Smart Distancing and Contact Tracing - Installation and Configuration Guide

Nymi Connected Worker Platform

v4.0

2022-02-04

Contents

Preface

Nymi™ provides periodic revisions to the Nymi Connected Worker Platform. Therefore, some functionality that is described in this document might not apply to all currently supported Nymi products. The product release notes provide the most up to date information.

Purpose

This document is part of the `Connected Worker Platform` (CWP) documentation suite.

The Nymi Smart Distancing and Contact Tracing Installation and Configuration Guide provides information about installing Smart Distancing and Contact Tracing (SDCT) components and configuration options based on your SDCT and NES deployment.

Audience

This guide provides information to NES and Smart Distancing and Contact Tracing Administrators. An NES and Smart Distancing and Contact Tracing Administrator is the person in the enterprise that manages the `Connected Worker Platform` with Smart Distancing and Contact Tracing solution in their workplace.

Revision history

The following table outlines the revision history for this document.

Table 1: Revision history

Version	Date	Revision history
4.0	February 4, 2022	Updates for CWP 1.2.1, which include new content about Edge Agents installations in a local and remote configuration
3.0	November 10, 2021	Updates for CWP 1.2
2.0	June 18, 2021	Updates for the Connected Worker Platform 1.1.2 to include information about how to encrypt the truststore password in the section "Installing and Running the Contact Tracing Collection Agent".

Version	Date	Revision history
1.0	May 03, 2021	First version of this document for the Nymi Connected Worker Platform 1.1 release. This update covers Smart Distancing and Contact Tracing features, installation, and deployment on the CWP platform.

Related documentation

- **Nymi Connected Worker Platform Overview Guide**

  This document provides overview information about the `Connected Worker Platform` (CWP) solution, such as component overview, deployment options and supporting documentation information.
- **Nymi Connected Worker Platform NES Deployment Guide**

  This document provides the steps that are required to deploy the Nymi Enterprise Server (NES). This installation uses the `Nymi Token Service` to install certificates that enable communication between components. This document also provides information about deploying the Connected Worker Platform in a Citrix or RDP environment.
- **Nymi Connected Worker Platform Administration Guide**

  This document provides information about how to use the `NES Administrator Console` to manage the `Connected Worker Platform` (CWP) system. This document describes how to set up, use and manage the Nymi Band™, and how to use the `Nymi Band Application`. This document also provides instructions on deploying the Nymi Band Application and Nymi Runtime components.
- **Nymi API C Interface Application and Developer's Guide**

  This document provides information about how to use the functionality that is available in the `NAPI` that is part of the Connected Worker Platform.
- **Connected Worker Platform Release Notes**

  This document provides supplemental information about the Connected Worker Platform, including new features, limitations, and known issues with the Connected Worker Platform components.
- **Nymi Connected Worker Platform Troubleshooting Guide**

  This document provides information about how to troubleshoot issues and the error messages that you might experience with the `NES Administrator Console`, the Nymi Enterprise Server deployment, the Nymi Band, and the `Nymi Band Application`.

How to get product help

If the Nymi software or hardware does not function as described in this document, contact your administrator for immediate support. Alternatively, you can submit a support ticket to Nymi, or email support@nymi.com

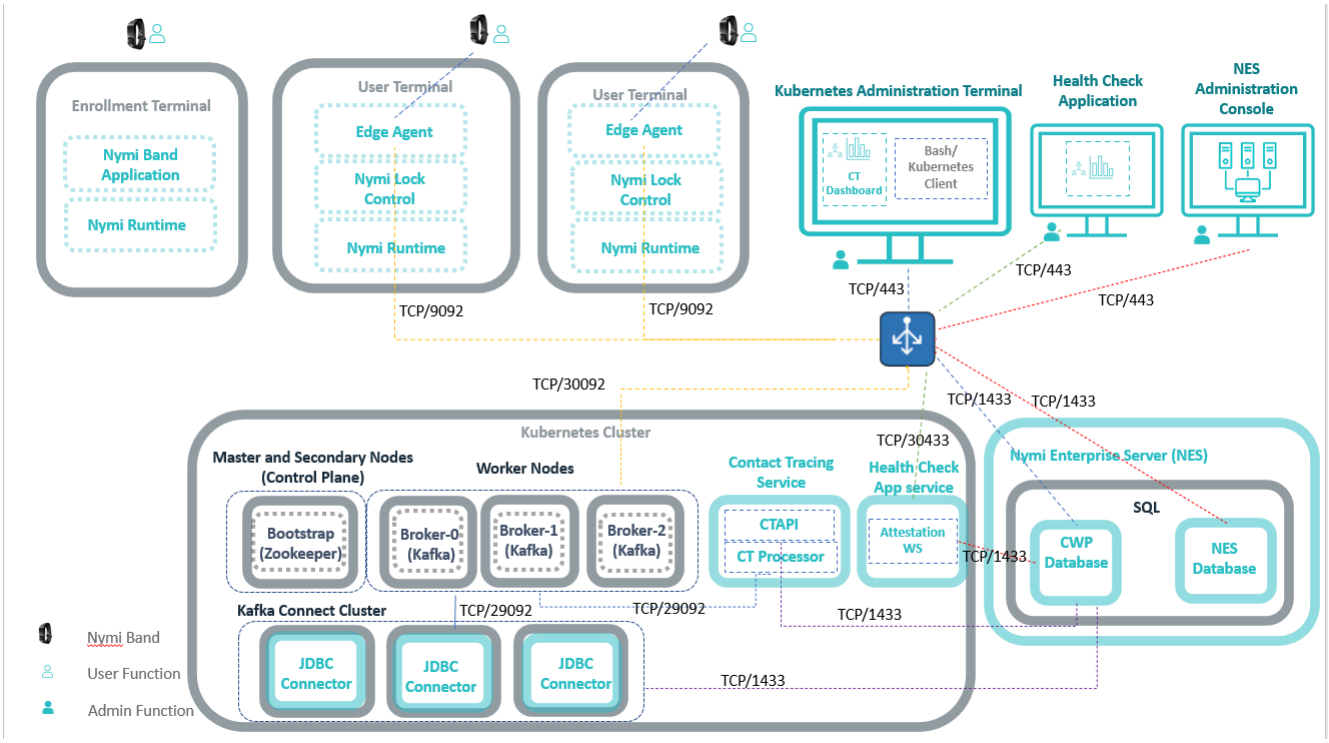
How to provide documentation feedback

Feedback helps Nymi to improve the accuracy, organization, and overall quality of the documentation suite. You can submit feedback by using support@nymi.com

Smart Distancing and Contact Tracing Overview

`Connected Worker Platform` includes many components, which together provide a Smart Distancing and Contact Tracing(SDCT) container clustered service that addresses contact tracing, smart reminders, and attestations requirements to support a safe workplace environment.

The following section provides an overview of the `Connected Worker Platform` with the SDCT service.



`Figure 1: Connected Worker Platform with SDCT Environment`

Enrollment Terminal

Terminal on which you install the `Nymi Band Application`. User access the `Nymi Band Application` to enroll to a Nymi Band. The `Nymi Band Application` also installs the `Nymi Runtime` application.

Contact Tracing Events

When a user wearing an authenticated Nymi Band stays in close proximity to another user wearing an authenticated Nymi Band for approximately 15 cumulative minutes over a 24-hour period.

User Terminal

A thick or thin client that is used by a Nymi Band user to perform daily tasks. When you install `Nymi Lock Control` and `Nymi Runtime` on the user

terminal, users can lock and unlock the user terminal with an authenticated Nymi Band.

Edge Agents

The Edge Agents(EA) is installed on each user terminal in the environment and establishes BLE communication with Nymi Bands via `Nymi Bluetooth Endpoint` and the Nymi Agent services that are installed by `Nymi Runtime`.

EA retrieves contact tracing data from Nymi Bands within 3-4 meters of the user terminal. EA sends the data to the Kafka processing system in the Kubernetes cluster.

Kubernetes Cluster

Provides container cluster deployment, orchestration, scaling, failover and management for the SDCT container clusters. The Kubernetes cluster includes:

- At least one master node and one or more secondary nodes on which Zookeeper resides. Also referred to as the control plane. The control plane consists of control plane master nodes that manage Kubernetes controllers, such as replication controller, endpoint controller, namespace controller, and service accounts controller. A control plane may consist of one or more master nodes (control plane master nodes) to run across multiple computers for high availability, however only one master node may be active at a time.
- Three or more worker nodes on which the brokers and Kafka reside. The worker nodes run containerized applications and host pods (components of an application's workload).

  Each node consists of:

  - kubelet, which ensures that containers are running in a pod
  - kube-proxy, which directs network traffic to and from pods
  - container runtime, which runs the containers.

  Each node is managed by the control plane. Typically, there are several nodes in a cluster, and a pod typically has one container.

  Zookeeper sends jobs to the brokers. SDCT interacts with 5 services in the cluster: bootstrap, broker-0, broker-1, broker-2, and `Contract Tracing API` (`CTAPI`).

- Contact Tracing Service, which includes:
  - `CTAPI` is the part of Contact Tracing Service that provides the contact tracing dashboard and an API to access contact tracing graph data.
  - `Contact Tracing Processor` is the part of Contact Tracing Service that generates contact tracing graph data from contact tracing events.
- Kafka, receives and processes contact tracing data from EA. The Kafka processing system and the `Contact Tracing Processor` transform contact tracing data and then send the data to the CWP Database.
- Kafka Connect Cluster contains Kafka Connect Nodes. Each Kafka Connect Node contains a JDBCConnector.
- JDBCConnector uses Kafka Connect to store authentication and temperature information events in the CWP Database. The configuration of the JDBCConnector determines how the information that is retrieved from the Nymi Band (and published on Kafka) is stored in the CWP Database.

CWP Database

Stores information about contact tracing, Nymi Band authentication, temperature sensing, and Health Attestation results that are received from `CTCS`. Contact tracing data contains information from the Nymi Enterprise Server for contact tracing purposes.

SDCT Management Terminal

Provides the CWP Administrator with the following tools to manage the SDCT environment.

- bash to create and manage the Kubernetes cluster.
- kubectl (AWS only) to manage the Kubernetes cluster and services.
- Contact Tracing Dashboard, a web-based application that allows you to visualize and analyze contact tracing data from the CWP Database for employees that are enrolled in the Contact Tracing program. Use the Contact Tracing Dashboard to view the relationships between contact tracing events from different users. Contact Tracing Dashboard retrieves events in the `CWP Database` via `CTAPI`.

Prepare for Kubernetes and SDCT Deployment

Review the following sections for infomration about hardware, software, and firewall port requirements, as well as how to prepare the Kubernetes Administration Terminal and create the CWP Database.

Hardware and Software Requirements

This section describes the hardware and software requirements for Smart Distancing and Contact Tracing.

NES

- Microsoft SQL Server Management Studio
- Windows 2016
- Windows 2019

Note: Refer to the Nymi Connected Worker Platform NES Deployment Guide for more information about NES requirements and deployment information.

User Terminal

Review the following requirements for the user terminals on which you install Edge Agents.

- Supported Operating Systems:
  - Windows 10 64-bit
- TLS 1.2 enabled
- Oracle Java SE Runtime 8 32-bit or 64-bit (included in Oracle JDK 1.8.x)
- OpenSSL 1.1.1c and later
- Resides on the same domain as NES
- Root CA certificate for NES and Kafka

Kubernetes Administration Terminal

Review the following requirements for the user terminal that you use to access the Kubernetes cluster:

- OS that supports *kubectl* and *bash* terminal, including Windows 10 64-bit with Linux Bash Shell
- Oracle Java JDK 8 or later (32-bit or 64-bit)
- Javascript-enabled browser, such as Microsoft Edge, Google Chrome, Safari, or Mozilla Firefox

Kubernetes Cluster Deployment

The requirements for the Amazon Linux 2 Kubernetes Cluster deployment include:

- Control plane node: AWS Elastic Kubernetes Services (EKS)

- Worker node: EC2 instance with Amazon Linux II OS (ex. t3.xlarge, t4g.large), 4-core CPU, 16 GB RAM, 512 GB SSD
- Self signed or CA-issued TLS certificates for Kafka and `CTAPI`.

  Note: TLS certificate for `CTAPI` and Kafka can be the same, but the SubjectAlternativeNames must include all of the FQDNs.

CWP Database

The Contact Tracing, Health Attestation and Temperature Alert features store data in a Microsoft SQL database that supports TLS 1.2 and later.

The following Microsoft SQL versions are supported:

- SQL Server or SQL Server Express 2016
- SQL Server or SQL Server Express 2017
- SQL Server or SQL Server Express 2019

If your NES database is one of these supported versions, you can deploy the CWP Database on the SQL server with the NES instance.

Note: SQL Server / SQL Express 2016 and SQL Server / SQL Express 2017 require a patch to provide TLS 1.2 support. Microsoft provides more information.

DNS Requirements

CWP requires DNS or */etc/hosts* file entries for the following components.

Note: *`namespace`* is the namespace of CWP and *`domain`* is the public domain name that you will define for the *`KAFKA_BROKER_PUBLIC_DOMAIN`* environment variable.

Host	Purpose
Kubernetes API Server Endpoint	Required when joining a node to the cluster and by kubtctl, should be the same as the server specified in *~/.kube/admin.conf*.
Kafka bootstrap server	Can be any valid DNS name of your choice.
broker.*`namespace`*.svc.cluster.local	For internal Kafka Client access to Kafka Bootstrap Server.
broker-0.broker.*`namespace`*.svc.cluster.local	For internal kafka client access to kafka broker 0.
broker-1.broker.*`namespace`*.svc.cluster.local	For internal kafka client access to kafka broker 1.
broker-2.broker.*`namespace`*.svc.cluster.local	For internal kafka client access to kafka broker 2.
`FQDN_CWP_web_server`	For web access to the `Health Check Application`. The FQDN of the virtual server on which the `Health Check Application` Service resides. **Note:** The FQDN that you choose must match the Subject Alternative Name for the CWPWEB TLS certificate.

Host	Purpose
FQDN_CTAPI	For web access to the Contact Tracing Dashboard. The FQDN of the virtual server on which `CTAPI` resides. **Note:** The FQDN that you choose must match the Subject Alternative Name for the `CTAPI` TLS certificate.
Load Balancers	For a managed Kubernetes cluster in AWS or Azure, the deployment process creates load balancers for the Kafka bootstrap server, Kafka brokers, `CTAPI` and the `Health Check Application` service. The DNS entries that are required for the public IP addresses of the load balancers differ based the configuration: • When you use static IP addresses, define a type A record for the IP address of the load balancer. • When you use dynammic IP addresses, define a CNAME record for the FQN of the load balancer.

Firewall Port Requirements

The following tables outline the port requirements for the `Connected Worker Platform` components.

Table 2: Control Plane Nodes Firewall Port Requirements

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	443	Kubernetes Dashboard service	Cluster
TCP	Inbound	6443 (overridable)	Kubernetes API server	Cluster
TCP	Inbound	2379-2380	etcd server client Cluster	kube-apiserver, etcd
TCP, UDP	Inbound	53	Core DNS	Cluster
TCP	Inbound	179	BGP routing	Calico CNI
TCP	Inbound	9500	Longhorn CSI (for self-managed Kubernetes Clusters	Cluster
TCP	Inbound	9090	Prometheus API access	Frontend
TCP	Inbound	9100	Prometheus Node Exporter	Cluster
TCP	Inbound	22	ssh	remote access

Table 3: Worker Nodes Firewall Port Requirements

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	10250	kubelet API	Self, Control plane
TCP	Inbound	10255	kubelet API read-only	Control plane
TCP	Inbound	10256	Health check	Control plane
TCP	Inbound	179	BGP routing, for pod to pod networking	Pods
TCP	Inbound	22	ssh for deployment and maintenance	administrators
TCP	Inbound	30090-30094	Kafka Broker (internal access)	EA
TCP	Inbound	31443	CTAPI Web service (external access)	Client, Load Balancer

Table 4: Load Balancer Firewall Port Requirements

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	9092	Load balancer for Kafka Broker	EA
TCP	Inbound	443	CTAPI Web service	`Contact Tracing Dashboard`, `Health Check Application`, NES Administrator Console

Table 5: Kafka Connect Requirements

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	29092	JDBC Connector	Worker Nodes, Contact Tracing Service

Table 6: CWP Database Requirements

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	1443	SQL database	JDBC Connector, Contact Tracing Service

Certificate Requirements

The environment requires TLS certificates to gain secure access to the Kubernetes cluster, the `Contact Tracing Dashboard` and the `Health Check Application`.

Kafka TLS Certificate

Create a TLS certificate for Kafka.

