Event Driven Automation

Release 25.4

Connect Guide

3HE 21777 AAAB TQZZA
Edition: 1
April 2025

Table of contents

1 About this document...... 6

1.1 Precautionary and information messages...... 6

1.2 What's new...... 6

2 Introduction...... 8

3 Cloud Connect Core...... 10

3.1 Installation...... 10

3.2 Resources...... 10

3.2.1 Plugins...... 11

3.2.2 Heartbeats...... 12

3.2.3 Connect interfaces...... 12

3.3 Namespace support...... 12

3.4 Connect UI...... 12

3.5 Connect integration modes...... 13

3.5.1 CMS-managed integration mode...... 13

3.5.2 EDA-managed integration mode...... 13

3.6 Troubleshooting...... 14

4 OpenShift Connect plugin...... 15

4.1 Prerequisites...... 15

4.2 Architecture...... 16

4.3 Deployment...... 17

4.3.1 EDA Kubernetes preparation...... 17

4.3.1.1 Creating a service account and token...... 17

4.3.2 Fetching the EDA Connect OpenShift plugin Helm charts...... 18

4.3.3 Deploying the plugin in OpenShift...... 18

4.3.4 Deployment verification...... 20

4.4 Operational modes...... 21

4.4.1 Using the Transparent operational mode...... 21

4.4.2 Using the Connect Network Definition operational mode...... 21

4.4.3 Using the NAD Annotation operational mode...... 23

4.5 Audit...... 23

4.6 Troubleshooting...... 25

5 VMware vSphere plugin....26
5.1 Prerequisites....26
5.2 Architecture....26
5.3 Deployment....27
5.3.1 Plugin app deployment....27
5.3.2 Plugin deployment....28
5.4 Functionality....28
5.4.1 Startup....29
5.4.2 Event monitoring....29
5.4.3 Operational modes....29
5.5 Troubleshooting....30

6 OpenStack ML2 plugin....32
6.1 Architecture....32
6.2 Deployment....33
6.2.1 CBIS extension preparation....34
6.2.1.1 Creating a service account and token....34
6.2.1.2 Deploying the CBIS extension....35
6.2.1.3 Updating the bearer token....36
6.3 Neutron ML2 plugin configurations....37
6.3.1 Virtualization and network types....37
6.3.2 Networking models....38
6.3.3 Forwarding capabilities....38
6.3.4 Bonding....38
6.3.5 Trunking....39
6.3.6 Network VLAN segmentation and segregation....39
6.4 Usage and managed networks....40
6.4.1 OpenStack managed networks....40
6.4.2 EDA managed networks....41
6.4.3 Edge topology introspection....42

7 Connect alarms....43
7.1 Connect service alarms....43
7.2 Plugin alarms....43
7.2.1 OpenShift plugin alarms....43

7.2.2 VMware plugin alarms... 44

1 About this document

This Event Driven Automation (EDA) *Connect Guide* describes the EDA Connect solution (or "Connect"), which acts as a bridge between EDA and different cloud environments.

This document is intended for network technicians, administrators, operators, service providers, and others who use EDA.

Note: This manual covers the current release and may also contain some content that will be released in later maintenance loads. See the *Event Driven Automation Release Notes* for information about features supported in each load.

1.1 Precautionary and information messages

The following are information symbols used in the documentation.

DANGER: Danger warns that the described activity or situation may result in serious personal injury or death. An electric shock hazard could exist. Before you begin work on this equipment, be aware of hazards involving electrical circuitry, be familiar with networking environments, and implement accident prevention procedures.

WARNING: Warning indicates that the described activity or situation may, or will, cause equipment damage, serious performance problems, or loss of data.

Caution: Caution indicates that the described activity or situation may reduce your component or system performance.

Note: Note provides additional operational information.

Tip: Tip provides suggestions for use or best practices.

1.2 What's new

This section lists the changes that were made in this release.

Table 1: What's new in 25.4.1

Feature	Location
Adds Connect OpenStack ML2 plugin support including: • architecture • deployment	OpenStack ML2 plugin

Feature	Location
• configuration	
Updates supported versions for OpenShift Connect plugin	OpenShift Connect plugin
Updates OpenShift CND operational mode to include pre-provisioning to the Connect interface	Using the Connect Network Definition operational mode
Updates Audit section	Audit

2 Introduction

Overview

The EDA Cloud Connect solution (or "Connect") acts as a bridge between EDA and different cloud environments like Red Hat OpenShift, VMware vSphere and others.

Connect is aware of the different processes and workloads running on the servers that make up the cloud environment, while at the same time being aware of the fabric as configured on EDA itself.

This dual awareness enables Connect to configure the fabric dynamically based on workloads coming and going on the cloud platform. It does this by inspecting the cloud itself and learning the compute server, network interface and VLAN on which a specific workload is scheduled. By also learning the topology based on the LLDP information arriving in the fabric switches, it connects those two information sources.

Components

The Connect solution is built around a central service, called the Cloud Connect Core, and plugins for each supported cloud environment.

The Connect Core is responsible for managing the plugins and the relation between Connect interfaces (compute interfaces) and EDA interfaces (Fabric interfaces or Edge-Links). It keeps track of the LLDP information of EDA interfaces and correlates that back to the Connect interfaces created by plugins to identify the different physical interfaces of the computes of a cloud environment.

Connect plugins are responsible for tracking the state of compute nodes, their physical interfaces, the virtual networks created in the cloud environment and their correlation to the physical network interfaces. As applications create networks and virtual machines or containers, the plugins inform Connect Core of the changes needed to the fabric. Plugins also create or manage EDA `BridgeDomains` to make sure the correct subinterfaces are created for the application connectivity.

Plugins overview

Connect plugins are specifically made to inspect one type of cloud environment. While these plugins can be developed specifically targeting a custom cloud environment, Connect comes with three Nokia supported plugins:

- Connect OpenShift plugin
- Connect VMware plugin
- Connect OpenStack ML2 plugin (CBIS only)

Feature overview

Connect supports the following features:

- Creating Layer 2 EVPN Overlay Services on EDA.
- Automatically discovering the cloud compute resources and connectivity to the fabric using LLDP.
- Automatically resolving inconsistent states between Connect and the fabric by performing an audit between Connect and EDA.
- Using pre-existing LAGs in the fabric.
- Using the cloud management's standard network management tools to manage the fabric transparently.

- Using EVPN services that are managed by EDA. This is the case in which an operator provisions a service in EDA before making them available in the compute environment for use. This allows for more advanced use cases than the compute environment might support natively.

3 Cloud Connect Core

The Cloud Connect Core is deployed as part of the EDA deployment itself. It lives as an application in EDA and handles the configuration of EDA resources based on the needs of the compute environments, provided through the different plugins.

3.1 Installation

Cloud Connect is an application in the EDA app eco-system. It can be easily installed using the App Store UI.

Installation using Kubernetes API

If you prefer installing the Connect Core using the Kubernetes API, you can do so by creating the following Workflow resource:

```
apiVersion: core.eda.nokia.com/v1
kind: Workflow
metadata:
  name: connect-nokia
  namespace: eda-system
spec:
  input:
    app: connect
    catalog: eda-catalog-builtin-apps
    operation: install
    vendor: nokia
    version:
      type: semver
      value: v2.0.0
  type: app-installer
```

3.2 Resources

Connect is based on a pluggable architecture. The Cloud Connect core installation is a collection of controllers responsible for bridging the hypervisor world with the fabric world. It is the plugin that is responsible for introspecting the cloud environment.

The following Custom Resources are involved:

Table 2: Custom Resources

Custom Resources	Description
ConnectPlugin	The logical representation of the plugin, created for each plugin automatically when the plugin starts with valid credentials.

Custom Resources	Description
ConnectPluginActionable	An actionable is an action to be taken by the `ConnectPlugin`. It is used by the Core to tell the plugin to do something (for example: initiate an Audit).
ConnectPluginHeartbeat	The `ConnectPlugin` will continuously send heartbeats to the Cloud Connect service to report its status and alarms.
ConnectInterface	The logical representation of a hypervisor NIC and/or LAG. The labels on the `ConnectInterface` are used to label the EDA interface (leaf interface) correctly so that the correct subinterfaces are created.

3.2.1 Plugins

Plugins are a core component of the Event Driven Automation (EDA) Connect environment. In the Connect environment, a plugin represents the component that communicates with the external cloud services. The following plugins are supported by EDA, and are further documented in their respective sections:

- OpenShift Connect plugin
- VMware vSphere plugin
- OpenStack ML2 plugin

Plugins are automatically registered within the Connect service when they are deployed. Each is stored in the database with the following main properties:

Table 3: Plugin properties

Plugin property	Description
Name	A unique name based on the plugin type and compute environment it is connected to.
Plugin Type	The type of plugin, for example, VMware or Open Shift.
Heartbeat Interval	The interval, in seconds, between heartbeats that the plugin intends to use.
Supported actions	The different actions a plugin can support. These are actions the Core can request the plugin to do. For example, to trigger an Audit.

