Hardware and Software Requirements

RightITnow ECM (Event Correlation Manager) is designed to work on a variety of hardware and software setups, reflecting the different needs of our customers. This document outlines the basic requirements and recommended best practices for deploying ECM in a production environment.

Standard Deployment

The core of RightITnow ECM is delivered as a single, fully integrated package - both the Java backend and HTML5 Web Client are packaged in a Tomcat container, and do not require separate installation processes. Ancillary tools and services such as event proxies and APIs are delivered separately and available as downloads from our website.

A standard, single-node ECM deployment requires the core ECM application and a database to be setup. The database instance does not need to be located on the same server as ECM, however network latencies should be kept in mind and it is generally recommended that the database instance is local to ECM or on the local network.

As an alternative to installing ECM from scratch, we provide a virtual appliance (OVA) and a Docker image for download from our website. Please see the Installation Guide for more details.

Hardware Requirements

Hardware requirements can vary between deployments, depending on the volume of alerts being processed and the amount of data persisted to the database. Generally ECM can run off a single production-grade server. The following are the standard recommended specifications for running ECM in production:

- **CPU:** Dual quad-core processors (8 cores)
- **Memory:** 64GB DDR3 ECC Registered
- **Hardisk:** 128GB+ SSD for the ECM application, 1TB+ SSD for the database (see sections below about database and disk space requirements)
- **RAID:** RAID configurations are recommended where/if applicable
- **Network:** 1Gbit/s Ethernet connection

For smaller-scale deployments or development/UAT servers, these specifications can be scaled down considerably. A dual-core CPU with 16GB of RAM and 256GB HDD would suffice.

Software Requirements

The ECM application is developed in Java and thus a variety of operating systems are supported. Linux-based setups are recommended due to the nature of the product.

Operating Systems (64-bit)

- Red Hat Enterprise Linux 6 or later
- CentOS 6 or later
- Ubuntu 14.04 LTS or later
- Microsoft Windows 10 or Microsoft Windows Server 2016 or later
- Mac OS X 10.6 (Snow Leopard) or later

Software (64-bit where possible)

- Oracle JRE/JDK 8 or Oracle JDK 11 LTS or OpenJDK 11 (including Microsoft Build of OpenJDK and Amazon Corretto)
- MySQL Server 5.6/5.7/8.0 or MariaDB Server 10.4 or Microsoft SQL Server 2016/2017/2019

Database Setup

The ECM database can be setup on the same server as the ECM application, or on a separate server on the same local network. You can also run ECM in a clustered environment (MySQL Cluster), and on Amazon Aurora if deploying on AWS. It is important to avoid high latencies between separate servers as this will have a noticeable effect on performance and reduce the event-processing rate of the application.

Data replication and backup depends on the chosen database system, please see the relevant sections below for more information and links to documentation.

MySQL and MariaDB

In MySQL 5.7, some SQL modes are enabled by default, but are not supported by ECM. Only the following SQL modes should be enabled:

- sql-mode=”STRICT_TRANS_TABLES, ERROR_FOR_DIVISION_BY_ZERO, NO_AUTO_CREATE_USER, NO_ENGINE_SUBSTITUTION”

In MySQL 8.0 and MariaDB 10.4, some SQL modes are enabled by default, but are not supported by ECM. Only the following SQL modes should be enabled:

- sql-mode=”STRICT_TRANS_TABLES, ERROR_FOR_DIVISION_BY_ZERO, NO_ENGINE_SUBSTITUTION”

For further information about MySQL replication and failover, please refer to the documentation at *http://dev.mysql.com/doc/refman/5.7/en/replication.html* and *http://dev.mysql.com/doc/refman/5.7/en/replication-solutions-switch.html*

If using Amazon Aurora, Aurora Replicas can be used to achieve replication as described at https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Aurora.Replication.html

The default settings that MySQL and MariaDB ship with should suffice, however the following settings can be changed beforehand which can avoid certain known issues:

- change the **innodb_lock_wait_timeout** setting from 50 to 300
- for any errors related to "Packet for query is too large", change the **max_allowed_packet** setting to a larger size such as 8MB

Further optimizations to the database can be done – it is recommended to follow best practices and to read the relevant optimization guides:

- http://dev.mysql.com/doc/refman/5.7/en/optimization.html
- https://mariadb.com/kb/en/optimization-and-tuning/

MySQL Cluster

In addition to the requirements mentioned in this section, the following server configuration is required:

- Gigabit network with low latencies
- MySQL 5.7 or above with NDB 7.5.5 or above
- "NDBCLUSTER" as the cluster engine name

MySQL Cluster requires several nodes to function properly and to provide automatic redundancy. Please refer to the documentation here for installation and setup instructions.

