Event Driven Automation

Release 24.12.3

Installation Guide

3HE 20938 AAAD TQZZA
Edition: 1
March 2025

Nokia is committed to diversity and inclusion. We are continuously reviewing our customer documentation and consulting with standards bodies to ensure that terminology is inclusive and aligned with the industry. Our future customer documentation will be updated accordingly.

Table of contents

1 About this document

This document provides the information you need to prepare for the installation of the EDA application and provides EDA installation procedures.

This document is intended for network technicians, administrators, operators, service providers, and others who need to understand how to install and provision EDA for deployment.

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

1.1 What's new

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

Table 1: What's new in EDA 24.12.3

Added a new section for EDA installations in an air-gapped environment	Air-gapped installations
Updates Creating the VM on bridged networks on VMware vSphere	Adds a note to Step 4.

Table 2: What's new in EDA 24.12.2

Description	Location
Improvements to procedures	• Networking for EDA nodes • Deployment models • Installation platform requirements • Downloading the VMware OVA image • Generating the Talos machine configurations • Creating the VM on bridged networks on KVM • Setting up the Rook Ceph storage cluster • Bootstrap the Talos Kubernetes cluster • Installing Nokia EDA

1.2 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.

2 EDA installation overview

This chapter describes the Nokia Event Driven Automation (EDA) components, the requirements for these components, and provides an overview of the installation process.

2.1 Deployment models

Nokia EDA is deployed on one, three, or more nodes (validated for up to six nodes). Nokia EDA is deployed as an application on virtual machine servers.

The nodes (VMs) run a Kubernetes cluster with the following composition:

- One or three Kubernetes master nodes that also function as worker nodes: one, in case a single-VM deployment is used; otherwise, three Kubernetes master nodes.
- Any remaining nodes (in a four or more node deployment) function as worker nodes.
- One, two or more nodes must also be designated as storage nodes. For redundancy, two is the minimum in a three or more node deployment. These nodes still function as worker (and potentially master) nodes as well. Rook-Ceph is used to create a storage cluster across the nodes indicated as storage nodes.

Related topics

Nokia EDA node requirements

2.2 Networking for EDA nodes

This guide describes the deployment of EDA on a Kubernetes cluster with two networks, where access from users and orchestrators to the UI and API, and access from EDA to the fabric (for example, SR Linux devices) will use different interfaces.

It is possible to use two separate networks for the EDA nodes:

- OAM network
  This interface is used to access the UI and the API of Nokia EDA. It is also through this network that the deployment tool reaches the nodes.
- Fabric management network
  This interface is used to communicate with the management interfaces of the fabric (for example, SR Linux devices). This interface is where Nokia EDA exposes its DHCP and ZTP services.

2.3 EDA nodes

The Nokia EDA nodes are the VMware vSphere-based or KVM-based virtual machines (VMs) that host the Kubernetes environment on which the Nokia EDA application and Digital Sandbox are run.

These nodes run a hardened Talos Kubernetes environment. Talos is a secure, up-to-date and hardened platform for running Kubernetes.

EDA supports the following deployment models:

- an environment with one node, which hosts only the Nokia EDA application for small scale deployments
- an environment with three nodes or more nodes, which hosts only the Nokia EDA application

2.4 Installation options

This document provides the provides procedures for the following installation scenarios:

- Internet-based installations
  In this type of installation, Internet connectivity is available throughout the installation procedures and the lifecycle of the EDA deployment. For instructions see, Internet-based installations.
- Air-gapped installations
  In an air-gapped environment, during most of the procedures, the system on which EDA is installed is isolated, that is, without Internet connectivity. An Assets VM is deployed with the services to provide the container images, git repositories and artifacts used during installation of the EDA Talos Kubernetes cluster and EDA itself. For instructions, see Air-gapped installations

3 Internet-based installations

This section provides the provides the procedures for typical installations.

3.1 Installation process overview

The installation consists of the following high-level tasks:

1. Downloading the EDA Installation playground
   This task describes how to access the EDAADM tool and initiates the EDA installation playground for use during the installation. It also covers how to configure the playground.
2. Download the Talos machine image
   This task describes how to download the Talos base image from the official Talos image factory for your environment.
3. Preparing the EDAADM configuration file
   This task describes the details of the EDAADM configuration file and how to set it up.
4. Generating the Talos machine configurations
   Using the EDA ADM tool and the configuration file, this task generates specific Talos machine configuration files for each Talos VM.
5. Deploying the Talos virtual machines
   This task describes how to use the Talos base image and machine configuration files to deploy the Talos VMs in your KVM or VMware vSphere environment.
6. Bootstrap the Talos Kubernetes cluster
   This task bootstraps the Talos Kubernetes environment using the VMs you have created.
7. Installing the EDA application
   Using the EDA Installation playground, this step installs EDA on the Kubernetes environment in the EDA nodes.

3.2 Requirements for deployment

This section describes the platform requirements, node requirements, and virtual IP requirements for deploying EDA for a typical installation.

3.2.1 Installation platform requirements

To execute the installation process, you need access to a Linux environment with the following components installed.

Note: When installing on a KVM-based hypervisor, the platform must use Red Hat Enterprise Linux or Rocky Linux. The installation steps of the installation platform requirements are for the

platform running the edaadm and playground steps, not creating the VMs on KVM (this happens on the RHEL/Rocky hypervisor directly).

Table 3: Platform requirements

Component	Requirement
Linux environment	Any Linux distribution. The procedures provided in this document are validated on Ubuntu.
Container runtime: Docker	Docker must be running and you should be able to run containers
Tools	• **make** • **git** • **curl** • **wget** • **helm** The following tools are also helpful. If they are not present, the installation tool downloads them later: • **kubectl** • **yq** • **k9s** • **kpt**
Internet access	Either directly or through a proxy

3.2.2 Nokia EDA node requirements

The Nokia EDA nodes are deployed as virtual machine servers. Table 4: Node requirements summarizes the requirements of Nokia EDA nodes in KVM and VMware hypervisor.

Table 4: Node requirements

Component	Requirement
CPU	32 vCPU on a modern x86-64 CPU
Memory	64GB
Storage	• Operating system: 100GB of available SSD-based storage • Storage nodes: 300GB of available SSD-based storage on a separate virtual disk
Networking	• at least one 10 Gbps NIC • the configured DNS servers must be reachable, functional, and able to resolve the hostnames used for the Nokia EDA nodes

Component	Requirement
	• Internet access directly or through a proxy
Virtualization platform	You can run the Nokia EDA nodes as virtual machines using the following virtualization platforms: • operating system: VMware vSphere 7.0 or 8.0 or RHEL/Rocky • hypervisor: ESXi 7.0 or 8.0 or KVM • resource reservation for CPU, memory, and disks must be set to 100% for the Nokia EDA node virtual machines

3.2.3 Virtual IP address requirements

The deployment requires two virtual IP addresses in the management network work:

- Kubernetes VIP: the virtual IP address used by all the control plane nodes in the Kubernetes cluster.
- Nokia EDA API/UI VIP: the virtual IP address used by the Nokia EDA API and UI.
- Nokia EDA fabric management VIP address: in a setup with two interfaces, the virtual IP address used by the Nokia EDA API and DHCP relay for the fabric management network.

3.3 Preparing for installation

This section includes the tasks that you need to complete to prepare for the installation of EDA.

3.3.1 Downloading the EDA Installation playground

Prerequisites

Ensure that your Linux installation environment meets the requirements described in Installation platform requirements. Git must already be installed on your computer.

Procedure

Clone the `playground` repository locally.

```
git clone https://github.com/nokia-eda/playground && cd playground
```

3.3.1.1 Downloading the EDAADM tool

Procedure

The EDAADM tool is used to generate configuration files for use while deploying the Talos Linux virtual machines and the Kubernetes environment.

You can download the binary for different platforms from https://github.com/nokia-eda/edaadm/releases/. Make sure to download the version that matches your operating system and CPU architecture.

Place the tool in a handy location and make sure you can execute it.

3.3.1.2 Installing additional tools

Procedure

Step 1. Set up additional tools that can be used during the installation.

You can download these tools to a local folder.

Example

```
make download-tools
```

Step 2. Verify that tools have been downloaded.

The `kind`, `kubectl`, `kpt`, and `yq` utilities are installed in the `~/tools/` directory.

3.3.1.3 Obtaining the EDA packages

About this task

EDA is packaged using the Kubernetes Package Tool (kpt). EDA uses this package manager tool to install core EDA components. The installer downloads two kpt packages by downloading their relevant git repositories.

Procedure

To obtain the EDA package, enter the following command:

```
make download-pkgs
```

This command downloads the following git repositories to their respective directories:

- EDA kpt package in the `~/eda-kpt` directory
- EDA built-in catalog in the `~/catalog` directory

3.3.2 Download the Talos machine image

The EDAADM tool provides you with the URL where you can download the latest image VMware OVA image or the image for use with KVM.

To deploy the Talos Kubernetes environment, download the Talos machine image based on the environment in which you want to deploy the VMs.

3.3.2.1 Downloading the KVM image

Procedure

Step 1. Use the EDAADM tool to display the URL from where you can download the latest image for use with KVM for the supported Talos version.

Example

```
$ edaadm images --mach-type nocloud
Schematic ID is :376567988ad370138ad8b2698212367b8edcb69b5fd68c80be1f2ec7d603b4ba
```

```
Asset URLs are:
https://factory.talos.dev/image/
376567988ad370138ad8b2698212367b8edcb69b5fd68c80be1f2ec7d603b4ba/v1.8.3/nocloud-
amd64.iso
https://factory.talos.dev/image/
376567988ad370138ad8b2698212367b8edcb69b5fd68c80be1f2ec7d603b4ba/v1.8.3/nocloud-
amd64.raw.xz
```

Step 2. Download the `nocloud-amd64.iso` image from the ISO URL, *`filepath`*`.iso`.

You can download using your browser or you can use the **curl** or **wget** commands.

3.3.2.2 Downloading the VMware OVA image

Procedure

Step 1. Use the EDAADM tool to display the URL from where you can download latest image for use with VMware vSphere for the supported Talos version.

Example

```
$ edaadm images --mach-type vmware
Schematic ID is : 903b2da78f99adef03cbbd4df6714563823f63218508800751560d3bc3557e40
Asset URLs are:
https://factory.talos.dev/image/
903b2da78f99adef03cbbd4df6714563823f63218508800751560d3bc3557e40/v1.8.3/vmware-
amd64.iso
https://factory.talos.dev/image/
903b2da78f99adef03cbbd4df6714563823f63218508800751560d3bc3557e40/v1.8.3/vmware-
amd64.ova
```

Step 2. Download the `vmware-amd64.iso` image from the OVA URL, *`filepath`*`.ova`.

