytes
latest-mac.yml	573 Bytes
latest.yml	385 Bytes
Source code (zip)	
Source code (tar.gz)	

2. Run GraphQL Playground and you should see a window like the image below. Select "URL ENDPOINT" and enter the URL for the nLighten instance plus /graphql on the end.

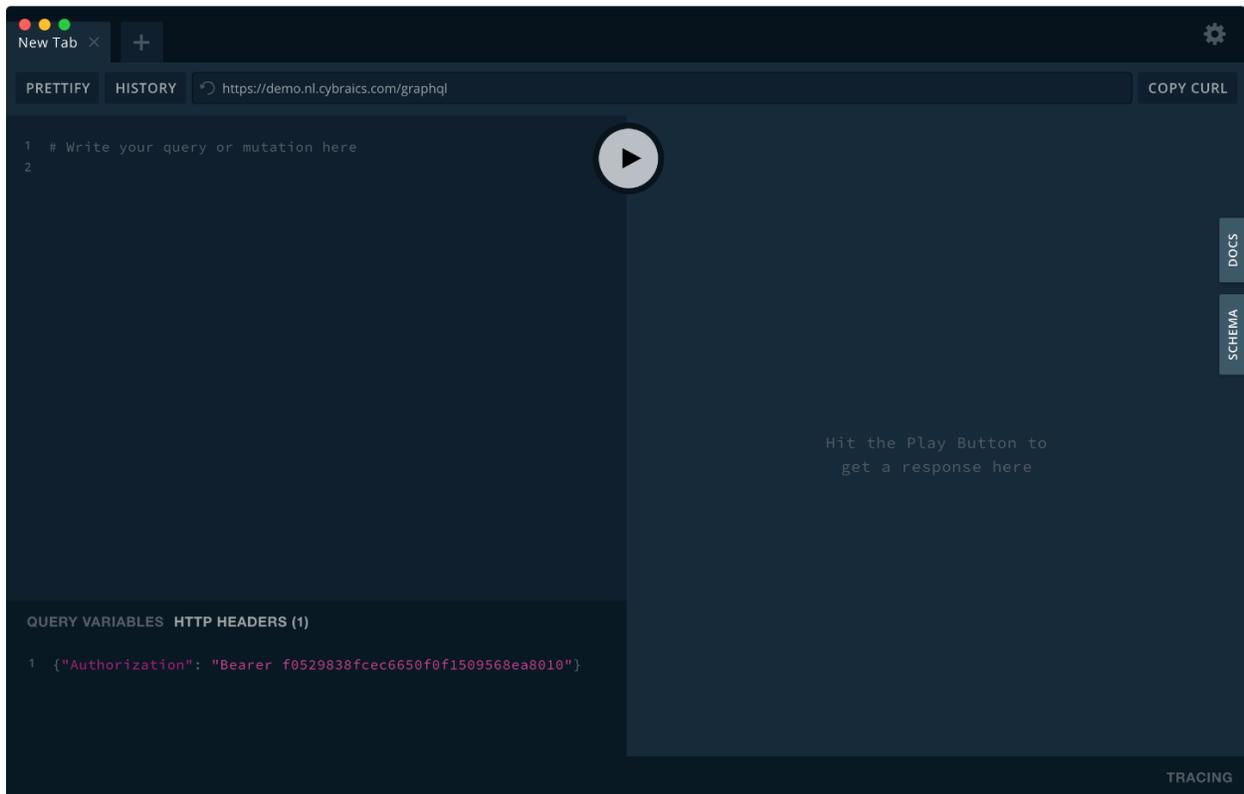
```
https://<pea>.nl.cybraics.com/graphql
```



