ThreatConnect® Installation Guide: Containerized Deployment

Software Version 7.10

Technical Guide

August 22, 2025

10032-10 EN Rev. A

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

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

Table of Contents

Overview

This guide describes how to install ThreatConnect®. As of ThreatConnect version 7.5, you will no longer be required to install Java®, Python®, OpenSearch®, and Redis® as part of the ThreatConnect installation process. Instead, all of this software, along with ThreatConnect, is now packaged together in a containerized solution using Docker® or Podman.

Important: The containerized deployment was tested on AlmaLinux OS™ 9, Red Hat® Enterprise Linux® (RHEL8), and RHEL 9 and is the standard deployment method for all production and non-production systems. For instructions on installing ThreatConnect and having it run on an operating system (OS), see *ThreatConnect Installation Guide: Linux Operating System Legacy Deployment*.

Important: The `.env` file holds all passwords and configurations for the containerized deployment. Once the container is running, the `.env` file can be purged.

Prerequisites

Credentials

- Amazon Web Services® (AWS) Access Key ID
- AWS Secret Access Key

System Requirements

Hardware

ThreatConnect requires a server, virtual or physical, that meets the specifications listed in Tables 1–3.

Note: Multi-server installations are for advanced users only, who should consult with ThreatConnect as to the correct sizing that will meet their needs. See *ThreatConnect System Requirements* for additional information.

Table 1

	Memory Min (GB)[1],[2]	Min CPU Cores / vCPUs (2GHz)[3]	Estimated Storage (GB) [4],[5]
ThreatConnect Application	64	16	300
Containerized Redis	8	2	20
Containerized OpenSearch	32	12	60
Containerized Database	64	16	120

Important: The following guidelines apply to production deployments:

- The ThreatConnect Application and Redis can be deployed to the same server (virtual or physical).
- OpenSearch containers should be deployed to a dedicated server.
- Database containers should be deployed to a dedicated server.

[1]Allocated to ThreatConnect containers; the OS will need additional space.
[2]While Java virtual machines will be allocated memory, there is some allowance for additional memory available for Feed and Playbook Apps.
[3]While Java virtual machines will be allocated memory, there is some allowance for additional memory available for Feed and Playbook Apps.
[4]High IOPS, ideally SSDs, are preferred.
[5]ThreatConnect must be installed on an ext4 or XFS partition.

Table 2

	Highly Available Document Storage (usually network-mounted storage)
Document Storage	Equal to the desired capacity of documents stored

Table 3

	Memory Minimum (GB)	**Memory Recommended (GB)**
Swap Space	4	8

Note: As the number of users increases, or as the frequency or complexity of automated analysis increases, the need to increase system resources will likely occur.

Installation Steps

Step 1: Install Docker

Important: Steps 1 and 2 apply to Docker deployments only. For Podman deployments, skip to Step 3.

Note: You must complete this step on all hosts that will run ThreatConnect or some component of ThreatConnect.

Run the following commands to install Docker:

```
None
yum-config-manager --add-repo \
  https://download.docker.com/linux/centos/docker-ce.repo
yum install docker-ce docker-ce-cli containerd.io
systemctl start docker.service
systemctl enable docker.service
docker version
```

Step 2: Install Docker Compose

Important: Steps 1 and 2 apply to Docker deployments only. For Podman deployments, skip to Step 3.

Note: You must complete this step on all hosts that will run ThreatConnect or some component of ThreatConnect.

Run the following commands to install Docker Compose:

```
None
curl -SL
https://github.com/docker/compose/releases/download/v2.24.5/docker-compose-linu
x-x86_64 \
    -o /usr/local/bin/docker-compose
ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
chmod 755 /usr/local/bin/docker-compose
docker-compose version
```

Step 3: Install Podman

Important: Steps 3 and 4 apply to Podman deployments only. For Docker deployments, skip to Step 5.

Note: You must complete this step on all hosts that will run ThreatConnect or some component of ThreatConnect.

