MobileFirst Analytics Server Configuration Guide

Overview

Some configuration for the MobileFirst Analytics Server is required. Some of the configuration parameters apply to a single node, and some apply to the whole cluster, as indicated.

Jump to

Properties

For a complete list of configuration properties and how to set them in your application server, see Configuration properties.

- The **discovery.zen.minimum_master_nodes** property must be set to **ceil((number of master-eligible nodes in the cluster** / **2) + 1)** to avoid split-brain syndrome.
    - Elasticsearch nodes in a cluster that are master-eligible must establish a quorum to decide which master-eligible node is the master.
    - If you add a master eligible node to the cluster, the number of master-eligible nodes changes, and thus the setting must change. You must modify the setting if you introduce new master-eligible nodes to the cluster. For more information about how to manage your cluster, see Cluster management and Elasticsearch.
- Give your cluster a name by setting the **clustername** property in all of your nodes.
    - Name the cluster to prevent a developer's instance of Elasticsearch from accidentally joining a cluster that is using a default name.
- Give each node a name by setting the **nodename** property in each node.
    - By default, Elasticsearch names each node after a random Marvel character, and the node name is different on every node restart.
- Explicitly declare the file system path to the data directory by setting the **datapath** property in each node.
- Explicitly declare the dedicated master nodes by setting the **masternodes** property in each node.

Cluster Recovery Settings

After you scaled out to a multi-node cluster, you might find that an occasional full cluster restart is necessary. When a full cluster restart is required, you must consider the recovery settings. If the cluster has 10 nodes, and as the cluster is brought up, one node at a time, the master node assumes that it needs to start balancing data immediately upon the arrival of each node into the cluster. If the master is allowed to behave this way, much unnecessary rebalancing is required. You must configure the cluster settings to wait for a minimum number of nodes to join the cluster before the master is allowed to start instructing the nodes to rebalance. It can reduce cluster restarts from hours down to minutes.

- The **gateway.recover_after_nodes** property must be set to your preference to prevent

Elasticsearch from starting a rebalance until the specified number of nodes in the cluster are up and joined. If your cluster has 10 nodes, a value of 8 for the **gateway.recover_after_nodes** property might be a reasonable setting.

- The **gateway.expected_nodes** property must be set to the number of nodes that you expect to be in the cluster. In this example, the value for the **gateway.expected_nodes** property is 10.
- The **gateway.recover_after_time** property must be set to instruct the master to wait to send rebalanced instructions until after the set time elapsed from the start of the master node.

The combination of the previous settings means that Elasticsearch waits for the value of **gateway.recover_after_nodes** nodes to be present. Then, it begins recovering after the value of **gateway.recover_after_time** minutes or after the value of **gateway.expected_nodes** nodes joined the cluster, whichever comes first.

What not to do

- Do not ignore your production cluster.
    - Clusters need monitoring and nurturing. Many good Elasticsearch monitoring tools are available that are dedicated to the task.
- Do not use network-attached storage (NAS) for your **datapath** setting. NAS introduces more latency, and a single point of failure. Always use the local hosts disks.
- Avoid clusters that span data centers and definitely avoid clusters that span large geographic distances. The latency between nodes is a severe performance bottleneck.
- Roll your own cluster configuration management solution. Many good configuration management solutions, such as Puppet, Chef, and Ansible, are available.

Configuration properties

The MobileFirst Analytics Server can start successfully without any additional configuration.

Configuration is done through JNDI properties on both the MobileFirst Server and the MobileFirst Analytics Server. Additionally, the MobileFirst Analytics Server supports the use of environment variables to control configuration. Environment variables take precedence over JNDI properties.

The Analytics runtime web application must be restarted for any changes in these properties to take effect. It is not necessary to restart the entire application server.

To set a JNDI property on WebSphere Application Server Liberty, add a tag to the **server.xml** file as follows.

```
<jndiEntry jndiName="{PROPERTY NAME}" value="{PROPERTY VALUE}}" />
```

To set a JNDI property on Tomcat, add a tag to the context.xml file as follows.

```
<Environment name="{PROPERTY NAME}" value="{PROPERTY VALUE}" type="java.lang.S
tring" override="false" />
```

The JNDI properties on WebSphere Application Server are available as environment variables.

- In the WebSphere Application Server console, select **Applications → Application Types → WebSphere Enterprise applications**.
- Select the **MobileFirst Administration Service** application.