Before you generate and install the TLS certicate, create a certificate that:

- Is in PKCS#12 format.
- Contains the TLS certificate, the private key of the certificate, and the certificate of the signing authority.
- Has extended key usage Server Authentication. (1.3.6.1.5.5.7.3.1) and Client Authentication (1.3.6.1.5.5.7.3.2).
- Has subject alternative names that match the FDQNs of the following internal and external Kafka broker listeners:
  - broker-0.broker.*`namespace`*.svc.cluster.local
  - broker-1.broker.*`namespace`*.svc.cluster.local
  - broker-2.broker.*`namespace`*.svc.cluster.local
  - *`namespace`*-broker-0.*`domain`*
  - *`namespace`*-broker-1.*`domain`*
  - *`namespace`*-broker-2.*`domain`*

  where:

  - *`namespace`* is the namespace of CWP.
  - *`domain`* is the public domain name that you will later define for the *`KAFKA_BROKER_PUBLIC_DOMAIN`* environment variable.

CTAPI and Health Check Application Service TLS Certificate

Create a TLS certificates for the `CTAPI` and the `Health Check Application` Service.

Before you generate and install the TLS certificate, create a certificate that:

- Is in PKCS#12 format.
- Contains the TLS certificate, the private key of certificate, and the certificate of the signing authority.

Deployment, Installation, and Configuration Overview

The following section provides a high level overview of the deployment, installation and configuration process.

1. Review the required firewall ports, and update as required.
2. Install `Nymi Runtime`, `Nymi Bluetooth Endpoint`, and `Nymi Agent`.
   a) Install `Nymi Bluetooth Endpoint` on the user terminal used to collect contact tracing data from Nymi Bands.
   b) Install `Nymi Agent` on the terminal used to access the Contact Tracing Dashboard (for example, the server).

      For users running Citrix or RDP, install the `Nymi Agent` on the Contact Tracing server.
3. Install bash.exe (Ubuntu). Refer to Installing Bash on the Kubernetes Administration Terminal on page 17.
4. Deploy an AWS with EKS Kubernetes Cluster. Refer to Deploy A Kubernetes Cluster in AWS Using EKS on page 23.
5. Create the SQL database for Contact Tracing. Refer to Creating the CWP Database and tables on page 19.
6. Prepare the Kubernetes environment for Contact Tracing services. This process involves obtaining certificates and editing environment variables. Refer to Customize the Kubernetes environment for SDCT on page 26.
   a) Set up the service account password.
   b) Install TLS certificates for Kafka and CTAPI. Refer to Preparing Certificates to install SDCT on page 27.
   c) Set up environment variables. Refer to ENV variables.
7. Encrypt passwords to Kubernetes components. Refer toEncrypting the Passwords for Kubernetes Components on page 40. Verify that the passwords in the env file are updated.
8. Launch Kubernetes (refer to Launching the Kubernetes Environment on page 40), then ensure the Kubernetes cluster is running.

   ```
   kubectl get pods -A
   ```
9. Install `Edge Agents` (EA) on a domain-joined (NES) terminal. Refer to .Installing and Running the Contact Tracing Collection Agent. You will need to:
   a) Encrypt the SASL username and password.
   b) Encrypt the truststore password.
   c) Update TLS certificates.
   d) Update the *edge_agents.conf* file to include the newly encrypted username and password.
   e) Install EA and configure the environment variables.
10. Enable SDCT components in Nymi Enterprise Server.

    On Nymi Enterprise Server, go to the NES Administrator Console and enable Smart Distancing and Contact Tracing.

11. Update the configuration settings on the Nymi Bands by having Nymi Band users log into Nymi Band Application.

Installing Bash on the Kubernetes Administration Terminal

If you do not have a Linux system, install Linux Bash shell on a user terminal in your environment that will act as the Kubernetes Administration Terminal.

Before you perform these steps, ensure that Oracle JDK 8 or later is installed on the terminal.

The following procedure describes how to install Linux Bash Shell on a Windows 10 machine.

Note: You will need to restart the computer to apply changes to the terminal.

1. Log into Windows as an administrator.
2. Go to **Programs and Features**.
   Start > Control Panel > Programs > Programs and Features
3. Select **Turn Windows Features On or Off**.
   A window appears with a list of Windows features.
4. Select **Virtual Machine Platform** and **Windows Subsystem for Linux**, and then click **OK**.
   Windows will search for and install the required files. Restart the terminal to apply the changes.
5. Perform the following steps to obtain **Ubuntu** from the **Microsoft Store**.
   a) From the **Start** menu, type `Microsoft Store`, and then select **Microsoft Store**.
   b) In the **Search** field, type `Ubuntu`.
   c) From the list of apps that appear. select the **Ubuntu** application without the version number.

      Note: The **Ubuntu** application without the version number contains the most recent version. You may choose an earlier version if it is more applicable for your system.
   d) Click **Get**.

      A window appears that prompts you to sign in. You may skip this by closing the pop-up window.

      The **Ubuntu** application downloads in the background.
   e) When complete, click **Launch**.
      An **Ubuntu** windows appears and installs Ubuntu.
   f) At the **Enter a new UNIX username** prompt, type a new username.
   g) As the **New password** and **Retype new password** prompt, type a password for the user.
6. From the **Microsoft Store**, search for the **Windows Terminal** application.
7. Click **Get**
   The **Windows Terminal** application installs.

To open **bash**, go to the **Start** menu and type `bash`.

Alternatively, you can open **Windows Terminal**, click on the dropdown arrow from the top tool bar, and then select **Windows Powershell** or **Ubuntu**.

Obtaining the SDCT packages

Your Nymi Solution Consultant provides you with packages that contains several files and scripts to assist you in the SDCT and Kubernetes configuration and deployment.

- *edgeagents-x64.u.v+wx-yz.zip* - Contains the files to deploy and configure EA on the User Terminal/Collection Agent.
- *cwp-deployment-1.2.u.bc* - Contains the files to deploy and configure CWP in the Kubernetes cluster.
- *cwp-deployment-update-1.2.u.bc* - Contains the files to update CWP in the Kubernetes cluster.

1. Download and extract the *edgeagents-x64.v+wx-yz.zip* package to a central location that is accessible to the user terminals.
   The *edgeagents* folder is created.
2. For a new install, download and extract the file *cwp-deployment-1.2.v.bc* to a central location that is accessible to the Kubernetes Administration Terminal, and then copy to the folder to the Kubernetes Administration Terminal.
   For example, the *cwp-deployment-1.2.x.y* folder is created that contains the *cwp* and *deploy*folders.

   Note: The folder in which you extracted the CWP 1.2 installation package is referred to as the CWP 1.2 deployment folder (*CWP_12_deployment_folder*) in this guide.
3. For an upgrade, download and extract the file *cwp-deployment-update-1.2.v.de* to a central location that is accessible to the Kubernetes Administration Terminal, and then copy the folder to the Kubernetes Administration Terminal.
   For example, the *cwp-deployment-update-1.2.x.y* folder is created that contains the the *cwp* and *kube* folders.

   Note: The folder in which you extracted the CWP 1.2 upgrade package is referred to as the CWP 1.2 deployment folder (*CWP_12_deployment_folder*) in this guide.

Recording the SDCT Variables

Throughout the deployment process, you will perform configuration tasks that you will be required to remember later on.

Use the following table to keep track of values for variables that you define during the deployment.

Table 7: Environment Variable Values

Variable Name	When Used	Value
CT_DB_CATALOG	The SQL database name for contact tracing. Required when you configure the *.env* file.	
CT_DB_USERNAME	Required when you configure the *.env* file.	

Variable Name	When Used	Value
CT_DB_PWD	Define this value when you run the *init-crypto* script.	
CT_DB_INSTANCE	Required when you configure the *.env* file.	
KAFKA_BROKER _PUBLIC_DOMAIN	The public domain. Required when you configure the *.env* file.	

Prepare the Database

Before you deploy CWP, prepare the SQL database.

Creating the CWP Database and tables

Perform the following steps in SSMS to create the `CWP Database` and tables, and then ensure that `CTAPI` can to connect to the NES and SQL database.

- Ensure that the extracted CWP deployment package is in a central location.

Perform the following steps on a machine that has SSMS installed and has access to the NES database server.

1. Navigate to the *`CWP_12_deployment_folder`**cwp**ctprocessor* folder in the shared location, and then copy the the *ct-mssql.sql* file to a local directory.
2. Navigate to the *`CWP_12_deployment_folder`**kafka-connect**config* folder in the shared location, and then copy the *auth-mssql.sql* file to a local directory.
3. Open SSMS, and then login to the SQL Server.
4. Right-click the SQL instance, and the select **`Properties`**.
5. In the **`Object Explorer`**, select **`Security`**
6. Select **`SQL Server and Windows Authentication Mode`**, and then click **`OK`**.
7. In the **`Object Explorer`** right-click `Databases`, and the select `New Database`.
8. Name the database `ContactTracing`, and then click **`OK`**.
   Record the value that you specify in the SDCT Variables table for the variable *`CT_DB_CATALOG`*.
9. In the **`Object Explorer`**, go to **`Security > Logins`**.
10. Right-click **`Logins`** and click **`New Login`**.
    A **`Login - New`** window appears.
11. On the **`General`** page, perform the following actions.
    a) Specify a login name.
       Record the value that you specify in the SDCT Variables table for the variable *`CT_DB_USERNAME`*.
    b) Click **`SQL Server authentication`** and enter a password.
       Record the value that you specify in the SDCT Variables table for the variable *`CT_DB_PWD`*

12. On the **User Mapping** page, perform the following actions.
    a) In the **Map** column, select the checkbox beside the **CWP** database, that you created previously.
    b) In the **Database role membership for CWP** section at the bottom of the window, select **db_owner** and leave the default selection **public** enabled.
    c) Click **OK**.
13. From the **File** menu, select **Open > File...**, navigate to folder that contains the *ct-mssql.sql* script file that you downloaded, and then click **Open**.
    The script opens in the query window.
14. On the menu bar, click **Execute**.

    The script creates a table for Contact Tracing proximity events with the username and password that you specified previously.

    The script creates the dbo.CovidContacts table.
15. From the **File** menu, select **Open > File...**, navigate to folder that contains the *auth-mssql.sql* script file that you downloaded, and then click **Open**.
    The script opens in the query window.
16. On the menu bar, click **Execute**.

    The script creates the SQL table for authentication events with the username and password that you specified previously.

    The script creates the and dbo.AuthInfo table.
17. Close SSMS.

The following figure shows SSMS with the new `CWP Database` and the **Message** window after successfully executing the *ct-mssql.sql* script.

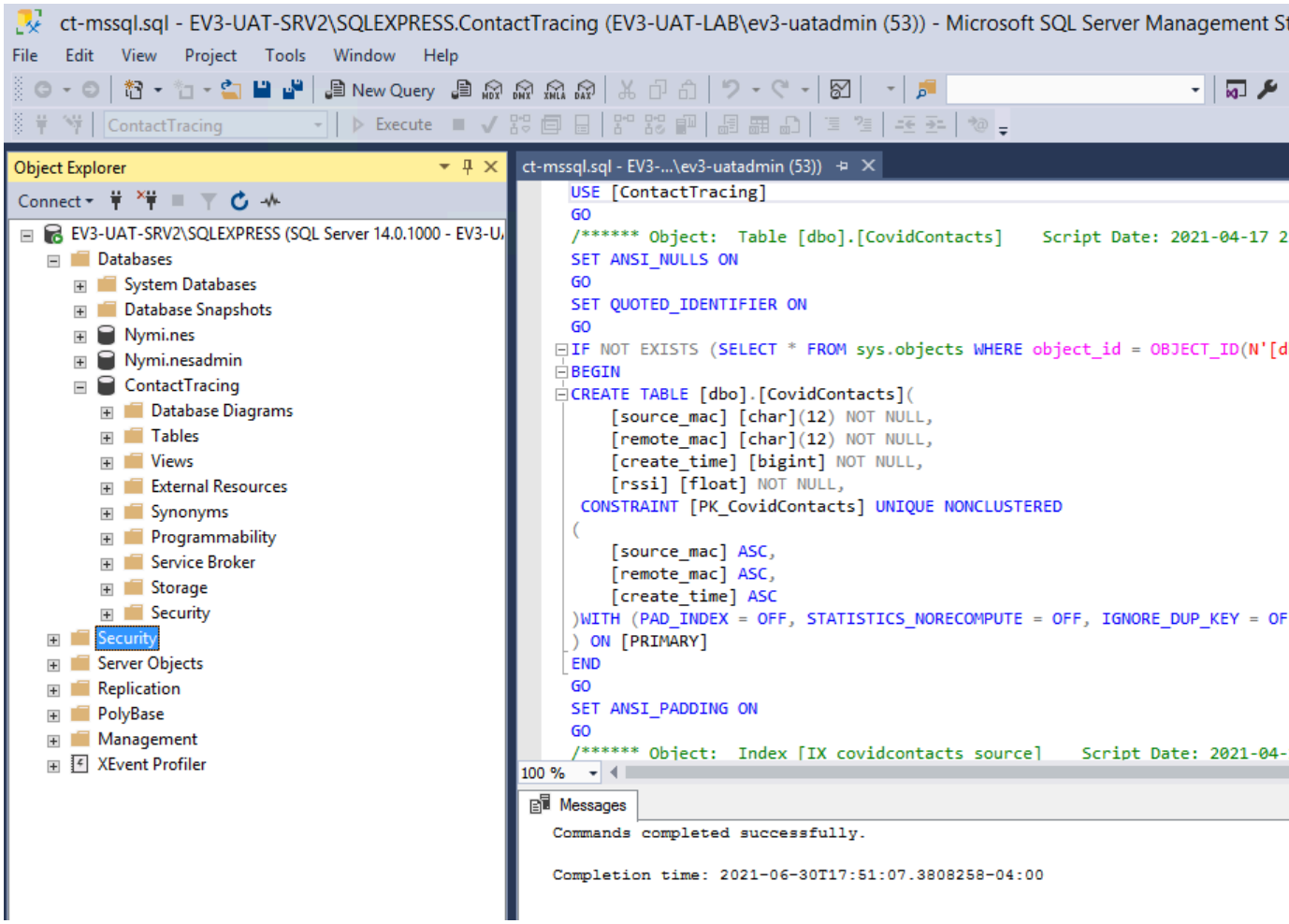


Figure 2: SSMS with ContactTracing Database

Creating Tables for Health Attestation and Temperature Alerts

If your environment uses the Health Attestation and Temperature Alerts features, run the Nymi-provided SQL scripts to create the appropriate tables.

- The user that creates the database must have db_owner or db_ddladmin privilege to the `CWP Database`.
- Ensure that the extracted CWP deployment package is in a central location.

Perform the following steps from a computer that has SSMS installed and access to the SQL Server with the `CWP Database`.

1. Open SSMS and connect to the SQL server.
2. Navigate to the *`CWP_12_deployment_folder`\cwp\cwpweb\config* folder in the shared location.
3. Copy the *mssql.sql* file to local directory.

4. Navigate to the *`CWP_12_deployment_folder`\cwp\kafka-connect\config* folder in the shared location.
5. Copy the *temp-mssql.sql* file to a local directory.
6. In the local directory, double-click on the *mssql.sql* file.
   A new query window appears.
7. Click the **`Execute`** button.
   The script creates the dbo.NymiAttestationInfo and dbo.JwtStore tables.
8. In the local directory, double-click on the *temp-mssql.sql* file.
   A new query window appears.
9. Click the **`Execute`** button.
   The script creates the dbo.NymiTempInfo table.

Configuring the SQL Server

Ensure that the TCP/IP is enabled fo the SQL instance.

Perform the following actions in the `SQL Server Configuration Manager` application.

1. In the left navigation pane, expand `SQL Server Network Configuration`, and then select the appropriate Protocols for the SQL Server option.
2. In the right pane, select TCP/IP, and then right-click and select **`Enabled`**.
3. Double-click **`TCP/IP`**.
4. In the `TCP/IP Properties` window, select the **`IP addresses`** tab.
5. Navigate to the `IPALL` section, and then for the **`TCP port`** value, type `1433`.
6. Click **`OK`**, and then click **`Apply`**.
7. On the prompt to restart the SQL services, click **`OK`**.
8. Restart SQL server services.

Kubernetes Deployment

To implement Smart Distancing and Contact Tracing, you must deploy a Kubernetes cluster with at least physical computer or virtual machine that acts as the worker node.

For high availability, the number of control plane master nodes that you require depends on how the ETCD cluster is deployed. There are two choices:

- In a stacked Kubernetes production environment with bundled ETCD cluster. There must be at least 3 control plane master nodes.
- In a Kubernetes environment that uses an external ETCD cluster. There must be at least 2 control plane master nodes.