Microsoft SQL Server

Installing and configuring SQL Server depends on your environment and choice of operating system. The official documentation can be found here. To replicate your database and distribute it to different locations, please refer to the documentation here.

Azure SQL Database is supported. The Standard or Premium tiers are recommended for a production ECM deployment, while Basic can be used for a development/UAT server. Please refer to the Service Tiers documentation.

Disk Space Requirements

The amount of disk space required by ECM typically depends on the number of alerts that is to be persisted to the database. The majority of disk space is taken up by the database. An

approximate guide is 10mb per 1,000 alerts/month i.e. if 30,000 alerts are processed every month, database size would increase by 300MB monthly.

The **purge utility** in ECM is designed to restrict the volume of alerts, events and other data being persisted in ECM by routinely cleaning up old and expired data. It is highly advisable to configure the purging in the initial stages of setting up ECM.

High-Availability and Failover

HA and failover scenarios vary from customer to customer, but a typical setup consists of a dual-server configuration (hot and cold standby), load balancers and database replication if using MySQL/MariaDB/SQL Server, or a MySQL Cluster which has built-in redundancy. The diagram and explanation below illustrates this setup.

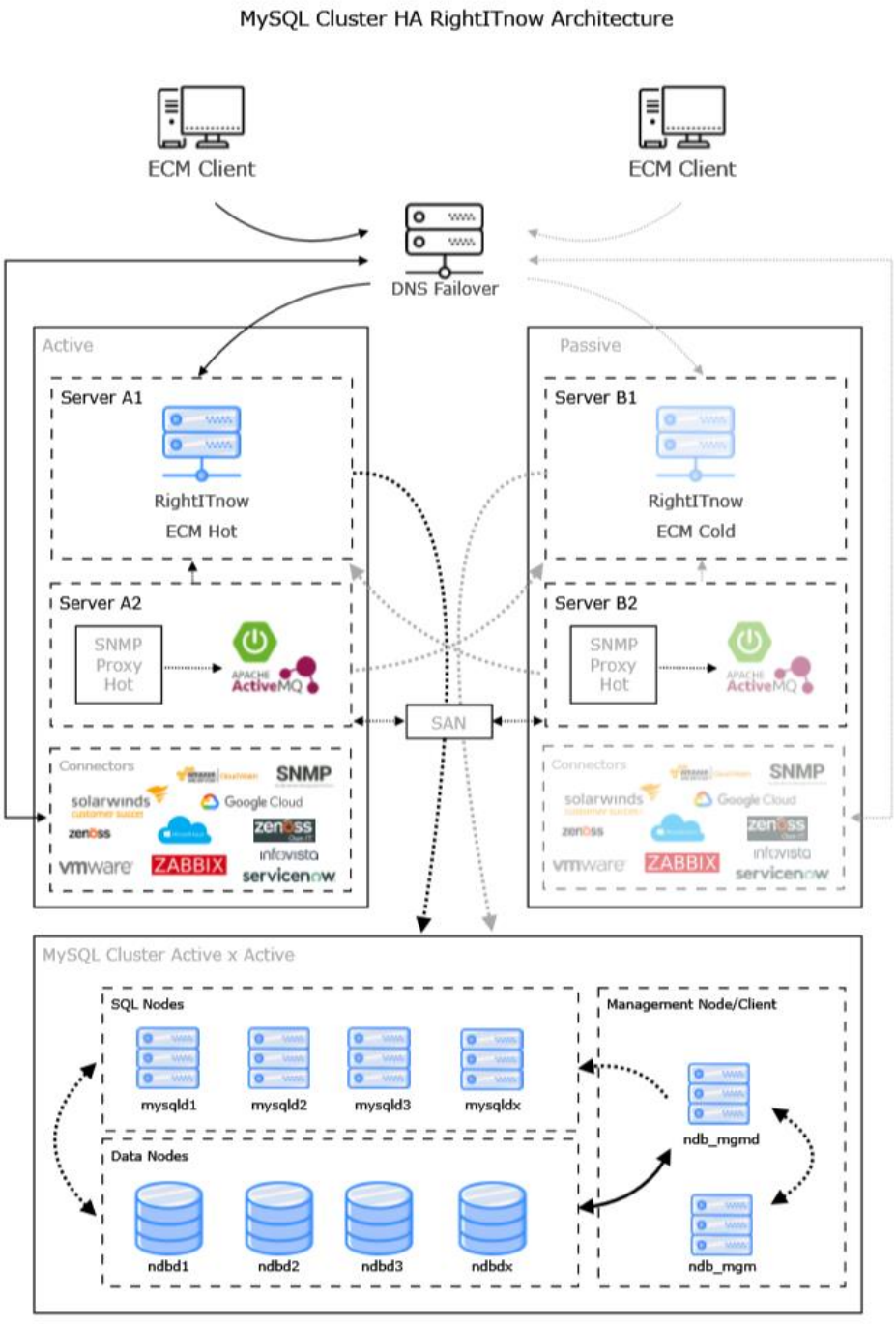

Figure 1

ECM Nodes

An ECM node consists of the ECM application itself and the database that it is using, regardless of whether this database is local or not. To achieve HA/failover, a backup node is required also consisting of the ECM application and a database instance. The backup node will be an exact replica of the primary node and typically runs on cold standby. Once the primary ECM node is determined to be down, switching to the backup node is a matter of redirecting web users towards the backup node and switching the backup database instance to become the new master (if not using MySQL Cluster).

Replication of the database depends on the database system in use. MySQL Cluster automatically replicates data across its nodes, and as such no extra steps need to be done when failing over to the backup node, since it will also point to the same cluster.

For MySQL, we recommend using **semi-synchronous** replication where the master only commits the transaction after at least one slave acknowledges receiving and logging the events for this transaction. This is the recommended approach when dealing with the scenario in Figure 1, since it performs better than fully synchronous replication while guaranteeing consistency between the databases.

Proxies

In the context of RightITnow, a proxy is a standalone application that listens for a particular type of event, performs some transformation logic on the event and then forwards it onto RightITnow ECM for processing. RightITnow provides the following standard proxies, which are developed in Java and Python, and generally have very little overhead:

- SNMP – receives SNMP v1, v2 and v3 traps from any source
- Syslog – receives Syslog messages from any source
- Mail – connects to a mail server and retrieves emails to be processed as events

Proxies can send events to ECM in 2 ways:

