Micro Focus Transformation Hub

Software Version: 3.2.0

Deployment Guide

Document Release Date: May 29, 2020
Software Release Date: April 30, 2020

Legal Notices

Copyright Notice

Trademark Notices

Support

Contact Information

Phone	A list of phone numbers is available on the Technical Support Page: https://softwaresupport.softwaregrp.com/support-contact-information
Support Web Site	https://softwaresupport.softwaregrp.com/
ArcSight Product Documentation	https://community.softwaregrp.com/t5/ArcSight-Product-Documentation/ct-p/productdocs

Contents

Chapter 1: Transformation Hub Deployment Guide

The Transformation Hub Deployment Guide describes the process of deploying and configuring Transformation Hub. The instructions contained here presume you have already reviewed the ***CDF Planning Guide***, which describes the process of planning your CDF and associated product deployment, and is a prerequisite to the actual deployment process.

For further information on Transformation Hub, consult the following documentation:

- Transformation Hub Administrator's Guide: This guide describes the operations and maintenance of Transformation Hub in detail.
- Transformation Hub Release Notes: This document includes important information about the current release of Transformation Hub.

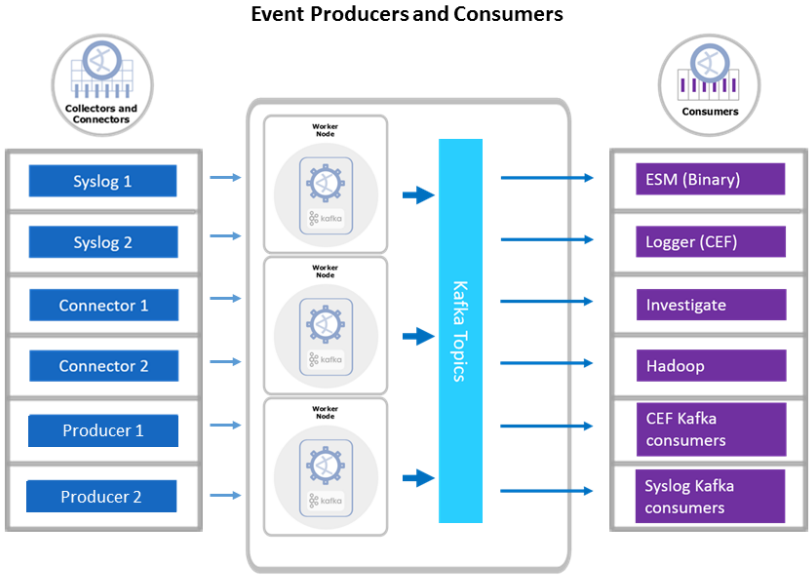
Transformation Hub documentation is available for download from the Micro Focus support community.

Chapter 2: Transformation Hub

Transformation Hub is the high-performance message bus for ArcSight security, network, flows, application, and other events. It can queue, transform, and route security events to other ArcSight or third party software. This Kafka-based platform allows ArcSight components like Logger, ESM, and Investigate to receive the event stream, while smoothing event spikes, and functioning as an extended cache.

Transformation Hub ingests, enriches, normalizes, and then routes event data from data producers to connections between existing data lakes, analytics platforms, and other security technologies and the multiple systems within the Security Operations Center (SOC). Transformation Hub can seamlessly broker data from any source and to any destination. Its architecture is based on Apache Kafka and it supports native Hadoop Distributed File System (HDFS) capabilities, enabling both the ArcSight Logger and ArcSight Investigate technologies to push event data to HDFS for long-term, low-cost storage. ArcSight Investigate is integrated with the Transformation Hub for raw CEF events. ArcSight ESM receives binary event data for dashboarding and further correlation.

This architecture reduces the overall ArcSight infrastructure footprint, scales event ingestion using built-in Kafka and Kubernetes capabilities, and greatly simplifies upgrades to newer Transformation Hub releases. It also positions the platform to support an analytics streaming plug-in framework, with automated machine learning and artificial intelligence engines for data source on-boarding, event enrichment, and entities and actors detection and attribution.



Secure Open Data Platform (SODP)

The Secure Open Data Platform (SODP) centralizes management, monitoring and configuration of the entire data-centric ecosystem using an open architecture. It is configured and monitored through the ArcSight Management Center (ArcMC) user interface.

SODP comprises the following ArcSight products:

- Transformation Hub
- Management Center (ArcMC)
- Smart Connectors

Management Center (ArcMC)

The ArcMC management console enables central administration of SODP infrastructure, including users, configurations, backups, updates, and health monitoring to connectors and storage instances.

ArcMC's Topology view shows administrators event flow through the entire environment, including a specific focus on monitoring endpoint device log delivery.

SmartConnectors

SmartConnectors serve to collect, parse, normalize and categorize log data. Connectors are available for forwarding events between and from Micro Focus ArcSight systems like Transformation Hub and ESM, enabling the creation of multi-tier monitoring and logging architectures for large organizations and for Managed Service Providers.

The connector framework on which all SmartConnectors are built offers advanced features that ensures the reliability, completeness, and security of log collection, as well as optimization of network usage. Those features include: throttling, bandwidth management, caching, state persistence, filtering, encryption and event enrichment. The granular normalization of log data allows for the deterministic correlation that detects the latest threats including Advanced Persistent Threats and prepares data to be fed into machine learning models.

SmartConnector technology supports over 400 different device types, leveraging ArcSight's industry-standard Common Event Format (CEF) for both Micro Focus and certified device vendors. This partner ecosystem keeps growing not only with the number of supported devices but also with the level of native adoption of CEF from device vendors.

Chapter 3: Deployment Planning and Preparation

Before proceeding with your deployment, it is assumed that you have already planned and provisioned your network, storage, and the cluster of host systems based on requirements described in the ***CDF Planning Guide.***, which is available from the Micro Focus software support community.

The complete process of deploying Transformation Hub comprises the following high-level steps:

1. **Configure and Install the CDF Installer:** The CDF Installer installs the container management infrastructure. Containerized applications, like the Transformation Hub, run in this environment. Depending on your environment, you may need to adjust the default installation parameter values.
2. **Configure and Deploy the Kubernetes Cluster:** Configure and deploy the master and worker nodes, NFS storage, network connectivity, and other infrastructure requirements.
3. **Configure and Deploy Transformation Hub:** Using the CDF Installer wizard, configure and deploy Transformation Hub to run in the CDF-managed Kubernetes cluster. Then, start the Transformation Hub services.
4. **Manage Transformation Hub from the Management Center:** Configure the Management Center (ArcMC) to recognize and manage the Transformation Hub cluster.
5. **Integrate Transformation Hub with Other ArcSight Products:** Configure your SmartConnectors and Collectors as producers of events into Transformation Hub, as well as configure event Consumers such as Logger and ESM.

Note: The installation process will validate the Transformation Hub infrastructure environment before performing the installation, as well as after the installation has completed.

For detailed instructions on the operation and management of Transformation Hub after initial deployment, see the Transformation Hub Administration Guide, available from the Micro Focus support community.

Gather Required Information

During the process described in the ***CDF Planning Guide***, you made configuration decisions about your environment, platforms, network, and storage. You will need this information handy now in order to complete the installation of CDF and Transformation Hub.

- **Master and Worker Node Info:** Ensure you have relevant configuration information of the master and worker Nodes, including:
  - Credentials for the root or `sudo` (non-root) user that will be used to run the deployment
  - IP Address and FQDN for every host system in the cluster
  - NFS Server IP Address and FQDN
  - Virtual IP (only required if master nodes are configured for high-availability)
- **License Keys:** Ensure you have all required Micro Focus License keys for the software being installed.

- **Security Mode:** Determine security settings for communication between ArcSight components.
- **Infrastructure:** Validate, and if necessary, remediate Transformation Hub infrastructure prerequisites.
  - Review, analyze and adjust your Transformation Hub infrastructure configuration properties to meet throughput expectations (for example, Events per Second processing rates).
  - Copy the CDF Deployment Disk Sizing Calculator spreadsheet (available from the Micro Focus support community) and edit its contents to determine your disk storage requirements and apply these during the pre-deployment configuration process.
- **Download Access:** Finally, ensure you have access to the Micro Focus software download location. You will download installation packages to the Initial Master Node.

Secure Communication Between Micro Focus Components

Determine which security mode you will use for communication between your infrastructure components. The security mode of connected producers and consumers must be the same across all components.

> **Note:** The secure communication described here applies only in the context of the components that relate to the Micro Focus container-based application you are using, which is specified in that application's documentation.

Set up the other Micro Focus components with the security mode you intend to use ***before*** connecting them to the Transformation Hub. Changing the security mode after deployment will require system downtime. (If you do need to change a component's security mode after deployment, refer to the appropriate Administrator's Guide for the affected component.)

The following table lists Micro Focus products, preparations needed for secure communication with components, ports and security modes, and documentation for more information on the product. (**Note:** product documentation is available for download from the Micro Focus support community.)

Product	Preparations needed...	Ports	Supported security modes	More information
Management Center (ArcMC) version 2.93 or later	ArcMC can be installed before or after Transformation Hub installation.	443, 38080	• TLS • FIPS • Client Authentication	ArcMC Administrator's Guide
SmartConnectors and Collectors	SmartConnectors and ArcMC onboard connectors can be installed and running prior to installing Transformation Hub, or installed after the Transformation Hub has been deployed. • FIPS mode setup is not supported between SmartConnector v7.5 and Transformation Hub. Only TLS and Client Authentication are supported. • FIPS mode is supported between Connectors v7.6 and later and Transformation Hub	9093	• TLS • FIPS (SC 7.6+ only) • Client Authentication	SmartConnector User Guide, ArcMC Administrator's Guide
ArcSight ESM	ESM can be installed and run either prior to installing Transformation Hub, or after. Note that changing ESM from a FIPS mode to TLS (default) mode requires a reinstallation of ESM. Refer to the ESM documentation for more information.	9093	• TLS • FIPS • Client Authentication	ESM Administrator's Guide
ArcSight Logger	Logger can be installed and run either prior to or after installing Transformation Hub.	9093	• TLS • FIPS • Client Authentication	Logger Administrator's Guide

Leader Acknowledgement ("ACK") and TLS Enablement: In general, enabling leader ACKs and TLS results in significantly lower throughput rates, but greater fidelity in ensuring events are received by Subscribers. For more information on leader acknowledgements and TLS enablement and their effects on processing throughput, refer to the Kafka documentation which explains these features.

Download Installation Packages

Download the installation packages for both the CDF Installer and the Transformation Hub to your Initial Master Node from the Micro Focus Entitlement Portal. After download, validate the digital signature of each file, and then unarchive them.

For a complete list of files and file versions to be downloaded, consult the Transformation Hub Release Notes from the Micro Focus support community.

To access the ArcSight software in the Micro Focus ArcSight Entitlement Portal, use your Micro Focus credentials which will be authenticated before allowing the download.

Navigate to the version of Transformation Hub you wish to install and download the installation packages for the CDF Installer and the Transformation Hub to a directory such as `/opt/download/`.

About the Micro Focus Entitlement Portal

The Micro Focus Entitlement Portal contains ArcSight installation and other product-related materials. This is the only location where you can download the full set of materials needed for Transformation Hub installation.

Some downloaded software will be in compressed format, and in addition, and will have associated signature files (.md5 or .sig) that are used to ensure that the downloaded software is authentic.

Validating Downloaded File Signatures

Micro Focus provides a digital public key that is used to verify the software you downloaded from the Micro Focus software entitlement site is indeed from Micro Focus and has not been tampered with by a third party. Visit the Micro Focus Code Signing site for information and instructions on validating the downloaded software.

To verify that the downloaded files are authentic by comparing MD5 file signatures, perform the following steps as the root user on the Initial Master Node for downloaded files.

```
cd /opt/download/

md5sum <File Name>

cat <File Name>.md5
```

Outputs from each set of compressed installation packages should match their corresponding MD5 signatures. If they do not match, try the download again or contact Micro Focus Customer Support.

Unarchive Installation Packages

Run the following commands to unarchive your installation packages.

```
cd /opt/download

unzip cdf-2020.02.xxxx.zip

tar -xvf transformationhub-3.2.0.xxxx.tar
```

Where `xxxx` is the build number of the file.

Note: Do not uncompress the `arcsight-installer-metadata` file.

Resulting Directories

After the successful validation and decompression of the installation packages, the following directories and files will be located on your Initial Master Node and contain the installation materials:

`opt/download/cdf-2020.02.xxxx/`

`opt/download/transformationhub-3.2.0.xxxx`

`opt/download/arcsight-installer-metadata.2.2.0.xxxx.tar`

Chapter 4: Installation and Deployment

Once the installation packages have been downloaded, validated, and uncompressed, you are ready to proceed with installation and deployment. In outline, the complete installation and deployment of Transformation Hub consists of these steps, which must be performed in order:

1. Configure and Deploy the CDF Installer
2. Configure and Deploy Kubernetes (k8s)
3. Upload Core Images to the Docker Registry
4. Configure and Deploy Transformation Hub

Each of these steps is explained in detail in this chapter.

Configure and Install the CDF Installer

Once the installation packages have already been downloaded, validated and uncompressed in the download folder, you are ready to configure and install the CDF Installer.

You can install the CDF Installer as a root user, or, optionally, as a **sudo** user. However, if you choose to install as a **sudo** user, you must first configure installation permissions from the root user. For more information on providing permissions for the **sudo** user, see Appendix B of the ***CDF Planning Guide.***

To configure and install the CDF Installer:

1. Log in to one of the local master nodes where you downloaded and extracted the installation files as the root user. (In this document, the selected master node will be referred to as the Initial Master Node. Installations will be initiated from the Initial Master Node.)
2. Install the CDF Installer on the Initial Master Node with the following commands.

Note:

```
cd /opt/download/<unzipped CDF directory>
```

```
./install -m <path_to_a_metadata_file> --k8s-home <path_to_installation_
directory> --docker-http-proxy <your_docker_http_proxy_value> --docker-https-
proxy <your_docker_https_proxy_value> --docker-no-proxy <your_docker_no_
proxy_value> --nfs-server <your_nfs_server_FQDN or IP Address> --nfs-folder
<itom_volume_folder> --ha-virtual-ip <your_HA_ip> --tmp-folder <your_temp_
folder>
```

Note: In the above command, the *italicized* Docker parameters are optional, based on your network environment.

You are prompted for a password, which will be used to log in to the CDF installer portal.

Example:

```
cd /opt/arcsight/download/cdf-2020.02.xxxx
```

```
./install -m /tmp/arcsight-installer-metadata-2.2.0.xxx.tar --k8s-home
/opt/arcsight/kubernetes --docker-http-proxy "http://web-
proxy.example.com:8080" --docker-https-proxy "http://web-
proxy.example.com:8080" --docker-no-proxy "localhost,127.0.0.1,my-vmenv-
node1,my-vmenv-node1.example.com,example.com,216.3.128.12" --nfs-server
pueas-vmenv-nfs.swinfra.net --nfs-folder /opt/nfs/volumes/itom/itom_vol --ha-
virtual-ip 216.3.128.12 --tmp-folder /opt/tmp
```

Note: In the above command, the ***italicized*** Docker parameters are optional, based on your network environment.

You may need to configure some additional parameters, depending on your organization's OS, network, and storage configurations.

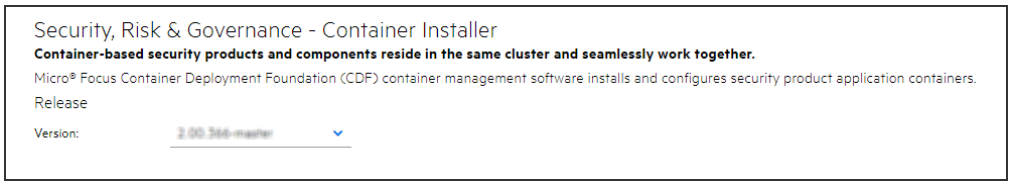
Note: For NFS parameter definitions, refer to the CDF Planning Guide section "Configure an NFS Server environment." For a description of valid CDF Installer command line parameters, see Installer CLI Commands.

Once the CDF Installer is configured and installed, you can use it to deploy one or more products or components into the cluster.

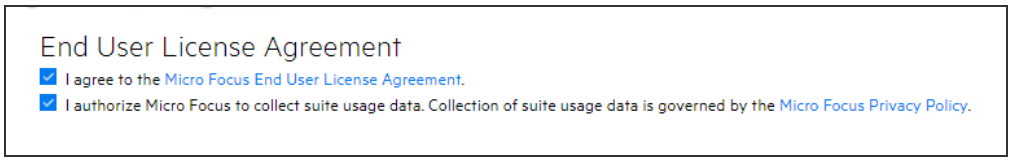
Configure and Deploy the Kubernetes Cluster

After you install the CDF Installer, complete the following steps to deploy your Kubernetes cluster.

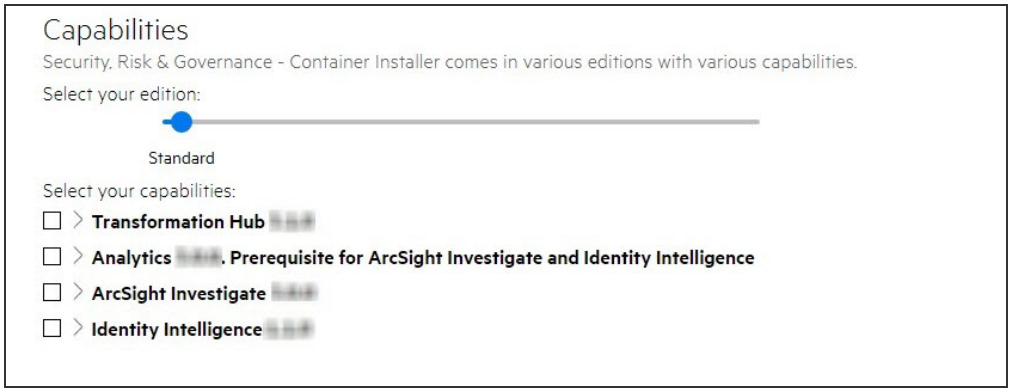
1. Browse to the Initial Master Node at:
   `https://{master_FQDN or IP}:3000`
2. Log in using ***admin*** userid and the password you specified during the platform installation. (This URL is displayed at the successful completion of the CDF installation shown earlier.)
3. On the **Security Risk and Governance - Container Installer** page, choose the CDF base product metadata version. Then, click **Next**.



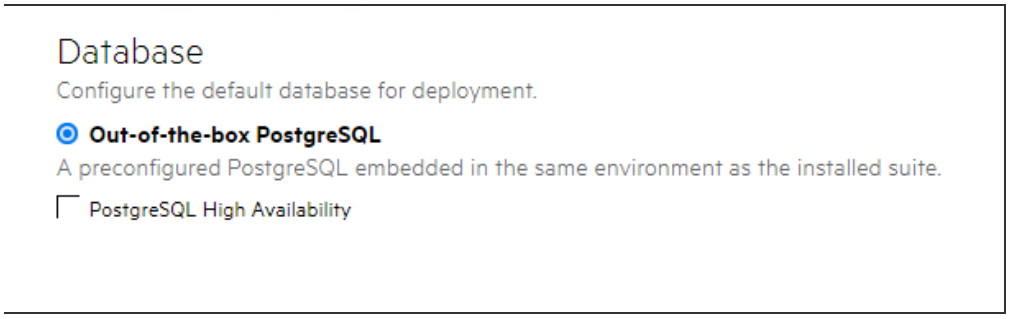
4. On the **End User License Agreement** page, review the EULA and select the ***'I agree...'*** checkbox. You may optionally choose to have suite utilization information passed to Micro Focus. Then, click **Next**.