You can download using your browser or you can use the **curl** or **wget** commands. You can also use the URL directly with the `ovftool` command to deploy the OVA to your VMware vSphere environment.

3.4 Setting up the EDA virtual machine nodes

This section describes how to the prepare the configurations file, generate the configuration files, and deploy the Talos virtual machines.

3.4.1 Preparing the EDAADM configuration file

The EDAADM tool helps with the creation of the necessary machine configuration files for the Talos VMs that are part of your deployment.

3.4.1.1 EDAAM configuration file fields

The EDAADM configuration file is a YAML file that describes your Talos Kubernetes environment. You can use it to configure the different nodes and the general Kubernetes cluster environment.

Table 5: Talos Kubernetes environment parameters

Top-level parameters	Description	
`version`	The version of the EDA environment to be deployed. Example: `24.12.1`	
`clusterName`	The name of your EDA environment. Example: `eda-production-cluster`	
`machines`	A list of Kubernetes nodes. Each Kubernetes node has the following settings:	
	`name`	The name of a node. Example: `eda-node01`
	`endpoint`	The IP address on which the node is reachable for Talos to control. This field must be set to one of the IP addresses configured for the interfaces on the node. Optional.
	`interfaces`	A list of interfaces present in the node, each with the following settings: • `name`: the name of the interface. Example: `eth0` • `dhcp`: indicates if DHCP is to be used for the interface. Values: `true` or `false`. For production environments, set to `false`. • `mtu`: the MTU setting for the interface. For an interface used to connect to nodes under management, set to 9000 for best practice. Optional. • `interface`: the interface name as it appears in Linux. Typically, `eth0`, `eth1`, and so forth. Optional. • `addresses`: a list of IP addresses; for dual-stack deployments, you can specify both IPv4 and IPv6 addresses. If DHCP is not provided, specify at least one address. • `routes`: a list of static routes to configure, including the default route. Optional. Routes have the following components: – `gateway`: the next-hop or gateway for the route. – `metric`: a metric to indicate the priority of the route. Optional. – `mtu`: a specific MTU for the route. Optional.

Top-level parameters	Description	
		– `network`: the destination CIDR of the route. – `source`: a source interface for the route to apply to. Optional. • `deviceSelector`: specifies how to select the device associated with this interface. – `busPath`: a PCI buspath that can contain wildcards. Optional. – `hardwareAddr`: a MAC address that can contain wildcards. Optional.
	`disks`	Identifies the disks available in the node: • `os`: Specifies which disk to use for the OS. Required setting. Typically `/dev/sda` or `/dev/vda`, depending on the hypervisor platform. • `storage`: Optional disk for use with nodes that are to be part of the storage cluster.
`k8s`	The Kubernetes-specific configuration. The following parameters define the Kubernetes cluster:	
	`stack`	Indicates the network stack to support. Values: `ipv4`, `ipv6`, or `dual`
	`primaryNode`	The first control plane node in the cluster to be used for bootstrapping the Kubernetes cluster. Specify the a the name of a machine.
	`endpointUrl`	The URL on which to reach the Kubernetes control plane. This setting uses the Kubernetes VIP address. Example: `https://192.0.2.10:6443`
	`allowScheduling OnControlPlanes`	Specifies if workloads can be deployed on the control plane node. Values: `true` or `false`. For best practice, set to `true`.
	`control-plane`	A list of control plane nodes. Specify a machine name.
	`worker`	A list of worker nodes. Specify a machine name.
	`vip`	The VIP addresses used for Kubernetes and the interfaces to which they should be attached in the control plane nodes. Depending on the IP stack in use, some values are required: • `interface`: the interface to which the VIP is attached on the nodes.

Top-level parameters	Description	
		Example: `eth0` • `ipv4`: the IPv4 VIP address. Example: `192.0.2.10` • `ipv6`: the IPv6 VIP address
	`env`	Section that includes the optional proxy settings for the Kubernetes nodes: • `http_proxy`: The HTTP proxy URL to use. Example: `http://192.0.2.254:808` • `https_proxy`: the HTTPS proxy URL to use. Example: `http://192.0.2.254:808` • `no_proxy`: the no proxy setting for IP addresses, IP ranges, and hostnames
	`time`	Defines NTP settings. • `disabled`: Specifies whether NTP is enabled. For production environments, set to `false` to enable NTP. • `servers`: A list of NTP servers; required for production environments.
	`nameservers`	A list of DNS servers specified under the following sub-element: `servers`: the list of DNS servers
	`certBundle`	An optional set of PEM-formatted certificates that need to be trusted; this setting is used for trust external services.

3.4.1.2 Example EDAADM configuration file

The following example shows an EDAADM configuration file for a 6-node Kubernetes cluster called `eda-input-6-node.yaml`. This file serves as the reference and example throughout this guide.

In this configuration file, two networks are used. If your deployment only uses one network, the second interface would not be present for each of the machines.

```
version: 24.12.1
clusterName: eda-compute-cluster
machines:
    - name: eda-node01
      endpoint: "192.0.2.11"
      interfaces:
        - name: eth0
          dhcp: false
          interface: eth0
```

```
        addresses:
          - 192.0.2.11/24
        routes:
          - network: 0.0.0.0/0
            gateway: 192.0.2.1
        mtu: 9000
      - name: eth1
        dhcp: false
        interface: eth1
        addresses:
          - 203.0.113.11/24
        mtu: 9000
    disks:
      os: /dev/vda
      storage: /dev/vdb
  - name: eda-node02
    endpoint: "192.0.2.12"
    interfaces:
      - name: eth0
        dhcp: false
        interface: eth0
        addresses:
          - 192.0.2.12/24
        routes:
          - network: 0.0.0.0/0
            gateway: 192.0.2.1
        mtu: 9000
      - name: eth1
        dhcp: false
        interface: eth1
        addresses:
          - 203.0.113.12/24
        mtu: 9000
    disks:
      os: /dev/vda
      storage: /dev/vdb
  - name: eda-node03
    endpoint: "192.0.2.13"
    interfaces:
      - name: eth0
        dhcp: false
        interface: eth0
        addresses:
          - 192.0.2.13/24
        routes:
          - network: 0.0.0.0/0
            gateway: 192.0.2.1
        mtu: 9000
      - name: eth1
        dhcp: false
        interface: eth1
        addresses:
          - 203.0.113.13/24
        mtu: 9000
    disks:
      os: /dev/vda
      storage: /dev/vdb
  - name: eda-node04
    endpoint: "192.0.2.14"
    interfaces:
      - name: eth0
        dhcp: false
        interface: eth0
        addresses:
```

```
            - 192.0.2.14/24
          routes:
            - network: 0.0.0.0/0
              gateway: 192.0.2.1
          mtu: 9000
        - name: eth1
          dhcp: false
          interface: eth1
          addresses:
            - 203.0.113.14/24
          mtu: 9000
      disks:
        os: /dev/vda
    - name: eda-node05
      endpoint: "192.0.2.15"
      interfaces:
        - name: eth0
          dhcp: false
          interface: eth0
          addresses:
            - 192.0.2.15/24
          routes:
            - network: 0.0.0.0/0
              gateway: 192.0.2.1
          mtu: 9000
        - name: eth1
          dhcp: false
          interface: eth1
          addresses:
            - 203.0.113.15/24
          mtu: 9000
      disks:
        os: /dev/vda
    - name: eda-node06
      endpoint: "192.0.2.16"
      interfaces:
        - name: eth0
          dhcp: false
          interface: eth0
          addresses:
            - 192.0.2.16/24
          routes:
            - network: 0.0.0.0/0
              gateway: 192.0.2.1
          mtu: 9000
        - name: eth1
          dhcp: false
          interface: eth1
          addresses:
            - 203.0.113.16/24
          mtu: 9000
      disks:
        os: /dev/vda
k8s:
    stack: ipv4
    primaryNode: eda-node01
    endpointUrl: https://192.0.2.5:6443
    allowSchedulingOnControlPlanes : true
    control-plane:
        - eda-node01
        - eda-node02
        - eda-node03
    worker:
        - eda-node04
```

```
        - eda-node05
        - eda-node06
    vip:
        ipv4: 192.0.2.5
        interface: eth0
    env:
        http_proxy: http://192.0.2.254:8080
        https_proxy: http://192.0.2.254:8080
        no_proxy: 192.0.2.0/24,203.0.113.0/24,.domain.tld,172.22.0.0/
16,localhost,127.0.0.1,10.0.1.0/24,0.0.0.0,169.254.116.108
    time:
        disabled: false
        servers:
            - 192.0.2.253
            - 192.0.2.254
    nameservers:
        servers:
            - 192.0.2.253
            - 192.0.2.254
```

3.4.2 Generating the Talos machine configurations

About this task

After creating the EDAADM configuration file, the next step is to generate all the configuration files that are necessary to deploy the Kubernetes environment using Talos.

Procedure

Use the EDAADM tool to generate the deployment files.

Example

```
$ edaadm generate -c eda-input-6-node.yaml
ConfigFile is eda-input-6-node.yaml
...
[1/4] Validating Machines
[1/4] Validated Machines
[2/4] Validating PrimaryNode
[2/4] Validated PrimaryNode
[3/4] Validating Endpoint URL
[3/4] Validated Endpoint URL
[4/4] Validating Virtual IP
[4/4] Validated Virtual IP
[  OK  ] Spec is validated
Generating secrets for eda-compute-cluster
Created eda-compute-cluster/secrets.yaml
generating PKI and tokens
Created eda-compute-cluster/eda-node01.yaml
Created eda-compute-cluster/talosconfig.yaml
generating PKI and tokens
Created eda-compute-cluster/eda-node02.yaml
generating PKI and tokens
Created eda-compute-cluster/eda-node03.yaml
generating PKI and tokens
Created eda-compute-cluster/eda-node04.yaml
generating PKI and tokens
Created eda-compute-cluster/eda-node05.yaml
generating PKI and tokens
Created eda-compute-cluster/eda-node06.yaml
```

The configuration files created by the EDAADM tool are used in the next steps when you deploy the virtual machines. EDAADM creates a folder with the name of your cluster to store these files in. The output shows where the files are located.

Note: Nokia strongly recommends that you store these files securely and keep a backup.

3.4.3 Deploying the Talos virtual machines

This section provides the procedures for deploying an EDA node as a virtual machine on KVM or VMware vSphere.

