Smart Distancing and Contact Tracing - Installation and Configuration Guide

Nymi Connected Worker Platform

v1.0

2021-05-03

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
01	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 NES Deployment Guide**

  This document provides an overview of the components and 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

Hardware and Software Requirements

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

NES

- Microsoft SQL Server Management Studio

Contact Tracing Dashboard

- Javascript enabled browser, such as Microsoft Edge, Google Chrome, Safari, or Mozilla Firefox

Contact Tracing Collection Agent

- Windows 10 64 bit with Oracle Java SE Runtime 8 (included in JDK 1.8.x 64 bit)
- Windows Server 2016 and 2019, TLS 1.2
- Open SSL 1.1.1 and above

Kubernetes Cluster Deployment

On Premise Deployment:

- Operating System: Ubuntu 18.04 LTS with Oracle Java SE Runtime 8 (included in JDK 1.8.x 64 bit)
- Control plane node: 4-core CPU, 8 GB RAM, 64 GB SSD
- Worker node: 4-core CPU, 16 GB RAM, 512 GB SSD
- Control plane node and worker node (combined without control plan isolation): 4-core CPU, 32 GB RAM, 512 GB SSD (formatted using LVM)
- Backup storage: SAN, Network file share (ex. NFS), or cloud storage (ex. AWS S3)
- Java

Note: A machine running Ubuntu 20.04 LTS can serve as a control plane node or a worker node.

AWS Deployment:

- 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

Firewall Port Requirements

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

Table 2: Control Plane Nodes

Protocol	Direction	Port Range	Purpose	Used By	Note
TCP	Inbound	443	Kubernetes dashboard service	Cluster	
TCP	Inbound	6443	Kubernetes API server	Cluster	Overridable
TCP	Inbound	2379-2380	etcd server client Cluster	kube-apiserver, etcd	
TCP, UDP	Inbound	53	Core DNS	Cluster	
TCP	Inbound	179	BGP routing	used by Calico CNI	
TCP	Inbound	9500	Longhorn CSI	Cluster	for self managed Kubernetes 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

Protocol	Direction	Port Range	Purpose	Used By	Note
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	Pods	Pod to pod networking
TCP	Inbound	22	ssh	administrators	deployment and maintenance

Protocol	Direction	Port Range	Purpose	Used By	Note
TCP	Inbound	30090-30094	Kafka Broker	CTCA	internal access
TCP	Inbound	31443	CTAPI Web service	Client, LB	External access

Table 4: Load Balancer

Protocol	Direction	Port Range	Purpose	Used By	Note
TCP	Inbound	9092	Load balancer for Kafka Broker	CTCA	
TCP	Inbound	443	CTAPI Web service	CT Dashboard	

Deployment, Installation, and Configuration Overview

1. Update firewall ports if 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. Prepare the deployment environment for Kubernetes clusters. Refer to Preparing for Kubernetes Deployment.
4. Install bash.exe (Ubuntu). Refer to Installing the Bash Terminal on page 12.
5. Deploy a Kubernetes Cluster.
   - Bare metal Kubernetes Cluster. Refer to Deploy a Bare Metal Kubernetes Cluster on page 14.
   - AWS with EKS Kubernetes Cluster. Refer to Deploy A Kubernetes Cluster in AWS Using EKS on page 20.
6. Create the SQL database for Contact Tracing. Refer to Installing the Contact Tracing Database on page 13.
7. Prepare the Kubernetes environment for Contact Tracing services. This process involves obtaining certificates and editing environment variables. Refer to Customize the Kubernetes environment for SD/CT on page 24.
   a) Set up the service account password.
   b) Install TLS certificates for Kafka and CTAPI. Refer to Preparing Certificates to install SD/CT on page 24.
   c) Set up environment variables. Refer to ENV variables on page 26.
8. Encrypt passwords to Kubernetes components. Refer toEncrypting the Passwords for Kubernetes Components on page 32. Verify that the passwords in the env file are updated.
9. Launch Kubernetes (refer to Launching the Kubernetes Environment on page 32), then ensure the Kubernetes cluster is running.

   ```
   kubectl get pods -A
   ```
10. Install `Contact Tracing Collection Agent` (CTCA) on a domain-joined (NES) terminal. Refer to .Installing and Running the Contact Tracing Collection Agent on page 43. You will need to:
    a) Encrypt the SASL username and password.
    b) Update TLS certificates.
    c) Update the *ctca.properties* file to include the newly encrypted username and password.
    d) Install CTCA and configure the environment variables.

11. Enable SDCT components in Nymi Enterprise Server.

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

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

Kubernetes Deployment

You will deploy Smart Distancing and Contact Tracing in a Kubernetes cluster. There are two methods to deploy a Kubernetes cluster; bare-metal (on premise or server) and AWS (cloud). Both methods require a control plane and at least one worker node (a node is a virtual machine or physical computer).

A Kubernetes cluster is controlled and managed by its 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.

A Kubernetes cluster consists of worker nodes that run containerized applications and host pods (components of an application's workload). Each node will consist of a kubelet (ensures containers are running in a pod), kube-proxy (directs network traffic to and from pods), and a Container runtime (runs containers), and each node is managed by the control plane. Typically, there are several nodes in a cluster, and a pod typically has one container.

For high availability, the number of control plane master nodes required will depend 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 5: 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

Preparing for Deployment

This section provides details about how to prepare for the Kubernetes cluster configuration.

1. If there is a designated load balancer for the Control Plan API Service, configure the load balancer to load balance the API Service running, control plane nodes
2. Add the DNS entry for all the nodes, if DNS is not available, add the records to ***/etc/hosts*** on every nodes
3. Add the DNS entry for the Control Plan API Service's public FQDN, if DNS is not available, add the record to ***/etc/hosts*** on every nodes
4. Install openssh server on all the node and enable sshd
5. For bare-metal Debian, make sure sudo is installed and the installer account has been added to member of sudo group
6. To install using root account, enable root login through ssh by setting ***PermitRootLogin yes*** in ***/etc/ssh/sshd_config***
7. To use a non-root account for the installation, add the account into ***/etc/sudoers***, make sure the same account exist in all machines have have the the same password. To enable sudo without password, add ***user-name ALL=(ALL) NOPASSWD:ALL*** to the end of /etc/sudoers, or edit with ***sudo visudo***
8. Allocate the POD network CIDR for the kubernetes cluster, make sure it does not overlap with any existing network segments
9. On the worker nodes, prepare the data storage volume:

CSI	Preparation	Note
Longhorn	formatted using LVM, mount to ***/var/lib/longhorn***	RWX is supported with nfs provisioner
AWS EBS CSI driver		

10. For unattended installation:
    - Create an ssh key using ***ssh-keygen*** on the initial master node where the Kubernetes cluster installation will be run
    - run ***ssh-copy-id user@other-node*** to use the key to authenticate *user* on *other node*, accept the signature when prompted
11. Copy the deployment package to the designated initial master node, and extract to the installer's home folder, there should be ***deploy*** and ***cwp*** folders.
12. Determine the number of nodes needed. These instructions cover the installation of 3 nodes. To add more, refer to Adding a Node on page 35.

Installing the Bash Terminal

Note: You must install, at minimum, Oracle Java SE Runtime 8 to run the commands described in this guide. Oracle JDK 8 includes JRE so you do not have to download both separately.

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 and Features`
3. On the left pane, click **`Turn Windows Features On or Off`**. A window appears with a list of Windows features.

4. Select the checkboxes for **Virtual Machine Platform** and **Windows Subsystem for Linux**, then click **OK** to confirm your choice. Windows will search for the required files and install them. Restart the terminal to apply the changes.
5. Obtain **Ubuntu** from the **Microsoft Store**. A window appears where you can search for Windows applications.
   a) Go to the **Start** menu and type Microsoft Store. Click **Microsoft Store**.
   b) Search for and **Get** Ubuntu. You will be presented with multiple options for **Ubuntu**. **Get** the app without the version number.