Run the following commands to install Podman:

```
None
dnf install -y podman
export PODMAN_PATH=$(which podman)
ln -s $PODMAN_PATH /usr/bin/docker
```

Step 4: Install Podman Compose

Important: Steps 3 and 4 apply to Podman deployments only. For Docker deployments, skip to Step 5.

Note: You must complete this step on all hosts that will run ThreatConnect or some component of ThreatConnect.

Run the following commands to install Podman Compose:

```
None
dnf install -y python3.11 python3.11-pip
pip3.11 install podman-compose
ln -s /usr/local/bin/podman-compose /usr/bin/docker-compose
ln -s /usr/local/bin/podman-compose /usr/local/bin/docker-compose
ln -s /usr/local/bin/podman-compose /bin/docker-compose
```

Step 5: Install AWS CLI

Note: You must complete this step on all hosts that will run ThreatConnect or some component of ThreatConnect.

Amazon Web Services Command Line Interface (AWS CLI) is used to download container images directly from ThreatConnect's Elastic® Container Registry (ECR).

Run the following commands to install AWS CLI:

```
None

cd /opt
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" \
       -o "awscliv2.zip" &&\
       unzip awscliv2.zip &&\
       ./aws/install
```

Step 6: Increase vm.max_map_count

Note: You must complete this step on the host that will run OpenSearch.

Run the following commands to increase `vm.max_map_count`:

```
None
echo 'vm.max_map_count=262144' >> /etc/sysctl.conf
echo 'net.ipv4.ip_unprivileged_port_start=25' >> /etc/sysctl.conf
sysctl -p
```

Step 7: Open Ports

Note: You must complete this step on all hosts that will run ThreatConnect or some component of ThreatConnect.

Use either Option 1 or Option 2 to open the ports:

Option 1: Add the following rules to the firewallD:

```
None

#tc-app
firewall-cmd --permanent --zone=public --add-port=25/tcp
firewall-cmd --permanent --zone=public --add-port=443/tcp
#tc-mon
firewall-cmd --permanent --zone=public --add-port=5445/tcp
```

```
firewall-cmd --permanent --zone=public --add-port=6379/tcp
firewall-cmd --permanent --zone=public --add-port=62000/tcp
#opensearch
firewall-cmd --permanent --zone=public --add-port=9200/tcp
#postgres
firewall-cmd --permanent --zone=public --add-port=5432/tcp
firewall-cmd --reload
```

Option 2: Add the following rules to the iptables:

Important: These rules will not persist after an OS reboot. It is suggested that firewall-cmd be used for persistent rules.

```
None

#tc-app
iptables -A INPUT -p tcp --dport 25 -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
#tc-mon
iptables -A INPUT -p tcp --dport 5445 -j ACCEPT
iptables -A INPUT -p tcp --dport 6379 -j ACCEPT
iptables -A INPUT -p tcp --dport 62000 -j ACCEPT
#opensearch
iptables -A INPUT -p tcp --dport 9200 -j ACCEPT
#postgres
iptables -A INPUT -p tcp --dport 5432 -j ACCEPT
```

Step 8: Create Local Users

Note: This step does not apply to Podman deployments running ThreatConnect. If the database, Redis, and OpenSearch all run on the same host, you must use one user for all containers.

1. Create **threatconnect** and **tc-job** user accounts. This example uses `2000` and `threatconnect` for the **threatconnect** user account's user ID (UID) and username, respectively, and `2001` and `tc-job` for the **tc-job** user account's UID and username, respectively.

   Note: You must complete this step on the hosts that will run messaging (**tc-mon**), applications (**tc-app**), and Playbooks (**tc-job**).

```
None

adduser --uid 2000 threatconnect
adduser --uid 2001 tc-job
```

2. Create the **postgres** user account. This example uses `2002` and `postgres` for the UID and username, respectively.

   Note: You must complete this step on the host that will run the database container.

```
None
adduser --uid 2002 postgres
```

3. Create the **redis** user account. This example uses `2003` and `redis` for the UID and username, respectively.

   Note: You must complete this step on the host that will run the database container.

```
None
adduser --uid 2003 redis
```

4. Create the **opensearch** user account. The **opensearch** UID must be 1000. Therefore, if UID=1000 already exists, skip this step. This example uses `opensearch` for the username.

