Wiremap - Getting started guide

Written by Luca Di Bello - luca.dibello@student.supsi.ch - 2022/2023 - Version 1.2.0

Table of contents

1. Prerequisites

The following tools are required to run the project:

- **Docker** version 17.06.0 or later (see installation instructions)
- **Docker Compose** version 1.29.0 or later (see installation instructions)
- **Rover CLI** v0.13.0 or later (see installation instructions)
- **Go** version 1.18 or later (see installation instructions)
- **Node.js** version 14 or later (see installation instructions)
- **npm** version 7.24.0 or later (see installation instructions) or **yarn** version 1.22.0 or later (see installation instructions)
- **make** version 3.81 or later (see installation instructions)

2. Running the project

To run the project, you need to follow these steps:

2.1. Clone the repository

```
git clone git@github.com:lucadibello/Wiremap.git
cd Wiremap
```

2.2. Generate a self-signed certificate for gRPC communication

> If you are deploying to production, edit the `certificate.conf` file by setting the `IP.4` to the IP address of the machine **where the recorder service will be running**. Then, run the following command:

```
make generate-cert
```

This command will generate a self-signed certificate (extension `.pem`), a `.cert` file and a `.key` file. The `.cert` and `.key` files will be used by the `decoder` service to enable TLS communication with the `recorder` service. The `.pem` file will be used by the `recorder` service to enable TLS communication with the `decoder` service'.

2.3. Configure decoder microservice

2.3.1. Setup ElasticSearch, Kibana and MongoDB for decoder microservice

Copy the `.env.example` file to `.env` and set the `MONGO_USERNAME` and `MONGO_PASSWORD` environment variables:

```
cp decoder/.env.example decoder/.env
vim decoder/.env
```

Start the required containers (ElasticSearch + Kibana + MongoDB):

```
cd decoder && docker-compose up -d && cd ..
```

Before starting the decoder service, both ElasticSearch and Kibana must be up and running. Kibana provides a graphical interface to explore data in ElasticSearch. During Kibana's first start, a setup link will appear in the logs to guide us through the setup process. Once Kibana is set up, we can configure the decoder service to use it. Example of unconfigured Kibana logs:

```
kibana           |
kibana           |
kibana           | i Kibana has not been configured.
kibana           |
kibana           | Go to http://0.0.0.0:5601/?code=450299 to get started.
kibana           |
kibana           |
```

2.3.2. Generate an enrollment token for Kibana

```
docker exec -it elasticsearch /bin/bash
./bin/elasticsearch-create-enrollment-token -s kibana
```

Paste this token in the configuration page.

Note: If Kibana requires a verification code, you can access it by running the following command:

```
docker exec -it kibana /bin/bash
./bin/kibana-verification-code
```

2.3.3. Reset ElasticSearch default user password

```
docker exec -it elasticsearch /bin/bash
./bin/elasticsearch-reset-password --auto --silent --batch -u elastic
```

Use the new password to log in to Kibana (**default username is `elastic`**).

> Note: Keep this password in a safe place, as it will be used to configure the `decoder` service later.

2.3.4. Generate the GraphQL schema

```
cd decoder && make generate && cd ..
```

This command will download all the required dependencies and generate the GraphQL schema.

2.3.5. Generate the JWT RSA-256 key pair

```
cd decoder && make generate-jwt-keypair && cd ..
```

This command generates the required RSA-256 key pair for JWT token generation.

2.3.6. Create the `config.yaml` file Now, we need to configure the settings for the `decoder` service. Copy the `decoder/configs/config.example.yaml` file to `decoder/configs/config.yaml` and set the required fields:

```
cp decoder/configs/config.example.yaml decoder/configs/config.yaml
vim decoder/configs/config.yaml
```

2.4. Configure recorder microservice

2.4.1. Generate the GraphQL schema

```
cd recorder && make generate && cd ..
```

This command will download all the required dependencies and generate the GraphQL schema.

2.4.2. Setup PostgreSQL for recorder microservice

We need to define the username and the password for PostgreSQL:

```
cp recorder/.env.example recorder/.env
vim recorder/.env
```

Now, we can start the required containers:

```
cd recorder && docker-compose up -d && cd ..
```

2.4.3. Create the `config.yaml` file

```
cp recorder/configs/config.example.yaml recorder/configs/config.yaml
vim recorder/configs/config.yaml
```

2.5. Start Keycloak and import the realm

> To import a realm, put your `realm.json` file in the `keycloak/config` directory before starting the container.

```
cd keycloak && docker-compose up -d && cd ..
```

2.6. Start the project

2.6.1. Start the recorder microservice

```
cd recorder && make start && cd ..
```

2.6.2. Start the decoder microservice

```
cd decoder && make start && cd ..
```

2.6.3. Build supergraph and start the router API gateway

```
cd router && make build && make dev && cd ..
```

2.6.4. Build and start the client web application

```
yarn install && yarn dev
```