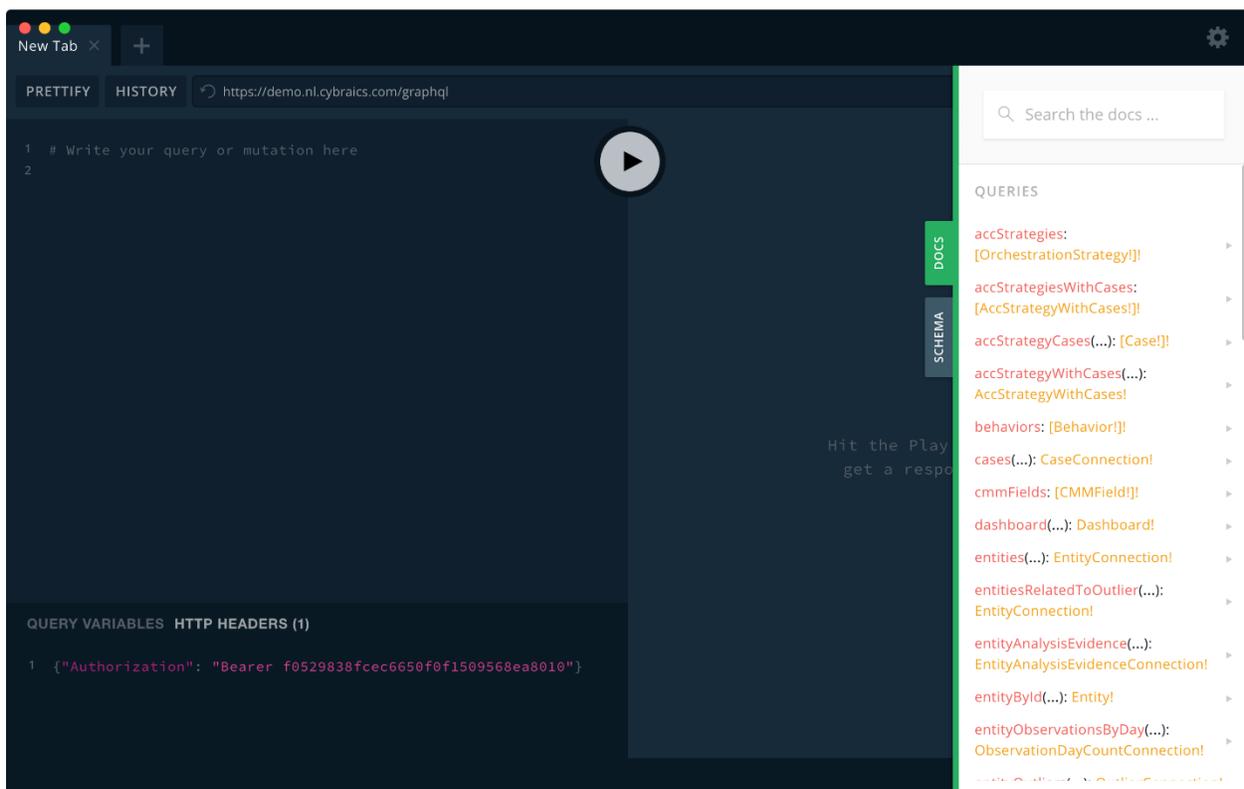
3. Click the "OPEN" button and you will see a screen that looks like the following image. There will be an error message displayed because you haven't added your access token yet so the server is responding with 401 Unauthorized.