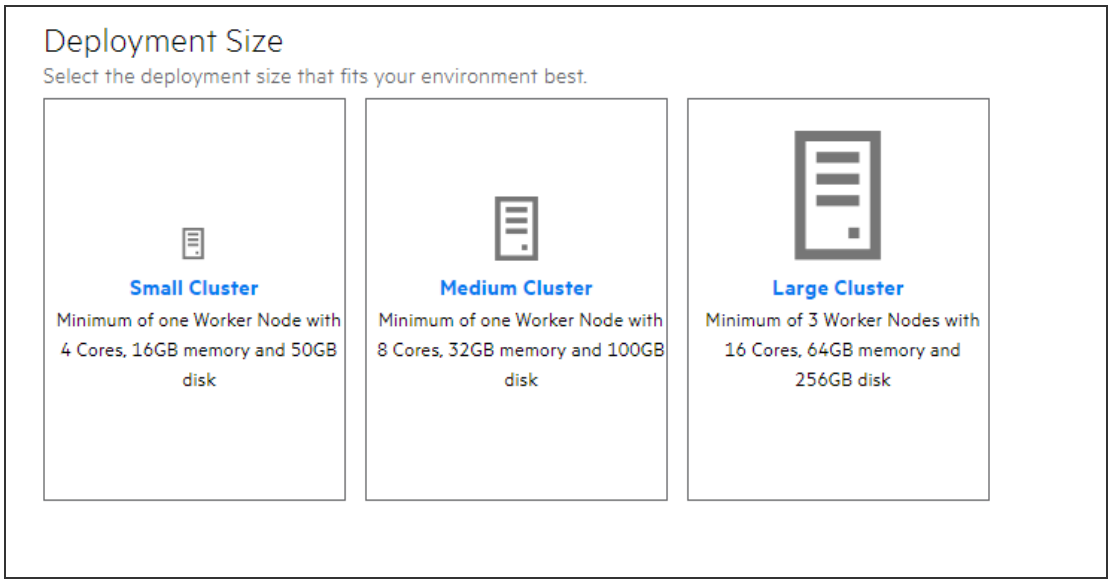
5. On the **Capabilities** page, choose the capabilities and/or products to be installed. To install Transformation Hub as a standalone install, select it. (Note that other products may require Transformation Hub or other capabilities as prerequisites. Such requirements will be noted in the pull-down text associated with the capability.) To show additional infomation associated with the product, click the **>** (greater than) arrow. Then, click **Next.**



6. On the **Database** page, make sure the **PostgreSQL High Availability** box is *deselected*. This database is not used by capabilities in SODP.



7. Click **Next.**
8. On the **Deployment Size** page, choose a size for your deployment based on your planned implementation.



- **Small Cluster:** Minimum of one worker node deployed (each node should have 4 cores, 16 GB memory, 50 GB disk)
- **Medium Cluster:** Minimum of 1 worker node deployed (each node should have 8 cores, 32 GB memory, 100 GB disk)
- **Large Cluster:** Minimum of 3 worker nodes deployed (each node should have 16 cores,64 GB memory, 256 GB disk)

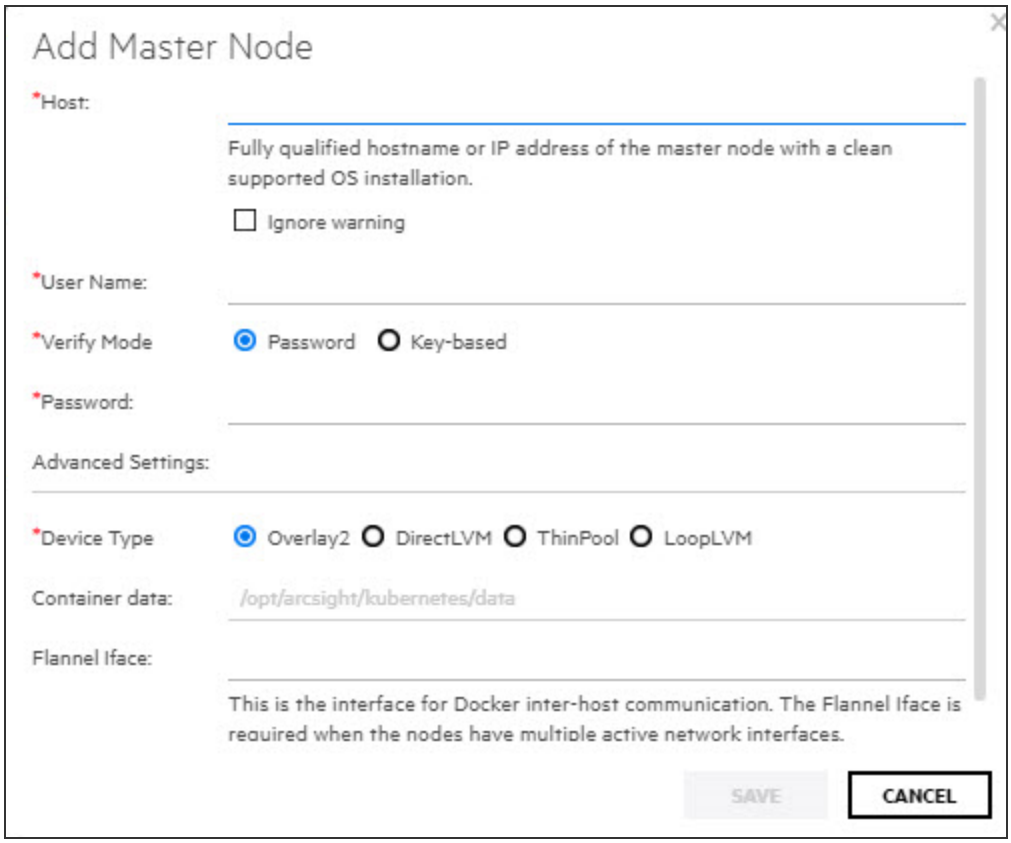
Note: The installation will not proceed until the minimal hardware requirements for the deployment are met.

Additional worker nodes, with each running on their own host system, can be configured in subsequent steps.

Select your appropriate deployment size, and then click **Next**.

8. For High Availability clusters, the installer prompts to add additional master nodes depending on your selected deployment size. On the **Add Master Node** page, specify the details of your first master

node and then click **Save**. Repeat for any additional master nodes.

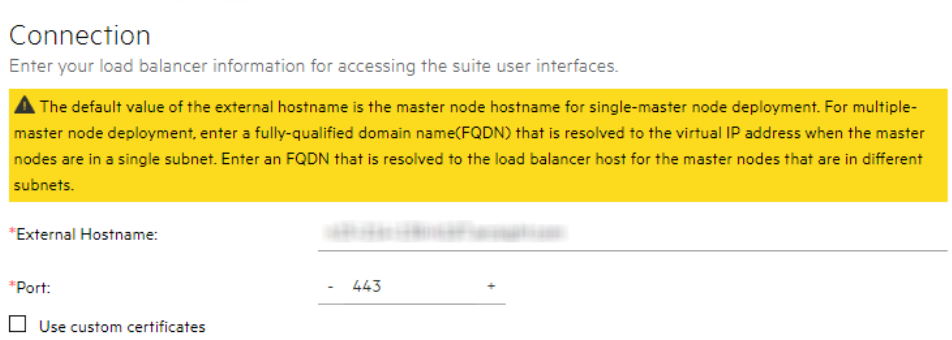


Master node parameters include:

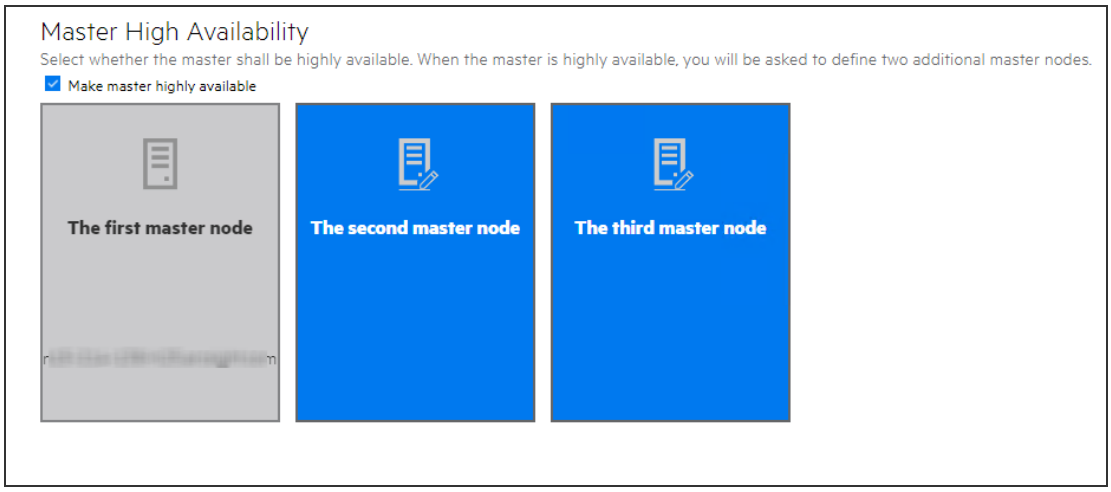
- **Host:** FQDN (only) of Node you are adding.
- **Ignore Warnings:** If selected, the installer will ignore any warnings that occur during the pre-checks on the server. If deselected, the add node process will stop and a window will display any warning messages. We recommend that you start with **Ignore Warnings** deselected in order to view any warnings displayed. You may then evaluate whether to ignore or rectify any warnings, clear the warning dialog, and then click Save again with the box selected to avoid stopping.
- **User Name:** Root or `sudo` user name.
- **Verify Mode:** Choose the verification mode as *Password* or *Key-based*, and then either enter your password or upload a private key file. If you choose **Key-based**, you must first enter a username and then upload a private key file when connecting the node with a private key file.
- **Device Type:** Select a device type for the master node from one of the following options.
  - Overlay 2: For production, Overlay 2 is recommended.
  - **Thinpool Device:** (optional) Enter the Thinpool Device path, that you configured for the master node (if any). For example: `/dev/mapper/docker-thinpool`. You must have already set up the Docker thin pool for all cluster nodes that need to use thinpools, as described in the *CDF Planning Guide*.
- **Container data:**

- **flannel IFace:** (optional) Enter the flannel IFace value if the master node has more than one network adapter. This must be a single IPv4 address (or name of the existing interface) and will be used for Docker inter-host communication.

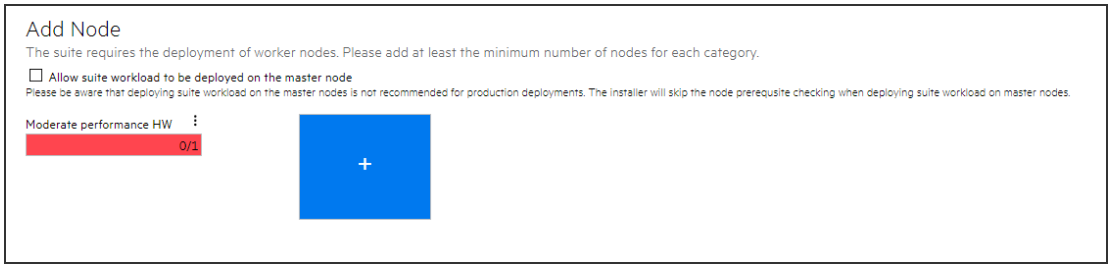
9. On the **Connection** page, an external hostname is automatically populated. This is resolved from the Virtual IP (VIP) specified earlier during the install of CDF (`--ha-virtual-ip parameter`), or the master node hostname if the `--ha-virtual-ip` parameter was not specified during CDF installation. Confirm the VIP is correct and then click **Next**.



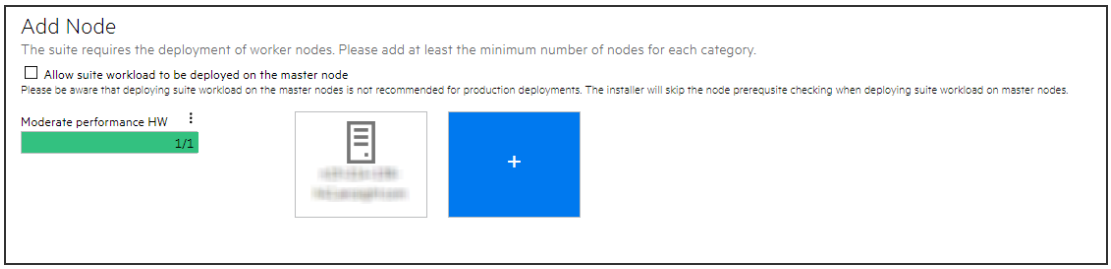
10. On the **Master High Availabilty** page , if high availabilty (HA) is desired, select **Make master highly available** and add 2 additional Master nodes. (CDF requires 3 Master nodes to support high availability.) When complete, or if HA is not desired, click **Next**.



11. On the **Add Node** page, add the first worker node as required for your deployment by clicking on the **+** (Add) symbol in the box to the right. The current number of nodes is initially shown in red.

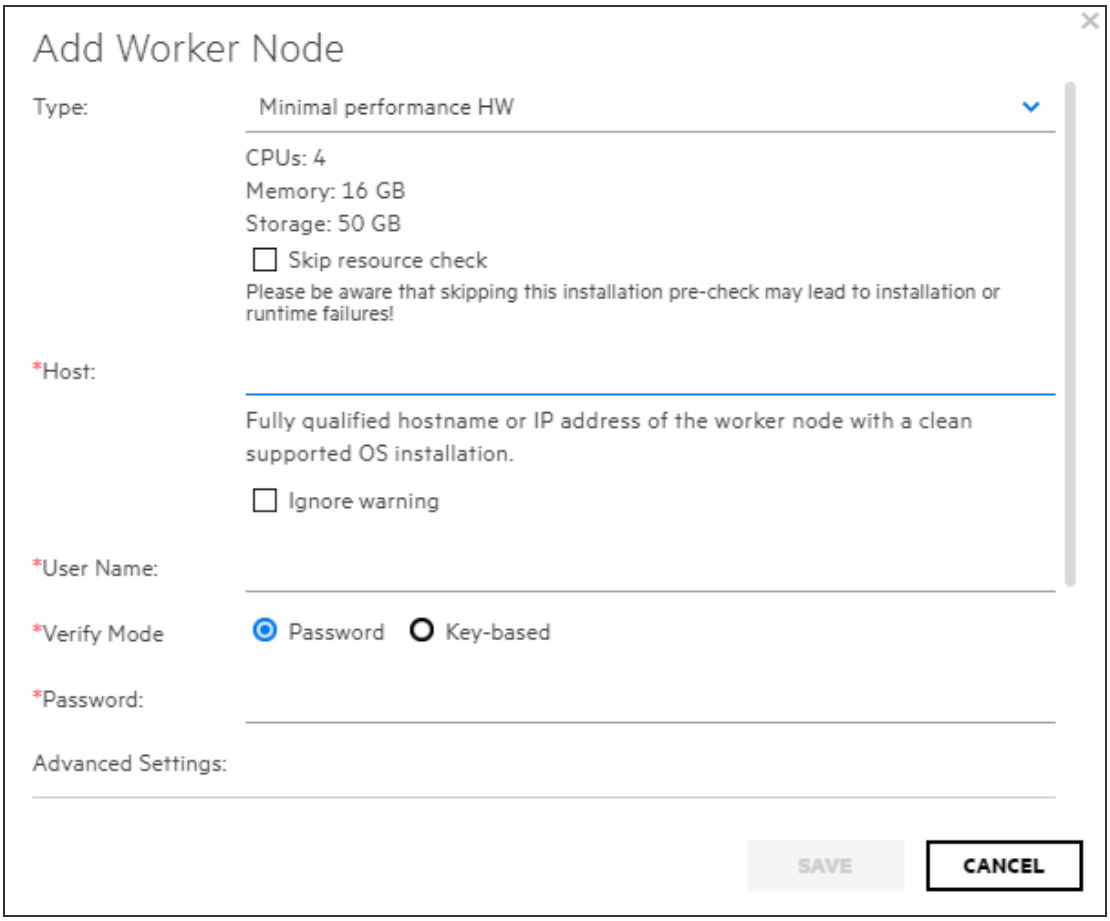


As you add worker nodes, each Node is then verified for system requirements. The node count progress bar on the **Add Node** page will progressively show the current number of verified worker nodes you have configured. This progress will continue until the necessary count is met. The progress bar will turn from red to green, which indicates you have reached the minimum number of worker nodes as shown selected in Step 7, above. You may add more Nodes than the minimum number.



Note: Check the **Allow suite workload to be deployed on the master node** to combine master/worker functionality on the same node (Not recommended for production).

On the **Add Worker Node** dialog, enter the required configuration information for the worker node, and then click **Save**. Repeat this process for each of the worker nodes you wish to add.



Worker node parameters include:

- **Type:** Default is based on the deployment size you selected earlier, and shows minimum system requirements in terms of CPU, memory, and storage.
- **Skip Resource Check:** If your worker node does not meet minimum requirements, select **Skip resource check** to bypass minimum node requirement verification. (The progress bar on the **Add Node** page will still show the total of added worker nodes in green, but reflects that the resources of one or more of these have not been verified for minumum requirements.)
- **Host:** FQDN (only) of node you are adding.

Warning: When adding any worker node for Transformation Hub workload, on the **Add Node** page, **always** use the FQDN to specify the Node. **Do not use the IP address.**

- **Ignore Warnings:** If selected, the installer will ignore any warnings that occur during the pre-checks on the server. If deselected, the add node process will stop and a window will display any warning messages. You may wish to start with this deselected in order to view any warnings displayed. You may then evaluate whether to ignore or rectify any warnings, and then run the deployment again with the box selected to avoid stopping.
- **User Name:** Root or `sudo` user name.
- **Verify Mode:** Select a verification credential type: Password or Key-based. Then enter the actual credential.

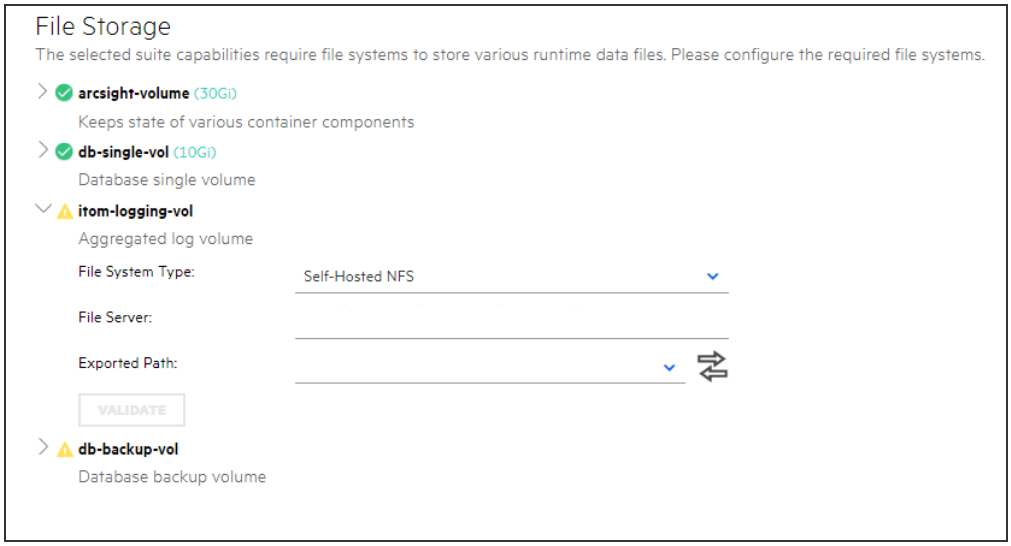
Once all the required worker nodes have been added, click **Next**.