      Note: The **Ubuntu** app without the version number contains the most recent version. You may choose an earlier version if it is more applicable for your system.
   c) Click **Install**. You will be prompted to sign in. You may skip this by closing the pop-up window. The **Ubuntu** app will download in the background.
   d) When complete, click **Launch**. A **Ubuntu** windows appears and the installation will complete.
   e) Create a username and password for the UNIX user account.
6. Obtain **Windows Terminal** from the **Microsoft Store**. Repeat the steps above, but search for Windows Terminal instead.
7. To open bash, go to the **Start** menu and type Bash.

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

Installing the Contact Tracing Database

This is required for the CTAPI to connect to the SQL database. NES is installed with SQL Server. Use and install Microsoft SQL Server Management Studio (SSMS) to connect to the SQL Database.

1. Download the script *ct-mssql.sql* from the engineer.
2. Open SSMS.
3. Login to the NES SQL instance.
4. In the **Object Explorer** right-click Databases and click New Database.
5. Name the database ContactTracing. Then click **OK**.

   This is used in the environment configuration file, *env*, under *CT_DB_CATALOG*.
6. In the **Object Explorer**, go to **Security > Logins**. Right-click **Logins** and click **New Login**. A **Login - New** window appears. Go to the **General** page.
   a) Specify a login name. This will be used in the *env* file as *CT_DB_USERNAME*.
   b) Click **SQL Server authentication** and enter a password.

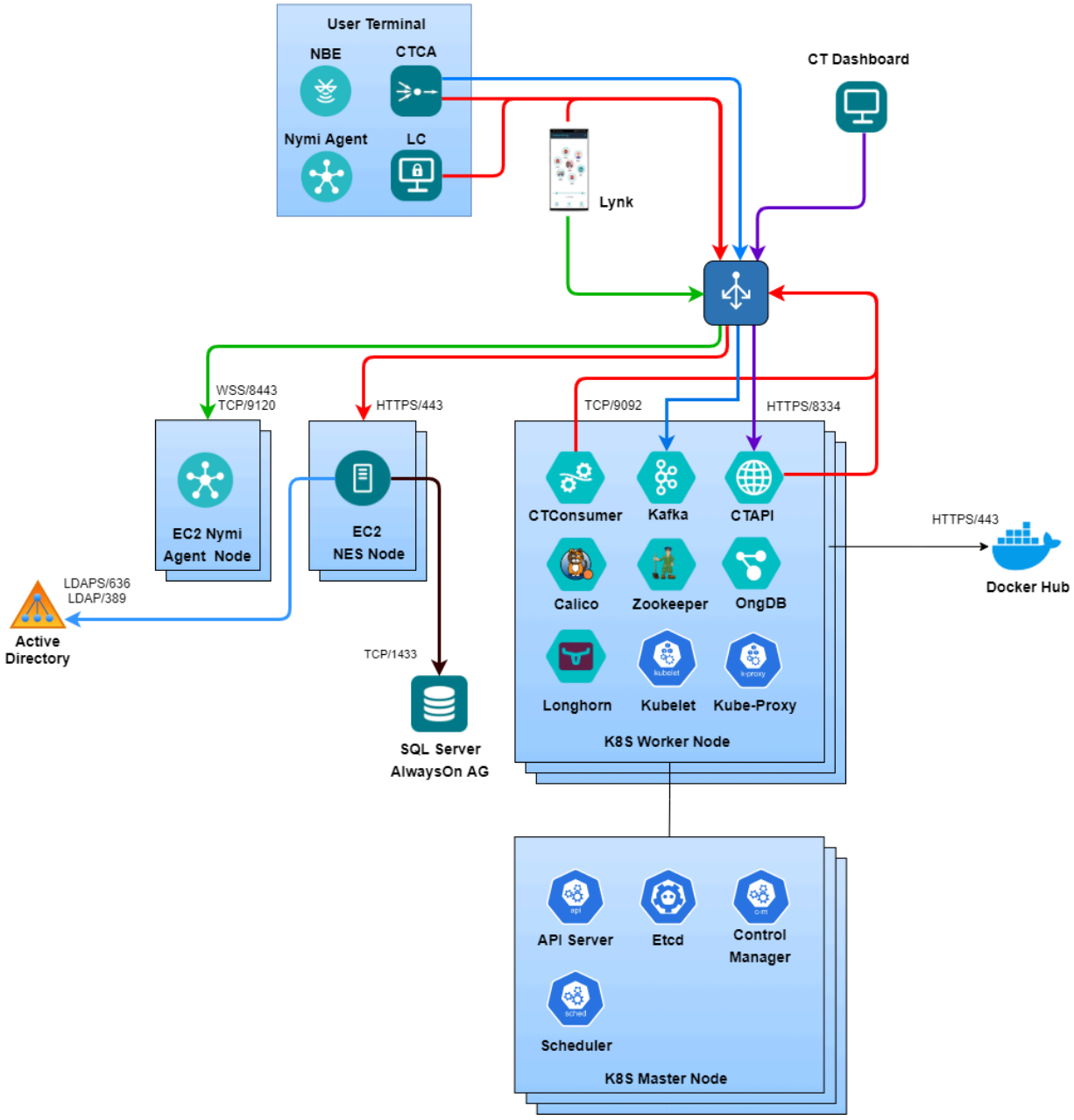
      This will be specified in the *init-crypto* file (refer to Encrypting the Passwords for Kubernetes Components on page 32).

7. In the **Login - New** window, go to the **User Mapping** page.
   You are presented with a list of the databases in SQL.
   a) Check the **ContactTracing** database. This was the database created in step 4.
   b) In the **Database role membership for ContactTracing** section at the bottom of the window, check **db_owner** and **public**.
   c) Click **OK**.
8. In SSMS, click **File > Open > File...** and navigate to the script file (step 1). Click **Open**. The script opens in the query window.
9. Click **Query > Execute**. This will run the script.
   The SQL database for Contact Tracing is created with a user and password.