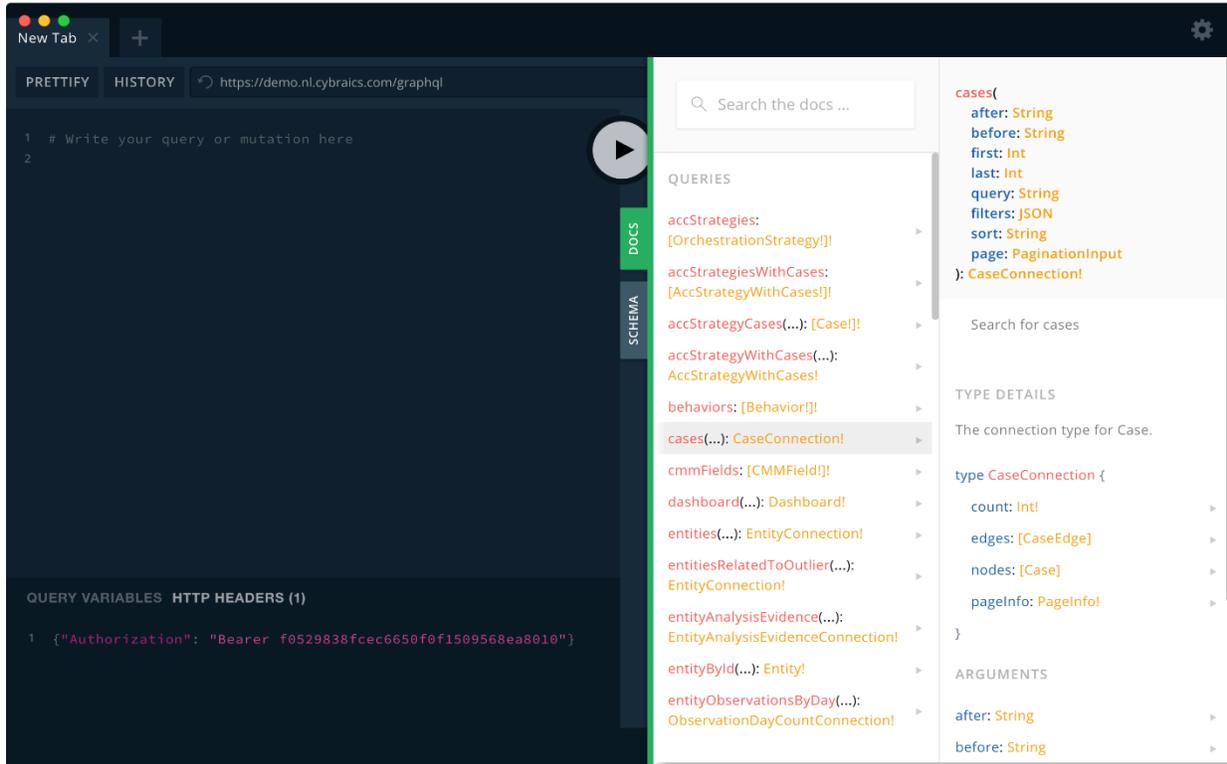
- Click on "HTTP HEADERS" at the lower left of the window. Enter your access token as shown in the image below.



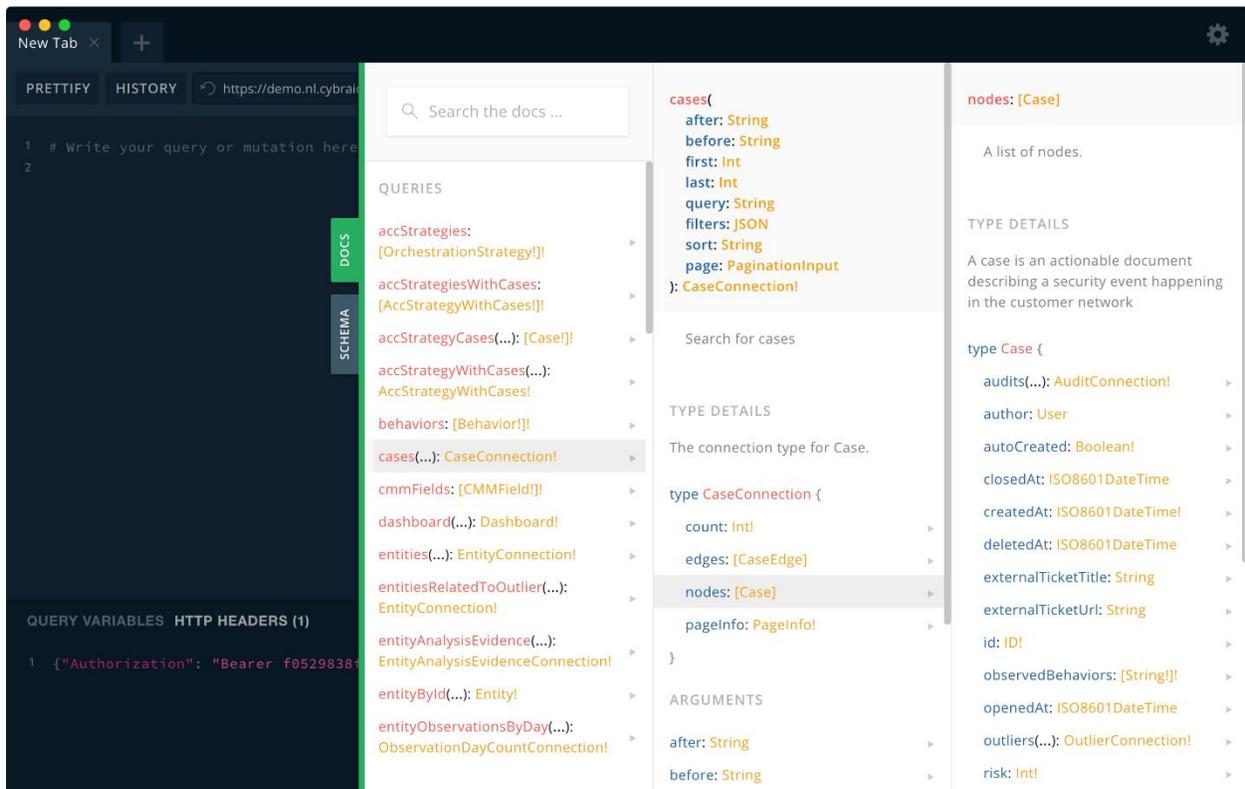
- Click on the "DOCS" tab on the right to see the list of available queries and mutations.



