ThreatConnect® High Availability and Disaster Recovery Guide

Document Version 7.0

Technical Guide

December 13, 2022

10016-07 EN Rev. A

ThreatConnect, Inc.
3865 Wilson Blvd., Suite 550, Arlington, VA 22203

P: 1.800.965.2708 I F: 703.229.4489
www.ThreatConnect.com

Table of Contents

Overview **4**

High Availability **4**

Setting Up Replication for MySQL Databases 4

Setting Up Replication for PostgreSQL Databases 7

Primary/Master 7

Secondary/Standby 8

Testing 9

Setting Up Replication for the ThreatConnect Application Server 9

Document Storage on the ThreatConnect Application Server 9

Setting Up Replication for OpenSearch 10

Disaster Recovery **11**

Requirements 11

Preparation 11

Checkpoint 11

Phases 12

Pre-reboot 12

Check 12

Failover 12

Failover Check 13

Failback 13

Failback Check 14

Overview

To achieve High Availability and Disaster Recovery, two instances of the App, MySQL® or PostgreSQL®, and OpenSearch® 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.

Important: ThreatConnect requires an available instance of the MySQL 8.0 database or PostgreSQL v14 database.

High Availability

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 one becomes 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.

1. Shut down the **mysql** server before modifying each **my.cnf** file. This file is typically located at **/etc/my.cnf**:

```
# service mysql stop
```

2. Configure the following properties in the **my.cnf** file for the **primary** database:

```
[mysqld]
# Replication Settings
server-id=1
auto_increment_offset=1
auto_increment_increment=2
sync_binlog=1
log-bin=""
max_connections = 300
[mysqld_safe]
pid-file=/var/run/mysqld/mysqld.pid
```

3. Configure the following properties in the **my.cnf** file for the **secondary** database:

```
[mysqld]
# Replication Settings
server-id=2
auto_increment_offset=2
auto_increment_increment=2
sync_binlog=1
log-bin=""
max_connections = 300
[mysqld_safe]
pid-file=/var/run/mysqld/mysqld.pid
```

4. 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
```

5. In a master–master replication configuration, each database connects to the other as a slave. Ensure that both instances have a user configured to allow the other database to connect and have permissions granted for replication.

6. 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>';
```

7. 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;
```

8. A screen similar to Figure 1 will be displayed.

```
mysql> SHOW MASTER STATUS;
+------------------+----------+--------------+------------------+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+------------------+----------+--------------+------------------+-------------------+
| mysql_bin.000001 |   396456 |              |                  |                   |
+------------------+----------+--------------+------------------+-------------------+
1 row in set (0.00 sec)
```

Figure 1

9. 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 previous command:

```
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;
```

10. Repeat the foregoing steps on the opposite server as shown below. On the secondary server, run the following command:

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

11. A screen similar to Figure 1 will be displayed.

12. 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 previous command:

```
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;
```

13. 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 following command:

```
mysql> START SLAVE;
```

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

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

15. 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 PostgreSQL Databases

PostgreSQL leverages streaming replication to provide a viable High Availability solution. To set up the HA feature, two or more PostgreSQL servers need to be built and run. One must be configured as the Master/Primary node, and the other as Secondary/Standby node. This section provides instructions on how to configure these nodes to communicate in a streaming, asynchronous design to minimize data loss in a failover scenario.

There are other options to configure replication between two or more nodes in a cluster. Synchronous communication can be enabled instead of asynchronous, and File-Based Log Shipping can be used instead of direct streaming of the Write-Ahead Log (WAL) records. ThreatConnect has determined that streaming the WAL records in an asynchronous manner is the best option to maximize data integrity.

Note: These instructions are for configuring replication for multiple PostgreSQL instances, but the failover implementation is solely at the discretion of the customer. Third-party HA software is responsible for identifying which nodes are down and which secondary nodes are available to activate for operation.

Primary/Master

Important: Shut down any active Postgres® instances before proceeding.

1. Initialize the database on the master node.
2. Create a user with replication privileges by running the following command:

```
CREATE USER <username> REPLICATION LOGIN ENCRYPTED PASSWORD '<password>';
```

3. Configure properties in **postgresql.conf**:

```
# Possible values are replica|minimal|logical
wal_level = replica
# required for pg_rewind capability when standby goes out of sync with master
wal_log_hints = on
# sets the maximum number of concurrent connections from the standby servers.
max_wal_senders = 3
# The below parameter is used to tell the master to keep the minimum number of
# segments of WAL logs so that they are not deleted before standby consumes
them.
# each segment is 16MB
```

```
wal_keep_segments = 8
# The below parameter enables read only connection on the node when it is in
# standby role. This is ignored when the server is running as master.
hot_standby = on
max_connections = 300
```

4. Add replication entry in **pg_hba.conf** file:

```
# Allow replication connections from localhost,
# by a user with the replication privilege.
# TYPE    DATABASE    USER    ADDRESS    METHOD
host    replication    <repl_user>    IPaddrpgess(CIDR)    md5
```

5. Restart the master node for changes to take effect.

Secondary/Standby

1. Create the backup of the master node using the **pg_basebackup** utility. This is the baseline of secondary from primary snapshot. Use the following command:

```
pg_basebackup -R -D <data_directory> -h <master_host> -X stream -c fast -U
<repl_user> -W
```

2. Update properties in **recovery.conf** file (which should be generated automatically):

