rodeo-wp2-poc Documentation

Chapter 1. Introduction

This is the documentation for Rodeo work package 2 prototype. Which outlines its purpose functionality, usage and build instructions.

Chapter 2. Components

The rodeo-wp2-poc project consist of **five** different components:

API Service/Manager
: Python program which handles the rewriting o metadata links and the proxying of actual data. Also, includes serverless.yml file for quick AWS deployment. Found in directory `./api-manager-demo`

Shared catalogue/Metadata catalogue
: Stores and validates wcmp2 discovery metadata records. This code is forked from WMO-project Wis2-gdc [https://github.com/wmo-im/wis2-gdc] . Found in git submodule `./rodeo-poc-wis2-gdc`

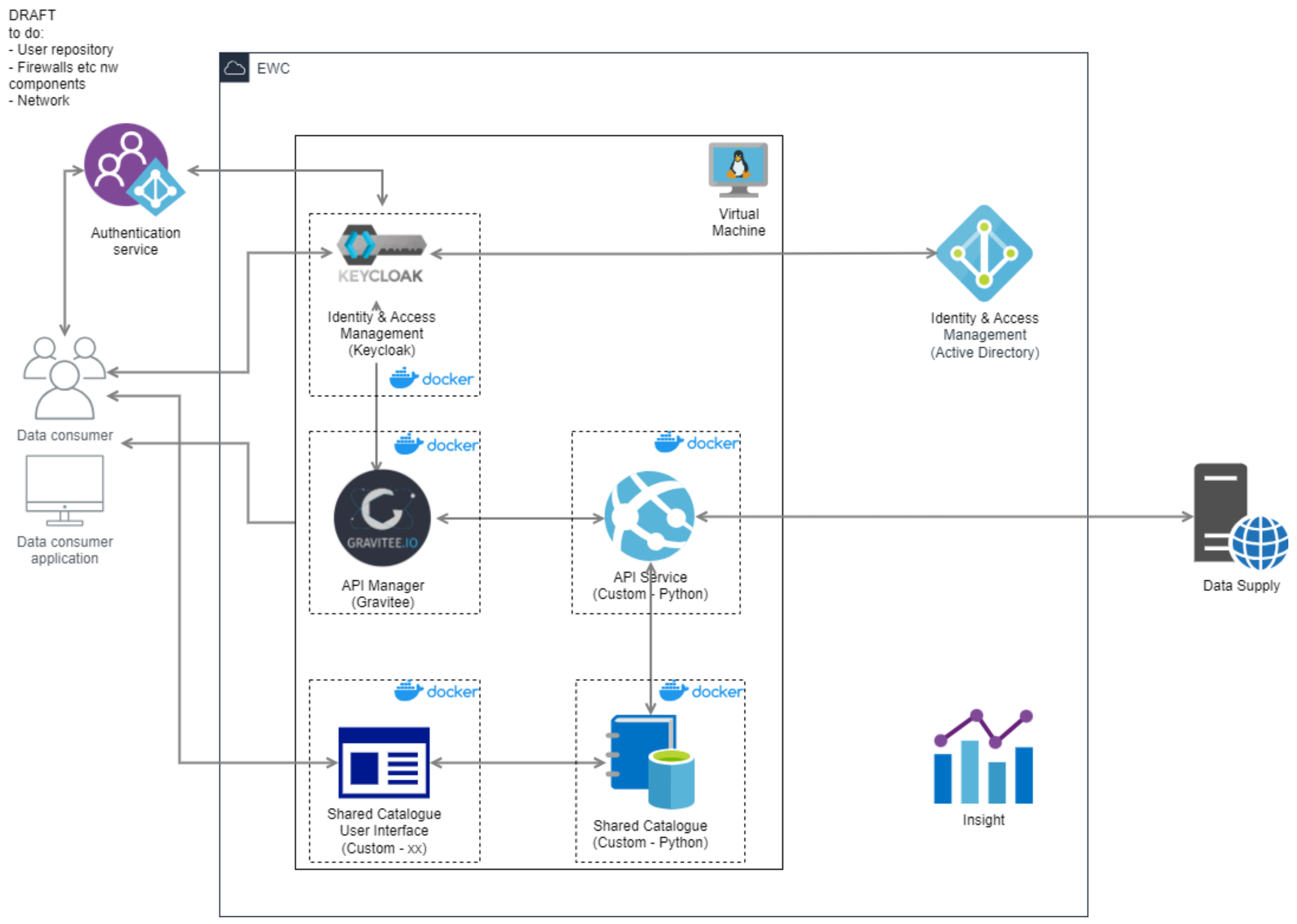
Shared catalogue UI
: Next.js based web UI. Presents the discovery metadata as browsable elements and links to actual data collections. Found in directory `./catalog-ui`

Gravitee
: Gravitee is used as an API Manager in front of Shared catalogue and API Service. Default config is found in `./gravitee-config` directory and `./gravitee-api-definitions` contains default API-definitions to import to Gravitee.

Keycloak
: Keycloak is used as an identity and access provider for Gravitee. Needed Keycloak Gravitee plugins are stored in directory `./gravitee`

2.1. EWC

EWC deployment scenario is handled by docker compose and Nginx is used as a reverse proxy for Gravitee and for setting up possible TLS/SSL connection. Every service is registered as Gravitee API expect for Keycloak.