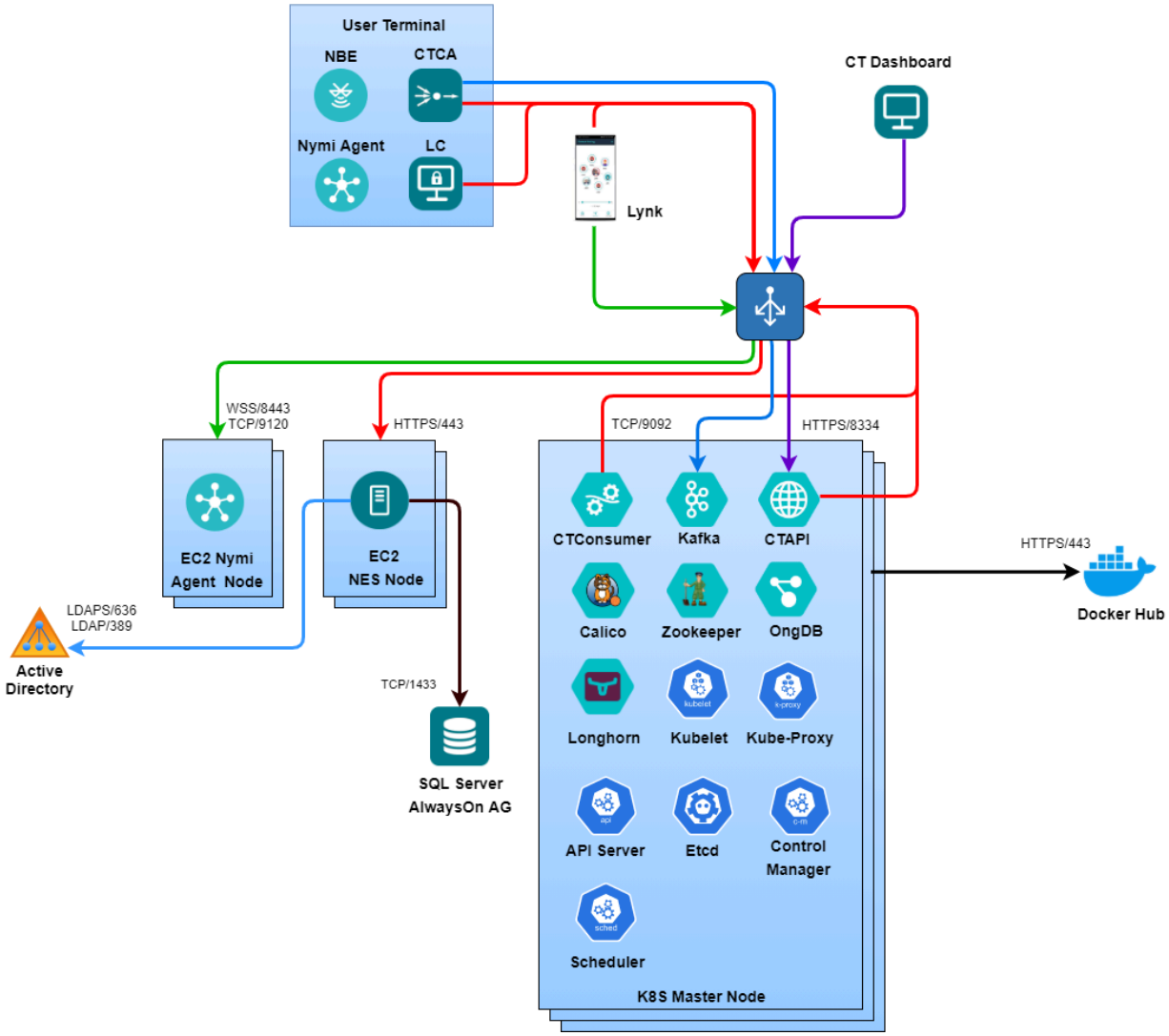
Deploy a Bare Metal Kubernetes Cluster

The deployment script will deployed Kubernetes clusters with stacked Etcd services in which Etcd servicer instances will co-reside with the master nodes. This reduce the trouble and expense of deploying a separated Etcd cluster which will require 3 additional machines.

The CWP Kubernetes deployment architecture for an isolated Kubernetes cluster environment is depicted in the following diagram:


User Terminal
NBE
CTCA
Nymi Agent
LC
CT Dashboard
Lynk
WSS/8443
TCP/9120
HTTPS/443
TCP/9092
HTTPS/8334
EC2 Nymi
Agent Node
EC2
NES Node
CTConsumer
Kafka
CTAPI
HTTPS/443
Docker Hub
LDAPS/636
LDAP/389
Active
Directory
Calico
Zookeeper
OngDB
TCP/1433
Longhorn
Kubelet
Kube-Proxy
SQL Server
AlwaysOn AG
K8S Worker Node
API Server
Etcd
Control
Manager
Scheduler
K8S Master Node

The deployment architecture for a non-isolated Kubernetes cluster environment consisting of only control plane masters where the master nodes are allowed to have pods scheduled on them is shown in the diagram below:


User Terminal
NBE
CTCA
Nymi Agent
LC
Lynk
CT Dashboard
WSS/8443
TCP/9120
HTTPS/443
TCP/9092
HTTPS/8334
EC2 Nymi
Agent Node
EC2
NES Node
CTConsumer
Kafka
CTAPI
HTTPS/443
Docker Hub
Calico
Zookeeper
OngDB
LDAPS/636
LDAP/389
Active
Directory
TCP/1433
Longhorn
Kubelet
Kube-Proxy
SQL Server
AlwaysOn AG
API Server
Etcd
Control
Manager
Scheduler
K8S Master Node

Configuration Variables

The `env` stores the environment variables used for the installation.

There are two environment, *env*, files - one for bare metal installations and one for AWS. The file location for the environment file is located in the CWP deployment package. The *env* file defines the environment variables for the deployment of Kubernetes.

- For bare-metal:*deploy/kube/init/bare/*
- For AWS: *deploy/kube/init/aws/*

The important variables for bare-metal includes:

- `K8S_VERSION`: the Kubernetes version to install
- `CLUSTER_NAME`: name of the Kubernetes cluster

Installing Kubernetes Client (Bare Metal)

Perform the following steps to install the Kubernetes client.

1. Open a bash terminal.
2. Change to *deploy/kube/client* folder.
3. Run:

```
sudo ./init-client -m user@master-node
```

Where:

- *`master-node`* is the address of the initial master node
- *`user`* is a user on the node.

Creating the Kubernetes Cluster

Perform the following actions on the designated initial master node:

1. Open a bash terminal
2. Change to the *deploy/kube/init/bare* folder.
3. Type

```
./create-cluster -c  master-node -w worker-node -x -n pod-network-cidr service-network-cidr -e api-server-fqdn -a admin-account
```

where:

`master-node`	adds a controller plane master node, specify multiple -c options for multiple secondary masters
`worker-node`	adds a worker node , specify multiple -w options for multiple secondary masters
`pod-network-cidr`	is the pod network CIDR. The default is 172.198.0.0/18.

service-network-cidr	is the service network CIDR. The default is 10.198.0.0/16.
api-server-fqdn	is the API Server's endpoint FQDN. Default to *kube-api-server* with IP mapped to the that of the first control plan master node
admin-account	is the Kubernetes administrator account to create, and you can specify any valid account name. The script adds the to the kubernetes cluster to enable the management of the cluster. The default value is *kube-admin*
-x	disables control plane node isolation mode, this will allow pods to run on control plane nodes

For example, the following command deploys a Kubernetes cluster with 3 master nodes that have the control plane node isolation turned off:

```
./create-cluster -c 192.168.66.188 -c 192.168.66.190 -x -n
 172.192.128.0/18 -e kube.nymi.com
```

The **./create-cluster** script invokes **init-kube** to install Docker and Kubernetes, then calls **create-node** to create the kubernetes cluster. The machine that runs the **create-cluster** is the first control plan master node. Then other master nodes and worker nodes will be added to the cluster. You can specify multiple secondary master nodes and worker nodes by specifying multiple *-c* and *-w* arguments respectively.

Load Balancing

When you deploy the CWP cluster on-premise, the load balancer does not provide Kubernetes integration. Set the service type to *NodePort*, and set the *nodePort* parameter to a designated port on the Kubernetes (worker) nodes that is open for external access.

Additionally, set the *externalTrafficPolicy* of the service needs to *Local* so that the service will not direct incoming traffic to other nodes, therefore preventing double-hop and preserving the source IP address. On the external load balancer, the respect server should be configured to use the nodes' IP addresses and the designated node port as the backend servers.

For layer 7 traffic, in alternative to using an external load balancer is to use an Ingress controller. However, ingress controller will not be used for simplicity, but can be added as needed.