12. On the **File Storage** page, configure your NFS volumes.

(For NFS parameter definitions, refer to the CDF Planning Guide section "Configure an NFS Server environment".) For each NFS volume, do the following:

- In **File Server**, enter the IP address or FQDN for the NFS server.
- On the **Exported Path** drop-down, select the appropriate volume.
- Click **Validate**.

Note: All volumes must validate successfully to continue with the installation.



Note: A *Self-hosted NFS* refers to the external NFS that you prepared when you configured an NFS server environment, as outlined in the CDF Planning Guide. Always choose this value for **File System Type.**

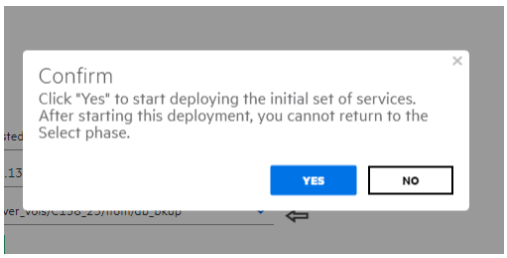
The following volumes must be available on your NFS server.

CDF NFS Volume claim	Your NFS volume
`arcsight-volume`	`<NFS_ROOT_FOLDER>/arcsight`
`itom-vol-claim`	`<NFS_ROOT_FOLDER>/itom/itom_vol`
`db-single-vol`	`<NFS_ROOT_FOLDER>/itom/db`
`itom-logging-vol`	`<NFS_ROOT_FOLDER>/itom/logging`
`db-backup-vol`	`<NFS_ROOT_FOLDER>/itom/db_backup`

13. Click **Next**.

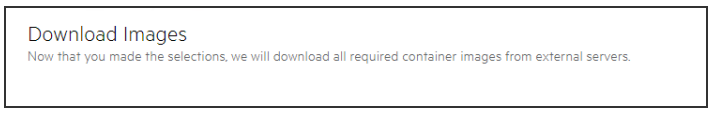
Warning: After you click **Next**, the infrastructure implementation will be deployed. ***Please ensure that your infrastructure choices are adequate to your needs.*** An incorrect or insufficient configuration may require a reinstall of all capabilities.

14. On the **Confirm** dialog, click **Yes** to start deploying master and worker nodes.



Download Transformation Hub and Core Images to the Local Docker Registry

By this point, the Transformation Hub images to be installed have already been downloaded from the Micro Focus software site, validated and uncompressed.



None of the files should require downloading at this point, so on the **Download Images** page, click **Next** to skip this step.

Uploading Images

The **Check Image Availability** page lists the images which have currently been loaded into the local Docker Registry from the originally-downloaded set of images. For a first install, it is expected that no images have already been uploaded yet. You will upload the images at this step.

To upload the images to the local Docker Registry:

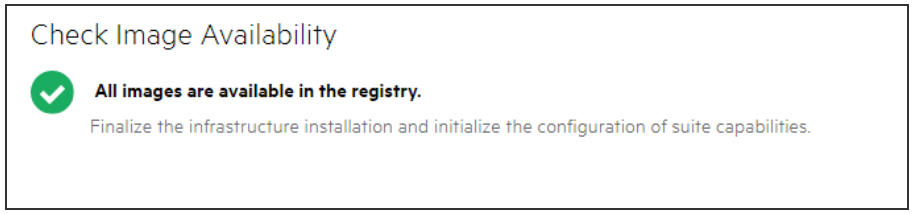
1. Log on to the Initial Master Node in a terminal session as the root or sudo user
2. Run the following commands to upload the Transformation Hub images to the local Docker Registry:

```
cd ${K8S_HOME}/scripts

./uploadimages.sh –u registry-admin -d /opt/download/transformationhub-3.2.0.xxx
```

Note: Prior to running the image upload process by script, you will be prompted for the administrator password previously specified in "Configure and Install the CDF Installer" on page 15.

Verify Prerequisite and Installation Images

The pre-deployment validation process will verify that all environment prerequisites have been met prior to installing the Transformation Hub.



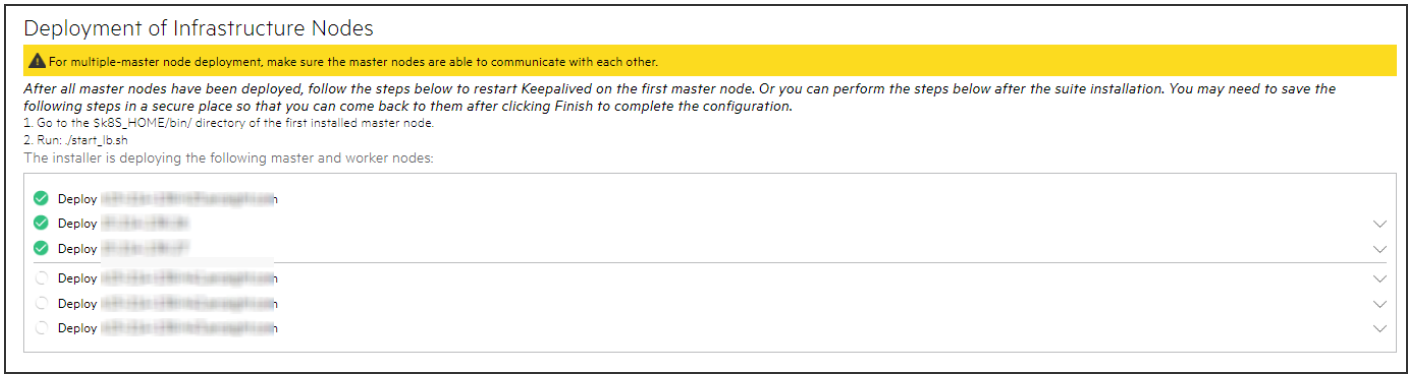
To verify completion of the upload of all images, return to the CDF Management Portal's Check Availability page and click **Check Image Availability Again**. All required component uploads are complete when the message displayed is: ***All images are available in the registry.***

Once verified, click **Next**.

Deploy Node Infrastructure and Services

Node Infrastructure

After the images are verified and you click **Next**, the node infrastructure is deployed. The **Deployment of Infrastructure Nodes** page will display progress.



Please be patient. Wait for all master and worker nodes to be properly deployed (showing a green check icon). Depending on the speed of your network and node servers, this can take up to 15 minutes to complete.

> **Note:** Clicking the **Retry** button will trigger additional communication with a problematic node, until the button converts to a spinning progress wheel. This indicates that the node deployment process is being started again. Until this occurs, refrain from clicking **Retry** again.

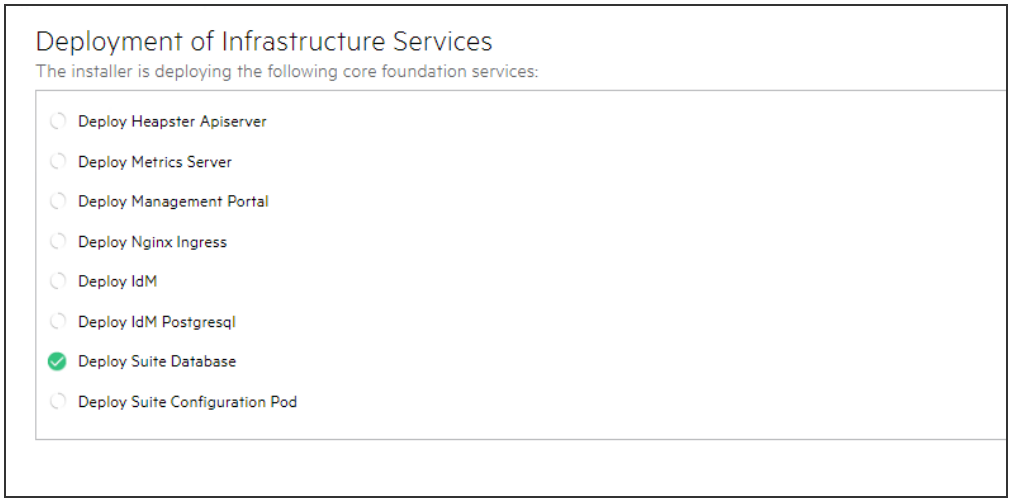
Monitoring Progress: You can monitor deployment progress on a node in the following ways:

- During installation, check the log on the node of interest, in `/tmp/install<timestamp>.log.` Run the command:
  `tail -f <logfilename>`
  - After installation has finished, the logs are copied to `${K8S_HOME}/log/scripts/install`
- You can watch the status of deployment pods with the command:
  `kubectl get pods --namespace core -o wide | grep -i cdf-add-node`

> **Note:** the Initial Master Node is not reflected by its own `cdf-add-node` pod.

Infrastructure Services

Infrastructure services are then deployed. The **Deployment of Infrastructure Services** page shows the deployment progress.



Please be patient. Wait for all services to be properly deployed (showing a green check icon). This can take up to 15 minutes to complete. Should any service show a red icon, then this process may have timed out. If this occurs, click the **Retry** icon to retry the deployment for that service.

To monitor progress as pods are being deployed, on the Initial Master Node, run the command:

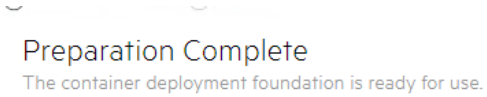
```
watch 'kubectl get pods --all-namespaces'
```

Note: If you try to access the CDF Management Portal Web UI (port 3000) too quickly after this part of the install has finished, you might receive a 'Bad Gateway' error. Allow more time for the web UI to start (3 to 5 minutes) before retrying your login attempt.

After all services show a green check mark, click **Next**.

Preparation Complete

Once all Nodes have been configured, and all services have been started on all nodes, the **Preparation Complete** page will be shown, meaning that the installation process is now ready to configure product-specific installation attributes.

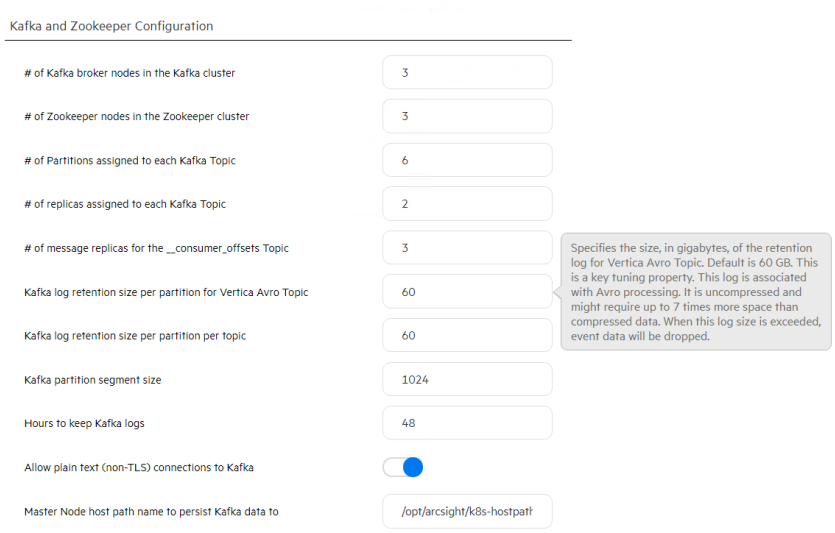


Click **Next** to configure the products and components of the deployment.

Configure and Deploy Transformation Hub

The Transformation Hub is now ready to be configured. The Transformation Hub Pre-Deployment Configuration page is displayed to configure the products and capabilities chosen at the start of the installation process.

The pre-deployment configuration page allows tuning of the initial installation properties. Click the **Transformation Hub** tab and modify the configuration properties as required, based on the size of your cluster and its throughput requirements. Refer to the CDF Deployment Disk Sizing Calculator spreadsheet (available from the Micro Focus support community) for guidance on setting any of these properties. Hover over a value to see a detailed description associated with the configuration property.



Worker Node Properties: You must adjust several of these properties with the number of worker nodes installed earlier in this installation process.

Minimally, synchronize the following properties to the worker nodes (default value is 3 for each property except the one noted below).

- # of Worker Nodes in the Kafka cluster
- # of Worker Nodes running Zookeeper in the Kafka cluster
- # of Schema Registry nodes in the Kafka cluster
- # of Kafka nodes required to run Schema Registry (Default: 2)
- Kafka replication factor (this must be set to '1' for a Single Worker deployment)

For example, if you chose a Single Worker installation, you would set the values of all of these properties to 1.

Log Properties: It is highly likely the following configuration properties should also be adjusted from their default values. Note that proper log sizes are critical. Should a node run out of disk space, messages (events) will be dropped and are not recoverable.

- Retention log size for each partition of a Vertica Topic
- Retention log size per Kafka Topic
- Hours to keep Kafka logs

ArcMC Properties: For managing your cluster with ArcMC, you can specify your Management Center FQDN: {port}. Note that this can only be configured on the post-deployment configuration page.

After updating configuration property values, click **Next** to deploy Transformation Hub. After a few minutes, the CDF Management Portal URL will be displayed. Select this URL to finish Transformation Hub deployment.

Security Mode Configuration

Prior to deployment, you must choose and configure the security mode that Transformation Hub will use to connect.

- By default, plain-text (or non-TLS) connections are permitted from external producers and consumers (such as connectors, ESM, and Logger), to maximize performance.
- For higher security, you can disable plain-text connections .

The following table shows the effect of each security mode configuration setting on communication over the given port.

Security Mode Setting	Value	Connect to Port 9092 (Plain Text)	Connect to Port 9093 (TLS)
Allow Plain Text Connections	true	yes	yes
Allow Plain Text Connections	false	no	yes
Client Authentication	true	N/A	yes
Client Authentication	false	N/A	yes
FIPS	true	N/A	yes
FIPS	false	N/A	yes

- 9093 is the endpoint used for TLS, FIPS and Client Authentication. It is always enabled. Note that when Plain Text is disabled, the plain text port is still open (due to pending technical issues). However, the Kafka endpoint will not accept data.
- 9092 is the endpoint used for plain text, and is enabled by the *Allow Plain Text connections* configuration setting, which is new in Transformation Hub version 3.2. This setting has no effect on the FIPS and Client Authentication settings.

Note: Configure these settings ***before*** deployment of the Transformation Hub. Changing them after deployment will result in cluster downtime.

Label Worker Nodes

.Labeling is a means for identifying application processing and qualifying the application as a candidate to run on a specific node. For example, labeling a node with the label `kafka:yes` indicates that a Kafka instance will run on that node.

In this step, you will first define labels and then apply them to each node.

Immediately following Transformation Hub deployment the following pods will remain in a ***Pending*** state awaiting the labeling process to be completed: `th-kafka, th-zookeeper, th-kafka-manager`, `th-web-service,` and `th-schemaregistry` . Once labeling is completed, Kubernetes will immediately schedule and start the label-dependent containers on the labeled nodes. (Note that starting of services may take 15 minutes or more to complete.)

To label your worker nodes:

Note: Upon first login, you are prompted to change the admin password.

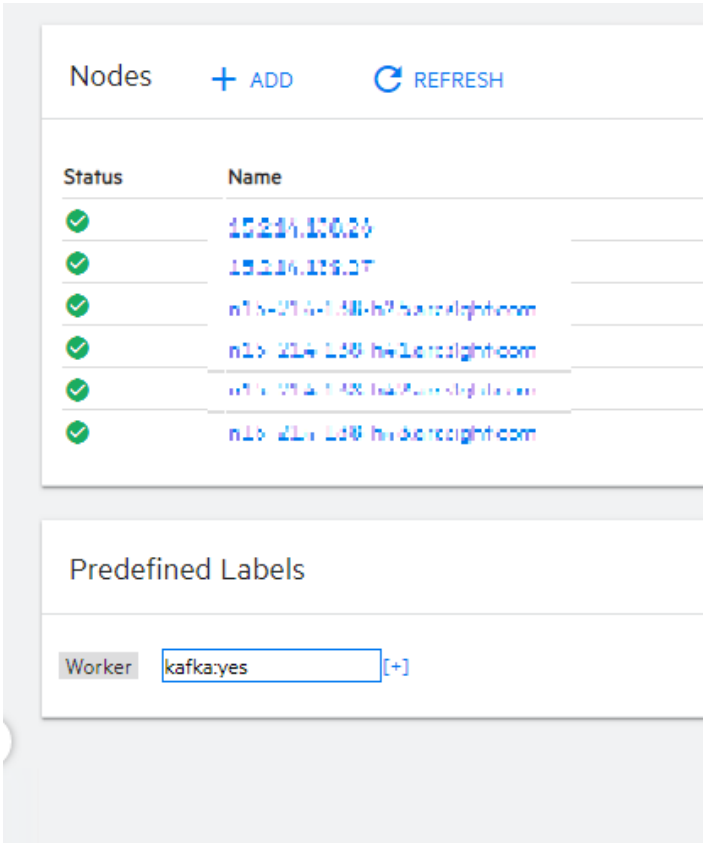
1. Login to Management Portal by clicking the link on the **Deployment status** (Configuration complete) page or browsing to (`https://<ha-address>:5443`), where:

- *Ha-address:* FQDN corresponding to the Virtual IP address provided during installation (`--ha-virtual-ip`) (or, for a single-master installation, the IP address of the master node).
- *User Name:* admin
- *Password:* Password that you changed to during the first login to port 5443.

2. Go to **CLUSTER> Nodes.**
3. In **PredefinedLabels** enter the label `zk:yes` (case-sensitive) and then click the **+** icon. This will add the `zk:yes` label to the list of predefined labels you can use to label nodes. The label list will be shown to the left of the text box.
4. Repeat Step 3 for each of the following labels to add them to the list of predefined labels. Enter the text of the entire label, as shown here, including the `:yes` text. Labels are case-sensitive.

