High Availability and Disaster Recovery Guide

Document Version 2.0

July 9, 2018

10016-02 EN Rev. A

www.ThreatConnect.com
info@threatconnect.com

TOLL FREE: 1.800.965.2708
LOCAL: +1.703.229.4240
FAX: +1.703.229.4489

THREATCONNECT, INC.
3865 WILSON BLVD., SUITE 550
ARLINGTON, VA 22203

Table of Contents

Introduction 4

Setting Up Replication for MySQL Databases 4

Primary 5

Secondary 6

Setting Up Replication for the ThreatConnect Application Server 8

Document Storage on the ThreatConnect Application Server 9

Setting Up Replication for Elasticsearch 9

INTRODUCTION

In order to achieve High Availability and Disaster Recovery, two instances of the App, MySQL, and Elasticsearch® servers are needed. However, it is not recommended for any of these servers to stay on the same machine, or virtual machine, with any of the others. This document will outline all necessary steps to achieve successful replication.

SETTING UP REPLICATION FOR MYSQL DATABASES

This section discusses the built-in replication capability of MySQL in a master-master replication configuration in Linux®. In this configuration, the application server will write to the primary database, while available, but can switch to using the secondary database if the primary becomes database unavailable. It is assumed that this configuration will be completed before the ThreatConnect database is created, since it requires different steps if that is the case.

Below are sample configuration options for **my.cnf**, for both the primary and secondary MySQL instances. Properties that have different values are located at the top for easy reference. These examples show limited options and may not reflect the proper values or all those required to run MySQL. Please shut down the **mysql** server before modifying these files. The **my.cnf** file is typically located at /etc/my.cnf:

```
# service mysql stop
```

Example my.cnf files:

Primary

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
# Disabling symbolic-links is recommended to prevent assorted security
risks
symbolic-links=0
# Recommended in standard MySQL setup
sql_mode=NO_ENGINE_SUBSTITUTION,NO_ZERO_IN_DATE,NO_ZERO_DATE
lower_case_table_names=1
character_set_server=utf8
collation_server=utf8_general_ci
innodb_flush_log_at_trx_commit=2
innodb_table_locks=0
innodb_autoinc_lock_mode=2
eq_range_index_dive_limit=200
innodb_large_prefix=on
innodb_file_format=barracuda
innodb_buffer_pool_size=1G
user=mysql
group_concat_max_len=1000000
#Replication Settings
server-id=1
auto_increment_offset=1
auto_increment_increment=2
sync_binlog=1
log-bin=""
[mysqld_safe]
pid-file=/var/run/mysqld/mysqld.pid
```

Secondary

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
# Disabling symbolic-links is recommended to prevent assorted security
risks
symbolic-links=0
# Recommended in standard MySQL setup
sql_mode=NO_ENGINE_SUBSTITUTION,NO_ZERO_IN_DATE,NO_ZERO_DATE
lower_case_table_names=1
character_set_server=utf8
collation_server=utf8_general_ci
innodb_flush_log_at_trx_commit=2
innodb_table_locks=0
innodb_autoinc_lock_mode=2
eq_range_index_dive_limit=200
innodb_large_prefix=on
innodb_file_format=barracuda
innodb_buffer_pool_size=1G
user=mysql
group_concat_max_len=1000000
#Replication Settings
server-id=2
auto_increment_offset=2
auto_increment_increment=2
sync_binlog=1
log-bin=""
[mysqld_safe]
pid-file=/var/run/mysqld/mysqld.pid
```

Make sure that the directories specified exist and are owned by the user as which the **mysql server** runs. After configuring the respective server instances, start MySQL on each, and confirm they started without any errors:

```
# service mysql start
```

In a master-master replication configuration, each database connects to the other as a slave. Both instances must have a user configured to allow the other database to connect and also have permissions granted for replication.

On each database, run the command listed below from the **mysql** client, replacing <server> with the hostname or 'IP' of the opposite machine on which it is being run. If using a hostname, DNS must be configured for the user to authenticate. Alternatively, replace '<server>' with '%' to allow the replication user to connect from any server.

```
mysql> CREATE USER 'replication'@'<server>' IDENTIFIED BY 'password';
mysql> GRANT REPLICATION SLAVE ON *.* TO 'replication'@'<server>';
```

Configure each server with the position of the binary log of the opposite server, so it can start replicating from that point. On the primary server, run the following command:

```
mysql> SHOW MASTER STATUS;
```

A screen similar to Figure 1 will be displayed.