SOAP API: Events are sent in via the ECM SOAP Event API. This is recommended for proxies which do not generate a large volume of events and are not mission-critical.

ActiveMQ: Events are not sent directly to ECM but to an ActiveMQ instance, which is capable of handling larger volumes and can backup these events to disk. A connector is then configured in ECM to listen on the ActiveMQ queue and retrieve any events for processing. This is the recommended approach for handling large volumes of mission-critical events.

As shown in Figure 1, multiple proxies of the same type can deployed on different servers. This ensures HA/failover, and can also accommodate load balancing of the client's incoming event flow. Note that the proxies do not synchronize the data that they receive or communicate with each other in any way, since deduplication and categorization of events into alerts will be performed in ECM.

For further information, please refer to the documentation pertaining to each proxy type, which can be found on the Getting Started displet of the ECM web interface.

ActiveMQ

ActiveMQ is an open source messaging and integration server which is widely used in conjunction with Java applications for messaging between components. ECM is able to collect events (that were sent by proxies) using the ActiveMQ connector, which is defined and configured through the ECM web interface. In this scenario, the proxies are ActiveMQ *producers* while ECM is a *consumer*.

In order to maximize HA/failover in the context of ECM, the ActiveMQ instances can implement the Shared File System Master-Slave concept, where multiple brokers point on the same queue, and if one of them fails another is ready to take its place automatically. This is applicable only where a low-latency SAN or shared file system is available, since the brokers will be on different servers. It is also desirable for the SAN or shared file system to support replication, so that ActiveMQ messages are never lost. In the above scenario, the connector in ECM is setup to automatically failover between different ActiveMQ brokers by using the failover protocol in the URI pointing to ActiveMQ e.g. failover:(tcp://broker1:61616,tcp://broker2:61616).

Note that ECM makes use of internal ActiveMQ queues when processing events. It is possible for users of ECM to create and configure these queues on their own infrastructure and then configure ECM to use these queues instead of the internal ones. This has several advantages in a HA/failover setup as it allows users to allocate more disk space and memory for the queues and backs up the queues to disk, preventing events from being lost during a failover.

For further information on ActiveMQ, please refer to the documentation at *http://activemq.apache.org*. Documentation for the Shared File System Master-Slave setup can be found at *http://activemq.apache.org/shared-file-system-master-slave.html*

Native Connectors

A native connector is one that is configured directly within ECM and does not make use of proxies, but rather they collect events directly, typically via polling the target system, and immediately passing in the event into the ECM processing subsystem. SolarWinds, VMware, ManageEngine and MS SCOM are just some examples of the native connectors supported within ECM. Achieving HA and failover with these connectors is automatically provided for with the proposed configuration, since the backup node will be configured to poll the same connectors as the primary node.