- In **Web Module Properties**, click **Environment entries for Web Modules** to display the JNDI properties.

MobileFirst Server

The following table shows the properties that can be set in the MobileFirst Server.

Property	Description	Default Value
mfp.analytics.console.url	Set this property to the URL of your MobileFirst Analytics Console. For example, http://hostname:port/analytics/console. Setting this property enables the analytics icon on the MobileFirst Operations Console.	None
mfp.analytics.logs.forward	If this property it set to true, server logs that are recorded on the MobileFirst Server are captured in MobileFirst Analytics.	true
mfp.analytics.url	Required. The URL that is exposed by the MobileFirst Analytics Server that receives incoming analytics data. For example, http://hostname:port/analytics-service/rest/v2.	None
analyticsconsole/mfp.analytics.url	Optional. Full URI of the Analytics REST services. In a scenario with a firewall or a secured reverse proxy, this URI must be the external URI, not the internal URI inside the local LAN. This value can contain * in places of the URI protocol, host name, or port, to denote the corresponding part from the incoming URL. *://*:*/analytics-service, with the protocol, host name, and port dynamically determined	
mfp.analytics.username	The user name that is used if the data entry point is protected with basic authentication.	None
mfp.analytics.password	The password that is used if the data entry point is protected with basic authentication.	None

MobileFirst Analytics Server

The following table shows the properties that can be set in the MobileFirst Analytics Server.

Property	Description	Default Value
analytics/nodetype	Defines the Elasticsearch node type. Valid values are master and data. If this property is not set, then the node acts as both a master-eligible node and a data node.	None
analytics/shards	The number of shards per index. This value can be set only by the first node that is started in the cluster and cannot be changed.	1
analytics/replicas_per_shard	The number of replicas for each shard in the cluster. This value can be changed dynamically in a running cluster.	0
analytics/masternodes	A comma-delimited string that contains the host name and ports of the master-eligible nodes.	None
analytics/clustername	Name of the cluster. Set this value if you plan to have multiple clusters that operate in the same subset and need to uniquely identify them.	worklight
analytics/nodename	Name of a node in the cluster.	A randomly generated string
analytics/datapath	The path that analytics data is saved to on the file system.	./analyticsData

Property	Description	Default Value
analytics/settingspath	The path to an Elasticsearch settings file. For more information, see Elasticsearch.	None
analytics/transportport	The port that is used for node-to-node communication.	9600
analytics/httpport	The port that is used for HTTP communication to Elasticsearch.	9500
analytics/http.enabled	Enables or disables HTTP communication to Elasticsearch.	false
analytics/serviceProxyURL	The analytics UI WAR file and analytics service WAR file can be installed to separate application servers. If you choose to do so, you must understand that the JavaScript run time in the UI WAR file can be blocked by cross-site scripting prevention in the browser. To bypass this block, the UI WAR file includes Java proxy code so that the JavaScript run time retrieves REST API responses from the origin server. But the proxy is configured to forward REST API requests to the analytics service WAR file. Configure this property if you installed your WAR files to separate application servers.	None
analytics/bootstrap.mlockall	This property prevents any Elasticsearch memory from being swapped to disk.	true
analytics/multicast	Enables or disables multicast node discovery.	false
analytics/warmupFrequencyInSeconds	The frequency at which warmup queries are run. Warmup queries run in the background to force query results into memory, which improves web console performance. Negative values disable the warmup queries.	600
analytics/tenant	Name of the main Elasticsearch index.	worklight

In all cases where the key does not contain a period (like **httpport** but not **http.enabled**), the setting can be controlled by system environment variables where the variable name is prefixed with **ANALYTICS_**. When both the JNDI property and the system environment variable are set, the system environment variable takes precedence. For example, if you have both the **analytics/httpport** JNDI property and the **ANALTYICS_httpport** system environment variable set, the value for **ANALYTICS_httpport** is used.

Document Time to Live (TTL)

TTL is effectively how you can establish and maintain a data retention policy. Your decisions have dramatic consequences on your system resource needs. The long you keep data, the more RAM, disk, and scaling is likely needed.

Each document type has its own TTL. Setting a document's TTL enables automatic deletion of the document after it is stored for the specified amount of time.

Each TTL JNDI property is named **analytics/TTL_[document-type]**. For example, the TTL setting for **NetworkTransaction** is named **analytics/TTL_NetworkTransaction**.

These values can be set by using basic time units as follows.