   Note: You must complete this step on the host that will run the database container.

```
None
adduser --uid 1000 opensearch
```

Step 9: Configure Rootless Podman

Important: This step applies to rootless Podman deployments only. For all other deployments, skip to Step 10.

Note: You must complete this step on all hosts that will run ThreatConnect or some component of ThreatConnect.

In the following steps, use the username created for the **threatconnect** user account in Step 8:

1. Execute `enable-linger` on user **threatconnect**:

```
None
loginctl enable-linger threatconnect
```

2. Log in as the **threatconnect** user:

```
None
su - threatconnect
```

Important: For rootless Podman deployments, perform the rest of the steps in this guide as non-root user **threatconnect**.

Step 10: Download ThreatConnect Docker ZIP File

Important: This step applies to Docker *and* Podman deployments. The term "Docker" in the ZIP file is just part of the filename and is not specific to Docker deployments.

Note: You must complete this step on all hosts intended to run ThreatConnect or some component of ThreatConnect.

Download `ThreatConnect-Docker-v<version number>.zip`, where `<version number>` is a placeholder value for the ThreatConnect version you are installing. For example, to download the ThreatConnect Docker ZIP file for ThreatConnect 7.10.0, run the following commands:

```
None
unzip threatconnect-docker-v7.10.0.zip
cd threatconnect-docker
```

Important: The rest of the steps in this guide assume you are in the unzipped ThreatConnect Docker directory.

Step 11: Fix Shell Scripts

Note: You must complete this step on all hosts that will run ThreatConnect or some component of ThreatConnect.

Reformat and change permissions on shell scripts:

```
None
sed -i 's/\r$//' load_schema.sh
chmod 755 load_schema.sh
sed -i 's/\r$//' docker-entrypoint.d/00_init.sh
chmod 755 docker-entrypoint.d/00_init.sh
sed -i 's/\r$//' docker-entrypoint.d/97_trusted_certs.sh
chmod 755 docker-entrypoint.d/97_trusted_certs.sh
sed -i 's/\r$//' docker-entrypoint.d/98_custom_ca.sh
chmod 755 docker-entrypoint.d/98_custom_ca.sh
sed -i 's/\r$//' docker-entrypoint.d/99_deploy.sh
chmod 755 docker-entrypoint.d/99_deploy.sh
sed -i 's/\r$//' docker-entrypoint.d/pythonwrapper-3.11
chmod 755 docker-entrypoint.d/pythonwrapper-3.11
sed -i 's/\r$//' docker-entrypoint.d/pythonwrapper-3.6
chmod 755 docker-entrypoint.d/pythonwrapper-3.6
sed -i 's/\r$//' create_db_user.sh
chmod 755 create_db_user.sh
sed -i 's/\r$//' set_opensearch_password.sh
chmod 755 set_opensearch_password.sh
sed -i 's/\r$//' tc-containers.sh
chmod 755 tc-containers.sh
```

Step 12: Update Environment Variables

Note: You must complete this step on all hosts that will run ThreatConnect or some component of ThreatConnect.

Follow these steps to update the environment variables:

1. Copy `.env.sample` to your `.env` file, and then update each variable in your `.env` file with the appropriate value. For descriptions of the values, refer to the comments in that file.

```
None
cp .env.sample .env
```

2. Fix the `.env` file and source it to your environment:

```
None
sed -i 's/\r$//' .env
source .env
```

Important: The rest of the steps in this guide assume you are in the `/opt/threatconnect-docker` directory and `source.env` has been run.

Step 13: Install ThreatConnect License

Note: You must complete this step on the hosts that will run messaging (**tc-mon**), applications (**tc-app**), and Playbooks (**tc-job**).