Details of the create-cluster Script

create-cluster used the following scripts during different stage of the deployment:

>Install Kubernetes - init-kube

The script installs kubernetes. It needs to be invoked with **sudo** and can take the following command line options:

- `-c` installa a Kubernetes master node, if this option is not given, a Kubernetes worker will be installed
- `-d` installs and uses Docker for the container runtime, by default `containerd` is used for container runtime

- `-x` disables control plane node isolation mode, which allows pods to run on the control plane nodes

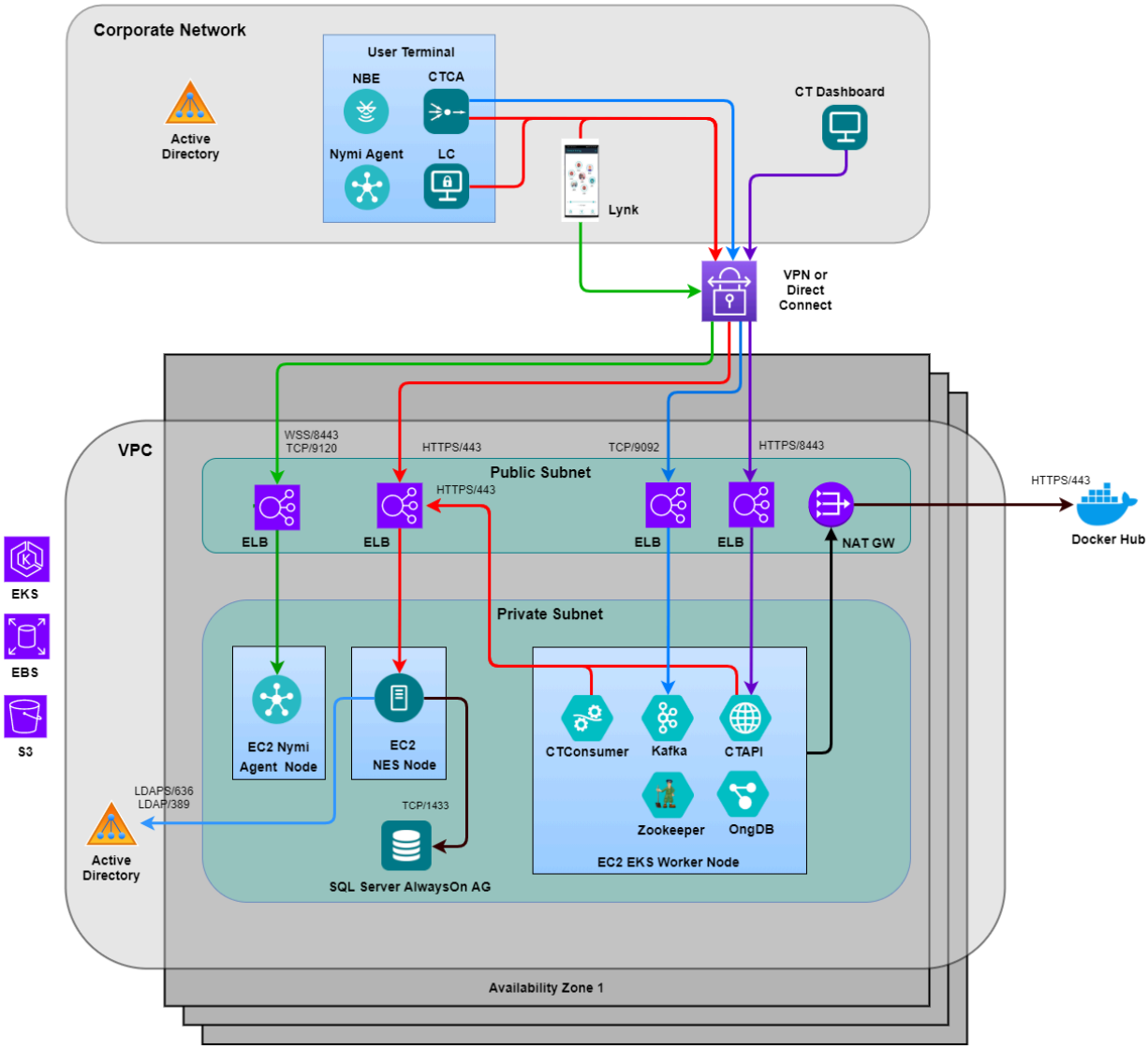
Create Initial Control Plane Master or a Node - create-node

This script creates 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 control-plane API Server endpoint's FQDN. The default value is *`kube-api-server`* with IP that is mapped to the first control plan 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

Deploy A Kubernetes Cluster in AWS Using EKS

The CWP Kubernetes cluster deployment architecture for AWS EKS is depicted in the following diagram:


Corporate Network
Active Directory
User Terminal
NBE
CTCA
Nymi Agent
LC
Lynk
CT Dashboard
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
CTConsumer
Kafka
CTAPI
Zookeeper
OngDB
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.

Configuration Variables

the *env* store the environment variables used for the installation.

There are two environment, *env*, files - one for bare metal installations and one for AWS. The file location for the environment file is located in the CWP package. The *env* file defines the environment variables for the deployment of Kubernetes.

- For bare-metal:*deploy/kube/init/bare/*
- For AWS: *deploy/kube/init/aws/*

The important variables for AWS includes:

- *CLUSTER_NAME* is the name of the Kubernetes cluster
- *AWS_REGION* is the AWS region where the EKS cluster is to be installed
- *INSTANCE_TYPE* is the AWS EC2 instance type to use for EKS worker nodes
- *MIN_WORKER_NODES*: an auto-scaling parameter for the minimal number of worker nodes in the cluster
- *DESIRED_WORKER_NODES* is an auto-scaling parameter for the desired number of worker nodes in the cluster
- *MAX_WORKER_NODES* is an auto-scaling parameter for the maximal number of worker nodes in the cluster
- *ROOT_VOLUME_SIZE* is the disk size of each node in Gigabyte

Creating a VPC

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 as follows:

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 EKS worker nodes. NAT Gateway is required to allow nodes to access internet	Not for internet facing

Running The VPC Creation Script

Perform a new VPC deployment by using the **`deploy/kube/init/aws/create-vpc`** script.

Perform the following steps on the designated initial master node.

1. Open a PowerShell or bash terminal.
2. Change to *deploy/kube/init/aws* folder.

```
cd deploy/kube/init/aws
```

3. Enter

```
./create-vpc -region <aws-region> [-public] [-private] [-cidr
 <network_CIDR>]/>
```

where:

- *`-region`* is the AWS region 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`* is 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
```

The subnets' network CIDRs will be distributed according to the size of the VPC's CIDR blocks. Their size will be 1/16 of that of the VPC's.

Creating a EKS Cluster

Create a new EKS cluster by using the **`deploy/kube/init/aws/create-cluster`** script:

Perform the following steps on the designated initial master node.

1. Open a bash terminal
2. Change to *deploy/kube/init/aws* folder.

3. 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`*
   - *`-admin`*, *`-a kubernetes_admin`* is the Kubernetes administrator account to create, and can be any valid account name. The command adds the account to the kubernetes cluster to enable managing the cluster. The default value is *`kube-admin`*.

   For example, the following command creates an EKS cluster with a VPC that has both public and private subnets: `./create-cluster -private`

   while

   `./create-cluster -public`

   will create a VPC with only public subnets.

Installing Kubernetes Client (AWS)

1. Open a bash terminal.
2. Change to *deploy/kube/client* folder.
3. Run:

```
./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

Load Balancing

AWS EKS will automatically create a Elastic Load Balancing (ELB) for a service whose type is *LoadBalancer*.

Network traffic is load balanced using AWS Network Load Balancer at layer 4 of the OSI model.

For layer 7 traffic, in alternative to using an external load balancer is to use an Ingress controller. However, ingress controller will not be used for simplicity, but can be added as needed.

Customize the Kubernetes environment for SD/CT

Preparing Certificates to install SD/CT

Perform the following steps to prepare for the launch of the environment.

1. Extract the deployment package to the deployment folder.

2. Copy the CTAPI TLS certificates (PKCS12) into *cwp/certs* of the unzipped deployment folder. Rename the certificate to `tls.pfx`.

   Note: The deployment script will extract values for `tls.crt` and `tls.key` from the certificate.
3. Copy the Kafka TLS certificate (PKCS12) into *cwp/certs* of the unzipped deployment folder. Rename the Kafka certificate to `kafka-tls.pfx`.
4. Navigate to *../deploy/kube* and edit the environment variables specified in the `.env` file. If applicable, edit the `.prod-env` or `.dev-env` files accordingly. The variables defined here provide values to the configuration maps used by CWP components.

Data Persistence

After you launch the `Connected Worker Platform` platform , all the data folders under the host(s) need to be kept beyond the termination of the respective containers.

