13

Other Content

The deployment process, as defined in the deployment guide section, is for the whole project process. However, if a developer wants to deploy an individual service or component, they can still do so by cloning the code from GitLab/GitHub, making changes, and then deploying either manually or using Docker Compose, for example.

Run secure data access API using docker-compose:

- Clone the repository.

```
git clone git@gitlab.com:i3-market/code/sp2/secure-data-access-api.
```

- In the project root, create a .env file to insert environment variables. You have an example in templates/template.env.
- To start secure data access API, run with this command:

```
            docker-compose up –build
```

Local development components like *OpenId Provider*:
Clone the repository!

```
git clone git@ https://github.com/i3-Market-V3-Public-Repository/SP3-
SSIAM-NodeOidcProvider
```

13.1 Local Development using Node.js

To run the service locally using Node.js, it is necessary to download it before. After that, you can install the dependencies and start the service in the following way:

```
$ cd node-oidc-provider/app
$ npm i
$ npm start
```

You should also update the configuration file app/src/config.ts before running the service. Specifically, it is necessary to fill the default environment variables, in the same way they are filled in the .env file.

13.2 Local Development using Docker

Run the following command in the project root. The first time, it will take a while (be patient) since it has to build images and download all the npm dependencies.

```
./docker-dev-start
```

The OAS documentation can be accessed from http://localhost:3000/oidc/api-spec/ui.

You can stop the container at any time with Ctrl-C.

If you want to delete and prune all the created images, containers, networks, and volumes, just run:

```
./docker-dev-prune
```

Since the app directory is shared with the docker container with mapped user permissions, you can just edit any files in the app directory locally. The container will be running ts-node and nodemon to directly execute the source code and refresh the server if any file has changed. You can also attach any debugger in your local machine to the container, which will be listening at default port 9229.

13.2.1 Development scripts in the docker container

Besides rebuilding, you can execute any command in the oidc-provider-app container:

- to execute it in the running container:
  docker-compose -f docker-compose.dev.yaml exec oidc-provider-app <command>.
- to create and delete on-the-fly a new container (that will update the same files):
  docker-compose -f docker-compose.dev.yaml run –rm –no-deps oidc-provider-app <command>.