pubsubplus-connector-file: Java dependencies in the `/libs` folder

Table of Contents

Preface

This `/libs` directory is provided as a default location for the Java library dependencies (external `jar` files) that are either required or 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).

Solace does not provide the required JAR files due to licensing considerations. These JAR files are required as part of the deployment of the connector for it to operate correctly.

Micrometer Metrics exporter dependencies

The connector makes use of Micrometer.io to provide easy, configuration-driven exporting of connector metrics to many 3rd-party metrics monitoring solutions (see the full list on the Micrometer site).

These services can be configured in the connector configuration (there are commented in the sample configuration provided to guide you), but must have the necessary Java dependencies in the `/libs` directory to operate.

The JAR files that are necessary depend on which metrics service you are configuring. The details for each provider can be found in the Micrometer documentation. For example, for Prometheus, the Micrometer instructions list `io.micrometer.micrometer-registry-prometheus` as the main client library required and the Maven Central page for this library provides the download links for the main `jar` and its compile dependencies. These JAR files are placed in the connector's `/libs` directory.

Solace

Solace Google PubSub Spring Cloud stream connector enables reading of data from Google PubSub and publishing it to Solace. Binders for Google PubSub and Solace are used to consume/publish the data.

Free Solace Access

- Sign up for a free Solace Cloud service
- Download the free feature-complete Standard Edition software broker
- Quickstart Video for Solace PubSub+ Docker container

Setup Solace PubSub+ Broker

This broker can be a part of a Solace event mesh which has many interconnected Solace Brokers.

Access to a Solace messaging service, Solace PubSub+, can be achieved in either one of the three flavours

1. Hardware Appliance
2. Software broker image (Docker, Virtual image)
3. Solace Cloud service instance

This tutorial will walk you through setting up a Solace Cloud service instance, which also gives you access to the Event Portal. If you are interested in setting up a local broker running on Docker or a virtual machine check out the PubSub+ Event Broker: Software documentation

Sign up for free Solace Cloud account

Navigate to the Create a New Account page and fill out the required information. No credit card required!