These folders include:

- *kafka* under *runtime/kafka/data*.
- *Zookeeper* under *runtime/zookeeper/data*
- *Redis* under *runtime/redis/data*

Setting the Service Account Password

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.

You require the TLS certificate password.

Refer to Encrypting the Passwords for Kubernetes Components on page 32 to obtain the encrypted passwords. When you run the command `./init-crypto` you will be prompted to enter a TLS certificate password and service account password. The script will encrypt these passwords and store them in the *env* file.

Setting the Environment Variables

CWP is customized using *../CWPCONFIG.md* environment variables.

Navigate to the *deploy/kube* folder to find the environment variables `env` and `prod-env` and `dev-env`. These files are created upon deployment of the Kubernetes cluster and are used to define the configuration of CWP components.

To change the environment variables:

1. Open the deployment distribution package. This is the package where you run the scripts to install Kubernetes. Navigate to *cwpall/deploy/kube*.

2. Edit the *env*, *prod-env* or *dev-env* file, depending on your setup.

   - *env* defines the environment variables common to all environments
   - *prod-env* defines the environment variables for the production environment.
   - *dev-env* defines the environment variables for the development environment.

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

ENV variables

The variables in the *env* file must be updated for CWP components to be added. This information includes IP addresses of the domain, NES, and the Contact Tracing Dashboard (including their relevant credentials). Administrator login information for theNymi Enterprise Server and Contact Tracing Dashboard must be hardened via the `./init-crypto` script (refer to Encrypting the Passwords for Kubernetes Components on page 32).

Table 6: Configuration Parameters - General

Environment Variable	Description (with example)
CWP_COMPONENTS	List of CWP components to deploy. =(zookeeper broker ctapi ctprocessor)
NES_SAME_ACCOUNT_USER_DOMAIN	The Active Directory domain (ex. the admin account name to log into NES Administration Console). =qa-lab
NES_SAM_ACCOUNT_USERNAME	The Active Directory domain service account. =NESadminusername
NES_SAM_ACCOUNT_PASSWORD	The Active Directory domain service account password, should be set using init-crypto. Once the script is run, this field will be populated. =
CORP_LDAP_PROTOCOL	The LDAP protocol, should be `ldap` or `ldaps`. =ldap
CORP_LDAP_PORT	`389` for LDAP or `636` for LDAPS. This is needed for Kubernetes to connect to the domain. =389

Environment Variable	Description (with example)
CORP_LDAP_DC	The IP address of the AD domain controller. This can be found by logging into the terminal hosting the domain and using the **`Command Prompt`** to run `ipconfig all`. Alternatively, you can enter `nslookup`, then `type=all`, then enter the full domain name. =10.0.2.50
CORP_LDAP_BASEDN	The base DN for searching LDAP. The first DC should be the domain (ie. *NES_SAME_ACCOUNT_USER_DOMAIN*. The second DC should be `local`. ="DC=qa-lab,DC=local"
CORP_LDAP_API_GROUP	The LDAP group that has access to AWS. Change if necessary. ="Domain Users"
CORP_LDAP_USERDN	The AD account's distinguished name for the service account. Provide a name. This is the username that will be used for LDAP. This should consist of an object's name and its LDAP designator. DC=<username>, DC=local ="CN=adeed,CN=Users,DC=qa-lab,DC=local"
NES_USER_API_BASE_URL	The IP address of the NES server URL. This is the URL that is used to access the `NES Administrator Console` in a web browser, except with the IP address, and the NES instance. You may also find these values by running the NES installer on the NES server and selecting **`Review Settings`**. To find the IP address, log into the terminal used for NES and run `ipconfig all` in the **`Command Prompt`**. =https://10.0.4.167/nes_33/
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. This is not advised. =1

Table 7: Configuration Parameters - Kafka

Environment Variable	Description (with example)
KAFKA_BROKER_EXTERNAL_PORT	Kafka broker listener port for external client. Leave this value as-is unless necessary. =9092
KAFKA_BROKER_INTERNAL_PORT	Kafka broker listener port for internal client. Leave this value as-is unless necessary. =29092
KAFKA_BROKER_INTER_BROKER_PORT	Kafka broker listener port between broker instances. Leave this value as-is unless necessary. =9095
KAFKA_LISTENER_SECURITY_PROTOCOL_INTERNAL	Kafka broker listener protocol in the kubernetes cluster. =SSL
KAFKA_LISTENER_SECURITY_PROTOCOL_EXTERNAL	Kafka broker listener protocol outside the kubernetes cluster. =SASL_SSL
KAFKA_LISTENER_SECURITY_PROTOCOL_INTER_BROKER	Kafka broker listener protocol between the broker instances. =SSL
KAFKA_SASL_ENABLED_MECHANISMS	Kafka SASL enabled mechanisms. Leave this as-is. =PLAIN
KAFKA_SASL_ENABLED_MECHANISM_INTER_BROKER_PROTOCOL	Kafka SASL enabled mechanisms inter broker protocol. Leave this as-is. =PLAIN
KAFKA_TLS_CERTIFICATE	Kafka broker TLS certificate. This will be created when you run the init-crypto script. =kafka-tls.pfx

Environment Variable	Description (with example)
KAFKA_SSL_KEYSTORE_PASSWORD	Kafka broker TLS certificate keystore password, should be set using init-crypto. Once the init-crypto script is run, this field will be populated. =
KAFKA_SSL_TRUSTSTORE_TYPE	Kafka broker TLS certificate CA keystore type (JKS or PKCS12). Leave this as-is. =JKS
KAFKA_SSL_KEYSTORE_TYPE	Kafka broker TLS certificate keystore type (JKS or PKCS12). Leave this as-is. =JKS
KAFKA_SSL_KEYSTORE_LOCATION	Kafka TLS certificate keystore location. =/var/lib/certs/kafka-tls.jks
KAFKA_SSL_TRUSTSTORE_LOCATION	Kafka TLS certificate CA keystore location =/var/lib/certs/kafka-tls-ca.jks
KAFKA_SASL_ADMIN_PASSWORD	Kafka SASL admin password, should be set using init-crypto. Once the init-crypto script is run, this field will be populated. =

Table 8: Configuration Parameters - Logging (levels - info, warn, error, debug)

Environment Variable	Description (with example)
KAFKA_LOG_LEVEL	the kafka's default logging level. =info
CT_LOG_LEVEL	CT Processor default log level. =info
CT_HTTP_LOG_LEVEL	CT Processor HTTP client log level. =debug

Environment Variable	Description (with example)
CT_KAFKA_LOG_LEVEL	CT Processor kafka log level. =info
CT_KAFKA_CLIENT_LOG_LEVEL	CT Processor kafka client log level. =debug
CT_KAFKA_STREAM_LOG_LEVEL	CT Processor kafka stream log level. =info
CT_KAFKA_CONSUMER_LOG_LEVEL	CT Processor kafka consumer log level. =info
CT_CONSUMER_LOG_LEVEL	CT consumer log level. =debug
CT_STREAMER_LOG_LEVEL	CT streamer log level. =debug
CT_LOG_FOLDER	CT Processor log folder. Leave this as-is. =logs
CT_LOG_FILE	CT Processor log file. Specify the name of the log file. This will be stored in *CT_LOG_FOLDER*. =CTlog
CT_CONSUMER_LOG_FILE	CT Processor consumer log file. Specify the name of the log file. This will be stored in *CT_LOG_FOLDER*.This will be stored in *CT_LOG_FOLDER*. =consumer
CT_STREAMER_LOG_FILE	CT Processor streamer log file. Specify the name of the log file. This will be stored in *CT_LOG_FOLDER*. =streamer

Table 9: Configuration Parameters - CT Database

Environment Variable	Description (with example)
CT_DB_HOST	This is the IP address of the terminal hosting the SQL database. =10.0.4.102
CT_DB_PORT	This is the port for the SQL database. Update this as necessary. =1433
CT_DB_INSTANCE	This is the name of the instance where the SQL database is created. =SQLEXPRESS
CT_DB_CATALOG	This is the location where various schema and mappings are kept in an SQL environment. Here, the `ContactTracing` catalogue contains schema for the Contact Tracing. This value is the database name that was created when you created the Contact Tracing SQL database. =ContactTracing
CT_DB_SCHEMA	The SQL server schema for the CT database. If this is not particularly defined, use the default, `dbo`. =dbo
CT_DB_USERNAME	This is the SQL username used to log into the CT database. Specify a username. Running the =ct_sa
CT_DB_PASSWORD	This is the encrypted password for the CT database. Once the init-crypto script is run, this field will be populated. =
CT_JDBC_DRIVER	This is the driver that provides Java Database Connectivity a connection to the SQL Server (and Azure SQL database). Review the software requirements on Microsoft's website for more information. =com.microsoft.sqlserver.jdbc.SQLServerDriver

Encrypting the Passwords for Kubernetes Components

This section describes the instructions to harden the passwords for the Kubernetes components you will install.

