Plugins with Apollo

Plugins enable you to extend Apollo Server's core functionality by performing custom operations in response to certain events. Currently, these events correspond to individual phases of the GraphQL request lifecycle, and to the startup of Apollo Server itself (read more here). For example, a basic logging plugin might log the GraphQL query string associated with each request that's sent to Apollo Server.

Custom plugins

To create a plugin, declare a class annotated with the `@Plugin` decorator exported from the `@nestjs/apollo` package. Also, for better code autocompletion, implement the `ApolloServerPlugin` interface from the `@apollo/server` package.

```
import { ApolloServerPlugin, GraphQLRequestListener } from
'@apollo/server';
import { Plugin } from '@nestjs/apollo';

@Plugin()
export class LoggingPlugin implements ApolloServerPlugin {
  async requestDidStart(): Promise<GraphQLRequestListener<any>> {
    console.log('Request started');
    return {
      async willSendResponse() {
        console.log('Will send response');
      },
    };
  }
}
```

With this in place, we can register the `LoggingPlugin` as a provider.

```
@Module({
  providers: [LoggingPlugin],
})
export class CommonModule {}
```

Nest will automatically instantiate a plugin and apply it to the Apollo Server.

Using external plugins

There are several plugins provided out-of-the-box. To use an existing plugin, simply import it and add it to the `plugins` array:

```
GraphQLModule.forRoot({
  // ...
```

```
    plugins: [ApolloServerOperationRegistry({ /* options */})]
  }),
```

> info **Hint** The `ApolloServerOperationRegistry` plugin is exported from the `@apollo/server-plugin-operation-registry` package.

Plugins with Mercurius

Some of the existing mercurius-specific Fastify plugins must be loaded after the mercurius plugin (read more here) on the plugin tree.

> warning **Warning** mercurius-upload is an exception and should be registered in the main file.

For this, `MercuriusDriver` exposes an optional `plugins` configuration option. It represents an array of objects that consist of two attributes: `plugin` and its `options`. Therefore, registering the cache plugin would look like this:

```
GraphQLModule.forRoot({
  driver: MercuriusDriver,
  // ...
  plugins: [
    {
      plugin: cache,
      options: {
        ttl: 10,
        policy: {
          Query: {
            add: true
          }
        }
      },
    }
  ]
}),
```