Solace PubSub+ Connector for GCP Pub/Sub

Solace Corporation

Version 1.1.0

solace.

Table of Contents

Preface

Solace PubSub+ Connector for GCP Pub/Sub bridges data between the Solace PubSub+ Event Broker and GCP Pub/Sub providing a flexible and efficient way to integrate GCP Pub/Sub application data with your Solace-backed, event-driven architecture and the Event Mesh. The connector is deployable standalone or in redundancy modes of “active-standby” or “active-active” to allow for high-availability and horizontal scaling of your data movement. Each connector instance supports up to 20 individual workflows (source-to-target pipeline), minimizing the number of connector instances deployed and managed. The use of various Spring Framework technologies allows for easy configuration of the connector, advanced logging capabilities, and export of live metrics data to external monitoring solutions.

Getting Started

Release Notes

Prerequisites

- Docker or Podman
- PubSub+ Event Broker
- GCP Pub/Sub

Usage

1. Create a directory called `config`
2. Create an `application.yml` or `application.properties` file in the `config` directory containing the properties needed for your connector.
   - For an example of such a config file, see the one packaged in the connector zip at https://solace.com/connectors/pubsubplus-connector-gcppubsub
3. The Application Default Credentials (ADC) by using the gcloud cli has been setup and `application_default_credentials.json` is in default location.
4. Set the `spring.cloud.gcp.credentials.location` property in `application.yml` or `application.properties` to `/app/external/gcp/config/application_default_credentials.json`
5. Run the container with minimal configuration:

Docker

```
docker run -d --name my-connector \
  -v `pwd`/libs/:/app/external/libs/:ro \
  -v `pwd`/config/:/app/external/spring/config/:ro \
  -v
$HOME/.config/gcloud/application_default_credentials.json:/app/external/gcp/config/
application_default_credentials.json:ro \
  solace/solace-pubsub-connector-gcppubsub:1.1.0
```

Podman

```
podman run -d --name my-connector \
  -v `pwd`/libs/:/app/external/libs/:ro \
  -v `pwd`/config/:/app/external/spring/config/:ro \
  -v
$HOME/.config/gcloud/application_default_credentials.json:/app/external/gcp/config/
application_default_credentials.json:ro \
  solace/solace-pubsub-connector-gcppubsub:1.1.0
```

On Windows, the default location of the `application_default_credentials.json` is

`%APPDATA%\gcloud\application_default_credentials.json`, so change the volume mount accordingly.

Assuming the Google Cloud `application_default_credentials.json` is in the default location, don't forget to add a volume mount to all docker/podman commands in reset of this document depending on your environment.

On Linux/MacOS:

```
-v
$HOME/.config/gcloud/application_default_credentials.json:/app/external/gcp/con
fig/application_default_credentials.json:ro \
```

On Windows:

```
-v
%APPDATA%\gcloud\application_default_credentials.json:/app/external/gcp/config/
application_default_credentials.json:ro \
```

Connecting to Services on the Host

If services (e.g. PubSub+ event broker) are exposed on the localhost, they can be referred to using the container platform's special DNS name which resolves to the internal IP address used by the host.

For example:

Docker

```
docker run -d --name my-connector \
  -v `pwd`/libs/:/app/external/libs/:ro \
  -v `pwd`/config/:/app/external/spring/config/:ro \
  --env SOLACE_JAVA_HOST=host.docker.internal:55555 \
  solace/solace-pubsub-connector-gcppubsub:1.1.0
```

Podman

```
podman run -d --name my-connector \
  -v `pwd`/libs/:/app/external/libs/:ro \
  -v `pwd`/config/:/app/external/spring/config/:ro \
  --env SOLACE_JAVA_HOST=host.containers.internal:55555 \
  solace/solace-pubsub-connector-gcppubsub:1.1.0
```

Configuring a Healthcheck

Here's a basic example command of how to configure the healthcheck for container:

Docker

```
docker run -d --name my-connector \
  -v `pwd`/libs/:/app/external/libs/:ro \
  -v `pwd`/application.yml:/app/external/spring/config/application.yml:ro \
  --env SOLACE_CONNECTOR_SECURITY_USERS_0_NAME=healthcheck \
  --env SOLACE_CONNECTOR_SECURITY_USERS_0_PASSWORD=healthcheck \
  --healthcheck-command="curl -X GET -u healthcheck:healthcheck --fail
localhost:8090/actuator/health" \
  solace/solace-pubsub-connector-gcppubsub:1.1.0
```

