Installing with Docker Compose

Overview

Mission Control can be installed using Docker Compose allowing you to orchestrate your setup using Docker Compose. This will ensure you have all the required services specified in a single JSON file with pre-configured parameters.

For more details on Docker Compose, please refer to the Docker documentation.

Mission Control can be installed using the jfmc-compose-<version> tool.

> Available from Mission Control version 3.3

Page Contents

Zip Contents

The *jfmc-compose-<version>. zip* contains the following files:

- ***setenv.sh**:* An environment file that contains environment variables that will be resolved in the Mission Control Compose file. It contains core application properties, Postgres details and Elasticsearch details, and can be modified as per your requirements. This needs to be loaded in the target environment before running *docker-compose* actions.
- ***jfmc-compose.json**:* A compose file.
- ***jfmc-compose-ha.json**:* A compose file with no database services. This can be used for HA installation or an installation which needs to connect to external PosgreSQL and Elasticsearch.

Standard Installation

The JFrog Mission Control Docker Compose installer can be downloaded from the Mission Control Download Page.

> The docker-compose actions refer to the project name as "jfmc" in the *docker-compose -p <project name> <action>* command.

1. Download and extract the *jfmc-compose-<version>.zip*.

   ```
   unzip jfmc-compose-<version>.zip
   ```

2. Set the *JFMC_MOUNT_ROOT* variable using the mount path in the *setenv.sh file.* This will be used to store data, config and logs of all the JFMC services, including the databases Mission Control uses.
   Load the environment variables to the session which will run the *docker-compose* actions.

   ```
   source ./setenv.sh
   ```

   > **Reload required**
   >
   > You need to reload the setenv.sh file and restart services every time a value of an environment is modified.
   >
   > ```
   > source ./setenv.sh
   > docker-compose  -f ./jfmc-compose.json -p jfmc down
   > docker-compose  -f ./jfmc-compose.json -p jfmc up -d
   > ```

3. Mission Control services running as non-root user with UID and GID as 1050 by default. The mount for each service should be owned by the user running within it.
   Each service is enabled to run with a custom UID and GID. This can be done by setting a new key value pair under each service as follows:

   ```
   "user": "<uid>:<gid>"
   ```

> Make sure the mount point for each service is owned by this set UID and GID.

For the default UID and GID, execute following steps to **prepare mounted directories**:

```
# For Mission Control services default is 1050:1050,
mkdir -p ${JFMC_MOUNT_ROOT}/jfmc/logs/insight-server
mkdir -p ${JFMC_MOUNT_ROOT}/jfmc/logs/insight-scheduler
mkdir -p ${JFMC_MOUNT_ROOT}/jfmc/logs/insight-executor

mkdir -p ${JFMC_MOUNT_ROOT}/jfmc/etc/insight-scheduler
mkdir -p ${JFMC_MOUNT_ROOT}/jfmc/etc/insight-executor

mkdir -p ${JFMC_MOUNT_ROOT}/jfmc/support/insight-scheduler
mkdir -p ${JFMC_MOUNT_ROOT}/jfmc/support/insight-executor
mkdir -p ${JFMC_MOUNT_ROOT}/jfmc/support/insight-server

chown -R 1050:1050 ${JFMC_MOUNT_ROOT}/jfmc

# For Elasticsearch default is 1000:1000,
mkdir -p ${JFMC_MOUNT_ROOT}/elasticsearch/data
mkdir -p ${JFMC_MOUNT_ROOT}/elasticsearch/sgconfig
chown -R 1000:1000 ${JFMC_MOUNT_ROOT}/elasticsearch

# For PostgreSQL,
# Ignore this part if you are running with default compose file
# PostgreSQL runs as root user by default, docker will take care of creating mount point with right
permissions
# For custom UID and GID,
# mkdir -p ${JFMC_MOUNT_ROOT}/postgres/data
# chown -R customUID:customGID ${JFMC_MOUNT_ROOT}/postgres

# For MongoDB (removed in 3.4.0),
# Ignore this part if you are running with default compose file
# Mongo DB runs as root user by default, docker will take care of creating mount point with right
permissions
# For custom UID and GID,
# mkdir -p ${JFMC_MOUNT_ROOT}/mongodb/db
# chown -R customUID:customGID ${JFMC_MOUNT_ROOT}/mongodb
```

4. Create a PostgreSQL database and users.
    a. Launch the "postgres *container*", and login as admin user.

    ```
    #Start the postgres container
    docker-compose  -f ./jfmc-compose.json -p jfmc up -d postgres

    #Check if it started successfully and get container-id
    docker ps -a

    #Exec into the container
    docker exec -it <Container_id> /bin/bash

    #Login as admin user and login to psql with admin credentials
    psql -U postgres
    ```

    b. Create the postgres database, schema and users as in the Configuring PostgreSQL section.
5. Launch Mission Control.

```
docker-compose  -f ./jfmc-compose.json -p jfmc up -d
```

6. Initialise Elasticsearch guard plugin.

```
docker exec -it jfmc_elasticsearch_1 bash -c "cd /usr/share/elasticsearch/plugins/search-guard-6
/tools; ./sgadmin.sh -p ${ELASTIC_TRANSPORT_PORT} -cacert root-ca.pem -cert sgadmin.pem -key sgadmin.
key -nhnv -icl -cd ../sgconfig/"
```

HA Installation

Setting up the First Node

1. Download and extract the *jfmc-compose-<version>.zip*.

   ```
   unzip jfmc-compose-<version>.zip
   ```

2. Set the *JFMC_MOUNT_ROOT* variable using the mount path in the *setenv.sh file*. This will be used to store data, config and logs of all the Mission Control services, including the databases Mission Control uses.
   Load the environment variables to the session which will run the *docker-compose* actions.