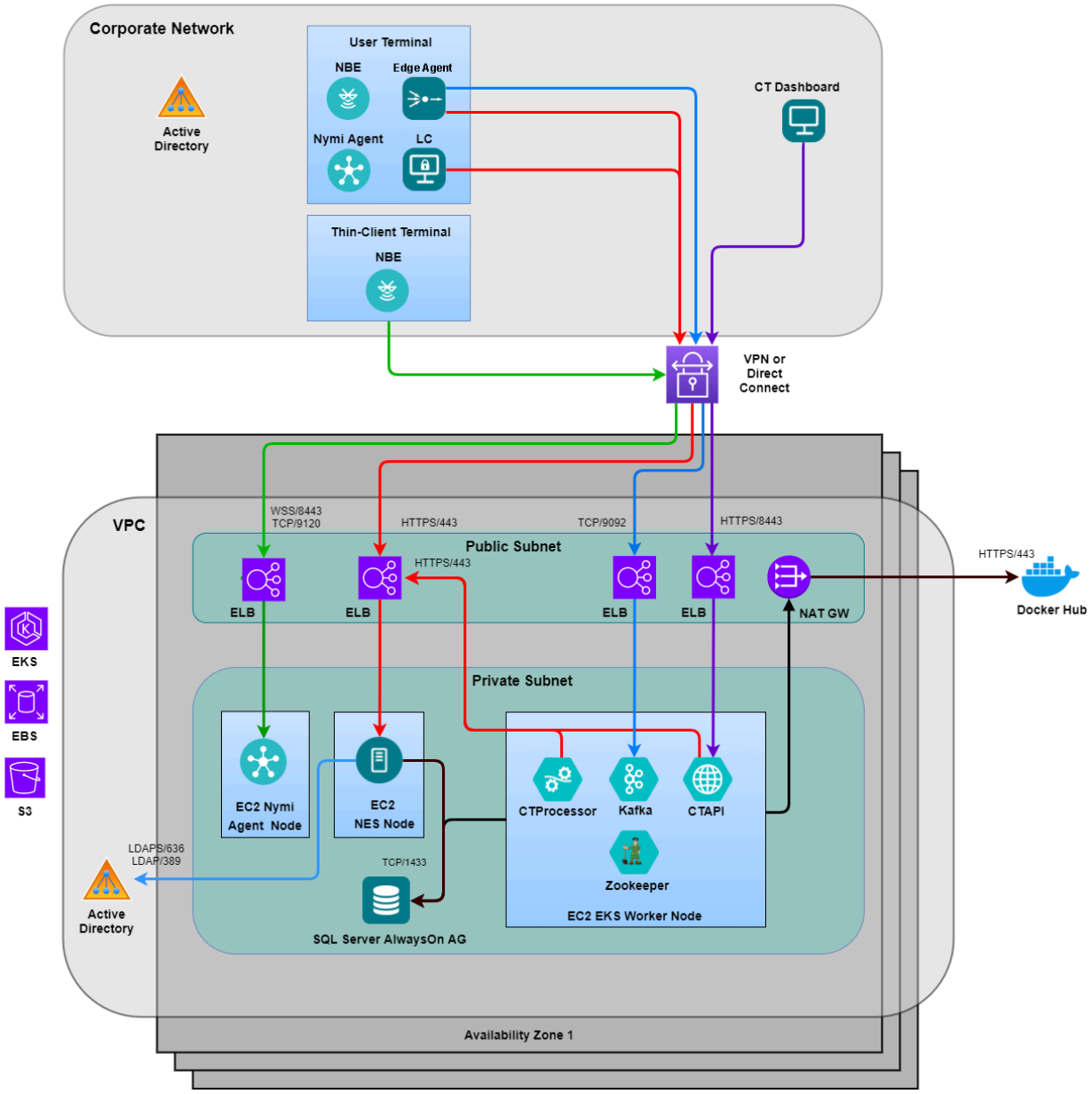
Note: ETCD is an open source distributed key-value store that manages data for Kubernetes. ETCD is a quorum based system. The number of nodes required in an N-node cluster is N/2 + 1.

Table 8: Number of Nodes Required for High Availability

No. of Nodes	Quorum Required	No. of Nodes Allowed to Fail
2	2	0
3	2	1
4	3	1
5	3	2
6	4	2

Deploy A Kubernetes Cluster in AWS Using EKS

The following diagram provides an overview of the CWP Kubernetes cluster deployment architecture for AWS EKS.


Corporate Network
User Terminal
NBE
Edge Agent
Nymi Agent
LC
Active Directory
CT Dashboard
Thin-Client Terminal
NBE
VPN or Direct Connect
VPC
WSS/8443
TCP/9120
HTTPS/443
TCP/9092
HTTPS/8443
Public Subnet
HTTPS/443
ELB
ELB
ELB
ELB
NAT GW
HTTPS/443
Docker Hub
EKS
EBS
S3
Private Subnet
EC2 Nymi Agent Node
EC2 NES Node
CTProcessor
Kafka
CTAPI
Zookeeper
EC2 EKS Worker Node
LDAPS/636
LDAP/389
TCP/1433
Active Directory
SQL Server AlwaysOn AG
Availability Zone 1

Kubernetes deployment in AWS with EKS includes the following steps:

1. Create a AWS VPC for EKS Cluster
2. Create a ESK Cluster

Note: It is very important to keep the *env* file after the deployment. It contains crucial parameters required for removing the cluster and VPC.

VPC Considerations

To create an EKS cluster, you require an available VPC.

The VPC should include subnets across multiple available zone in the respective AWS region. In a production environment, consider using 3 or more availability zones when possible.

The VPC may include both public and private subnets. The following table summarize the subnet configurations:

Table 9: VPC Subnet Configurations

Subnet Configuration	Detail	Application
Both public and private subnets	Public subnet for ELB and Private subnets for EKS worker nodes	Internet facing
Public subnets only	Public subnet for ELB and EKS worker nodes	Internet facing
Private subnets only	Private subnets for the EKS worker nodes. This configuration requires a NAT Gateway to allow nodes to access internet	Not for internet facing

Installing Kubernetes Client (AWS)

Perform the following steps to create the Kubernetes client when using AWS.

1. On the Kubernetes Administration Terminal, open a **bash** terminal and change to the *CWP_12_deployment_folder/deploy/kube/client* folder.
2. Type `./init-client -a aws-region -c cluster name`

   Where:

   - *aws-region* defines the AWS region where the cluster resides.
   - *cluster name* defines the name of the EKS cluster.

Running the VPC Creation Script

Perform a new VPC deployment by using the **create-vpc** script.

Perform the following steps on the designated initial master node.

1. On the Kubernetes Administration Terminal, open a PowerShell or **bash** terminal and change to the *CWP_12_deployment_folder/deploy/kube/init/aws* folder in the shared location.

2. Type `./create-vpc -region aws-region [-public] [-private] [-cidr network_CIDR]`

   where:

   - *aws-region* - Defines the AWS region on which to create the VPC.
   - -public - Creates a public subnet in each availability zone, up to a maximum of 3 availability zones may be used
   - -private - Creates a private subnet in each availability zone, up to a maximum of 3 availability zones may be used
   - -cidr *network_CIDR* - Defines the network CIDR of the VPC. It is recommended to use a class A private network CIDR with 16 bit blocks.

   For example, the following command creates a VPC with private subnets in the designated region *us-east-2*: `./create-vpc -region us-east-2 -private -cidr 10.120.0.0/16`

   This command distributes the network CIDRs of the subnet according to the size of the CIDR blocks on the VPC. The size of the network CIDRs will be 1/16 of the size of the VPCs.

Creating a EKS Cluster

Create a new EKS cluster by using the **`create-cluster`** script:

Perform the following steps on the designated initial master node.

1. Open a **bash** terminal and change to the *CWP_12_deployment_folder*/deploy/kube/init/aws folder.
2. Type `./create-cluster [-public] [-private] -admin|-a kubernetes_admin`

   where:

   - *-public* - Is specified if the respective VPC has public subnet.
   - *-private* - Is specified if the respective VPC has private subnet. This also enables *-public*.
   - *kubernetes_admin* - Defines the Kubernetes administrator account that manages the Kubernetes cluster. You can define any valid account name. The default value is *kube-admin*.

   For example,

   - To create an EKS cluster with a VPC that has both public and private subnets, type `./create-cluster -private`
   - To create a VPC with only public subnets, type `./create-cluster -public`

AWS EKS automatically creates an Elastic Load Balancing (ELB) for a *LoadBalancer* type service.

An AWS Network Load Balancer at layer 4 of the OSI model load balances the network traffic.

you can use an Ingress controller instead of an external load balancer. This guide does not document the use of an ingress controller.

Customize the Kubernetes environment for SDCT

Perform the follow steps to customize the Kubernetes environment.

Preparing Certificates to install SDCT

SDCT requires TLS certificates to secure communications.

Perform the following steps on the Kubernetes Administration Terminal.

1. Copy the CTAPI TLS certificates (PKCS12) into the *`CWP_12_deployment_folder`*`/cwp/certs` folder, and then rename the certificate file to `tls.pfx`.

   Note: The deployment script will extract values for *`tls.crt`* and *`tls.key`* from the certificate.
2. Copy the Kafka TLS certificate (PKCS12) into the *`CWP_12_deployment_folder`*`/cwp/certs` folder, and then rename the Kafka certificate file to `kafka-tls.pfx`.

When you run the *./init-crypto* script, you are prompted to specify the TLS certificate password.

Setting the Environment Variables in the .env File

Nymi-supplied scripts create the environment based on variables that you define in the *.env* file.

The *.env* contains environment variables that are common to all environments. The file *.env* includes the credentials and IP addresses of the domain, NES, and the Contact Tracing Dashboard. Refer to the SDCT Variables table for the values that you recorded during the SQL database and domain setup.

1. On the Kubernetes Administration Terminal, in Windows Explorer, navigate to *`CWP_12_deployment_folder`*`\deploy\kube` folder.
2. Edit the *.env* file and configure the values to match your environment.

   The following table summarizes the general configuration variables.

Variable	Description
`CWP_COMPONENTS`	Specifies the list of CWP components to deploy. The default value is `=(zookeeper broker ctapi ctprocessor cwpweb kafka-connect)`
`NES_USER_API_BASE_URL`	Specifies the URL that an NES Administrator uses to access the `NES Administrator Console` in a web browser. Specify the value in the format `https://`*`nes_ip_address/service_name`*. You can also find these values by running the NES installer on the NES server, and then selecting the **`Review Settings`** tab. For example: `=https://10.0.4.167/nes`
`CORP_LDAP_PROTOCOL`	Specifies the LDAP protocol that used in the domain, `ldap` or `ldaps`.

Variable	Description
CORP_LDAP_PORT	Specifies the port on which Kubernetes uses to connect to the domain. Type `389` for LDAP or `636` for LDAPS.
CORP_LDAP_DC	Specifies the IP address of the Active Directory domain controller.
CORP_LDAP_BASEDN	Specifies the base Distinguished Name (DN) to use to search LDAP. The first DC should be the Active Directory domain (ie. *`CORP_LDAP_DOMAIN`*. For example: `="DC=qa-lab,DC=local"`
CORP_LDAP_API_GROUP	Specifies the LDAP group that has access to AWS. The default value is Domain Users, which you can change if necessary.
CORP_LDAP_DASHBOARD_GROUP	Specifies the LDAP group that contains users that require administrator access to the `Contact Tracing Dashboard`.
CORP_LDAP_ATTESTATION_ADMIN_GROUPS	Specifies the name of the Health and Safety AD group. Use commas to separate multiple group names. **Note:** Users in this group have administrator access to the Health Check Application.
CORP_LDAP_DOMAIN	Specifies FDQN of the Active Directory domain in which NES resides. For example: `=qa-lab.local`
CORP_LDAP_USER	Specifies the service account, which is an Active Directory domain account that is a member of the NES administrator group. For example: `=adeed`
CORP_LDAP_USERDN	Specifies the DN of the *`CORP_LDAP_USER`* that is used by LDAP. The format should consist of the name of an object and the LDAP designator.

Variable	Description
	For example: `="CN=adeed,CN=Users,DC=qa-lab,DC=local"`
CORP_LDAP_PASSWORD	Specifies password of the service account. Do not type a value for this variable. The *init-crypto* command will encrypt the password and update the file with the appropriate value.
NODE_TLS_REJECT_UNAUTHORIZED	This value is dependent on the NES certificate trust. If this value is `0` certificate validation is disabled for TLS connections. Nymi recommends that you specify a value of 1.

The following table provides information on the configuration parameters that are specific to Kafka. Modify values if your configuration uses values that differ from the default.

Variable	Description
KAFKA_BROKER_PUBLIC_DOMAIN	Required. Specfies the public domain for the Kubernetes cluster, to allow machines that are on a different domain from the Kubernetes cluster, access to the Kubernetes cluster. You can obtain this value from the SDCT Environment Variables
KAFKA_BROKER_EXTERNAL_PORT	Specifies the Kafka broker listener port for external client. The default value is `=9092`
KAFKA_BROKER_INTERNAL_PORT	Specifies the Kafka broker listener port for internal client. The default value is `=29092`
KAFKA_BROKER_INTER_BROKER_PORT	Specifies the Kafka broker listener port between broker instances. The default value is `=9095`
KAFKA_BROKER_BOOTSTRAP_NODE_PORT	Specifies the node port on which the kafka broker bootstrap service connects to the external load balancer. The default value is `30090` .

Variable	Description
KAFKA_LISTENER_SECURITY_PROTOCOL_INTERNAL	Specifies the Kafka broker listener protocol in the Kubernetes cluster. The default value is `=SSL`
KAFKA_LISTENER_SECURITY_PROTOCOL_EXTERNAL	Specifies the Kafka broker listener protocol outside the kubernetes cluster. The default value is `=SASL_SSL` .
KAFKA_LISTENER_SECURITY_PROTOCOL_INTER_BROKER	Specifies the Kafka broker listener protocol between the broker instances. The default value is `=SSL` .
KAFKA_SSL_KEYSTORE_PASSWORD	Specifies the Kafka broker TLS certificate keystore password. Do not type a value for this variable. The *init-crypto* command will encrypt the password and update the file with the appropriate value.
KAFKA_SASL_ADMIN_PASSWORD	Specifies the Kafka SASL admin password. Do not type a value for this variable. The *init-crypto* command will encrypt the password and update the file with the appropriate value.
KAFKA_SASL_CTPROCESSOR_PASSWORD	Specifies the CT Processor password. Do not type a value for this variable. The *init-crypto* command will encrypt the password and update the file with the appropriate value.
KAFKA_SASL_CTCA_PASSWORD	Specifies the `Edge Agents` password. Do not type a value for this variable. The *init-crypto* command will encrypt the password and update the file with the appropriate value.
KAFKA_BROKER_BOOTSTRAP_NODE_PORT	Bootstrap node port. The default value is `30090` .
KAFKA_SSL_ENABLED_PROTOCOLS	Comma separated list of secure protocols. The default value is `TLSv1.3,TLSv1.2`

Variable	Description
	.
KAFKA_SSL_CIPHER_SUITES	Lists the cipher suites. Leave the default value "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256, TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256,TLS_DH TLS_DHE_RSA_WITH_AES_128_GCM_SHA256"
KAFKA_CONNECT_CPU_REQUESTS	CPU request for Kafka-connect.
KAFKA_CONNECT_MEMORY_REQUESTS	Defines the memory request for Kafka-connect.
KAFKA_CONNECT_CPU_LIMITS	Defines the CPU limits for Kafka-connect.
KAFKA_CONNECT_MEMORY_LIMITS	Defines the memory request for Kafka-connect.

The following table provides information about the variables that you defined when you configured the `CWP Database`. Use the *SDCT Variables* table to obtain the values that are specific to your environment.

Variable	Description
CT_DB_HOST	Specifies the IP address of the machine that is hosting the CWP Database. For example: =10.0.4.102
CT_DB_PORT	Specifies the port for the SQL database. The default value is =1433
CT_DB_INSTANCE	Specifies name of the SQL instance, on which you created the `CWP Database`. For example: =SQLEXPRESS

Variable	Description
CT_DB_CATALOG	Specifies the location of the `ContactTracing` catalogue, where various schema and mappings are kept in an SQL environment. This value is the name that you specified for the `CWP Database`. For example: `=ContactTracing`
CT_DB_SCHEMA	The SQL server schema for the `CWP Database`. If this is not particularly defined, use the default, `dbo`. `=dbo`
CT_DB_USERNAME	Specifies the SQL username to use to log into the `CWP Database`. For example: `=ct_sa`
CT_DB_PASSWORD	Specifies password of the SQL database user. The init-crypto script prompts you for this password and then encrypts the password. It is not necessary to type the password.
TLS_CA_CERT	Specifies the location of the TLS signing CA certificate.
TLS_SKIP_CERTIFICATE_VALIDATION	Determines if the script should skip Certificate verification. Acceptable values are True or False.
CWPWEB_PORT	Specifes the port on which the Health Check Application connects to the CWP Database. The default value is 443.