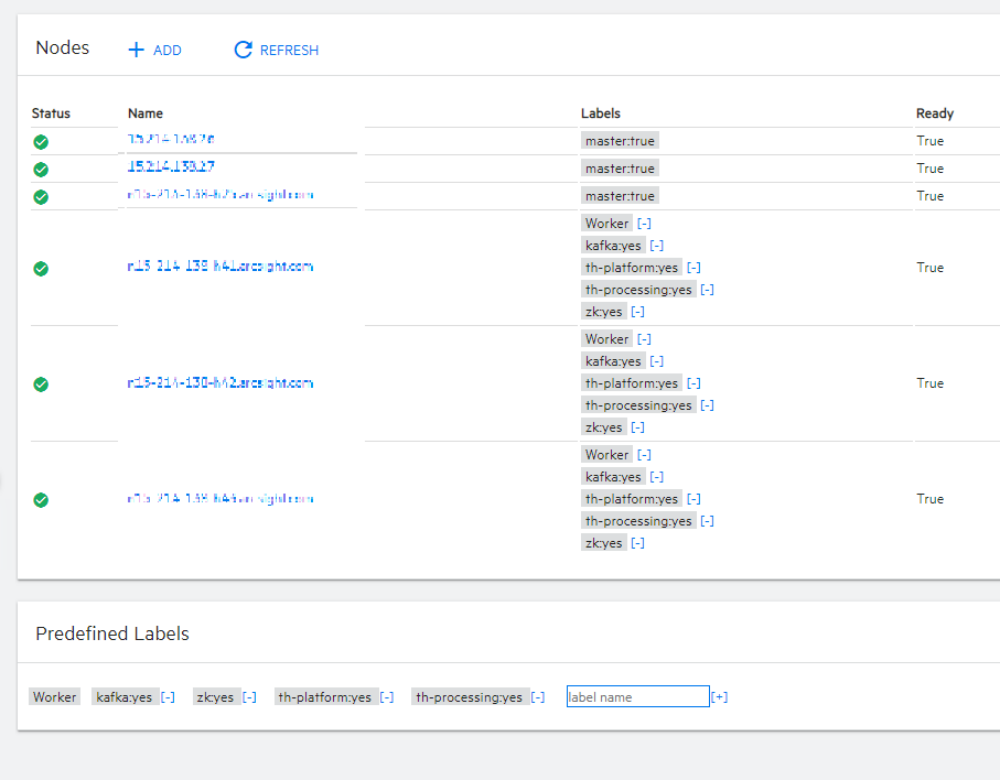
```
kafka:yes

th-processing:yes

th-platform:yes
```

Note: The `master:yes` and `Worker` labels are already predefined, and already applied to your Nodes based on your installation. You will not need to take any action regarding these labels.



5. Drag and drop each label from the **Predefined Labels** list to each of the *worker nodes*, based on your workload sharing configuration. This will apply the dragged label to the selected node.

For Kafka and ZooKeeper, make sure that the number of the nodes you have labeled corresponds to the number of worker nodes in the Kafka cluster and the number of worker nodes running Zookeeper in the Kafka cluster properties from the pre-deployment configuration page. The default number is 3 for a Multiple Worker deployment.



Once the nodes have been properly labeled, the status of the Transformation Hub pods will change from a *Pending* to a *Running* state. You can monitor the pod startup process by running the following command on the Initial master node:

```
kubectl get pods --all-namespaces -o wide
```

Check Deployment Status

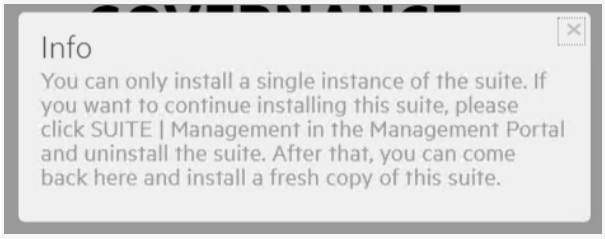
When the **Configuration Complete** page displays, the pod deployment is finished.

- Pods that have not been labeled will remain in the *Pending* state until labeled.
- For a pod that is not in the *Running* state, you can find out more details on the pod by running the following command:

```
kubectl describe pod <pod name> -n <namespace>
```

The `Events` section in the output provides detailed information on the pod status.

Note: If the following error is displayed when attempting to log in to the CDF Management Portal on port 3000, this typically means that the CDF installation process has completed, port 3000 is no longer required, and has been closed. Instead of port 3000, log in to the Management Portal on port 5443.



Check Cluster Status

To verify the success of the deployment, check the cluster status and make sure all pods are running.

Note: You may need to wait 10 minutes or more for all pods to be in a *Running* or *Completed* state.

1. Log into the Initial Master Node.
2. Run the command:

```
kubectl get pods --all-namespaces
```

Review the output to determine the status of all pods.

Post-Deployment Configuration

Depending on your architecture, after deployment, you may need to adjust some of the post-deployment configuration properties in order for Transformation Hub to function correctly.

If you plan to manage Transformation Hub with ArcMC, then you will need to adjust some settings in the post-deployment stage with your ArcMC details. Whether you need to adjust other properties during post-configuration will depend on the specifics of your implementation.

For a more detailed discussion of post-deployment configuration settings, see the Transformation Hub Administrator's Guide.

To configure post-deployment settings:

1. Browse to the CDF Management Portal.
2. Click **Suite**.
3. Click the **...** (Browse) icon to the right of the main window.
4. From the drop-down, click **Reconfigure**. The post-deployment settings page is displayed.
5. Set values for the parameters as needed.
6. For configuring management of Transformation Hub with ArcMC, see Configuring ArcMC Management of Transformation Hub

7. In addition, under **Stream Processors and Routers**, adjust the '# of CEF-to-Avro Stream Processors and Routing Stream Processors' as per group as required.
8. Click **Save.**

Web services in the cluster will be restarted (in a rolling manner) across the cluster nodes.

Note: In order to enable ArcMC management, some configuration of ArcMC is also necessary. For more information, see Configuring ArcMC Management of Transformation Hub

Management Center: Configuring Transformation Hub

The Management Center (ArcMC) is the centralized console for managing Micro Focus products.

Connectivity between Transformation Hub and ArcMC is configured in ArcMC when you add Transformation Hub as a managed host into ArcMC. For details on adding your Transformation Hub to ArcMC, see here.

Reminder: Install Your License Key

Transformation Hub ships with a 90-day instant-on evaluation license, which will enable functionality for 90 days after installation. In order for Transformation Hub to continue working past the initial evaluation period, you will need to apply a valid license key to Transformation Hub. A Transformation Hub license key, as well as a valid legacy ArcMC ADP license key, can be used for licensing Transformation Hub.

For details on how to apply a your license key to Transformation Hub, see the Licensing chapter of the Transformation Hub Administrator's Guide.

IMPORTANT: To ensure continuity of functionality and event flow, make sure you apply your product license **before** the evaluation license has expired.

Chapter 5: Integrating Transformation Hub Into Your ArcSight Environment

Transformation Hub centralizes event processing and enables event routing, which helps you to scale your ArcSight environment and opens event data to ArcSight and third-party solutions. Transformation Hub takes advantage of scalable and highly-available clusters for publishing and subscribing to event data. Transformation Hub integrates with ArcSight SmartConnectors and Collectors, Logger, ESM, and ArcSight Investigate. It is managed and monitored by ArcSight Management Center.

After you install and configure Transformation Hub you can use SmartConnectors and Collectors to produce and publish data to the Transformation Hub, and to subscribe to and consume that data with Logger, ESM, ArcSight Investigate, Apache Hadoop, or your own custom consumer.

Transformation Hub supports both Common Event Format (CEF) versions, 0.1 and 1.0.

Transformation Hub third-party integration and other product features are explained in detail in the Transformation Hub Administrator's Guide, available from the Micro Focus support community.

This chapter includes the following sections:

Default Topics

Transformation Hub manages the distribution of events to topics, to which consumers can subscribe and receive events from.

Transformation Hub includes the following default topics:

Topic Name	Event Type	Valid Destinations
th-cef	CEF event data.	Can be configured as a SmartConnector or Connector in Transformation Hub (CTH) destination.
th-binary_esm	Binary security events, which is the format consumed by ArcSight ESM.	Can be configured as a SmartConnector or CTH destination.

Topic Name	Event Type	Valid Destinations
th-syslog	The Connector in Transformation Hub (CTH) device receives raw syslog data from this topic using a Collector.	Can be configured as Collector destination.
th-cef-other	CEF event data destined for a non-ArcSight subscriber.	
th-arcsight-avro-sp_ metrics	For ArcSight product use only. Stream processor operational metrics data.	
th-arcsight-avro	For ArcSight product use only. Event data in Avro format for use by ArcSight Investigate.	
th-arcsight-json-datastore	For ArcSight product use only. Data in JSON format for use by ArcSight infrastructure management.	

In addition, using ArcSight Management Center, you can create new custom topics to which your SmartConnectors can connect and send events.

Data Preservation

Topic data is preserved across node restarts and reinstalls.

- When a Transformation Hub reinstall or redeployment is performed, all data that resides in Kafka topics is preserved. No data is lost. On a worker node, by default, events data is stored in:`/opt/arcsight/k8s-hostpath-volume/th/kafka`.
- When a consumer resumes data consumption from Kafka topics, the consumer re-starts where it last left off. No data is lost.
- If a worker node is stopped, that node will be reported as unavailable to the cluster. All events data stored on the worker node will be preserved and events processing will resume as soon as the node is started again.
- If a master node is stopped, that node will be reported as unavailable to the cluster. All other functionality, including events processing on the worker nodes, will continue.

Configuring ArcMC to Manage a Transformation Hub

ArcMC serves as the management UI for Transformation Hub. In order for ArcMC to manage a Transformation Hub, the Transformation Hub must be added as a managed host to ArcMC. This process will include these steps, explained below:

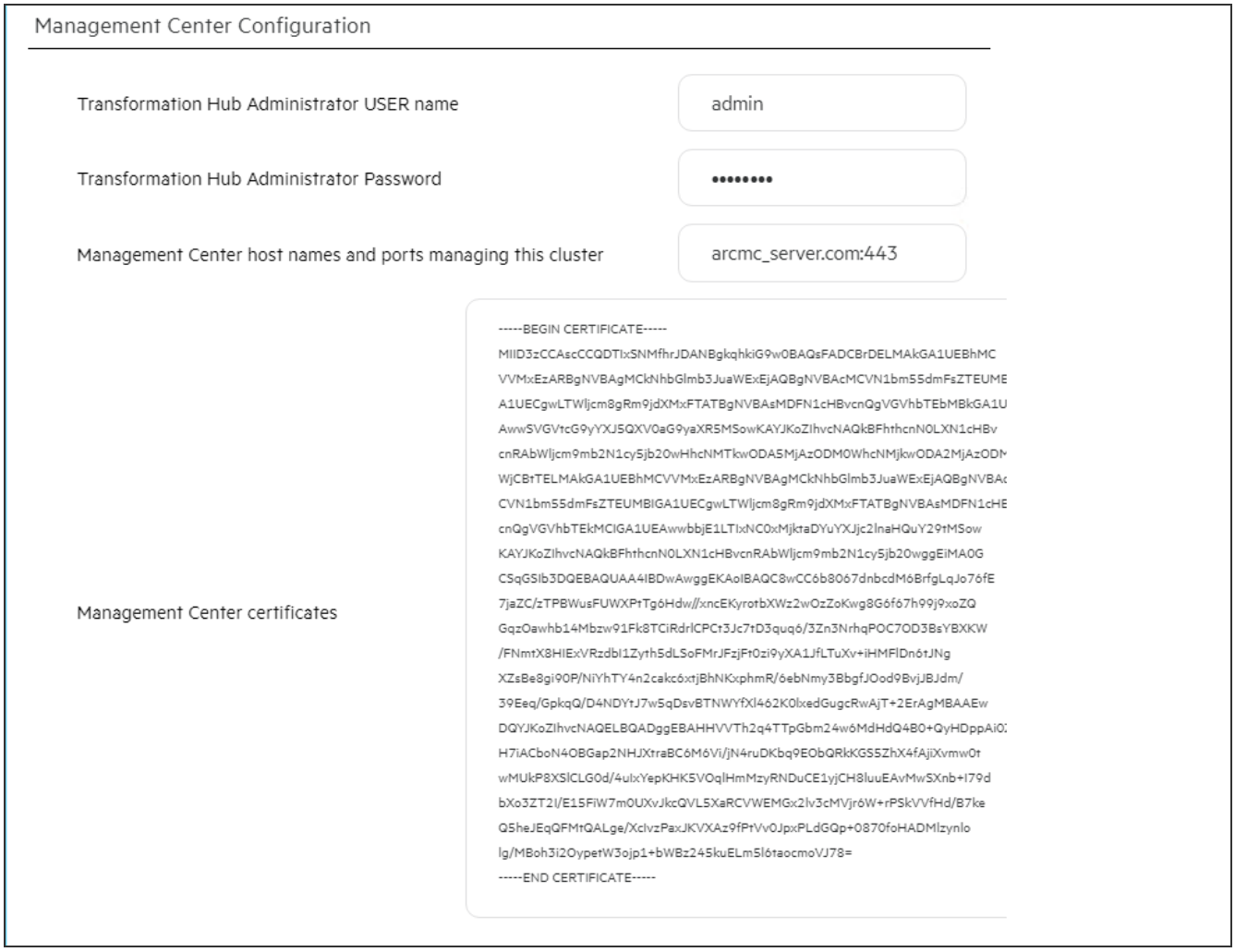
- Retrieve the ArcMC certificate from your ArcMC
- Configure the CDF cluster with ArcMC details
- Retrieve the CDF certificate
- Configure ArcMC

Retrieve the ArcMC certificate:

1. Log into ArcMC.
2. Click **Administration > System Admin > SSL Server Certificate > Generate Certificate.**
3. On the **Enter Certificate Settings** dialog, enter the required settings. In **Hostname**, your certificate settings must match the FQDN of your ArcMC.
4. Click **Generate Certificate.**
5. Once the certificate is generated, click **View Certificate** and copy the full content from `--BEGIN cert to END cert--` to the clipboard.

Configure the CDF cluster:

1. Log in to the CDF management portal.
2. Click **Suite**.
3. Click **...** (Browse) on the far right and choose **Reconfigure**. A new screen will be opened in a separate tab.
3. Scroll down to the Management Center Configuration section. Then, enter values as described for the following:
   - **Username:** `admin`
   - Enter the ArcMC hostname and port 443 (for example, `arcmc.example.com:443`). If ArcMC was installed as a non-root user, enter port 9000 instead.
   - **ArcMC certificates:** Paste the text of the generated server certificates you copied to the clipboard as described above.



4. Click **Save**. Web services pods in the cluster will be restarted

Retrieve the CDF certificate:

1. On the initial master node of the cluster, run the following:
   `${K8S_HOME}/scripts/cdf-updateRE.sh`
2. Copy the contents of this certificate, from `--BEGIN cert to END cert--`, to the clipboard.

Configure ArcMC:

1. Log in to the ArcMC.
2. Click **Node Management > View All Nodes.**
3. In the navigation bar, click Default (or the ArcMC location where you wish to add Transformation Hub). Then click **Add Host,** and enter the following values:
   - **Hostname:** Hostname for the Virtual IP for an HA environment, or master node host name for a single-master node environment

- **Type:** Select Transformation Hub Containerized
- **Port**: 38080
- **Cluster Port:** 443
- **Cluster Username:** admin
- **Cluster Password:** <admin password created when logging into the CDF UI for the first time>
- **Cluster Certificate:** Paste the contents of the CDF certificate you copied earlier.

4. Click **Add**. The Transformation Hub is added as a managed host.

Configuring Security Mode for Transformation Hub Destinations

Follow these instructions to configure a security mode for SmartConnectors with Transformation Hub destinations. For additional Transformation Hub configuration, see the Transformation Hub ***Administrator's Guide*** and "Transformation Hub" in the ***Smart Connector User Guide*** on the Micro Focus support community.

Note: These procedures are provided with the following assumptions:

- You use the default password. See the appendix for FIPS Compliant SmartConnectors in the *SmartConnector User Guide* on the Micro Focus support community to set a non-default password.
- You are on the Linux platform. For Windows platforms, use backslashes (\) when entering commands instead of the forward slashes given here.
- You are using a command window to enter Windows commands. Do not use Windows PowerShell.

Configuring a SmartConnector with aTransformation Hub Destination without Client Authentication in non-FIPS Mode

Follow these steps to configure a SmartConnector with a Transformation Hub destination without client authentication in non-FIPS mode. This is the default security mode configuration when installing Transformation Hub.

On the SmartConnector Server

1. Prepare the SmartConnector:
   - **If the connector is not yet installed:** Run the installer. After core software has been installed, you will see a window that lets you select **Add a Connector** or **Select Global Parameters**. Check **Select Global Parameters**, and on the window displayed, select **Set FIPS mode**. Set to **Disabled**.

- **If the connector is already installed:** Run the installer. Select **Set Global Parameters** and set **Set FIPS Mode** to **Disabled**.

2. Navigate to the connector's `current` directory, for example:
   `cd <install dir>/current`
3. Set the environment variables for the static values used by keytool, for example:

```
export CURRENT=<full path to this "current" folder>
export TH=<Transformation Hub hostname>_<Transformation Hub port>
export STORES=${CURRENT}/user/agent/stores
export CA_CERT=ca.cert.pem
export STORE_PASSWD=changeit
```

   On Windows platforms:

```
set CURRENT=<full path to this "current" folder>v
set TH=<Transformation Hub hostname>_<Transformation Hub port>
set STORES=%CURRENT%\user\agent\stores
set STORE_PASSWD=changeit
```

4. Create the `${CURRENT}/user/agent/stores` directory if it does not already exist, for example:
   `mkdir -p ${STORES}`

   On Windows platforms:

```
mkdir -p %STORES%
```

On the Transformation Hub:

Create a `${CA_CERT}` file with the content of the root CA certificate as follows:

1. Set the environment:
   `export CA_CERT=/tmp/ca.cert.pem`
2. Create a certificate:
   `${k8s-home}/scripts/cdf-updateRE.sh > ${CA_CERT}`
3. Copy this file from the Transformation Hub to the connector `STORES` directory.

On the Connector:

1. Import the CA certificate to the trust store in the `${CURRENT}` folder; for example:

```
jre/bin/keytool -importcert -file ${STORES}/${CA_CERT} -alias CARoot -
keystore ${STORES}/${TH}.truststore.jks -storepass ${STORE_PASSWD}
```

On Windows platforms:

```
jre\bin\keytool -importcert -file %STORES%\%CA_CERT% -alias CARoot -
keystore %STORES%\%TH%.truststore.jks -storepass %STORE_PASSWD%
```

2. When prompted, enter `yes` to trust the certificate.
3. Note the trust store path:

   ```
   echo ${STORES}/${TH}.truststore.jks
   ```

   On Windows platforms:

   ```
   echo %STORES%\%TH%.truststore.jks
   ```

4. Navigate to the `bin` directory and run agent setup. Install a connector with Transformation Hub as the destination, for example:

   ```
   cd <installation dir>/current/bin
   ./runagentsetup.sh
   ```

   On Windows platforms:

   ```
   cd <installation dir>\current\bin
   runagentsetup.bat
   ```

5. Set **Use SSL/TLS** to **true**.
6. Set **Use SSL/TLS Authentication** to **false**.
7. When completing the Transformation Hub destination fields, use the value from Step 3 for the trust store path and the password used in Step 4 for the trust store password.
8. Cleanup. Delete the certificate file, for example:

   Caution: The following file should be deleted to prevent the distribution of security certificates that could be used to authenticate against the Transformation Hub. These files are very sensitive and should not be distributed to other machines.

   ```
   rm ${STORES}/${CA_CERT}
   ```

   On Windows platforms:

   ```
   del %\STORES%\%CA_CERT%
   ```

Configuring a SmartConnector with aTransformation Hub Destination with Client Authentication in FIPS Mode

Follow these steps to configure a SmartConnector with a Transformation Hub (TH) destination with client authentication in FIPS mode.

Note: You will need to supply an intermediate certificate and key. See Appendix B for an example of how to generate these.