Place your ThreatConnect license XML file into `config/license.xml`.

Step 14: Add Certificates

Note: You must complete this step on all hosts that will run ThreatConnect or some component of ThreatConnect.

Follow these steps to add certificates:

1. Obtain the required certificates. These are the certificate authority (CA) certificate (`ca.pem`), the CA-signed certificate (`server.pem`), and the private key (`privkey.pem`). Copy them to the `certs/` folder. If you do not have a signed certificate bundle, refer to the following steps:
    - CA Signed Certificate
    - Self-Signed Certificate
2. Concatenate `server.pem`, `ca.pem`, and any provided intermediate certificates (e.g., `intermediate.pem`) to a file called `fullchain.pem`:

```
None
cp /path/to/server.pem certs/fullchain.pem
cat /path/to/intermediate.pem >> certs/fullchain.pem
cat /path/to/ca.pem >> certs/fullchain.pem
cp /path/to/privkey.pem certs/privkey.pem
cp certs/privkey.pem certs/postgres-privkey.pem
```

3. Create the trusted directory:

```
None
mkdir certs/trusted
```

4. Copy pem-formatted certificates of any root CAs you wish ThreatConnect to trust, such as in the following example:

    Note: Any certificates added in this step will automatically be added to the environments of all ThreatConnect Playbooks, Applications, and Services.

```
None
cp ~/my.trusted.ca.pem certs/trusted/
```

5. Correct ownership and permissions on certificate files:

```
None
chmod 744 -R certs
chmod 644 certs/privkey.pem
chmod 600 certs/postgres-privkey.pem
```

Step 15: Configure Podman Home Container

Important: Steps 15 applies to rootless Podman deployments only. For Docker deployments and deployments running Podman as root, skip to Step 16.

Run the following commands to configure the Podman home container:

```
None
mkdir run
echo "export XDG_DATA_HOME=/opt/threatconnect-docker/run" >>
/home/threatconnect/.bashrc
source /home/threatconnect/.bashrc
```

Step 16: Log Into ThreatConnect's ECR

Configure AWS CLI using the credentials your ThreatConnect Customer Success Manager shared with you:

Important: If your system is located in a time zone other than `us-east-1`, you can replace `us-east-1` with a different AWS region before running these commands. The following AWS regions are supported at this time: `ap-northeast-1`, `ap-southeast-2`, `ca-central-1`, `eu-central-1`, `eu-north-1`, `eu-south-1`, `eu-west-2`, `me-south-1`, `sa-east-1`, `us-east-1`, `us-east-2`, `us-west-1`, `us-west-2`.

```
None
/usr/local/bin/aws configure
Access Key ID:****
Secret Access Key:****
Region:us-east-1
```

```
None
docker login \
       -u AWS \
       -p $(/usr/local/bin/aws ecr get-login-password --region us-east-1) \
       373319941383.dkr.ecr.us-east-1.amazonaws.com
```

Step 17: Configure OpenSearch Data Folder

Note: This is an optional step that can be performed on the host that will run OpenSearch.

Follow these steps to configure the OpenSearch data folder:

1. Create the OpenSearch mount folders:

   Note: Execute this step only if these variables contain a value in your `.env` file.

```
None
mkdir -p ${OPENSEARCH_DATA}
mkdir -p ${OPENSEARCH_LOGS}
mkdir -p ${OPENSEARCH_SNAPSHOTS}
```

2. Run the following additional commands only if you are running rootless Podman:

   Note: Execute this step only if these variables contain a value in your `.env` file.

```
None
docker unshare chown opensearch:opensearch -R ${OPENSEARCH_DATA}
docker unshare chown opensearch:opensearch -R ${OPENSEARCH_LOGS}
docker unshare chown opensearch:opensearch -R ${OPENSEARCH_SNAPSHOTS}
```

Step 18: Configure Log Folders

Note: This is an optional step that can be performed on the hosts that will run messaging (**tc-mon**), applications (**tc-app**), and Playbooks (**tc-job**).