The environment variables defined in these file provides values to the ConfigMaps that are used by the CWP components. ConfigMaps are defined in *kube/cwp/base.in/config.yml* folder.

Setting the Envrionment Variables in the .prod-env

Nymi-supplied scripts create the environment based on variables that you define in the *.prod-env* file.

The *.prod-env* contains environment variables that are customer environment-specific.

1. On the Kubernetes Administration Terminal, in Windows Explorer, navigate to *`CWP_12_deployment_folder`\deploy\kube* folder.

2. Edit the *.prod-env* file and configure the values to match your environment.

   The following table summarizes the general configuration variables.

Variable	Description
CWP_NAMESPACE	Defines the global namespace. Leave the default value cwp.
CWP_ENV	Defines the global namespace. Leave the default value production.
DOCKER_HUB_ENV_SUFFIX	Specifies the docker hub repositories where images for the respective environment are stored. Leave this variable undefined.
KAFKA_BROKER_SERVER	Defines the internal port on which Kafka connects to the boostrap server. Leave the default value `broker.${CWP_NAMESPACE}.svc.cluster.local: ${KAFKA_BROKER_INTERNAL_PORT}` **Note:** The *KAFKA_BROKER_INTERNAL_PORT* is defined in the *.env* file.

ZOOKEEPER_SERVERS	Defines the Zookeeper servers in the Kubernetes cluster. Leave the default value
	`"zookeeper-0.zookeeper. ${CWP_NAMESPACE}.svc.cluster.local:2888:3888;`
	`zookeeper-1.zookeeper.${CWP_NAMESPACE}`
	`.svc.cluster.local:2888:3888;zookeeper-2.zookeeper. ${CWP_NAMESPACE}.svc.cluster.local:2888:3888"`

ZOOKEEPER_REPLICAS	Specifies the number of initial Zookeeper replicas. The default value is 3.
KAFKA_REPLICAS	Specifies the number of initial Kafka replicas. The default value is 3.
VERNE_REPLICAS	Specifies the number of initial Verne replicas. The default value is 3.
WEB_REPLICAS	Specifies the number of initial Web replicas. The default value is 1.
CT_REPLICAS	Specifies the number of initial CT replicas. The default value is 3.
ZOOKEEPER_MAX_REPLICAS	Specifies the maximum number of initial Zookeeper replicas. The default value is 6.

Variable	Description
KAFKA_MAX_REPLICAS	Specifies the maximum number of initial Kafka replicas. The default value is 9.
VERNE_MAX_REPLICAS	Specifies the maximum number of initial Verne replicas. The default value is 9.
WEB_MAX_REPLICAS	Specifies the maximum number of initial Web replicas. The default value is 9.
CT_MAX_REPLICAS	Specifies the maximum number of initial CT replicas. The default value is 9.
ZOOKEEPER_MIN_AVAILABLE	Defines the minimum number of Zoopkeeper replicas to always have available. The default value is `$(($ZOOKEEPER_REPLICAS/2 + 1))` .
KAFKA_MIN_AVAILABLE	Defines the minimum number of Zoopkeeper replicas to always have available. The default value is `$(($KAFKA_REPLICAS/2 + 1))` .
VERNE_MIN_AVAILABLE	Defines the minimum number of Verne replicas to always have available. The default value is `$(($VERNE_REPLICAS/2 + 1))` .
WEB_MIN_AVAILABLE	Specfies the minimum number of Web replicas to always have available. The default value is 1.
CT_MIN_AVAILABLE	Specifies the minimum number of CT replicas to always have available. The default value is 1.
ZOOKEEPER_DATA_SIZE	Specifies the persistent volume size for Zookeeper data. The default values is `8Gi` .
KAFKA_DATA_SIZE	Specifies the persistent volume size for Kafka data. The default values is `32Gi` .

Variable	Description
VERNE_DATA_SIZE	Specifies the persistent volume size for Verne data. The default values is `8Gi` .
PV_RECLAIM_POLICY	Specifies the persistent volume policy to reclaim disk space. Supported values are Delete, Retain or Recycle. The default value is `Delete` **Note:** Most systems do not support Recycle.
KAFKA_LOG_RETENTION_HOURS	Specifies the number of hours to retain the Kafka log file entries. The default value is `1` .
KAFKA_TOPIC_MULTIPLIER	The default value is `3` .
KAFKA_REPLICATION_FACTOR	The default value is `3` .
KAFKA_MIN_REPLICATION_FACTOR	The default value is `1` .
KAFKA_DEFAULT_PARTITIONS	The default value is `$((KAFKA_TOPIC_MULTIPLIER*KAFKA_REPLICAS))`

Variable	Description
	.

The following table summarizes variables that control the resource requests and limits.

Variable	Description
ZOOKEEPER_CPU_REQUESTS	Specifies the amount of CPU initially requested by Zookeeper in milliCPUs. The default value is `100m` .
ZOOKEEPER_MEMORY_REQUESTS	Specifies the amount of memory initially requested by Zookeeper in mebibytes. The default value is `512Mi` .
ZOOKEEPER_CPU_LIMITS	Specifies the maximum amount of CPU that Zookeeper can request in milliCPUs. The default value is `200m` .
ZOOKEEPER_MEMORY_LIMITS	Specifies the maximum amount of memory that Zookeeper can request in mebibytes. The default value is `1024Mi` .
ZOOKEEPER_BROKER_CPU_REQUESTS	Specifies the amount of CPU initially requested by the Zookeeper Broker in milliCPUs. The default value is `1000m` .
ZOOKEEPER_BROKER_MEMORY_REQUESTS	Specifies the amount of memory initially requested by the Zookeeper Broker in mebibytes. The default value is `1024Mi` .

Variable	Description
ZOOKEEPER_BROKER_CPU_LIMITS	Specifies the maximum amount of CPU that the Zookeeper Broker can request in milliCPUs. The default value is `2000m` .
ZOOKEEPER_BROKER_MEMORY_LIMITS	Specifies the maximum amount of memory that the Zookeeper Broker can request in mebibytes. The default value is `4096Mi` .
VERNE_CPU_REQUESTS	Specifies the amount of CPU initially requested by Verne in milliCPUs. The default value is `1000m` .
VERNE_MEMORY_REQUESTS	Specifies the amount of memory initially requested by Verne in mebibytes. The default value is `1024Mi` .
VERNE_CPU_LIMITS	Specifies the maximum amount of CPU that Verne can request in milliCPUs. The default value is `2000m` .
VERNE_MEMORY_LIMITS	Specifies the maximum amount of memory that the Verne can request in mebibytes. The default value is `4096Mi` .
CTPROCESSOR_CPU_REQUESTS	Specifies the amount of CPU initially requested by the CTProcessor in milliCPUs. The default value is `200m` .

Variable	Description
CTPROCESSOR_MEMORY_REQUESTS	Specifies the amount of memory initially requested by CTProcessor in mebibytes. The default value is `2564Mi`.
CTPROCESSOR_CPU_LIMITS	Specifies the maximum amount of CPU that CTProcessor can request in milliCPUs. The default value is `1500m`.
CTPROCESSOR_MEMORY_LIMITS	Specifies the maximum amount of memory that the CTProcessor can request in mebibytes. The default value is `1024Mi`.
KAFKA_CONNECT_CPU_REQUESTS	Specifies the amount of CPU initially requested by Kafka Connect in milliCPUs. The default value is `100m`.
KAFKA_CONNECT_MEMORY_REQUESTS	Specifies the amount of memory initially requested by Kafka Connect in mebibytes. The default value is `2564Mi`.
KAFKA_CONNECT_CPU_LIMITS	Specifies the maximum amount of CPU that Kafka Connect can request in milliCPUs. The default value is `1000m`.
KAFKA_CONNECT_MEMORY_LIMITS	Specifies the maximum amount of memory that the Kafka Connect can request in mebibytes. The default value is `512Mi`.

Variable	Description
	.
CTAPI_CPU_REQUESTS	Specifies the amount of CPU initially requested by `CTAPI` in milliCPUs. The default value is `100m` .
CTAPI_MEMORY_REQUESTS	Specifies the amount of memory initially requested by `CTAPI` in mebibytes. The default value is `245Mi` .
CTAPI_CPU_LIMITS	Specifies the maximum amount of CPU that `CTAPI` can request in milliCPUs. The default value is `1000m` .
CTAPI_MEMORY_LIMITS	Specifies the maximum amount of memory that the `CTAPI` can request in mebibytes. The default value is `2048Mi` .
CWPWEB_CPU_REQUESTS	Specifies the amount of CPU initially requested by CWWeb in milliCPUs. The default value is `200m` .
CWPWEB_MEMORY_REQUESTS	Specifies the amount of memory initially requested by CWWweb in mebibytes. The default value is `512Mi` .
CWPWEB_CPU_LIMITS	Specifies the maximum amount of CPU that CWWeb can request in milliCPUs. The default value is `1000m` .

Variable	Description
CWPWEB_MEMORY_LIMITS	Specifies the maximum amount of memory that the CWWeb can request in mebibytes. The default value is `2048Mi` .

3. Save the file.

The logging environment variables are described in the *Troubleshooting* chapter.

Encrypting the Passwords for Kubernetes Components

An RSA key pair protects the password of the Active Directory account that is used as a `Connected Worker Platform` service account.The password is encrypted using the public key and is stored in the *env* file. The private key is passed to the CWP containers for decrypting the password.

This section describes the instructions to harden the passwords that are contained in the *env* file and accessed during installation of the Kubernetes components.

Note: Run the *./init-crypto* script, before you run the `./cwp` *`env_name`* `up` command.

Ensure that you only run the init-crypto script once, unless the .env and certs directory is overridden.

1. On the Kubernetes Administration Terminal, open a **`bash`** terminal, and then change to the *../deploy/kube* directory.
2. Type `./init-crypto` *`env_name`*.
3. When prompted, specify the password for each Kubernetes component.

   The script encrypts the following passwords and stores them in the */* *`conkeyref="prod_names/cwp_deploy"`**/deploy/kube/.env* environment variable file:

   - *`CORP_LDAP_PASSWORD`* - Password used to log into the NES Administrator Console.
   - *`KAFKA_SSL_KEYSTORE_PASSWORD`* - Kafka broker TLS certificate keystore password.
   - *`KAFKA_SASL_ADMIN_PASSWORD`* - Kafka SASL admin password.
   - *`KAFKA_SASL_CTPROCESSOR_PASSWORD`* - Kafka password for the CT Processor.
   - *`KAFKA_SASL_CTCA_PASSWORD`* - Kafka password for the EA.

Launching the Kubernetes Environment

Ensure that the extracted CWP installation package has been copied to the Kubernetes Administration Terminal.

The instructions in this section enables you to launch the Kubernetes environment with 3 nodes.

1. On the Kubernetes Administration Terminal, open a **`bash`** terminal, and then change to *`CWP_12_deployment_folder`**/deploy/kube* folder.

2. Initialize the client by typing one of the following commands:
   - For bare-metal deployments, type `./init-client -m user@master-node`

     Where:

     - *`master-node`* is the address of the initial master node
     - *`user`* is a user on the node.
   - For AWS deployments, type `./init-client -a aws-region -c cluster name`

     Where:

     - `aws-region` is the AWS region where the cluster resides, and
     - `cluster name` is the name of the EKS cluster
3. Update the environment by typing `./cwp update prod update`
4. Launch the environment by typing `./cwp prod up`.

Creating a Backup of the Data Folders

After you launch the `Connected Worker Platform` platform and terminate the respective containers, ensure that you save the contents of the following data folders under the host(s):

- *runtime/kafka/data/kafka*.
- *runtime/zookeeper/data/Zookeeper*.

Smart Distancing and Contact Tracing

Nymi offers a smart distancing and contact tracing solution that will allow users to accurately track contact events and encourage social distancing behavior.

To implement the Smart Distancing and Contact Tracing (SDCT) functionality, you will install the Edge Agents(EA) on each user terminal in the environment and use the NES Administrator Console to enable settings in the NES active policy.

Customizing Smart Distancing and Contact Tracing

Nymi offers a smart distancing and contact tracing solution that will allow users to accurately track contact events and encourage social distancing behavior. Smart Distancing and Contact Tracing (SDCT) functionality is enabled through the NES Administrator Console.

Smart Distancing and Contact Tracing enables the Nymi Band to scan for other authenticated Nymi Bands in vicinity. When two or more Nymi Bands remain in close proximity of each other for 5 consecutive minutes, each Nymi Band logs a proximity event. Proximity events are sent to the SDCT services for analysis. When a Nymi Band reports 3 proximity events (cumulative total of 15 minutes) with another Nymi Band over a 24-hour period, a contact event is recorded in a `CWP Database`. Optionally, you can configure the Nymi Band to generate smart distancing reminders. When users are in close proximity of each other for 5 minutes, a smart distancing reminder causes the Nymi Band to vibrate and display a message to the users advising them to maintain social distancing.

Configuring SDCT

Perform the following steps to enable the Smart Distancing and Contact Tracing functionality on the Nymi Band during enrollment. This includes **`Smart Distancing Reminders`** and contact tracing event collection.

1. Log in to the NES Administrator Console with an account that is an NES Administrator.
2. On the navigation bar, click **`Policies`**.
3. In the `Policies` window, select the active policy.
4. Select the **`Smart Distancing and Contact Tracing`** option.
   The **`Smart Distancing Reminders`** option appears.
5. Select **`Smart Distancing Reminders`**, to allow users to receive smart distancing reminders on their Nymi Bands.
6. Click **`Save`**.

During enrollment the `Nymi Band Application` updates the Nymi Band to enable SDCT support.

Changing this option does not change the Nymi Band behaviour for existing enrolled Nymi Bands until the user logs into the `Nymi Band Application` while wearing their authenticated Nymi Band.

Disabling SDCT options

Perform the following steps to disable contact tracing and smart distancing reminders.

1. Log in to the NES Administrator Console with an account that is an NES Administrator.
2. On the navigation bar, click **Policies**.
3. In the Policies window, select the active policy.
4. Perform one of the following actions:
   - To disable the smart distancing reminders that a user receives when a Nymi Band detects that it is in close proximity of another Nymi Band for 5 consecutive minutes, clear the **Smart Distancing Reminders** option.

     Note: TheNymi Band continues to record proximity events.
   - To prevent the solution from recording proximity and contact tracing events, and providing users with smart distancing reminders, clear the **Smart Distancing and Contact Tracing** option
5. Click **Save**.

Changing this option does not change the Nymi Band behaviour for existing enrolled Nymi Bands until the user logs into the Nymi Band Application while wearing their authenticated Nymi Band.

Install Edge Agents and Nymi-Based Applications

Install the Edge Agents(EA) and the Nymi Runtime software on User Terminals or the RDP session host / Citrix server in the environment to collect information from Nymi Bands.

The installation process differs for local and RDP/Citrix configurations:

- In a local configuration, install EA and the Nymi Runtime application on each thick client user terminal. The Nymi Connected Worker Platform Administration Guide provides detailed information about how to install Nymi Runtime.
- In a remote configuration install:
  - EA on a RDP session host or Citrix server, which installs the Edge Agent service. When a user logs into a remote session, one EdgeAgentsLauncher process and one EdgeAgents process start for each user session.
  - Nymi Bluetooth Endpoint application on each thin client user terminal.
  - Nymi Agent application on any server in the environment.

  The Nymi Connected Worker Platform NES Deployment Guide provides detailed information about how to install and configure the Nymi Bluetooth Endpoint and the Nymi Agent application for configurations that use RDP or Citrix.

Edge Agent Certificate Requirements

Kafka requires a TLS certificate that is issued by a trusted public CA or trusted enterprise CA. If this is not possible, for example in a POC or Pilot environment), then you can use a TLS certificate that is issued by an untrusted root CA. Additional steps are required and described in the following sections.

Note: Self-signed certificates with Kafka are not supported.

Installing the Edge Agents Application on the RDP session host / Citrix server

Install EA on the RDP session host / Citrix server in your environment that is on the same domain as NES.

- Install the latest version of OpenSSL for Windows and add the *bin* directory is included in the system path.
- The Edge Agents package has been extracted to a central location.
- If an untrusted root CA issues the NES certificate, you must import the root CA certificate for the untrusted root. *Importing the Root CA certificate* provides more information.