Step 1: On the Connector Server

1. Prepare the connector:
   - **If the connector is not yet installed:** Run the installer. After core software has been installed, you will see a window that lets you select **Add a Connector** or **Select Global Parameters**. Check **Select Global Parameters**, and on the window displayed, select **Set FIPS mode**. Set to **Enabled**.
   - **If the connector is already installed:** Run the installer. Select **Set Global Parameters** and set **Set FIPS Mode** to **Enabled**.
2. Navigate to the connector's `current` directory, for example:

   ```
   cd <install dir>/current
   ```
3. Apply the following workaround for a Java keytool issue:
   a. Create a new file, `agent.security`, in `<install dir>/current/user/agent` (or in Windows platforms, `<install dir>\current\user\agent`).
   b. Add the following content to the new file and then save it:

   ```
   security.provider.1=org.bouncycastle.jcajce.provider
   .BouncyCastleFipsProvider
   security.provider.2=com.sun.net.ssl.internal.ssl.Provider BCFIPS
   security.provider.3=sun.security.provider.Sun
   ```

   c. Move the `lib/agent/fips/bcprov-jdk14-119.jar` file to the `current` directory.
4. Set the environment variables for static values used by keytool:

   ```
   export CURRENT=<full path to this "current" folder>
   export BC_OPTS="-storetype BCFKS -providername BCFIPS
   -J-Djava.security.egd=file:/dev/urandom
   -J-Djava.ext.dirs=${CURRENT}/jre/lib/ext:${CURRENT}/lib/agent/fips
   -J-Djava.security.properties=${CURRENT}/user/agent/agent.security"
   export TH=<Transformation Hub hostname>_<Transformation Hub port>
   export STORES=${CURRENT}/user/agent/stores
   export STORE_PASSWD=changeit
   export TH_HOST=<TH master host name>
   export CA_CERT=ca.cert.pem
   export INTERMEDIATE_CA_CRT=intermediate.cert.pem
   export FIPS_CA_TMP=/opt/fips_ca_tmp
   ```

   On Windows platforms:

   ```
   set CURRENT=<full path to this "current" folder>
   ```

```
set BC_OPTS=-storetype BCFKS -providername BCFIPS
-J-Djava.ext.dirs=%CURRENT%\jre\lib\ext;%CURRENT%\lib\agent\fips
-J-Djava.security.properties=%CURRENT%\user\agent\agent.security
set TH=<Transformation Hub hostname>_<Transformation Hub port>
set STORES=%CURRENT%\user\agent\stores
set STORE_PASSWD=changeit
set TH_HOST=<TH master host name>
set CA_CERT=C:\Temp\ca.cert.pem
set INTERMEDIATE_CA_CRT=C:\Temp\intermediate.cert.pem
set FIPS_CA_TMP=\opt\fips_ca_tmp
```

5. Create the `${CURRENT}/user/agent/stores` directory if it does not already exist, for example:

   ```
   mkdir -p ${STORES}
   ```

   On Windows platforms:

   ```
   mkdir -p %STORES%
   ```

6. Create the connector key pair, for example (the connector `FQDN`, `OU`, `O`, `L`, `ST`, and `C` values must be changed for your company and location):

   ```
   jre/bin/keytool ${BC_OPTS} -genkeypair -alias ${TH} -keystore
   ${STORES}/${TH}.keystore.bcfips -dname "cn=<Connector
   FQDN>,OU=Arcsight,O=MF,L=Sunnyvale,ST=CA,C=US" -validity 365
   ```

   On Windows platforms:

   ```
   jre\bin\keytool %BC_OPTS% -genkeypair -alias %TH% -keystore
   %STORES%\%TH%.keystore.bcfips -dname "cn=<Connector
   FQDN>,OU=Arcsight,O=MF ,L=Sunnyvale,ST=CA,C=US" -validity 365
   ```

   When prompted, enter the password. Note the password; you will need it again in a later step. Press **Enter** to use the same password for the key. If you want to match the default value in the properties file, use the password `changeit`.

7. List the key store entries. There should be one private key.

   ```
   jre/bin/keytool ${BC_OPTS} -list -keystore ${STORES}/${TH}.keystore.bcfips
   -storepass ${STORE_PASSWD}
   ```

   On Windows platforms:

   ```
   jre\bin\keytool %BC_OPTS% -list -keystore %STORES%\%TH%.keystore.bcfips
   -storepass %STORE_PASSWD%
   ```

8. Create a Certificate Signing Request (CSR), for example:

```
jre/bin/keytool ${BC_OPTS} -certreq -alias ${TH} -keystore
${STORES}/${TH}.keystore.bcfips -file ${STORES}/${TH}-cert-req -storepass
${STORE_PASSWD}
```

On Windows platforms:

```
jre\bin\keytool %BC_OPTS% -certreq -alias %TH% -keystore
%STORES%\%TH%.keystore.bcfips -file %STORES%\%TH%-cert-req -storepass
%STORE_PASSWD%
```

Step 2: On the Transformation Hub Server

1. When Transformation Hub is first installed, it's setup to use self-signed certificates. To replace the self-signed certificates, obtain your company's root CA certificate, and an intermediate certificate and key pair. Copy them to `/tmp` with the following names:

   `/tmp/intermediate.cert.pem`

   `/tmp/intermediate.key.pem`

   `/tmp/ca.cert.pem`

   Use the following command to update the certificate on the Transformation Hub:

   ```
   /opt/arcsight/kubernetes/scripts/cdf-updateRE.sh write --re-
   key=/tmp/intermediate.key.pem --re-crt=/tmp/intermediate.cert.pem --re-
   ca=/tmp/ca.cert.pem
   ```

   Note: After the new certificate is imported to the Transformation Hub, the Transformation Hub will need to be uninstalled and then re-installed with FIPS and Client Authentication enabled. See the *Transformation Hub Deployment Guide* for details.

2. `export CA_CERT=/tmp/ca.cert.pem`

   `export INTERMEDIATE_CA_CRT=/tmp/intermediate.cert.pem`

   `export INTERMEDIATE_CA_KEY=/tmp/intermediate.key.pem`

   `export FIPS_CA_TMP=/opt/fips_ca_tmp`

   `export TH=<Transformation Hub hostname>_<Transformation Hub port>`

3. Create a temporary location on the Transformation Hub master server: `mkdir $FIPS_CA_TMP`

Step 3: On the Connector Server

Copy the `${STORES}/${TH}-cert-req` file (`%STORES%\%TH%-cert-req` on Windows platforms) from the connector to the Transformation Hub directory created above, `/opt/fips_ca_tmp`.

Step 4: On the Transformation Hub Server

Create the signed certificate, for example:

```
/bin/openssl x509 -req -CA ${INTERMEDIATE_CA_CRT} -CAkey ${INTERMEDIATE_CA_
KEY} -in ${TH}-cert-req -out ${FIPS_CA_TMP}/${TH}-cert-signed-days 365 -
CAcreateserial -sha256
```

Step 5: On the Connector Server

1. Copy the `${TH}-cert-signed` certificate from the Transformation Hub to the connector's `${STORES}` directory. (On the Windows platform, copy the `%TH%-cert-signed` certificate to the connector's `%STORES%` directory.)
2. Copy the `ca.cert.pem` certificate from the Transformation Hub to the connector's `${STORES}` directory. (On the Windows platform, copy the certificate to the `%STORES%` directory.)
3. Copy the `intermediate.cert.pem` certificate from the Transformation Hub to the connector's `${STORES}` directory. (On the Windows platform, copy the certificate to the `%STORES%` directory.)
4. Import the CA certificate to the trust store, for example:

   ```
   jre/bin/keytool ${BC_OPTS} -importcert -file ${STORES}/${CA_CERT} -alias
   CARoot -keystore ${STORES}/${TH}.truststore.bcfips -storepass ${STORE_
   PASSWD}
   ```

   On Windows platforms:

   ```
   jre\bin\keytool %BC_OPTS% -importcert -file %STORES%\%CA_CERT% -alias
   CARoot -keystore %STORES%\%TH%.truststore.bcfips -storepass %STORE_
   PASSWD%
   ```

5. Import the intermediate certificate to the trust store, for example:

   ```
   jre/bin/keytool ${BC_OPTS} -importcert -file ${STORES}/${INTERMEDIATE_CA_
   CRT} -alias INTCARoot -keystore ${STORES}/${TH}.truststore.bcfips -
   storepass ${STORE_PASSWD}
   ```

   On Windows platforms:

   ```
   jre\bin\keytool %BC_OPTS% -importcert -file %STORES%\%INTERMEDIATE_CA_
   CRT% -aliasINTCARoot -keystore %STORES%\%TH%.truststore.bcfips -
   storepass %STORE_PASSWD%
   ```

6. Import the CA certificate to the key store, for example:

   ```
   jre/bin/keytool ${BC_OPTS} -importcert -file ${STORES}/${CA_CERT} -alias
   CARoot -keystore ${STORES}/${TH}.keystore.bcfips -storepass ${STORE_
   PASSWD}
   ```

On Windows platforms:

```
jre\bin\keytool %BC_OPTS% -importcert -file %STORES%\%CA_CERT% -alias
CARoot -keystore %STORES%\%TH%.keystore.bcfips -storepass %STORE_
PASSWD%
```

7. When prompted, enter **yes** to trust the certificate.
8. Import the intermediate certificate to the key store, for example:

   ```
   jre/bin/keytool ${BC_OPTS} -importcert -file ${STORES}/${INTERMEDIATE_CA_
   CRT} -alias
   INTCARoot -keystore ${STORES}/${TH}.keystore.bcfips -storepass ${STORE_
   PASSWD}
   ```

   When completed successfully, this command will return the message, `Certificate reply was installed in keystore`.

   On Windows platforms:

   ```
   jre\bin\keytool %BC_OPTS% -importcert -file %STORES%\%INTERMEDIATE_CA_
   CRT% -alias
   INTCARoot -keystore %STORES%\%TH%.keystore.bcfips -storepass %STORE_
   PASSWD%
   ```

9. Import the signed certificate to the key store, for example:

   ```
   jre/bin/keytool ${BC_OPTS} -importcert -file ${STORES}/${TH}-cert-signed
   -alias ${TH} -keystore ${STORES}/${TH}.keystore.bcfips -storepass ${STORE_
   PASSWD}
   ```

   On Windows platforms:

   ```
   jre\bin\keytool %BC_OPTS% -importcert -file %STORES%\%TH%-cert-signed
   -alias %TH% -keystore %STORES%\%TH%.keystore.bcfips -storepass %STORE_
   PASSWD%
   ```

   When successfully complete, this command will return the message, *Certificate reply was installed in keystore.*

10. Navigate to the `bin` directory and run agent setup. Install a connector with Transformation Hub as the destination, for example:

    ```
    cd <installation dir>/current/bin
    ./runagentsetup.sh
    ```

    On Windows platforms:

    ```
    cd <installation dir>\current\bin
    runagentsetup.bat
    ```

a. When completing the Transformation Hub destination fields, use the same values as in Step 8 for the path and password.
b. Set **Use SSL/TLS** to **true**.
c. Set **Use SSL/TLS Authentication** to **true**.
d. Set keystore path to:
`${STORES}/${TH}.keystore.bcflips`
e. Set truststore path to:
`${STORES}/${TH}.keystore.bcflips`

11. Cleanup. Delete the following files:

Caution: The following files should be deleted to prevent the distribution of security certificates that could be used to authenticate against the Transformation Hub. These files are very sensitive and should not be distributed to other machines.

```
rm ${STORES}/${INTERMEDIATE_CA_CRT}
rm ${STORES}/intermediate.key.pem
rm ${STORES}/${TH}-cert-signed
rm ${STORES}/${TH}-cert-req
```

On Windows platforms:

```
del %STORES%\intermediate.cert.pem
del %STORES%\intermediate.key.pem
del %STORES%\%TH%-cert-signed
del %STORES%\%TH%-cert-req
```

12. Move the `bcprov-jdk14-119.jar` file back to the `lib/agent/fips` directory (or `lib\agent\fips` on Windows platforms).

Step 6: On the Transformation Hub Server

To clean up the Transformation Hub server, delete the temporary folder where the certificate was signed and the certificate and key files in `/tmp`.

Caution: The temporary certificate folder should be deleted to prevent the distribution of security certificates that could be used to authenticate against the Transformation Hub. These files are very sensitive and should not be distributed to other machines.

Configuring a SmartConnector with Transformation Hub Destination with Client Authentication in Non-FIPS Mode

Follow these steps to configure a SmartConnector with a Transformation Hub (TH) destination with client authentication, in non-FIPS mode.

Note: You will need to supply an intermediate certificate and key. See Appendix B for an example of how to generate these.

Step 1: On the Connector Server

1. Prepare the SmartConnector:
   - **If the connector is not yet installed:** Run the installer. After core software has been installed, you will see a window that lets you select **Add a Connector** or **Select Global Parameters**. Check **Select Global Parameters**, and on the window displayed, select **Set FIPS mode**. Set to **Disabled**.
   - **If the connector is already installed:** Run the installer. Select **Set Global Parameters** and set **Set FIPS Mode** to **Disabled**.
2. Navigate to the connector's `current` directory, for example:

   ```
   cd <install dir>/current
   ```

   On Windows platforms:

   ```
   cd <install dir>\current
   ```

3. Set the environment variables for the static values used by keytool, for example:

   ```
   export CURRENT=<full path to this "current" folder>
   export TH=<th hostname>_<th port>
   export STORES=${CURRENT}/user/agent/stores
   export STORE_PASSWD=changeit
   export TH_HOST=<TH master host name>
   export CA_CERT=ca.cert.pem
   export INTERMEDIATE_CA_CRT=intermediate.cert.pem
   export CERT_CA_TMP=/opt/cert_ca_tmp
   ```

   On Windows platforms:

   ```
   set CURRENT=<full path to this "current" folder>
   set TH=<th hostname>_<th port>
   set STORES=%CURRENT%\user\agent\stores
   set STORE_PASSWD=changeit
   set TH_HOST=<TH master host name>
   set CA_CERT=C:\Temp\ca.cert.pem
   set INTERMEDIATE_CA_CRT=C:\Temp\intermediate.cert.pem
   set CERT_CA_TMP=\opt\cert_ca_tmp
   ```

4. Create the `${CURRENT}/user/agent/stores` directory if it does not already exist, for example:

   ```
   mkdir -p ${STORES}
   ```

   On Windows platforms:

```
mkdir -p %STORES%
```

5. Create the connector key pair, for example:

   ```
   jre/bin/keytool -genkeypair -alias ${TH} -keystore
   ${STORES}/${TH}.keystore.jks -dname "cn=<Connector
   FQDN>,OU=Arcsight,O=MF,L=Sunnyvale,ST=CA,C=US" -validity 365
   ```

   On Windows platforms:

   ```
   jre\bin\keytool -genkeypair -alias %TH% -keystore
   %STORES%\%TH%.keystore.jks -dname "cn=<Connector
   FQDN>,OU=Arcsight,O=MF,L=Sunnyvale,ST=CA,C=US" -validity 365
   ```

   When prompted, enter the password. Note the password; you will need it again in a later step. Press Enter to use the same password for the key.

6. List the key store entries. There should be one private key.

   ```
   jre/bin/keytool -list -keystore ${STORES}/${TH}.keystore.jks -storepass
   ${STORE_PASSWD}
   ```

   On Windows platforms:

   ```
   jre\bin\keytool -list -keystore %STORES%\%TH%.keystore.jks -storepass
   %STORE_PASSWD%
   ```

7. Create a Certificate Signing Request (CSR), for example:

   ```
   jre/bin/keytool -certreq -alias ${TH} -keystore
   ${STORES}/${TH}.keystore.jks -file ${STORES}/${TH}-cert-req -storepass
   ${STORE_PASSWD}
   ```

   On Windows platforms:

   ```
   jre\bin\keytool -certreq -alias %TH% -keystore
   %STORES%\%TH%.keystore.jks -file %STORES%\%TH%-cert-req -storepass
   %STORE_PASSWD%
   ```

Step 2: On the Transformation Hub Server

1. When Transformation Hub is first installed, it's setup to use self-signed certificates. To replace the self-signed certificates, obtain your company's root CA certificate, and an intermediate certificate and key pair. Copy them to `/tmp` with the following names:

   `/tmp/intermediate.cert.pem`

   `/tmp/intermediate.key.pem`

   `/tmp/ca.cert.pem`

   Use the following command to add them to Transformation Hub:

```
/opt/arcsight/kubernetes/scripts/cdf-updateRE.sh write --re
key=/tmp/intermediate.key.pem --re-crt=/tmp/intermediate.cert.pem --re-
ca=/tmp/ca.cert.pem
```

Note: After the new certificate is imported to the Transformation Hub, the Transformation Hub will need to be uninstalled and then re-installed with Client Authentication enabled. See the *Transformation Hub Deployment Guide* for details.

2. Run the following commands:

```
export CA_CERT=/tmp/ca.cert.pem
export INTERMEDIATE_CA_CRT=/tmp/intermediate.cert.pem
export INTERMEDIATE_CA_KEY=/tmp/intermediate.key.pem
export CERT_CA_TMP=/opt/cert_ca_tmp
export TH=<Transformation Hub hostname>_<Transformation Hub port>
```

3. Create a temporary location on the Transformation Hub master server:

```
mkdir $CERT_CA_TMP
```

Step 3: On the Connector Server

Copy the `${STORES}/${TH}-cert-req` file (`%STORES%\%TH%-cert-req` on Windows platforms) from the connector to the Transformation Hub directory created above.

Step 4: On the Transformation Hub Server

Create the signed certificate, for example:

```
/bin/openssl x509 -req -CA ${INTERMEDIATE_CA_CRT} -CAkey ${INTERMEDIATE_CA_
KEY} -in ${TH}-cert-req -out ${CERT_CA_TMP}/${TH} -cert-signed-days 365 -
CAcreateserial -sha256
```

Step 5: On the Connector Server

1. Copy the `${TH}-cert-signed` certificate from the Transformation Hub to the connector's `${STORES}` directory. (On the Windows platform, copy the `%TH%-cert-signed` certificate to the connector's `%STORES%` directory.)
2. Copy the `ca.cert.pem` certificate from the Transformation Hub to the connector's `${STORES}` directory. (On the Windows platform, copy the certificate to the `%STORES%` directory.)
3. Copy the `intermediate.cert.pem` certificate from the Transformation Hub to the connector's `${STORES}` directory. (On the Windows platform, copy the certificate to the `%STORES%` directory.)
4. Import the CA certificate to the trust store, for example:

```
jre/bin/keytool -importcert -file ${STORES}/${CA_CERT} -alias CARoot
-keystore ${STORES}/${TH}.truststore.jks -storepass ${STORE_PASSWD}
```

On Windows platforms:

```
jre\bin\keytool -importcert -file %STORES%\%CA_CERT% -alias CARoot
-keystore %STORES%\%TH%.truststore.jks -storepass %STORE_PASSWD%
```

5. Import the intermediate certificate to the trust store, for example:

   ```
   jre/bin/keytool -importcert -file ${STORES}/${INTERMEDIATE_CA_CRT} -alias
   INTCARoot -keystore ${STORES}/${TH}.truststore.jks -storepass ${STORE_
   PASSWD}
   ```

   On Windows platforms:

   ```
   jre\bin\keytool -importcert -file %STORES%\%INTERMEDIATE_CA_CRT% -
   aliasINTCARoot -keystore %STORES%\%TH%.truststore.jks -storepass
   %STORE_PASSWD%
   ```

6. When prompted, enter **yes** to trust the certificate.
7. Import the CA certificate to the key store, for example:

   ```
   jre/bin/keytool -importcert -file ${STORES}/${CA_CERT} -alias CARoot -
   keystore ${STORES}/${TH}.keystore.jks -storepass ${STORE_PASSWD}
   ```

   On Windows platforms:

   ```
   jre\bin\keytool -importcert -file %STORES%\${CA_CERT} -alias CARoot -
   keystore %STORES%\%TH%.keystore.jks -storepass %STORE_PASSWD%
   ```

8. Import the intermediate certificate to the key store, for example:

   ```
   jre/bin/keytool -importcert -file ${STORES}/${INTERMEDIATE_CA_CRT} -alias
   INTCARoot -keystore ${STORES}/${TH}.keystore.jks -storepass ${STORE_
   PASSWD}
   ```

   On Windows platforms:

   ```
   jre\bin\keytool -importcert -file %STORES%\%INTERMEDIATE_CA_CRT% -alias
    INTCARoot -keystore %STORES%\%TH%.keystore.jks -storepass %STORE_
   PASSWD%
   ```

   When successfully completed, this command will return the message, *Certificate reply was installed in keystore*.