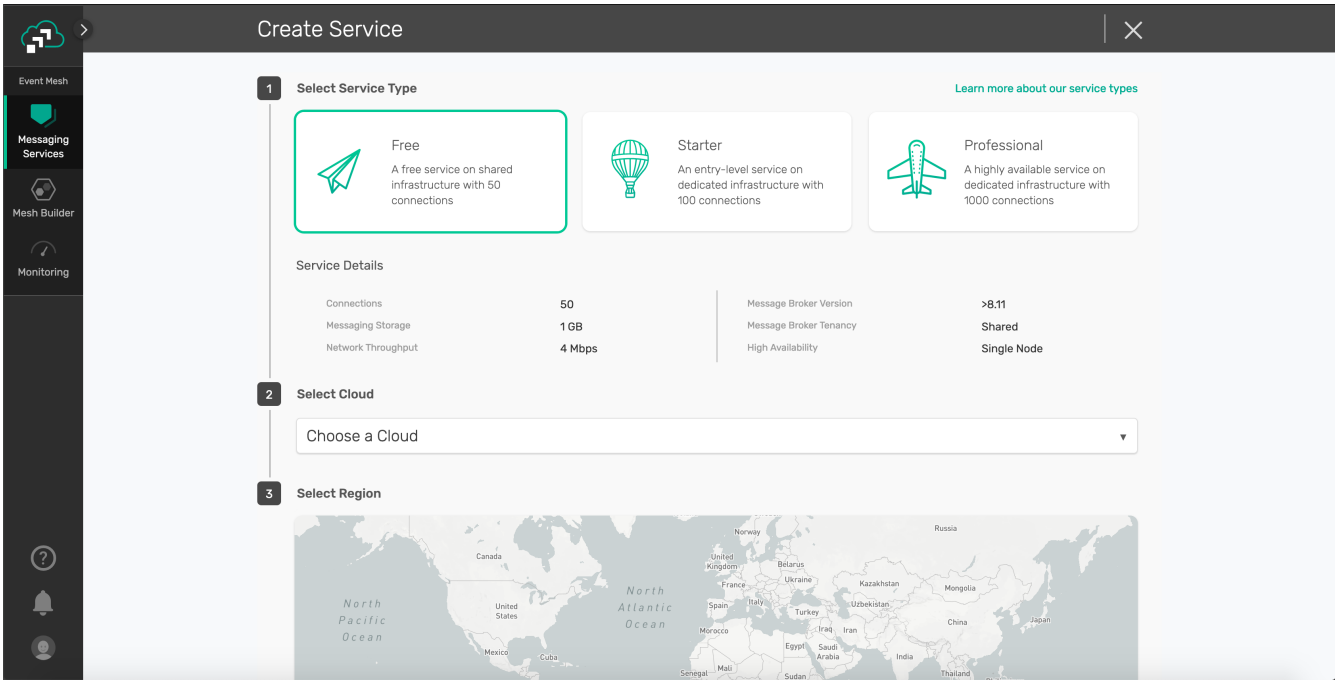
Create a messaging service

After you create your Solace Cloud account and sign in to the Solace Cloud Console, you'll be routed to the event mesh page.

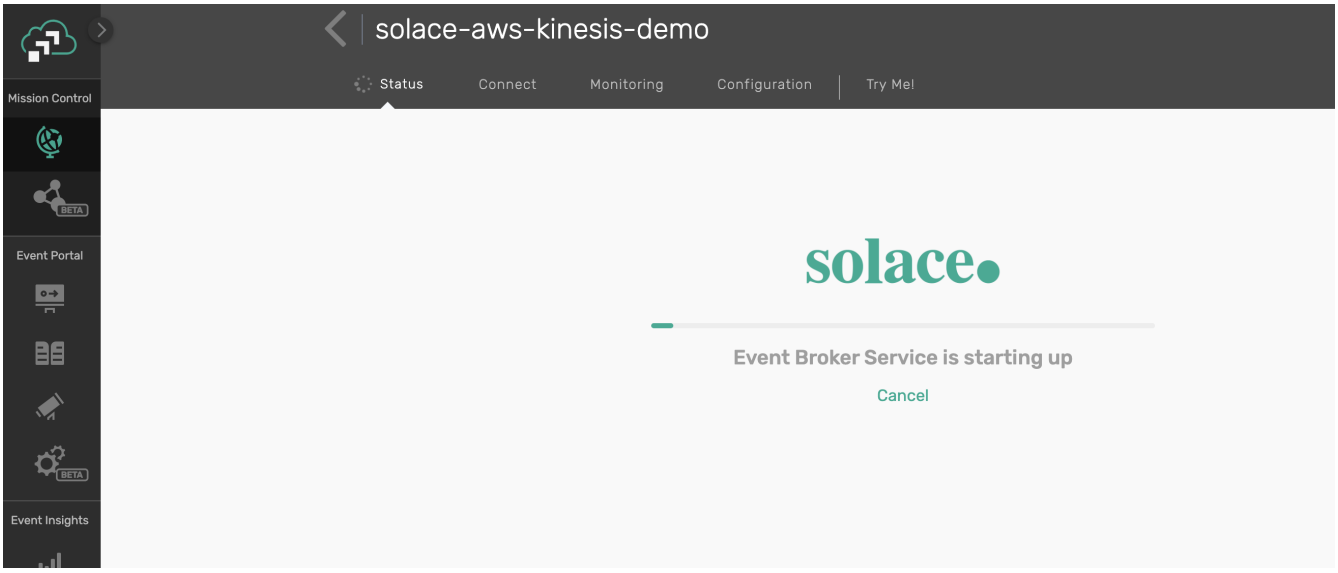
[solace setup1] | */Users/deepak/Documents/Official/Codes/connectors/Untitled/pubsubplus-connector-file/src/docs/asciidoc/libs/../../images/solace_setup1.webp*

Click on ‘Cluster Manager' and all the messaging services associated with your account will show up if you have any already created. To create a new service, click either button as depicted in the image below:

Fill out all the details for your messaging service, and then click "Create" at the bottom of the page.



Your service should be ready to use in a few minutes!



File Binder

File binder is based on Spring Cloud Stream Architecture. File binder has two components * Source - Reads file data from the source location, splits the data into multiparts and publishes onto Solace. * Sink - Reads the multiparts from Solace, assembles and writes them to destination location

Architecture

This connector is based on Spring Cloud Stream architecture and Solace Connector Framework. Solace Connector framework is built on Spring Java and it is the core component to manage to connector lifecycle. Solace Connector Framework provides the ability to configure upto 20 workflows(Each workflow represents data flow from a Source to Sink) and enables the connector to configure in High Availability mode. Apart from this the framework handles logging and exporting connector metrics to different sources. The connector uses File Binder to replicate file from source to sink

Features

File Types

The following file types are supported

1 - Static File

2 - Dynamic File(A file where data is written continuously)

3 - Directory(1st level)

4 - Directory(Recursive)

WildCards

Wildcard expressions can be configured to read/ignore files

Bandwidth Management

In case where bandwidth needs to be regulated the connector provides a configuration to configure max_bandwidth_limit

Checkpoint

The connector checkpoints by publishing last successful multipart/timestamp data to LVQ(Last Value queue) on Solace. In case of restart the connector reads the data from LVQ and resumes reading from last successful multipart/timestamp.