Perform the following steps on the RDP session host / Citrix server:

1. Copy the extracted *edgeagents* folder to the RDP session host / Citrix server.

   The folder contains the following files:

   - *decrypt.key* file, which is used to decrypt the SASL and NES usernames and passwords.
   - *edgeagents-service-x64-`version`.msi* file, which installs the EA software on a thick client user terminal and uses the parameters detailed in the *edge_agents.conf* file.
   - *edgeagents-terminal-service-x64-`version`.msi* file, which installs the EA software on a RDP sessions host/Citrix server and uses the parameters detailed in the *edge_agents.conf* file.
   - *secretutil.cmd* file, which is Windows command utility that encrypts secrets.
   - *edge_agents.conf* file, which used to configure the parameters of the EA installation, and includes keys generated from the PowerShell utility.
   - *KafkaCA.pem* file, which is a default client truststore certificate.
2. Perform the following steps to generate the secret keys by using the *secretutil.cmd* file.
   a) Click the **`Start`** menu and type `cmd`. Right-click **`Command Prompt`** and click **`Run as administrator`**.
   b) Change the directory that contains the extracted EA installation package. For example, the *C:\edgeagents* folder.
   c) Initialize *secretutil.cmd* with the following command:

   ```
   secretutil.cmd -init
   ```

   d) Use the *secretutil.cmd* command to encrypt the sasl username and password.

   1. By default, the username is ctca. type the following command to encrypt the username in an output file.

      ```
      secretutil.cmd -enc ctca>OUTPUT FILE NAME.txt
      ```

      where *`OUTPUT FILE NAME.txt`* is the name of the file that contains the encrypted username.
   2. Type the following command to encrypt the SASL password.

      ```
      secretutil.cmd -enc PASSWORD>OUTPUT FILE NAME 2.txt
      ```

      where *`PASSWORD`* is the password specified by the person who implemented the server side components when they ran the *init-crypto* command and *`OUTPUT FILE NAME 2.txt`* is the name of the file that contains the encrypted password.

   The output files contain the secret keys used in the *edge_agents.conf* file.

3. Perform the following steps to update the *edge_agents.conf* file with the secret keys that you created in the previous step.
   a) Open the *edge_agents.conf* file with a text editor.
   b) Update the value for the key sasl.username. It is the encrypted value in the username output text file.

   ```
   sasl.username=[encrypted username]
   ```

   c) Update the value for the key sasl.password. It is encrypted value in the password output text file.

   ```
   sasl.password=[encrypted password]
   ```

4. Save the *edge_agents.conf* file.
5. For Kafka TLS certificates that are issued from an untrusted CA only, perform the following steps to install the Kafka Truststore.
   a) Obtain the Kafka Broker root CA certificate from the person who implemented the CWP cluster. The file is stored in the CWP deployment package, in the *cwp/certs* folder.
   b) If required, rename the Kafka Broker root CA cert to *KafkaCA.pem*.
   c) Backup the *KafkaCA.pem* certificate in the Edge Agents installation directory.
   d) Replace the default *KafkaCA.pem* file in the Edge Agents installation package directory with the new *KafkaCA.pem* certificate file that you obtained from the implementation engineer.
6. Open the *edge_agents.conf*, and ensure that the value defined in the `sasl.ca.path` key is *C:\Nymi\Edge_Agents\certs\KafkaCA.pem*.
7. Uncomment the line `launcher.mode = 1`.
8. Edit the following configuration parameters in the *edge_agents.conf* file.

   Producer Specific Properties:

   - `bootstrap.servers`, which defines a list of host and port pairs of Kafka brokers.

   NES Specific Properties:

   - `nes.url`, which specifies the NES URL.
   - `agent.url`, which specifies the Nymi Agent URL. When you do not specifiy a value, EA will pick up the local Nymi Agent URL.

9. Save the *edge_agents.conf* file.

   Note: Ensure the *edge_agents.conf* file is configured prior to installing *edge_agents.msi*. This configuration file can then be copied to different machines being installed with EA.

10. Run the installer file *edgeagents-terminal-services-x64-`version`.msi*.

    The Edge Agents application is installed in the *C:\Nymi\Edge_Agents* folder and an EdgeAgent service starts. Each time a user logs in to a Citrix or RDP session, an EdgeAgentLauncher process starts and the EdgeAgent service starts an EdgeAgent process. When the user session ends, the additional EdgeAgent and EdgeAgentLauncher processes terminate.

    Note: The section *Edge Agents Log Files* provides information about EdgeAgents log files.

Installing the Edge Agents Application on VMWare Horizon

Install EA on the VMWare Horizon server in your environment that is on the same domain as NES.

- Install the latest version of OpenSSL for Windows and add the *bin* directory is included in the system path.
- The Edge Agents package has been extracted to a central location.
- If an untrusted root CA issues the NES certificate, you must import the root CA certificate for the untrusted root. *Importing the Root CA certificate* provides more information.

Perform the following steps on the RDP session host / Citrix server:

1. Copy the extracted *edgeagents* folder to the RDP session host / Citrix server.

   The folder contains the following files:

   - *decrypt.key* file, which is used to decrypt the SASL and NES usernames and passwords.
   - *edgeagents-service-x64-`version`.msi* file, which installs the EA software on a thick client user terminal and uses the parameters detailed in the *edge_agents.conf* file.
   - *edgeagents-terminal-service-x64-`version`.msi* file, which installs the EA software on a RDP sessions host/Citrix server and uses the parameters detailed in the *edge_agents.conf* file.
   - *secretutil.cmd* file, which is Windows command utility that encrypts secrets.
   - *edge_agents.conf* file, which used to configure the parameters of the EA installation, and includes keys generated from the PowerShell utility.
   - *KafkaCA.pem* file, which is a default client truststore certificate.
2. Perform the following steps to generate the secret keys by using the *secretutil.cmd* file.
   a) Click the **`Start`** menu and type `cmd`. Right-click **`Command Prompt`** and click **`Run as administrator`**.
   b) Change the directory that contains the extracted EA installation package. For example, the *C:\edgeagents* folder.
   c) Initialize *secretutil.cmd* with the following command:

   ```
   secretutil.cmd -init
   ```

   d) Use the *secretutil.cmd* command to encrypt the sasl username and password.

      1. By default, the username is ctca. type the following command to encrypt the username in an output file.

      ```
      secretutil.cmd -enc ctca>OUTPUT FILE NAME.txt
      ```

      where *`OUTPUT FILE NAME.txt`* is the name of the file that contains the encrypted username.
      2. Type the following command to encrypt the SASL password.

      ```
      secretutil.cmd -enc PASSWORD>OUTPUT FILE NAME 2.txt
      ```

      where *`PASSWORD`* is the password specified by the person who implemented the server side components when they ran the *init-crypto* command and *`OUTPUT FILE NAME 2.txt`* is the name of the file that contains the encrypted password.

   The output files contain the secret keys used in the *edge_agents.conf* file.

3. Perform the following steps to update the *edge_agents.conf* file with the secret keys that you created in the previous step.
   a) Open the *edge_agents.conf* file with a text editor.
   b) Update the value for the key sasl.username. It is the encrypted value in the username output text file.

   ```
   sasl.username=[encrypted username]
   ```

   c) Update the value for the key sasl.password. It is encrypted value in the password output text file.

   ```
   sasl.password=[encrypted password]
   ```

4. Save the *edge_agents.conf* file.
5. For Kafka TLS certificates that are issued from an untrusted CA only, perform the following steps to install the Kafka Truststore.
   a) Obtain the Kafka Broker root CA certificate from the person who implemented the CWP cluster. The file is stored in the CWP deployment package, in the *cwp/certs* folder.
   b) If required, rename the Kafka Broker root CA cert to *KafkaCA.pem*.
   c) Backup the *KafkaCA.pem* certificate in the Edge Agents installation directory.
   d) Replace the default *KafkaCA.pem* file in the Edge Agents installation package directory with the new *KafkaCA.pem* certificate file that you obtained from the implementation engineer.
6. Open the *edge_agents.conf*, and ensure that the value defined in the `sasl.ca.path` key is *C:\Nymi\Edge_Agents\certs\KafkaCA.pem*.
7. Uncomment the line `launcher.mode = 1`.
8. Uncomment the line `launcher.vmwarehorizon` and change the value to 1.
9. Uncomment the line `launcher.vmwarehorizon.remoteipmode`, and set the value to one of the following numbers, depending on your configuration .
   - If the `HKEY_CURRENT_USER\Volatile Environment\ViewClient_IP_Address` key defines the remote IP address, set the value to `1`.
   - If the `HKEY_CURRENT_USER\Volatile Environment\ViewClient_Broker_Remote_IP_Address` key defines the remote IP address, set the value to `0`.
10. Edit the following configuration parameters in the *edge_agents.conf* file.

    Producer Specific Properties:

    - `bootstrap.servers`, which defines a list of host and port pairs of Kafka brokers.

    NES Specific Properties:

    - `nes.url`, which specifies the NES URL.
    - `agent.url`, which specifies the Nymi Agent URL. When you do not specifiy a value, EA will pick up the local Nymi Agent URL.
11. Save the *edge_agents.conf* file.

    Note: Ensure the *edge_agents.conf* file is configured prior to installing *edge_agents.msi*. This configuration file can then be copied to different machines being installed with EA.

12. Run the installer file *edgeagents-terminal-services-x64-`version`.msi*.

    The Edge Agents application is installed in the *C:\Nymi\Edge_Agents* folder and an EdgeAgent service starts. Each time a user logs in to a Citrix or RDP session, an EdgeAgentLauncher process starts and the EdgeAgent service starts an EdgeAgent process. When the user session ends, the additional EdgeAgent and EdgeAgentLauncher processes terminate.

    Note: The section *Edge Agents Log Files* provides information about EdgeAgents log files.

Installing the Edge Agents Application in a Local Configuration

Install EA on the user terminals in your environment that are on the same domain as NES.

- Install the latest version of OpenSSL for Windows and add the *bin* directory is included in the system path.
- The Edge Agents package has been extracted to a central location.

Perform the following steps on each user terminal:

1. Copy the extracted *edgeagents* folder to the user terminal.

   The folder contains the following files:

   - *decrypt.key* file, which is used to decrypt the SASL and NES usernames and passwords.
   - *edgeagents-service-x64-`version`.msi* file, which installs the EA software on a thick client user terminal and uses the parameters detailed in the *edge_agents.conf* file.
   - *edgeagents-terminal-service-x64-`version`.msi* file, which installs the EA software on a RDP sessions host/Citrix server and uses the parameters detailed in the *edge_agents.conf* file.
   - *secretutil.cmd* file, which is Windows command utility that encrypts secrets.
   - *edge_agents.conf* file, which used to configure the parameters of the EA installation, and includes keys generated from the PowerShell utility.
   - *KafkaCA.pem* file, which is a default client truststore certificate.

2. Perform the following steps to generate the secret keys by using the *secretutil.cmd* file.
   a) Click the **`Start`** menu and type `cmd`. Right-click **`Command Prompt`** and click **`Run as administrator`**.
   b) Change the directory that contains the extracted EA installation package. For example, the *C:\edgeagents* folder.
   c) Initialize *secretutil.cmd* with the following command:

```
secretutil.cmd -init
```

   d) Use the *secretutil.cmd* command to encrypt the sasl username and password.
      1. By default, the username is ctca. type the following command to encrypt the username in an output file.

```
secretutil.cmd -enc ctca>OUTPUT FILE NAME.txt
```

         where *`OUTPUT FILE NAME.txt`* is the name of the file that contains the encrypted username.
      2. Type the following command to encrypt the SASL password.

```
secretutil.cmd -enc PASSWORD>OUTPUT FILE NAME 2.txt
```

         where *`PASSWORD`* is the password specified by the person who implemented the server side components when they ran the *init-crypto* command and *`OUTPUT FILE NAME 2.txt`* is the name of the file that contains the encrypted password.

   The output files contain the secret keys used in the *edge_agents.conf* file.
3. Perform the following steps to update the *edge_agents.conf* file with the secret keys that you created in the previous step.
   a) Open the *edge_agents.conf* file with a text editor.
   b) Update the value for the key `sasl.username`. It is the encrypted value in the username output text file.

```
sasl.username=[encrypted username]
```

   c) Update the value for the key `sasl.password`. It is encrypted value in the password output text file.

```
sasl.password=[encrypted password]
```

4. Save the *edge_agents.conf* file.
5. For Kafka TLS certificates that are issued from an untrusted CA only, perform the following steps to install the Kafka Truststore.
   a) Obtain the Kafka Broker root CA certificate from the person who implemented the CWP cluster. The file is stored in the CWP deployment package, in the *cwp/certs* folder.
   b) If required, rename the Kafka Broker root CA cert to *KafkaCA.pem*.
   c) Backup the *KafkaCA.pem* certificate in the Edge Agents installation directory.
   d) Replace the default *KafkaCA.pem* file in the Edge Agents installation package directory with the new *KafkaCA.pem* certificate file that you obtained from the implementation engineer.
6. Open the *edge_agents.conf*, and ensure that the value defined in the *`sasl.ca.path`* key is *C:\Nymi\Edge_Agents\certs\KafkaCA.pem*.

7. Edit the following configuration parameters in the *edge_agents.conf* file.

   Producer Specific Properties:

   - `bootstrap.servers`, which defines a list of host and port pairs of Kafka brokers.

   NES Specific Properties:

   - `nes.url`, which specifies the NES URL.
   - `agent.url`, which specifies the Nymi Agent URL. When you do not specifiy a value, EA will pick up the local Nymi Agent URL.

8. Save the *edge_agents.conf* file.

   Note: Ensure the *edge_agents.conf* file is configured prior to installing *edge_agents.msi*. This configuration file can then be copied to different machines being installed with EA.

9. Run the installer file *edgeagents-service-x64-`version`.msi*.
   The Edge Agents application is installed without any user interactions in the *C:\Nymi\Edge_Agents* folder and the .Nymi Edge Agents service appears with a Running status in Windows Services.

   Note: The section *Edge Agents Log Files* provides information about Edge Agents log files.

Encrypting the Key File with EFS

The `Edge Agents` service runs under the Network Service account. Nymi recommends that you encrypt the *decrypt.key* file with EFS on each host after you install the `Edge Agents` application.

Nymi provides you with a script to perform the encryption.

1. Run Power Shell as an administrator.
2. Enable the ability to run a script file in one of the following ways:

   - By setting the execution policy to `RemoteSigned`.

     a. From Power Shell, type `get-executionpolicy`.

        The output displays the current execution policy setting.

     b. Type `set-executionpolicy RemoteSigned`, and when prompted, type `Y`

        The execution policy is set to RemoteSigned.

     MSDN provides more information about how to modify execution policy settings.

   - By unblocking the script file.

     a. From Power Shell, type `unblock-file c:\Nymi\Edge_Agents\tools\encrypt-withEFS.ps1`

     MSDN" provides more information about how to unblock script files.

3. Change to the *c:\Nymi\Edge_Agents\tools* directory.
4. Type *.\encrypt-withEFS.ps1 -file ..\conf\certs\decrypt.key*
5. Optional, use the **`set-executionpolicy`** command to set the execution policy to the original value.

Optional, Updating the hosts File for CWP Components

If you cannot use DNS to map external AWS components to the internal FQDN for the component, update *hosts* file on each user terminal.

Use the **`get-lbs`** command to get a list of load balancers and then the **`nslookup`** command to determine the IP address.

1. Perform the following steps on the Kubernetes Administration Terminal from the bash terminal.
   a) Change to the *`master deployment folder`/deploy/kube* folder.
   b) To get a list of load balancers, type `./get-lbs prod`.
      The command displays output similar to the following:

```
                                    cwpweb IN CNAME
 aaec3da308c0c4755b5d64bc1f9cc2a8-06727299dacf77ad.xyz.cab-
central-1.amazonaws.com.
                                    cwp-broker-2 IN CNAME
 aada3d877365c41ad938e73d4e2ab48a-730a1b35afac8d48.xyz.cab-
central-1.amazonaws.com.
                                    bootstrap IN CNAME
 a40ff19ccb5ad474b9acb48c26c29936-93ac7d6f74fd12dd.xyz.cab-
central-1.amazonaws.com.
                                    cwp-broker-0 IN CNAME
 aa0d5d9cb8b6f47d5a826cf441490596-b82a18c161e6cae5.xyz.cab-
central-1.amazonaws.com.
                                    cwp-broker-1 IN CNAME
 aa3aa3245e4f242cc9ffe88e228cdeb7-ee12ab370ba83f63.xyz.cab-
central-1.amazonaws.com.
                                    ctapi IN CNAME
 a175dbca766b54a27bbc5a19d5dcad01-fa0ef3f5885fcf2e.xyz.cab-
central-1.amazonaws.com.
```

   c) Use the **`nslookup`** command to determine the IP address that is associated with each component.
   d) Record the AWS address, IP address, and internal FQDN of each component in the *C:\Windows\System32\drivers\etc\hosts* file.

      Note: Edit the in `Notepad ++`. When you save the file, you will be prompted to open the file in adminsitrators mode, click **`OK`**.

