PLATFORM SETUP GUIDES / platform-setup-guides-docs

Contents

- PLATFORM SETUP GUIDES / Overview
  - Environment variables
  - Authorization & access roles
  - Datasource configuration
  - Kafka
  - Redis configuration
  - Debugging
  - Logging
  - Tracing via Jaeger
  - License model
  - Third-party components

PLATFORM SETUP GUIDES / Overview

The setup guides in this section will provide information on how to install, configure, and use FLOWX.AI services.

Deploying microservices typically involves breaking down the application into smaller, modular components. Each microservice should be independently deployable, with all the necessary dependencies and configurations included.

Once the microservices have been created, they can be deployed using a container management system such as Docker or Kubernetes. These systems allow for the deployment of multiple microservices in a single environment.

Environment variables

Environment variables are variables that are set in the system environment and can be used by applications and services to store and access configuration information. Environment variables typically include settings such as paths to directories, file locations, settings for the operating system and applications, and more.

Environment variables are used to store and access configuration information in a secure and efficient manner. Below you will find some examples of common/shared environment variables that need to be set for different services and components.

Authorization & access roles

An identity management platform is a software system that helps you manage authorization & access roles, including user accounts, passwords, access control, and authentication. Identity management platforms typically offer features such as user provisioning, identity federation, and single sign-on.

The following variables need to be set in order to connect to the identity management platform:

- `SECURITY_OAUTH2_BASE_SERVER_URL` - the base URL for the OAuth 2.0 Authorization Server, which is responsible for authentication and authorization for clients and users, it is used to authorize clients, as well as to issue and validate access tokens

- `SECURITY_OAUTH2_CLIENT_CLIENT_ID` - a unique identifier for a client application that is registered with the OAuth 2.0 Authorization Server, this is used to authenticate the client application when it attempts to access resources on behalf of a user
- `SECURITY_OAUTH2_CLIENT_CLIENT_SECRET` - secret key that is used to authenticate requests made by an authorization client
- `SECURITY_OAUTH2_REALM` - security configuration env var in the Spring Security OAuth2 framework, it is used to specify the realm name used when authenticating with OAuth2 providers

» Access Management

Datasource configuration

Datasource configuration is the process of configuring a data source, such as a database, file, or web service, so that an application can connect to it and use the data. This typically involves setting up the connection parameters, such as the host, port, username, and password.

In some cases, additional configuration settings may be required, such as specifying the type of data source (e.g. Oracle, MySQL, etc.) or setting up access control for data access.

Environment variables are more secure than hard-coding credentials in the application code and make it easier to update data source parameters without having to modify the application code.

CAUTION

Some microservices (**Admin** microservice, for example, connects to the same Postgres / Oracle database as the **Engine**).

INFO

Depending on the data source type, various parameters may need to be configured. For example, if connecting to an Oracle database, the driver class name, and the database schema must be provided. For MongoDB, the URI is needed.

The following variables need to be set in order to set the datasource:

- `SPRING_DATASOURCE_URL` - environment variable used to configure a data source URL for a Spring application, it typically contains the JDBC driver name, the server name, port number, and database name
- `SPRING_DATASOURCE_USERNAME` - environment variable used to set the username for the database connection, this can be used to connect to a database instance
- `SPRING_DATASOURCE_PASSWORD` - environment variable used to store the password for the database connection, this can be used to secure access to the database and ensure that only authorized users have access to the data
- `SPRING_DATASOURCE_DRIVERCLASSNAME` (❗ only for Oracle DBs) - environment variable used to set the class name of the JDBC driver that the Spring DataSource will use to connect to the database

- `SPRING_JPA_PROPERTIES_HIBERNATE_DEFAULTSCHEMA` (❗ only for Oracle DBs) - environment variable used to overwrite the name of the database schema

- `SPRING_DATA_MONGODB_URI` (❗ only for MongoDB) - environment variable used to provide the connection string for a MongoDB database that is used with, this connection string provides the host, port, database name, user credentials, and other configuration details for the MongoDB server

> ⚠ **CAUTION**
>
> 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.