Follow these steps to configure the log folders:

1. Create log folders for **tc-mon**, **tc-app**, and **tc-job**:

   Note: Execute this step only if these variables contain a value in your `.env` file.

```
None
mkdir -p ${TC_MON_LOGS}
mkdir -p ${TC_APP_LOGS}
mkdir -p ${TC_JOB_LOGS}
chown threatconnect:threatconnect -R ${TC_MON_LOGS}
chown threatconnect:threatconnect -R ${TC_APP_LOGS}
chown threatconnect:threatconnect -R ${TC_JOB_LOGS}
```

2. Run the following additional commands only if you are running rootless Podman:

   Note: Execute this step only if these variables contain a value in your `.env` file.

```
None
docker unshare chown threatconnect:threatconnect -R ${TC_MON_LOGS}
docker unshare chown threatconnect:threatconnect -R ${TC_APP_LOGS}
docker unshare chown threatconnect:threatconnect -R ${TC_JOB_LOGS}
```

Step 19: Configure ThreatConnect Storage Data

Note: You must complete this step on the hosts that will run messaging (**tc-mon**), applications (**tc-app**), and Playbooks (**tc-job**).

Follow these steps to configure the ThreatConnect storage data:

1. Run the following commands if you set `documentStorageType=LOCAL` in your `.env` file:

```
None
mkdir -p ${TC_DOC_STORAGE}
chown threatconnect:threatconnect -R ${TC_DOC_STORAGE}
```

2. Run the following additional command only if you are running rootless Podman:

```
None
docker unshare chown threatconnect:threatconnect -R ${TC_DOC_STORAGE}
```

Step 20: Configure TC Exchange Data

Note: This is an optional step that can be performed on the hosts that will run messaging (**tc-mon**), applications (**tc-app**), and Playbooks (**tc-job**).

Follow these steps to configure TC Exchange™ data:

1. Create the TC Exchange directory and sub-directories. For example, if ThreatConnect is to use `/threatconnect-data/exchange` as the TC Exchange directory, create the following subdirectories:

   Note: Execute this step only if these variables contain a value in your `.env` file.

```
None
mkdir -p ${TC_EXCHANGE}/jobs
mkdir ${TC_EXCHANGE}/programs
```

2. Ensure the TC Exchange directory and its subdirectories are owned by user `threatconnect` (or the name you gave to the **threatconnect** user):

   Note: Execute this step only if these variables contain a value in your `.env` file.

```
None
chown threatconnect:threatconnect -R ${TC_EXCHANGE}
```

3. Run the following additional command only if you are running rootless Podman:

   Note: Execute this step only if these variables contain a value in your `.env` file.

```
None
docker unshare chown threatconnect:threatconnect -R ${TC_EXCHANGE}
```

Step 21: Start ThreatConnect

Start each of the following services in the following order: OpenSearch → Postgres® → **tc-mon** → **tc-app** → **tc-job**. After starting each service, make sure to perform the following actions:

- Run `docker-compose logs --tail=10 --follow` to verify the service starts up before moving on to the next.
- Press **Ctrl+C** once the service is started.

If you encounter issues starting ThreatConnect, see the "Troubleshooting Notes" section for more information about known issues that may occur during this step.

Start OpenSearch

Note: You must complete this step on the host that will run OpenSearch.

Follow these steps to start OpenSearch:

1. Start OpenSearch:

```
None
docker-compose up -d opensearch
```

2. Set the OpenSearch password:

```
None
./set_opensearch_password.sh
```

Start Postgres

Note: You must complete this step on the host that will run Postgres.