      The following provides an example of a hosts file for a CWP environment.

```
                                    ######################### Kafka
 Broker ##########################################
                                    # DNS:
 a2ad5527103ba435dab025cbb5e5fb1f-1133710120.us-
east-2.elb.amazonaws.com
                                    3.15.88.46 kafka.cwp.qa-lab
```

```
                                    # DNS:
 aa0d5d9cb8b6f47d5a826cf441490596-b82a18c161e6cae5.xyz.cab-
central-1.amazonaws.com.
                                    18.116.180.69
 broker-0.broker.cwp.svc.cluster.local

                                    # DNS:
 aa3aa3245e4f242cc9ffe88e228cdeb7-ee12ab370ba83f63.xyz.cab-
central-1.amazonaws.com.
                                    3.140.220.152
 broker-1.broker.cwp.svc.cluster.local

                                    # DNS:
 aada3d877365c41ad938e73d4e2ab48a-730a1b35afac8d48.xyz.cab-
central-1.amazonaws.com.
                                    3.131.241.193
 broker-2.broker.cwp.svc.cluster.local

                                    ######################## CTAPI
 ##################################################
                                    # DNS:
 a175dbca766b54a27bbc5a19d5dcad01-fa0ef3f5885fcf2e.xyz.cab-
central-1.amazonaws.com
                                    18.223.249.217
 ctapi.lab.nymi.com

                                    ######################## CWPWEB
 ##################################################
                                    #DNS:
 aaec3da308c0c4755b5d64bc1f9cc2a8-06727299dacf77ad.xyz.cab-
central-1.amazonaws.com.
                                    18.223.249.216
 cwpweb.lab.nymi.com
```

a) Put a copy of the *hosts* file in a shared location that is accessible to the user terminals in the environment.

2. On each user terminal, replace the *hosts* file with a copy of the *hosts* file in the shared location.

Use the Contact Tracing Dashboard

Access the Contact Tracing Dashboard to visualize and analyze contact tracing data for employees that are enrolled in the Contact Tracing program.

Proximity events that are logged in each employee's Nymi Band are uploaded to the Contact Tracing Dashboard via the Edge Agents (EA) that is installed on a user terminal. These contact events are viewed on the Contact Tracing Dashboard, where they can be analyzed to provide:

- Timely and targeted response for positive diagnosis.
- Timely and reliable contact tracing history.
- Coverage information and trends on social distancing performance.
- Targeted intervention to modify behavior and prevent an outbreak.

Accessing the Contact Tracing Dashboard

Connect to the Contact Tracing server in a web browser.

1. From web browser, navigate to `https://FQDN_CTAPI/dashboard`.
   where *`FQDN_CTAPI`* is the FQDN of the virtual server on which `CTAPI` resides.
2. Log in using the Contact Tracing Dashboard username and password. Alternatively, you can log in with the credentials of an NES Administrator.
   The format of the username is *`username@domain`*. For example, for user dredmond in domain qa-lab.local, specify the username as `dredmond@qa-lab.local`.

Viewing the Contact Tracing Dashboard

The Contact Tracing Dashboard is used to visualize and analyze contact tracing data for employees enrolled in the Contact Tracing program. This section provides detailed information on viewing the Contact Tracing Dashboard and analyzing the contact tracing data.

Contact Tracing Dashboard Example

The following figure shows the Contact Tracing Dashboard

Figure 4: Example of the Contact Tracing Dashboard

Enrolled Employees

Identifies the number of employees that are enrolled in the Smart Distancing and Contact Tracing program.

Social Distancing Compliance (%)

Identifies the percentage of employees who have had no contact with other employees for a given day.

The higher the score, the more compliant employees are in maintaining distancing. This report is based on data from the last 30 days.

For example, with a program population of 100 employees, if 2 people are in contact with each other, the compliance score drops to 98%.

Average and Maximum Contacts (%)

The average identifies the percentage of protected employees that are contacted by an individual expressed as an average across all employees registered in the program. The maximum identifies the highest percentage of employees contacted by any single individual.

For example, with a program population of 100 employees, if on a given day, only one employee came in contact with 10 people (i.e., 10% of the enrolled employee population), the average contact percentage would be 0.1% and maximum contact % would be 10% for that day.

Most Contacted Employees

Identifies the employees who have the most contact events in the program. This report is based on data from the last 30 days.

Total Contacts by Day

Identifies The total number of contact events within the program per day.

A contact event is recorded when a Nymi Band user is closer than ~2 meters from another Nymi Band for approximately 15 or more cumulatative minutes over a 24-hour period.

Most Contacted Employee Details

Identifies the number of contact event and the employees contacted for the employee with the most contact events.

This report is based on data from the last 30 days.

Employee Contact Timeline

Provides adminstrators with a timeline of all contact events for a specific employee within a date range.

You can search by username or date. Consider the following:

- Default date range is last 30 days.
- Search terms are case sensitive.
- Timestamp for a contact event is date and time of the last proximity event.
- Timestamps appear in the timezone that is defined on the browser that you use to access the Contact Tracing Dashboard

Update Smart Distancing and Contact Tracing

Review this section to determine how to update the user terminals and Kubernetes environment when there is a new CWP release.

Update the User Terminal

On each user terminal, you must, upgrade Nymi Runtime, remove the Contact Tracing Collection Agent (CTCA) and install `Edge Agents`.

Note: The Nymi Connected Worker Platform NES Deployment Guide describes how to update the Nymi Runtime.

(CWP 1.1.x only) Uninstalling the Contact Tracing Collection Agent

CWP 1.2 does not make use of CTCA on user terminals.

Perform the following steps on each user terminal on which you installed CTCA for CWP 1.1.

1. From the desktop, go to **`Start > Settings > Apps > Apps and Features`**.

   Note: You may also go to **`Start > Control Panel > Programs > Programs and Features`**.
2. Click **`Contact Tracing Collection Agent`**, and then select **`Uninstall`**.

(CWP 1.2 only) Uninstalling the Edge Agents Application

Perform the following steps on each user terminal on which you installed `Edge Agents` for CWP 1.2.

1. From the desktop, go to **`Start > Settings > Apps > Apps and Features`**.

   Note: You may also go to **`Start > Control Panel > Programs > Programs and Features`**.
2. Click **`Edge Agents`**, and then select **`Uninstall`**.

Install Edge Agents and Nymi-Based Applications

Install the `Edge Agents`(EA) and the `Nymi Runtime` software on User Terminals or the RDP session host / Citrix server in the environment to collect information from Nymi Bands.

The installation process differs for local and RDP/Citrix configurations:

- In a local configuration, install EA and the `Nymi Runtime` application on each thick client user terminal. The Nymi Connected Worker Platform Administration Guide provides detailed information about how to install `Nymi Runtime`.

- In a remote configuration install:
  - EA on a RDP session host or Citrix server, which installs the Edge Agent service. When a user logs into a remote session, one EdgeAgentsLauncher process and one EdgeAgents process start for each user session.
  - `Nymi Bluetooth Endpoint` application on each thin client user terminal.
  - `Nymi Agent` application on any server in the environment.

  The Nymi Connected Worker Platform NES Deployment Guide provides detailed information about how to install and configure the `Nymi Bluetooth Endpoint` and the `Nymi Agent` application for configurations that use RDP or Citrix.

Installing the Edge Agents Application on the RDP session host / Citrix server

Install EA on the RDP session host / Citrix server in your environment that is on the same domain as NES.

- Install the latest version of OpenSSL for Windows and add the *bin* directory is included in the system path.
- The Edge Agents package has been extracted to a central location.
- If an untrusted root CA issues the NES certificate, you must import the root CA certificate for the untrusted root. *Importing the Root CA certificate* provides more information.

Perform the following steps on the RDP session host / Citrix server:

1. Copy the extracted *edgeagents* folder to the RDP session host / Citrix server.

   The folder contains the following files:

   - *decrypt.key* file, which is used to decrypt the SASL and NES usernames and passwords.
   - *edgeagents-service-x64-`version`.msi* file, which installs the EA software on a thick client user terminal and uses the parameters detailed in the *edge_agents.conf* file.
   - *edgeagents-terminal-service-x64-`version`.msi* file, which installs the EA software on a RDP sessions host/Citrix server and uses the parameters detailed in the *edge_agents.conf* file.
   - *secretutil.cmd* file, which is Windows command utility that encrypts secrets.
   - *edge_agents.conf* file, which used to configure the parameters of the EA installation, and includes keys generated from the PowerShell utility.
   - *KafkaCA.pem* file, which is a default client truststore certificate.

2. Perform the following steps to generate the secret keys by using the *secretutil.cmd* file.
   a) Click the **`Start`** menu and type `cmd`. Right-click **`Command Prompt`** and click **`Run as administrator`**.
   b) Change the directory that contains the extracted EA installation package. For example, the *C:\edgeagents* folder.
   c) Initialize *secretutil.cmd* with the following command:

```
secretutil.cmd -init
```

   d) Use the *secretutil.cmd* command to encrypt the sasl username and password.
      1. By default, the username is ctca. type the following command to encrypt the username in an output file.

```
secretutil.cmd -enc ctca>OUTPUT FILE NAME.txt
```

         where *`OUTPUT FILE NAME.txt`* is the name of the file that contains the encrypted username.
      2. Type the following command to encrypt the SASL password.

```
secretutil.cmd -enc PASSWORD>OUTPUT FILE NAME 2.txt
```

         where *`PASSWORD`* is the password specified by the person who implemented the server side components when they ran the *init-crypto* command and *`OUTPUT FILE NAME 2.txt`* is the name of the file that contains the encrypted password.

   The output files contain the secret keys used in the *edge_agents.conf* file.
3. Perform the following steps to update the *edge_agents.conf* file with the secret keys that you created in the previous step.
   a) Open the *edge_agents.conf* file with a text editor.
   b) Update the value for the key `sasl.username`. It is the encrypted value in the username output text file.

```
sasl.username=[encrypted username]
```

   c) Update the value for the key `sasl.password`. It is encrypted value in the password output text file.

```
sasl.password=[encrypted password]
```

4. Save the *edge_agents.conf* file.
5. For Kafka TLS certificates that are issued from an untrusted CA only, perform the following steps to install the Kafka Truststore.
   a) Obtain the Kafka Broker root CA certificate from the person who implemented the CWP cluster. The file is stored in the CWP deployment package, in the *cwp/certs* folder.
   b) If required, rename the Kafka Broker root CA cert to *KafkaCA.pem*.
   c) Backup the *KafkaCA.pem* certificate in the Edge Agents installation directory.
   d) Replace the default *KafkaCA.pem* file in the Edge Agents installation package directory with the new *KafkaCA.pem* certificate file that you obtained from the implementation engineer.
6. Open the *edge_agents.conf*, and ensure that the value defined in the *`sasl.ca.path`* key is *C:\Nymi\Edge_Agents\certs\KafkaCA.pem*.
7. Uncomment the line *`launcher.mode = 1`*.

8. Edit the following configuration parameters in the *edge_agents.conf* file.

   Producer Specific Properties:

   - `bootstrap.servers`, which defines a list of host and port pairs of Kafka brokers.

   NES Specific Properties:

   - `nes.url`, which specifies the NES URL.
   - `agent.url`, which specifies the Nymi Agent URL. When you do not specifiy a value, EA will pick up the local Nymi Agent URL.

9. Save the *edge_agents.conf* file.

   Note: Ensure the *edge_agents.conf* file is configured prior to installing *edge_agents.msi*. This configuration file can then be copied to different machines being installed with EA.

10. Run the installer file *edgeagents-terminal-services-x64-`version`.msi*.

    The Edge Agents application is installed in the *C:\Nymi\Edge_Agents* folder and an EdgeAgent service starts. Each time a user logs in to a Citrix or RDP session, an EdgeAgentLauncher process starts and the EdgeAgent service starts an EdgeAgent process. When the user session ends, the additional EdgeAgent and EdgeAgentLauncher processes terminate.

    Note: The section *Edge Agents Log Files* provides information about EdgeAgents log files.

Installing the Edge Agents Application on VMWare Horizon

Install EA on the VMWare Horizon server in your environment that is on the same domain as NES.

- Install the latest version of OpenSSL for Windows and add the *bin* directory is included in the system path.
- The Edge Agents package has been extracted to a central location.
- If an untrusted root CA issues the NES certificate, you must import the root CA certificate for the untrusted root. *Importing the Root CA certificate* provides more information.

Perform the following steps on the RDP session host / Citrix server:

1. Copy the extracted *edgeagents* folder to the RDP session host / Citrix server.

   The folder contains the following files:

   - *decrypt.key* file, which is used to decrypt the SASL and NES usernames and passwords.
   - *edgeagents-service-x64-`version`.msi* file, which installs the EA software on a thick client user terminal and uses the parameters detailed in the *edge_agents.conf* file.
   - *edgeagents-terminal-service-x64-`version`.msi* file, which installs the EA software on a RDP sessions host/Citrix server and uses the parameters detailed in the *edge_agents.conf* file.
   - *secretutil.cmd* file, which is Windows command utility that encrypts secrets.
   - *edge_agents.conf* file, which used to configure the parameters of the EA installation, and includes keys generated from the PowerShell utility.
   - *KafkaCA.pem* file, which is a default client truststore certificate.

2. Perform the following steps to generate the secret keys by using the *secretutil.cmd* file.
   a) Click the **`Start`** menu and type `cmd`. Right-click **`Command Prompt`** and click **`Run as administrator`**.
   b) Change the directory that contains the extracted EA installation package. For example, the *C:\edgeagents* folder.
   c) Initialize *secretutil.cmd* with the following command:

```
secretutil.cmd -init
```

   d) Use the *secretutil.cmd* command to encrypt the sasl username and password.
      1. By default, the username is ctca. type the following command to encrypt the username in an output file.

```
secretutil.cmd -enc ctca>OUTPUT FILE NAME.txt
```

         where *`OUTPUT FILE NAME.txt`* is the name of the file that contains the encrypted username.
      2. Type the following command to encrypt the SASL password.

```
secretutil.cmd -enc PASSWORD>OUTPUT FILE NAME 2.txt
```

         where *`PASSWORD`* is the password specified by the person who implemented the server side components when they ran the *init-crypto* command and *`OUTPUT FILE NAME 2.txt`* is the name of the file that contains the encrypted password.

   The output files contain the secret keys used in the *edge_agents.conf* file.
3. Perform the following steps to update the *edge_agents.conf* file with the secret keys that you created in the previous step.
   a) Open the *edge_agents.conf* file with a text editor.
   b) Update the value for the key `sasl.username`. It is the encrypted value in the username output text file.

```
sasl.username=[encrypted username]
```

   c) Update the value for the key `sasl.password`. It is encrypted value in the password output text file.

```
sasl.password=[encrypted password]
```

4. Save the *edge_agents.conf* file.
5. For Kafka TLS certificates that are issued from an untrusted CA only, perform the following steps to install the Kafka Truststore.
   a) Obtain the Kafka Broker root CA certificate from the person who implemented the CWP cluster. The file is stored in the CWP deployment package, in the *cwp/certs* folder.
   b) If required, rename the Kafka Broker root CA cert to *KafkaCA.pem*.
   c) Backup the *KafkaCA.pem* certificate in the Edge Agents installation directory.
   d) Replace the default *KafkaCA.pem* file in the Edge Agents installation package directory with the new *KafkaCA.pem* certificate file that you obtained from the implementation engineer.
6. Open the *edge_agents.conf*, and ensure that the value defined in the *`sasl.ca.path`* key is *C:\Nymi\Edge_Agents\certs\KafkaCA.pem*.
7. Uncomment the line *`launcher.mode = 1`*.

8. Uncomment the line *launcher.vmwarehorizon* and change the value to 1.
9. Uncomment the line *launcher.vmwarehorizon.remoteipmode*, and set the value to one of the following numbers, depending on your configuration .
   - If the *HKEY_CURRENT_USER\Volatile Environment\ViewClient_IP_Address* key defines the remote IP address, set the value to 1.
   - If the *HKEY_CURRENT_USER\Volatile Environment \ViewClient_Broker_Remote_IP_Address* key defines the remote IP address, set the value to 0.
10. Edit the following configuration parameters in the *edge_agents.conf* file.

    Producer Specific Properties:

    - *bootstrap.servers*, which defines a list of host and port pairs of Kafka brokers.

    NES Specific Properties:

    - *nes.url*, which specifies the NES URL.
    - *agent.url*, which specifiy the Nymi Agent URL. When you do not specifiy a value, EA will pick up the local Nymi Agent URL.
11. Save the *edge_agents.conf* file.

    Note: Ensure the *edge_agents.conf* file is configured prior to installing *edge_agents.msi*. This configuration file can then be copied to different machines being installed with EA.
12. Run the installer file *edgeagents-terminal-services-x64-version.msi*.