```
# Specifies whether to start the server as a standby. In streaming replication,
# this parameter must be set to on.
standby_mode = 'on'
# Specifies a connection string which is used for the standby server to connect
# with the primary/master.
primary_conninfo = 'host=<master_host> port=<postgres_port>
user=<replication_user> password=<password> application_name="host_name"'
# Specifies recovering to a particular timeline. The default is to recover
along the
# same timeline that was current when the base backup was taken.
# Setting this to latest recovers to the latest timeline found
# in the archive, which is useful in a standby server.
recovery_target_timeline = 'latest'
```

3. Start the standby server and check for errors.

Testing

1. Run the following command to test the replication function:

```
"SELECT * FROM pg_stat_replication;"
```

2. Create an object in the Master, and see if the object exists in the Slave (e.g., create a test database).

Setting Up Replication for the ThreatConnect Application Server

This section discusses the setup of replication for the ThreatConnect application server. Follow the instructions provided in *ThreatConnect Installation Guide_Linux Operating System*, and include the modifications detailed next.

Each ThreatConnect application server needs to be pointed to its own instance of the database. In addition, make sure that there is only ***one*** server with monitors enabled in **threatconnect.xml** (in **/opt/threatconnect/config** or in **C:\threatconnect\config**). If this is not the case, key collisions will result because of 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 OpenSearch

This section discusses the setup of replication for OpenSearch. Follow the instructions provided in the "OpenSearch Installation" section of *ThreatConnect Installation Guide_Linux Operating System*, 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 OpenSearch servers are managed by a load balance server in order to achieve a true HA setup.

```
HOST1:
cluster.name: opensearch-cluster
node.name: ${HOSTNAME}
discovery.seed_hosts: {"HOST1", "HOST2"}

HOST2:
cluster.name: opensearch-cluster
node.name: ${HOSTNAME}
discovery.seed_hosts: {"HOST1", "HOST2"}
```

If the hostnames are not resolvable via DNS, use IP server addresses as shown in the following code:

```
HOST1:
cluster.name: opensearch-cluster
node.name: node-1
discovery.seed_hosts: {"IP.ADDRESS.OF.SERVER1", "IP.ADDRESS.OF.SERVER2"}

HOST2:
cluster.name: opensearch-cluster
node.name: node-2
discovery.seed_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 OpenSearch servers in the cluster to prevent data loss if choosing the cluster option with OpenSearch.

Disaster Recovery

Disaster recovery (DR) procedures are highly specific to the individual organization. Use the general procedure outlined in this section as a guideline, and contact the ThreatConnect Deployment Team for assistance in setting up a procedure for your organization.

Note: This procedure includes additional phases for a DR test outage. For a true outage, only the steps listed in the "Failover" and "Failover Check" sections need to be completed.

Requirements

- A fully qualified domain name (FQDN) for both infrastructures (load balancer, firewall, etc.).

Preparation

- Arrange the team's availability for root access, VM access, and end-user testing.
- Notify the system operation process team and any teams that will be impacted by the outage.
- Confirm that services impacted by the failover are running correctly.

Checkpoint

- Disable service monitoring (maintenance mode).
- Ensure that backups and snapshots are completed.

Phases

Pre-reboot

Note: This phase is used to check if the main servers are restarting correctly.

1. Stop ThreatConnect on the primary site.
2. Stop OpenSearch on the primary site.
3. Stop the database service on the main server.
4. Reboot the cluster (ESX, Nutanix, etc.) on the primary site.
5. Start OpenSearch on the primary site.
6. Start the database service on the primary site.
7. Start ThreatConnect on the primary site.
8. Check OpenSearch service on the primary site.
9. Check the database service on the primary site.

Check

1. Check ThreatConnect on the primary site.
2. Check the system health of the primary site.
3. Check Indicators on the primary site.
4. Check Playbooks on the primary site.
5. Download Indicators from the primary site.
6. Check all apps on the primary site.
7. Perform regression testing and day-to-day actions on the primary site.
8. Monitor all jobs on the primary site.

Failover

1. Stop ThreatConnect on the primary site.
2. Stop OpenSearch on the primary site.

3. Stop the database service on the primary site.
4. Stop the cluster (ESX, Nutanix, etc.) on the primary site.
5. Check the load balancer or firewall, if a manual configuration needs to be activated.
6. Start the cluster on the primary site.
7. Start OpenSearch on the secondary site.
8. Start the database service on the secondary site.
9. Start ThreatConnect on the secondary site.

Failover Check

1. Check OpenSearch service on the secondary site.
2. Check the database service on the secondary site.
3. Check ThreatConnect on the secondary site.
4. Check the system health of the secondary site.
5. Check Indicators on the secondary site.
6. Check Playbooks on the secondary site.
7. Download Indicators from the secondary site.
8. Check all apps on the secondary site.
9. Perform regression testing and day-to-day actions on the secondary site.
10. Monitor all jobs on the secondary site.

Failback

1. Stop ThreatConnect on the secondary site.
2. Stop OpenSearch on the secondary site.
3. Stop the database service on the secondary site.
4. Stop the cluster (ESX, Nutanix, etc.) on the secondary site.
5. Check the load balancer or firewall, if a manual configuration needs to be activated.
6. Start the cluster on the primary site.
7. Start OpenSearch on the primary site.

8. Start the database service on the primary site.
9. Start ThreatConnect on the primary site.

Failback Check

1. Check ThreatConnect on the primary site.
2. Check the system health of the primary site.
3. Check Indicators on the primary site.
4. Check Playbooks on the primary site.
5. Download Indicators from the primary site.
6. Check all apps on the primary site.
7. Perform regression testing and day-to-day actions on the primary site.
8. Monitor all jobs on the primary site.