3.2.2 Heartbeats

When plugins register with the Connect core service, they can indicate that they support heartbeats. When a plugin supports heartbeats, the plugin is expected to send a heartbeat to the Connect core service at an interval of the configured value (or more frequently). If the Connect core does not receive a heartbeat from the plugin after two intervals, it raises an alarm in EDA to indicate that there could be an issue with the plugin.

3.2.3 Connect interfaces

Connect interfaces are managed by the plugins and represent the network interfaces of a compute node. When a plugin notices a new compute or new network interface on a compute node, it will create a Connect interface in EDA for Connect Core to monitor.

Connect Core uses the information from the Connect interface to determine the matching EDA interface. This is the interface on a leaf managed by EDA to which the interface on the compute is connected to, or potentially multiple interfaces in case of a LAG or Bond.

The plugin will label these Connect interfaces to indicate that Connect Core needs to make sure the matching leaf interfaces have a subinterface created in the corresponding overlay service (`Bridge Domain`).

This way, only those subinterfaces that are truly necessary are configured in the fabric. This limits configuration bloat and possible security risks.

3.3 Namespace support

The EDA Connect service supports multiple namespaces. Multiple namespaces are supported on the level of the plugin.

A plugin is a namespaced object, meaning that it must be created as part of a specific namespace. That plugin will only have access to the fabrics, services and interfaces of that namespace and cannot use objects from other namespaces.

This also means that a compute cluster can only belong to a single namespace, and cannot span multiple namespaces. This is to be expected as compute clusters belong to a single fabric, and a fabric is part of a single namespace.

3.4 Connect UI

The Connect UI can be found as part of the **System Administrator** section of the EDA UI, and allows for inspection of the different resources owned and managed by Connect. This Connect UI follows the same design as the regular EDA UI, where the left menu for Connect opens and displays the different resources available.

Note:
Do not create new resources manually, as this could interfere with the behavior of the plugins.

If you made changes manually, an audit will revert them. Changes should be made through the Cloud orchestration platform.

3.5 Connect integration modes

Integration modes define how plugins create resources in EDA for use by the applications in the compute environments.

Connect supports two integration modes:

- **CMS-managed mode**

  Networking concepts of the CMS are used to create new services in EDA.
- **EDA-managed mode**

  Network services are created in EDA and the networking concepts in the CMS are linked or associated with these pre-existing services.

Each of these modes can be used by the plugins. For the plugins provided by Nokia, both modes are supported, and you can combine them and switch between them as needed. For instance, you can use one integration mode for one application, while using the other for another application.

3.5.1 CMS-managed integration mode

In the Cloud Management mode, Connect creates an EDA `BridgeDomain` for each subnet that is created in the Cloud Management system. In this mode, the changes in the Cloud Management system are transparently reflected into EDA. The administrator of the Cloud Management system does not require any knowledge about how to use EDA.

3.5.2 EDA-managed integration mode

For more advanced use cases, a more complex EVPN Service (or set of services) may be needed. This can include features of these services that are supported by EDA, but not natively by the CMS. Examples are configuring complex routing or QoS policies, or using BGP PE/CE for route advertisement from the application into the network service.

In such cases, Nokia recommends using the EDA-managed integration mode, which instructs Connect to associate the subnets in the CMS with existing `BridgeDomain` in EDA, instead of creating new resources in EDA based on the cloud management networking.

In this mode, an administrator (or orchestration engine) with knowledge of EDA first creates the necessary resources in EDA directly. You can create more complex configurations than the cloud management system itself would be able to do. When creating the networking constructs in the Cloud Management system, you provide a set of unique identifiers referring to those pre-created networking constructs. This way, the Connect plugin and Connect service know not to create their own resources, but to use the pre-created items.

3.6 Troubleshooting

Missing `ConnectPlugin` for deployed plugin

If a plugin does not show up in the list of `ConnectPlugins`, it indicates a connection problem from the plugin toward the EDA Kubernetes cluster.

Verify the following information:

- Check the plugin logs for error messages.
- Verify the plugin's configuration, especially the Kubernetes information (location/url, certificates, user certificates and so on).

Application connectivity

An application missing connectivity can have multiple causes. The following are some of the most common causes:

- Check whether there are any alarms reported in EDA.
- Check whether there are any `TransactionResults` in FAILED state.
- Check whether `BridgeDomain` and `VLAN` are up, and `VLAN` is showing the expected number of UP subinterfaces.
- Check the `ConnectInterface` corresponding to the compute node NIC where you expect to see traffic.
  - If no `ConnectInterface` can be found, check the plugin. It is responsible for creating the `ConnectInterface`.

    Note: The Connect interface is located in the namespace of the Fabric.

  - If a `ConnectInterface` can be found, check the status.
    - If the status is blank the connect-interface-controller is not online.
    - If the status is Disconnected it cannot find an EDA interface to label.
- Check the EDA interface corresponding to the NIC on the SRL that is connected to the relevant compute node.
  - If it is not found: operator has to create the "downlink" interfaces
  - If it is found: check the status
    - If no members with the LLDP info are found, check the LLDP process on the compute node and on the SRL node.

4 OpenShift Connect plugin

EDA Cloud Connect integrates with OpenShift to provide fabric-level application networks for OpenShift pods and services. The Connect integration leverages the OpenShift Multus CNI solution to support managing the fabric directly from OpenShift and making the fabric dynamically respond to the networking needs of the application.

OpenShift provides the following advantages and capabilities:

- Direct integration into the network management workflow of OpenShift.
- Use of the common CNIs used by enterprise applications and CNFs like IPVLAN and SR-IOV.
- Automatic provisioning of the fabric based on where the application pods need the connectivity.
- Support for advanced workflows.

Supported versions

The following versions of OpenShift are supported:

- Red Hat OpenShift 4.16
- Red Hat OpenShift 4.18

4.1 Prerequisites

Before using the Connect OpenShift plugin, ensure the following prerequisites are met:

- Openshift cluster is up and running.
- NMState-Operator and Multus are installed on the OpenShift cluster.

  Note: For more information about the NMState Operator, see the Red Hat OpenShift documentation About the Kubernetes NMState Operator.

- EDA cluster is up and running.
- EDA Cloud Connect app is installed.
- You have access to the controller container image named `ghcr.io/nokia-eda/eda-connect-k8s-controller:3.0.0`.
- NMState Operator is configured to listen for LLDP TLVs. Create the following resource in your OpenShift cluster and make sure to include all interfaces in the list that are connected to leaf switches managed by EDA:

```
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: enable-receive-lldp
spec:
  desiredState:
    interfaces:
    - name: <interface-name>
```

```
      lldp:
        enabled: true
```

4.2 Architecture

The OpenShift plugin consists of a controller which monitors the following resources in OpenShift:

- The physical NIC configuration and correlation to `NetworkAttachmentDefinitions` through the NMState Operator.
- `NetworkAttachmentDefinitions` and their master interfaces.
- `ConnectNetworkDefinitions` for the configuration of Layer 2 and Layer 3 services configuration.

Operational modes

The OpenShift plugin supports three methods of associating `NetworkAttachmentDefinitions` (NADs) to EDA `BridgeDomains`, called Operational modes:

Table 4: Operational modes

Operational mode	Description
Transparent association	This association does not require any information from EDA in OpenShift. For every unique master interface defined in `NADs` in the OpenShift Cluster, the plugin will create a unique `BridgeDomain`. This association only supports OpenShift Managed Mode.
Annotation based association	In this association, annotations are added to the `NAD` that reference an existing EDA `Bridge Domain`. This association only supports EDA-managed mode.
`ConnectNetworkDefinition` association	The `ConnectNetworkDefinition` is a Custom Resource Definition that gets added to the OpenShift Cluster and is used to describe the relationship between the different services and `NetworkAttachmentDefinitions`, and how the services relate to each other. This association supports both OpenShift-managed and EDA-managed modes.

Note: It is possible to use a mix of these modes in a single OpenShift cluster, as well as switch and migrate between them.

Supported features

The following are some of the supported OpenShift features:

- CMS-managed integration mode

- EDA-managed integration mode
- Network Attachment Definition Transparent operational mode (CMS-managed only)
- Connect Network Definition operational mode (CMS-managed and EDA Managed)
- Network Attachment Definition Annotation operational mode (EDA-managed only)
- Optimally configure subinterfaces to minimize configuration and security footprint of network services
- LAG/LACP interfaces
- VLAN Trunking
- Audits

The following are supported CNIs:

- MACVLAN
- IPVLAN
- SRIOV
- Dynamic SRIOV

4.3 Deployment

This section describes how to prepare for the deployment of the OpenShift plugin and how to deploy the plugin.

4.3.1 EDA Kubernetes preparation

4.3.1.1 Creating a service account and token

Creating a service account

The EDA Connect OpenShift plugin uses a `ServiceAccount` in the EDA Kubernetes cluster to create the necessary resources in the EDA cluster for the integration to work.

To create a service account in the EDA Kubernetes cluster, use the following resource.