1. Open a bash terminal.
2. Change directory to *../deploy/kube*.

   Note: This path may be under the *cwp* or *cwpall* folder.
3. Run the encryption script. You will be prompted to enter the passwords to several Kubernetes components. The script will encrypt these passwords and store them in the environment variable file, *env*, in *deploy/kube*. Refer to ENV variables on page 26 for more information on the environment variables.

   ```
   ./init-crypto
   ```

   The encrypted passwords include the following

   - *NES_SAM_ACCOUNT_PASSWORD* - the password used to log into the NES administration 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* - the Kafka password for the CTCA.

Launching the Kubernetes Environment

The instructions in this section enables you to launch the Kubernetes environment with 3 nodes. Instructions on adding more nodes are described in Adding a Node on page 35.

1. Ensure you have the deployment package saved on the computer. If not, contact Nymi. The deployment package contains files and scripts that will deploy Kubernetes and Connected Worker Platform components.
2. Open a bash terminal on any Kubernetes node.

3. Initialize the Bash setup for Kubernetes.
   a) Open the bash terminal and change the directory to the client folder in the deployment package.

```
cd deploy/kube/client
```

   b) Initialize the client.

   For bare-metal deployments, enter:

```
./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, enter:

```
./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

4. To launch Kubernetes environment:

   Change directory to...

```
cd deploy/kube
```

   Update the environment (with changes in the Kubernetes YAML files):

```
./cwp update prod update
```

   Launch the environment:

```
./cwp prod up
```

To shutdown the environment, type:

```
./cwp prod down
```

To update the environment after changing the Kubernetes YAML files, type:

```
./cwp update prod update
```

Maintain Kubernetes

The following section provides information and commands that you can run to maintain and make changes to the Kubernetes cluster.

Note: Ensure that you shut down Kubernetes before performing any tasks on the Kubernetes cluster.

To shut down the Kubernetes environment, change directory

```
cd deploy/kube
```

Enter the following:

```
./cwp prod down
```

Deleting an EKS Cluster

Perform the following steps to delete an EKS cluster in a VPC environment.

1. Open a PowerShell or bash terminal.
2. Change the directory

   cd deploy/kube/init/aws
3. Enter

   ```
   sudo ./delete-cluster
   ```

Deleting a VPC

Perform the following steps to delete the VPC environment.

Note: Deleting the environment will also delete the clusters in the environment.

1. Open a PowerShell or bash terminal.
2. Change the directory

   cd deploy/kube/init/aws
3. Enter

   ```
   sudo ./delete-vpc
   ```

Changing the Kubernetes Cluster Name

After the script creates the Kubernetes cluster, you can change the name of a Kubernetes cluster, if required.

Perform the following steps to review the current name of the Kubernetes cluster and then change the name.

1. To determine the current name of the cluster, enter

   ```
   kubectl get configmaps -n kube-system kubeadm-config -o yaml
   ```
2. To change the cluster name, enter

   ```
   kubectl edit configmaps -n kube-system kubeadm-config
   ```

Adding a Node

Open PowerShell or bash, and enter the following to add an additional node to the cluster.

```
kubeadm join <API_SERVER>:6443 --control-plane --discovery-token
 <TOKEN> --discovery-token-ca-cert-hash <HASH> --certificate-key
 <CERTIFICATE KEY>
```

Deleting a Node

Perform this task to remove a node from the Kubernetes Cluster.

1. Open PowerShell or bash.
2. Navigate to the *client* folder.

```
cd deploy/kube/client
```

3. Run the delete node script.

```
./delete-node <NODE NAME>
```

Draining a Node

Drain the node before shutting down the node. This allows the pods in the node to terminate properly and avoids damage to the cluster or persistent data.

1. Open PowerShell or bash.
2. Confirm the node to drain.

```
kubetcl get nodes
```

3. Drain the node.

```
kubectl drain <NODE NAME> --ignore-daemonsets --delete-emptydir-
data --timeout 180s || kubectl drain <NODE NAME> --ignore-
daemonsets --delete-emptydir-data --disable-eviction
```

4. Allow the change to propagate to the entire cluster. Approximately 5 minutes. The node can then be shutdown (refer to Shutting Down a Node on page 35).

Shutting Down a Node

Do not reboot the node or power it off without first draining the node and shutting it down though PowerShell or bash to avoid damage to the cluster or persistent data.

1. Open PowerShell or bash.
2. Confirm and drain the node. Refer to Draining a Node on page 35. Allow approximately 5 minutes for the change to propogate through the cluster.

3. Shutdown the node (different from powering the node off).

   Note: This may require you to log in remotely to the node using ssh.

   ```
   sudo shutdown now
   ```

   This will take the node to runlevel 1 (recovery mode). The node can now be rebooted.
4. (Optional) Reboot the node.

   Note: This may require you to log in remotely to the node using ssh.

   ```
   sudo reboot now
   ```

Deleting a Pod

There are instances where you need to delete a pod from a node, such as when debugging or scaling the node. Additionally, draining a node may resolve the issue of a pod stuck in the ContainerCreating state, however it could trigger failure in other working pods. In this case, it is advisable to delete the pod instead.

1. Open a PowerShell or bash terminal.
2. Confirm the node from which you will delete the pod, and the pods in the node.

   ```
   kubetcl get nodes
   ```

   ```
   kubetcl get pods -o wide | grep <NODE NAME>
   ```
3. Cordon the node to prevent new pods from being scheduled to the node.

   ```
   kubetcl cordon <NODE NAME>
   ```
4. Delete the pod.

   ```
   kubetcl delete pod <POD NAME>
   ```
5. Uncordon the node to allow scheduling on node again.

   ```
   kubetcl uncordon <NODE NAME>
   ```

Uncordoning a Node

Uncordon a node to allow scheduling on the node after performing maintenance.

1. Open PowerShell or bash.
2. Confirm the node.

   ```
   kubetcl get nodes
   ```
3. Uncordon the node.

   ```
   kubetcl uncordon <NODE NAME>
   ```

Backup and Restore the Kubernetes Environment

Velero (Apache 2 License) can be used for whole cluster backup and restore for disaster recovery purpose in a self-managed or managed Kubernetes cluster.

Note: Currently the installation script for Velero supports backup to an AWS S3 bucket. The installer needs to have full IAM and S3 privileges, capable of installing and editing user policies and creating new users. AWS CLI needs to be installed first.

