Configuring Elasticsearch

Overview

Elasticsearch is a highly scalable search and analytics engine. It is used to store and retrieve historical data for Artifactory services and their repositories, and provide it to Mission Control to display in the Service Trends feature.

Elasticsearch Version

JFrog Mission Control currently uses **ElasticSearch version 6.6.0**.

Elasticsearch Resources

- Installation Guide
- Getting Started
- Snapshot and Restore

Page Contents

How Elasticsearch is Packaged within Mission Control

Elasticsearch is packaged into the Mission Control installation. The following describes the different variations for each distribution package type:

Distribution	Packaging
Docker	Elasticsearch for Docker is added as a Docker container to the Mission Control docker-compose project. This installs version 6.1.1.
Debian	Elasticsearch 6.1.1 is included in the Mission Control Debian project. Linux service files are added.
RPM	Elasticsearch 6.1.1 is included in the Mission Control RPM project. Linux service files are added.

Installation Structure

After Mission Control is fully installed, the Elasticsearch data files can be found in the following locations:

File	Location
Binaries	<JMFC_HOME>/elasticsearch/
Data files	Linux: <JMFC_HOME>/elasticsearch/ Docker: <JMFC_HOME>/elasticsearch/

Ports

Elasticsearch uses the following communication ports:

Service	Port
HTTP API	9200
Java Client	9300

Indexes and Aliases

There are two aliases to store and retrieve data:

Alias Name	Description
active_insight_data	Used point to active index to push data.
search_insight_data	Used to search and retrieve data

On installation, these aliases and indices of format active_insight_data_timestamp* are created.

Index Cleanup

Mission Control is pre-configured to periodically cleanup indexes to keep data for a period of one year.

Setting Up a Cluster

As the amount of historical data collected by Mission Control accumulates, you may want to scale the Elasticsearch database used to store that data to maintain performance and responsiveness.

To scale a running Elasticsearch instance to a two-node cluster, follow the steps below:

1. Add the following settings to your startup scripts. In the case of a Docker installation, add them to the `docker-compose` file, for a non-Docker installation, add them to the *`jfmc.sh`* script:

```
discovery.zen.minimum_master_nodes=2 #(The recommendation is for (N/2+1), where N is the number of
eligible master nodes).
node.master=true
node.data=true
discovery.zen.ping.unicast.hosts=<Published IP address of each node>
network.publish_host=<IP address to be published for the node>
node.name=<Node name>
```

2. Restart Elasticsearch

> **Make sure to restart your first node before starting up the second one**
>
> First time you run Elasticsearch as a two-node cluster, you need to make sure to restart your first node (the one that already contains data) before adding the second one. Starting up the new node first could cause index mapping templates to be deleted.

3. Start up the second node using the same configuration file (that you just modified) used for your first Elasticsearch node.

Once your cluster is set up, Mission Control will automatically detect the new node at runtime and start sending it requests.

Using a Load Balancer

If your Elasticsearch cluster is behind a load balancer, you need to add the following environment variables to the `insight_server` service to provide it with the load balancer read and write URLs:

`ELASTIC_SEARCH_URL`

`ELASTIC_LB_WRITE_URL`

`ELASTIC_LB_READ_URL`