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: k8s-controller-plugin
  namespace: eda-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: k8s-controller-plugin
subjects:
- kind: ServiceAccount
  name: k8s-controller-plugin
  namespace: eda-system
roleRef:
  kind: ClusterRole
  # This cluster role is assumed to be already installed by connect app.
  name: eda-connect-plugin-cluster-role
  apiGroup: rbac.authorization.k8s.io
```

Note: This service account must be created in the eda-system namespace.

Creating a service account token

A service account token is used by the plugin to connect to the EDA Kubernetes cluster. You can create a service account token by using the following manifest, which should be applied on the EDA Kubernetes cluster.

```
---
apiVersion: v1
kind: Secret
type: kubernetes.io/service-account-token
metadata:
  name: k8s-controller-plugin

  annotations:
    kubernetes.io/service-account.name: k8s-controller-plugin
```

After creating the service account token, you can retrieve the actual token itself using the following command. This token will need to be provided to the plugin during deployment.

```
kubectl get secrets/k8s-controller-plugin \
  --template={{.data.token}} | base64 --decode
```

4.3.2 Fetching the EDA Connect OpenShift plugin Helm charts

About this task

There are two ways to obtain the Helm charts to deploy the EDA Connect OpenShift plugin:

Procedure

Step 1. Using the EDA Playground, which was used to install EDA, you can clone the Github repository:

```
make download-connect-k8s-helm-charts
```

Step 2. Downloading the release tarball and unpacking it:

```
curl -sLO https://github.com/nokia-eda/connect-k8s-helm-charts/archive/refs/tags/
3.0.0.tar.gz
tar zxf 3.0.0.tar.gz
```

4.3.3 Deploying the plugin in OpenShift

The following procedures are used on the OpenShift cluster in which the plugin needs to be deployed.

Creating a namespace for the OpenShift plugin

The OpenShift plugin uses its own namespace to separate it from other resources in the OpenShift cluster. You can create the correct namespace using the following command:

```
kubectl create namespace eda-connect-k8s-controller
```

Configuring a Pull Secret for the Controller Image

If the EDA Connect OpenShift plugin Controller image is hosted in a registry that requires authentication, a Kubernetes Secret needs to be created for OpenShift to be able to pull the image.

The following command does so for the officially hosted image with a secure read-only token to the registry.

```
export PULL_TOKEN=<PULL_TOKEN>
kubectl create secret docker-registry eda-k8s-image-secret \
  --docker-server=ghcr.io/nokia-eda/eda-connect-k8s-controller \
  --docker-username=nokia-eda-bot \
  --docker-password=${PULL_TOKEN} \
  -n eda-connect-k8s-controller
```

Setting up the local Helm values

Create a `helm-values.yaml` file with the following content and update the fields as appropriate:

```
controllerEnvConfig:
  connectpluginname: eda-openshift-controller-plugin
  heartbeat: "10"
  namespace: <eda-namespace> # The namespace of the Service Account in EDA
  skiptlsverify: False
  loglevel: info
  tlscertificatedata: <EDA K8s certificate data can be extracted from kubeconfig from cluster
 `certificate-authority-data`>
  tlsenabled: True
controllerEnvSecret:
  connectHost: https://<EDA-k8s-cluster-ip or hostname>:<port> # (Caution - Do not use EDA API
 values, use EDA k8s API values)
  connectPassword: <Secret long-lived token of the service account created before>
  connectUsername: <Name of the service account based on which secret token was created>
```

The following table describes possible Helm values:

Table 5: Possible Helm values

Helm value	Description
`connectpluginname`	A name for the plugin. Ensure this is a unique name within your EDA environment.
`heartbeat`	The interval in seconds at which the plugin should send heartbeats. Good values are 10-30, lower values can cause extra unnecessary load on the system.
`namespace`	A name of a namespace, where EDA Connect service is deployed.

Helm value	Description
`skiptlsverify`	This value can be enabled to disable server TLS certificate verification when connecting to the EDA Kubernetes cluster.
`tlscertificatedata`	When certificate validation is enabled, this property can contain the certificate information of the EDA Kubernetes cluster, similar to what a kubeconfig would contain. This is only needed if certificate validation is enabled and if the EDA Kubernetes certificate has not been signed by a trusted authority.
`tlsenabled`	This value should always be set to true to ensure TLS is used to secure the communication with the EDA Kubernetes cluster.
`connectHost`	The URL used to reach the EDA Kubernetes cluster API.
`connectPassword`	The long-lived token created in*Creating a service account token* in Creating a service account and token.
`connectUsername`	The service account name for the account created in *Creating a service account* in Creating a service account and token.

Deploying the OpenShift plugin using its Helm charts

You can now deploy the EDA Connect OpenShift plugin using its Helm charts with the following command:

```
helm install eda-k8s connect-k8s-helm-charts/ \
  -n eda-connect-k8s-controller \
  -f helm-values.yaml \
  --set controller.imagePullSecretName=eda-k8s-image-secret
```

4.3.4 Deployment verification

You can verify if the plugin was deployed by checking if the controller is running in the OpenShift Cluster.

```
$ kubectl get pods -n connect-k8s-controller
NAME                                             READY   STATUS  RESTARTS AGE
connect-k8s-controller-manager-c8d4875bc-bpzrx  2/2     Running 0        66m
```

On the EDA Kubernetes environment you can verify the plugin has been registered.

```
$ kubectl get connectplugins
NAME                                    PROVIDED NAME          PLUGIN TYPE   AGE
470e9af1-b85b-439b-b81a-ab71a7166bb0   k8s-controller-plugin   KUBERNETES    2h
```

4.4 Operational modes

The OpenShift Connect plugin operates in the following operational modes:

- Transparent operational mode
- Connect Network Definition operational mode
- NAD annotation operational mode

4.4.1 Using the Transparent operational mode

To use the Transparent operational mode, create network attachment definitions (NADs) in OpenShift without any EDA Connect annotations or any reference to the NAD in a Connect Network Definition (CND).

By doing so, the plugin creates a new EDA `BridgeDomain` for each NAD with a unique master (+VLAN) interface. If a NAD is created for which a NAD with the same master interface and VLAN already exists, it is associated with the existing `BridgeDomain`.

When you remove the NAD, the EDA bridge domain is also removed.

4.4.2 Using the Connect Network Definition operational mode

Connect Network Definition (CND) is a custom resource definition (CRD) that is added to the OpenShift cluster on deployment of the plugin.

A CND contains a design of all the network services and configuration an application may need. It can be used to define EDA Routers and EDA `BridgeDomain` resources. For each `BridgeDomain`, it is possible to associate one or more NADs with it. By doing so, the plugin knows how to connect applications into different network services.

A `preProvisionHostGroupSelector` can be used to pre-provision the Connect interface for a NAD interface on a set of selected hosts, regardless of whether they are consumed by any pod.

Note:
The `preProvisionHostGroupSelector` attribute is only applicable to IPVLAN and MACVLAN types of `NetworkAttachmentDefinitions`.

You can use the CMS-managed integration mode, the EDA-managed integration mode, or a combination of both.

Multiple CNDs can exist, for instance one per application.

Using the `ConnectNetworkDefinition`

The CND identifies how NADs are associated with different `BridgeDomain` resources. Multiple NADs can be associated with the same `BridgeDomain`.

Example: Multiple `NADs` in one `BridgeDomain`

The following is a sample configuration of the CND usage for multiple `NetworkAttachmentDefinitions` to be residing in a single subnet, when they belong to different master interfaces. This is an example of OpenShift Managed Mode.

```
apiVersion: connect.eda.nokia.com/v1
kind: ConnectNetworkDefinition
metadata:
  name: cnd1
spec:
  subnets:
    - name: "subnet1"
      networkAttachmentDefinitions:
        - name: ipvlan-ns1/ipvlan-nad1
        - name: ipvlan-ns1/ipvlan-nad2
```

Example: Multiple `NADs` in one `BridgeDomain` with VLAN Trunking

The following is a sample configuration of the CND usage for multiple `NetworkAttachmentDefinitions` to be residing in a single subnet, while using trunk VLANs.

```
apiVersion: connect.eda.nokia.com/v1
kind: ConnectNetworkDefinition
metadata:
  name: cnd2
spec:
  subnets:
    - name: "trunked-subnet1"
      networkAttachmentDefinitions:
        - name: ipvlan-ns1/ipvlan-nad1-untagged
          trunkVlans:
            - 10
        - name: sriov-ns1/sriov-nad1-untagged
          trunkVlans:
            - 20
```

Example: Using EDA-managed

The following is a sample configuration of the CND usage for multiple `NetworkAttachmentDefinitions` to be part of a single `BridgeDomain` that was pre-created in EDA.

```
apiVersion: connect.eda.nokia.com/v1
kind: ConnectNetworkDefinition
metadata:
  name: cnd3
spec:
  subnets:
    - name: "eda-managed-subnet1"
      linkedBridgeDomain: eda_bridgedomain_1
      networkAttachmentDefinitions:
        - name: ipvlan-ns1/ipvlan-nad1
        - name: sriov-ns1/sriov-nad1
```

Example: Using `preProvisionHostGroupSelector` to pre-provision Connect interface with a label selector

To use the `preProvisionHostGroupSelector` attribute in the CND, ensure that the OpenShift/K8s nodes are labeled correctly and the same value is used in the CND. The following example shows the labeling of a node.