Follow these steps to start Postgres:

1. Create the `postgres` data folder:

```
None
mkdir -p ${POSTGRES_DATA}
chown postgres:postgres -R ${POSTGRES_DATA}
```

2. Run the following additional commands only if you are running rootless Podman:

```
None
docker unshare chown postgres:postgres -R ${POSTGRES_DATA}

docker unshare chown postgres:postgres certs/postgres-privkey.pem
```

3. Start Postgres:

```
None
docker-compose up -d postgres
```

4. Load the database schema on the database server:

```
None
./load_schema.sh
```

Start Redis

Note: You must complete this step on the host that will run Redis.

Follow these steps to start Redis:

1. Create the `redis` data folder:

```
None
mkdir -p ${REDIS_DATA}
chown redis:redis -R ${REDIS_DATA}
```

2. Run the following additional command only if you are running rootless Podman:

```
None
docker unshare chown redis:redis -R ${REDIS_DATA}
```

3. Start Redis:

```
None
docker-compose up -d redis
```

4. Test Redis:
    a. Navigate to a command prompt in the `redis` container:

```
None
docker exec -ti redis bash
```

    b. Use either Option 1 or Option 2, depending on your configuration:
        - **Option 1**: Execute this command if you are using non-secure Redis (default):

```
None
redis-cli
```

        - **Option** 2: Execute this command if you are using Secure Redis, replacing `<REDIS_USER>` and `<REDIS_PASS>` with the appropriate values:

```
None
redis-cli --tls --cacert /certs/cert.pem --cert /certs/cert.pem --key
/certs/privkey.pem
auth <REDIS_USER> <REDIS_PASS>
```

    c. Ping the redis server (expected result: `PONG`):

```
None
ping
```

Start tc-mon

Note: You must complete this step on the host that will run the ThreatConnect messaging server.

Run the following command to start **tc-mon**:

```
None
docker-compose up -d tc-mon
```

Start tc-app

Note: You must complete this step on the host that will run the ThreatConnect application server.

Run the following command to start **tc-app**:

```
None
docker-compose up -d tc-app
```

Start tc-job

Note: You must complete this step on the host that will run the ThreatConnect Playbooks server.

Run the following command to start **tc-job**:

```
None
docker-compose up -d tc-job
```

Step 22: Log Into ThreatConnect

After all services are started successfully, log into ThreatConnect:

```
None
admin/password1
```

Step 23: Monitor ThreatConnect

Follow these steps to restart and monitor the ThreatConnect containers without an `.env` file in place:

1. Move your `.env` file to a secure location (e.g., a server where passwords are stored).
2. Docker Compose commands cannot be run without an `.env` file in place. Therefore, run the following command to check the status of the ThreatConnect containers. Note that the container names are in the first column.

```
None
docker ps --format "table {{.Names}}\t{{.Image}}\t{{.Status}}"
```

3. If you need to restart ThreatConnect containers without restoring the `.env` file, run the following commands one at a time, waiting for each container to start (about 60–90 seconds) before entering the next command:

```
None
docker restart tc-mon
docker restart tc-app
docker restart tc-job
```

4. If environment variables for `TC_APP_LOGS`, `TC_JOB_LOGS`, and `TC_MON_LOGS` are set, then run the following commands to tail monitor the ThreatConnect logs, respectively:

```
None
tail -f $TC_APP_LOGS/server.log $TC_APP_LOGS/tc.log
tail -f $TC_JOB_LOGS/server.log $TC_JOB_LOGS/tc.log
tail -f $TC_MON_LOGS/server.log $TC_MON_LOGS/tc.log
```

ThreatConnect logs can also be found in the following locations:

- Docker: `/var/lib/docker/volumes/`
- Podman: `$XDG_DATA_HOME/containers/storage/`

Step 24: Create Search Index

Follow these steps to create the search index:

1. Log into ThreatConnect with a System Administrator account.
2. From the **Settings** menu on the top navigation bar, select **System Settings**.
3. Set the values of **searchAdminPassword** and **searchAdminUsername** to `OPENSEARCH_PASSWORD` and `OPENSEARCH_USERNAME`, respectively, in the `.env` file.
4. Bounce ThreatConnect by running the following commands one at a time, waiting for each container to restart (about 60–90 seconds) before entering the next command:

```
None
docker restart tc-mon
docker restart tc-app
docker restart tc-job
```

Appendix

Document Storage Network Share

If you intend to run ThreatConnect in a multi-server configuration (i.e., a configuration where applications, messaging, and Playbooks all run on different hosts), you must set up a network shared folder for document storage that can be shared by all three hosts.