   ```
   source ./setenv.sh
   ```

   > **Reload required**
   >
   > You need to reload the setenv.sh file and restart services every time a value of an environment is modified.
   >
   > ```
   > source ./setenv.sh
   > docker-compose  -f ./jfmc-compose-ha.json -p jfmc down
   > docker-compose  -f ./jfmc-compose-ha.json -p jfmc up -d
   > ```

3. Mission Control services are running as non-root user with UID and GID as 1050 by default. The mount for each service should be owned by the user running within it.
   Each service is enabled to run with a custom UID and GID. This can be done by setting a new key value pair under each service as follows:

   ```
   "user": "<uid>:<gid>"
   ```

   > Make sure the mount point for each service is owned by this set UID and GID.

   For the default UID and GID, execute following steps to prepare mounted directories.

   ```
   # For Mission Control services default is 1050:1050,
   mkdir -p ${JFMC_MOUNT_ROOT}/jfmc/logs/insight-server
   mkdir -p ${JFMC_MOUNT_ROOT}/jfmc/logs/insight-scheduler
   mkdir -p ${JFMC_MOUNT_ROOT}/jfmc/logs/insight-executor
   chown -R 1050:1050 ${JFMC_MOUNT_ROOT}/jfmc

   # For Elasticsearch default is 1000:1000,
   mkdir -p ${JFMC_MOUNT_ROOT}/elasticsearch/data
   mkdir -p ${JFMC_MOUNT_ROOT}/elasticsearch/sgconfig
   chown -R 1000:1000 ${JFMC_MOUNT_ROOT}/elasticsearch

   # Create elasticsearch unicast file
   # This file will be modified by insight-server and read by elasticsearch
   # Note : Will be utilized only in Mission Control HA mode
   mkdir -p ${JFMC_MOUNT_ROOT}/elasticsearch/config
   echo "" > ${JFMC_MOUNT_ROOT}/elasticsearch/config/unicast_hosts.txt
   chown -R 1000:1000 ${JFMC_MOUNT_ROOT}/elasticsearch
   ```

4. Create a PostgreSQL database, schema and users by following steps from Using External Databases.

5. Modify PostgreSQL connection details to point to newly setup external PostgresQL in setenv.sh
6. Modify HA related environment variables.

```
export JFMC_ES_CLUSTER_SETUP=“YES”
# Host IP will be used by other HA nodes to connect and join the elastic cluster through transport
port
export JFMC_HOST_IP=<host_ip>
export ELASTIC_TRANSPORT_PORT=9300
# Set this to true for HA installation - mission-control service will commit suicide if insight-
server is unhealthy which in turn will indicate the node is unhealthy
export NODEHEALTHCHECK_KILL_ONMAXFAILURES=true
```

⚠ Make sure <JFMC_HOST_IP>:<ELASTIC_TRANSPORT_PORT> is accessible from other nodes

7. Load the environment variables.

```
source ./setenv.sh
```

8. Launch Mission Control.

```
docker-compose  -f ./jfmc-compose-ha.json -p jfmc up -d
```

9. Initialise elastic search guard plugin.

```
docker exec -it jfmc_elasticsearch_1 bash -c "cd /usr/share/elasticsearch/plugins/search-guard-6
/tools; ./sgadmin.sh -p ${ELASTIC_TRANSPORT_PORT} -cacert root-ca.pem -cert sgadmin.pem -key sgadmin.
key -nhnv -icl -cd ../sgconfig/"
```

Setting up the Second/Additional Nodes

1. Complete the first 3 steps from the First node instructions above.
2. Modify PostgreSQL connection details to point to existing external PostgresQL in setenv.sh
3. Modify HA related environment variables,

```
export JFMC_ES_CLUSTER_SETUP="YES"
# Host IP will be used by other HA nodes to connect and join the elastic cluster through transport
port
export JFMC_HOST_IP=<host_ip>
export ELASTIC_TRANSPORT_PORT=9300
# Set this to true for HA installation - mission-control service will commit suicide if insight-
server is unhealthy which in turn will indicate the node is unhealthy
export NODEHEALTHCHECK_KILL_ONMAXFAILURES=true
export ELASTIC_MIN_MASTER_NODES=2
```

⚠ Make sure <JFMC_HOST_IP>:<ELASTIC_TRANSPORT_PORT> is accessible from other nodes

4. Copy mc.key content from first node, can be found in ${JFMC_MOUNT_ROOT}/jfmc/data/security/mc.key
5. Paste the copied mc.key in ${JFMC_MOUNT_ROOT}/jfmc/data/security/mc.key of current node.

⚠ Make sure ${JFMC_MOUNT_ROOT}/jfmc/data/security/mc.key is set to right ownership (default 1050:1050).

6. Load the environment variables.

```
source ./setenv.sh
```

7. Launch Mission Control.

```
docker-compose  -f ./jfmc-compose-ha.json -p jfmc up -d
```