3.4.3.1 Creating the VM on bridged networks on KVM

About this task

Complete the following steps to deploy an EDA node as a virtual machine on KVM. These steps are executed on the RedHat Enterprise Linux or Rocky Linux hypervisor directly. The steps below assume the deployment of the eda-node01 virtual machine as per the above configuration file. Ensure that you use the correct machine configuration file generated by the EDAADM tool.

Note: This procedure expects two networks to be available on the KVM hypervisors. The OAM network is referred to as br0 and the fabric management network is referred to as br1. Both of these networks are standard Linux bridge networks. If you use only one interface, adapt Step 7 to only use the br0 network.

Procedure

Step 1. Ensure that the **virt-install** tool is installed on the KVM hypervisor.

If you need to install the tool, use the following command:

```
yum install virt-install
```

Step 2. Verify that the ISO image downloaded in Downloading the KVM image is available on the hypervisor.

Step 3. Copy the machine configuration file generated for this specific node to a file called `user-data`.

Example

```
cp eda-node01.yaml user-data
```

Step 4. Create a file called `meta-data` for the node.

Use the appropriate `instance-id` and `local-hostname` values.

Example

```
instance-id: eda-node01
local-hostname: eda-node01
```

Step 5. Create a file called `network-config` for the node.

Example

The file should have the following content:

```
version: 2
```

Step 6. Create an ISO file containing the newly created files.

For ease of use, name the ISO file with the name of the node for which you are creating the ISO.

Example

```
mkisofs -o eda-node01-data.iso -V cidata -J -r meta-data network-config user-data
```

Step 7. Create the virtual machine.

This step uses both the newly created ISO file and the ISO file downloaded from the Talos Machine Factory.

Example

```
virt-install -n eda-node01 \
  --description "Talos 1.8.3 vm for node eda-node01" \
  --noautoconsole --os-type=generic \
  --memory 65536 --vcpus 32 --cpu host \
  --disk eda-node01-rootdisk.qcow2,format=qcow2,bus=virtio,size=100 \
  --disk eda-node01-storagedisk.qcow2,format=qcow2,bus=virtio,size=300 \
  --cdrom nocloud-amd64.iso \
  --disk eda-node01-data.iso,device=cdrom \
  --network bridge=br0,model=virtio \
  --network bridge=br1,model=virtio
```

Note: If the node is not a storage node, you can remove the second `--disk` line.

3.4.3.2 Creating the VM on bridged networks on VMware vSphere

About this task

Complete the following steps to deploy an EDA node as a virtual machine on VMware vSphere. The steps below assume the deployment of the eda-node01 virtual machine as per the above configuration file. Ensure that you are using the correct machine configuration file generated by the EDAADM tool.
You can use one of the following methods to deploy the VM on VMware vSphere:

- the VMware vSphere vCenter or ESXi UI

  For instructions, see *Deploy an OVF or OVA Template* in the VMware vSphere documentation.

- the VMware Open Virtualization Format Tool CLI

  This procedure provides an example of how to use the VMware OVF Tool CLI.

Note: This procedure uses two networks (portgroups) to be available on the ESXi hypervisors. The OAM network is referred to as `OAM` and the fabric management network is referred to as `FABRIC`. Both of these networks can be standard PortGroups or distributed PortGroups. If you only use one network, you do not need to create a second interface on the VM.

Procedure

Step 1. Download and install the latest version of the VMware OVF Tool from the VMware Developer website.

Step 2. Display details about the OVA image.

Example

```
$ ovftool vmware-amd64.ova
OVF version:   1.0
VirtualApp:    false
Name:          talos

Download Size:  104.05 MB

Deployment Sizes:
  Flat disks:   8.00 GB
  Sparse disks: Unknown

Networks:
  Name:        VM Network
  Description: The VM Network network

Virtual Machines:
  Name:               talos
  Operating System:   other3xlinux64guest
  Virtual Hardware:
    Families:         vmx-15
    Number of CPUs:   2
    Cores per socket: automatic
    Memory:           2.00 GB

    Disks:
      Index:          0
      Instance ID:    4
      Capacity:       8.00 GB
      Disk Types:     SCSI-VirtualSCSI

    NICs:
      Adapter Type:   VmxNet3
      Connection:     VM Network

Properties:
  Key:         talos.config
  Label:       Talos config data
  Type:        string
  Description: Inline Talos config

References:
  File:  disk.vmdk
```

Step 3. Create a base64 encoded hash from the Talos machine configuration for the node.

In this example, the output is stored as an environment variable to make it easy to use in the command to deploy the image using the OVF Tool.

Example

```
export NODECONFIG=$(base64 -i eda-node01.yaml)
```

Step 4. Deploy the OVA image using the OVF Tool.

For details about command line arguments, see the OVF Tool documentation from the VMware website.

Note: If you prefer using the VMware vCenter UI to create the virtual machines, use the regular method of deploying an OVA/OVF template. In this process, in the Customize template step, when you are prompted to provide the Inline Talos config, you must provide the `base64` encoded data from the Talos machine configuration for the node. This very long string that is returned when you execute the `base64 -i eda-node01.yaml` command. Copy that long string and paste it into the field in the UI, then continue.

Example

```
$ ovftool --acceptAllEulas --noSSLVerify \
 -dm=thin \
 -ds=DATASTORE \
 -n=eda-node01 \
 --net:"VM Network=OAM" \
 --prop:talos.config="${NODECONFIG}" \
vmware-amd64.ova \
vi://administrator%40vsphere.local@vcenter.domain.tld/My-DC/host/My-Cluster/Resources/
My-Resource-Group

Opening OVA source: vmware-amd64.ova
The manifest validates
Enter login information for target vi://vcenter.domain.tld/
Username: administrator%40vsphere.local
Password: ***********
Opening VI target: vi://administrator%40vsphere.local@vcenter.domain.tld:443/My-DC/
host/My-Cluster/Resources/My-Resource-Group
Deploying to VI: vi://administrator%40vsphere.local@ vcenter.domain.tld:443/My-DC/
host/My-Cluster/Resources/My-Resource-Group
Transfer Completed
Completed successfully
```

Expected outcome

This step deploys the VM with the CPU, memory, disk, and NIC configuration of the default OVA image. The next step updates these settings.

Step 5. In vCenter, edit the VM settings.

Make the following changes:

- Increase the number of vCPU to 32.
- Increase the memory to 64G.
- Increase the main disk size to 100G. On boot, Talos automatically extends the file system.
- Optionally, if this VM is a storage node, add a new disk with a size of 300G.
- Optionally, add a second network interface and connect it to the `FABRIC PortGroup`.
- Enable 100% resource reservation for the CPU, memory and disk.

Step 6. Power on the virtual machine.

3.5 Bootstrap the Talos Kubernetes cluster

When all the virtual machines are deployed and running, you can set up the Kubernetes cluster on the virtual machines using Talos.

Note: The procedures in this chapter use the **edadm** command. Ensure that the command is available, as well as the original EDAADM configuration file from which you generated Talos files.

3.5.1 Bootstrapping Kubernetes on the primary node

About this task

After booting the Talos VMs, you can now bootstrap the Kubernetes cluster using the **talosconfig** command. The following parameter is relevant for this procedure:

`-c`: Specifies the **edaadm** tool.

Procedure

Execute the following command in the folder where the `eda-input-6-node.yaml` is stored, as well as the folder where the Talos configuration files were executed:

```
edaadm bootstrap-k8s -c eda-input-6-node.yaml
```

Expected outcome

Wait for several minutes for the Kubernetes cluster to come up and for all the nodes join the cluster. The process should take less than 15 minutes.

3.5.2 Obtaining the Kubernetes config file for kubectl

About this task

Use the **edaadm** command to obtain the Kubernetes configuration file for use with kubectl. The following parameter is relevant for this procedure:

`-c`: Specifies your EDAADM configuration file.

Procedure

Step 1. Obtain the Kubernetes configuration file.

Execute the following command in the folder where the `eda-input-6-node.yaml` is stored, as well as where the folder with the Talos configuration files was executed:

Example

```
edaadm get-kubeconfig -c eda-6-node-deployment.yaml
```

Expected outcome

The command creates a `kubeconfig` file in the folder with the name of your cluster according to the EDAADM config file.

Step 2. You can configure your environment to use the `kubeconfig` file for use with the **kubectl** command.

Example

```
export KUBECONFIG=eda-compute-cluster/kubeconfig
```

Step 3. Inspect your server and check if all nodes are up and running.

You can use the typical `kubectl` commands.

Example

```
kubectl get nodes
```

Expected outcome

When all the nodes are up and Kubernetes is stable, continue with Setting up the Rook Ceph storage cluster.

3.5.3 Setting up the Rook Ceph storage cluster

About this task

EDA uses Rook Ceph as a secure, distributed, and redundant data store for all the data it stores. Using Ceph guarantees redundancy and high availability of all data by providing multiple copies of all data. The following steps guide you through the configuration and deployment of Rook Ceph.

Note: To ensure that you deploy Rook Ceph on the correct Kubernetes cluster, verify that you are using the `kubeconfig` file created through the **edaadm** tool .

Procedure

Step 1. Add the Rook Ceph Helm chart.

Example

```
helm repo add rook-release https://charts.rook.io/release
```

Step 2. Using the `rook-ceph-operator-values.yaml` file that `edaadm` generated based on the configuration, deploy the Rook Ceph Operator.

Example

```
helm install --create-namespace \
  --namespace rook-ceph \
  -f path/to/rook-ceph-operator-values.yaml \
  rook-ceph rook-release/rook-ceph
```

Step 3. Using the `rook-ceph-cluster-values.yaml` file that the `edaadm` tool generated, deploy the Rook Ceph Cluster.

Example

```
helm install \
  --namespace rook-ceph \
  --set operatorNamespace=rook-ceph \
  -f path/to/rook-ceph-cluster-values.yaml \
```

```
  rook-ceph-cluster rook-release/rook-ceph-cluster
```

Expected outcome

The output from this command can report missing CRDs; wait until the Rook Ceph Operator is running in the Kubernetes cluster.

Step 4. Using **kubectl** commands, verify that the operator is deployed and the necessary pods are deployed before installing the EDA application.