- 1Y = 1 year
- 1M = 1 month
- 1w = 1 week
- 1d = 1 day
- 1h = 1 hour

- 1m = 1 minute
- 1s = 1 second
- 1ms = 1 millisecond

> Note: If you are migrating from previous versions of MobileFirst Analytics Server and previously configured any TTL JNDI properties, see Migration of server properties used by previous versions of MobileFirst Analytics Server (../installation/#migration-of-server-properties-used-by-previous-versions-of-mobilefirst-analytics-server).

Elasticsearch

The underlying storage and clustering technology that serves the MobileFirst Analytics Console is Elasticsearch.
Elasticsearch provides many tunable properties, mostly for performance tuning. Many of the JNDI properties are abstractions of properties that are provided by Elasticsearch.

All properties that are provided by Elasticsearch can also be set by using JNDI properties with **analytics/** prepended before the property name. For example, **threadpool.search.queue_size** is a property that is provided by Elasticsearch. It can be set with the following JNDI property.

```
<jndiEntry jndiName="analytics/threadpool.search.queue_size" value="100" />
```

These properties are normally set in a custom settings file. If you are familiar with Elasticsearch and the format of its properties files, you can specify the path to the settings file by using the **settingspath** JNDI property, as follows.

```
<jndiEntry jndiName="analytics/settingspath" value="/home/system/elasticsearch
.yml" />
```

Unless you are an expert Elasticsearch IT manager, identified a specific need, or were instructed by your services or support team, do not be tempted to fiddle with these settings.

Backing up Analytics data

Learn about how to back up your MobileFirst Analytics data.

The data for MobileFirst Analytics is stored as a set of files on the MobileFirst Analytics Server file system. The location of this folder is specified by the datapath JNDI property in the MobileFirst Analytics Server configuration. For more information about the JNDI properties, see Configuration properties.

The MobileFirst Analytics Server configuration is also stored on the file system, and is called server.xml.

You can back up these files by using any existing server backup procedures that you might already have in place. No special procedure is required when you back up these files, other than ensuring that the MobileFirst Analytics Server is stopped. Otherwise, the data might change while the backup is occurring, and the data that is stored in memory might not yet be written to the file system. To avoid inconsistent data, stop the MobileFirst Analytics Server before you start your backup.

Cluster management and Elasticsearch

Manage clusters and add nodes to relieve memory and capacity strain.

Add a Node to the Cluster

You can add a new node to the cluster by installing the MobileFirst Analytics Server or by running a standalone Elasticsearch instance.

If you choose the standalone Elasticsearch instance, you relieve some cluster strain for memory and capacity requirements, but you do not relieve data ingestion strain. Data reports must always go through the MobileFirst Analytics Server for preservation of data integrity and data optimization prior to going to persistent store.

You can mix and match.

The underlying Elasticsearch data store expects nodes to be homogenous, so do not mix a powerful 8-core 64 GB RAM rack system with a leftover surplus notebook in your cluster. Use similar hardware among the nodes.

Adding a MobileFirst Analytics Server to the cluster

Learn how to add a MobileFirst Analytics Server to the cluster.

Because Elasticsearch is embedded in the MobileFirst Analytics Server, and it is responsible for participating in the cluster, do not use the application server's features to define cluster behavior. You do not want to create a WebSphere Application Server Liberty farm, for example. Trust the underlying Elasticsearch run time to participate in the cluster. However, you must configure it properly.

In the following sample instructions, do not configure the node to be a master node or a data node. Instead, configure the node as a "search load balancer" whose purpose is to be up temporarily so that the Elasticsearch REST API is exposed for monitoring and dynamic configuration.

Notes:

- Remember to configure the hardware and operating system of this node according to the System requirements (../installation/#system-requirements).
- Port 9600 is the transport port that is used by Elasticsearch. Therefore, port 9600 must be open through any firewalls between cluster nodes.

1. Install the analytics service WAR file and the analytics UI WAR file (if you want the UI) to the application server on the newly allocated system. Install this instance of the MobileFirst Analytics Server to any of the supported app servers.
    - Installing MobileFirst Analytics on WebSphere Application Server Liberty (../installation/#installing-mobilefirst-analytics-on-websphere-application-server-liberty)
    - Installing MobileFirst Analytics on Tomcat (../installation/#installing-mobilefirst-analytics-on-tomcat)
    - Installing MobileFirst Analytics on WebSphere Application Server (../installation/#installing-mobilefirst-analytics-on-websphere-application-server)
2. Edit the application server's configuration file for JNDI properties (or use system environment variables) to configure at least the following flags.