9. When prompted, enter **yes** to trust the certificate.
10. Import the signed certificate to the key store, for example:

    ```
    jre/bin/keytool -importcert -file ${STORES}/${TH}-cert-signed -alias ${TH}
    -keystore ${STORES}/${TH}.keystore.jks -storepass ${STORE_PASSWD}
    ```

    On Windows platforms:

    ```
    jre\bin\keytool -importcert -file %STORES%\%TH%-cert-signed -alias %TH%
    ```

```
-keystore %STORES%\%TH%.keystore.jks -storepass %STORE_PASSWD%
```

When successfully complete, this command will return the message, `Certificate reply was installed in keystore`.

11. Note the key store and trust store paths:

```
echo ${STORES}/${TH}.truststore.jks
echo ${STORES}/${TH}.keystore.jks
```

On Windows platforms:

```
echo %STORES%\%TH%.truststore.jks
echo %STORES%\%TH%.keystore.jks
```

12. Navigate to the `bin` directory and run agent setup. Install a connector with Transformation Hub as the destination, for example:

```
cd <installation dir>/current/bin
./runagentsetup.sh
```

On Windows platforms:

```
cd <installation dir>\current\bin
runagentsetup.bat
```

   a. When completing the Transformation Hub destination fields, use the same values as in Step 8 for the path and password.
   b. Set **Use SSL/TLS** to **true**.
   c. Set **Use SSL/TLS Authentication** to **true**.

13. Cleanup. Delete the following files:

Caution: The following files should be deleted to prevent the distribution of security certificates that could be used to authenticate against the Transformation Hub. These files are very sensitive and should not be distributed to other machines.

```
rm ${STORES}/${INTERMEDIATE_CA_CRT}
rm ${STORES}/intermediate.key.pem
rm ${STORES}/${TH}-cert-signed
rm ${STORES}/${TH}-cert-req
```

On Windows platforms:

```
del %STORES%\intermediate.cert.pem
del %STORES%\intermediate.key.pem
del %STORES%\%TH%-cert-signed
del %STORES%\%TH%-cert-req
```

Step 6: On the Transformation Hub Server

To clean up the Transformation Hub server, delete the temporary folder where the certificate was signed and the certificate and key files in `/tmp`.

Caution: The temporary certificate folder should be deleted to prevent the distribution of security certificates that could be used to authenticate against the Transformation Hub. These files are very sensitive and should not be distributed to other machines.

Configuring a SmartConnector with a Transformation Hub Destination without Client Authentication in FIPS Mode

Follow these steps to configure a SmartConnector with a Transformation Hub destination without client authentication in FIPS mode.

On the SmartConnector Server

1. Prepare the SmartConnector:
   - **If the connector is not yet installed:** Run the installer. After core software has been installed, you will see a window that lets you select **Add a Connector** or **Select Global Parameters**. Check **Select Global Parameters**, and on the window displayed, select **Set FIPS mode**. Set to **Enabled**.
   - **If the connector is already installed:** Run the installer. Select **Set Global Parameters** and then **Set FIPS Mode** to **Enabled**.
2. Navigate to the connector's `current` directory, for example:

   ```
   cd <install dir>/current
   ```
3. Set the environment variables for the static values used by keytool, for example:

   ```
   export CURRENT=<full path to this "current" folder>
   export BC_OPTS="-storetype BCFKS -providername BCFIPS -providerclass
   org.bouncycastle.jcajce.provider.BouncyCastleFipsProvider -providerpath
   ${CURRENT}/lib/agent/fips/bc-fips-1.0.0.jar
   -J-Djava.security.egd=file:/dev/urandom"
   export TH=<Transformation Hub hostname>_<Transformation Hub port>
   export STORES=${CURRENT}/user/agent/stores
   export STORE_PASSWD=changeit
   export CA_CERT=ca.cert.pem
   ```

   On Windows platforms:

   ```
   set CURRENT=<full path to this "current" folder>
   ```

```
set BC_OPTS="-storetype BCFKS -providername BCFIPS
-J-Djava.ext.dirs=%CURRENT%\jre\lib\ext;%CURRENT%\lib\agent\fips
-J-Djava.security.properties=%CURRENT%\user\agent\agent.security"
set TH=<Transformation Hub hostname>_<Transformation Hub port>
set STORES=%CURRENT%\user\agent\stores
set STORE_PASSWD=changeit
```

4. Create the `${CURRENT}/user/agent/stores` directory if it does not already exist, for example:

   `mkdir -p ${STORES}`

   On Windows platforms:

   `mkdir -p %STORES%`

5. Create a `ca.cert.pem` file with the contents of the root CA certificate with the following command: `${k8s-home}/scripts/cdf-updateRE.sh > /tmp/ca.cert.pm`
6. Copy the just-created `ca.cert.pem` file from the Transformation Hub to the connector's `${STORES}` directory. (On the Windows platform, copy the certificate to the `%STORES%` directory.)
7. Import the CA certificate to the trust store, for example:

   `jre/bin/keytool ${BC_OPTS} -importcert -file ${STORES}/${CA_CERT} -alias CARoot -keystore ${STORES}/${TH}.truststore.bcfips -storepass ${STORE_PASSWD}`

   On Windows platforms:

   `jre\bin\keytool %BC_OPTS% -importcert -file %STORES%\%CA_CERT% -alias CARoot -keystore %STORES%\%TH%.truststore.bcfips -storepass %STORE_PASSWD%`

8. When prompted, enter **yes** to trust the certificate.
9. Note the trust store path:

   `echo ${STORES}/${TH}.truststore.bcfips`

   On Windows platforms:

   `echo %STORES%\%TH%.truststore.bcfips`

10. Navigate to the `bin` directory and run agent setup. Install a connector with Transformation Hub as the destination, for example:

    `cd <installation dir>/current/bin`
    `./runagentsetup.sh`

    On Windows platforms:

```
cd <installation dir>\current\bin
runagentsetup.bat
```

a. When completing the Transformation Hub destination fields, use the value from Step 7 for the trust store path and the password used in Step 6 for the trust store password.

b. Set **Use SSL/TLS** to **true**.

c. Set **Use SSL/TLS Authentication** to **false**.

11. Cleanup. Delete the certificate file, for example:

Caution: The following file should be deleted to prevent the distribution of security certificates that could be used to authenticate against the Transformation Hub. These files are very sensitive and should not be distributed to other machines.

```
rm ${STORES}/${CA_CERT}
```

On Windows platforms:

```
del %\STORES%\ca.cert.pem
```

Troubleshooting SmartConnector Integration

The following troubleshooting tips may be useful in diagnosing SmartConnector integration issues.

Error Message	Issue
Unable to test connection to Kafka server: [Failed to construct kafka producer]	SmartConnector can't resolve the short or full hostname of the Transformation Hub node(s).
Unable to test connection to Kafka server: [Failed to update metadata after 30000 ms.]	SmartConnector can resolve the short or full hostname of the Transformation Hub node(s) but can't communicate with them because of routing or network issues.
Unable to test connection to Kafka server: [Failed to update metadata after 40 ms.]	You may have mistyped the topic name. Try re-entry.
Destination parameters did not pass the verification with error [; nested exception is: java.net.SocketException: Connection reset]. Do you still want to continue?	If using SSL/TLS, you did not configure the SSL/TLS parameters correctly.

Configuring Logger as a Transformation Hub Consumer

The procedure for configuring a Logger as a Transformation Hub consumer will depend on whether the Logger will be using SSL/TLS.

To configure a Logger as a Transformation Hub consumer (not using SSL/TLS):

1. Log in to Logger.
2. Select **Configuration > Receivers > Add.**
3. In the **Add Receiver** dialog, enter the following:
   - **Name:** Enter a unique name for the new receiver.
   - **Type:** Transformation Hub Receiver
4. Select and edit the Transformation Hub Receiver and enter the following parameters:
   - **Transformation Hub host(s) and port:** {Kafka broker Host IP 1}:9092, {Kafka broker Host IP 2}:9092, {Kafka broker Host IP 3}:9092
   - **Event Topic List:** th-cef (If additional topics are needed, enter multiple topics with a comma-separated list.)
   - **Retrieve event from earliest offset:** true
   - **Consumer Group (Logger Pool):** Logger Pool
   - **Use SSL/TLS:** false
   - **Use Client Authentication:** false
   - **Enable:** Checked

To configure a Logger as a Transformation Hub consumer (using SSL/TLS):

1. Log in to Logger.
2. Select **Configuration > Receivers > Add.**
3. In the **Add Receiver** dialog, enter the following:
   - **Name:** Transformation Hub Receiver
   - **Type:** Transformation Hub Receiver
4. Select and edit the Transformation Hub Receiver and enter the following parameters:
   - **Transformation Hub host(s) and port:** {Kafka broker Host IP 1}:9093, {Kafka broker Host IP 2}:9093, {Kafka broker Host IP 3}:9093
   - **Event Topic List:** th-cef (You can enter multiple topics with a comma-separated list.)

- **Retrieve event from earliest offset:** true
- **Consumer Group (Logger Pool):** Logger Pool
- **Use SSL/TLS:** true
- **Use Client Authentication:** false
- **Enable:** Checked

To configure a Logger as a Transformation Hub consumer (using SSL/TLS with Client Authentication):

1. Log in to Logger.
2. Select **Configuration > Receivers > Add.**
3. In the **Add Receiver** dialog, enter the following:
   - **Name:** Transformation Hub Receiver
   - **Type:** Transformation Hub Receiver
4. Select and edit the Transformation Hub Receiver and enter the following parameters:
   - **Transformation Hub host(s) and port:** {Kafka broker Host IP 1}:9093, {Kafka broker Host IP 2}:9093, {Kafka broker Host IP 3}:9093
   - **Event Topic List:** th-cef (You can enter multiple topics with a comma-separated list.)
   - **Retrieve event from earliest offset:** true
   - **Consumer Group (Logger Pool):** Logger Pool
   - **Use SSL/TLS:** true
   - **Use Client Authentication:** true
   - **Enable:** Checked

Troubleshooting

The following troubleshooting tips may be useful in diagnosing Logger integration issues.

Error Message	Issue
IP Address th1.example.com is not a valid address	Use IP addresses in Receiver configuration, not host names.
There was a problem contacting Transformation Hub: Timeout expired while fetching topic metadata, please check the receiver configuration	Logger can't communicate with Transformation Hub because of routing or network issues.
The specified Event Topic (th-<topicname>) is not valid	You have mistyped the topic name.

Note: This process is explained in more detail in the Logger Administrator's Guide, available from the Micro Focus support community.

Configuring ESM as a Transformation Hub Consumer

This procedure describes how to configure ESM as a Transformation Hub consumer with client authentication using a User (intermediate) certificate:

1. On Transformation Hub, run:

```
${K8S_HOME}/scripts/cdf-updateRE.sh write --re-key={path to intermediate certificate}/intermediate.key.pem --re-crt={path to intermediate certificate}/intermediate.cert.pem --re-ca={path to intermediate certificate}/ca.cert.pem
```

2. On an ESM host which has not been configured as a Transformation Hub consumer, switch to the manager directory:

```
cd /opt/arcsight/manager
```

3. Run each of these commands, one at a time. When prompted by the keytool for a password, enter the ESM password.

```
touch config/client.properties
```

```
bin/arcsight keytool -store clientkeys -storepasswd -storepass ""
```

```
bin/arcsight keytool -store clientkeys -keypasswd -keypass "" -alias services-cn
```

```
bin/arcsight changepassword -f config/client.properties -p ssl.keystore.password
```

4. Import the intermediate certificates to the ESM client keystore.
5. Run these commands:

```
bin/arcsight keytool -store clientcerts -importcert -file /tmp/ca.cert.pem -alias thcert
```

```
bin/arcsight keytool -store clientkeys -importcert -file /tmp/intermediate.cert.pem -alias thintcert
```

```
bin/arcsight keytool -store clientcerts -importcert -file /tmp/intermediate.cert.pem -alias thintcert
```

```
/etc/init.d/arcsight_services stop manager
```

```
bin/arcsight keytool -store clientkeys -genkeypair -dname "cn=<your CN>,ou=<your OU>, o=<your org short name>, c=<your country>" -keyalg rsa -keysize 2048 -alias th -startdate -1d -validity 366
```

```
bin/arcsight keytool -certreq -store clientkeys -alias th -file thkey.csr
```

6. Generate a certificate signing request (`.csr` file) so the Transformation Hub can sign a client certificate.
7. Copy the `.csr` file to the Transformation Hub initial master node.
8. On the Transformation Hub Initial Master Node, run:

```
openssl x509 -req -CA /opt/intermediate_cert_files/intermediate.cert.pem -CAkey /opt/intermediate_cert_files/intermediate.key.pem -in /opt/thkey.csr -out /opt/signedTHkey.crt -days 3650 -CAcreateserial -sha256
```

9. Copy the signed certificate to `/tmp` on the ESM host.
10. On the ESM host import the signed client certificate into the client keystore so it can be used to authenticate to Transformation Hub. Run these commands:

```
/opt/arcsight/manager/bin/arcsight keytool -store clientkeys -alias th -importcert -file /tmp/signedTHkey.crt -trustcacerts
```

11. Start the manager configuration wizard:

```
/opt/arcsight/manager/bin/arcsight managersetup
```

Note: if on a host without X Window access, run the `managersetup` command with the -i parameter. Consult the ESM documentation for more information regarding the `managersetup` command.

9. Proceed through the wizard for adding the Transformation hub to the ESM., until the dialog is displayed that prompts for a connection to Transformation Hub. On the dialog, under **"ESM can consume events from a Transformation Hub..."**, enter *Yes*. Then enter then the following parameters. (This will put an entry in the Manager `cacerts` file, displayed as `ebcaroot`):

Host:Port(s): `example.com:9093,192.example.com:9093,example.com:9093`

Note: You must use host names, not IP addresses. In addition, ESM does not support non-TLS port 9092.

Topic to read from: `th-binary_esm`

Path to Transformation Hub root cert:{leave this empty}

8. On the ESM, restart the ESM Manager:

```
/etc/init.d/arcsight_services stop manager
/etc/init.d/arcsight_services start manager
```

Chapter 6: Maintaining the Transformation Hub

Administration of the Transformation Hub cluster is performed from the Transformation Hub Kafka Manager Portal, available at `https:<Your high-availabiliy FQDN>:5443.`

Changing Transformation Hub Configuration Properties

To change Transformation Hub configuration properties:

1. In the Management Portal, select **Suite.**
2. Click **...** (Browse) on the far right and choose and choose **Reconfigure**. A new screen will be opened in a separate tab.
3. Update configuration properties as needed.
4. Click **Save.**

All services in the cluster affected by the configuration change will be restarted (in a rolling manner) across the cluster nodes.

Adding a Product (Capability)

To add a product (capability) to your cluster:

1. As explained under Upload Images, upload the offline images for the product you want to add.
2. Click **Suite.**.
3. Click **...** (Browse) on the far right and choose **Change**. A new screen will be opened in a separate tab.
4. On the next page, select a product you want to add, and click **Next**
5. On the **File Storage** page, fill in the NFS volume data if needed, and click **Next**.
6. Wait until the spinner disappears (This page will remain blank) and click **Next.**
7. Update configuration values if needed, and click **Next**.

After a short wait, the Configuration Complete page confirms the change to the cluster.

Removing a Product

To remove a product (capability) from your cluster:

1. Click **Suite**.
2. Click **... (Browse)** on the far right and choose **Install**. A new screen will be opened in a separate tab.
3. On the next page, deselect the product you want to remove, and click **Next**
4. On the **File Storage** page click **Next**.
5. Update configuration values if needed, and click **Next**.

After a short wait, the **Configuration Complete** page confirms the change to the cluster.

Uninstalling ArcSight Suite (including Transformation Hub)

To (gracefully) uninstall the ArcSight Suite (including Transformation Hub):

1. Stop all collectors and Connectors from sending events to Transformation Hub.
2. Stop all consumers from receiving events after they have consumed all events from their topics.
3. Click **Suite > Management**.
4. Click on the far right button and choose **Uninstall**.

The pods are progressively shut down and then uninstalled.

Resetting the Administrator Password

You can change the administrator password on a CDF installation.

1. Browse to CDF Installer UI at:
   `https://{master_FQDN or IP}:5443.`
2. Log in using admin USERID and the password you specified during the platform installation in the command line argument. (This URL is displayed at the successful completion of the CDF installation shown earlier.)
3. In the left navigation page, click **IdM Administration.**
4. In the main panel, click the large **SRG** button on the right.
5. In the left navigation bar, click **Users**.
6. In the list of users on the right, select *Admin* and click **Edit**.
7. In the bottom right, click **Remove Password.**

8. Click **Add Password.**
9. Enter a new admin password, and then click **Save**.

Viewing and Changing the Certificate Authority

The cluster maintains its own certificate authority (CA) to issue certificates for external communication. A self-signed CA is generated during installation by default. Pods of deployed products use the certificates generated by the CA on pod startup.

To display the current CA for external communication:

Run the following command on the Initial Master Node:

```
${K8S_HOME}/scripts/cdf-updateRE.sh read
```

To update the CA, run:

```
${K8S_HOME}/scripts/cdf-updateRE.sh write --re-key={New Intermediate Key Name}.pem --re-crt={New Intermediate Key Name}.pem --re-ca={New CA Cert Name}.pem}
```

Note: Changing the CA after Transformation Hub deployment will require undeploying and then deploying the Transformation Hub capability. This will result in a loss of configuration changes. As a result, it is highly recommended that if you need to perform this task, do so at the beginning of your Transformation Hub rollout. See the section on Deploying Transformation Hub for information on re-deploying the capability.

Manual Upgrade of CDF Version 2019.05 to CDF Version 2020.02

Transformation Hub 3.2.0 is supported on CDF version 2020.02. As a result, users running an earlier version of CDF (version 2019.05) must upgrade to version 2020.02. The manual CDF upgrade process, which is run on each node in your environment, is explained here.