Example

This example is for a six-node cluster, with six storage nodes.

```
$ kubectl -n rook-ceph get pods
NAME                                                READY   STATUS      RESTARTS
 AGE
csi-cephfsplugin-22rmj                              2/2     Running     1 (6m32s ago)
 7m6s
csi-cephfsplugin-25p9d                              2/2     Running     1 (6m30s ago)
 7m6s
csi-cephfsplugin-2gr8v                              2/2     Running     4 (5m16s ago)
 7m6s
csi-cephfsplugin-48cwk                              2/2     Running     1 (6m30s ago)
 7m6s
csi-cephfsplugin-fknch                              2/2     Running     2 (5m32s ago)
 7m6s
csi-cephfsplugin-provisioner-67c8454ddd-mpq4w       5/5     Running     1 (6m1s ago)
 7m6s
csi-cephfsplugin-provisioner-67c8454ddd-qmdrq       5/5     Running     1 (6m18s ago)
 7m6s
csi-cephfsplugin-vfxnf                              2/2     Running     1 (6m32s ago)
 7m6s
rook-ceph-mds-ceph-filesystem-a-7c54cdf5bc-lmf6n    1/1     Running     0
 2m40s
rook-ceph-mds-ceph-filesystem-b-6dc794b9f4-2lc64    1/1     Running     0
 2m37s
rook-ceph-mgr-a-55b449c844-wpps8                    2/2     Running     0
 4m30s
rook-ceph-mgr-b-5f97fd5746-fzngx                    2/2     Running     0
 4m30s
rook-ceph-mon-a-76fcb96c4c-vscnc                    1/1     Running     0
 5m53s
rook-ceph-mon-b-68bf5974bb-p2vnj                    1/1     Running     0
 4m57s
rook-ceph-mon-c-6d7c64dcb6-phs99                    1/1     Running     0
 4m47s
rook-ceph-operator-5f4c4bff8d-2fsq2                 1/1     Running     0
 7m54s
rook-ceph-osd-0-bf89f779-zh4kd                      1/1     Running     0
 3m49s
rook-ceph-osd-1-64dcd64c5f-7xcbm                    1/1     Running     0
 3m49s
rook-ceph-osd-2-54ddd95489-5qkdt                    1/1     Running     0
 3m49s
rook-ceph-osd-3-56cbd54bd6-7mt8w                    1/1     Running     0
 3m39s
rook-ceph-osd-4-567dcff476-wljll                    1/1     Running     0
 2m56s
rook-ceph-osd-5-6f69c998b6-2l5wp                    1/1     Running     0
 2m54s
rook-ceph-osd-prepare-eda-dev-node01-7rfkn          0/1     Completed   0
 4m8s
```

```
rook-ceph-osd-prepare-eda-dev-node02-rqdkx         0/1     Completed   0
 4m8s
rook-ceph-osd-prepare-eda-dev-node03-xtznb         0/1     Completed   0
 4m8s
rook-ceph-osd-prepare-eda-dev-node04-db4v8         0/1     Completed   0
 4m7s
rook-ceph-osd-prepare-eda-dev-node05-29wwm         0/1     Completed   0
 4m7s
rook-ceph-osd-prepare-eda-dev-node06-zxp2x         0/1     Completed   0
 4m7s
rook-ceph-tools-b9d78b5d4-8r62p                    1/1     Running     0
 7m6s
```

Note: Some of the pods may restart as they initiate Ceph. This behavior is expected.

3.6 Installing the EDA application

After setting up Kubernetes on the VMs, you can now install Nokia EDA using the playground.

3.6.1 Customizing the installation file

Update the settings for the parameters in the `prefs.mk` file to control the way EDA is installed.

Table 6: Customizable parameters in the `prefs.mk` file

Parameter	Description
`NO_KIND`	Specifies not to deploy the lab KIND node. Must be set to `1`.
`METALLB_VIP`	Specifies the VIP address of your EDA deployment. Make sure to use a CIDR format, preferably as a /32 (or /128 for an IPv6 VIP). If you use two networks, this VIP address must be the one used on the fabric management network. If you use a single network, this setting must match the VIP address used for `EXT_DOMAIN_NAME` FQDN or IP. Example: `203.0.113.10/32`
`EXT_DOMAIN_NAME`	The FQDN that resolves to the EDA VIP or the VIP itself. This value must be the FQDN or VIP address that is used to access the UI. If you use two networks, this value must be the FQDN or IP address of the OAM network.
`EXT_HTTP_PORT`	The HTTP port that the EDA UI/API should use to redirect to HTTPS. Set to 80.
`EXT_HTTPS_PORT`	The HTTPS port on which the EDA UI/API listens. Set to 443.

Parameter	Description
EXT_IPV4_ADDR	The IPv4 IP address used as the VIP address. If you use two networks, this VIP address must be the one used on the fabric management network. If you use a single network, this VIP address must be the VIP that matches your EXT_DOMAIN_NAME FQDN (or IP address).
EXT_IPV6_ADDR	The IPv6 IP address used as the VIP. If you use two networks, this VIP address must be the one used on the fabric management network. If you use a single network, this VIP address must be the VIP that matches your EXT_DOMAIN_NAME FQDN (or IP address).
HTTPS_PROXY	Optional: The proxy address for the HTTPS proxy.
HTTP_PROXY	Optional: The proxy address for the HTTP proxy.
NO_PROXY	Optional: The list of IP addresses, IP ranges, and hostnames that should not be proxied.
https_proxy	Optional: The proxy address for the HTTPS proxy.
http_proxy	Optional: The proxy address for the HTTP proxy.
no_proxy	Optional: The list of IP addresses, IP ranges and hostnames that should not be proxied.
LLM_API_KEY	Optional: The OpenAI API key for the LLM functionality.
SINGLESTACK_SVCS	Optional: Indicates that internal services should be single stack instead of dual stack, if Kubernetes is dual stack. Boolean.
SIMULATE	Specifies if the EDA deployment is to manage simulated workloads (Digital Sandbox) or real hardware. Values: `true` or `false`. By default, this parameter is set to `true` if the parameter is not provided in the file.
KPT_SETTERS_FILE	Advanced configuration file for kpt.

3.6.1.1 The prefs.mk file

Example: Settings for configurable parameters in the `prefs.mk` file

Note: The settings for METALLB_VIP and EXT_IPV4_ADDR match the fabric management network.

```
NO_KIND=1
METALLB_VIP=203.0.113.10/32
EXT_DOMAIN_NAME=eda.domain.tld
EXT_HTTP_PORT=80
EXT_HTTPS_PORT=443
EXT_IPV4_ADDR=203.0.113.10
```

```
EXT_IPV6_ADDR=""
HTTPS_PROXY=http://192.0.2.254:8080
HTTP_PROXY=http://192.0.2.254:8080
NO_PROXY=192.0.2.0/24,203.0.113.0/24,.domain.tld,172.22.0.0/
16,localhost,127.0.0.1,10.0.1.0/24,0.0.0.0,169.254.116.108
https_proxy=http://192.0.2.254:8080
http_proxy=http://192.0.2.254:8080
no_proxy=192.0.2.0/24,203.0.113.0/24,.domain.tld,172.22.0.0/
16,localhost,127.0.0.1,10.0.1.0/24,0.0.0.0,169.254.116.108
LLM_API_KEY=...
```

3.6.2 Installing Nokia EDA

About this task

Follow these steps to install EDA.

Procedure

Step 1. Download the latest tools.

Example

```
make download-tools
```

Step 2. Download the latest packages, including the eda-kpt package.

Example

```
make download-pkgs
```

Step 3. Set up the MetalLB environment for VIP management.

Example

```
make metallb
```

Step 4. Install the necessary external packages.

Example

```
make install-external-packages
```

Note: If this command exits with an error, wait 30 seconds and try again. Sometimes Kubernetes is a bit slower in reconciling the change than the command waits for.

Step 5. Change the eda-git Kubernetes service to a ClusterIP service instead of a LoadBalancer type.

Example

```
kubectl -n eda-system patch service eda-git -p '{"spec": {"type": "ClusterIP"}}'
```

Step 6. Generate the EDA core configuration.

Example

```
make eda-configure-core
```

Step 7. Install EDA core components.

Example

```
make eda-install-core
```

Note: If the command hangs for a long time (>5 minutes) on "reconcile pending" for a workflow definition, cancel the command and try again, KPT is designed to handle these cases. This can occasionally happen depending on the Kubernetes cluster

Step 8. Verify that the EDA Config Engine is up and running.

Example

```
make eda-is-core-ready
```

Step 9. Install all the standard EDA apps.

This step can take approximate 5 to 15 minutes, depending on your connectivity.

Example

```
make eda-install-apps
```

Step 10. Bootstrap EDA.

Example

```
make eda-bootstrap
```

Step 11. If your deployment uses two networks, create a second VIP pool for the OAM VIP address.

Example

```
make metallb-configure-pools METALLB_VIP=<OAM VIP> LB_POOL_NAME=pool-nb
```

Step 12. If your deployment uses two networks, create the OAM UI/API service using the new VIP pool.

Example

```
make eda-create-api-lb-svc API_LB_POOL_NAME=pool-nb
```

Step 13. Optional: Deploy an example topology.

Example

```
make topology-load
```

3.6.3 Accessing the EDA deployment

Procedure

You can now access the new EDA deployment using the following methods:

- use `https://OAM-VIP`
- if an FQDN is configured for the `EXT_DOMAIN_NAME` field, use `https://FQDN`

4 Air-gapped installations

In scenarios where the environment in which EDA is being deployed does not have any connectivity to the Internet, a local mirror with all the resources EDA needs, must be deployed.

Air-gapped environments

In an air-gapped environment, an Assets VM is deployed to provide the services that serve the container images, git repositories and artifacts used during installation of the EDA Talos Kubernetes cluster and EDA itself.

The goal of the air-gapped solution design is to allow flexibility in the deployment and content of the Assets VM in the air-gapped environment. By providing a standalone Assets VM without any assets automatically included, there is freedom of choice of what assets are uploaded to the Assets VM. It allows for a single Assets VM to be used for multiple deployments and versions of EDA, as the assets for multiple versions of EDA can be uploaded to the same Assets VM.

Similarly, by splitting up the assets in bundles, it is possible to only upload specific content to the Assets VM. The bundle concept also allows for the creation of custom bundles, such as for third-party apps, so they can also be hosted on the Assets VM.

Public and air-gapped systems

In the context of EDA installation in this section, procedures are executed in a public or air-gapped environment.

- public environment - an environment that has Internet access. You use a system with Internet access to create the Assets VM image and to download all the necessary assets and tools.
- air-gapped environment - the system that does not have Internet access. This is the environment in which EDA is deployed.