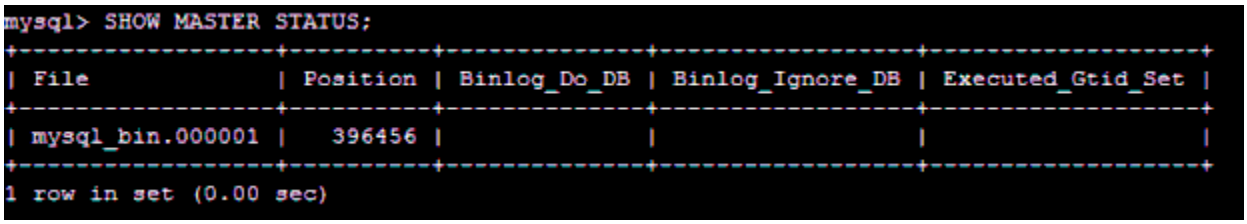

Figure 1

Record the values under the **File** and **Position** columns. On the secondary server run the following commands, replacing <server> with the hostname of the primary server and using the appropriate file and position obtained from the command above:

```
mysql> CHANGE MASTER TO MASTER_HOST='<server>',
mysql> MASTER_USER='replication',
mysql> MASTER_PASSWORD='password',
mysql> MASTER_LOG_FILE='mysql_bin.000001',
mysql> MASTER_LOG_POS=396456;
```

Repeat the steps above on the opposite server as shown below.

On the secondary server, run the following command:

```
mysql> SHOW MASTER STATUS;
```

A screen similar to Figure 1 will be displayed.

Record the values under the **File** and **Position** columns. On the primary server, run the following commands, replacing <server> with the hostname of the secondary server and using the appropriate file and position obtained from the command above:

```
mysql> CHANGE MASTER TO MASTER_HOST='<server>',
mysql> MASTER_USER='replication',
mysql> MASTER_PASSWORD='password',
mysql> MASTER_LOG_FILE='mysql_bin.000001',
mysql> MASTER_LOG_POS=396456;
```

Both instances have now been configured to be master and slave to each other, and the slave processes can now be started. On each instance, run the command below:

```
mysql> START SLAVE;
```

To verify that replication is configured correctly, run the following command on each instance:

```
mysql> SHOW SLAVE STATUS \G;
```

On each instance, look for a line stating “Slave_IO_State: Waiting for master to send event.”

At this point, the ThreatConnect database can be created on the primary node, and it will be replicated to the secondary.

SETTING UP REPLICATION FOR THE THREATCONNECT APPLICATION SERVER

This section discusses the setup of replication for the ThreatConnect application server. Follow the instructions explained in the *ThreatConnect Installation Guide_Linux Operating System_Software Version 5.6 or Newer*, and include the modifications detailed below.

Each ThreatConnect application server needs to be pointed to its own instance of the database. Additionally, there can only be **one** server with monitors enabled in threatconnect.xml (in /opt/threatconnect/config or in C:\threatconnect\config). If this is not the case, it will result in MySQL key collisions due to the jobs being executed on both servers and writing duplicate data.

To ensure that only one server executes jobs and to prevent duplicate data, set the following parameters accordingly:

Path: /<ThreatConnect install location>/threatconnect/config/

File: threatconnect.xml

Change: <property name="monitorsEnabled" value="true"/>

To: <property name="monitorsEnabled" value="false"/>

DOCUMENT STORAGE ON THE THREATCONNECT APPLICATION SERVER

To achieve High Availability and Disaster Recovery for the application server, **'documentStorageLocalPath'** defined within System Settings needs to be an area of common storage shared between the servers mounted locally, such as SAN or NAS. Note that ThreatConnect does not provide specific guidelines for SAN or NAS setup; however, it is strongly recommended that SAN and NAS have redundancy with multiple drives, and that the area in which **'documentStorageLocalPath'** resides be backed up accordingly with a user's organizational standards.

SETTING UP REPLICATION FOR ELASTICSEARCH

This section discusses the setup of replication for Elasticsearch. Follow the instructions explained in the *ThreatConnect Elasticsearch Setup Guide Software Version 5.6 or Newer*, and include the modifications to the configuration that are detailed below. Replace HOST1 and HOST2 with the hostname that is resolvable via DNS. It is suggested that both ElasticSearch servers are managed by a load balance server in order to achieve a true HA setup.

```
HOST1:
cluster.name: elasticTconnect
node.name: ${HOSTNAME}
discovery.zen.ping.unicast.hosts: {"HOST1", "HOST2"}

HOST2:
cluster.name: elasticTconnect
node.name: ${HOSTNAME}
discovery.zen.ping.unicast.hosts: {"HOST1", "HOST2"}
```

If the hostnames are not resolvable via DNS, use IP server addresses as shown below.

```
HOST1:
cluster.name: elasticTconnect
node.name: node-1
discovery.zen.ping.unicast.hosts: {“IP.ADDRESS.OF.SERVER1”,
“IP.ADDRESS.OF.SERVER2”}

HOST2:
cluster.name: elasticTconnect
node.name: node-2
discovery.zen.ping.unicast.hosts: {“IP.ADDRESS.OF.SERVER1”,
“IP.ADDRESS.OF.SERVER2”}
```

NOTE: It is preferable for the hostnames to be resolvable via DNS, because if an IP address changes, it will break replication.

NOTE: It is recommended that the user have a minimum of three Elasticsearch servers in the cluster to prevent data loss, if choosing the cluster option with Elasticsearch.