

Deploying an EFM Cluster

Date published: 2019-04-15

Date modified: 2024-03-08



Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Deploying Edge Flow Manager cluster.....	4
EFM cluster database setup.....	4
EFM cluster configuration.....	4
EFM cluster high availability.....	5
Securing an EFM cluster.....	6

Deploying Edge Flow Manager cluster

Learn how to deploy an EFM cluster and the requirements to do so including supported number of nodes, supported databases, load balancer, and MiNiFi agents.

The Edge Flow Manager (EFM) server component of CEM supports clustered deployments for horizontal scalability and high availability. An EFM cluster deployment requires the following:

- Three or more EFM nodes
- A shared and external EFM database (MySQL, MariaDB, or PostgreSQL)
- An EFM load balancer for Web UI users

You can use any load balancer of your choice, provided it can proxy HTTP(s) traffic and can be configured as described in this document.

- MiNiFi agents that forward client requests to one of the EFM cluster nodes

In addition to communicating with the external, centralized, and shared EFM database, EFM cluster members also communicate with each other to establish ephemeral state, such as distributed locks and caches, within the cluster. Due to this, each EFM node must have network connectivity to every other EFM cluster member.

EFM cluster database setup

Before you first run or launch the Edge Flow Manager (EFM) cluster, you must learn how to install and configure MySQL, MariaDB, or PostgreSQL.

For more information, see *Installing Databases*.

You can use an existing external database of a single and standalone EFM instance for the cluster deployment. None of the existing data will be lost.

However, you cannot use an H2 database of a standalone EFM instance and you must start over with an external database. When moving from H2 to MySQL, MariaDB, or PostgreSQL, prior to stopping the standalone instance, export the existing flows from the EFM Flow Designer and import into the cluster to avoid data loss. For more information on exporting and importing flows, see the [REST API Reference](#).

Related Information

[Installing databases for EFM](#)

EFM cluster configuration

Learn how to configure every Edge Flow Manager (EFM) cluster member to make it part of the same cluster.

You can find the cluster configuration in the `efm.properties` file located in the `conf` directory of the EFM installation. The following is an example cluster configuration:

```
# Web Server Properties
# address: the hostname or ip address of the interface to bind to; to bind
# to all, use 0.0.0.0
efm.server.address=10.0.0.1
efm.server.port=10090
efm.server.servlet.contextPath=/efm
efm.cluster.enabled=true
# Database Properties
efm.db.url=jdbc:mysql://database.example.com:3306/efm
efm.db.driverClass=com.mysql.cj.jdbc.Driver
efm.db.username=efm
```

```
efm.db.password=efmPassword
efm.db.maxConnections=50
efm.db.sqlDebug=false
```



Note: Except for the node-specific configurations, such as `efm.server.address` and `efm.cluster.address`, which can be a hostname or IP address, all other configurations must match for all nodes in the cluster. It is especially important that the nodes use the same database URL (`efm.db.url`), and the same list of cluster members (`efm.infinispan.jgroups[tcping.initial_hosts]`). If these values do not match on all cluster nodes, the cluster fails on startup.

Cluster node discovery

Cluster node discovery allows you to configure multiple node discovery strategies to establish an EFM cluster, tailored to different deployment environments.

You have to define the `efm.infinispan.environment` property with the name of the environment where EFM is deployed. The default value is `tcp`, and the supported values are `tcp`, `tcp-fixed`, and `kubernetes`.

Here are the specifics for each environment setting:

tcp

TCP discovery uses IP multicast to discover the initial membership, eliminating the need for manual configuration on each node. This is the default setting.

Property required: `efm.infinispan.environment=tcp`

tcp-fixed

You have to provide a static list of host:port combinations for all nodes.

Properties required:

- `efm.infinispan.environment=tcp-fixed`
- `efm.infinispan.jgroups[tcping.initial_hosts]`

Example value: `localhost[7801],localhost[7802],localhost[7803]`



Note: Port numbers have to match the `efm.infinispan.jgroups[bind.port.jgroups.tcp.port]` startup settings where Infinispan is running. The default port is 7800.

kubernetes

This strategy is designed to automatically discover pods based on namespace and label filtering.

Properties required:

- `efm.infinispan.environment=kubernetes`
- `efm.infinispan.jgroups[kubeping.namespace]`

Example value: `cem-namespace`

- `efm.infinispan.jgroups[kubeping.labels]`

Example value: `app=efm`



Note:

Setting up labels and namespaces in Kubernetes is the responsibility of the user.

EFM cluster high availability

Learn why you are able to operate in an Edge Flow Manager (EFM) cluster with high availability.

In a clustered mode, EFM is highly available.

Even if individual nodes are stopped the cluster can continue operation. Nodes can stop or fail due to a variety of reasons, such as routine maintenance, loss of network connectivity, or underlying physical server host failure. By

default, an EFM cluster can continue operating with up to two nodes down. If the failed nodes are restored, the cluster reforms. Note that any load that was served by a failed node is shifted to other cluster nodes. Hence capacity may fall if all nodes in the cluster were optimally utilized with regards to resources including network, CPU, memory, and disk.

Securing an EFM cluster

Learn how to secure intra-cluster communication between Edge Flow Manager (EFM) instances in a clustered mode.

By default, intra-cluster communication is through unsecure TCP. It is assumed that all EFM instances must be running in a Virtual Private Cloud and the cluster communication port must not be open to public networks.



Note: The default cluster communication port is 7800, and it is configurable as a part of the `efm.infinispan.jgroups[bind.port.jgroups.tcp.port]` property value.

Configuring secure communications between EFM nodes

To enable secure communications you need to configure a few properties. Follow the steps below to make the necessary configuration:

1. Set `efm.infinispan.ssl.enable` to 'true' to activate SSL for Infinispan.
2. Define the path to the keystore for Infinispan by setting `efm.infinispan.ssl.key-store`.
3. Set the password for the keystore by configuring `efm.infinispan.ssl.key-store-password`.

After configuring these properties, you should see similar log entries like the one below in your application logs:

```
2023-08-22 16:44:12.208 INFO org.infinispan.CLUSTER : ISPN000078: Starting JGroups channel `efm` with stack `tcp-ssl`
```

Generating a Java Keystore

You can use your existing Java Keystore (JKS) for a secure setup, or you can create a new one. Follow the steps below to create a new Java Keystore (JKS) with a signed certificate for secure communications with Infinispan.

1. Generate a private key and a self-signed certificate (root certificate).

```
openssl genrsa -out root.key
openssl req -new -x509 -key root.key -out root.crt
chmod 600 root.key
chmod 644 root.crt
```

2. Create a keystore (`infini.keystore.jks`) and generate a key pair with a subject alternative name (SAN).

```
keytool -keystore infini.keystore.jks -alias localhost -validity 365 -genkey -keyalg RSA -ext SAN=DNS:localhost -storetype PKCS12
```

3. Generate a certificate signing request (CSR) for the key pair.

```
keytool -keystore infini.keystore.jks -alias localhost -certreq -file infini.unsigned.crt
```

4. Sign the certificate with the root certificate to create the signed certificate.

```
openssl x509 -req -CA root.crt -CAkey root.key -in infini.unsigned.crt -out infini.signed.crt -days 365 -CAcreateserial
```

5. Import the root certificate into the keystore.

```
keytool -keystore infini.keystore.jks -alias CARoot -import -file root.crt
```

6. Import the signed certificate into the keystore.

```
keytool -keystore infini.keystore.jks -alias localhost -import -file inf  
ini.signed.crt
```