This example uses Network File System (NFS) Utils to set up a network shared folder on the host that will run the ThreatConnect messaging server (**tc-mon**). On each host, there must be a `threatconnect` user. If there is no such user, you must create one.

Follow these steps to set up a network shared folder for document storage:

1. Verify that user `threatconnect` exists:

```
None
grep threatconnect /etc/passwd
```

2. Set the ThreatConnect messaging host (replace `<tc-mon-host>` with the FQDN of the server that will run **tc-mon**):

```
None
yum install nfs-utils
echo "Domain = <tc-mon-host>" >> /etc/idmapd.conf
```

3. Run the following commands (replace the two IP addresses [`10.9.8.186` and `10.9.8.187`] with those of the servers that will run **tc-app** and **tc-job**):

```
None
echo "/threatconnect-data/storage
10.9.8.186(rw,sync,no_root_squash,no_subtree_check)" > /etc/exports
echo "/threatconnect-data/storage
10.9.8.187(rw,sync,no_root_squash,no_subtree_check)" > /etc/exports
```

4. Start the NFS and add a firewall rule:

```
None
systemctl start nfs-server
systemctl enable nfs-server
systemctl status nfs-server
firewall-cmd --add-service={nfs,nfs3,mountd,rpc-bind} --permanent
firewall-cmd --reload
```

5. Verify the NFS:

```
None
exportfs -v
```

6. Run the following commands on the ThreatConnect application host (replace the IP address [`10.9.8.185`] with that of the server that will run **tc-mon**):

```
None
yum install nfs-utils
showmount -e 10.9.8.185
mkdir -p /threatconnect-data/storage
mount 10.9.8.185:/threatconnect-data/storage /threatconnect-data/storage
ls -l /threatconnect-data/storage
```

7. Run the following commands on the ThreatConnect Playbooks host (replace the IP address [`10.9.8.185`] with that of the server that will run **tc-mon**):

```
None
yum install nfs-utils
showmount -e 10.9.8.185
mkdir -p /threatconnect-data/storage
mount 10.9.8.185:/threatconnect-data/storage /threatconnect-data/storage
ls -l /threatconnect-data/storage
```

Troubleshooting Notes

If you forget to put a value into the `.env` file or put a wrong value in the `.env` file and then started the containers, use one the following methods as a fix:

Method 1: (Recommended) Pause the installation process and remove one or more containers by running the following command:

```
None
docker rm -f tc-mon tc-app tc-job
```

The next time you start the containers, the corrected environment variable will get processed.

Method 2: Pause the installation process and remove all containers by running the following command:

```
None
docker-compose down
```

If you receive the following error the first time you execute `docker-compose up -d`, you must add more IP address space to Docker:

```
None
could not find an available, non-overlapping IPv4 address pool among the
defaults to assign to the network
```

To add more IP address space to Docker, add an IP address block that applies to your environment to `/etc/docker/daemon.json`:

```
None
{
    ...
    "default-address-pools":[
        {"base":"172.20.0.0/16","size":24},
        {"base":"172.21.0.0/16","size":24}
    ]
```

```
}
```

If you experience difficulties connecting to OpenSearch, try connecting to it with curl as follows:

```
None
curl -k -s -u $OPENSEARCH_USERNAME:$OPENSEARCH_PASSWORD https://localhost:9200/
```

If your attempts to connect to OpenSearch return an error saying that access is unauthorized, try running the following command to reset the password:

Note: You may need to update the value of the **searchAdminPassword** system setting on the **System Settings** screen once ThreatConnect is running. Note that you will need to restart ThreatConnect after changing the **searchAdminPassword** system setting in order for the change to take effect.

```
None
./set_opensearch_password.sh
```

Enabling SAML

Follow these steps to enable the Security Assertion Markup Language™ (SAML) configuration on ThreatConnect:

1. In the `.env` file associated with the containerized deployment of ThreatConnect, update each variable in the "SAML Settings" section with the appropriate value. For descriptions of the values that you must provide in the `.env` file, reference the comments in the "SAML Settings" section of that file.