```
kubectl label nodes node-1 connect.eda.nokia.com/hostGroup=net-group-1
kubectl label nodes node-2 connect.eda.nokia.com/hostGroup=net-group-2
```

The following sample configuration shows the pre-configuration of the Connect interface in the CND.

```
apiVersion: connect.eda.nokia.com/v1
kind: ConnectNetworkDefinition
metadata:
  name: cnd4
spec:
  subnets:
    - name: "subnet-1"
      networkAttachmentDefinitions:
        - name: ipvlan-ns1/ipvlan-nad1
          preProvisionHostGroupSelector: 'net-group-1'
        - name: macvlan-ns1/macvlan-nad1
          preProvisionHostGroupSelector: 'net-group-2'
```

4.4.3 Using the NAD Annotation operational mode

The NAD Annotation operational mode only works for the EDA-managed integration mode because it relies on an annotation on the NAD that identifies the pre-existing EDA `BridgeDomain` resource to which the NAD needs to be associated with.

To use this operational mode, when creating or updating a NAD, add the following annotation to it:

```
'connect.eda.nokia.com/bridgedomain': <eda-bridge-domain-name>
```

In case of VLAN trunking, a more complex annotation can be used, for example:

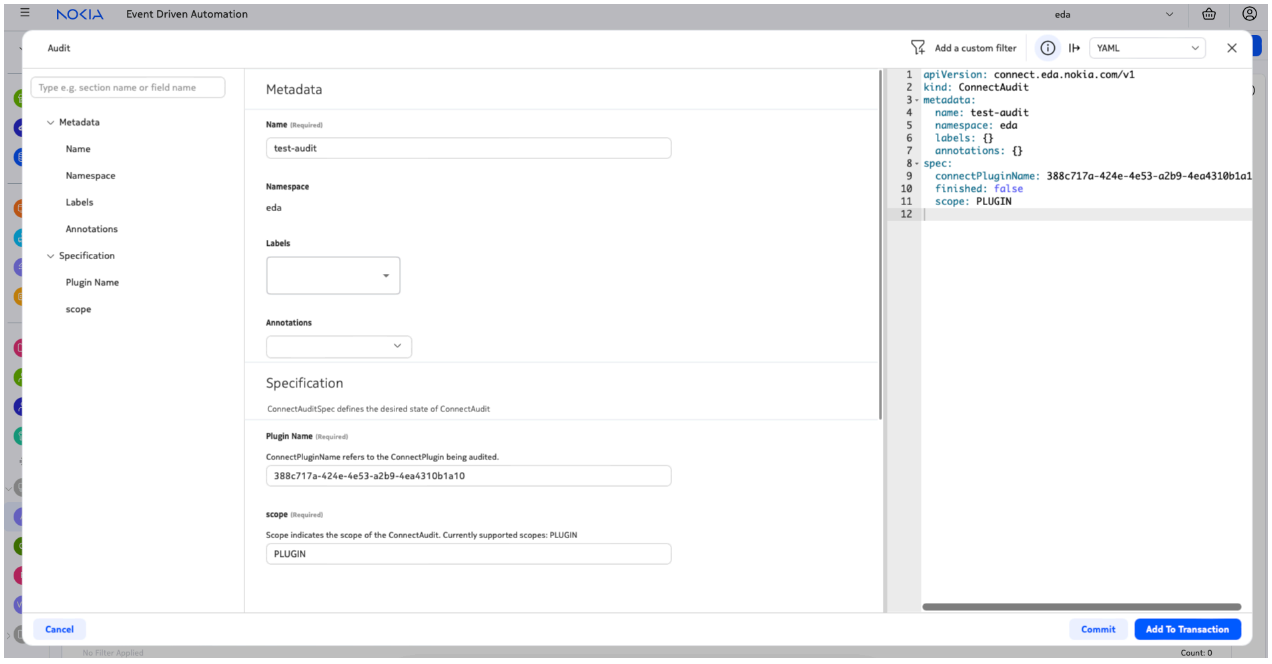
```
'connect.eda.nokia.com/bridgedomain': <eda-bridge-domain-name>:<vlan-id>, <eda-bridge-domain-
name-2>:<vlan-id>
```

4.5 Audit

The EDA Cloud Connect Plugins are listening to or directly interacting with the cloud platforms they manage. This connection can be broken temporarily or be out of sync. To automatically fix these out of sync events, Audit can be run on the Plugin.

An Audit can be launched through the UI, by navigating to **System Administration** → **Connect** → **Audit**.

Figure 1: Launch an audit



Metadata

Name (Required)
test-audit

Namespace
eda

Labels

Annotations

Specification
ConnectAuditSpec defines the desired state of ConnectAudit

Plugin Name (Required)
ConnectPluginName refers to the ConnectPlugin being audited.
388c717a-424e-4e53-a2b9-4ea4310b1a10

scope (Required)
Scope indicates the scope of the ConnectAudit. Currently supported scopes: PLUGIN
PLUGIN

```
apiVersion: connect.eda.nokia.com/v1
kind: ConnectAudit
metadata:
  name: test-audit
  namespace: eda
  labels: {}
  annotations: {}
spec:
  connectPluginName: 388c717a-424e-4e53-a2b9-4ea4310b1a1
  finished: false
  scope: PLUGIN
```

Alternatively, you can create an Audit resource in the Kubernetes cluster of EDA with the following content:

```
apiVersion: connect.eda.nokia.com/v1
kind: ConnectAudit
metadata:
  name: test-audit
  namespace: eda
spec:
  connectPluginName: <name of the plugin>
  scope: PLUGIN
```

Audit result

An Audit runs out of band inside the `Plugin`. This process can be followed using the `.status.state` field, which progresses from `Scheduled` to `InProgress` to `Finished`. When the Audit is finished, the `spec.finished` field will be true.

The following shows an example outout.

```
status:
  endTime: "2024-12-16T13:37:56Z"
  enqueueTime: "2024-12-16T13:37:56Z"
  outcome: Success
  results:
  - auditType: ConnectPluginAudit
    foundDiscrepancies:
    - connectResourceKind: BridgeDomain
      connectResourceName: 4030b313-60c5-4256-8135-57833913ce67
      outcome: Success
      pluginResourceKind: vlan on Distributed Virtual Portgroup
      type: Missing
    - connectResourceKind: Vlan
      connectResourceName: 33b01228-b522-434c-b099-26ff69ec57c4
      outcome: Success
      pluginResourceKind: vlan on Distributed Virtual Portgroup
      type: Dangling
    - connectResourceKind: Vlan
      connectResourceName: 0c7f00d6-5d0a-469f-85a7-6733e457df8c
      outcome: Success
      pluginResourceKind: vlan on Distributed Virtual Portgroup
      type: Missing
    outcome: Success
    state: Finished
  state: Finished
  totalNumberOfDiscrepancies: 3
  totalNumberOfSuccessfulDiscrepancies: 3
```

An Audit Status consists of one or more results, each one referencing a different part of the Audit. In the case of *PLUGIN*`Audit`, the only stage is the `Plugin` auditing against EDA, so typically there will be only one Result. The Result has an `AuditType` to indicate the stage of the `Audit`, as well as an `outcome`. The `outcome` is `Success` if the `Audit` was able to correct any found discrepancies.

If there are any discrepancies found, they are listed in the `foundDiscrepancies` list, detailing what resources were involved and what the taken action is. `Dangling` resources are resources left in EDA that are not available in the `Plugin` environment, while `missing` resources are the opposite. Finally, `Misconfigured` resources are available in both environments, but have one or more misconfigured fields.

A count is provided of the `totalNumberOfDiscrepancies` as well as successfully and failed fixes.

4.6 Troubleshooting

Controller plugin not running

Verify the following items:

- Service Account Token configuration is correct.
- Connectivity between controller pod in OpenShift and the EDA cluster.
- Heartbeat interval is a non-negative integer.
- Plugin name must comply with this regex check: `'([A-Za-z0-9][-A-Za-z0-9_.]*)?[A-Za-z0-9]'`. It may only contain alphanumerical characters and '.', '_', '-' (dot, underscore, and dash) and must start and end with an alphanumerical character.

Nothing is created in EDA

Verify the following items:

- Plugin controller can access the `NMstate` API and `NetworkAttachmentDefinition` API on the OpenShift cluster.
- Plugin can reach EDA cluster correctly.

Plugin is not configuring the correct state

Verify the following items:

- Inspect EDA resources, such as `VLAN`, `BridgeDomain` and `ConnectInterface`.
- Check the logs of the plugin pod.

5 VMware vSphere plugin

The VMware vSphere plugin leverages the VMware vSphere distributed vSwitch architecture to support management of the fabric directly from the VMware vCenter and ensures that the fabric responds to the networking needs of the environment.

The VMware vSphere plugin provides the following advantages and capabilities:

- Direct integration into the network management workflow of VMware vCenter.
- The use of the common distributed vSwitches and port groups for both regular Virtual Machine NICs as well as SR-IOV use cases.
- Supports the following VLAN types for port groups.
  - None: Vlan 0
  - VLAN: (1-4094)
- Automatic provisioning of the fabric based on where the virtual machines require the connectivity.
- Support advanced workflows through the EDA-managed solution, including for VNF use cases with features like QoS, ACLs, and BGP PE-CE.
- Interconnectivity between different cloud environments, allowing for flexible network configurations.

Supported versions

- VMware vSphere 7
- VMware vSphere 8

5.1 Prerequisites

Before installing or deploying the VMware vSphere plugin components, ensure that the Cloud Connect Core application is properly installed in the cluster.

5.2 Architecture

The VMware vSphere plugin consists of two components:

- VMware vSphere plugin app
- VMware vSphere plugin

VMware vSphere plugin app

This app runs in EDA and manages the lifecycle of the VMware vSphere plugins. It does so in the standard app model where a custom resource is used to manage the VMware vSphere plugins.

VMware vSphere plugin

The plugin itself is responsible for connecting and monitoring the VMware vCenter environment for changes. The plugin listens to the events of the following objects:

- Distributed vSwitch (dvS)
- Distributed port groups (dvPG)
- Host to dvS associations
- Custom attributes

Supported features

The following are some of the supported VMware vSphere features:

- CMS-managed integration mode
- EDA-managed integration mode
- Optimally configure subinterfaces to minimize configuration and security footprint of network services
- LAG/LACP interfaces
- SRIOV interfaces
- Audits

5.3 Deployment

To deploy the VMware vSphere plugin, complete the following tasks:

1. Deploy the plugin app.
2. Deploy the plugin.

5.3.1 Plugin app deployment

The VMware vSphere plugin app is an application in the EDA app eco-system. You can install it using the app Store UI.

Installation using Kubernetes API

If you prefer installing the Connect Core using the Kubernetes API, you can do so by creating the following Workflow resource:

```
apiVersion: core.eda.nokia.com/v1
kind: Workflow
metadata:
  name: vmware-plugin
  namespace: eda-system
spec:
  input:
    app: vmware-plugin
    catalog: eda-catalog-builtin-apps
    operation: install
    vendor: nokia
    version:
      type: semver
      value: v2.0.0
  type: app-installer
```

5.3.2 Plugin deployment

A prerequisite for creating a `vmwarePluginInstance` resource is a `Secret` with username and password fields that contain the account information for an account that can connect to the VMware vCenter environment and has read-only access to the cluster so that it can monitor the necessary resources.

See the following example:

```
apiVersion: v1
kind: Secret
type: Opaque
metadata:
  name: my-vmware-creds
  namespace: eda-system
data:
  username: YWRtaW4K # base64 encoded
  password: YWRtaW4K # base64 encoded
```

As the VMware vSphere plugins are managed through the operator, you can use the EDA UI to create a new `VmwarePluginInstance` resource under the **System Administration → Connect → VMware Plugins** menu item.

Alternatively, you can create the same `VmwarePluginInstance` resource using the following custom resource example. Make sure to replace the specified values with their relevant content.

```
apiVersion: vmware.eda.nokia.com/v1
kind: VmwarePluginInstance
metadata:
  name: my-vmware-plugin-instance # A unique name for the plugin resource (can be the same as
 the spec.name, or different)
  namespace: eda-system
spec:
  externalId: example-external-id # A unique Identifier for the plugin (can be same as the
 name)
  heartbeatInterval: 30
  name: example-vSphere # A unique name for the plugin
  vcsaHost: example-host # The IP address of the vCenter Server
  vcsaTlsVerify: true # To verify TLS of the VCSA.
  vcsaCertificate: "" # If the VCSA certificate is self signed, add it here to be able to
 verify from the plugin
  authSecretRef: my-vmware-creds # Credentials are hosted in a seperate Secret.
```

The plugin name and external ID must comply with the regex check of `'([A-Za-z0-9][-A-Za-z0-9_.]*)?[A-Za-z0-9]'` and can only contain alphanumerical characters and `.`, `_`, and `-`. It must start with an alphanumerical character.

5.4 Functionality

This section describes VMware vSphere plugin operations including startup, event monitoring, and the plugin's operational modes.

5.4.1 Startup

When the plugin is started, the following actions are taken by the plugin:

- The plugin registers itself with Connect, based on the provided `externalID`. If a matching `Connect Plugin` resource pre-exists, it is reused.
- The plugin performs an audit where any Connect-related state that was programmed in vCenter while the plugin was not running is synchronized with Connect.

5.4.2 Event monitoring

A plugin connects to a VMware vCenter environment and subscribes to VMware events. The plugin configures Connect and EDA based on the events it receives.

The following table describes the different event triggers and purposes.

Table 6: VMware events

Event trigger	Custom resource	Purpose
VLAN-tagged distributed Port Group events	`BridgeDomain`	In CMS-managed mode, each dvPG results in its own unique bridge domain.
VLAN-tagged distributed Port Group events	`VLAN`	Each dvPG with a specific VLAN tag has an EDA `VLAN` resource so it can be attached to the `BridgeDomain`.
Host NIC distributed Switch Uplink events	`ConnectInterface`	Each Host NIC that is added as an Uplink to a dvS triggers the creation of a `Connect Interface` which is mapped by Connect Core to an EDA `Interface`.

The uplink names must comply with the regex check of `^[a-zA-Z0-9][a-zA-Z0-9._-]*[a-zA-Z0-9]$`. It can only contain alphanumerical characters and " " (space), `.` (period), `_` (underscore), and `-` (dash). It must also have a length of 64 characters or less.

5.4.3 Operational modes

The plugin supports the following operational modes:

- CMS-managed mode
- EDA-managed mode

These modes can be used simultaneously.

CMS-managed mode

This mode is also referred to as VMware-managed mode. When using this mode, the plugin creates a unique `BridgeDomain` for each VLAN-tagged dvPG in the VMware vCenter environment.

EDA-managed mode

In EDA-managed mode, a dvPG is given a special custom attribute that refers to an existing EDA `BridgeDomain`. When the plugin detects this custom attribute, and it refers to an existing `BridgeDomain` resource in EDA, it does not create a new `BridgeDomain` but instead associates the dvPG with the existing one. This allows for more advanced configuration of the application networks.

To use the EDA-managed mode:

1. Create a `BridgeDomain` resource in EDA with the desired settings.
2. When creating a distributed PortGroup in vCenter, configure a custom attribute called `ConnectBridgeDomain` and set its value to the key of the EDA `BridgeDomain`.

Note: Both the key of the custom attribute and the value are case sensitive.

You can configure multiple dvPGs with the same `BridgeDomain` resource.

Switching between operational modes

You can switch between EDA-managed and CMS-managed modes at any time. You can switch back to CMS-managed mode by setting the `ConnectBridgeDomain` custom attribute to `none`, or by deleting the custom attribute entirely.

5.5 Troubleshooting

Plugin is not running

If an incorrect vCenter hostname or IP is configured in the `VmwarePluginInstance` resource, the plugin will try to connect for three minutes and crash/restart if it fails to connect. If the credentials are incorrect, the plugin will crash/restart immediately.

Verify the following items:

- Raised plugin alarms
- Connectivity from the EDA cluster to vCenter
- Credentials for vCenter
- Heartbeat interval is a positive integer
- Logs of the plugin pod

Plugin is not creating any resources in EDA

Verify the following items:

- Raised plugin alarms
- Connectivity from the EDA cluster to vCenter

- Logs of the plugin pod
- Heartbeats are being updated
- Plugin staleness state field

Plugin is not configuring the correct state

Verify the following items:

- Raised plugin alarms
- Uplinks for the dvPG in vCenter are configured as active or standby; if there are no active or standby uplinks configured, the plugin will not associate any `ConnectInterface` with the `VLAN`
- Uplink names can only contain alphanumerical characters and `.`, `_`, `-` and must have a length of 64 characters or less
- VLAN ranges are not supported on dvPGs
- EDA resources, like `VLAN`, `BridgeDomain` and `ConnectInterface`
- Logs of the plugin pod

6 OpenStack ML2 plugin

EDA Cloud Connect integrates with OpenStack to provide fabric level application networks for OpenStack virtual machines. The EDA Cloud Connect integration leverages the OpenStack Neutron architecture to support managing the fabric directly from OpenStack and allows the fabric to dynamically respond to the networking needs of the application.

It provides the following advantages and capabilities:

- Direct integration into the network management workflow of OpenStack.
- The use of the common ML2 plugins used by enterprise applications and VNFs like OVS, OVS-DPDK and SR-IOV.
- Automatic provisioning of the fabric based on where the virtual machines need the connectivity.
- Support of advanced workflows through EDA, including for VNF use cases with features like QoS, ACLs, and BGP PE-CE.
- Interconnectivity between different cloud environments, allowing for flexible network configurations.

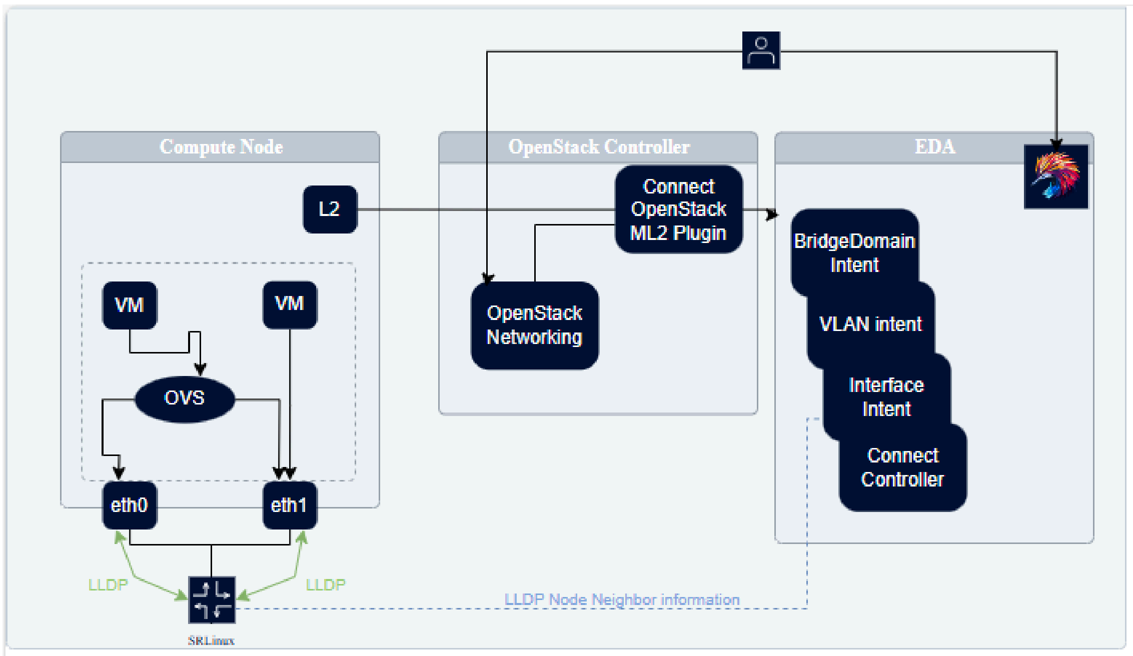
Supported versions

Currently, the OpenStack plugin is only supported with the Nokia CBIS 24 OpenStack distribution.

6.1 Architecture

The OpenStack Plugin deploys some components into the OpenStack environment to allow the management of the SR Linux-based fabric through OpenStack. Below is an overview of these components.

Figure 2: OpenStack Architecture



Connect ML2 plugin

The Connect ML2 plugin is the heart of the integration between OpenStack and EDA. This plugin integrates with OpenStack Neutron and reacts to the creation of networks, network segments, and VM ports.

When a network segment is created in OpenStack Neutron, a matching `BridgeDomain` is created in EDA.

When a VM port is created inside a Neutron Subnet and the VM is started on an OpenStack compute node, the ML2 plugin learns which compute node the VM is deployed and through the internal topology ensures the necessary `ConnectInterfaces` are configured in EDA.

This information is learned from the L2 Agent extension which stores the Neutron Network to physical interfaces topology in the Neutron database. This information is then provided to the Connect service and together with the LLDP information of the fabric, the Connect service knows which downlinks in the fabrics need to be configured.

Connect L2 Agent Extensions

The Connect L2 Agent Extensions extend the already existing L2 Agent that is present on every OpenStack compute. These extensions are responsible for mapping the relation between the physical NICs and the different networking constructs set up for the Neutron networks.

6.2 Deployment

This section describes how to prepare for the deployment of the OpenStack plugin and how to prepare and install the CBIS extension.

6.2.1 CBIS extension preparation

This section describes how to prepare for the installation of the CBIS extension for the EDA ML2 plugin.

The EDA ML2 plugin creates a `ConnectPlugin` within the Connect Service using the name of the CBIS Cloud deployment, as provisioned at installation time. Ensure that no `ConnectPlugin` exists with that name before the CBIS deployment.

6.2.1.1 Creating a service account and token

Creating a service account

The EDA Connect OpenStack Plugin uses a `ServiceAccount` resource in the EDA Kubernetes cluster to create the necessary resources in the EDA cluster for the integration to properly work.

To create a service account in the EDA Kubernetes cluster, the following resource can be used.

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: openstack-plugin
  namespace: eda-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: openstack-plugin
subjects:
- kind: ServiceAccount
  name: openstack-plugin
  namespace: eda-system
roleRef:
  kind: ClusterRole
  # This cluster role is assumed to be already installed by connect app.
  name: eda-connect-plugin-cluster-role
  apiGroup: rbac.authorization.k8s.io
```