For each environment, you must have a system from where you can execute the steps. This system can be the same system that you used to first connect to the Internet, follow the steps for the public network, and then move the system to the air-gapped environment to continue.

You can also have two systems, where you copy the data from the public system to the air-gapped system.

Related topics

Requirements

4.1 Components

Talos Linux and Kubernetes	EDA uses Talos Linux and Kubernetes to host its services in. Talos Linux is a minimalistic, locked-down, read-only and secured Linux environment in which Kubernetes is run. This assures a more secure environment with significantly lower security footprint than regular Linux and Kubernetes environments.
`edaadm`	The **edaadm** tool is used for the following steps in the installation process:

	• getting the location to download the base Talos image for KVM and VMware environments. • generating the Talos machine configuration files for the deployment of both the Assets VM and the EDA Kubernetes cluster VMs. • initiating Talos Kubernetes clusters.
edaadm git repository	A publicly available git repository that contains details and definitions for: • the assets bundles: The EDA Assets are defined in different bundles, based on their purpose. The repository provides the bundles, and has a way to download the content of the bundles from the Internet and then upload them to the deployed Assets VM. • the KPT Package: A KPT package to initiate the Assets VM.
Assets VM	The Assets VM is a VM deployed on a KVM or VMware environment. It is a single VM K8s cluster that runs: • a container registry to host all the container images used by EDA • a git server to host the App Store Catalog • a web server to host some artifacts used by EDA
Bundles	A bundle is a definition of a group of assets that are related, for instance, a bundle for the core components of EDA for a specific version, or a bundle of the standard Apps for a specific version. Bundles are downloaded using the `edaadm` tool from the Internet, and then uploaded using `edaadm` to the Assets VM. The product comes with a set of standard bundles and custom bundles can be created based on their examples.
EDA Shipyard	A name used to describe the combination of the container registry, git server, and web server running on the Assets VM.
Playground git repository	The Playground git repository is publicly available and is used to deploy EDA itself.

4.2 Overview of the installation process in air-gapped environments

The installation in air-gapped environments consists of the following high-level tasks:

1. Downloading extra tools and the KPT packages
2. Create the Assets VM image
3. Preparing the air-gapped environment
4. Deploying the Assets VM
5. Bootstrap the Assets VM
6. Updating the EDAADM file for the EDA Kubernetes cluster
7. Bootstrapping Kubernetes on the Assets VM
8. Bootstrapping the EDA Talos - air-gapped environment
9. Installing the EDA application in an air-gapped environment

4.3 Requirements

Ensure that you satisfy the requirements for air-gapped deployments:

- Environment and tools
- Repositories
- Assets VM

4.3.1 Environment and tools

The following system requirements apply to the public and air-gapped environment. In this section, this system is called the public or air-gapped tools-system. This Linux system must have the following commands and tools:

- `docker` - A docker environment needs to be running as it is used to update KPT configuration using the `kpt-apply-setters` image.
- `curl` - Used to download files.
- `git` - Used to check out git repositories.
- `jq` - Used to parse JSON data.
- `sed` - Used to parse and replace content.
- `tar` and `zip` - Used to create and unpack bundles and assets.
- `edaadm` - Used to generate configuration for Talos and other useful commands to initiate the Talos environments. It can be downloaded from the `edaadm` repository releases page.
- `htpasswd` - (Optional) Used in case a custom username and password is required for the Assets VM web server.
- `base64` - (Optional) Used in case a custom username and password is required for the Assets VM web server or git server.
- `ovftool` - (Optional) Used to deploy the VMs in a VMware vSphere environment. Can be downloaded from the Broadcom Developer Portal

4.3.2 Repositories

Clone, copy or download, and unpack the contents of the following repositories to the both public and air-gapped systems:

- `edaadm` repository
- Playground repository

4.3.2.1 Downloading extra tools and the KPT packages

About this task

Complete this procedure in the public environment. Then, copy the content to the air-gapped environment.

Procedure

Step 1. Download the playground tools and packages.

In the Playground repository, run the following commands to download additional tools and the KPT packages needed for the EDA installation.

```
make download-tools
make download-pkgs
```

Step 2. In the `edaadm` repository that you cloned or downloaded, go to the `bundles` folder.

```
cd path/to/edaadm-repository/bundles
```

Step 3. Download the tools for the bundles.

```
make download-tools
```

Step 4. In the `edaadm` repository that you cloned or downloaded, go to the `kpt` folder.

```
cd path/to/edaadm-repository/kpt
```

Step 5. Download the tools for the kpt package.

```
makedownload-tools
```

4.3.3 Assets VM

The Assets VM runs as a single VM inside the air-gapped environment. This VM holds all of the assets and can be used across multiple deployments and EDA versions, containing the assets for multiple versions. This VM has the following requirements:

- CPU: 4 vCPUs on a modern x86-64 CPU that supports virtualization
- Memory: 16GB RAM
- Storage: 300GB of storage for the main disk
- Networking:
  - 1GbE interface
  - 1 IPv4 IP and optionally 1 IPv6 IP
  - Preferably in the same OAM network as the EDA Kubernetes VMs, but minimally accessible by the EDA Kubernetes VMs via the OAM network

4.3.4 Create the Assets VM image

The creation of the Assets VM starts from a base Talos VM image for KVM or VMware. Then, you rebuild it with the local cache needed to deploy the VM, Kubernetes, and the Assets VM services in the air-gapped environment.

4.3.4.1 Preparing to create the assets

About this task

Before creating the Assets VM image for a specific environment, create an image cache for the Assets VM.

Procedure

Step 1. In the `edaadm` repository that you cloned or downloaded, go to the `bundles` folder.

```
cd path/to/edaadm-repository/bundles
```

Step 2. Log in to the `ghcr.io` registry with docker so the system can pull private images from `ghcr.io`.

Log in with a user account that has access to images hosted by Nokia EDA, for example, the nokia-eda-bot user.

```
docker login ghcr.io -u nokia-eda-bot
```

Note: The token (password) for the nokia-eda-bot user is present in every bundle file in the `edaadm` repository, where it is twice encoded using base64. This token is a read-only token and is not a secret; no sensitive information is accessible using this token.

Step 3. Prepare the image cache for the Assets VM.

This step downloads and prepares an image cache from where the Assets VM is built.

```
make create-assets-host-bootstrap-image-cache
```

What to do next

Create the Assets VM image for your deployment scenario:

- Creating the KVM Assets VM image
- Creating the VMware Assets VM on VMware vSphere

4.3.4.2 Creating the KVM Assets VM image

About this task

Follow these steps to create the Assets VM image for KVM. This procedure generates an ISO file based on the Talos VM base image containing a local cache. This image is different from the base Talos image ISO file that you use for the EDA Kubernetes VMs, but is based on it.

Procedure

Step 1. In the `edaadm` repository that you cloned or downloaded, go to the `bundles` folder.

```
cd path/to/edaadm-repository/bundles
```

Step 2. Generate the Assets VM ISO for KVM .

The following command to generates the KVM Talos ISO for the Assets VM

```
make create-asset-vm-nocloud-boot-iso
```

Example

The output should be similar to the following:

```
--> INFO: List of goals: create-asset-vm-nocloud-boot-iso
docker pull ghcr.io/siderolabs/imager:v1.9.2
v1.9.2: Pulling from siderolabs/imager
Digest: sha256:b99d29d04df9eea89d50cb0d13d57e1e035e54cbd9970a26af99b18154e443a9
Status: Image is up to date for ghcr.io/siderolabs/imager:v1.9.2
ghcr.io/siderolabs/imager:v1.9.2
skipped pulling overlay (no overlay)
profile ready:
arch: amd64
platform: nocloud
secureboot: false
version: v1.9.2
input:
  kernel:
    path: /usr/install/amd64/vmlinuz
  initramfs:
    path: /usr/install/amd64/initramfs.xz
  baseInstaller:
    imageRef: ghcr.io/siderolabs/installer:v1.9.2
  imageCache:
    imageRef: ""
    ociPath: /image-cache.oci
output:
  kind: iso
  imageOptions:
    diskSize: 2147483648
  outFormat: raw
skipped initramfs rebuild (no system extensions)
kernel command line: talos.platform=nocloud console=tty1 console=ttyS0 net.ifnames=
0 talos.halt_if_installed=1 init_on_alloc=1 slab_nomerge pti=on consoleblank=0 nvme_
core.io_timeout=4294967295 printk.devkmsg=on ima_template=ima-ng ima_appraise=fix ima_
hash=sha512
ISO ready
output asset path: /out/nocloud-amd64.iso
```

Step 3. Rename the KVM Assets VM image.

Rename the generated image to a convenient name so that you can copy or use it in the future.

```
mv eda-cargo/talos-asset-vm-boot-imgs/nocloud-amd64.iso eda-cargo/talos-asset-vm-boot-
imgs/eda-asset-vm-nocloud-amd64.iso
```

4.3.4.3 Creating the VMware Assets VM on VMware vSphere

About this task

Follow these steps to create the Assets VM Image for VMware vSphere. This procedure generates an ISO file based on the Talos VM base image containing a local cache. This image is different from the base Talos image ISO file used for the EDA Kubernetes VMs, but is based on it.

Note: This procedure is needed if you plan to deploy the Assets VM on VMware vSphere

Procedure

Step 1. In the `edaadm` repository that you cloned or downloaded, go to the `bundles` folder.

```
cd path/to/edaadm-repository/bundles
```

Step 2. Generate the Assets VM OVA for VMware vSphere.

The following command generates the VMware vSphere Talos OVA for the Assets VM.

```
make create-asset-vm-vmware-boot-ova
```

Example

The output should be similar to the following:

```
--> INFO: List of goals: create-asset-vm-vmware-boot-ova
docker pull ghcr.io/siderolabs/imager:v1.9.2
v1.9.2: Pulling from siderolabs/imager
Digest: sha256:b99d29d04df9eea89d50cb0d13d57e1e035e54cbd9970a26af99b18154e443a9
Status: Image is up to date for ghcr.io/siderolabs/imager:v1.9.2
ghcr.io/siderolabs/imager:v1.9.2
skipped pulling overlay (no overlay)
profile ready:
arch: amd64
platform: vmware
secureboot: false
version: v1.9.2
input:
  kernel:
    path: /usr/install/amd64/vmlinuz
  initramfs:
    path: /usr/install/amd64/initramfs.xz
  baseInstaller:
    imageRef: ghcr.io/siderolabs/installer:v1.9.2
  imageCache:
    imageRef: ""
    ociPath: /image-cache.oci
output:
  kind: image
  imageOptions:
    diskSize: 2147483648
    diskFormat: ova
  outFormat: raw
skipped initramfs rebuild (no system extensions)
kernel command line: talos.platform=vmware talos.config=guestinfo console=tty0
 console=ttyS0 earlyprintk=ttyS0,115200 net.ifnames=0 init_on_alloc=1 slab_nomerge
 pti=on consoleblank=0 nvme_core.io_timeout=4294967295 printk.devkmsg=on ima_template=
ima-ng ima_appraise=fix ima_hash=sha512
disk image ready
output asset path: /out/vmware-amd64.ova
```

Step 3. Rename the VMware vSphere Assets VM image.