Podman

```
podman run -d --name my-connector \
  -v `pwd`/libs/:/app/external/libs/:ro \
  -v `pwd`/application.yml:/app/external/spring/config/application.yml:ro \
  --env SOLACE_CONNECTOR_SECURITY_USERS_0_NAME=healthcheck \
  --env SOLACE_CONNECTOR_SECURITY_USERS_0_PASSWORD=healthcheck \
  --healthcheck-command="curl -X GET -u healthcheck:healthcheck --fail
localhost:8090/actuator/health" \
  solace/solace-pubsub-connector-gcppubsub:1.1.0
```

This command does a few things:

- Creates a regular read-only user called `healthcheck`.
- Uses the `healthcheck` user as the user to poll the management health endpoint in the container's `healthcheck-command` and fails it if the connector is unhealthy.

Providing Configuration

There are two ways to provide Spring configuration properties to this container:

1. Using environment variables.
2. Using volume(s) containing Spring configuration files.

Ports

Port	Usage
8090	The connector's management endpoint.

Volumes

These are the supported directories for which volumes and bind mounts can be created.

Contents	Container Path	Optional	Recommended Permission
Spring configuration files	`/app/external/spring/config/`	Required unless all properties are defined using environment variables	Read-Only
Libraries	`/app/external/libs/`	Required	Read-Only
Classpath files	`/app/external/classpath/`	Optional	Read-Only
Output files	`/app/external/output/`	Optional	Read/Write

Volume: Spring Configuration Files

To add Spring configuration files (e.g. `application.yml`, etc), add a read-only volume or bind mount to `/app/external/spring/config/`.

This directory follows the same semantics as Spring's default `config/` directory. That is to say, this connector will automatically find and load Spring configuration files from the following locations when the connector starts:

1. The root of `/app/external/spring/config/`
2. Immediate child directories of `/app/external/spring/config/`

If config files for multiple, different, connectors will exist within the same config folder for use in different environments (e.g. dev, prod, etc), then consider using Spring Boot Profiles instead of child directories to do this.

i.e.:

- Do this:
  - `/app/external/spring/config/application-prod.yml`
  - `/app/external/spring/config/application-dev.yml`
- Instead of this:
  - `/app/external/spring/config/prod/application.yml`
  - `/app/external/spring/config/dev/application.yml`

Child directories are intended to be used for merging configuration from multiple sources of config properties. For more information and an example of when you might want to use multiple child directories to compose your application's configuration, please see the Spring Boot documentation.

Volume: Libraries

To add additional libraries, add a read-only volume or bind mount to `/app/external/libs/`.

This directory is provided as the location for the Java library dependencies (external `jar` files) that are required only when using certain features of the Connector (such as Prometheus libraries when using the metrics export to Prometheus feature in your Connector configuration).

See the documentation provided in the `libs` directory of the connector zip for more info.

Volume: Classpath Files

To add arbitrary files (i.e. not jar libraries and not Spring Boot configuration files), add a read-only volume or bind mount to `/app/external/classpath/`.

This directory should not contain jar libraries nor Spring Boot configuration files. Otherwise, there is a risk of libraries not getting picked up by the connector and/or overwriting the connector's internal configuration.

Volume: Output Files

Some features may support writing files (e.g. logging to a file). To capture these, add a read/write volume or bind mount to `/app/external/output/`.

When using features which generates files, they must be configured so that files are generated to this directory. Generating files anywhere else is not supported.

Configuring the JVM

To configure the JVM, set the `JDK_JAVA_OPTIONS` environment variable on the container.

See the JDK documentation for more info.

This container has been tested using:

- 2 active processors → `-XX:ActiveProcessorCount=2`
- 2GB of max heap memory → `-Xmx2048m`

Support

Support is offered best effort via our Solace Developer Community.

Premium support options are available, please Contact Solace.

License

This project is licensed under the Solace Community License, Version 1.0. - See the `LICENSE` file under the container's `/licenses` for details.