Configure Apollo Client for React and GitHub's GraphQL API

This lesson will teach you how to configure Apollo Client for React and GitHub's GraphQL API by introducing all the additional packages required for the setup.

WE'LL COVER THE FOLLOWING

Let's set up an Apollo Client instance as we did previously. However, this time we will use Apollo Client directly without the zero-configuration package- **Apollo Boost** which means that we'll need to configure the Apollo Client ourselves without any sensible defaults. While it's best to use a tool with sensible defaults for learning, configuring Apollo ourselves exposes the composable ecosystem of Apollo Client and makes clear on using it for initial setup and advancing the setup later.

Additional Packages Required

Two utility packages are required for two mandatory configurations which are used to create the Apollo Client:

1. The apollo-cache-inmemory is a recommended cache for the Apollo Client to manage the data.
2. The **apollo-link-http** is used to configure the URI and additional network information once for an Apollo Client instance.

There are two additional packages required for Apollo Client to work with GraphQL that are to be used as internal dependencies by Apollo:

1. graphql

2. graphql-tag

The latter is also used to define queries and mutations. Previously, these utilities came directly from Apollo Boost.

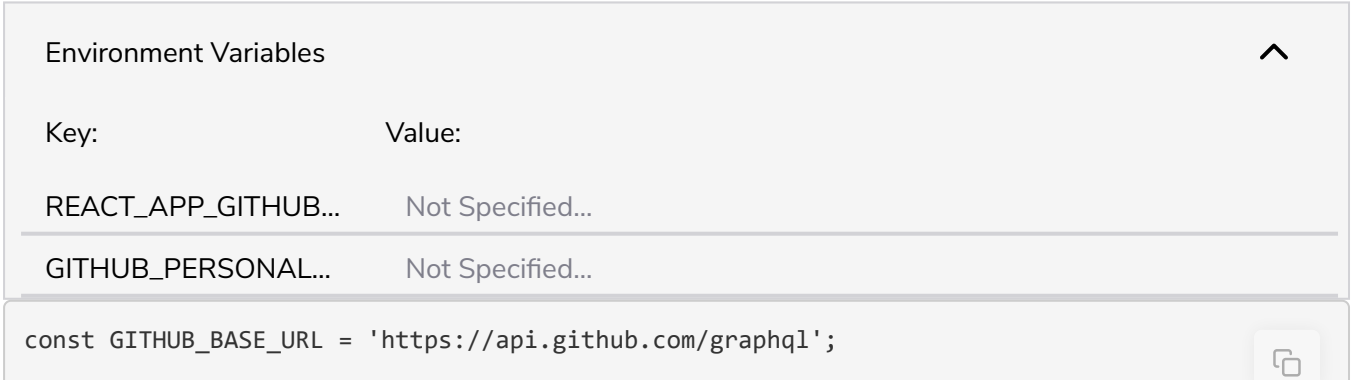
That's it for package installation! So now we have configured the Apollo Client setup.

In our top-level *src/index.js* file, where all the Apollo Client setup will be done in this lesson, we will import the necessary classes for the Apollo Client setup from the previously installed packages:

src/index.js

The `ApolloClient` class is used to create the client instance, and the `HttpLink` and `InMemoryCache` are used for its mandatory configurations.

First of all, we will create a configured `HttpLink` instance, which will be fed to the Apollo Client creation.



```
const httpLink = new HttpLink({
  uri: GITHUB_BASE_URL,
  headers: {
    authorization: `Bearer ${
      process.env.REACT_APP_GITHUB_PERSONAL_ACCESS_TOKEN
    }`,
  },
});
```

src/index.js

You may recall the mandatory configuration from previous applications.

The `uri` is a mandatory value to define the only GraphQL API endpoint used by the Apollo Client. In this case, Github's GraphQL endpoint is passed as value. When consuming the GitHub GraphQL API, you have to authorize yourself with your personal access token. You should have already defined the token in the previous chapters using the environment variables.

Secondly, let's create the cache as the place where the data is managed in the Apollo Client. The cache normalizes our data, caches requests to avoid duplicates, and makes it possible to read and write data to the cache. We will use it multiple times while developing this application. The cache instantiation is straight forward, as it doesn't require you to pass any arguments to it.

Environment Variables

Key:	Value:
REACT_APP_GITHUB...	Not Specified...
GITHUB_PERSONAL...	Not Specified...

```
const code = new InMemoryCache();
```

src/index.js

Finally, you can use both instantiated configurations, the link and the cache, to create the instance of the Apollo Client in the *src/index.js* file.

Environment Variables

Key:	Value:
REACT_APP_GITHUB...	Not Specified...

GITHUB_PERSONAL...	Not Specified...

```
const client = new ApolloClient({
  link: httpLink,
  cache,
});
```

src/index.js

To initialize Apollo Client, you must specify link and cache properties on the config object. Run the application below to verify the configuration and you will see a ‘Hello World’ output if everything works:

Environment Variables

Key:	Value:
REACT_APP_GITHUB...	Not Specified...
GITHUB_PERSONAL...	Not Specified...

```
import React from 'react';

import Link from '../../Link';

import './style.css';

const Footer = () => (
  <div className="Footer">
    <div>
      <small>
        <span className="Footer-text">Built by</span>{' '}
        <Link
          className="Footer-link"
          href="https://www.robinwieruch.de"
        >
          Robin Wieruch
        </Link>{' '}
        <span className="Footer-text">with &hearts;</span>
      </small>
    </div>
    <div>
      <small>
        <span className="Footer-text">
          Interested in GraphQL, Apollo and React?
        </span>{' '}
        <Link
          className="Footer-link"
          href="https://www.getrevue.co/profile/rwieruch"
        >
          Get updates
        </Link>{' '}
        <span className="Footer-text">
          about upcoming articles, books &
        </span>{' '}
```

```
        <Link className="Footer-link" href="https://roadtoreact.com">
          courses
        </Link>

        <span className="Footer-text">.</span>
      </small>
    </div>
  </div>
);

export default Footer;
```

If you run the application locally, there should be no errors but if it doesn't run, then check whether you have implemented a basic App component in your *src/App/index.js* file because the ReactDOM API needs to hook this component into the HTML.

Exercise

1. Confirm your source code for the last section.

Reading Task

1. Read more about the network layer configuration in Apollo Client