Note:
This service account must be created in the `eda-system` namespace.

Creating a service account token

The Service Account Token can be used by the plugin to connect to the EDA Kubernetes cluster. You can create a service account token by using the following manifest, which should be applied on the EDA Kubernetes cluster.

```
---
apiVersion: v1
kind: Secret
type: kubernetes.io/service-account-token
metadata:
  name: openstack-plugin
  namespace: eda-system
  annotations:
    kubernetes.io/service-account.name: openstack-plugin
```

After creating the Service Account Token, you can retrieve the bearer token and `ca_cert` information using the following commands from the `eda-system` namespace.

```
kubectl get secrets/openstack-plugin -n eda-system --template={{.data.token}} | base64 --decode
kubectl get secrets/openstack-plugin -n eda-system --template={{.data.ca_cert}} | base64 --
decode
```

Note:
When using the OpenStack plugin in production, create one service account per plugin. This way, tokens can be revoked on a per-plugin basis.

6.2.1.2 Deploying the CBIS extension

About this task

The following procedure is used to install the CBIS extension.

Procedure

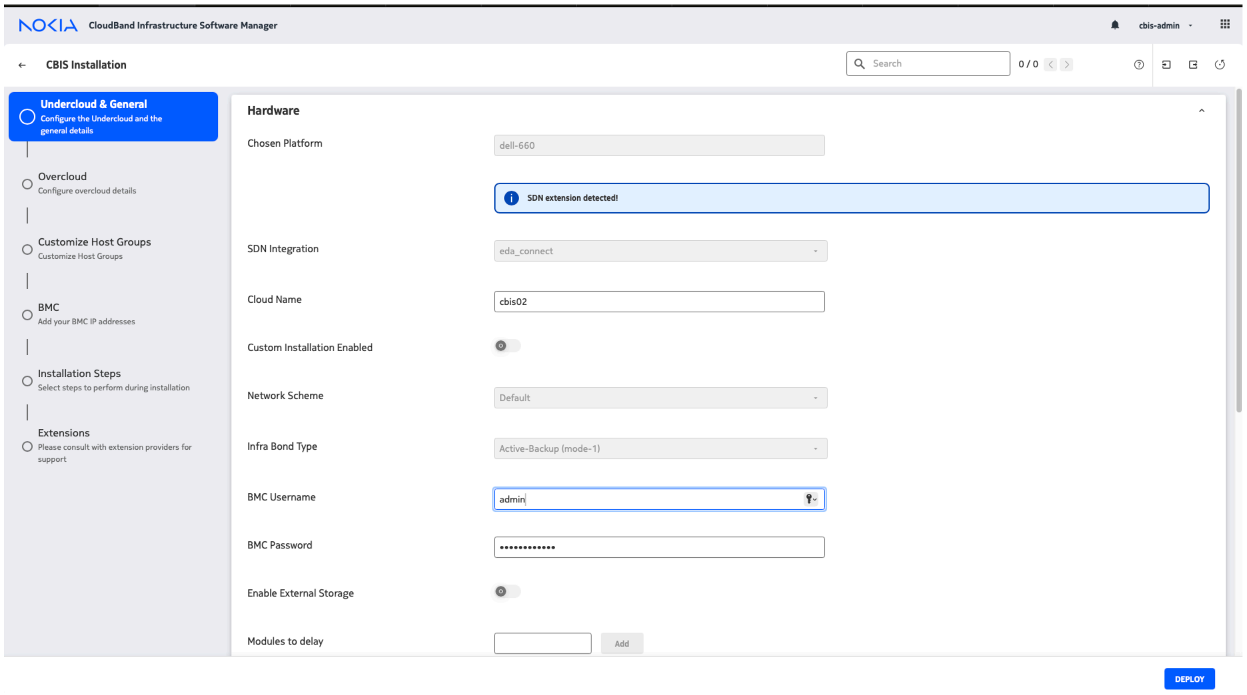
Step 1. Extract the tarball.

Untar the integration tarball and put the extracted content under `/root/cbis-extensions` inside the CBIS Manager VM.
This creates the `/root/cbis-extensions/eda_connect` folder.

Step 2. Refresh the CBIS manager UI.

The CBIS Manager should now detect `/root/cbis-extensions/eda_connect` as an SDN extension.