Rename the generated image to a convenient name so that you can copy or use it in the future.

```
mv eda-cargo/talos-asset-vm-boot-imgs/vmware-amd64.ova eda-cargo/talos-asset-vm-boot-
imgs/eda-asset-vm-vmware-amd64.ova
```

4.4 Downloading the Assets Bundles

Procedure

Step 1. In the `edaadm` repository that you cloned or downloaded, go to the `bundles` folder.

```
cd path/to/edaadm-repository/bundles
```

Step 2. Download the Assets bundles.

Example

The following example downloads all Assets Bundles defined in the bundles folder and store them in the `eda-cargo` directory.

```
make save-all-bundles
```

Example

The following examples show how to download a specific bundle.
Use the following command to display all bundles:

```
make ls-bundles
```

Then, use the following command to download a specific bundle:

```
make save-<bundle-name>
```

4.5 Downloading the Base Talos VM

About this task

To deploy the EDA Kubernetes VMs, you need the base Talos image for KVM or VMware vSphere or you can use the `edaadm` bundles folder as described below.

Note: This procedure applies to the public environment, and is executed in the public tools-system.

- In the `edaadm` repository that you cloned or downloaded, go to the `bundles` folder.

  ```
  cd path/to/edaadm-repository/bundles
  ```

- Download the base Talos images.

  ```
  make download-talos-stock-boot-media
  ```

  Expected outcome

  The output should be similar to the following:

  ```
  --> INFO: List of goals: download-talos-stock-boot-media
  --> Downloading boot media for vmware
  ```

```
    From: https://factory.talos.dev/image/
903b2da78f99adef03cbbd4df6714563823f63218508800751560d3bc3557e40/v1.9.2/vmware-amd64.iso
    To: /path/to/edaadm-repository/bundles/eda-cargo/talos-stock-boot-media/vmware-amd64.iso
######################################################################################################################
 100.0%
    From: https://factory.talos.dev/image/
903b2da78f99adef03cbbd4df6714563823f63218508800751560d3bc3557e40/v1.9.2/vmware-amd64.ova
    To: /path/to/edaadm-repository/bundles/eda-cargo/talos-stock-boot-media/vmware-amd64.ova
######################################################################################################################
 100.0%
--> Downloading boot media for nocloud
    From: https://factory.talos.dev/image/
376567988ad370138ad8b2698212367b8edcb69b5fd68c80be1f2ec7d603b4ba/v1.9.2/nocloud-amd64.iso
    To: /path/to/edaadm-repository/bundles/eda-cargo/talos-stock-boot-media/nocloud-
amd64.iso
######################################################################################################################
 100.0%
    From: https://factory.talos.dev/image/
376567988ad370138ad8b2698212367b8edcb69b5fd68c80be1f2ec7d603b4ba/v1.9.2/nocloud-amd64.raw.xz
    To: /path/to/edaadm-repository/bundles/eda-cargo/talos-stock-boot-media/nocloud-
amd64.raw.xz
######################################################################################################################
 100.0%
--> Downloading boot media for metal
    From: https://factory.talos.dev/image/
376567988ad370138ad8b2698212367b8edcb69b5fd68c80be1f2ec7d603b4ba/v1.9.2/metal-amd64.iso
    To: /path/to/edaadm-repository/bundles/eda-cargo/talos-stock-boot-media/metal-amd64.iso
######################################################################################################################
 100.0%
    From: https://factory.talos.dev/image/
376567988ad370138ad8b2698212367b8edcb69b5fd68c80be1f2ec7d603b4ba/v1.9.2/metal-amd64.raw.zst
    To: /path/to/edaadm-repository/bundles/eda-cargo/talos-stock-boot-media/metal-
amd64.raw.zst
######################################################################################################################
 100.0%
    From: https://factory.talos.dev/image/
376567988ad370138ad8b2698212367b8edcb69b5fd68c80be1f2ec7d603b4ba/v1.9.2/metal-amd64.qcow2
    To: /path/to/edaadm-repository/bundles/eda-cargo/talos-stock-boot-media/metal-
amd64.qcow2
######################################################################################################################
 100.0%
```

4.6 Preparing the air-gapped environment

Procedure

After downloading all the tools, packages, repositories, bundles and images, use one of the following options to make the data available in the air-gapped environment:

- Move the public tools-system to the air-gapped environment
  For example, for a laptop or a VM, move to the air-gapped environment by changing its network configuration.
- Copy the data from the public tools-system to the air-gapped tools-system using a USB key or a temporary network connection.
  The data should include:
  - the playground repository - includes the tools and standard installation KPT packages

– the edaadm repository - includes the bundles folder that holds the `eda-cargo` folder that has all the air-gapped data (bundles, asset VM image, and Talos base VM images)

4.6.1 Loading the KPT Setters image

About this task

The procedures for setting up the Assets VM and installing EDA use KPT. For both tasks, you may need to configure some setting in the KPT packages. KPT uses a container called `kpt-apply-setters` for this purpose. This image must be present in the local Docker image cache of the air-gapped tools-system.

The container image is part of the `eda-bundle-tools` bundle in the `edaadm/bundles` directory. If you used the `save-all-bundles` option when you downloaded the bundles, that bundle should be present in your air-gapped tools-system. If it is not present, you can download the bundle on the public tools-system and copy over the content of the bundle to the air-gapped tools-system before executing these steps.

This procedure loads the `kpt-apply-setters` image from the `eda-bundle-tools` bundle.

Note: This procedure applies to the air-gapped environment, and is executed in the air-gapped tools-system.

Procedure

Step 1. In the `edaadm` repository that you cloned or downloaded, go to the `bundles` folder.

```
cd path/to/edaadm-repository/bundles
```

Step 2. Import the image into the local docker image cache.

```
docker load -i eda-cargo/eda-bundle-tools-1-0-0/images/srl-labs-kpt-apply-setters-0-1-
1
```

Note: The version of the bundle may update to a newer version in the future.

4.7 Deploying the Assets VM

Deploying the Assets VM is very similar to deploying an EDA Kubernetes cluster.

The high level steps are:

- Preparing the Assets VM
- Generating the Talos machine config files
- Deploy the Assets VM
- Bootstrap the Assets VM
- Deploying the Assets VM services

4.7.1 Preparing the Assets VM

The EDAADM configuration file for the Assets VM is very similar to the EDAADM configuration file of an EDA Kubernetes environment. EDAAM configuration file fields shows the contents of the configuration files.

Be aware of the following requirements for the Assets VM:

- The Assets VM EDAADM configuration file is a config file for a single machine.
- The value for the `clusterName` field must be unique and different from the EDA Kubernetes cluster.
- The `machines` context must include the following entries:

```
enableImageCache: true
localPathProvisioner: "/var/local-path-provisioner"
```

- The Assets VM only needs one network interface, preferably on the OAM network of the EDA Kubernetes cluster. This network interface should reachable from the OAM network of the EDA Kubernetes cluster.
- The `edaadm` tool still expects the definition of a storage disk in the machine definition, but this can be a reference to a non-existing disk.

4.7.1.1 Example Assets VM EDAADM

In the following configuration file, the Assets VM uses local DNS and NTP servers.

Example

```
version: 24.12.2
clusterName: eda-airgap-assets
machines:
    - name: eda-assets.domain.tld
      endpoint: 192.0.2.228
      enableImageCache: true
      localPathProvisioner: "/var/local-path-provisioner"
      interfaces:
        - name: eth0
          dhcp: false
          interface: eth0
          addresses:
            - 192.0.2.228/23
          routes:
            - network: 0.0.0.0/0
              gateway: 192.0.2.1
          mtu: 9000
      disks:
        os: /dev/vda
        storage: /dev/vdb
k8s:
    stack: ipv4
    primaryNode: eda-assets.domain.tls
    endpointUrl: https://192.0.2.228:6443
    allowSchedulingOnControlPlanes: true
    control-plane:
        - eda-assets.domain.tld
    time:
        disabled: false
```

```
        servers:
            - 192.0.2.253
            - 192.0.2.254
    nameservers:
        servers:
            - 192.0.2.254
            - 192.0.2.253
```

4.7.2 Generating the Talos machine config files

About this task

After creating the Assets VM EDAADM configuration file, the next step is to generate all the necessary configuration files to deploy the Kubernetes environment using Talos.

Procedure

Use the `edaadm` tool to generate the Talos configuration out of the EDAADM configuration file:

```
edaadm generate -c eda-assets-deployment.yaml
```

The output should be similar to the following (portions removed)

```
ConfigFile is eda-assets-deployment.yaml
...
[1/5] Validating Machines
[1/5] Validated Machines
[2/5] Validating Primary Node
[2/5] Validated Primary Node
[3/5] Validating Endpoint URL
[3/5] Validated Endpoint URL
[4/5] Validating Virtual IP
[4/5] Validated Virtual IP
[5/5] Validating Storage
[5/5] Validated Storage
[  OK  ] Spec is validated
Generating secrets for eda-airgap-assets
Created eda-airgap-assets/secrets.yaml
generating PKI and tokens
Created eda-airgap-assets/eda-assets.domain.tld.yaml
Created eda-airgap-assets/talosconfig.yaml
Created eda-airgap-assets/rook-ceph-operator-values.yaml
Created eda-airgap-assets/rook-ceph-cluster-values.yaml
```

4.7.3 Deploy the Assets VM

About this task

You can deploy the Assets VM on a KVM or VMware vSphere environment.

4.7.3.1 Creating the VM on a bridged network - air-gapped environment

About this task

Complete the following steps to deploy an Assets VM on KVM.

Note: Execute this procedure on the KVM hypervisor where the Assets VM is hosted.

- Use the Assets VM ISO image generated in Creating the KVM Assets VM image.
- Use the Talos machine config file generated in the Generating the Talos machine config files step for `user-data`.
- Set the root disk is set to 300GB.
- The VM does not require a storage disk

Procedure

Step 1. Ensure that the **virt-install** tool is installed on the KVM hypervisor.