DRAFT
to do:
- User repository
- Firewalls etc nw components
- Network
EWC
Authentication service
Virtual Machine
KEYCLOAK
Identity & Access Management (Keycloak)
docker
Identity & Access Management (Active Directory)
Data consumer
Data consumer application
GRAVITEE.IO
API Manager (Gravitee)
API Service (Custom - Python)
Data Supply
Shared Catalogue User Interface (Custom - xx)
Shared Catalogue (Custom - Python)
Insight

2.2. AWS

AWS is used to host the python API Service while the AWS API Gateway is used instead of Gravitee. Identity and access is handled by AWS Cognito instead of Keycloak.

The API Service and API Gateway are deployed using Serverles wsgi [https://www.serverless.com/plugins/serverless-wsgi] based on `./api-manager-demo/serverless.yml` which creates a Cloudformation stack to host the API Service as Lambda function and an API Gateway to proxy connections to the Lambda.

Chapter 3. Usage instructions.

3.1. Changing default config values:

3.1.1. .env

The repository has a template for environment variables and can be found from file `.env copy` at the root of the repo.

- Change `KC_DB_PASSWORD` and `POSTGRESQL_PASS` to the same value as both connect to the same database.
- Also change `KEYCLOAK_ADMIN_PASSWORD` as this will be your default admin password for Keycloak's Administration Console
- We will change `ELASTICSEARCH_TOKEN` later after we have configured Gravitee API plans.

```
MONGODB_VERSION=4.4.18
APIM_VERSION=3

KC_DB=postgres
KC_DB_URL_HOST=postgres
KC_DB_URL_DATABASE=keycloakdb
KC_DB_USERNAME=keycloak
KC_DB_PASSWORD=<change_me>
KC_DB_SCHEMA=public
KEYCLOAK_ADMIN=admin
KEYCLOAK_ADMIN_PASSWORD=<change_me>

POSTGRESQL_VERSION=15.4
POSTGRESQL_DB=keycloak
POSTGRESQL_USER=keycloak
POSTGRESQL_PASS=<change_me>

GRAVITEE_API_BASE_URL=http://localhost:8083

ELASTICSEARCH_TOKEN=1234
EWC_URL=http://localhost/gateway/api-manager
AWS_URL=http://example.org
APIMANAGER_BASE_URL=http://localhost/gateway/api-manager

UI_BASE_PATH=/ui
```

3.1.2. gravitee.yml

- Change the default password for Gravitee admin by changing property `password` inside of file `./gravitee-config/gravitee.yml`
  - Default config uses bcrypt to hash passwords, so to generate new ones you can use

command line tool htpasswd. Where new_password is your password:

```
htpasswd -bnBC 10 "" new_password | tr -d ':\n'
```

- See: https://docs.gravitee.io/apim/3.x/apim_installguide_authentication_inmemory.html

3.2. Running

Run `docker compose up` at the root of the project

3.3. Import Gravitee API definitions

- Login to the Gravitee admin console on `localhost/management_ui` using your admin credentials that we set previously.
- Navigate to: APIs -→ Add API -→ Import an API
- Import all the API definitions inside `./gravitee-api-definitions` directory

3.3.1. Add JWT Token to Collection API

- Go to APIs -→ Collection -→ Start creating a plan
- Create a new plan and set Authentication type to JWT
- Follow this tutorial [https://www.gravitee.io/blog/secure-apis-with-jwt-tokens] to generate keys for JWT.

3.3.2. Subscribe to Collection API

- Go to Applications -→ Add application
- Set Client ID to value you remember as this will be needed when generating JWT token.
- Subscribe to the plan you created on the previous step.
- Generate new JWT Token Generate a new JWT Token using the keys and Client_id you created on previous steps.
  - For example using this site [https://jwt.io/]
    - NOTE: ONLY USE THIS WEBSITE WITH TEST KEY PAIRS!
- Set the payload to include client_id with the Client id you created previously

3.3.3. Update .env

- Copy the generated JWT Token to `.env` and update `ELASTICSEARCH_TOKEN` value
- For example:

```
ELASTICSEARCH_TOKEN=12345678901011AbcdefgHijlkmv
```

- Restart the apimanager service

```
docker compose up apimanager
```

3.4. Add example data

The project have some example metadata in `./metadata-files` directory. The directory is mounted to the metadata catalog container by default.

- You can add some example metadata to the metadata catalog by attaching to the catalog container:

```
docker exec -it catalog /bin/bash
```

- Run the catalog to register new metadata

```
wis2-gdc register /metadata-files
```

Chapter 4. Deploy to AWS

API Service/Manager can be deployed to Amazon Web Services using Serverles wsgi [https://www.serverless.com/plugins/serverless-wsgi]

- Make sure you have installed and configured npm and AWS CLI [https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html] with your credentials.
- Install serverless framework by running `npm install -g serverless` or following alternative instructions here [https://www.serverless.com/framework/docs/getting-started]
- Navigate to `./api-manager-demo` directory
- Run `serverless deploy`

Serverless deploys the AWS lambda and AWS API Gateway to eu-north-1 region and provides a URL where the API Gateway is running.

- Go to https://eu-north-1.console.aws.amazon.com/lambda
- Navigate to your function using AWS portal
- Go to Configuration -→ Environment variables
  - Change `BASE_URL` to your AWS API Gateway root path.
  - `ELASTICSEARCH_TOKEN` should be the JWT token that Gravitee Collections API uses
  - `ELASTICSEARCH_URL` should point to the URL where Gravitee gateway is running

For example:

```
BASE_URL=1234api.eu-north-1.amazonaws.com/dev
ELASTICSEARCH_TOKEN=1234
ELASTICSEARCH_URL=http://example.org:80/gateway
```

- Click Save