1. Install the AWS CLI.
   a) Create an AWS API access key for your AWS account. If you already have an AWS API access key, ignore this step.
   b) Open a bash terminal.
   c) Change directory to the aws/client folder.

```
cd deploy/kube/init/aws/client
```

   Note: This directory may be under either the *cwp* or *cwpall* folder.
   d) Run the *install-aws-cli* script. You will be asked for the API access key and ID.

```
./install-aws-cli
```

2. Install Velero. Ensure IAM priviledges are enabled.
   a) Return to the *deploy/kube/init* directory.

```
cd ..
```

   b) Run the install-velero script. The script will create the S3 bucket, create the AWS user account, and install Velero CLI on the user computer, as well as install Velero Snapshot Controller in the Kubernetes cluster. This script will also schedule a backup every "n" hours or days.

```
./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.

   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, enter:

```
./install-velero -b BUCKETNAME -h 12
```

   Another example:

   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) Review the backups.

   Backups of the Kubernetes cluster are stored in the S3 bucket name specified in the above steps. They contain timestamps used to identify potential restore points for the cluster. View timestamps and names by running the following in the bash terminal.

```
velero get backups
```

Restoring Kubernetes from a Backup

Restore a Kubernetes cluster from a backup to the same cluster or to a new one using Velero. Ensure Velero and git is installed. Use the steps described below to restore a cluster after a disaster, or after a change.

Create a restore object from the S3 bucket.

1. a) Open a bash terminal.
   b) Check the backups for the restore name and timestamp.

```
velero get backups
```

   c) Run:

```
--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
```

   d) Check the restore status.

```
velero restore logs <RESTORE NAME>
```

2. To restore from disaster on the same Kubernetes Cluster:
   a)

Restoring Kubernetes from Disaster on the Same Kubernetes Cluster

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.

Upon a disaster, restore the Kubernetes cluster from a backup as follows:

1. Change the backup storage location to read-only.

```
kubectl patch backupstoragelocation default --namespace
 <AWS USER ACCOUNT> --type merge --patch '{"spec":
{"accessMode":"ReadOnly"}}
```

Note: 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. Create a restore from the backup.

   Follow the steps in Restoring Kubernetes from a Backup on page 38.

```
velero restore create <RESTORE NAME> --from-backup <S3 BUCKET
 NAME>-backup-<TIMESTAMP>
```

3. Change the backup storage location back to read-write.

```
kubectl patch backupstoragelocation default --namespace velero --
type merge --patch '{"spec":{"accessMode":"ReadWrite"}}'
```

Removing Velero

1. Open a bash terminal and go to the *deploy/kube/init* folder. The folder will be under *cwp* or *cwpall*.
2. run the delete-velero script.

```
./delete-velero
```

Monitor the Kubernetes Cluster

Prometheus (Apache 2) is a popular tool for monitoring a Kubernetes cluster. It is often used together with Grafana for querying and visualizing Prometheus metrics.

Installing Prometheus and Grafana

Note: In order to access Kubernetes dashboard and Grafana UI from the computer, you must first install the Kubernetes client.

1. Open a bash terminal.
2. Change directory to the *init* folder.

```
cd deploy/kube/init
```

   Note: The directory will be under *cwp/deploy/kube/init* or *cwpall 3/deploy/kube/init*.

3. Run the *install-prometheus* script. This will install Prometheus and Grafana. The installation will also set up port forwarding from localhost:8000 to the Prometheus-Grafana service.

```
./install-prometheus
```

4. Open a browser and enter `http://localhost:8080` to access Prometheus and Grafana .

   Username: `admin`

   Password: `prom-operator`

5. Set up access to Kubernetes Dashboard and Grafana UI on a client computer.
   a) Open a bash terminal.
   b) Change directory to the *init* folder.
   c) Run the *start-proxy* script to start the kubectl proxy and port forwarding to Grafana. Follow the prompts from the script for further instructions on accessing the Kubernetes Dashboard and Grafana UI.

```
./start-proxy
```

   d) To stop access to Kubernetes Dashboard and Grafana UI on a client computer, run the *stop-proxy* script. This will stop kubectl proxy and port forwarding to Grafana.

```
./stop-proxy
```

Accessing Grafana

Grafana UI is accessible in a browser. Upon using Grafana, you will be asked to configure the dashboard for specific metrics to view.

1. Ensure Kubernetes Dashboard is enabled. Refer to Installing Prometheus and Grafana on page 40.
2. Open a web browser and enter the following URL, depending on the deployment of Kubernetes.
   - For bare-metal (on premise) Kubernetes:

     `http://localhost:8000`
   - For AWS Kubernetes:

     `http://localhost:8800`

   Login using `admin` as the username.

   Login using `prom-operator` as the password. If `prom-operator` does not work, try `admin`.
3. (Upon first use) Configure the Grafana Dashboard by creating a new dashboard, or importing a dashboard template.

   To import a dashboard (recommended):

   a) Click the **+** icon in the side menu, then click **`Import`**.
   b) Enter one of the following dashboard IDs:
      - `1860`

        visualize all node data exported by Prometheus
      - `8588`

        visualize Kubernetes Deployment and StatefulSet metrics
   c) Click **`Load`**.
   d) Click **`Import`**.
   e) Repeat the process for the other ID. You will be able to view both dashboards.

Accessing Kubernetes Dashboard

Kubernetes Dashboard is accessible from a browser.

1. Ensure Kubernetes Dashboard is enabled. Refer to Installing Prometheus and Grafana on page 40.
2. Open a web browser and enter the following URL, depending on the deployment of Kubernetes.
   - For bare-metal (on premise) Kubernetes:

     ```
     http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/
     services/https:kubernetes-dashboard:/proxy
     ```
   - For AWS Kubernetes:

     ```
     http://localhost:8801/api/v1/namespaces/kubernetes-dashboard/
     services/https:kubernetes-dashboard:/proxy
     ```

Removing Prometheus and Grafana

1. Open a bash terminal.
2. Change directory to the *init* folder.

   ```
   cd deploy/kube/init
   ```

   Note: The directory will be under *cwp/deploy/kube/init* or *cwpall 3/deploy/kube/init*.
3. Run the *delete-prometheus* script. This will uninstall Prometheus and Grafana.

   ```
   ./delete-prometheus
   ```

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.

From the NES Administrator Console, the CWP Administrator can enable smart distancing reminders for their workplace. When smart distancing reminders are enabled, users with their Nymi Bands are notified when they are in close proximity with each other.

Smart Distancing and Contact Tracing Overview

The Contact Tracing Collection Agent (CTCA) is installed on a user terminal and establishes BLE communication with Nymi Bands via `Nymi Bluetooth Endpoint` and the Nymi Agent.

The CTCA retrieves contact tracing data from the Nymi Band and sends them to the Kafka processing system. The Kafka processing system and the CT Consumer service transforms contact tracing data for use in the Contact Tracing database.

The events in the Contact Tracing database are then displayed on the Contact Tracing Dashboard via Contact Tracing (CT) API.

Note: Contact tracing data that are processed through Kafka contain information from the Nymi Enterprise Server for contact tracing purposes. Select **`Smart Distancing and Contact Tracing`** in the NES Administrator Console to enable this.

Contact Tracing Collection Agent

The `Contact Tracing Collection Agent` (CTCA) operates in collection mode. The CTCA listens for BLE presence notifications from Nymi Bands and downloads contact tracing data from the Nymi Bands.

You can deploy CTCA in local and RDP/Citrix configurations:

- In a local configuration, install CTCA, and the `Nymi Runtime` application on user terminal.
- In a remote configuration, install CTCA on an RDP session host or Citrix server, the `Nymi Bluetooth Endpoint` application on a user terminal, and the `Nymi Agent` application on any server in the environment. The Nymi Connected Worker Platform Administration 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 and Running the Contact Tracing Collection Agent

CTCA is implemented in Oracle Java 8 (64 bit) and is only launched when a user logs into their user terminal (Windows 10, 64 bit).

Prerequisites:

- Install CTCA on a user terminal that is domain-joined to NES to serve as a collection point for contact tracing logs.
- Ensure that OpenSSL (version 1.1.1c and above) for Windows has been installed, and that the */bin* directory is included in the system path. OpenSSL is not a pre-requisite to run the CTCA software, but required to generate keys and encrypt secrets, as described below.
- For local configurations, ensure that `Nymi Runtime` is installed on the user terminal.
- For remote configurations, ensure that:
  - `Nymi Bluetooth Endpoint` is installed on the user terminal and ensure that the agent_url is configured in the *nbe.toml* file.
  - `Nymi Agent` is installed on a server in the environment.

1. Unzip the contents of the *CTCA.zip* file. The contents are listed below.

- The *ctca.msi* file installs the CTCA software. It uses the parameters detailed in the ctca.properties file.
- The *secretutil.cmd* file, is a Windows command utility that encrypts secrets.
- The *ctca.properties* file is used to configure the parameters of the CTCA installation. It will include keys generated from the PowerShell utility.

2. Generate the secret keys 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 to the CTCA folder.