If you need to install the tool, use the following command:

```
yum install virt-install
```

Step 2. Verify that the ISO image downloaded in Downloading the Base Talos VM is available on the hypervisor.

Step 3. Copy the machine configuration file generated for this specific node to a file called `user-data`.

Example

```
cp eda-assets.domain.tld.yaml user-data
```

Step 4. Create a file called `meta-data` for the node.

Use the appropriate `instance-id` and `local-hostname` values.

Example

```
instance-id: eda-assets
local-hostname: eda-assets
```

Step 5. Create a file called `network-config` for the node.

Example

The file should have the following content:

```
version: 2
```

Step 6. Create an ISO file containing the newly created files.

For ease of use, name the ISO file with the name of the node for which you are creating the ISO.

Example

```
mkisofs -o eda-assets -data.iso -V cidata -J -r meta-data network-config user-data
```

Step 7. Create the virtual machine.

This step uses both the newly created ISO file and the ISO file downloaded from the Talos Machine Factory.

Example

```
virt-install -n eda-assets \
  --description "EDA Assets Vm for EDA" \
  --noautoconsole --os-type=generic \
```

```
  --memory 16384 --vcpus 4 --cpu host \
  --disk eda-assets-rootdisk.qcow2,format=qcow2,bus=virtio,size=300 \
  --cdrom eda-asset-vm-nocloud-amd64.iso  \
  --disk eda-assets-data.iso,device=cdrom \
  --network bridge=br0,model=virtio
```

4.7.3.2 Creating the VM on a bridged network on VMware vSphere-air-gapped environment

About this task

Execute this procedure on the air-gapped tools-system.

- Use the Assets VM ISO image generated by in the Creating the VMware Assets VM Image step.
- Use the Talos machine config file generated in the Generating the Talos Machine Configuration Files step for `user-data`.
- You do not need to create a storage disk on the VM.
- After deploying the VM using the OVA image:
  - Increase the number of vCPUs to 4.
  - Increase the memory to 16G.
  - Increase the main disk size to 300G. On boot, Talos automatically extends the file system.
  - Enable 100% resource reservation for the CPU, memory and disk.

Procedure

Step 1. Download and install the latest version of the VMware OVF Tool from the VMware Developer website.

Step 2. Display details about the OVA image.

Example

```
$ ovftool vmware-amd64.ova
OVF version:   1.0
VirtualApp:    false
Name:          talos

Download Size:  104.05 MB

Deployment Sizes:
  Flat disks:   8.00 GB
  Sparse disks: Unknown

Networks:
  Name:        VM Network
  Description: The VM Network network

Virtual Machines:
  Name:               talos
  Operating System:   other3xlinux64guest
  Virtual Hardware:
    Families:         vmx-15
    Number of CPUs:   2
    Cores per socket: automatic
    Memory:           2.00 GB

    Disks:
```

```
      Index:          0
      Instance ID:    4
      Capacity:       8.00 GB
      Disk Types:     SCSI-VirtualSCSI

    NICs:
      Adapter Type:   VmxNet3
      Connection:     VM Network

Properties:
  Key:         talos.config
  Label:       Talos config data
  Type:        string
  Description: Inline Talos config

References:
  File:  disk.vmdk
```

Step 3. Create a base64 encoded hash from the Talos machine configuration for the node.

In this example, the output is stored as an environment variable to make it easy to use in the command to deploy the image using the OVF Tool.

Example

```
export NODECONFIG=$(base64 -i eda-assets.domain.tld.yaml)
```

Step 4. Deploy the OVA image using the OVF Tool.

For details about command line arguments, see the OVF Tool documentation from the VMware website.

Example

```
ovftool --acceptAllEulas --noSSLVerify \
 -dm=thin \
 -ds=DATASTORE \
 -n=eda-assets \
 --net:"VM Network=OAM" \
 --prop:talos.config="${NODECONFIG}" \
eda-asset-vm-vmware-amd64.ova \
vi://administrator%40vsphere.local@vcenter.domain.tld/My-DC/host/My-Cluster/Resources/
My-Resource-Group
```

Expected outcome

This step deploys the VM with the CPU, memory, disk, and NIC configuration of the default OVA image. The next step updates these settings.

Step 5. In vCenter, edit the VM settings.

Make the following changes:

- Increase the number of vCPUs to 4.
- Increase the memory to 16G.
- Increase the main disk size to 300G. On boot, Talos automatically extends the file system.
- Enable 100% resource reservation for the CPU, memory and disk.

Step 6. Power on the virtual machine.

4.8 Bootstrap the Assets VM

This procedure applies to the air-gapped environment and is executed in the air-gapped tools-system

You can bootstrap the Assets VM using the **edaadm** tool, just as you would bootstrap an EDA Kubernetes cluster.

4.8.1 Bootstrapping Kubernetes on the Assets VM

About this task

Use the `edaadm` command with the EDAADM configuration file for the Assets VM to bootstrap Kubernetes:

```
edaadm boostrap-k8s -c eda-assets-deployment.yaml
```

4.8.2 Obtaining the Kubernetes the config file for kubectl

About this task

Use the **edaadm** command to obtain the Kubernetes configuration file for use with **kubectl**. The following parameter is relevant for this procedure:

`-c`: Specifies your EDAADM configuration file.

Procedure

Step 1. Obtain the Kubernetes configuration file.

Execute the following command in the folder where the `eda-assets-deployment.yaml` file is stored.

Example

```
edaadm get-kubeconfig -c eda-assets-deployment.yaml
```

Expected outcome

The command creates a `kubeconfig` file in the folder with the name of your cluster according to the EDAADM config file.

Step 2. Configure the Kubernetes configuration file in our environment.

Example

Use the `kubeconfig` file for use with the **kubectl** command.

```
export KUBECONFIG=eda-airgap-assets/kubeconfig
```

Step 3. Inspect your server and check if all nodes are up and running.

You can use the typical `kubectl` commands.

Example

```
kubectl get nodes
```

Expected outcome

When the node is up and ready, continue with Deploying the Assets VM services.

4.9 Deploying the Assets VM services

About this task

After deploying and bootstrapping the Assets VM, deploy the container registry, git server, and web server.

Note: This procedure applies to the air-gapped environment, and is executed in the air-gapped tools-system

Procedure

Step 1. In the `edaadm` repository that you cloned or downloaded, go to the `kpt` folder.

```
cd path/to/edaadm-repository/kpt
```

Step 2. Deploy the Assets VM services.

Ensure that your `kubeconfig` environment variable points to the kubeconfig of the Assets VM as you got it from the Obtaining the Kubernetes the config file for kubectl section.

```
make eda-setup-shipyard
```

4.10 Uploading assets to the Assets VM

About this task

Note: This procedure applies to the air-gapped environment, and is executed in the air-gapped tools-system.

Procedure

Step 1. In the `edaadm` repository that you cloned or downloaded, go to the `bundles` folder.

```
cd path/to/edaadm-repository/bundles
```

Step 2. Upload the assets.

Make sure your kubeconfig environment variable points to the kubeconfig of the Assets VM as you got it from the Obtaining the Kubernetes the config file for kubectl section.

Replace the setting for `ASSET_HOST` with the IP address of your Asset VM.

```
make load-all-bundles \
  ASSET_HOST=192.0.2.228 \
  ASSET_HOST_GIT_USERNAME="ZWRh" \
  ASSET_HOST_GIT_PASSWORD="ZWRh" \
  ASSET_HOST_ARTIFACTS_USERNAME="ZWRh" \
  ASSET_HOST_ARTIFACTS_PASSWORD="ZWRh"
```

Note: The username and passwords are currently not configurable.

Expected outcome

After all uploads have finished successfully, the Assets VM is ready for use with the installation process of EDA.

4.11 Updating the EDAADM file for the EDA Kubernetes cluster

About this task

Note: This procedure applies to the air-gapped environment, and is executed in the air-gapped tools-system.

Procedure

Add a `mirror` subsection to the EDAADM configuration file. The contents of the subsection should be similar to the following:

```
mirror:
  name: 192.0.2.228
  url: https://192.0.2.228
  insecure: true
  overridePath: false
  skipFallback: true
  mirrors:
    - docker.io
    - gcr.io
    - ghcr.io
    - registry.k8s.io
    - quay.io
```

Note: No proxy configuration should be present in the EDAADM configuration file, as the Assets VM should be directly reachable.

No other changes are needed to run the installation process of the EDA Talos Kubernetes cluster and EDA itself.

4.11.1 Example EDAADM configuration file

Example

```
version: 24.12.1
clusterName: eda-compute-cluster
machines:
    - name: eda-node01
      endpoint: "192.0.2.11"
      interfaces:
        - name: eth0
          dhcp: false
          interface: eth0
```

```
        addresses:
          - 192.0.2.11/24
        routes:
          - network: 0.0.0.0/0
            gateway: 192.0.2.1
        mtu: 9000
      - name: eth1
        dhcp: false
        interface: eth1
        addresses:
          - 203.0.113.11/24
        mtu: 9000
    disks:
      os: /dev/vda
      storage: /dev/vdb
  - name: eda-node02
    endpoint: "192.0.2.12"
    interfaces:
      - name: eth0
        dhcp: false
        interface: eth0
        addresses:
          - 192.0.2.12/24
        routes:
          - network: 0.0.0.0/0
            gateway: 192.0.2.1
        mtu: 9000
      - name: eth1
        dhcp: false
        interface: eth1
        addresses:
          - 203.0.113.12/24
        mtu: 9000
    disks:
      os: /dev/vda
      storage: /dev/vdb
  - name: eda-node03
    endpoint: "192.0.2.13"
    interfaces:
      - name: eth0
        dhcp: false
        interface: eth0
        addresses:
          - 192.0.2.13/24
        routes:
          - network: 0.0.0.0/0
            gateway: 192.0.2.1
        mtu: 9000
      - name: eth1
        dhcp: false
        interface: eth1
        addresses:
          - 203.0.113.13/24
        mtu: 9000
    disks:
      os: /dev/vda
      storage: /dev/vdb
  - name: eda-node04
    endpoint: "192.0.2.14"
    interfaces:
      - name: eth0
        dhcp: false
        interface: eth0
        addresses:
```

```
            - 192.0.2.14/24
          routes:
            - network: 0.0.0.0/0
              gateway: 192.0.2.1
          mtu: 9000
        - name: eth1
          dhcp: false
          interface: eth1
          addresses:
            - 203.0.113.14/24
          mtu: 9000
      disks:
        os: /dev/vda
    - name: eda-node05
      endpoint: "192.0.2.15"
      interfaces:
        - name: eth0
          dhcp: false
          interface: eth0
          addresses:
            - 192.0.2.15/24
          routes:
            - network: 0.0.0.0/0
              gateway: 192.0.2.1
          mtu: 9000
        - name: eth1
          dhcp: false
          interface: eth1
          addresses:
            - 203.0.113.15/24
          mtu: 9000
      disks:
        os: /dev/vda
    - name: eda-node06
      endpoint: "192.0.2.16"
      interfaces:
        - name: eth0
          dhcp: false
          interface: eth0
          addresses:
            - 192.0.2.16/24
          routes:
            - network: 0.0.0.0/0
              gateway: 192.0.2.1
          mtu: 9000
        - name: eth1
          dhcp: false
          interface: eth1
          addresses:
            - 203.0.113.16/24
          mtu: 9000
      disks:
        os: /dev/vda
k8s:
    stack: ipv4
    primaryNode: eda-node01
    endpointUrl: https://192.0.2.5:6443
    allowSchedulingOnControlPlanes : true
    control-plane:
        - eda-node01
        - eda-node02
        - eda-node03
    worker:
        - eda-node04
```

```
        - eda-node05
        - eda-node06
    vip:
        ipv4: 192.0.2.5
        interface: eth0
    time:
        disabled: false
        servers:
            - 192.0.2.253
            - 192.0.2.254
    nameservers:
        servers:
            - 192.0.2.253
            - 192.0.2.254
    mirror:
      name: 192.0.2.228
      url: https://192.0.2.228
      insecure: true
      overridePath: false
      skipFallback: true
      mirrors:
        - docker.io
        - gcr.io
        - ghcr.io
        - registry.k8s.io
        - quay.io
```

