Use a Remote Connector with Pulsar Proxy

Table of Contents

- Required infrastructure and software versions
- Install and configuration process overview
- Security configuration
- Configure Fusion
- Configure the remote (V2) connector
  - Description of Pulsar proxy properties
- Remote connector with Transport Security Layer (TLS) enabled
  - Obtain certificates to connect to pulsar proxy

To use a remote (V2)connector, configure both Fusion and the connector so on-premise customer content is transported to a Fusion instance running in the cloud. Remote connectors are controlled from Fusion.

Required infrastructure and software versions

- Google Cloud Platform
- Java 11
- Fusion 5.3 and higher

Install and configuration process overview

An overview of the process to install and use a remote connector is:

- Install Fusion
- Install the remote connector in the network so it can access the content source
- Configure and run the remote connector
- Access the Fusion Datasource tab and add the remote connector as a datasource
- Configure the connector
- Perform all datasource tasks as if the connector was installed directly into Fusion

Security configuration

The security configuration is used to expose IP addresses and ports in Google Cloud Platform (GCP) that are not typically exposed in the production standard setup. In addition, the customer server requires direct outbound TCP ports 80/443 and 6650/6651.

Configure Fusion

1. During the Fusion installation, add or modify the last four lines in this example of the `_fusion_values.yaml` file to enable the **pulsar-proxy** information:

```
pulsar:
  broker:
    annotations:
      prometheus.io/scrape: "true"
      prometheus.io/port: "8080"
  bookkeeper:
    annotations:
      prometheus.io/scrape: "true"
      prometheus.io/port: "8000"
  components:
    proxy: true
  image:
    repository: "lucidworks"
```

2. Run the `_upgrade_fusion` script:

```
./gke_lw-sales-us-west1_poc-remote-f5-poc_upgrade_fusion.sh
```

> Note
>
> For more information, see the fusion-cloud-native repository.

3. Execute the `kubectl get svc | grep pulsar-proxy` command to determine the load balancer IP and the required ports. The following example indicates the Fusion load balancer for the pulsar-proxy has an IP address of **34.105.102.171** and uses ports **80** and **6650**. The IP address and ports you generate may vary.

```
$ kubectl get svc | grep pulsar-proxy
NAME                               TYPE           CLUSTER-IP     EXTERNAL-IP
<namespace>-pulsar-pulsar-proxy    LoadBalancer   10.75.1.81     34.105.102.171
```

Configure the remote (V2) connector

1. Access the Lucidworks Plugin page .
2. Select and download the **connector-plugin-standalone.jar** file.
3. Create a `config.yaml` file and enter the following basic values:

- **pulsar.service-url** - The value is typically 6650, but is obtained running the `kubectl get svc | grep pulsar-proxy` command.
- **pulsar.admin-url** - The value is typically 80, but is obtained running the `kubectl get svc | grep pulsar-proxy` command.

> Note
>
> *See Description of Pulsar proxy properties for other values.*

The following is an example of the `config.yaml` file:

```
#
# The connector process will send messages through Pulsar.
# tenant-name is the namespace of the Fusion cluster
#
pulsar:
  service-url: pulsar://35.230.38.171:6650
  admin-url: https://35.230.38.171:80
  tenant-name: ppt
  authenticationEnabled: true
  # tlsEnabled: false
  # tlsTrustCertsFilePath: ca.crt


#
```

```
# The user name/password to Fusion and the URL where the
#
proxy:
  user: test
  password: test
  url: https://fusion.servername.here:6764/

#
# The name/location of the connector zip file
#
plugin:
  path: fs.zip
  type:
    suffix: remote
```

4. Execute the following command:

```
java -Xms256m -Xmx2048m -jar connector-plugin-standalone.jar config.yaml
```

The command output is:

```
For help use: 'java -jar connector-plugin-standalone.jar --help'

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::

[additional connectivity output]
```

Description of Pulsar proxy properties

Property	Description
`pulsar.service-url`	Pulsar Service URL. For example, when Transport Layer Security (TLS) is: • Disabled, the URL is pulsar://35.247.112.3:6650 • Enabled, the URL is pulsar+ssl://35.230.38.171:6651
`pulsar.admin-url`	Pulsar Admin URL. For example, when Transport Layer Security (TLS) is: • Disabled, the URL is http://35.247.112.3:8080 • Enabled, the URL is https://35.247.112.3:443
`tenant-name`	Pulsar Tenant Name (kube namespace)
`authenticationEnabled`	Pulsar authentication enabled flag
`tlsEnabled`	TLS enabled flag
`tlsTrustCertsFilePath`	Trust certs file path
`proxy.user`	Fusion proxy user
`proxy.password`	Fusion proxy password
`proxy.url`	Fusion proxy url
`plugin.path`	Path of plugin zip file

Property	Description
`plugin.type.suffix`	Plugin type suffix. For example, the: - `lucidworks.testplugin` ID with suffix `remote` is `lucidworks.testplugin.remote` - Name **Test Connector** is **Test Connector (remote)**

The following is an example of the YAML file for the JDBC V2 connector:

+

```
pulsar:
  service-url: pulsar://35.230.38.171:6650
  admin-url: http://35.230.38.171:80
  tenant-name: poc-f5-instance
  authenticationEnabled: true
  # tlsEnabled: false
  # tlsTrustCertsFilePath: ca.crt

proxy:
  user: test
  password: test
  url: http://35.197.110.199:6764/

plugin:
  path: lucidworks.connector.jdbc-1.0.0.zip
  type:
    suffix: remote
```

Remote connector with Transport Security Layer (TLS) enabled

To use remote connector with TLS enabled, deploy Fusion with TLS and pulsar proxy enabled and then obtain the certificates to connect to pulsar proxy.

Obtain certificates to connect to pulsar proxy

1. Execute the following command to Generate the certificate file from Kubernetes secrets used by the pulsar proxy component:

```
kubectl get secret <namespace>-pulsar-pulsar-proxy-0-tls -o yaml | grep ca.crt |


LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUZ2akNDQTZhZ0F3SUJBZ0lVRjR
2cG5paFIxYk5JV3FQWDdJb09pOXlyR3dBd0RRWUpLb1pJaHZjTkFRRU4KQlFBd1pURUxN
QWtHQTFVRUJoTUNWVk14RXpBUkJnTlZCQWdUQ2t0aGJHbG1iM0p1YVdFeEZqQVVCZ05WQk==
```

2. Copy the value generated and create a `ca.cart` file that includes the value and the following content:

```
-----BEGIN CERTIFICATE-----
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUZ2akNDQTZhZ0F3SUJBZ0lVRjR
2cG5paFIxYk5JV3FQWDdJb09pOXlyR3dBd0RRWUpLb1pJaHZjTkFRRU4KQlFBd1pURUxN
QWtHQTFVRUJoTUNWVk14RXpBUkJnTlZCQWdUQ2t0aGJHbG1iM0p1YVdFeEZqQVVCZ05WQk==
-----END CERTIFICATE-----
```

3. Execute the following YAML configuration to run the remote connector:

```
pulsar:
  service-url: pulsar+ssl://35.230.38.171:6651
  admin-url: http://35.230.38.171:443
  tenant-name: poc-f5-instance
  authenticationEnabled: true
  tlsEnabled: true
  tlsTrustCertsFilePath: ca.crt

proxy:
  user: test
  password: test
  url: http://35.197.110.199:6764/
```