Figure 3: Detection of SDN extension by CBIS Manager


NOKIA CloudBand Infrastructure Software Manager
cbis-admin
CBIS Installation
Search
0 / 0
Undercloud & General
Configure the Undercloud and the general details
Overcloud
Configure overcloud details
Customize Host Groups
Customize Host Groups
BMC
Add your BMC IP addresses
Installation Steps
Select steps to perform during installation
Extensions
Please consult with extension providers for support
Hardware
Add
DEPLOY

SDN extension detected!

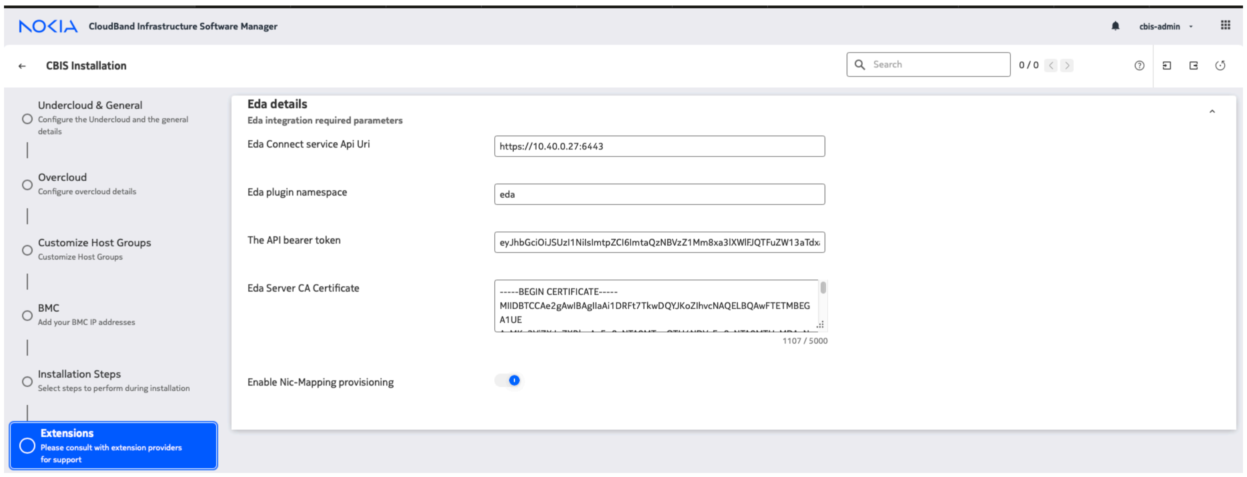
Chosen Platform	dell-660
SDN Integration	eda_connect
Cloud Name	cbis02
Custom Installation Enabled	
Network Scheme	Default
Infra Bond Type	Active-Backup (mode-1)
BMC Username	admin
BMC Password	••••••••••••
Enable External Storage	
Modules to delay	

Step 3. Add the extension configuration in CBIS manager UI.

Choose a unique name for each cloud. If there is another cloud name registered to the same EDA environment, the CBIS deployment will fail before undercloud deployment. An error message indicates that a ConnectPlugin exists with the same name.

Step 4. Enter the EDA environment details.

Figure 4: EDA environment details


CloudBand Infrastructure Software Manager
cbis-admin
CBIS Installation
Search
0 / 0
Undercloud & General
Configure the Undercloud and the general details
Overcloud
Configure overcloud details
Customize Host Groups
Customize Host Groups
BMC
Add your BMC IP addresses
Installation Steps
Select steps to perform during installation
Extensions
Please consult with extension providers for support

Eda details

Eda integration required parameters

Eda Connect service Api Uri	https://10.40.0.27:6443
Eda plugin namespace	eda
The API bearer token	eyJhbGciOiJSUzI1NiIsImtpZCI6ImtaQzNBVzZ1Mm8xa3lXWlFJQTFuZW13aTdx
Eda Server CA Certificate	-----BEGIN CERTIFICATE----- MIIDBTCCAe2gAwIBAgIIaAi1DRFt7TkwDQYJKoZIhvcNAQELBQAwFTETMBEG A1UE (1107 / 5000)
Enable Nic-Mapping provisioning	●

The Connect OpenStack plugin communicates with EDA using the Kubernetes API.

- **EDA Connect service API URI**
  URI of the Kubernetes API hosting EDA. This can be found in the `kubeconfig` file for your cluster.
- **EDA plugin namespace**
  The namespace in which the nodes for this cloud are deployed.
- **API Bearer Token**
  The bearer token needed to communicate with the Kubernetes API. This can be extracted from the service account token created in a previous step.
- **EDA Server CA Certificate**
  The self-signed CA certificate of the Kubernetes API. This can be extracted from the service account token created in a previous step.
- **Nic-Mapping provisioning**
  Allows manual creation of NIC-mapping entries.

6.2.1.3 Updating the bearer token

About this task

The following procedure is used when the bearer token needs to be updated after installation.

Procedure

Step 1. Update the `plugin.ini` file with the new token.

The configuration file can be found at `/etc/neutron/plugins/ml2/ml2_conf_eda_connect.ini`.

```
[ml2_eda_connect]
# Api host of the Connect service
#api_host = None
# Api bearer autentication token
api_token = <new api token>
# CA certificate file
ca_cert_path = /opt/stack/data/eda.pem
# Verify SSL
#verify_ssl = True
# Used as an identifier of the Connect plugin
#plugin_name = openstack
# Plugin heartbeat interval min=3, max=60 to Connect service
# in seconds.
#heartbeat_interval = 10
```

If you need to update the certificate info for connecting to the EDA Kubernetes API, replace the file referred to in `ca_cert_path`.

Step 2. Restart Neutron.

6.3 Neutron ML2 plugin configurations

The EDA Connect OpenStack plugin and the wider Neutron ML2 framework support a range of configurations that fine tune their behavior.

6.3.1 Virtualization and network types

The EDA Connect OpenStack ML2 plugin supports the following segmentation types:

- **VLAN**
  The plugin orchestrates on VLAN neutron networks, programming the EDA Cloud Connect service for fabric-offloaded forwarding.
- **VXLAN and GRE**
  The plugin does not orchestrate on VXLAN or GRE neutron networks, but is designed to be tolerant of other Neutron ML2 mechanism drivers.

When using another ML2 mechanism driver to provision these networks, create the relevant `VNET` or `BridgeDomain` and `VLAN` resources in EDA, as the plugin does not automatically take care of those. Typically, these utilise the `untagged` VLAN to communicate between the nodes.

The ML2 plugin also supports the following virtualization types:

- VIRTIO
- SRIOV
- DPDK

6.3.2 Networking models

The ML2 plugin supports the following networking models:

- OpenStack managed networking
- EDA managed networking

6.3.3 Forwarding capabilities

The ML2 plugin supports the following forwarding capabilities:

- L2 forwarding
  - Fabric-offloaded
- L3 forwarding
  - Using EDA managed networking model, fabric-offloaded
  - Using OpenStack managed networking model, using virtualized routing on OpenStack controller nodes

6.3.4 Bonding

The ML2 plugin supports the following CBIS-supported bonding models:

- VIRTIO and SRIOV (Linux bonds)
- DPDK (OVS bonds)

VIRTIO and SRIOV (Linux bonds)

Active/Backup for VIRTIO and SRIOV ports is supported.

No LAG should be configured for Active/Backup in the `Interfaces` in EDA.

Active/Active for VIRTIO and SRIOV ports is supported by the ML2 plugin, but not by CBIS.

DPDK (OVS bonds)

LACP-based Active/Active is supported.

A LAG should be configured for Active/Active in the `Interfaces` in EDA, with corresponding LACP configuration.

As on CBIS SRIOV computes, both the infra bonds and tenant bonds are SRIOV agent-managed. The fabric must be wired accordingly and must have all NICs wired into the fabric.

Active/Standby is not supported by the ML2 plugin for DPDK ports.

If required, a simplified setup can be created, where only the tenant bond NICs into the fabric. The following table shows an example of this setup.

Table 7: DPDK simplified setup

Port name	Number of VFs on port	Enable trust on port	Port to Physnet Mapping	Allow untagged traffic in VGT	Allow infra VLANs in VGT
nic_1_port_1	45	false	none	false	false
nic_1_port_2	45	false	none	false	false
nic_2_port_1	45	false	physnet1	false	false
nic_2_port_2	45	false	physnet2	false	false

6.3.5 Trunking

The network trunk service allows multiple networks to be connected to an instance using a single virtual NIC (vNIC). Multiple networks can be presented to an instance by connecting it to a single port.

For details about the configuration and operation of the network trunk service, see the OpenStack Neutron Documentation.

Trunking is supported for VRTIO, DPDK and SRIOV.

- For vnic_type=normal ports (VIRTIO/DPDK), trunking is supported through an upstream openvswitch trunk driver.
- For vnic_type=direct ports (SRIOV), trunking is supported by the EDA Connect ML2 plugin trunk driver.