4.12 Bootstrapping the EDA Talos - air-gapped environment

This section applies to the air-gapped environment, and is executed in the air-gapped tools-system.

Complete the following tasks:

- Bootstrapping Kubernetes on the primary node
- Obtaining the Kubernetes config file for kubectl
- Setting up the Rook Ceph for air-gapped environments

4.12.1 Setting up the Rook Ceph for air-gapped environments

About this task

Note: This procedure applies to the air-gapped environment, and is executed in the air-gapped tools-system.

Ensure that you are using the correct paths and the correct Assets VM IP address in the commands below.

Procedure

Step 1. Deploy the Rook Ceph operator.

Using the `rook-ceph-operator-values.yaml` file that **edaadm** tool generated based on the configuration, deploy the Rook Ceph Operator using the Rook Ceph charts present on the Assets VM.

```
helm install --create-namespace \
  --namespace rook-ceph \
```

```
  --version v1.15.0 \
  -f path/to/rook-ceph-operator-values.yaml \
  rook-ceph \
  http://eda:eda@<ASSETS VM IP>/artifacts/rook-ceph-v1.15.0.tgz
```

Step 2. Deploy the Rook Ceph cluster.

Using the `rook-ceph-cluster-values.yaml` file that the **edaadm** tool generated, deploy the Rook Ceph cluster.

```
helm install \
  --namespace rook-ceph \
  --set operatorNamespace=rook-ceph \
  -f path/to/rook-ceph-cluster-values.yaml \
  rook-ceph-cluster \
  http://eda:eda@<ASSETS VM IP>/artifacts/rook-ceph-cluster-v1.15.0.tgz
```

Step 3. Using **kubectl** commands, verify that the operator is deployed and the necessary pods are deployed before installing the EDA application.

The following sample output is for a deployment with 6 nodes used for storage:

```
$ kubectl -n rook-ceph get pods
NAME                                                READY   STATUS      RESTARTS
 AGE
csi-cephfsplugin-22rmj                              2/2     Running     1 (6m32s ago)
 7m6s
csi-cephfsplugin-25p9d                              2/2     Running     1 (6m30s ago)
 7m6s
csi-cephfsplugin-2gr8v                              2/2     Running     4 (5m16s ago)
 7m6s
csi-cephfsplugin-48cwk                              2/2     Running     1 (6m30s ago)
 7m6s
csi-cephfsplugin-fknch                              2/2     Running     2 (5m32s ago)
 7m6s
csi-cephfsplugin-provisioner-67c8454ddd-mpq4w       5/5     Running     1 (6m1s ago)
 7m6s
csi-cephfsplugin-provisioner-67c8454ddd-qmdrq       5/5     Running     1 (6m18s ago)
 7m6s
csi-cephfsplugin-vfxnf                              2/2     Running     1 (6m32s ago)
 7m6s
rook-ceph-mds-ceph-filesystem-a-7c54cdf5bc-lmf6n    1/1     Running     0
 2m40s
rook-ceph-mds-ceph-filesystem-b-6dc794b9f4-2lc64    1/1     Running     0
 2m37s
rook-ceph-mgr-a-55b449c844-wpps8                    2/2     Running     0
 4m30s
rook-ceph-mgr-b-5f97fd5746-fzngx                    2/2     Running     0
 4m30s
rook-ceph-mon-a-76fcb96c4c-vscnc                    1/1     Running     0
 5m53s
rook-ceph-mon-b-68bf5974bb-p2vnj                    1/1     Running     0
 4m57s
rook-ceph-mon-c-6d7c64dcb6-phs99                    1/1     Running     0
 4m47s
rook-ceph-operator-5f4c4bff8d-2fsq2                 1/1     Running     0
 7m54s
rook-ceph-osd-0-bf89f779-zh4kd                      1/1     Running     0
 3m49s
rook-ceph-osd-1-64dcd64c5f-7xcbm                    1/1     Running     0
 3m49s
rook-ceph-osd-2-54ddd95489-5qkdt                    1/1     Running     0
 3m49s
```

```
rook-ceph-osd-3-56cbd54bd6-7mt8w                  1/1     Running     0
 3m39s
rook-ceph-osd-4-567dcff476-wljll                  1/1     Running     0
 2m56s
rook-ceph-osd-5-6f69c998b6-2l5wp                  1/1     Running     0
 2m54s
rook-ceph-osd-prepare-eda-dev-node01-7rfkn        0/1     Completed   0
 4m8s
rook-ceph-osd-prepare-eda-dev-node02-rqdkx        0/1     Completed   0
 4m8s
rook-ceph-osd-prepare-eda-dev-node03-xtznb        0/1     Completed   0
 4m8s
rook-ceph-osd-prepare-eda-dev-node04-db4v8        0/1     Completed   0
 4m7s
rook-ceph-osd-prepare-eda-dev-node05-29wwm        0/1     Completed   0
 4m7s
rook-ceph-osd-prepare-eda-dev-node06-zxp2x        0/1     Completed   0
 4m7s
rook-ceph-tools-b9d78b5d4-8r62p                   1/1     Running     0
 7m6s
```

Note: Some of the pods may restart as they initiate Ceph. This behavior is expected.

4.13 Installing the EDA application in an air-gapped environment

The procedures in this section apply to the air-gapped environment and are executed in the air-gapped tools-system.

Before you install the EDA application, customize the `prefs.mk` file to use the Assets VM instead of the Internet. Then, you can install the EDA application.

4.13.1 Customizing the `prefs.mk` file

Procedure

Step 1. Remove any proxy configuration from the `prefs.mk` file.

The EDA Kubernetes cluster must be able to reach the Assets VM directly.

Step 2. Add the following settings to the `prefs.mk` file.

```
USE_ASSET_HOST=1
ASSET_HOST=192.0.2.228
ASSET_HOST_GIT_USERNAME="eda"
ASSET_HOST_GIT_PASSWORD="eda"
ASSET_HOST_ARTIFACTS_USERNAME="eda"
ASSET_HOST_ARTIFACTS_PASSWORD="eda"
```

where the ASSET_HOST setting points to the IP address of the Assets VM.

Example: Updated `prefs.mk` file.

```
NO_KIND=1
USE_ASSET_HOST=1
ASSET_HOST=192.0.2.228
```

```
ASSET_HOST_GIT_USERNAME="eda"
ASSET_HOST_GIT_PASSWORD="eda"
ASSET_HOST_ARTIFACTS_USERNAME="eda"
ASSET_HOST_ARTIFACTS_PASSWORD="eda"
METALLB_VIP=203.0.113.10/32
EXT_DOMAIN_NAME=eda.domain.tld
EXT_HTTP_PORT=80
EXT_HTTPS_PORT=443
EXT_IPV4_ADDR=203.0.113.10
EXT_IPV6_ADDR=""
LLM_API_KEY=...
```

Related topics

Customizing the installation file

4.13.2 Installing the Nokia EDA application

Procedure

Step 1. Set up the MetalLB environment for VIP management.

Example

```
make metallb
```

Step 2. Install the necessary external packages.

Example

```
make install-external-packages
```

Note: If this command exits with an error, wait 30 seconds and try again. Sometimes Kubernetes is a bit slower in reconciling the change than the command waits for.

Step 3. Change the eda-git Kubernetes service to a ClusterIP service instead of a LoadBalancer type.

Example

```
kubectl -n eda-system patch service eda-git -p '{"spec": {"type": "ClusterIP"}}'
```

Step 4. Generate the EDA core configuration.

Example

```
make eda-configure-core
```

Step 5. Install EDA core components.

Example

```
make eda-install-core
```

Note: If the command hangs for a long longer than 5 minutes on "reconcile pending" for a workflow definition, cancel the command and try again. KPT is designed to handle these cases. This can occasionally happen depending on the Kubernetes cluster

Step 6. Verify that the EDA config engine is up and running.

Example

```
make eda-is-core-ready
```

Step 7. Install all the standard EDA apps.

This step can take approximate 5 to 15 minutes, depending on your connectivity.

Example

```
make eda-install-apps
```

Step 8. Bootstrap EDA.

Example

```
make eda-bootstrap
```

Step 9. If your deployment uses two networks, create a second VIP pool for the OAM VIP address.

Example

```
make metallb-configure-pools METALLB_VIP=<OAM VIP> LB_POOL_NAME=pool-nb
```

Step 10. If your deployment uses two networks, create the OAM UI/API service using the new VIP pool.

Example

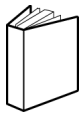
```
make eda-create-api-lb-svc API_LB_POOL_NAME=pool-nb
```

Step 11. Optional: Deploy an example topology.

Example

```
make topology-load
```

Customer document and product support



Customer documentation

Customer documentation welcome page

Technical support

Product support portal

Documentation feedback

Customer documentation feedback