Manual upgrade is a lengthy process and should be run with a stable and reliable SSH connection. The complete process includes two phases: upgrade from CDF 2019.05 to CDF 2019.08, and then an upgrade from 2019.08 to 2020.02.

Note: The upgrade of a single-master environment to a multi-master (high availability/HA) environment is not supported by this process.

Prerequisites

- Docker and Kubernetes versions must be upgraded separately.
  - CDF upgrade from 2019.05 to 2019.08 does ***not*** include the upgrade of Docker or Kubernetes versions.
  - CDF upgrade from 2019.08 to 2020.02 ***does include*** the upgrade of Kubernetes from v1.13.5 to v1.15.5.
  - CDF upgrade from 2019.08 to 2020.02 does i nclude the upgrade of Docker from v18.09.2 to v19.03.5-.3
- Verify that your environment meets the system requirements for a new cluster, as outlined in the ***CDF Deployment Guide***, including the following:
  - Linux OS version is RHEL/CentOS 7.5 , 7.6, or 7.7; Kernel version is 7.4 v3.10.0-693.21.1.el7 (or above).
  - Make sure you have enough space on all cluster nodes. The default value for the pod eviction threshold is 85% of used space for the filesystem where the Kubernetes home directory is mounted (by default `/opt/arcsight`). In addition, the cluster nodes should reserve 50 GB of disk space for upgrades, preferably under a different location than the Kubernetes home directory.
- The following packages must be installed on all master and worker nodes:

```
socat
container-selinux
```

Note: If these are not installed, then install each using the command: `yum install <packagename>`).

Preparation

1. Make sure that you have the permission to reboot the cluster nodes. You may need to reboot the nodes during the upgrade.
2. To make sure that all nodes (master nodes and worker nodes) are in running status, run: `kubectl get nodes`
3. To make sure all core pods are running and all necessary checks are passed, run: `${K8S_HOME}/bin/kube-status.sh` †
4. If you are using a non-root user to perform the manual upgrade, please verify that you have already configured your sudo permission from Appendix B.

Download the upgrade packages to each node

1. Download the CDF 2019.08 upgrade package to every node of your cluster into a `/tmp/upgrade-download` directory.

2. Download the CDF 2020.02 to every node of your cluster into the `/tmp/upgrade-download` directory.
3. Create a `/tmp/upgrade-backup` directory with a minimum size of 30GB on every master and worker node of your cluster. If you are a non-root user on the nodes inside the cluster, make sure you have permission to this directory: `mkdir /tmp/upgrade-backup)`.

Phase I: Upgrade process from 2019.05 to 2019.08

Beginning with the master nodes, upgrade your CDF infrastructure on every node of the cluster by running the following process ***on each node:***

1. Unzip the upgrade package on each node by running these commands:

```
cd /{download-directory}
unzip cdf-upgrade-2019.08.xxxx
```

2. Run the following commands on each node:

```
cd {download-directory}/cdf-upgrade-2019.08.xxxx
./upgrade.sh -i -t <path to upgrade directory>
```

   Examples:

```
./upgrade.sh -i -t /tmp/upgrade-backup
```

```
./upgrade.sh -i
```

3. ***On the initial master node,*** run the following commands to upgrade CDF components:

```
cd {download-directory}/cdf-upgrade-2019.08.xxxx
```

```
./upgrade.sh -u
```

4. Optionally, clean the unused docker images by running the following commands on all nodes (masters and workers):

```
cd {download-directory}/cdf-upgrade-2019.08.xxxx
```

```
./upgrade.sh -c
```

5. Verify the cluster status. First, check the CDF version on each node by running the command:

```
cat ${K8S_HOME}/version.txt
>> 2019.08.00125
```

6. Check the status of CDF on each node by running these commands:

```
cd ${K8S_HOME}/bin/
./kube-status.sh'
```

Phase II: Upgrade process from 2019.08 to 2020.02

Beginning with the master nodes, upgrade your CDF infrastructure on every node of the cluster by running the following process ***on each node:***

1. On one of the master nodes, configure IDM pod affinity and wait until IDM pods are up and running with this command:

```
kubectl patch deployment idm -n core --patch '{ "spec": { "template": { "spec": { "affinity": { "podAffinity": { "preferedDuringSchedulingIgnoredDuringExecution": [ { "labelSelector": { "matchExpressions": [ { "key": "app", "operator": "In", "values": [ "idm-app" ] } ] }, "topologyKey": "kubernetes.io/hostname" } ] } } } } } }'
```

Note: This step need only be performed on a single master node.

2. Unzip the upgrade package on each node by running these commands:

```
cd /{download-directory}
unzip cdf-2020.02.xxxx
```

3. Run the following commands on each node:

```
cd {download-directory}/cdf-2020.02.xxxx
./upgrade.sh -i -t <path to upgrade directory>
```

Examples:

```
./upgrade.sh -i -t /tmp/upgrade-backup
```

```
./upgrade.sh -i
```

4. Run the following commands to upgrade CDF components:

```
cd {download-directory}/cdf-2020.02.xxxx
```

```
./upgrade.sh -u
```

5. Optionally, clean the unused docker images by running the following commands on all nodes (masters and workers):

```
cd {download-directory}/cdf-2020.02.xxxx
```

```
./upgrade.sh -c
```

6. Verify the cluster status. First, check the CDF version on each node by running the command:

```
cat ${K8S_HOME}/version.txt
>> 2020.02.00120
```

7. Check the status of CDF on each node by running these commands:

```
cd ${K8S_HOME}/bin/
./kube-status.sh'
```

Automated Upgrade to CDF 2020.02

The automatic upgrade has 2 phases: the first for upgrade from CDF 2019.05 to CDF 2019.08, and the second for upgrade from CDF 2019.08 to 2020.02. The automated upgrade to CDF 2020.02 is run with a single command and requires no interaction until completion of each phase. Typically, each automated upgrade phase takes around 1 hour for a 3x3 cluster.

Preparing the Upgrade Manager

Automatic upgrade should be run from a host (for purposes of these instructions, known as the upgrade manager). The upgrade manager (UM) may be one of the following host types:

- One of the cluster nodes
- A host outside the cluster (a secure network location)

Configure Passwordless Communication: You must configure passwordless SSH communication between the UM and all the nodes in the cluster, as follows:

1. On the UM, run the following command to generate a key pair:
   `ssh-keygen -t rsa`
2. On the UM, run the following command to share the generated public key to each node of your cluster, including the node which you are logged into:
   `ssh-copy-id -i ~/.ssh/id_rsa.pub root@<node_fqdn_or_ip>`

Download Upgrade: Next, download the upgrade files for CDF 2018.08 and CDF 2020.02 to a download directory (referred to as `<download_directory>`) on the UM.

There are 4 directories involved in the auto-upgrade process:

- An auto-upgrade directory /`tmp/autoUpgrade` will be auto generated on the UM. It will store the upgrade process steps and logs.
- A backup directory `/tmp/CDF_201905_upgrade` will be auto generated on every node. (approximate size 1.5 GB)
- A backup directory `/tmp/CDF_201908_upgrade` will be auto generated on every node. (approximate size 1.7 GB)
- A working directory will be auto generated on the UM and every node at the location provided by the `-d` parameter The upgrade package will be copied to this directory. (approximate size 9 GB). The directory will be automatically deleted after the upgrade.

Note: The working directory can be created manually on UM and every node and then passed as `-d` parameter to the auto-upgrade script. If you are a non-root user on the nodes inside the cluster, make sure you have permission to this directory.

Phase I: Auto-upgrade from CDF 2019.05 to CDF 2019.08

On the upgrade manager, run the following commands:

```
cd {download-directory}
unzip cdf-upgrade-2019.08.xxxx.zip
cd /cdf-upgrade-2019.08.xxxx
./autoUpgrade.sh -d /path/to/workinig_directory -n {any_cluster_node_adress_
or_ip}
```

Example:

```
./autoUpgrade.sh -d /tmp/upgrade -n pueas-ansi-node1.swinfra.net
```

Phase II: Auto-upgrade from CDF 2019.08 to CDF 2020.02

Proceed with the second phase of the automated upgrade, as follows:

1. Remove the 2019.08 upgrade directory:
   `rm -rf {download-directory}/cdf-upgrade-2019.08.xxxx`
2. On one of the master nodes, configure IDM pod affinity and wait until IDM pods are up and running with this command:

```
   kubectl patch deployment idm -n core --patch '{ "spec": { "template": {
"spec": { "affinity": { "podAffinity": {
"preferedDuringSchedulingIgnoredDuringExecution": [ { "labelSelector": {
"matchExpressions": [ { "key": "app", "operator": "In", "values": [ "idm-app"
] } ] }, "topologyKey": "kubernetes.io/hostname" } ] } } } } } }'
```

3. Run the CDF 2020.02 auto-upgrade by executing these commands:
   ```
   cd {download-directory}
   unzip cdf-2020.02.xxxx.zip
   cd cdf-2020.02.xxxx
   ./autoUpgrade.sh -d /path/to/workinig_directory -n {any_cluster_node_
   adress_or_ip}
   ```

Example:

```
./autoUpgrade.sh -d /tmp/upgrade -n pueas-ansi-node1.swinfra.net
```

Remove the auto-upgrade temporary directory from UM

The auto-upgrade temporary directory contains the upgrade steps and logs. If you want to upgrade another cluster from the same UM, remove that directory with this command:

```
rm -rf /tmp/autoUpgrade
```

In Case of Automatic Upgrade Failure

- If the automatic upgrade fails, run **`autoUpgrade.sh`** again as outlined above. The process may take several attempts to succeed.
- In some cases, the automatic upgrade may return an error message about the upgrade process still running and the existence of a `*.lock` file which prevents autoupgrade.sh to continue. This file is automatically deleted in a few minutes. Alternatively, you can manually delete this file. Once the file is deleted either automatically or manually, run **`autoUpgrade.sh`** again.
- If the automated upgrade process is still unsuccessful, continue the process on the failed node using the procedure outlined in Manual Upgrade to CDF 2020.02.

Upgrading ArcSight Suite

Follow these steps to upgrade your Transformation Hub to the latest version.

Note: A properly-performed upgrade of ArcSight suite will not interrupt the flow of events from producers, through Transformation Hub, to the consumers, as long as the Transformation Hub environment includes more than 1 Kafka broker. No event data will be lost in this situation.

Prerequisite

Before proceeding, you should have accepted the browser certificate at least once. If not, you should take these steps now.

To ensure that you have accepted the certificate:

1. In the CDF management portal, select **Suite > Management.**
2. Click **... (Browse)** and then choose **Reconfigure.**
3. Accept the browser certificate.

Process Outline

The complete process of upgrade consists of these steps:

1. Download the upgrade files.
2. Add the new metadata file.
3. Initiate the upgrade process.
4. Upload the offline images to the Docker registry.
5. Finalize and complete the upgrade process.

These steps are explained in detail below.

Download the Upgrade Files

Download the metadata and product images files to a secure network location.

- **Metadata:** `arcsight-installer-metadata-<build number>.tar`
  - **Example**: `arcsight-installer-metadata-2.2.0.9.tar`
- **Offline Images:** `<product>-<build number>.tar`
  - **Example:**`transformationhub-3.2.0.9.tar`

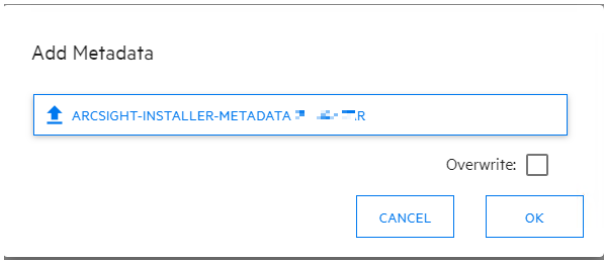
After downloading, unpack the offline images tar file:

```
tar -xvf <product-name>-<build number>.tar
```

Example:

```
tar -xvf transformationhub-3.2.0.9.tar
```

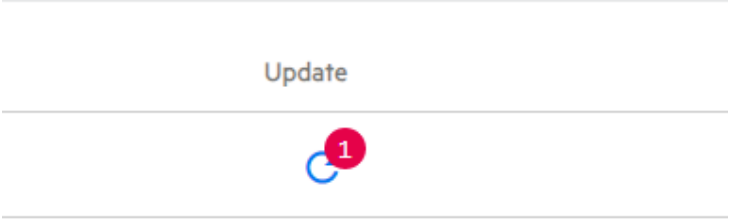
Add New Metadata



1. Copy the file `arcsight-installer-metadata` to the server from which you will be logging into the CDF UI.
2. Log into the CDF management portal from the host where you copied the file in Step 1.
3. In the left navigation pane, click **ADMINISTRATION**, then select **DEPLOYMENT> Metadata.**
4. Click **+ Add.**
5. Select your previously downloaded metadata file (`arcsight-installer-metadata-<upgrade version>.<Build number>.tar`).

Start the Upgrade Process

1. In the CDF management portal, select **DEPLOYMENT > Deployments.**
2. In the **Update** column, click the number 1 inside the red circle. Then, choose your recently-added metadata file version to initiate the upgrade.



3. On the **Update to** page, click **Next.**
4. On the **Transfer images page,** click **Next.**
5. On the **Import suite images** page, click **more** to verify the filenames of the expected images for the next step.(which will be in the format `3.2.<upgrade version>.<build number>`).

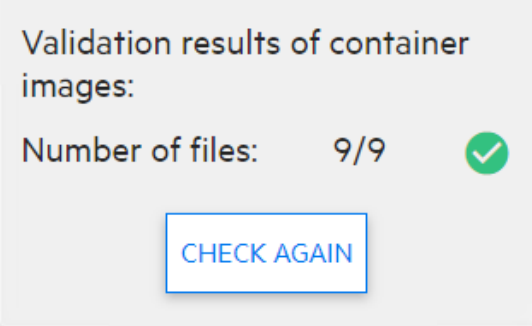
Upload Offline Images

1. `cd {K8S_HOME}/scripts`
   Example: `cd /opt/arcsight/kubernetes/scripts`
2. `./uploadimages.sh -u registry-admin -p {your CDF admin password} -d <extracted product directory path>`
   Example: `./uploadimages.sh -u registry-admin -p <password> -d opt/upgrade/transformationhub-3.2.0.9`

Note: Your CDF admin password is the password you were required to change when initially logging into the CDF UI.

Finalize and Complete

1. In the CDF management portal, return to the **Import suite images** page.
2. Click the **Check again** button to verify that all the required images are available and that the **Next** button is enabled. (There should be 9 of 9 images found and a green icon displayed.)Then click **Next**.



3. On the **Configure storage** click **Next**. The upgrade configuration container is being deployed to the cluster while the next page loads..
4. On the **Product upgrade** page click **Next**. Now the process of upgrading Transformation Hub pods has started. (You can monitor the process by running the command: `kubectl get pods -n {suite-namespace}`).

When the process has finished successfully, the upgrade is complete.

To verify upgrade success, browse to **DEPLOYMENT > Deployments** and check that the number in the Version column corresponds to your upgrade version.

Note: If you have successfully completed the upgrade image upload and the green icon is not displayed, wait 2 minutes and then click the **Check Again** button. If there are missing files, click **More** to determine the missing files and make sure all images are successfully deployed.

Upgrade Returns `INTERNAL SERVER ERROR`

After a successful upgrade of CDF, Transformation Hub, after attempting to reinstall Transformation Hub, the installer will display this error message on the **Configuration/Deployment** page. If this error is encountered, follow this procedure to resolve the issue:

1. Run this command:
   ```
   kubectl delete -n core $(kubectl get pods -n core -o name | grep itom-postgresql-default)
   ```
2. Wait for the pod to enter the ***Running*** state.

```
kubectl get pods -o wide -n core | grep itom-postgresql-default
```

3. On the **Configuration/Deployment** page, click **Deploy** again to deploy the product.

Appendix A: CDF Installer Script `install.sh` Command Line Arguments

Argument	Description
`--auto-configure-firewall`	Flag to indicate whether to auto configure the firewall rules during node deployment. The allowable values are true or false. The default is true.
`--cluster-name`	Specifies the logical name of the cluster.
`--deployment-log-location`	Specifies the absolute path of the folder for placing the log files from deployments.
`--docker-http-proxy`	Proxy settings for Docker. Specify if accessing the Docker hub or Docker registry requires a proxy. By default, the value will be configured from the `http_proxy` environment variable on your system.
`--docker-https-proxy`	Proxy settings for Docker. Specify if accessing the Docker hub or Docker registry requires a proxy. By default, the value will be configured from `https_proxy` environment variable on your system
`--docker-no-proxy`	Specifies the IPv4 addresses or FQDs that do not require proxy settings for Docker. By default, the value will be configured from the `no_proxy` environment variable on your system.
`--enable_fips`	This parameter enables suites to enable and disable FIPS. The expected values are true or false. The default is ***false***.
`--fail-swap-on`	If 'swapping' is enabled, specifies whether to make the kubelet fail to start. Set to true or false. The default is ***true***.
`--flannel-backend-type`	Specifies flannel backend type. Supported values are vxlan and host-gw. The default is host-gw.
`--ha-virtual-ip`	A Virtual IP (VIP) is an IP address that is shared by all master nodes. The VIP is used for the connection redundancy by providing failover for one machine. Should a master node fail, another master node takes over the VIP address and responds to requests sent to the VIP. Mandatory for a Multi-Master cluster; not applicable to a single-master cluster The VIP must be resolved (forward and reverse) to the VIP Fully Qualified Domain Name (FQDN)
`--k8s-home`	Specifies the absolute path of the directory for the installation binaries. By default, the Kubernetes installation directory is /opt/arcsight/kubernetes.
`--keepalived-nopreempt`	Specifies whether to enable nopreempt mode for KeepAlived. The allowable value of this parameter is true or false. The default is true and KeepAlived is started in nopreempt mode.

Argument	Description
`--keepalived-virtual-router-id`	Specifies the virtual router ID for KEEPALIVED. This virtual router ID is unique for each cluster under the same network segment. All nodes in the same cluster should use the same value, between 0 and 255. The default is 51.
`--kube-dns-hosts`	Specifies the absolute path of the hosts file which used for host name resolution in a non-DNS environment. **Note:** Although this option is supported by the CDF Installer, its use is strongly discouraged to avoid using DNS resolutuion in production environments due to hostname resolution issues and nuances involved in their mitigations.
`--load-balancer-host`	IP address or host name of load balancer used for communication between the master nodes. For a multiple master node cluster, it is required to provide `–load-balancer-hos`t or `–ha-virtual-ip` arguments.
`--master-api-ssl-port`	Specifies the https port for the Kubernetes (K8S) API server. The default is 8443.
`--nfs-folder`	Specifies the path to the ITOM core volume.
`--nfs-server`	Address of the NFS host.
`--pod-cidr-subnetlen`	Specifies the size of the subnet allocated to each host for pod network addresses. For the default and the allowable values see the CDF Planning Guide.
`--pod-cidr`	Specifies the private network address range for the Kubernetes pods. Default is 172.16.0.0/16. The minimum useful network prefix is /24. The maximum useful network prefix is /8. This must not overlap with any IP ranges assigned to services (see `--service-cidr` parameter below) in Kubernetes. The default is 172.16.0.0/16. For the default and allowable values see the CDF Planning Guide.
`--registry_orgname`	The organization inside the public Docker registry name where suite images are located. Not mandatory. Choose one of the following: • Specify your own organization name (such as your company name). For example: `--registry-orgname=Mycompany.` • Skip this parameter. A default internal registry will be created under the default name HPESWITOM.
`--runtime-home`	Specifies the absolute path for placing Kubernetes runtime data. By default, the runtime data directory is `${K8S_HOME}/data.`
`--service-cidr`	Kubernetes service IP range. Default is 172.30.78.0/24. Must not overlap the POD_CIDR range. Specifies the network address for the Kubernetes services. The minimum useful network prefix is /27 and the maximum network prefix is /12. If SERVICE_CIDR is not specified, then the default value is 172.17.17.0/24. This must not overlap with any IP ranges assigned to nodes for pods. See `--pod-cidr.`

Argument	Description
`--skip-check-on-node-lost`	Option used to skip the time synchronization check if the node is lost. The default is true.
`--skip-warning`	Option used to skip the warnings in precheck when installing the Initial master Node. Set to true or false. The default is false.
`--system-group-id`	The group ID exposed on server; default is 1999.
`--system-user-id`	The user ID exposed on server; default is 1999.
`--thinpool-device`	Specifies the path to the Docker devicemapper, which must be in the `/dev/mapper/` directory. For example: `/dev/mapper/docker-thinpool`
`--tmp-folder`	Specifies the absolute path of the temporary folder for placing temporary files. The default temporary folder is `/tmp.`
`-h, --help`	Displays a help message explaining proper parameter usage
`-m, --metadata`	Specifies the absolute path of the tar.gz suite metadata packages.