    The Edge Agents application is installed in the *C:\Nymi\Edge_Agents* folder and an EdgeAgent service starts. Each time a user logs in to a Citrix or RDP session, an EdgeAgentLauncher process starts and the EdgeAgent service starts an EdgeAgent process. When the user session ends, the additional EdgeAgent and EdgeAgentLauncher processes terminate.

    Note: The section *Edge Agents Log Files* provides information about EdgeAgents log files.

Installing the Edge Agents Application in a Local Configuration

Install EA on the user terminals in your environment that are on the same domain as NES.

- Install the latest version of OpenSSL for Windows and add the *bin* directory is included in the system path.
- The Edge Agents package has been extracted to a central location.

Perform the following steps on each user terminal:

1. Copy the extracted *edgeagents* folder to the user terminal.

   The folder contains the following files:

   - *decrypt.key* file, which is used to decrypt the SASL and NES usernames and passwords.
   - *edgeagents-service-x64-`version`.msi* file, which installs the EA software on a thick client user terminal and uses the parameters detailed in the *edge_agents.conf* file.
   - *edgeagents-terminal-service-x64-`version`.msi* file, which installs the EA software on a RDP sessions host/Citrix server and uses the parameters detailed in the *edge_agents.conf* file.
   - *secretutil.cmd* file, which is Windows command utility that encrypts secrets.
   - *edge_agents.conf* file, which used to configure the parameters of the EA installation, and includes keys generated from the PowerShell utility.
   - *KafkaCA.pem* file, which is a default client truststore certificate.
2. Perform the following steps to generate the secret keys by using the *secretutil.cmd* file.
   a) Click the **`Start`** menu and type `cmd`. Right-click **`Command Prompt`** and click **`Run as administrator`**.
   b) Change the directory that contains the extracted EA installation package. For example, the *C:\edgeagents* folder.
   c) Initialize *secretutil.cmd* with the following command:

   ```
   secretutil.cmd -init
   ```

   d) Use the *secretutil.cmd* command to encrypt the sasl username and password.

      1. By default, the username is ctca. type the following command to encrypt the username in an output file.

      ```
      secretutil.cmd -enc ctca>OUTPUT FILE NAME.txt
      ```

      where *`OUTPUT FILE NAME.txt`* is the name of the file that contains the encrypted username.
      2. Type the following command to encrypt the SASL password.

      ```
      secretutil.cmd -enc PASSWORD>OUTPUT FILE NAME 2.txt
      ```

      where *`PASSWORD`* is the password specified by the person who implemented the server side components when they ran the *init-crypto* command and *`OUTPUT FILE NAME 2.txt`* is the name of the file that contains the encrypted password.

   The output files contain the secret keys used in the *edge_agents.conf* file.
3. Perform the following steps to update the *edge_agents.conf* file with the secret keys that you created in the previous step.
   a) Open the *edge_agents.conf* file with a text editor.
   b) Update the value for the key `sasl.username`. It is the encrypted value in the username output text file.

   ```
   sasl.username=[encrypted username]
   ```

   c) Update the value for the key `sasl.password`. It is encrypted value in the password output text file.

   ```
   sasl.password=[encrypted password]
   ```
4. Save the *edge_agents.conf* file.

5. For Kafka TLS certificates that are issued from an untrusted CA only, perform the following steps to install the Kafka Truststore.
   a) Obtain the Kafka Broker root CA certificate from the person who implemented the CWP cluster. The file is stored in the CWP deployment package, in the *cwp/certs* folder.
   b) If required, rename the Kafka Broker root CA cert to *KafkaCA.pem*.
   c) Backup the *KafkaCA.pem* certificate in the Edge Agents installation directory.
   d) Replace the default *KafkaCA.pem* file in the Edge Agents installation package directory with the new *KafkaCA.pem* certificate file that you obtained from the implementation engineer.
6. Open the *edge_agents.conf*, and ensure that the value defined in the `sasl.ca.path` key is *C:\Nymi\Edge_Agents\certs\KafkaCA.pem*.
7. Edit the following configuration parameters in the *edge_agents.conf* file.

   Producer Specific Properties:

   - `bootstrap.servers`, which defines a list of host and port pairs of Kafka brokers.

   NES Specific Properties:

   - `nes.url`, which specifies the NES URL.
   - `agent.url`, which specifies the Nymi Agent URL. When you do not specifiy a value, EA will pick up the local Nymi Agent URL.
8. Save the *edge_agents.conf* file.

   Note: Ensure the *edge_agents.conf* file is configured prior to installing *edge_agents.msi*. This configuration file can then be copied to different machines being installed with EA.
9. Run the installer file *edgeagents-service-x64-*`version`*.msi*.
   The Edge Agents application is installed without any user interactions in the *C:\Nymi\Edge_Agents* folder and the .Nymi Edge Agents service appears with a Running status in Windows Services.

   Note: The section *Edge Agents Log Files* provides information about Edge Agents log files.

Encrypting the Key File with EFS

The `Edge Agents` service runs under the Network Service account. Nymi recommends that you encrypt the *decrypt.key* file with EFS on each host after you install the `Edge Agents` application.

Nymi provides you with a script to perform the encryption.

1. Run Power Shell as an administrator.

2. Enable the ability to run a script file in one of the following ways:
   - By setting the execution policy to ***RemoteSigned***.
     a. From Power Shell, type `get-executionpolicy`.

        The output displays the current execution policy setting.
     b. Type `set-executionpolicy RemoteSigned`, and when prompted, type `Y`

        The execution policy is set to RemoteSigned.

     MSDN provides more information about how to modify execution policy settings.
   - By unblocking the script file.
     a. From Power Shell, type `unblock-file c:\Nymi\Edge_Agents\tools\encrypt-withEFS.ps1`

     MSDN" provides more information about how to unblock script files.
3. Change to the *c:\Nymi\Edge_Agents\tools* directory.
4. Type *.\encrypt-withEFS.ps1 -file ..\conf\certs\decrypt.key*
5. Optional, use the **`set-executionpolicy`** command to set the execution policy to the original value.

Health Attestation and Temperature Alerts

CWP 1.2 introduces the Health Attestation and Temperature Alert features. The update scripts automatically cofigure the Kubernetes environment to support this new feature.

Before you run the update script, create the Active Directory group for Health and Safety users that require administrative access to the Health Check Application and create new tables for the features in the SQL database.

Creating AD Group for Health Check Application Access

Create a new Active Directory group with a list of Health and Safety users and groups that require administrative access to the Health Check Application

When creating the Active Directory group:

1. For the **`Group Type`**, select **`Security`**
2. For the **`Group Scope`**, select the option according to your IT policy.

Creating Tables for Health Attestation and Temperature Alerts

If your environment uses the Health Attestation and Temperature Alerts features, run the Nymi-provided SQL scripts to create the appropriate tables.

- The user that creates the database must have db_owner or db_ddladmin privilege to the `CWP Database`.
- Ensure that the extracted CWP deployment package is in a central location.

Perform the following steps from a computer that has SSMS installed and access to the SQL Server with the `CWP Database`.

1. Open SSMS and connect to the SQL server.
2. Navigate to the *CWP_12_deployment_folder**cwp**cwpweb**config* folder in the shared location.
3. Copy the *mssql.sql* file to local directory.
4. Navigate to the *CWP_12_deployment_folder**cwp**kafka-connect**config* folder in the shared location.
5. Copy the *temp-mssql.sql* file to a local directory.
6. In the local directory, double-click on the *mssql.sql* file.
   A new query window appears.
7. Click the **Execute** button.
   The script creates the dbo.NymiAttestationInfo and dbo.JwtStore tables.
8. In the local directory, double-click on the *temp-mssql.sql* file.
   A new query window appears.
9. Click the **Execute** button.
   The script creates the dbo.NymiTempInfo table.

Update the CWP environment

Nymi provides you with scripts to update the CWP environment, including environment variables and services.

In CWP 1.2, some environment variable names have changed. The script renames the CWP 1.1 environment variable names to the new environment variable names. The following table provides a list of the CWP 1.1 environment variables names and the corresponding new variable name.

\

Note: When you update to CWP 1.2, existing contact events remain unchanged in the Contact Tracing Dashboard.

CWP 1.1	CWP 1.2	Purpose
NES_SAM_ACCOUNT_USER_DOMAIN	*CORP_LDAP_DOMAIN*	Specifies FDQN of the Active Directory domain in which NES resides. For example: =qa-lab.local

CWP 1.1	CWP 1.2	Purpose
NES_SAM_ACCOUNT_USERNAME	*CORP_LDAP_USER*	Specifies the service account, which is an Active Directory domain account that is a member of the NES administrator group. For example: =adeed
NES_SAM_ACCOUNT_PASSWORD	*CORP_LDAP_PASSWORD*	Specifies password of the service account. Do not type a value for this variable. The *init-crypto* command will encrypt the password and update the file with the appropriate value.

Updating the CWP in Kubernetes Environment

Nymi provides customers with an update package that updates the configuration and installs new services.

- Ensure that you copied the extracted CWP update package to the Kubernetes Administration Terminal.

  Note: The folder in which you extracted the CWP 1.2 update package is referred to as the CWP 1.2 deployment folder (*CWP_12_deployment_folder*).
- Ensure that you know the path to the CWP 1.1.x deployment folder, referred to as the *master deployment folder*. For example */cwpall*.

Perform the following steps from the Kubernetes Administration Terminal, in a **bash** terminal.

1. Change to the *CWP_12_deployment_folder*/*deploy/kube/updates* folder, and then type `./update prod` *CWP_master_folder*.
   The update script revises the environment variables in the *master deployment folder* and creates a backup of the original deployment configuration files in *.cwp*.cwp folder in the home folder (~/) of the current user. The backup file/folder format is: *datetime-cwp-1.2-update*. Where *datetime* is in the format `time yyyymmdd.HHMM`. For example 10:35AM October 11 2021 will be: 20211011.1030.

2. Edit the *`master deployment folder`/deploy/kube/.env* file, and then specify values for the following environment variables:

Option	Description
`CORP_LDAP_USERDN`	Specifies the DN of the ***`CORP_LDAP_USER`*** that is used by LDAP. The format should consist of the name of an object and the LDAP designator. For example: `="CN=adeed,CN=Users,DC=qa-lab,DC=local"`
`CORP_LDAP_ATTESTATION_ADMIN_GROUPS`	Specifies the name of the Health and Safety AD group. Use commas to separate multiple group names. **Note:** Users in this group have administrator access to the Health Check Application.

3. Edit the *`master deployment folder`/deploy/kube/.prod-env* file, and then modify values for the following environment variables to match your environment, as appropriate:

Option	Description
`KAFKA_CONNECT_CPU_REQUESTS`	Specifies the amount of CPU initially requested by Kafka Connect in milliCPUs. The default value is `100m` .
`KAFKA_CONNECT_MEMORY_REQUESTS`	Specifies the amount of memory initially requested by Kafka Connect in mebibytes. The default value is `2564Mi` .
`KAFKA_CONNECT_CPU_LIMITS`	Specifies the maximum amount of CPU that Kafka Connect can request in milliCPUs. The default value is `1000m` .

Option	Description
KAFKA_CONNECT_MEMORY_LIMITS	Specifies the maximum amount of memory that the Kafka Connect can request in mebibytes. The default value is `512Mi` .

4. Change to the*`master deployment folder`*/*deploy/kube* folder, and then type `./cwp prod update`
   The new CWP services are deployed.

Creating the Authentication Table

Perform the following steps to create the authentication table in the CWP database.

- Ensure that the extracted CWP deployment package is in a central location.

Perform the following steps on a machine that has SSMS installed and has access to the NES database server.

1. Navigate to the *`CWP_12_deployment_folder`**kafka-connect**config* folder in the shared location, and then copy the *auth-mssql.sql* file to a local directory.
2. Open SSMS, and then login to the SQL Server.
3. From the **`File`** menu, select **`Open > File...`**, navigate to folder that contains the *auth-mssql.sql* script file that you downloaded, and then click **`Open`**.
   The script opens in the query window.
4. On the menu bar, click **`Execute`**.

   The script creates the SQL table for authentication events with the username and password that you specified previously.

   The script creates the and dbo.AuthInfo table.
5. Close SSMS.

Backup and Restore the Kubernetes Environment

You can use Velero (Apache 2 License) to backup and restore the whole cluster for disaster recovery purposes in a self-managed or managed Kubernetes cluster.

Note: Currently the installation script for Velero supports backups to an AWS S3 bucket. The installer requires full IAM and S3 privileges, with the ability to install and edit user policies, and create new users.

1. Peform the following steps to install the AWS CLI.
   a) Create an AWS API access key for your AWS account, if an AWS API access key does not already exist..
   b) Open a **bash** terminal.
   c) Change to the */CWP_12_deployment_folder/deploy/kube/init/aws/client* directory

      Note: This directory may be under either the *cwp* or *cwpall* folder.
   d) Type the `./install-aws-cli` script, and then provide the API access key and ID when prompted.
2. Perform the following steps to install Velero.
   a) Change to the *CWP_12_deployment_folder/deploy/kube/init* directory.
   b) Type `./install-velero -b <S3-Bucket-Name> [-h [n] | -d [n]] [-I <awsaccess-key-id>] [-k <aws-access-key>] [-u <velero user>]`

      Where:

      - *--s3-bucket* or *-b*: name of the S3 bucket to store backups. Default name is `cwp-backup`.
      - *--aws-region* or *-r*: name of the AWS region. Default region is *us-east-2*.
      - *--hourly* or *-h*: backup every *n* hours. This option cannot be used with *-daily* or *-d* option.
      - *--daily* or *-d*: backup every *n* days. This option cannot be used with *-hourly* or *-h* option.
      - *--user* or *-u*: AWS user account. Default is `velero`. A new Velero user is created if a Velero user does not already exist.
      - *--access-key-id* or *-i*: AWS access key ID associated with the Velero backup account. The script will create one if none is specified. An account can only have 2 access keys at any time.
      - *--access-key* or *-k*: AWS access key associated with the Velero backup account (with access-key-id specified above). The script will create one if none is specified.

      The script creates the S3 bucket,the AWS user account, installs Velero CLI on the user computer, and installs Velero Snapshot Controller in the Kubernetes cluster. This script will also schedule a backup every "n" hours or days.

      For example:

      - To install Velero with an S3 bucket called BUCKETNAME for a user without an AWS account (no access key ID and no access key), and no Velero account, with backup every 12 hours, type: `./install-velero -b BUCKETNAME -h 12`
      - To install Velero with an S3 bucket called BUCKETNAME for a user with a known access key ID, and a known access key. There will also be a Velero account, and backup every 2 days. `./install-`

```
velero --s3-bucket BUCKETNAME --daily 2 -i <AWS ACCESS KEY ID>
-k <AWS ACCESS KEY> --user <VELERO USER>
```

3. (Optional) To review the backups, type `velero get backups`.

   Backups of the Kubernetes cluster are stored in the S3 bucket name specified in the above steps. They contain timestamps that you can use to identify potential restore points for the cluster.

Restoring Kubernetes from a Backup

Use Velro to Restore a Kubernetes cluster from a backup to the same cluster or to a new one, after or disaster or after a change. Ensure that Velero and git is installed on the machine.

1. Open a **`bash`** terminal.
2. To check the backups for the restore name and timestamp, type `velero get backups`.
3. To restore the backup, type `--velero restore create <RESTORE NAME> --from-backup <S3 BUCKET NAME>-backup-<TIMESTAMP>`

   Where,

   `S3 BUCKET NAME` is the name of the bucket. Obtained when creating a bucket during the installation of Velero.

   `RESTORE NAME` is the name of the restore object. (Optional) Enter a name for this variable. If no name is specified, then a default name is created, "`<S3 BUCKET NAME>-<TIMESTAMP>`".

   `TIMESTAMP` is the timestamp of the latest backup. Obtained by looking at backup logs.

   This creates a restore object named <RESTORE NAME> from the backup bucket <S3 BUCKET NAME>.

   For example:`--velero restore create restorename --from-backup BUCKETNAME-backup-20200729154634`
4. To check the restore status, type `velero restore logs <RESTORE NAME>`

Restoring Kubernetes from a Disaster on the Same Kubernetes Cluster

Perform the following steps after a disaster, to restore the Kubernetes cluster from a backup.

Note: You can restore from disaster to another Kubernetes cluster. To do this, deploy a new Kubernetes cluster with Velero and use the same S3 Bucket and Velero credential file.

1. To change the backup storage location to read-only, type `kubectl patch backupstoragelocation default --namespace <AWS USER ACCOUNT> --type merge --patch '{"spec":{"accessMode":"ReadOnly"}}`

   If the *`AWS USER ACCOUNT`* was not specified during the installation of Velero, the default namespace is "`velero`".

   Note: The backup storage name can be retrieved using

   ```
   velero backup-location get -o yaml
   ```
2. Open a **`bash`** terminal.