Flag	Value (example)	Default	Note
cluster.name	worklight	worklight	The cluster that you intend this node to join.
discovery.zen.ping.multicast.enabled	false	true	Set to false to avoid accidental cluster join.

Flag	Value (example)	Default	Note
discovery.zen.ping.unicast.hosts	[“9.8.7.6:9600”]	None	List of master nodes in the existing cluster. Change the default port of 9600 if you specified a transport port setting on the master nodes.
node.master	false	true	Do not allow this node to be a master.
node.data	false	true	Do not allow this node to store data.
http.enabled	true	true	Open unsecured HTTP port 9200 for Elasticsearch REST API.

3. Consider all configuration flags in production scenarios. You might want Elasticsearch to keep the plug-ins in a different file system directory than the data, so you must set the **path.plugins** flag.
4. Run the application server and start the WAR applications if necessary.
5. Confirm that this new node joined the cluster by watching the console output on this new node, or by observing the node count in the **Cluster and Node** section of the **Administration** page in MobileFirst Analytics Console.

Adding a stand-alone Elasticsearch node to the cluster

Learn how to add a stand-alone Elasticsearch node to the cluster.

You can add a stand-alone Elasticsearch node to your existing MobileFirst Analytics cluster in just a few simple steps. However, you must decide the role of this node. Is it going to be a master-eligible node? If so, remember to avoid the split-brain issue. Is it going to be a data node? Is it going to be a client-only node? Perhaps you want a client-only node so that you can start a node temporarily to expose Elasticsearch’s REST API directly to affect dynamic configuration changes to your running cluster.

In the following sample instructions, do not configure the node to be a master node or a data node. Instead, configure the node as a “search load balancer” whose purpose is to be up temporarily so that the Elasticsearch REST API is exposed for monitoring and dynamic configuration.

Notes:

- Remember to configure the hardware and operating system of this node according to the System requirements (../installation/#system-requirements).
- Port 9600 is the transport port that is used by Elasticsearch. Therefore, port 9600 must be open through any firewalls between cluster nodes.

1. Download Elasticsearch from https://download.elastic.co/elasticsearch/elasticsearch/elasticsearch-1.7.5.tar.gz (https://download.elastic.co/elasticsearch/elasticsearch/elasticsearch-1.7.5.tar.gz).
2. Decompress the file.
3. Edit the **config/elasticsearch.yml** file and configure at least the following flags.

Flag	Value (example)	Default	Note
cluster.name	worklight	worklight	The cluster that you intend this node to join.
discovery.zen.ping.multicast.enabled	false	true	Set to false to avoid accidental cluster join.

Flag	Value (example)	Default	Note
discovery.zen.ping.unicast.hosts	[“9.8.7.6:9600”]	None	List of master nodes in the existing cluster. Change the default port of 9600 if you specified a transport port setting on the master nodes.
node.master	false	true	Do not allow this node to be a master.
node.data	false	true	Do not allow this node to store data.
http.enabled	true	true	Open unsecured HTTP port 9200 for Elasticsearch REST API.

4. Consider all configuration flags in production scenarios. You might want Elasticsearch to keep the plug-ins in a different file system directory than the data, so you must set the path.plugins flag.
5. Run `./bin/plugin -i elasticsearch/elasticsearch-analytics-icu/2.7.0` to install the ICU plug-in.
6. Run `./bin/elasticsearch`.
7. Confirm that this new node joined the cluster by watching the console output on this new node, or by observing the node count in the **Cluster and Node** section of the **Administration** page in MobileFirst Analytics Console.

Circuit breakers

Learn about Elasticsearch circuit breakers.

Elasticsearch contains multiple circuit breakers that are used to prevent operations from causing an **OutOfMemoryError**. For example, if a query that serves data to the MobileFirst Operations Console results in using 40% of the JVM heap, the circuit breaker triggers, an exception is raised, and the console receives empty data.

Elasticsearch also has protections for filling up the disk. If the disk on which the Elasticsearch data store is configured to write fills to 90% capacity, the Elasticsearch node informs the master node in the cluster. The master node then redirects new document-writes away from the nearly full node. If you have only one node in your cluster, no secondary node to which data can be written is available. Therefore, no data is written and is lost.

Last modified on November 17, 2016