Appendix B: Creating an Intermediate Key and Certificate

This appendix details the process for creating an intermediate key and certificate (cert) file. It contains the following sections:

Best Practice: In order to import an intermediate (user) certificate, the Transformation Hub must be deployed, the user certificate imported, undeployed, and then re-deployed to register the certificate change. We recommend that you perform this procedure when Transformation Hub is first installed to avoid downtime and data loss.

The Transformation Hub deployment process comprises these steps:

1. Install CDF.
2. Deploy Transformation Hub with default settings.
3. Perform the operations described here to create the intermediate certificate (detailed below).
4. Update the TH RE certificate (detailed below) .
5. From the CDF UI, uninstall the Transformation Hub.
6. After the Transformation Hub is uninstalled, redeploy the Transformation Hub with client authentication enabled .Configure your pre-deployment parameters as desired.

To obtain the contents of the RE certificate, use the following script:

```
${K8S_HOME}/scripts/cdf-updateRE.sh
```

The Transformation Hub CA (certificate authority) private key is not exposed. Therefore, in order to create a signed certificate for use in adding connector destinations or other consumers which are configured with client authentication enabled, you will need to create and update the Transformation Hub with an intermediate key, or, alternatively, use your organization's intermediate key.

Create a New CA Certificate (example)

1. Make the directory and configure:

```
mkdir /root/ca
```

```
cd /root/ca
```

```
mkdir certs crl

newcerts private

chmod 700 private

touch index.txt

echo 1000 > serial
```

2. Create the configuration file in a text editor (`vi /root/ca/openssl.cnf`), and then add the following contents (values shown are examples; change parameter values to match your requirements):

```
# OpenSSL root CA configuration file.

# Copy to `/root/ca/openssl.cnf`.

[ ca ]

default_ca = CA_default

[ CA_default ]

# Directory and file locations.

dir               = /root/ca

certs             = $dir/certs

crl_dir           = $dir/crl

new_certs_dir     = $dir/newcerts

database          = $dir/index.txt

serial            = $dir/serial

RANDFILE          = $dir/private/.rand

# The root key and root certificate.

private_key       = $dir/private/ca.key.pem

certificate       = $dir/certs/ca.cert.pem

# For certificate revocation lists.

crlnumber         = $dir/crlnumber

crl               = $dir/crl/ca.crl.pem

crl_extensions    = crl_ext

default_crl_days  = 30
```

```
# SHA-1 is deprecated, so use SHA-2 instead.
default_md        = sha256
name_opt          = ca_default
cert_opt          = ca_default
default_days      = 375
preserve          = no
policy            = policy_strict
[ policy_strict ]
# The root CA should only sign intermediate certificates that match.
# See the POLICY FORMAT section of `man ca`.
countryName             = match
stateOrProvinceName     = match
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
[ policy_loose ]
# Allow the intermediate CA to sign a more diverse range of certificates.
# See the POLICY FORMAT section of the `ca` man page.
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
[ req ]
# Options for the `req` tool (`man req`).
```

```
default_bits        = 2048
distinguished_name  = req_distinguished_name
string_mask         = utf8only
# SHA-1 is deprecated, so use SHA-2 instead.
default_md          = sha256
# Extension to add when the -x509 option is used.
x509_extensions     = v3_ca
[ req_distinguished_name ]
countryName                     = Country
stateOrProvinceName             = State
localityName                    = Locality
0.organizationName              = EntCorp
organizationalUnitName          = OrgName
commonName                      = Common Name
emailAddress                    = Email Address
# Optionally, specify some defaults.
countryName_default             = <your country code>
stateOrProvinceName_default     = <your state or province>
localityName_default            =
0.organizationName_default      = <your company name>
organizationalUnitName_default  =
emailAddress_default            =
[ v3_ca ]
# Extensions for a typical CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign
```

```
[ v3_intermediate_ca ]
# Extensions for a typical intermediate CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true, pathlen:0
keyUsage = critical, digitalSignature, cRLSign, keyCertSign
[ usr_cert ]
# Extensions for client certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = client, email
nsComment = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth, emailProtection
[ server_cert ]
# Extensions for server certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = server
nsComment = "OpenSSL Generated Server Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
[ crl_ext ]
# Extension for CRLs (`man x509v3_config`).
authorityKeyIdentifier=keyid:always
[ ocsp ]
```

```
# Extension for OCSP signing certificates (`man ocsp`).

basicConstraints = CA:FALSE

subjectKeyIdentifier = hash

authorityKeyIdentifier = keyid,issuer

keyUsage = critical, digitalSignature

extendedKeyUsage = critical, OCSPSigning
```

3. Generate the new CA root key:

```
cd /root/ca
```

And then run the following:

```
openssl genrsa -out private/ca.key.pem 4096

chmod 400 private/ca.key.pem
```

4. Create the new CA cert:

```
openssl req -config openssl.cnf \  -key private/ca.key.pem \  -new -x509 -
days 365 -sha256 -extensions v3_ca \ -out certs/ca.cert.pem
```

5. Verify the root CA:

```
chmod 444 certs/ca.cert.pemopenssl x509 -noout -text -in certs/ca.cert.pem
```

Create a New Intermediate Key and Certificate

1. Make the directory and configure:

```
mkdir /root/ca/intermediate/

cd /root/ca/intermediate

mkdir certs crl csr newcerts private

chmod 700 private

touch index.txt

echo 1000 > serial

echo 1000 > /root/ca/intermediate/crlnumber
```

2. Open the configuration file in a text editor (`vi /root/ca/openssl.cnf`), and then add the following contents (values shown are examples; change parameter values to match your requirements). Make sure the directory is unique for each intermediate CA.

```
[ ca ]
default_ca = CA_default
[ CA_default ]
# Directory and file locations.
dir               = /root/ca/intermediate
certs             = $dir/certs
crl_dir           = $dir/crl
new_certs_dir     = $dir/newcerts
database          = $dir/index.txt
serial            = $dir/serial
RANDFILE          = $dir/private/.rand
# The root key and root certificate.
private_key       = $dir/private/intermediate.key.pem
certificate       = $dir/certs/intermediate.cert.pem
# For certificate revocation lists.
crlnumber         = $dir/crlnumber
crl               = $dir/crl/intermediate.crl.pem
crl_extensions    = crl_ext
default_crl_days  = 30
# SHA-1 is deprecated, so use SHA-2 instead.
default_md        = sha256
name_opt          = ca_default
cert_opt          = ca_default
default_days      = 375
preserve          = no
policy            = policy_loose
[ policy_strict ]
# The root CA should only sign intermediate certificates that match.
```

```
# See the POLICY FORMAT section of `man ca`.
countryName             = match
stateOrProvinceName     = match
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
[ policy_loose ]
# Allow the intermediate CA to sign a more diverse range of certificates.
# See the POLICY FORMAT section of the `ca` man page.
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
[ req ]
# Options for the `req` tool (`man req`).
default_bits        = 2048
distinguished_name  = req_distinguished_name
string_mask         = utf8only
# SHA-1 is deprecated, so use SHA-2 instead.
default_md          = sha256
# Extension to add when the -x509 option is used.
x509_extensions     = v3_ca
[ req_distinguished_name ]
# See <https://en.wikipedia.org/wiki/Certificate_signing_request>.
```

```
countryName                     = Country Name (2 letter code)
stateOrProvinceName             = State or Province Name
localityName                    = Locality Name
0.organizationName              = Organization Name
organizationalUnitName          = Organizational Unit Name
commonName                      = Common Name
emailAddress                    = Email Address
# Optionally, specify some defaults.
countryName_default             = <your country code>
stateOrProvinceName_default     = <your state or province>
localityName_default            =
0.organizationName_default      = <your company name>
organizationalUnitName_default  =
emailAddress_default            =
[ v3_ca ]
# Extensions for a typical CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign
[ v3_intermediate_ca ]
# Extensions for a typical intermediate CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true, pathlen:0
keyUsage = critical, digitalSignature, cRLSign, keyCertSign
[ usr_cert ]
# Extensions for client certificates (`man x509v3_config`).
```

```
basicConstraints = CA:FALSE
nsCertType = client, email
nsComment = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth, emailProtection
[ server_cert ]
# Extensions for server certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = server
nsComment = "OpenSSL Generated Server Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
[ crl_ext ]
# Extension for CRLs (`man x509v3_config`).
authorityKeyIdentifier=keyid:always
[ ocsp ]
# Extension for OCSP signing certificates (`man ocsp`).
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, digitalSignature
extendedKeyUsage = critical, OCSPSigning
```

3. Generate the new Intermediate CA key:

```
cd /root/ca
```

```
openssl genrsa -out intermediate/private/intermediate.key.pem 4096
```

4. Create the intermediate CA signing request (CSR):

```
chmod 400 intermediate/private/intermediate.key.pem

openssl req -config intermediate/openssl.cnf -new -sha256 \ -key
intermediate/private/intermediate.key.pem \ -out
intermediate/csr/intermediate.csr.pem
```

5. Create the new intermediate CA cert:

```
cd /root/ca

openssl ca -config openssl.cnf -extensions v3_intermediate_ca \ -days 3650
-notext -md sha256 \ -in intermediate/csr/intermediate.csr.pem \ -out
intermediate/certs/intermediate.cert.pem

# Sign the certificate? [y/n]: y

chmod 444 intermediate/certs/intermediate.cert.pem
```

6. Verify the intermediate CA:

```
openssl x509 -noout -text \ -in intermediate/certs/intermediate.cert.pem
```

7. Verify the intermediate certificate against the root certificate

```
openssl verify -CAfile certs/ca.cert.pem \

intermediate/certs/intermediate.cert.pem

# intermediate.cert.pem: OK
```

8. Verify the intermediate CA against the root CA:

```
openssl verify -CAfile certs/ca.cert.pem \

intermediate/certs/intermediate.cert.pem

# intermediate.cert.pem: OK
```

Update the Certificate Set on the Transformation Hub Cluster

In order to update the CA cert used by Transformation Hub, copy your intermediate key and intermediate cert along with the CA cert from the server where they were created to the Initial Master Node, and then do the following:

1. Run:
   ```
   ${K8S_HOME}/scripts/cdf-updateRE.sh write --re-key=/tmp/intermediate.key.pem --re-crt=/tmp/intermediate.cert.pem --re-ca=/tmp/ca.cert.pem
   ```

Note: The path to the `intermediate.key.pem, intermediate.cert.pem,` and `ca.cert.pem` can be any path desired

2. From the CDF UI, uninstall the Transformation Hub.
3. After the Transformation Hub is uninstalled, redeploy the Transformation Hub.

Appendix C: Troubleshooting

The following troubleshooting tips may be helpful in resolving issues with your Transformation Hub cluster.

Issue: Installation of Master Nodes Fails

During installation, installation of Master Nodes can fail with the error:

```
Unable to connect to the server: context deadline exceeded
```

If this occurs, make sure that your `no_proxy` and `NO_PROXY` variables include valid virtual IP addresses and hostnames for each of the master and worker nodes in the cluster, as well as the NFS server.

Issue: Installation Times Out

During installation, the process may time out with the error:

```
Configure and start ETCD database
```

If this occurs, make sure your `no_proxy` and `NO_PROXY` variables include correct Master Node information.

Issue: During `sudo` Installation, Worker Node Fails to Install

During the Add Node phase, if one or more of the worker nodes fails to install and the log shows the following error message:

```
[ERROR] : GET Url: https://itom-vault.core:8200/v1/***/PRIVATE_KEY_
CONTENT_{hostname}_{sudo user}, ResponseStatusCode: 404
```

You can take the following steps to rectify the issue:

1. Click **Cancel**. This takes you back to the version selection screen
2. Go through the installation screens again (all previous data is preserved).
3. On the **Add Node** screen, where you added the Worker Node data, remove the worker node which failed by clicking on the **Delete** icon.
4. Click **Add Node** and add the node again.
5. Click **Next** and proceed with the installation.

Issue: Cluster List Empty in Kafka Manager

If cluster list is empty in the Kafka Manager UI, delete the existing Kafka Manager pod and try the UI again after a new Kafka Manager pod is back to the Running state.

Issue: Worker Nodes Out of Disk Space and Pods Evicted

If the worker nodes run out of disk space, causing the pods on the node to go into Evicted status, try one of the following steps:

- Fix the disk space issue by adding an additional drive, or by removing unnecesarry files.
- On the the node where the low disk space occurred, run the following command:

```
{install dir} /kubernetes/bin/kube-restart.sh
```

Refer to "Configuring Hard Eviction Thresholds for Worker Nodes" in the ***Transformation Hub Administrator's Guide*** for information on adjusting the eviction threshold.

Glossary

A

Avro

Avro is a row-oriented remote procedure call and data serialization framework developed within Apache's Hadoop project. It uses JSON for defining data types and protocols, and serializes data in a compact binary format.

C

Cluster

A group of nodes, pods, or hosts.

Common Event Format (CEF)

CEF is an open log management standard that simplifies log management, letting third parties create their own device schemas that are compatible with a standard that is used industry-wide for normalizing security events. Technology companies and customers can use the standardized CEF format to facilitate data collection and aggregation, for later analysis by an enterprise management system.

Connectors in Transformation Hub (CTH)

A feature where SmartConnector technology operates directly in Transformation Hub to collect data. CTH enables the enriching, normalizing and sending of syslog data and routing it to Kafka topics.

Consumer

A consumer of Transformation Hub event data. Consumers may be Micro Focus products such as Logger or ESM, third-party products like Hadoop, or can be made by customers for their own use.

Container Deployment Foundation (CDF)

CDF is the container-based delivery and management model built on Docker containers managed by Kubernetes, which standardizes distribution, installation, upgrade, and operation of Micro Focus products and product suites.

D

Dedicated Master Node

A node dedicated to running the Kubernetes control plane functionality only.

Destination

In Micro Focus products, a forwarding location for event data. A Transformation Hub topic is one example of a destination.

Docker Container

A Docker container is a portable application package running on the Docker software development platform. Containers are portable among any system running the Linux operating system.

F

flannel

flannel (spelled with a lower-case f) is a virtual network that gives a subnet to each host for use with container runtimes. Platforms like Google's Kubernetes assume that each container (pod) has a unique, routable IP inside the cluster. The advantage of this model is that it reduces the complexity of doing port mapping.

Fully Qualified Domain Name (FQDN)

A fully qualified domain name (FQDN) is the complete domain name for a specific computer, or host, on the internet. The FQDN consists of two parts: the hostname and the domain name. For example, an FQDN for a hypothetical mail server might be mymail.example.com. The hostname is mymail, and the host is located within the domain example.com.

I

Initial Master Node

The Master Node that has been designated as the primary Master Node in the cluster. It is from this node that you will install the cluster infrastructure.

K

Kafka

An open-source messaging system that publishes messages for subscribers to consume on its scalable platform built to run on servers. It is commonly referred to as a message broker.

Kubernetes

Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications. It groups containers that make up an application into logical units for easy management and discovery.

L

Labeling

Adding a Kubernetes label to a Master or Worker Node creates an affinity for the workload to the Master or Worker Node, enabling the node to run the specified workload on the labeled server.

Local Docker Registry

The Docker Registry location on the Master and Worker Nodes in the cluster. Application software is launched and managed from the Local Docker Registry.

M

Master Nodes

Master Nodes run the CDF Installer and process web services calls made to the cluster. A minimum of 1 Master Node is required for each cluster.

N

Network File System (NFS)

This is the location where the CDF Installer, Transformation Hub, and other components may store persistent data. A customer-provisioned NFS is required. This environment is referred to in this documentation as an "external" NFS. Although the CDF platform can host a CDF-provisioned NFS (Internal NFS), for high availability an External NFS service should implemented.

Node

A processing location. In CDF containerized applications, nodes come in two types: master and worker.

P

Pod

Applications running in Kubernetes are defined as “pods”, which group containerized components. CDF uses Docker Containers as these components. A pod consists of one or more containers that are guaranteed to be co-located on the host

server and can share resources. Each pod in Kubernetes is assigned a unique IP address within the cluster, allowing applications to use ports without the risk of conflict.

Producer

A gatherer of event data, such as a SmartConnector or CTH. Typically data from a producer is forwarded to a destination such as a Transformation Hub topic.

R

Root Installation Folder

The root installation folder is the top-level directory that the Transformation Hub, CDF Installer, and all supporting product files will be installed into. The default setting is /opt/arcsight. It is referred to as RootFolder in this document, supporting scripts, and installation materials.

S

Shared Master and Worker Nodes

A configuration where both Master and Worker Node functionality resides on the same host. This is not a recommended architecture for high availabillty.

SmartConnector

SmartConnectors automate the process of collecting and managing logs from any device and in any format.

T

Thinpool

Using thin provisioning in Docker, you can manage a storage pool of free space, known as a thinpool, which can be allocated to an arbitrary number of devices when needed by applications.

Transformation Hub

Also TH. A Kafka-based messaging service that enriches and transforms security data from producers and routes this data to consumers.

Transformation Hub cluster

The Transformation Hub cluster consists of all Master and Worker Nodes in the TH environment.

V

Virtual IP (VIP)

To support high availability on a multi-master installation, a VIP is used as the single IP address or FQDN to connect to a dedicated Master infrastructure that contains 3 or more master Nodes. The Master Nodes manage Worker Nodes. The FQDN of the VIP can also be used to connect to the cluster's Master Nodes.

W

Worker Nodes

Worker nodes contain Kafka brokers which ingest, enrich and route events from event producers to event consumers. Worker nodes are automatically load-balanced by the TH infrastructure.

Z

ZooKeeper

Used in conjunction with Kafka, a centralized service used to maintain naming and configuration data and to provide flexible and robust synchronization within distributed systems.

Send Documentation Feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this computer, click the link above and an email window opens with the following information in the subject line:

Feedback on Deployment Guide (Transformation Hub 3.2.0)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to arcsight_doc@microfocus.com.

We appreciate your feedback!