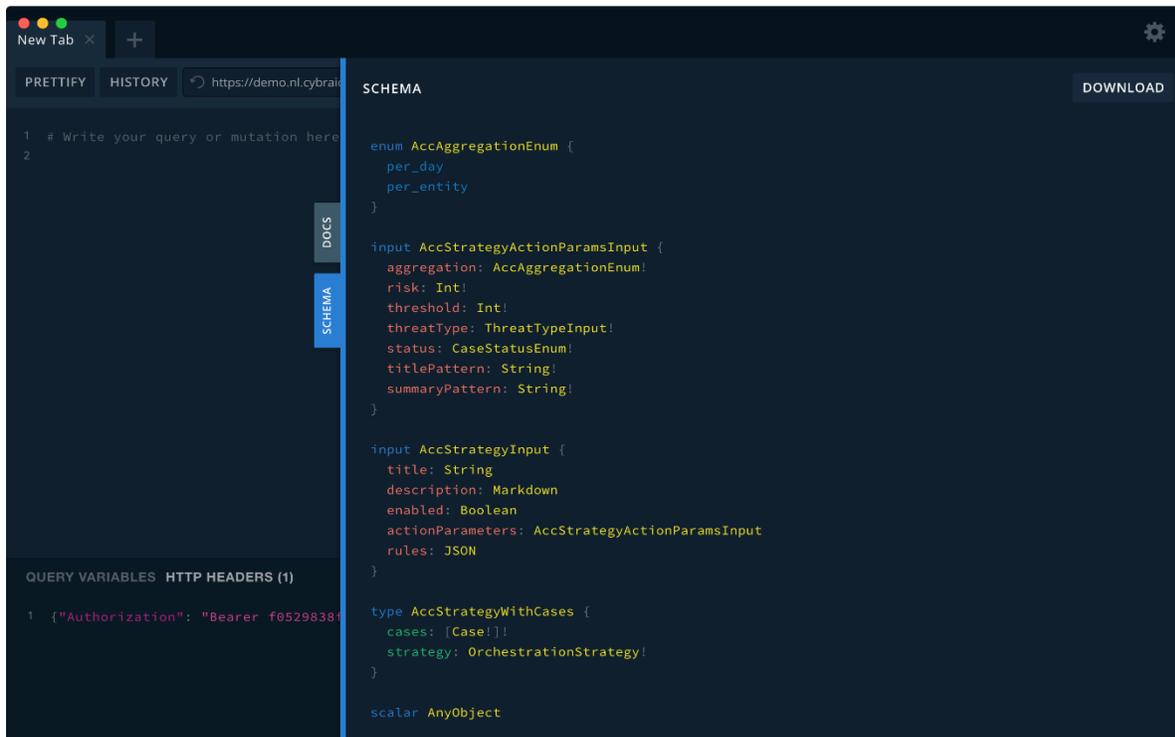
6. If you click on a query, it will expand to show details about that query. For example, the image below shows the case query.



If you click on a type in the query, it will expand further to show you information about that type. You can continue to click on types to drill down further.