3. To check the backups for the restore name and timestamp, type `velero get backups`.
4. To restore the backup, type `--velero restore create <RESTORE NAME> --from-backup <S3 BUCKET NAME>-backup-<TIMESTAMP>`

   Where,

   S3 BUCKET NAME is the name of the bucket. Obtained when creating a bucket during the installation of Velero.

   RESTORE NAME is the name of the restore object. (Optional) Enter a name for this variable. If no name is specified, then a default name is created, "`<S3 BUCKET NAME>-<TIMESTAMP>`".

   TIMESTAMP is the timestamp of the latest backup. Obtained by looking at backup logs.

   This creates a restore object named <RESTORE NAME> from the backup bucket <S3 BUCKET NAME>.

   For example:`--velero restore create restorename --from-backup BUCKETNAME-backup-20200729154634`
5. To check the restore status, type `velero restore logs <RESTORE NAME>`
6. To change the backup storage location back to read-write, type `kubectl patch backupstoragelocation default --namespace velero --type merge --patch '{"spec":{"accessMode":"ReadWrite"}}'`

Uninstalling the Edge Agents

1. Open the **Task Manager** and go to the **Processes** tab. Right-click *javaw.exe* and click **End Task**.
2. From the desktop, go to **Start > Settings > Apps > Apps and Features**.

   Note: You may also go to **Start > Control Panel > Programs > Programs and Features**.
3. Click Edge Agents and select **Uninstall**.

Troubleshooting

This section provides information about how to enable logging and how to troubleshoot common issues.

Configuring CWP Logging

The *.prod-env* contains variables that allow you to adjust the logging levels for CWP components, the location of the logs files, and the name of the log files.

To troubleshoot deployment issues, it might be necessary to modify the level at which CWP components log information. Each variable supports the following log level values:

- info
- warn
- error
- debug

Table 10: Configuration Parameters for Logging

Environment Variable	Description
KAFKA_LOG_LEVEL	Specifies the default logging level for Kafka. The default value is `info`.
CT_LOG_LEVEL	Specifies the CT Processor default log level. The default value is `info`.
CT_HTTP_LOG_LEVEL	Specifies the CT Processor HTTP client log level. The default value is `debug`.
CT_KAFKA_LOG_LEVEL	Specifies the CT Processor kafka log level. The default value is `info`.

Environment Variable	Description
CT_KAFKA_CLIENT_LOG_LEVEL	Specifies the CT Processor kafka client log level.The default value is debug .
CT_KAFKA_STREAM_LOG_LEVEL	Specifies the CT Processor kafka stream log level.The default value is info .
CT_KAFKA_CONSUMER_LOG_LEVEL	Specifies the CT Processor kafka consumer log level. The default value is info .
CT_CONSUMER_LOG_LEVEL	Specifies the CT consumer log level. The default value is debug .
CT_STREAMER_LOG_LEVEL	Specifies the CT streamer log level. The default value is debug .
CT_LOG_FOLDER	Specifies the location of the CT Processor log folder. Nymi recommends that you do not change the default value logs .
CT_LOG_FILE	Specifies the name of the CT Processor log file that is stored in the*CT_LOG_FOLDER*. The default filename is =CTlog .

Environment Variable	Description
CT_CONSUMER_LOG_FILE	Specifies the name of the CT Processor consumer log file that is stored in the *CT_LOG_FOLDER*. The default filename is consumer .
CT_STREAMER_LOG_FILE	Specifies the name of the CT Processor streamer log file that is stored in *CT_LOG_FOLDER*. The default filename is =streamer .

Determining the status of pods in the Kubernetes environment

You can display information and messages related to Kubernetes pods to stdout.

Perform the following commands on the Kubernetes Adminstration Terminal.

1. Change to the *CWP_12_deployment_folder**kube/deploy* folder.

2. To get a list of pods in the and the state of each pod, type `kubectl get pods -A`.
   The command displays a list that contains the name of each pod and the status. When a pod starts successfully the status is 1/1. If the pod cannot start, the value of the numerator for the ready count is less than the value of the denominator.

   The following figure provides an example of the output where two kafka-connect pods in the cw-qa environment have not started.

```
test@uat-01-24:/mnt/c/cwpall/deploy/kube$ kubectl get pods -A
NAMESPACE              NAME                                      READY   STATUS    RESTARTS   AGE
cwp-qa                 broker-0                                  1/1     Running   0          3m36s
cwp-qa                 broker-1                                  1/1     Running   0          3m1s
cwp-qa                 broker-2                                  1/1     Running   0          2m28s
cwp-qa                 ctapi-7d9b9f4c47-5g95g                    1/1     Running   0          2m53s
cwp-qa                 ctprocessor-747dcbbb49-7zktw              2/2     Running   1          2m15s
cwp-qa                 ctprocessor-747dcbbb49-gl2lv              2/2     Running   1          2m14s
cwp-qa                 ctprocessor-747dcbbb49-s2tzn              2/2     Running   1          2m14s
cwp-qa                 cwpweb-544c58d5f9-8nhlr                   1/1     Running   0          93s
cwp-qa                 kafka-connect-74df5888d-4qqxf             0/1     Pending   0          53s
cwp-qa                 kafka-connect-74df5888d-fp7gv             1/1     Running   0          53s
cwp-qa                 kafka-connect-74df5888d-qgxfh             0/1     Pending   0          53s
cwp-qa                 zookeeper-0                               1/1     Running   0          4m17s
cwp-qa                 zookeeper-1                               1/1     Running   0          4m17s
cwp-qa                 zookeeper-2                               1/1     Running   0          4m17s
kube-system            aws-node-5nk5n                            1/1     Running   0          153d
kube-system            aws-node-gk5fq                            1/1     Running   0          153d
kube-system            aws-node-ms5sr                            1/1     Running   0          153d
kube-system            cluster-autoscaler-78bbc94f4c-l6qf4       1/1     Running   0          153d
kube-system            coredns-56b458df85-gg968                  1/1     Running   0          153d
kube-system            coredns-56b458df85-nnjcm                  1/1     Running   0          153d
kube-system            ebs-csi-controller-d46b8f9d4-82jm8        6/6     Running   0          153d
kube-system            ebs-csi-controller-d46b8f9d4-c8z8j        6/6     Running   8          153d
kube-system            ebs-csi-node-7c96z                        3/3     Running   0          153d
kube-system            ebs-csi-node-kg2qj                        3/3     Running   0          153d
kube-system            ebs-csi-node-vk5mh                        3/3     Running   0          153d
kube-system            ebs-snapshot-controller-0                 1/1     Running   0          153d
kube-system            kube-proxy-ljpxb                          1/1     Running   0          153d
kube-system            kube-proxy-s2bgd                          1/1     Running   0          153d
kube-system            kube-proxy-sm9xw                          1/1     Running   0          153d
kube-system            metrics-server-56c59cf9ff-wjlwk           1/1     Running   0          153d
```

   Figure 5: Output of kubectl get pods Command

3. To generate the log files for a specific node, type `k logs -n` *namespace pod_name*.
   For example, to view log files for a pod named kafka-connect-74df5885d-4qqxf, type `k logs -n prod kafka-connect-74df5885d-4qqxf`

Error writing to file: C:\Nymi\ctca\ctca.bat

This error message appears during the installation of the Edge Agents (EA). It appears because EA was developed on Oracle JDK and not open JDK.

Cause

This error message appears when the incorrect version of Java is installed. The correct Java should be Oracle Java 8. To review the version of Java, go to the **`Registry Editor`** and review the *CurrentVersion* in *Computer\HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Runtime Environment*.

Resolution

To troubleshoot this issue, close the EA service and reinstall EA.

1. Open the **`Task Manager`** and kill *javaw.exe*.
2. Install Oracle JRE 8 (or Oracle JDK 8 - download from Oracle here).
3. Reinstall EA. Refer to Installing and Running the Contact Tracing Collection Agent.

Error: Unable to access jarfile ../CTCA-cwp_<ver>.jar

This error message appears while logging into EA.

Cause

This error message appears when the machine with EA is not domain joined, or there are not sufficient trusts in place with the AD domain (the domain with the NES instance). The exception can be seen in the *ctca.log* under *%AppData%\Roaming\Nymi\ctca\logs* directory. This indicates an incorrect URL in the *ctca.properties* file.

Note: The EA can be run manually in collection mode. Do this by executing the *ctca.bat* file manually from the command prompt.

Resolution

To troubleshoot this issue, edit the *ctca.properties* file for the EA.

If the machine running EA is not domain joined:

1. Install EA on a domain joined machine.
2. Configure EA to use basic authentication (based on AD username and password) with the NES instance. Ensure that network connectivity exists.
3. The `nes.username` and `nes.password` in the *ctca.properties* file must be encrypted.
   a. Run the *secretutil.cmd* command for the NES username and the password. The output file contains the encrypted username and password.

```
secretutil.cmd -enc [NES USERNAME]>[OUTPUT FILE NAME].txt
```

```
secretutil.cmd -enc [NES PASSWORD]>[OUTPUT FILE NAME].txt
```

   b. Open the newly created output files. They contain the encrypted keys for the username and password.
   c. Copy the encrypted NES usernames and passwords to the *ctca.properties* file. Update the following variables:

      `nes.username`

      `nes.password`

If the URLs in the *ctca.properties* is incorrect:

1. Open the *ctca.properties* file in *C:\Nymi\ctca\conf*.
2. Edit the `nes_url` and `agent_url`.

3. Ensure the trust store location `ssl.truststore.location` is correct and the truststore certificate *Kafka-tls-ca.jks* is renamed to `kafka.client.truststore`
4. Save *ctca.properties*, then run the *ctca<ver>.msi* to reinstall EA.

Note: The fall back logging directory is *C:\Nymi\ctca*. A directory named *ctca* will be generated if EA is unable to find the local *%Appdata%* path for the current user.

Database connection error while the CT consumer is writing to the database

Cause

This is an error that occurs when there is a disconnect between the system environment (env file) and the CT consumer. This occurs when the CT consumer fails to write to the database.

Resolution

To troubleshoot this issue, you will update the CT processor's configuration.

1. Go to *C:\Nymi\cwp\ctprocessor\config* and open the *context.xml* file with a text editor.
2. Comment-out the following line:

```
<property name="url" value="jdbc:sqlserver://
#{systemEnvironment['CT_DB_HOST']}:#{systemEnvironment['CT_DB_PORT']}" />
```

3. Comment-in the following line:

```
<property name="url" value="jdbc:sqlserver://#{systemEnvironment['CT_DB_HOST']}\
\#{systemEnvironment['CT_DB_INSTANCE']}" />
```

   The new section of code should look like the following (comments excluded):

```
<bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSource" destroy-method="close">
      <property name="driverClassName" value="com.microsoft.sqlserver.jdbc.SQLServerDriver" />
      <property name="url" value="jdbc:sqlserver://#{systemEnvironment['CT_DB_HOST']}\
\#{systemEnvironment['CT_DB_INSTANCE']}" />
      <property name="username" value="#{systemEnvironment['CT_DB_USERNAME']}" />
      <property name="password" value="#{systemEnvironment['CT_DB_PASSWORD']}" />
      <property name="defaultCatalog" value="#{systemEnvironment['CT_DB_CATALOG']}" />
      <property name="defaultSchema" value="#{systemEnvironment['CT_DB_SCHEMA']}" />
      <property name="initialSize" value="4"></property>
      <property name="maxTotal" value="32"></property>
      <property name="maxIdle" value="8"></property>
</bean>
```

4. Re-launch the entire cluster.

Edge Agents Log Files

The location and log files generated for `Edge Agents` differ in a local and remote configuration.

When you install `Edge Agents` locally on a user terminal, the *edge_agents.log file* appears in the *C:\Nymi\Edge_Agents\Logs* directory.

When you install the `Edge Agents` on an RDP or Citrix sessions host, the following log files are created:

- *C:\Nymi\Edge_Agents\Logs\edge_agents.log* file - Stores information about the `Edge Agents` service.
- *C:\Nymi\Edge_Agents\Logs\edge_agents`IPAv4ddress`.log* file - Stores information about the `Edge Agents` process that is started for a user session, where`IPAv4ddress` is the IP address of the host that connects to the RDP/Citrix session host. Each user session will have a separate log file.
- *%appdata%\Nymi\Edge_Agents\Logs\EdgeAgentsLauncher.log* - Stores information about the EdgeAgentsLauncher process that is started for a user session.

  Note: %appdata%

  applies to the *AppData\Roaming* directory of the user that starts the session.

Waiting for Cloudformation stack: eksctl-cwp-nodegroup-cwp-linux-workers to be deleted ...

This error appears when you run the *create.cluster* command to create the EKS cluster.

Cause

This is an error that occurs when you attempt to create a second cluster in the same region as an existing cluster with the same cluster name.

Resolution

To resolve this issue, perform the following steps on the Kubernetes Administration Terminal.

1. Edit the to *`CWP_12_deployment_folder`/deploy/kube/init/aws/env* file with a text editor.
2. Specify a new value for the *CLUSTER_NAME* variable.
3. Save the *env* file.
4. Run the *create.cluster* command.

Events Do Not Appear in the Contact Tracing Dashboard

When you log into the `Contact Tracing Dashboard` to view contact events, the events do not appear.

Cause

This issue can occur when `Edge Agents` cannot connect to the Kubernetes cluster because the *KafkaCA.pem* file is not valid. For example, an self-signed certificate was installed on the `Edge Agents` user terminal but the certificate on the server is not self-signed.

The following error messages appear in the *%AppData%\Nymi\Edge_Agents\Logs\edge_agents.log* file:

```
sasl_ssl://server_name/bootstrap: SSL handshake failed
routines:ssl3_get_server_certificate:certificate verify failed:
broker certificate could not be verified, verify that ssl.ca.location
is correctly configured or root CA certificates are installed (add
broker's CA certificate to the Windows Root certificate store) (after
256ms in state SSL_HANDSHAKE, 30 identical error(s) suppressed) .
```

Resolution

1. Obtain a valid *KafkaCA.pem* file from the person who implemented the CWP cluster.
2. Replace the *C:\Nymi\Edge_Agents\certs\KafkaCA.pem* with the correct certificate file.
3. Start the `Edge Agents` process.

Appendix 1 - Scripts

The following section provides information about the Nymi-provided CWP deployment scripts.

Environment Variables for Bare Metal Deployments

The `deploy/kube/init/bare/env` file contains the environment variables that are used during the Kubernetes deployment.

Important variables for bare-metal installation include:

- `K8S_VERSION`, which defines the Kubernetes version to install
- `CLUSTER_NAME`, which defines the name of the Kubernetes cluster

Details of the create-cluster Script

The **`create-cluster`** starts additional scripts during different stage of the deployment:

init-kube

The script that installs kubernetes and is invoked with **`sudo`**. The script supports the following options:

- `-c` - Installs a Kubernetes master node. When this option is excluded, the script installs a Kubernetes worker.
- `-d` - Installs and uses Docker for the container runtime instead of the default container runtime `containerd`.
- `-x` - Disables the Control Plane Node isolation mode. When you use this option, you cannot run pods on the control plane nodes

create-node

This script creates the initial Control Plane Master node in a new kubernetes cluster or join nodes to an existing Kubernetes cluster. It needs to be invoked with **`sudo`** and can take the following command line options:

- `-c` - Initialize kubernetes cluster on the initial master node
- `-s` - Joins a secondary master to a kubernetes cluster
- `-w` - Joins a worker node to a kubernetes cluster
- `-evAPI_server_fqdn` - Specifies the FQDN of the control-plane API Server endpoint. The default value is ***`kube-api-server`*** with IP address that is mapped to the first Control Plane Master node.

- `-t kubeadm_token` is cluster token for a node to join the cluster> This is required for secondary master nodes and worker nodes
- `-h hash` is cluster key hash for a node to join the cluster. This is required for secondary master nodes and worker nodes

Environment Variables For AWS

The *`deploy/kube/init/aws/env`* file contains the environment variables that are used for the installation.

The important variables for AWS include:

- *`CLUSTER_NAME`* - Name of the Kubernetes cluster.
- *`AWS_REGION`* - AWS region on which to install the EKS cluster.
- *`INSTANCE_TYPE`* - AWS EC2 instance type to use for EKS worker nodes.
- *`MIN_WORKER_NODES`*: - Auto-scaling parameter that defines minimal number of worker nodes in the cluster.
- *`DESIRED_WORKER_NODES`* - Auto-scaling parameter that defines the number of worker nodes in the cluster.
- *`MAX_WORKER_NODES`* - Auto-scaling parameter that defines the maximal number of worker nodes in the cluster
- *`ROOT_VOLUME_SIZE`* - Disk size of each node in Gigabyte

Published in Canada.
Nymi Inc.
Toronto, Ontario
www.nymi.com