2. Add the following `.pem` files to the `certs` folder:

    - `certs/saml_privkey.pem`
    - `certs/saml_fullchain.pem`
    - `certs/saml_host.pem`

Note: The `saml_fullchain.pem` and `saml_privkey.pem` files can have the same content as the `fullchain.pem` and `privkey.pem` files. The `saml_host.pem` file must contain the Identity Provider (IDP) certificate.

Certificate Authority Signed Certificate

If you do not have a server or certificate authority (CA) certificate, follow these steps to generate a certificate request:

1. Create the `privkey.pem`:

```
None
mkdir certs && cd certs
openssl genrsa -out privkey.pem 4096
```

2. Create the Certificate Signing Request Configuration File (`certreq.cnf`):

   Note: Replace all values surrounded with `<>` with real values.

```
None
[req]
distinguished_name = req_distinguished_name
req_extensions = req_ext
prompt = no

[req_distinguished_name]
CN = <The fully qualified domain name (FQDN) or IP address of the server>
OU = <The name of the organizational unit>
O = <The name of the organization>
L = <The location or city in which the organization resides>
S = <The state or province in which the organization resides>

[req_ext]
subjectAltName = @alt_names

[alt_names]
IP.1 = <The IP address of the server>
DNS.1 = <The FQDN of the server>
```

3. Create the Certificate Signing Request (`certreq.csr`):

```
None
openssl req -new -sha256 -key privkey.pem -out certreq.csr -config certreq.cnf
```

4. Submit `certreq.csr` to your CA, who will return a signed certificate bundle. If possible, specify PEM as the return format.
5. If your CA is not public, then update `CUSTOM_CA` in your `.env` file as follows:

```
None
CUSTOM_CA=true
```

Self-Signed Certificate

If you do not have a server or CA certificate, follow these steps to generate self-signed certificates:

1. Create a certificate authority (replace the `<country>`, `<state>`, `<city>`, `<company>`, and `<department>` placeholder values):

```
None
mkdir certs && cd certs
openssl genrsa -out ca-key.pem 4096
openssl req -new -x509 -sha256 -key ca-key.pem \
    -subj "/C=<country>/ST=<state>/L=<city>/O=<company>/OU=<department>/CN=My
Root Authority" \
    -out ca.pem -days 3650
```

2. Create a private key:

```
None
openssl genrsa -out privkey.pem 4096
```

3. Create a certificate signing request (replace the `<country>`, `<state>`, `<city>`, `<company>`, `<department>`, and `<FQDN/IP of server>` placeholder values):

```
None
openssl req -new -sha256 -key privkey.pem \
    -subj
"/C=<country>/ST=<state>/L=<city>/O=<company>/OU=<department>/CN=<FQDN/IP of
server>" \
    -out <FQDN/IP of Server>.csr
```

4. Create a new file for alternate names in `alt-names.ext` (replace the `<FQDN/IP of server>` placeholder value):

```
None
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
```

```
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
subjectAltName = @alt_names

[alt_names]
DNS.1 = <FQDN/IP of server>
```

5. Create a certificate signed by your CA (replace the `<FQDN/IP of server>` placeholder value):

```
None
openssl x509 -req -in <FQDN/IP of server>.csr -CA ca.pem \
    -CAkey ca-key.pem -CAcreateserial \
    -out server.pem -days 398 -sha256 \
    -extfile alt-names.ext
```

6. Update `CUSTOM_CA` in your `.env` file as follows:

```
None
CUSTOM_CA=true
```