FLOWX.AI

PLATFORM SETUP GUIDES / events-gateway-setup

Contents

PLATFORM SETUP GUIDES / Events gateway setup guide

Introduction

This guide will walk you through the process of setting up the events-gateway service.

Infrastructure prerequisites

Before proceeding with the setup, ensure that the following components have been set up:

- **Redis** - version 6.0 or higher
- **Kafka** - version 2.8 or higher

Dependencies

- **Kafka** - used for event communication
- **Redis** - used for caching

Configuration

Configuring Kafka

Set the following Kafka-related configurations using environment variables:

- `SPRING_KAFKA_BOOTSTRAP_SERVERS` - the address of the Kafka server, it should be in the format "host:port"

Groupd IDs

The configuration parameters "KAFKA*CONSUMER_GROUP_ID*" are used to set the consumer group name for Kafka consumers that consume messages from topics. Consumer groups in Kafka allow for parallel message processing by distributing the workload among multiple consumer instances. By configuring the consumer group ID, you can specify the logical grouping of consumers that work together to process messages from the same topic, enabling scalable and fault-tolerant message consumption in your Kafka application.

Configuration Parameter	

Configuration Parameter	
`KAFKA_CONSUMER_GROUP_ID_PROCESS_ENGINE_COMMANDS_MESSAGE`	e c m
`KAFKA_CONSUMER_GROUP_ID_PROCESS_ENGINE_COMMANDS_DISCONNECT`	e c d
`KAFKA_CONSUMER_GROUP_ID_PROCESS_ENGINE_COMMANDS_CONNECT`	e c c

Configuration Parameter	
`KAFKA_CONSUMER_GROUP_ID_PROCESS_TASK_COMMANDS`	t c m

Threads

The configuration parameters "KAFKA*CONSUMER_THREADS**" are utilized to specify the number of threads assigned to Kafka consumers for processing messages from topics. These parameters allow you to fine-tune the concurrency and parallelism of your Kafka consumer application, enabling efficient and scalable message consumption from Kafka topics.

Configuration Parameter	De va
`KAFKA_CONSUMER_THREADS_PROCESS_ENGINE_COMMANDS_MESSAGE`	10
`KAFKA_CONSUMER_THREADS_PROCESS_ENGINE_COMMANDS_DISCONNECT`	5

Configuration Parameter	De va
`KAFKA_CONSUMER_THREADS_PROCESS_ENGINE_COMMANDS_CONNECT`	5
`KAFKA_CONSUMER_THREADS_TASK_COMMANDS`	10
`KAFKA_AUTH_EXCEPTION_RETRY_INTERVAL`	10

Kafka topics related to process instances

Configuration Parameter	
`KAFKA_TOPIC_EVENTS_GATEWAY_PROCESS_INSTANCE_IN_MESSAGE`	`ai`
`KAFKA_TOPIC_EVENTS_GATEWAY_PROCESS_INSTANCE_IN_DISCONNECT`	`ai`
`KAFKA_TOPIC_EVENTS_GATEWAY_PROCESS_INSTANCE_IN_CONNECT`	`ai`

Kafka topics related to tasks

Configuration Parameter	
`KAFKA_TOPIC_EVENTS_GATEWAY_TASK_IN_MESSAGE`	`ai.flowx.eventsga`

Configuring authorization & access roles

Set the following environment variables to connect to the identity management platform:

Configuration Parameter	Description
`SECURITY_OAUTH2_BASE_SERVER_URL`	Base URL of the OAuth2 server
`SECURITY_OAUTH2_CLIENT_CLIENT_ID`	Client ID for OAuth2 authentication
`SECURITY_OAUTH2_CLIENT_CLIENT_SECRET`	Client secret for OAuth2 authentication
`SECURITY_OAUTH2_REALM`	Realm for OAuth2 authentication

Redis

The process engine sends the messages to the events-gateway, which is responsible for sending them to Redis.

Configuration Parameter	Description
`SPRING_REDIS_HOST`	Hostname of the Redis server
`SPRING_REDIS_PASSWORD`	Password for Redis server
`SPRING_REDIS_TTL`	Time-to-live for Redis keys (default value:5000000 # milliseconds)

Master replica

The events-gateway can be configured to communicate with Redis using the MASTER_REPLICA replication mode by configuring the following property:

spring.redis.sentinel.nodes: replica1, replica2, replica3, etc...

Example

```
spring.redis.sentinel.nodes=host1:26379,host2:26379,host3:2637
```

In the above example, the Spring Boot application will connect to three Redis Sentinel nodes: host1:26379, host2:26379, and host3:26379.

The property value should be a comma-separated list of host:port pairs, where each pair represents the hostname or IP address and the port number of a Redis Sentinel node.

INFO

> By default, Redis is standalone, so the configuration with `redis-replicas` is optional for high load use cases.

In the context of Spring Boot and Redis Sentinel integration, the `spring.redis.sentinel.nodes` property is used to specify the list of Redis Sentinel nodes that the Spring application should connect to. These nodes are responsible for monitoring and managing Redis instances.

Configuring logging

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

Configuration Parameter	Description
`LOGGING_LEVEL_ROOT`	Logging level for the root Spring Boot microservice logs
`LOGGING_LEVEL_APP`	Logging level for the application-level logs

Was this page helpful?