> ⓘ **INFO**
>
> The datasource is configured automatically via a liquibase script inside the engine. All updates will include migration scripts.

Kafka

The following Kafka-related configurations can be set by using environment variables:

- `SPRING_KAFKA_BOOTSTRAP_SERVERS` - environment variable used to configure the list of brokers to which the kafka client will connect, this is a comma-separated list of host and port pairs that are the addresses of the Apache Kafka brokers in a Kafka cluster

- `SPRING_KAFKA_CONSUMER_GROUP_ID` - environment variable is used to set the consumer group ID for the Kafka consumer, it is used to identify which consumer group the consumer belongs to and allows the Kafka broker to manage which messages are consumed by each consumer in the group

INFO

`SPRING_KAFKA_CONSUMER_GROUP_ID` - might be different for the services that have the group id separated in topics, also thread numbers.

- `KAFKA_CONSUMER_THREADS` - environment variable used to control the number of threads that a Kafka consumer instance can use to consume messages from a cluster, it defines the number of threads that the consumer instance should use to poll for messages from the Kafka cluster

- `KAFKA_AUTH_EXCEPTION_RETRY_INTERVAL` - environment variable used to set the interval at which Kafka clients should retry authentication exceptions (the interval between retries after AuthorizationException is thrown by KafkaConsumer)

- `KAFKA_MESSAGE_MAX_BYTES` - this is the largest size of the message that can be received by the broker from a producer.

Each action available in the service corresponds to a Kafka event. A separate Kafka topic must be configured for each use case.

CAUTION

FLOWX.AI Engine is listening for messages on topics with names of a certain pattern, make sure to use correct outgoing topic names when configuring the services.

Redis configuration

Redis configuration involves setting up the connection parameters, such as the host, port, username, and password. In some cases, additional configuration settings may be required, such as specifying the type of data store or setting up access control for data access.

- `SPRING_REDIS_HOST` - environment variable used to configure the hostname or IP address of a Redis server when using Spring Data Redis
- `SPRING_REDIS_PASSWORD` - environment variable is used to store the password used to authenticate with a Redis server, it is used to secure access to the Redis server and should be kept confidential
- `REDIS_TTL` - environment variable is used to specify the maximum time-to-live (TTL) for a key in Redis, it is used to set a limit on how long a key can exist before it is automatically expired (Redis will delete the key after the specified TTL has expired)

Debugging

Advanced debugging features can be enabled. When this happens, snapshots of the process status will be taken after each action and can be later used for debugging purposes. This feature comes with an exponential increase in database usage, so we suggest having the flag set to true on debugging media and false production ones.

Logging

The following environment variables could be set in order to control log levels:

- `LOGGING_LEVEL_ROOT` - root spring boot microservice logs
- `LOGGING_LEVEL_APP` - controls the verbosity of the application's logs and how much information is recorded (app level logs)

Tracing via Jaeger

Tracing via Jaeger involves collecting timing data from the components in a distributed application. This allows you to better identify bottlenecks and latency issues.

The following FLOWX.AI services use Jaeger tracing:

1. **scheduler-core**
2. **customer-management-plugin**
3. **document-plugin**
4. **notification-plugin**
5. **process-engine**

Environment variables to be set for tracing:

- `APPLICATION_JAEGER_ENABLED` - environment variable used to enable or disable Jaeger tracing
- `APPLICATION_JAEGER_PREFIX` - environment variable used to change the name in the Jaeger dashboard

License model

A license model is a set of rules and regulations governing how software can be used, distributed, and modified. It also outlines the rights and responsibilities of the software user and the software developer. Common license models include open source, freeware, shareware, and commercial software.

Most of the **third-party components used by FLOWX.AI** are under **Apache License 2.0** source code.

Third-party components

Third-party components are software components or libraries that are not part of FLOWX.AI but are instead created by another company or individual and used in a development project.

These components can range from databases and operating systems to user interface components and libraries that provide support for a specific feature or task.

Third party components are components such as libraries, frameworks, APIs, etc.

» Third-party components

Was this page helpful?