In case of directory the connector checkpoints the timestamp when snapshot of all the files are read. In next replication the connector considers the files that are changed/modified after the last snapshot.

Scheduler

Source Connector can be configured to run in fixed intervals using the scheduler configuration.

End of Day time check

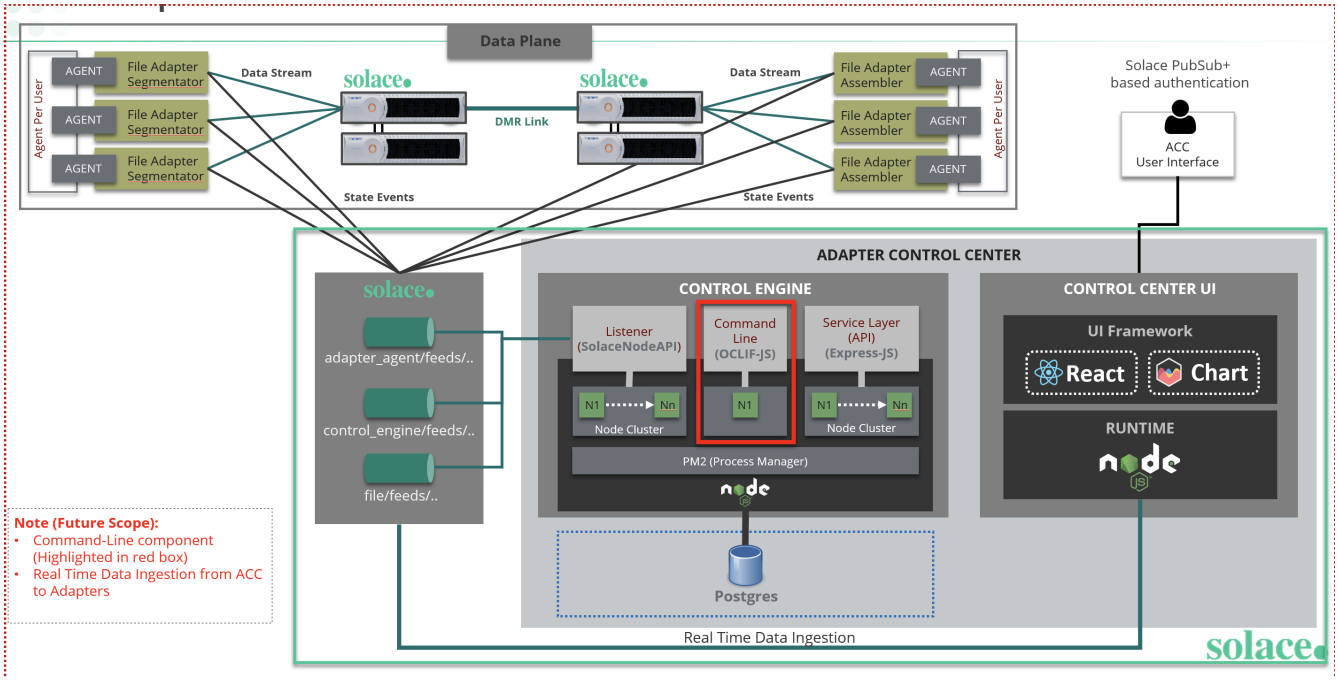
Both Source and Sink connector can be configured for End of Day time and they will exit once End of Day time is reached.

Configurations

Please refer to Configuration section in User Guide for Source and Sink connector configuration options.

Command Center

All the file replications can be monitored on Adapter Control Center dashboard. The connector publishes the health, replication state to Command Center which analyses and represents the data in graphical representation. Command Center is a seperate package and not packaged as part of connector. Please contact Solace if Command Center need to be setup for monitoring.



Prerequisites

Java, Maven

Running the connector

Please refer to User Guide on how to run the connector

Exploring the Code using IDE

Import the project into IntelliJ IDEA and all the maven commands are enabled automatically. Refer Build the connector section for next steps.

Authors

See the list of contributors who participated in this project.