```
cd C:\[location of the CTCA folder]
```

c) Initialize *secretutil.cmd*.

```
secretutil.cmd -init
```

d) Encrypt the sasl.jaas username and password, and provide names for the output files. These files will contain the secret keys used in the *ctca.properties* file. The username and password need to be supplied by the implementation engineer or administrator who installs the server side components.

For the username, enter the username and the filename that will hold the *`sasl.jaas.username`* key.

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

For the password, enter the password and the filename that will hold the *`sasl.jaas.password`* key.

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

Text files with the specified output filenames are created in the CTCA folder.

3. Update the *ctca.properties* file with the secret keys created in the previous step.
   a) Open the *ctca.properties* file with a text editor.
   b) Update the value for the key `sasl.jaas.username` to the encrypted value in the username output text file.

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

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

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

   d) Save the *ctca.properties* file.
4. Install the Kafka Truststore.
   a) Obtain the CTCA certificate truststore (*Kafka-tls-ca.jks*) from the implementation engineer. This should be in the CWP deployment package under *CWP/certs*. Rename this file to `kafka.client.truststore`.
   b) Backup the *kafka.client.truststore* certificate in the CTCA directory (step 1).
   c) Replace the default truststore certificate in the CTCA directory (step 1) with the new *kafka.client.truststore* certificate (obtained from the implementation engineer).
   d) Ensure that the name *kafka.client.truststore* is used for the file name key `ssl.truststore.location` in the *ctca.properties* file. The location value should be *C:\Nymi\ctca\certs\kafka.client.truststore*.
5. Edit the configuration parameters in the *ctca.properties* file.

   Producer Specific Properties:

   - `bootstrap.servers`: List of host and port pairs of Kafka brokers.
   - `topic`: The Kafka topic on which to publish events. Leave this as-is.
   - `log_file`: The name of the log file. The log file is located in *C:\Nymi\ctca\logs\[log_file].log*. Logs will contain information such as DEBUG, INFO, WARN, ERROR, and FATAL. The log files are saved on the user terminal with CTCA.

   Security (TLS) Specific Properties:

   - `security.protocol`: Specifies the security protocol used for communication. The default value is `SASL_SSL`.
   - `ssl.truststore.location`: Location of *truststore.jks* on *server/local*. Default value: *C:\Nymi\ctca\certs\kafka.client.truststore*.
   - `ssl.truststore.password`: The password for truststore.
   - `ssl.truststore.type`: The default is `jks`.

   NES Specific Properties:

   - `nes_url`: Specifies the NES URL.
   - `agent_url`: Specifies the Nymi Agent URL. If not specified, CTCA will pick up the local Nymi Agent URL.
6. Save the properties file.

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

7. Run the installer file, *ctca.msi*. The directory *C:\Nymi\ctca* is created after successful installation.

Uninstalling the Contact Tracing Collection Agent

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 **CTCA** and select **Uninstall**.

Contact Tracing Consumer

The Contact Tracing Consumer is deployed in a Kubernetes cluster as part of the Contact Tracing Service (CTS). The CT Consumer is composed of two docker containers that run as part of the same pod. At startup, CT Consumer connects to Kafka, the contact tracing relational database (MS SQL Server).

The Contact Tracing (CT) Consumer consumes the proximity events produced by the Contact Tracing Collection Agent, and determines if there has been a contact event (as per the configurable policy) for the pair of Nymi Bands.

If a contact event has been established, CT Consumer logs the contact event into the contact tracing relational database.

Information from contact tracing events are stored in a relational database (MS SQL Server). Information includes the source MAC address, remote MAC address, and the timestamp. Relationships between contact tracing events from different users are shown in the Contact Tracing Dashboard.

Contact Tracing API

The Contact Tracing API (CTAPI) is a standard Restful web API that allows applications to retrieve node information from the graph database for display and analysis on the Contact Tracing Dashboard. To learn more about the Contact Tracing Dashboard, refer to Contact Tracing Dashboard on page 48.

Enabling Smart Distancing and Contact Tracing

Enable Smart Distancing and Contact Tracing in your group policy through the NES Administrator Console.

1. Log in to the NES Administrator Console with an account that is an NES Administrator.
2. Click **Policies**.
3. Edit the active policy.
4. Select **Smart Distancing and Contact Tracing**.

Enabling Smart Distancing Reminders

Smart Distancing Reminders is only available if you enable **Smart Distancing and Contact Tracing** in the NES Administrator Console.

1. Log in to the NES Administrator Console with an account that is an NES Administrator.
2. Click **Policies**.
3. Edit the active policy.
4. Select **Smart Distancing and Contact Tracing**.
5. Enable or disable **Smart Distancing Reminders**. This option is only available if **Smart Distancing and Contact Tracing** is enabled.
   - Select the option to allow users to receive smart distancing reminders on their Nymi Bands. The Nymi Band will vibrate and display a reminder on the screen when Nymi Band users are in close proximity with each other for approximately 5 minutes.
   - Leave the option cleared to disable smart distancing reminders on Nymi Bands.

Contact Tracing Dashboard

The Contact Tracing Dashboard is used to visualize and analyze contact tracing data for employees enrolled in the Contact Tracing program.

Contact tracing events logged in each employee's Nymi Band are uploaded to the Contact Tracing Dashboard via the Contact Tracing Collection Agent (CTCA) installed on a user terminal. These 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.

Contact Tracing (CT) Services are used to process and store contact tracing data. Contact tracing data is transferred to CT Services through the Contact Tracing Collection Agent (CTCA).

Contact Tracing Dashboard Example

Figure 4: Example of the Contact Tracing Dashboard

Accessing the Contact Tracing Dashboard

Connect to the Contact Tracing server in a browser.

Open your web browser.

1. Open your web browser.
2. Go to `https://<host>/dashboard`

   The <host> is the Fully Qualified Domain Name (FQDN) of the Contact Tracing server. The host is derived from the DNS of he CTAPI in AWS.
3. Log in using the CT dashboard username and password. Alternatively, you can log in using credentials as an NES administrator.

Logging into the Contact Tracing Dashboard

Users who are granted access can enter their corporate credentials (username and password) to log into the Contact Tracing Dashboard.

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 Dashboard and analyzing the contact tracing data.

Enrolled Employees

Identifies the number of employees enrolled in the Social Distancing and Contact Tracing program. The number of enrolled employees is defined during the Contact Tracing Server installation process. This value will be fixed based on the number of users specified in the PoC project plan.

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 will drop to 98%.

Average and Maximum Contacts (%)

The average identifies the percentage of protected employees contacted by an individual expressed as an average across all employees registered in the program. The maximum identifies the highest percentage of users contacted by any single user. 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 contacted the most unique employees in the program. This report is based on data from the last 30 days.

Total Contacts by Day

Identifies The total number of contacts within the program per day. A contact is recorded when a Nymi Band user is closer than ~2 meters from another Nymi Band for approximately 15 minutes.

Most Contacted Employee Details

Identifies the total contacts and number of unique employees contacted for the employees with the most unique employees contacted. This report is based on data from the last 30 days.

Employee Contact Timeline

Timeline of all contact events for a specific employee within a date range. Search by username and date range (default date range is last 30 days). Search terms are case sensitive.

Troubleshooting

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

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