7. You can also click on the "SCHEMA" tab to see the GraphQL schema.



```
enum AccAggregationEnum {
  per_day
  per_entity
}

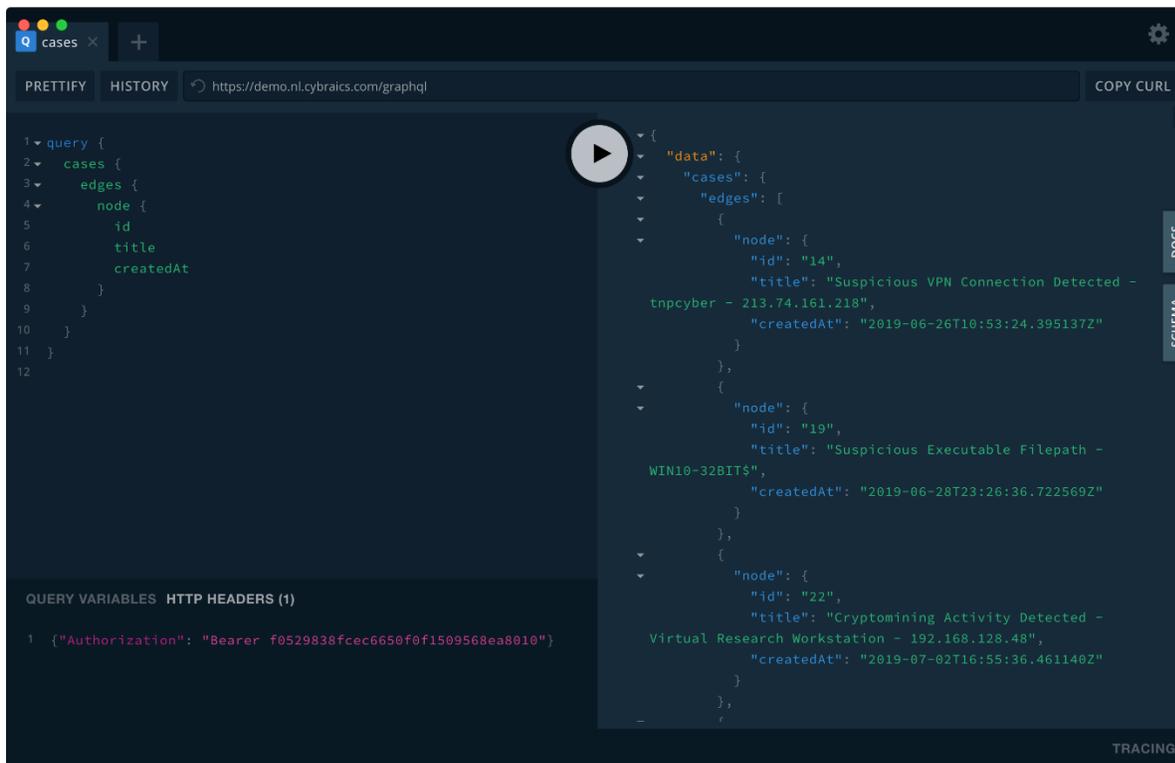
input AccStrategyActionParamsInput {
  aggregation: AccAggregationEnum!
  risk: Int!
  threshold: Int!
  threatType: ThreatTypeEnum!
  status: CaseStatusEnum!
  titlePattern: String!
  summaryPattern: String!
}

input AccStrategyInput {
  title: String!
  description: Markdown!
  enabled: Boolean!
  actionParameters: AccStrategyActionParamsInput!
  rules: JSON!
}

type AccStrategyWithCases {
  cases: [Case!]!
  strategy: OrchestrationStrategy!
}

scalar AnyObject
```

8. Finally, you can run a query, such as listing cases.



```
query {
  cases {
    edges {
      node {
        id
        title
        createdAt
      }
    }
  }
}
```

```
{
  "data": {
    "cases": {
      "edges": [
        {
          "node": {
            "id": "14",
            "title": "Suspicious VPN Connection Detected - tncyber - 213.74.161.218",
            "createdAt": "2019-06-26T10:53:24.395137Z"
          }
        },
        {
          "node": {
            "id": "19",
            "title": "Suspicious Executable Filepath - WIN10-32BIT$",
            "createdAt": "2019-06-28T23:26:36.722569Z"
          }
        },
        {
          "node": {
            "id": "22",
            "title": "Cryptomining Activity Detected - Virtual Research Workstation - 192.168.128.48",
            "createdAt": "2019-07-02T16:55:36.461140Z"
          }
        }
      ]
    }
  }
}
```

You can run both queries and mutations from GraphQL Playground but be *careful* because *mutations change data in the database!*