FLOWX.AI

PLATFORM DEEP DIVE / Plugins / Plugins setup guides / reporting-setup

Contents

PLATFORM DEEP DIVE / Plugins / Plugins setup guides / Reporting setup guide

The reporting plugin is available a docker image, and it has the following dependencies:

Dependencies

- a reporting PostgreSQL instance
- reporting-plugin helm chart - containing cronJob which performs the following actions:

  - reads from FLOWX.AI Engine db
  - writes in the FLOWX.AI Reporting plugin db
- Superset:
  - a Superset PostgreSQL db
  - a Redis instance for caching
  - exposes the UI through an ingress -> host needed

Postgres database

Basic Postgres configuration:

```
postgresql:
  enabled: true
  postgresqlUsername: {{userName}}
  postgresqlPassword: ""
  postgresqlDatabase: "reporting"
  existingSecret: {{scretName}}
  persistence:
    enabled: true
    storageClass: standard-rwo
    size: 5Gi
  resources:
    limits:
      cpu: 1000m
      memory: 1024Mi
    requests:
      memory: 256Mi
      cpu: 100m
  metrics:
    enabled: true
    serviceMonitor:
      enabled: false
```

```
    prometheusRule:
      enabled: false
  primary:
    nodeSelector:
      preemptible: "false"
```

Reporting plugin helm chart (containing CRON)

reporting-plugin helm.yaml

```
sync:
  cronjob:
    image:
      repository: {{env}}/reporting-plugin

    schedule: "*/5 * * * *"

    extraEnvVarsMultipleSecretsCustomKeys:
      - name: process-engine-application-config
        secrets:
          ENGINE_DATABASE_PASSWORD: {{db paswword}}
        secrets:
          REPORTING_DATABASE_PASSWORD: {{db password}}

    env:
      ENGINE_DATABASE_USER: {{engine db user}}
      ENGINE_DATABASE_URL: {{engine db URL}}
      ENGINE_DATABASE_NAME: {{engine db name}}

      REPORTING_DATABASE_USER: {{reporting db user}}
      REPORTING_DATABASE_URL: {{reporting db URL}}
```

```
        REPORTING_DATABASE_NAME: {{reporting db name}}
```

Superset

» Superset configuration

» Superset documentation

After installation

- datasource URL -> FLOWX.AI Reporting database
- Datasets
- Dashboards

Datasource configuration

To store data related to document templates and documents the service uses a Postgres database.

The following configuration details need to be added using environment variables:

`SPRING_DATASOURCE_URL`

`SPRING_DATASOURCE_USERNAME`

`SPRING_DATASOURCE_PASSWORD`

You will need to make sure that the user, password, connection link and db name are configured correctly, otherwise you will receive errors at start time.

The datasource is configured automatically via a liquibase script inside the service. All updates will include migration scripts.

> **INFO**
>
> Database schema is managed by a liquibase script that will create, manage and migrate future versions.

Redis configuration

The following values should be set with the corresponding Redis-related values:

`SPRING_REDIS_HOST`

`SPRING_REDIS_PORT`

Keycloak configuration

To enable a different user authentication than the regular one (database), you need to override the `AUTH_TYPE` parameter in your `superset` .yml file.

It would look something like this:

```
AUTH_TYPE: AUTH_OID
```

You will also need to provide a reference to your `openid-connect` realm:

```
OIDC_OPENID_REALM: 'flowx'
```

With this configuration, the login page changes to a prompt where the user can select the desired OpenID provider (in our case keycloak)

Extend the Security Manager

Firstly, you will want to make sure that flask stops using `flask-openid` and starts using `flask-oidc` instead.

To do so, you will need to create your own security manager that configures `flask-oidc` as its authentication provider.

```
extraSecrets:
  keycloak_security_manager.py: |
    from flask_appbuilder.security.manager import AUTH_OID
    from superset.security import SupersetSecurityManager
    from flask_oidc import OpenIDConnect
```

To enable OpenID in Superset, you would previously have had to set the authentication type to `AUTH_OID`.

The security manager still executes all the behavior of the super class, but overrides the OID attribute with the `OpenIDConnect` object.

Further, it replaces the default OpenID authentication view with a custom one:

```
    from flask_appbuilder.security.views import AuthOIDView
    from flask_login import login_user
    from urllib.parse import quote
    from flask_appbuilder.views import expose
    from flask import request, redirect

    class AuthOIDCView(AuthOIDView):
        @expose('/login/', methods=['GET', 'POST'])
        def login(self, flag=True):
            sm = self.appbuilder.sm
            oidc = sm.oid
            superset_roles = ["Admin", "Alpha", "Gamma",
"Public", "granter", "sql_lab"]
            default_role = "Admin"
            @self.appbuilder.sm.oid.require_login
            def handle_login():
                user =
sm.auth_user_oid(oidc.user_getfield('email'))
                if user is None:
                    info =
oidc.user_getinfo(['preferred_username', 'given_name',
'family_name', 'email', 'roles'])
                    roles = [role for role in superset_roles
if role in info.get('roles', [])]
                    roles += [default_role, ] if not roles
else []
                    user =
sm.add_user(info.get('preferred_username'),
info.get('given_name', ''), info.get('family_name', ''),
                                            info.get('email'),
[sm.find_role(role) for role in roles])
                login_user(user, remember=False)
                return
redirect(self.appbuilder.get_url_for_index)
```

```
            return handle_login()
        @expose('/logout/', methods=['GET', 'POST'])
        def logout(self):
            oidc = self.appbuilder.sm.oid
            oidc.logout()
            super(AuthOIDCView, self).logout()
            redirect_url = request.url_root.strip('/')
            # redirect_url = request.url_root.strip('/') +
self.appbuilder.get_url_for_login
            return redirect(
                oidc.client_secrets.get('issuer') +
'/protocol/openid-connect/logout?redirect_uri=' +
quote(redirect_url))
```

On authentication, the user is redirected back to Superset.

Configure Superset

Finally, we need to add some parameters to the superset .yml file:

```
    '''
    ------------------------KEYCLOACK ------------------
---------
    '''
    curr  =  os.path.abspath(os.getcwd())
    AUTH_TYPE = AUTH_OID
    OIDC_CLIENT_SECRETS =  curr +
'/pythonpath/client_secret.json'
    OIDC_ID_TOKEN_COOKIE_SECURE = True
    OIDC_REQUIRE_VERIFIED_EMAIL = True
    OIDC_OPENID_REALM: 'flowx'
    OIDC_INTROSPECTION_AUTH_METHOD: 'client_secret_post'
    CUSTOM_SECURITY_MANAGER = OIDCSecurityManager
```

```
    AUTH_USER_REGISTRATION = False
    AUTH_USER_REGISTRATION_ROLE = 'Admin'
    OVERWRITE_REDIRECT_URI = 'https://{{
.Values.flowx.ingress.reporting }}/oidc_callback'
    '''
    ------------------------------------------------------------
------
    '''
```

Was this page helpful?