Using trunks with SRIOV has some limitations with regard to the upstream OVS trunk model:

- Both parent ports of the trunk and all subports must be created with the `vnic_type` `direct`.
- To avoid the need for QinQ on switches, trunks for the SRIOV instance must be created with parent port belonging to a flat network (untagged).
- If multiple projects within a deployment must be able to use trunks, the Neutron network above must be created as shared (using the --share attribute).
- When adding subports to a trunk, their `segmentation-type` must be specified as VLAN and their `segmentation-id` must be equal to a `segmentation-id` of the Neutron network that the subport belongs to.

6.3.6 Network VLAN segmentation and segregation

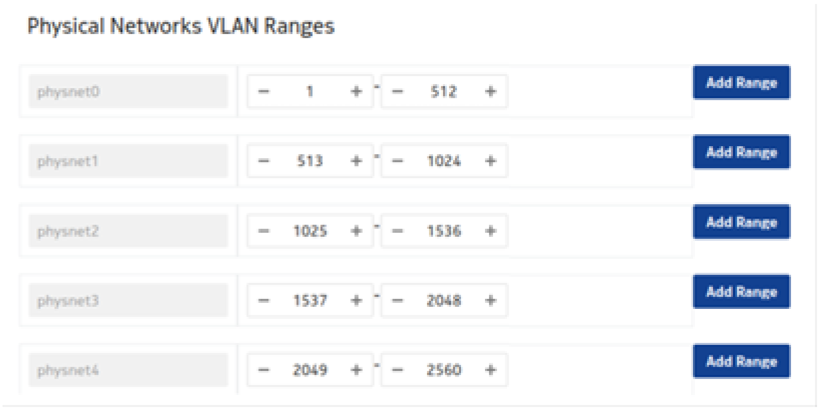
The ML2 plugin only acts on VLAN Neutron networks. To use VLAN networking, configure `provider_network_type = vlan`.

CBIS applies an interconnected bridge model on the OpenStack controller nodes to support multi-physnet host interfaces and to be more economical on the usage of physical NICs. As a result, when VLAN networks with overlapping segmentation IDs across physnets are applied, care must be taken that no overlapping segmentation IDs are wired on a host interface. Such a configuration would not be supported by the SR Linux fabric (or any other fabric).

A typical case would be if DHCP is enabled on the subnets created on `segmentation-id` overlapping networks, as the Neutron DHCP agent on the OpenStack controller nodes would effectively end up in the described conflicting state. This can be avoided by disabling DHCP on the subnets, that is, defining the subnet with `dhcp_enable = false`. Even when DHCP is not effectively consumed by any VNF deployed in the related networks, the conflict would still occur when DHCP is enabled on the subnets.

If the deployment use cases demand this wiring (for example, some of the deployed VNFs rely on DHCP), the system's VLAN ranges must be segregated per physical network at CBIS installation time. The following figure shows an example configuration.

Figure 5: Physical networks VLAN ranges segregation



Physical Networks VLAN Ranges

physnet0	1	512	Add Range
physnet1	513	1024	Add Range
physnet2	1025	1536	Add Range
physnet3	1537	2048	Add Range
physnet4	2049	2560	Add Range

6.4 Usage and managed networks

The Connect-based OpenStack solution is fully compatible with OpenStack. Connect also supports networking managed by EDA itself.

6.4.1 OpenStack managed networks

The Connect-based OpenStack solution is fully compatible with OpenStack. The standard API commands for creating networks and subnets remain valid.

When mapping to Connect:

- The network in OpenStack is created within the user's project. If the network is created by an administrator, the administrator can optionally select the project to which the network belongs. For each network segment, a `BridgeDomain` resource is created.
- Only VLAN-based networks (where `segmentation_type == vlan`) are handled by the EDA ML2 plugin, meaning only VLAN networks are HW-offloaded to the fabric managed by EDA. Neutron defines the default segmentation type for tenant networks. If you want non-administrators to create their own networks that are offloaded to the EDA fabric, the segmentation type must be set to `vlan`. However, if only administrators need this capability, then the segmentation type can be left to its default, because administrators can specify the segmentation type as part of network creation.
- The plugin only supports Layer 2 EDA fabric offloaded networking. When a Layer 3 model is defined in OpenStack (using router and router attachments), this model works. However, Layer 3 is software-based according to the native OpenStack implementation.

Layer 3-associated features, like floating IPs, also work but these are only software-based according to the native OpenStack implementation. This is tested with the traditional non-Distributed Virtual Routing (DVR) implementation only. DVR has not been tested, and might not work.

- MTU provisioning is not supported.

6.4.2 EDA managed networks

About this task

In addition to the OpenStack managed networking model, Connect supports networking managed by EDA itself. In this case, `VNET` or `BridgeDomain` resources are created first in EDA directly and then are consumed in OpenStack.

The `eda_bridge_domain` extension supports EDA managed networking. This refers to the `Name` of a `BridgeDomain` that you want to link the network to.

Procedure

Step 1. Using the EDA rest API, the Kubernetes interface or the UI, create a VNET with `BridgeDomain` or a standalone `BridgeDomain`.

Step 2. Obtain the name of the resources.

The Connect OpenStack ML2 plugin can only see resources in its own namespace, cross namespace referencing is not supported.

Step 3. Create OpenStack resources.

The following is an example workflow.

a. In OpenStack create the consuming network, and link it to the pre-created entity.

```
openstack network create --eda-bridge-domain xyz os-network-1
```

Since BridgeDomains are actually mapped at the network segment level, you can also specify `--eda-bridge-domain` when creating a network segment.

```
openstack network segment create --network os-network-1 --network-type vlan --eda-
bridge-domain xyz os-network-segment-1
```

b. Create a subnet within this network using the standard API syntax. For example:

```
openstack subnet create --network os-network-1 --subnet-range 10.10.1.0/24 os-
subnet-1
```

c. Create a Neutron port and Nova server within this network and a subnet that also uses standard APIs.

The OpenStack ML2 plugin handles the creation of `VLAN` and `ConnectInterface` resources as in the OpenStack managed use case. However, the `BridgeDomain` is fully owned by the operator. Possible use cases for this are fabric-based L3 or BGP peering.

6.4.3 Edge topology introspection

Automated Edge Topology Introspection

When the NIC-mapping agent extension is enabled, it persists the `physnet <-> compute,interface` relation in the Neutron database so it can wire Neutron ports properly in the Fabric.

The known interface mappings can be consulted using the CLI command **openstack eda interface mapping list**.

Automated LLDP Provisioning

When deploying using the CBIS integration, LLDP introspection is automatically enabled for all computes.

7 Connect alarms

EDA Connect, both Core and the plugins, can raise alarms if something is wrong and an action is needed by the administrator. These alarms are raised through the standard EDA Alarm system.

7.1 Connect service alarms

When a plugin is registered with Connect with a `HeartbeatInterval` greater than or equal to one, Connect's plugin controller periodically evaluates whether the plugin is sending heartbeats as promised.

If a plugin fails to do so, the ConnectPlugin's status includes a `stale=true` field and the following alarm is raised:

```
apiVersion: core.eda.nokia.com/v1
kind: Alarm
metadata:
  name: <plugin_name>-stale-alarm
  namespace: <plugin_namespace>
spec:
  description: This alarm is raised when a plugin fails to send heartbeats within the Heartbeat
Interval it is registered with.
  kind: ConnectPluginStale
  name: ConnectPluginStale-<plugin_name>
  objectKind: ConnectPlugin
  objectName: <plugin_name>
  probableCause: This alarm is likely to be triggered when the plugin has lost connection to
 EDA, or when the plugin is in a bad state.
  remedialAction: Check the plugin's connection to the EDA system. If there are no connection
 issues, try restarting the plugin.
  severity: critical
```

7.2 Plugin alarms

This section describes the OpenShift and VMware plugin alarms.

7.2.1 OpenShift plugin alarms

The OpenShift plugin can raise the following alarms:

Table 8: OpenShift plugin alarms

Alarm	Description
ConnectResourceConflict	Raised when a conflict is found between the configuration in the plugin versus the expected configuration in EDA.

Alarm	Description
ConnectEdaManagedBridgeDomainNotFound	Raised when EDA-managed is used and a `Bridge Domain` name is provided that does not exist in the namespace the plugin is registered in.
ConnectMissingHostInterface	Raised when a NAD references an interface, but it does not exist or is invalid.
ConnectMisconfiguration	Generic alarm for any misconfiguration found in a CND or NAD Annotation.
ConnectMissingLLDPNeighbor	Raised if the plugin cannot read the LLDP information from NMState for a specific interface.

7.2.2 VMware plugin alarms

The VMware plugin can raise the following alarms:

Table 9: VMware plugin alarms

Alarms	Description
ConnectCmsConnectivityFailure	The plugin fails to connect to the VMware vCenter.
ConnectCmsAuthenticationFailure	The plugin fails to authenticate to the VMware v Center.
ConnectDvpUplinkValidationFailure	An invalid Uplink configuration is found for a dv Switch or dvPortGroup or both.
ConnectEdaManagedBridgeDomainNotFound	Raised when EDA-managed is used and a `Bridge Domain` name is provided that does not exist in the namespace the plugin is registered in.

Customer document and product support

Customer documentation
Customer documentation welcome page

Technical support
Product support portal

Documentation feedback
Customer documentation feedback